CRANFIELD UNIVERSITY

Ivan Vitanov

# Kernel-based Fault Diagnosis of Inertial Sensors using Analytical Redundancy

Cranfield Defence and Security (CDS)

PhD Thesis
Academic Year: 2015 - 2016

Supervisor: Dr Nabil Aouf
December 2015

CRANFIELD UNIVERSITY

Cranfield Defence and Security (CDS)

PhD Thesis

Academic Year 2015 - 2016

Ivan Vitanov

# Kernel-based Fault Diagnosis of Inertial Sensors using Analytical Redundancy

Supervisor: Dr Nabil Aouf
December 2015

This thesis is submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

# ABSTRACT

Kernel methods are able to exploit high-dimensional spaces for representational advantage, while only operating implicitly in such spaces, thus incurring none of the computational cost of doing so. They appear to have the potential to advance the state of the art in control and signal processing applications and are increasingly seeing adoption across these domains.

Applications of kernel methods to fault detection and isolation (FDI) have been reported, but few in aerospace research, though they offer a promising way to perform or enhance fault detection. It is mostly in process monitoring, in the chemical processing industry for example, that these techniques have found broader application.

This research work explores the use of kernel-based solutions in model-based fault diagnosis for aerospace systems. Specifically, it investigates the application of these techniques to the detection and isolation of IMU/INS sensor faults – a canonical open problem in the aerospace field.

Kernel PCA, a kernelised non-linear extension of the well-known principal component analysis (PCA) algorithm, is implemented to tackle IMU fault monitoring. An isolation scheme is extrapolated based on the strong duality known to exist between probably the most widely practiced method of FDI in the aerospace domain – the parity space technique – and linear principal component analysis. The algorithm, termed partial kernel PCA, benefits from the isolation properties of the parity space method as well as the non-linear approximation ability of kernel PCA.

Further, a number of unscented non-linear filters for FDI are implemented, equipped with data-driven transition models based on Gaussian processes - a non-parametric Bayesian kernel method. A distributed estimation architecture is proposed, which besides fault diagnosis can contemporaneously perform sensor fusion. It also allows for decoupling faulty sensors from the navigation solution.

# ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr Nabil Aouf for his continual supervision, guidance and support.

I would also like to thank Professor Mark Richardson for his patience, support and understanding during this research.

Major thanks go to Major Mohammed Boulekchour for his invaluable help with various diagrams and LaTeX, which I sadly never managed to put to good use, and to Captain Oualid Araar for his equally invaluable help with simulation modelling.

Thanks to Krasin Georgiev for being a reliable drinking buddy and for all the interesting chats. Thanks go out also to my former housemate Abdenour A. and office mates Abdenour K. and Özgün. Thanks also to Colonel Ahmed Ayad for being a good sport.

I would like to extend my thanks to Professor Ron Patton for the useful talks and advice at several conferences.

Last, but by no means least, I would like to thank my mother for her tireless and unceassing love and support.

I should also like to thank BAE Systems for kindly sponsoring this research.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF EQUATIONS

# LIST OF ABBREVIATIONS

| | |
|---|---|
| PCA | Principal component analysis |
| kPCA | Kernel principal component analysis |
| RBF | Radial basis function |
| GOS | Generalised observer scheme |
| DOS | Dedicated observer scheme |
| FDI | Fault detection and isolation |
| FDIR | Fault detection, isolation and reconfiguration |
| FDD | Fault detection and diagnosis |
| GP | Gaussian process |
| KF | Kalman filter |
| EKF | Extended Kalman filter |
| UKF | Unscented Kalman filter |
| EHF | Extended H-infinity filter |
| UHF | Unscented H-infinity filter |
| UIF | Unscented information filter |
| CIF | Centralised information filter |
| FIF | Federated information filter |
| GP-EKF | Gaussian process extend Kalman filter |
| GP-UKF | Gaussian process unscented Kalman filter |
| GP-UHF | Gaussian process unscented H-Infinity filter |
| GP-CUIF | Gaussian process centralised unscented information filter |
| GP-FUIF | Gaussian process federated unscented information filter |
| INS | Inertial navigation system |
| IMU | Inertial measurement unit |
| MEMS | Micro electro-mechanical system |
| UAV | Unmanned aerial vehicle |
| SE | Squared exponential |

# 1 INTRODUCTION

## 1.1 Background

Highly complex, safety critical systems such as unmanned aerial vehicles (UAVs) require real-time health monitoring to ensure system integrity is not compromised in the event of a fault. If left unchecked or undetected, faults can degrade the performance of an autonomous vehicle and may lead to system failure, resulting in loss of physical assets, aborted missions and collateral damage. A US defence study (Schaefer 2003) looking into sources of system failures in the U.S. military UAV fleet (based on 100,000 flight hours) found that on average 83% of incidents were caused by faults affecting sensors, actuators and electro-mechanical processes; the remaining 17% being down to human error. This is in contrast to manned aircraft, where around 85% of failures are accounted for by human error. Thus, the dependability of UAVs can be greatly enhanced by fault diagnosis provision.

*Fault diagnosis* is typically part of a larger process known under various acronyms: FDD or *fault detection and diagnosis*; FDI or *fault detection and isolation*; FDIR or *fault detection, identification and reconfiguration*. The following nomenclature, initially proposed by the IFAC SAFEPROCESS Technical Committee, is now standard in FDIR (Marzat et al. 2012).

A *fault* is an unpermitted deviation of at least one characteristic property or parameter of the system from acceptable, usual or standard conditions. A fault may lead to a *failure*, which is a permanent interruption of the system ability to perform a required function under specified operating conditions. *Fault detection* is the determination of the presence of faults in a system and of their times of occurrence. It is generally followed by *fault isolation* to determine the type and location of the faults. *Fault identification* (or *estimation*) tries to determine the size and time-varying behaviour of the faults. Fault isolation and identification are together said to make up fault diagnosis. A natural extension of fault diagnosis is to try to compensate for any existing faults by modifying the control

law of a flight vehicle. This is the province of *fault tolerant control* (FTC), or *reconfiguration*.

## 1.2 UAV Navigation and Fault Monitoring

UAVs require accurate location information for a variety of tasks. This naturally includes navigation, but also motion planning and control. INS-based localisation in UAVs is generally realised through the use of some sort of navigation filter. With increased autonomy, and with humans increasingly out of the (control) loop, the need for guidance and navigation systems that meet the highest standards of integrity and robustness is paramount (Construit et al. 2009).

Inertial navigation in three dimensions necessitates two sensor triads consisting of 3-axis gyroscopes and accelerometers to measure, respectively, the attitude angles and accelerations along the three coordinate axes. Navigation systems are prone to the occurrence of faults in the normal course of operation; something that can impact the quality of the navigation solution. This degrades the navigation information coming in and hinders reliable localisation of the vehicle in space (Groves 2008). FDI can be used to counteract this type of problem. In this study, we focus on model-based diagnosis as a measure to prevent navigation system failure or decline.

Lately, low-cost, small-size micro-electro-mechanical system (MEMS) inertial sensors have been implemented in devices such as quadrotor UAVs. Aside from the obvious advantages of reduced size and cost, MEMS inertial sensors are characterised by high levels of noise and output uncertainties, such as biases/scale factors (Noureldin et al. 2009). Therefore, a MEMS INS incurs errors in position, velocity and attitude. Such errors can accumulate rapidly, degrading the accuracy of the navigation solution within a short time period. Therefore, bias modelling of low-cost inertial sensors is a prerequisite for improving their performance and amenability to fault detection.

A sensor fault can be defined as an unexpected change in a sensor signal due to degradation or damage to a sensing instrument resulting in a corrupted

output. The fault detection and isolation (FDI) problem resolves to a binary decision at a particular time instant: either there is sufficient evidence that a fault is in progress or there is not. When a fault is determined to have occurred, the task of fault isolation is to pinpoint its location (Chow & Willsky 1984).

Generally speaking, FDI techniques rely on the concept of redundancy: hardware or analytical. Hardware redundancy involves comparing replica of the same signal generated by various hardware components, such as two or more sensors measuring the same quantity. Analytical redundancy, on the other hand, replaces redundant hardware implementation with a mathematical model. The two approaches are contrasted diagrammatically in Figure 1.1.

The analytical approach has the advantage that it is more lightweight and cost-effective as additional hardware is not required (Hwang et al. 2010). This is an important consideration in UAV operation, since pilotless air vehicles are typically more lightweight and streamlined than piloted aircraft, and vehicle weight and payload should preferably be kept to a minimum. The analytical redundancy approach is, however, more of a challenge to implement, as it has to deal robustly with model inaccuracies (since no model is exact, this is critical), the presence of noise and unknown disturbances. Diagnosis often consists of generation of residuals (fault indicators based on differences between measurements and model-based predictions) followed by their evaluation within decision functions. A fault can be detected and localised when it causes a particular residual to increase in magnitude above a certain threshold.

For aerospace systems, there exists a widening gulf between the FDIR solutions proposed by academic researchers and the technical solutions being adopted by aerospace industry end-users (Zolghadri 2011). This is arguably due to aerospace professionals putting a premium on fail-safe solutions that militate against even the slightest risk of operational failure. This conservative tendency leads to overreliance on multiple tiers of hardware redundancy. However, as mentioned in the previous paragraph, increased payload is not an option for small UAVs, necessitating more robust analytical solutions.

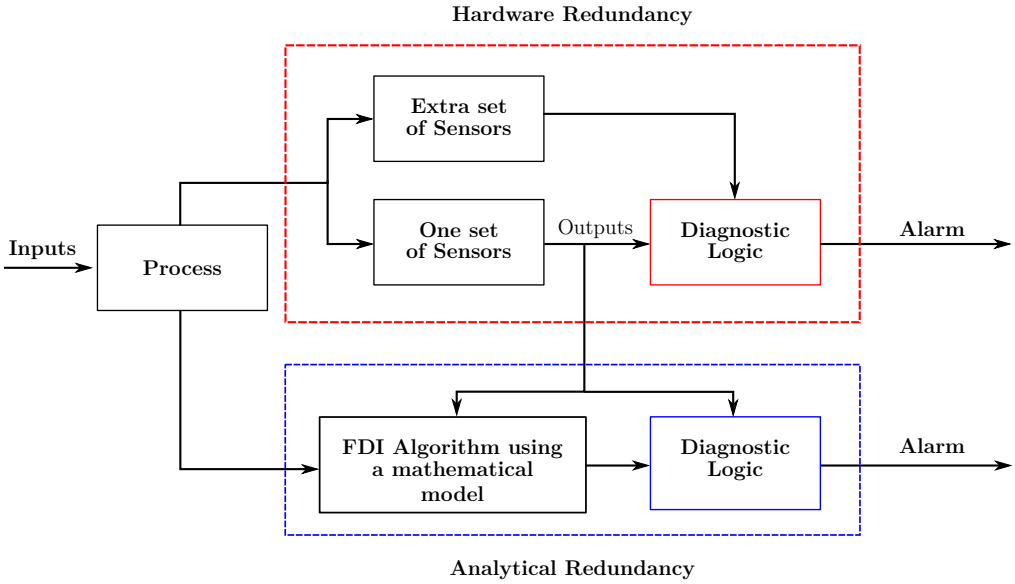



**Figure 1-1 Analytical and hardware redundancy**

From the foregoing it is seen that a robust diagnosis procedure to address the fault detection and isolation problem is an essential aspect in improving the performance of localisation and navigation sensors. With ever-increasing levels of autonomy for unmanned vehicles and, by extension, humans increasingly out of the (control) loop, the need for guidance and navigation systems that meet the highest standards of integrity and robustness is paramount. To mitigate the adverse effects of biases and drift errors inherent to inertial sensors, fusion with other navigation sensors, such as a global positioning system (GPS) or a magnetometer, is commonly employed to provide enhanced state estimation. The resultant hybrid sensor system is known as an integrated navigation system. There are, however, environments (e.g., indoor environments) where outages of the coupled instruments can mean the INS operating stand-alone. Thus, ensuring fault-free and stable INS operation is crucial to safe and reliable adherence to a designated flight trajectory.

Figures 1.2-1.4 display images of popular micro-UAV models. The advent of cheap and lightweight UAVs has proved invaluable in enabling in situ academic research into autonomous aerial platform technology, and has had the concomitant effect of providing a spur to the application of FDI to such systems.

**Figure 1-2 Pelican drone**


**Figure 1-3 Asctec FireFly**

## 1.3 Scope and Research Aims

From the foregoing we can define the principal research aims and scope of the research as follows.

The work described in this thesis aims to achieve robust & accurate (low false alarm and true detection rate) fault diagnosis of UAV inertial navigation systems so as to ensure reliable vehicle localisation. It aims to develop novel model-based fault diagnostic techniques exploiting analytical redundancy.

Concerning the novelty aspect, the intent is to merge well-established classical methodologies within the purview of aerospace FDI, such as the parity space technique and diagnostic observers, with more contemporary techniques arising out of the fields of machine learning and pattern recognition that afford non-linear extensions to linear systems. Techniques utilising kernels and the sigma point transform are felt to hold particular promise in this regard. Thus, the focus of the thesis is on updating well-tried concepts and methods within aerospace FDI via emergent paradigms in the rapidly developing data sciences. In this vein, there is the potential for crossover from the fields of anomaly and outlier detection, which are essentially application areas of machine learning and data mining that deal with the same fundamental problem as fault detection.

**Figure 1-4 Parrot drone**

## 1.4 Organisation of the Thesis

Below are listed the publications that relate to the contributions presented in this thesis. The particular thesis chapters they have relevance for are labelled in Figure 1.5, which illustrates the organisation of the thesis.

- Fault detection and isolation in inertial navigation systems with SDRE non-linear filter (Vitanov & Aouf 2013).

*5th European Conference for Aeronautics and Space Sciences (EUCASS).*

- Fault diagnosis for MEMS INS using unscented Kalman filter enhanced by Gaussian process adaptation (I Vitanov & Aouf 2014).

*2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS).*

- Fault detection and isolation in an inertial navigation system using a bank of unscented H∞ filters (Ivan Vitanov & Aouf 2014a).

*2014 UKACC International Conference on Control.*

- Fault diagnosis and recovery in MEMS inertial navigation system using information filters and Gaussian processes (Ivan Vitanov & Aouf 2014b).

*22nd Mediterranean Conference of Control and Automation (MED)*

- Kernel PCA applied to fault diagnosis of MEMS IMU (in preparation; provisional title)

*IMechE Journal of Aerospace Engineering*

The thesis is divided into 6 chapters; the dependencies between them are shown in Figure 1.5. The main contributions and technical results are discussed in Chapters 3-5. The chapters would ideally be read chronologically to get a more complete understanding of the work. Chapter 3 is not a prerequisite for understanding later chapters, except for Section 3.4, which covers certain implementation details that recur in Chapters 4-5. The remainder of Chapter 3 can be omitted without too much loss of generality. Below is an outline of the rest of the thesis.

**Chapter 2** overviews multi-sensor navigation systems and their properties. As already discussed, this is the target system which forms the test bed for the algorithms presented in this thesis. The chapter lays out the theoretical underpinnings of such systems and reviews the literature on the FDI methods most commonly applied to them.

In **Chapter 3** a detection and isolation algorithm is developed for IMU sensor faults by extending a kernel-based non-linear version of principal component analysis. The duality known to exist between linear PCA and the parity space technique widely adopted in the aerospace industry is exploited to modify the scope of the kernel PCA algorithm to include isolation as well as detection, where traditionally it has been limited to the latter. The algorithm formed from this conjunction, kernel partial PCA, has parity space-like isolation structures.

**Chapter 4** introduces the Bayesian non-parametric regression technique of Gaussian processes and its application to FDI. This technique has been recently combined with non-linear Kalman filters to produce highly flexible filtering solutions. We seek to leverage the state space formalism of Kalman-type observers and the modelling flexibility of Gaussian process towards achieving improved model-based analytical redundancy.

**Chapter 5** extend the Gaussian process methods encountered in Chapter 4 to distributed (information) filters, which possess certain advantages, such as the ability to perform sensor fusion and FDI side-by-side.

**Chapter 6** concludes the thesis and proposes ideas for future work.

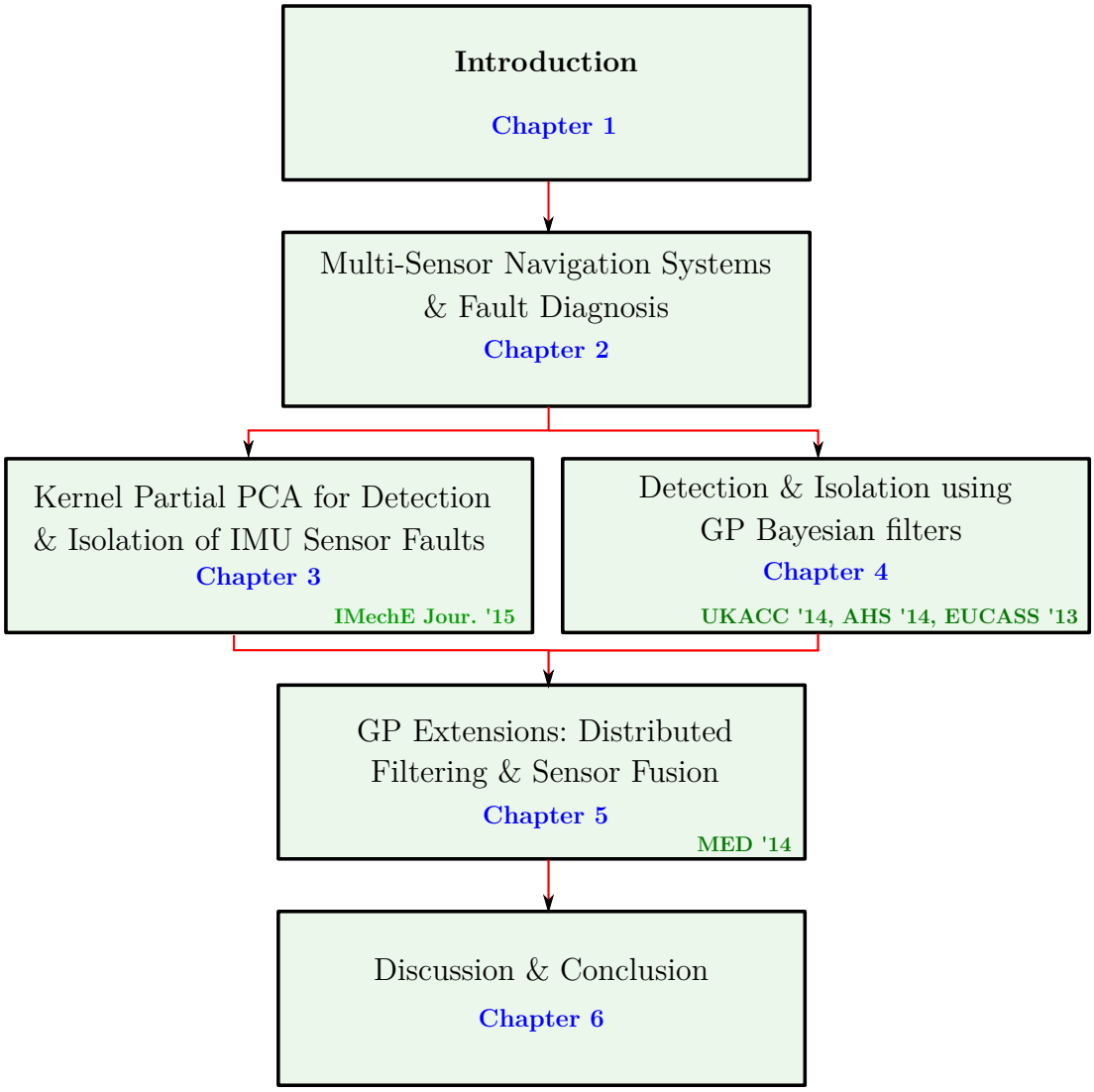



**Figure 1-5 Thesis structure**

# 2 MULTI-SENSOR NAVIGATION SYSTEMS AND FAULT DIAGNOSIS

## 2.1 Inertial Navigation Systems (INS)

UAVs rely on motion sensors to compute a real-time state represented by position, velocity and orientation of the vehicle. An inertial navigation system is the most commonly used navigation aid, providing UAV position, velocity and attitude information. It can be used as a stand-alone unit providing navigation information or in conjunction with a control system for autonomous movement.

In recent decades, INS development has advanced rapidly, which has led to this sensor modality being used more and more in military and commercial projects as the technology has matured.

The core of the INS is the inertial measurement unit (IMU), where angular rates and accelerations are measured in the vehicle body frame. Values of position, velocity and attitude of the vehicle are updated in the appropriate coordinate frame (Woodman 2007).

A number of reference frames exist; the following are the most common.

First, the *inertial frame* is a motionless frame in terms of acceleration and rotation, oriented with respect to fixed stars and the earth's centre of mass as its origin. The Earth-centred inertial frame (ECI) is not strictly inertial, because its origin is in the Earth's revolution around the Sun; however, this frame is considered inertial for navigation purposes. The Earth-centred Earth-fixed frame (ECEF) and the north-east-down frame (NED) are other common frames. ECI, ECEF and NED frames are shown in Figure 2.1 (Schumacher 2006).

The body frame is an orthogonal frame, whose axes are coincident with the axes of the IMU. In the gimballed INS, the body and navigation frames must be kept aligned, using the gyro information and external torques. In the strap-down arrangement, the IMU is rigidly mounted on the moving object to be positioned and conventionally is considered aligned with it (Noureldin et al. 2009).

A traditional INS consists of an accelerometer mounted on each of the three orthogonal axes of a stable platform. This would be aligned with the Earth local frame by ensuring that the acceleration due to gravity is detected by the vertical accelerometer and would be aligned to true north using a magnetic compass. The gimballed arrangement of the platform would ideally have three degrees of freedom, although practical systems are often limited by the manoeuvrability of the aircraft using the system.

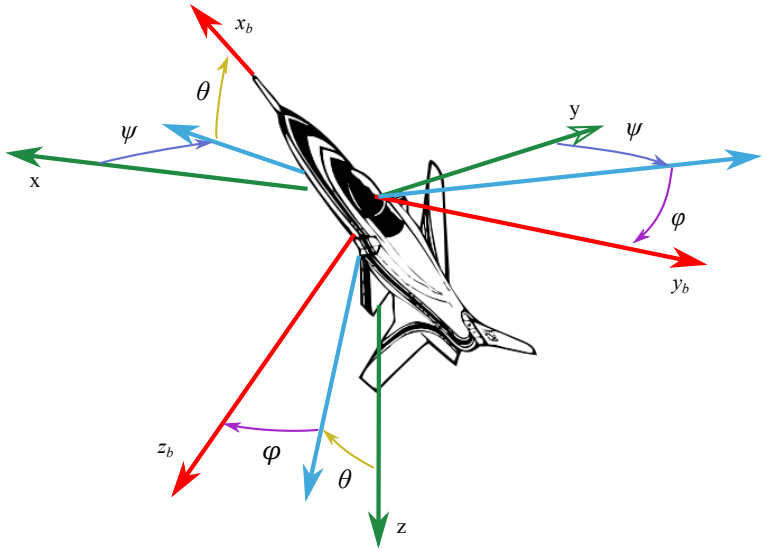



**Figure 2-1 Inertial, navigation and earth frames**

When in motion, gyroscopes maintain the platform level with the earth. They also, when spinning at high resolution per minute (rpm) rates, attempt to remain aligned to a given point in space, and move in a plane 90 degrees from the direction of rotation when receiving a rotational input, known as precession. The gyroscopes create a force proportional to the rotation rate which is usable as feedback to maintain the level attitude pointing north of the platform. The vehicle attitude can be read using the angles subtended between the gimbals and the platform (Groves 2008; Chatfield 1997). This is equivalent to integrating the rotation rates with respect to time. However, gyroscopes are subject to drift, which leads to time-related errors, due to tolerances within the components. This is equivalent to integrating the rotation rates with respect to time.

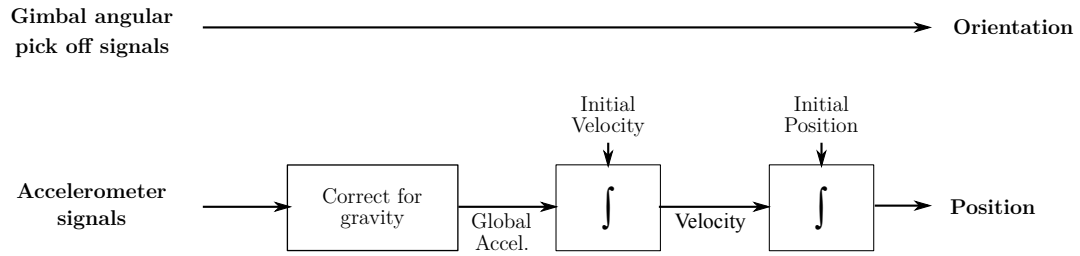



**Figure 2-2 Gimballed inertial navigation algorithm**

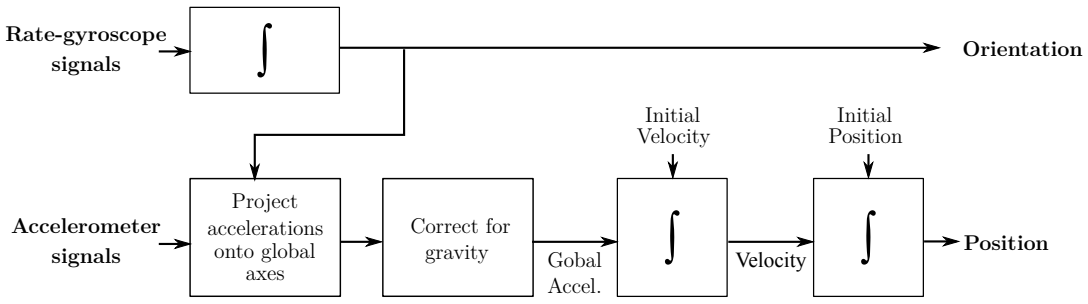



**Figure 2-3 Strap-down inertial navigation algorithm**

However, gyroscopes are subject to drift, which leads to time-related errors, due to tolerances within the components. Also the Earth's translation and rotation through time need to be considered. Corrections can be applied for this drift through calibration and using Earth-motion algorithms (Lopes 2011).

The outputs from the accelerometers mounted on the stable platform are integrated with respect to time to give vehicle velocity, and integrated again to give position (Groves 2008). The traditional INS process flow is depicted in Figure 2.4 with a 6 degree of freedom input of 3 translations and 3 rotations. Figures 2.2-2.3 provide more details for gimballed and strap-down INS (Chatfield 1997).

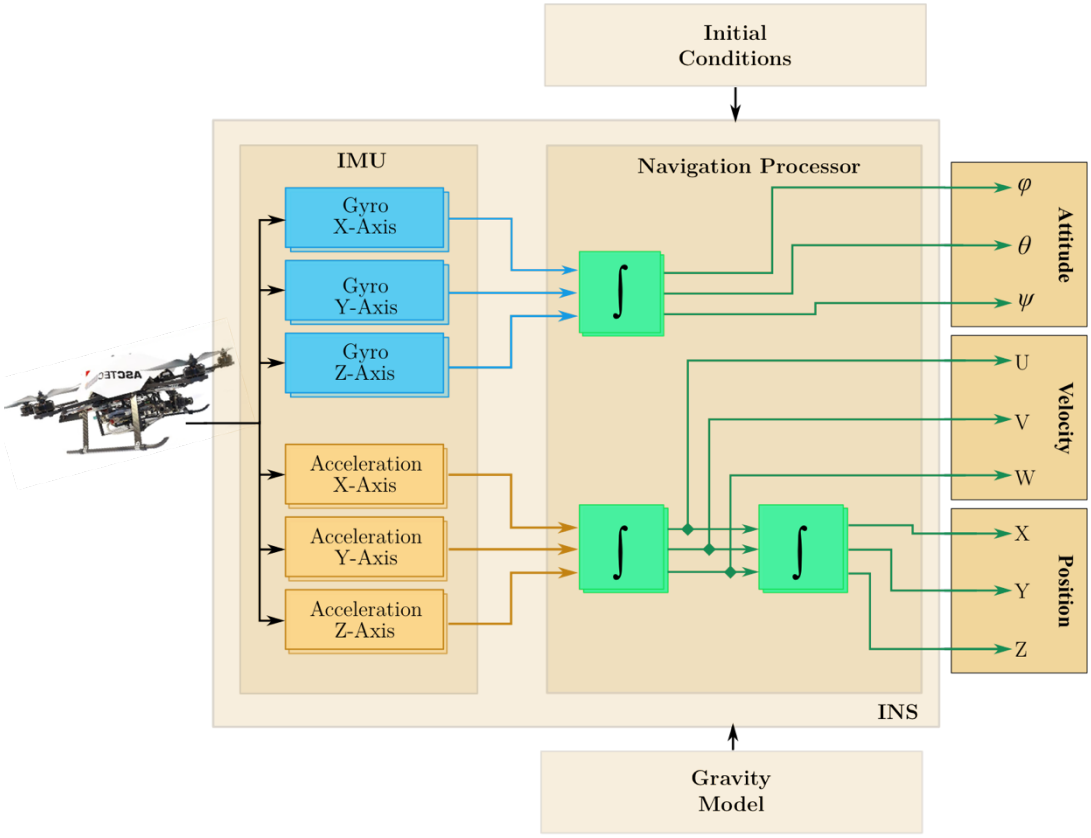



**Figure 2-4: INS architecture**

### 2.1.1 Strap-down INS

Strap-down inertial navigation enables navigation without the use of a mechanically stabilised platform in determining the vehicle's location. This has been achieved through the development of gyros and rate sensors technology, and the emergence of high speed microprocessors capable of maintaining a stable platform using digital means rather than mechanically. The required inertial reference outputs are provided by an inertial reference system for the aircraft avionics. The principal source of error causing drift is imperfections of gyro bearings and mass imbalances.

The development of an INS with fewer mechanical parts was motivated by greater accuracy and reliability of the device. Rotational gyros have given way to more sophisticated devices, such as fibre-optic and the ring laser gyros used in higher-end applications. The largest gain in regard to reducing complexity was achieved by mounting the INS components directly onto the vehicle itself, without a stable level platform, termed a strap-down INS. However, a component of acceleration due to gravity can be detected and this must be taken into account in the processing algorithms.

### 2.1.2 MEMS INS

The main factor in the cost of the INS is the design of the IMU. Recently, proliferation of micro-electro-mechanical-systems (MEMS) sensors has enabled the design of low-cost IMUs with a trade-off in accuracy (Jia 2004).

In order to cover six degrees of freedom, these IMUs generally require three accelerometers mounted orthogonally and three gyroscopes around the same orthogonal axes.

The manufacturing of these devices is based on integrated circuit technology with a mechanical element for sensing. Therefore, they are used where small size and low power are required (Tedaldi 2013).

Figure 2.5 depicts a cross-section from a MEMS IMU. Figure 2.6 depicts a gimballed IMU.

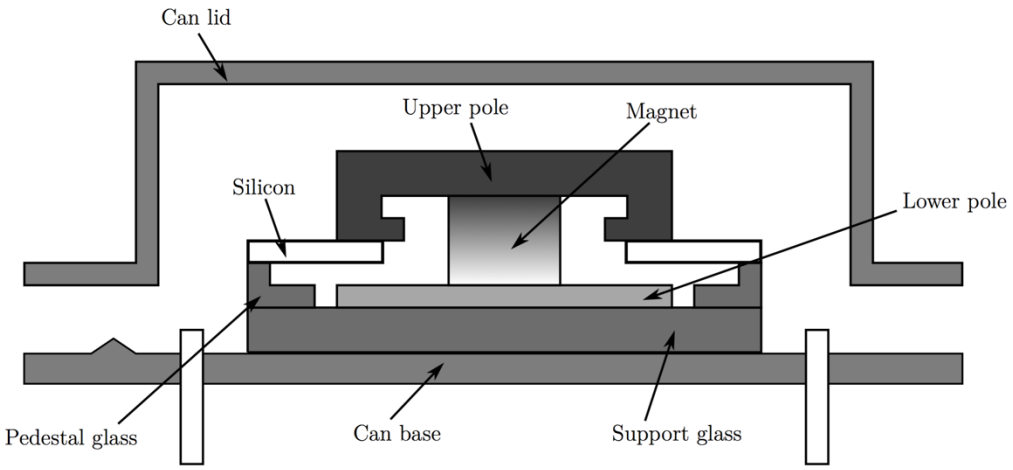



**Figure 2-5: MEMS IMU**

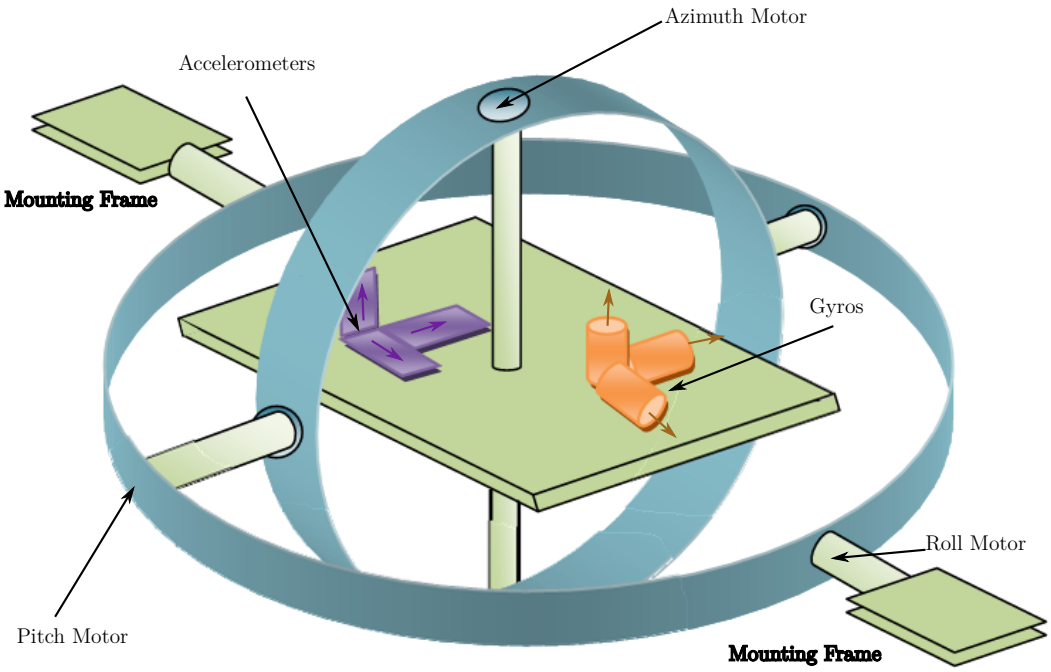



**Figure 2-6:Gimballed IMU**

### 2.1.3 INS analytical model

Inertial navigation is achieved by integrating the output of a set of sensors to compute position, velocity and attitude. The sensors used are a set of three gyroscopes to measure roll, pitch and yaw rotation rates (*p*, *q*, *r*); as well as three accelerometers measuring linear accelerations (*ax*, *ay*, *az*) along the three body axes, with respect to an inertial frame. Collectively the three accelerometers and three gyros make up the core sensing device of the INS, the inertial measuring unit (IMU).

The measurements from the IMU are processed through a series of integrations and transformed into an appropriate navigation frame - such as an earth-centred earth-fixed frame (ECEF) - yielding aerial position coordinates (*X*, *Y*, *Z*), velocities (*U*, *V*, *W*) and attitude Euler angles ($\varphi$, $\theta$, $\psi$).

The INS can be represented in the following continuous-valued non-linear state space form:

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x,u) \end{aligned} \tag{2-1}$$

With *x* the state vector, and *u* the input vector to the INS (angular rates and accelerations) – alternatively this is the output vector of the IMU. That is:

$$x = [X,\ Y, Z, U, V, W, \varphi, \theta, \psi]^T \tag{2-2}$$

$$u = [p, q, r, ax, ay, az]^T \tag{2-3}$$

The navigation equations require that we define at minimum two reference frames. One is a vehicle coordinate frame (body or inertial); the other is a navigation frame. System equations of motion can then be derived through basic integrations and frame transformations.

By integrating the following equation we can evaluate the Euler angles ($\varphi$, $\theta$, $\psi$):

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\varphi)\tan(\theta) & \cos(\varphi)\tan(\theta) \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi)\sec(\theta) & \cos(\varphi)\sec(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad \textbf{(2-4)}$$

We now have the attitude of the aircraft. Using the orientation values and the outputs of the accelerometers (*ax*, *ay*, *az*), we can then arrive at the vehicle accelerations in the body frame, given an IMU-positioned at the vehicle centre of gravity:

$$\begin{bmatrix} \dot{U} \\ \dot{V} \\ \dot{W} \end{bmatrix} = \begin{bmatrix} ax + Vr - Wq + g\sin(\theta) \\ ay - Ur + Wp - g\cos(\theta)\sin(\varphi) \\ az + Uq - Vp - g\cos(\theta)\cos(\varphi) \end{bmatrix} \quad \textbf{(2-5)}$$

Here g is the acceleration due to gravity. When integrated with respect to time this acceleration vector gives the body velocities (*U*, *V*, *W*) as follows:

$$\begin{bmatrix} U \\ V \\ W \end{bmatrix} = \int \begin{bmatrix} \dot{U} \\ \dot{V} \\ \dot{W} \end{bmatrix} dt \quad \textbf{(2-6)}$$

The position in the body frame can next be determined by integration of the velocity vector. If we simultaneously transform the velocity to the navigation frame, we obtain the position coordinates (*X*, *Y*, *Z*) in the navigation frame:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \int C_{bn}^{T}(\varphi,\theta,\psi) \begin{bmatrix} U \\ V \\ W \end{bmatrix} dt \quad \textbf{(2-7)}$$

Where $C^{T}_{bn}$ is the transform matrix from the body frame to the navigation frame. Combining the transformation expressions in (4), (5) and (7) above into a single matrix gives us our transition function *f*(*x*,*u*). We assume the observation function *h*(*x*,*u*) to be an unit matrix.

A difficulty with the INS is that it tends to drift as a cubic function of time due to accumulation of biases or errors. The greater part of the INS errors are imputed to the inertial sensors (instrument errors).

Calibration of the INS can cancel out some of the error sources, but some residual errors will inevitably remain behind. The effects of integration compound these: errors in the accelerations and angular rates lead to an accumulation of position and velocity errors (Groves 2008). The dominant INS error sources can be listed as follows:

- alignment errors
- accelerometer bias
- non-orthogonality of gyros & accelerometers
- gyro drift due to temperature change
- gyro scale factor error
- random noise

The analytical INS model described in equations (2.1)-(2.7) has been used to develop a simulator of a MEMS INS/IMU system affixed atop a quadrotor UAV. Figure 2.7 shows a screen capture of the global model interface. Local models within it are responsible for various sub-systems, Figure 2.8 for example shows the interface of the quadrotor UAV's dynamic model. The simulated UAV also has a PID controller which can generate a flight trajectory. To make the simulated on-board IMU realistic, realistic error terms such as those listed above have been injected into the idealised IMU readings, as depicted in Figure 2.9. Extraneous factors that would impact on the dynamics of a real-life UAV have also been incorporated, such as wind velocity. The model described has been used to generate the test and training data used in validating the algorithms described in Chapters 3-5.

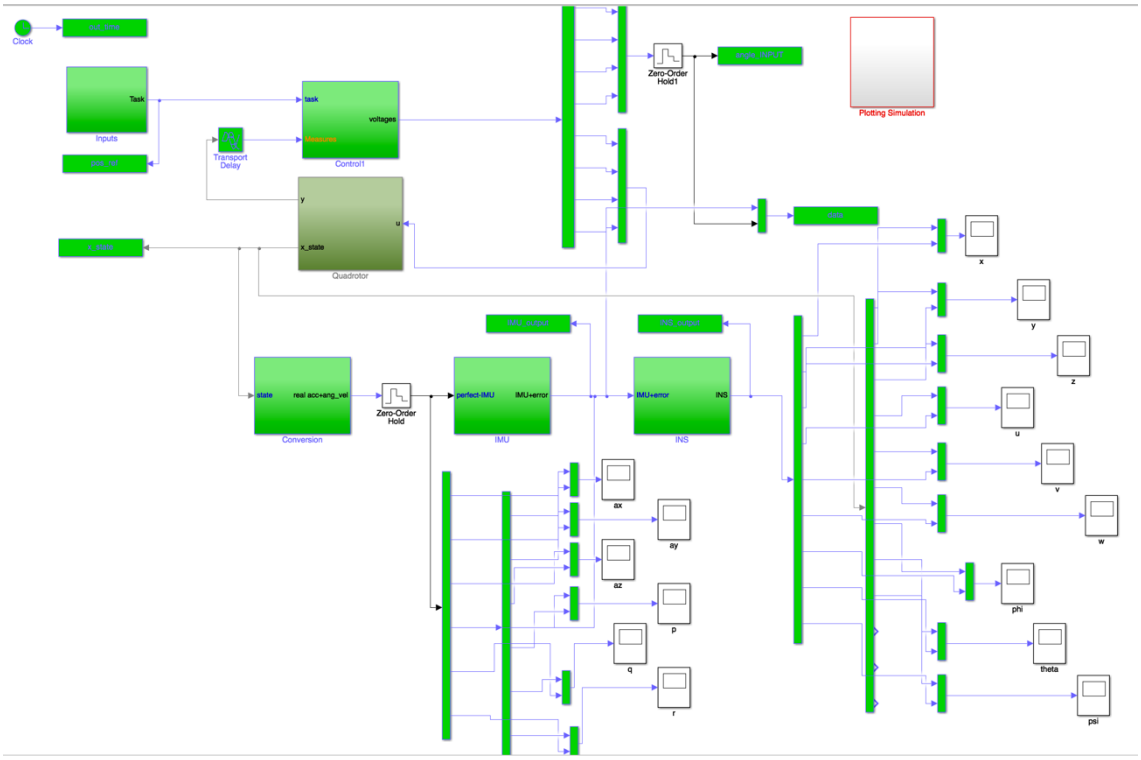



**Figure 2-7 Simulink model of UAV INS**

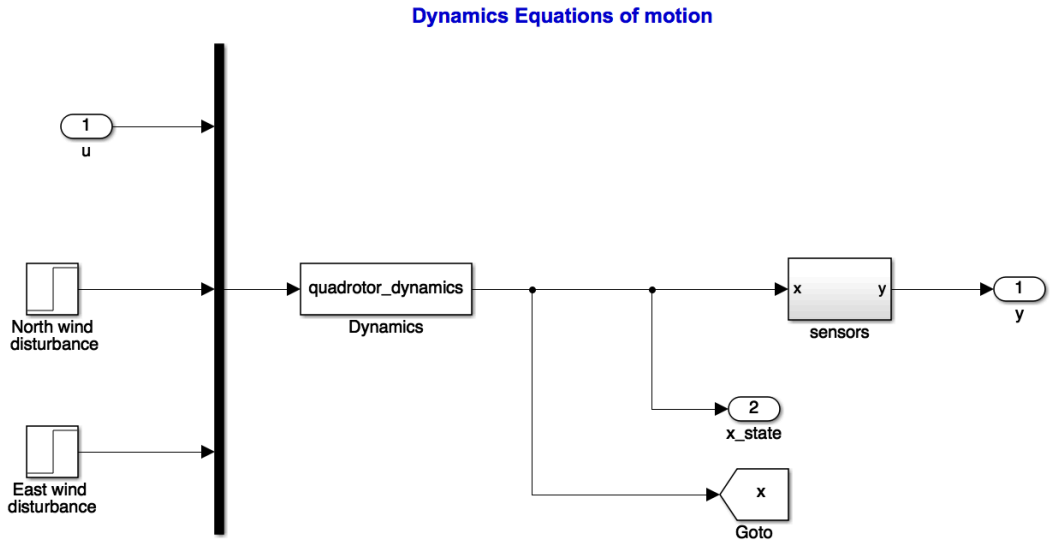



**Figure 2-8 Quadrotor dynamic model**

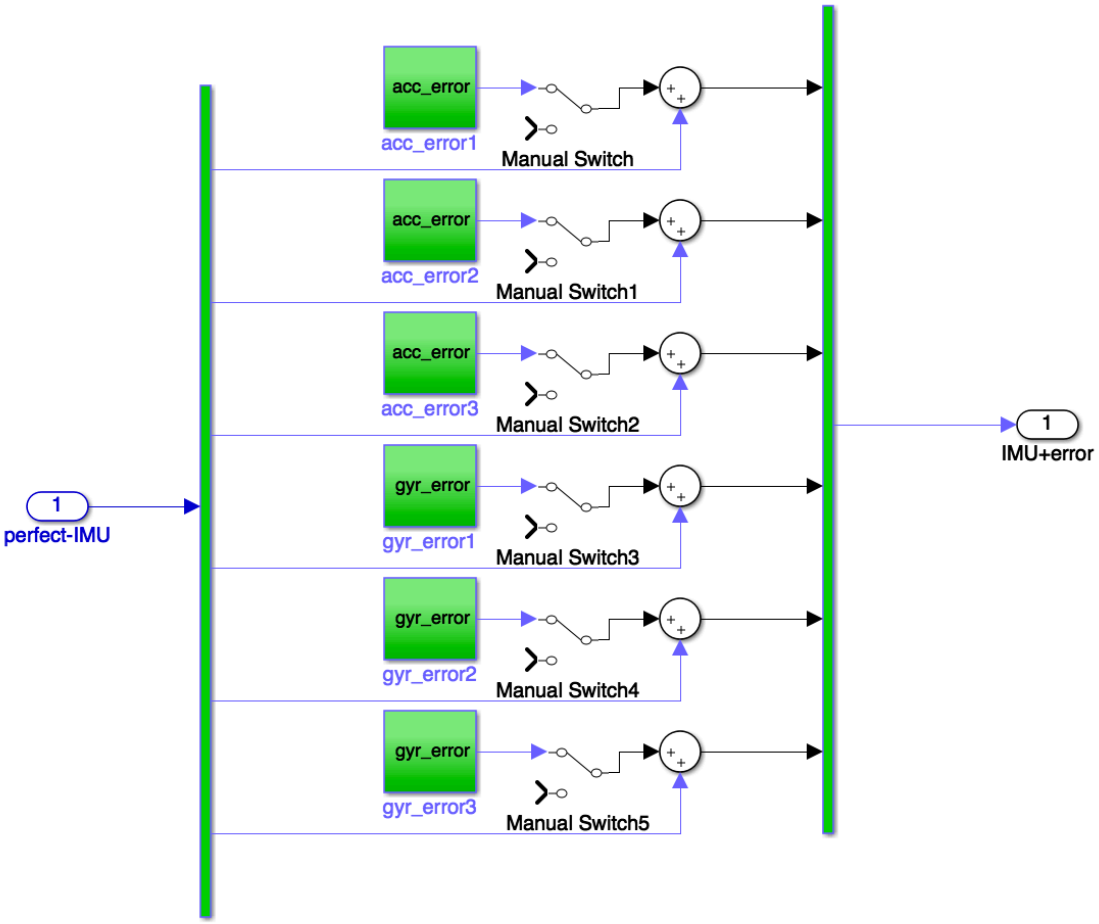



**Figure 2-9 Simulation of IMU biases & error terms**

### 2.1.4 Global positioning systems (GPS)

Global Positioning System (GPS) from The American Department of Defence and the Global Orbit Navigation Satellite System from Russia (GLONASS) are the two Global Navigation Satellite Systems (GNSS) currently used. Galileo is the third system which is launched by both the European Union and Space Agency, coming into use from the year 2008. The well-established GPS has the advantage of a broad range of relatively inexpensive receivers and more availability for users. The Standard Positioning Service (SPS) is open for civilian use while the Precise Positioning Service (PPS) is only intended to U.S. military users.

A minimum of 24 satellites on six equally spaced orbits around the earth should be in operation, enabling the GPS to provide 3-dimentional position estimation and the Coordinated Universal Time (UTC), known for its high accuracy.

There are 31 GPS satellites in orbit currently. The system transmits a Course Acquisition (C/A) code for civilian use one frequency, 1.57542GHz known as L1, and a Precision (P(Y)) code for military use, on a second L2 frequency, 1.22760GHz. The information is transmitted using pseudo-random codes with Code Division Multiple Access (CDMA) with an accuracy of circa 20-27m without Selective Availability. Additional correction can be made using 2 frequencies transmission of the P(Y) code acting against atmospheric effects on the signals for military applications.

In order to ensure accurate timing information between GPS satellites and the associated ground stations, atomic clocks are embedded and they operate on GPS time across the entire network. User's position is calculated using the Time of Arrival (TOA) of a one-way ranging technique, and then triangulation which requires three satellites and a fourth one to correct the clock in the user device with GPS time. The user's device must correlate the pseudo-random codes with those from the satellites (11 satellites could be in view and available at any time) and when 'locked-on' can read the GPA message. The device finds the location of each satellite in view and the range to each satellite is then calculated through the GPS time along with the time error of its own internal

clock. This equidistant range from the satellite to any position is described by the surface of a sphere and where 2 range spheres intersect a circular perimeter will be evident. Adding a third sphere with errors ignored would resolve this plane to a point. When timing and atmospheric errors are considered then the calculated location becomes a 3-D triangular shape. Adding information from more satellites allows this error volume to be reduced (Nemra 2011).

### 2.1.5 INS/GPS fusion

Inertial navigation systems have small bias errors which are continuously increasing with time. Hence, additional aerial vehicle position information from an accurate navigation sensor, such as a GPS system, is required. A GPS sensor will help to estimate the INS bias errors using a navigation filter, which will then give improved UAV position. Nowadays, fusing data from different sensors to improve performance of the overall sensing system becomes necessary in various applications. For aerial navigation, fusion of GPS measurements with INS measurements by means of filtering techniques is vital to deliver the level of localisation precision required by UAV missions.

Currently, the most used technique to fuse navigation data is the Kalman filter (KF) (Nemra & Aouf 2010). Although the Kalman filter is capable of providing real-time vehicle position updates, it is based on linear system models and it suffers from linearisation issues when dealing with non-linear models. In this case, an extended Kalman filter (EKF) is adopted, where by means of Taylor series expansions, the non-linear system is linearised and approximated around each current state estimate. A linear Kalman filter is then applied to produce the next state estimate. When large deviations between the estimated state trajectory and the nominal trajectory exist, the non-linear model is weakly approximated by a Taylor series expansion around the conditional mean. This makes higher-order terms of the Taylor series expansion necessary.

In the EKF, these high-order terms are neglected. Other data fusion techniques based on probabilistic approaches are also used in the literature. One of these techniques is particle filter (PF). The main drawback of this filter is its

computational requirement, which makes it less suitable to real-time applications such as aerial navigation problem.

## 2.2 FDI Methodology

### 2.2.1 Fault modes

Faults on individual inertial sensors are due to hardware failure and can reveal themselves as missing output values, null readings, replicated readings, or excessive errors beyond specified tolerances (Groves 2008). When a sensor fault is detected, further measurements from that sensor are no longer deemed acceptable. Without hardware redundancy, this may lead to the whole inertial navigation solution being compromised and no longer operable. When large errors manifest across all inertial sensors, this can be an indication of adverse environmental conditions causing the system to become unstable, or of incipient failure. The whole IMU or INS may also be subject to a power, software, or communications failure.

The inertial system model used as the basis for the implementations presented in this chapter comprises three orthogonal accelerometers along the body axes and three gyroscopes. The state and measurement variables derived from these which can be monitored for faults are, respectively, the aerial position coordinates ($x$, $y$, $z$) and the attitude Euler angles ($\varphi$, $\theta$, $\psi$). The time-dependency of faults can be classified as (Isermann 1997):

- Abrupt fault (stepwise)
- Incipient fault (drift-like)
- Intermittent fault (interval-wise)

Figure 2.10 overleaf visualises these time-dependencies.

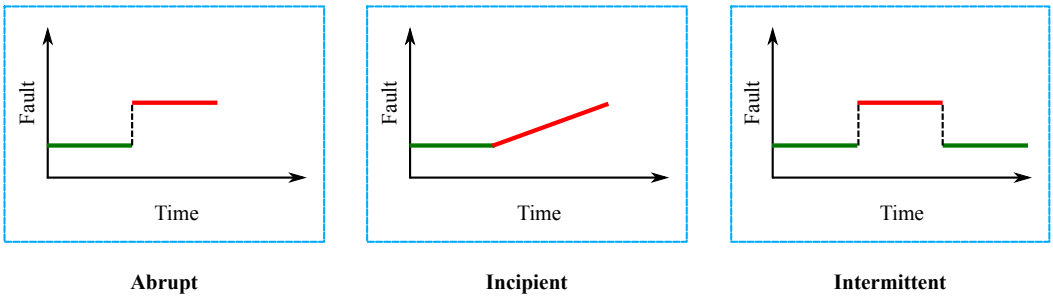



**Figure 2-10 Time-dependency of faults**

Faults, when structured (i.e., it is known how they enter the system dynamics), can be incorporated to complete the state space description:

$$x_{k+1} = Ax_k + B(u_k + f_{a,k}) + Cf_{c,k} + Ed_k \qquad \textbf{(2-8)}$$
$$z_k = Hx_k + Du_k + f_{s,k}$$

Where $f_{a,k}$ denotes an actuator fault, $f_{c,k}$ denotes a component (process) fault, $f_{s,k}$ denotes a sensor fault, and $d_k$ is a vector of disturbances acting on the system. $C$ and $E$ are called distribution matrices for $f_{c,k}$ and $d_k$.

In this study, we are concerned with sensor faults to the exclusion of the other fault types. Figure 2.11 summarises the different fault types.

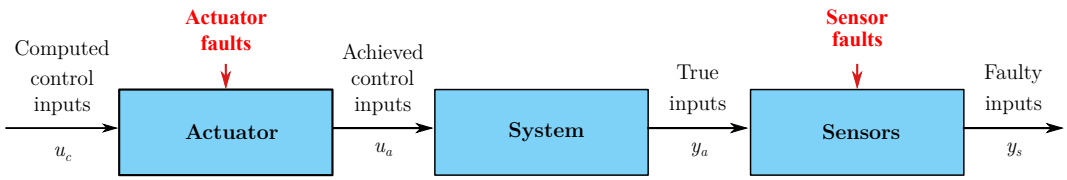



**Figure 2-11 Fault types**

### 2.2.2 FDI methods

Many classifications of FDI are available from the literature. One popular classification divides FDI methods into model- and non-model-based. In this study, the emphasis is on model-based FDI. Such schemes can be classified into two primary groupings: FDI involving residuals and FDI involving fault estimation, as can be seen from Figure 2.12.

Residual generators (Figure 2.13) compare signals from a quantitative model and hardware observations, and the filtered difference forms a residual signal; (Patton 2000) provides a comprehensive discussion of such schemes.

When fault-free conditions exist, the residuals should be zero or close-to-zero, and non-zero otherwise. A threshold is usually applied to residuals to avoid false alarms or misdetection.

State observers are discussed in (Abid 2010).These include the well-known Kalman filter and its derivatives amongst other techniques.

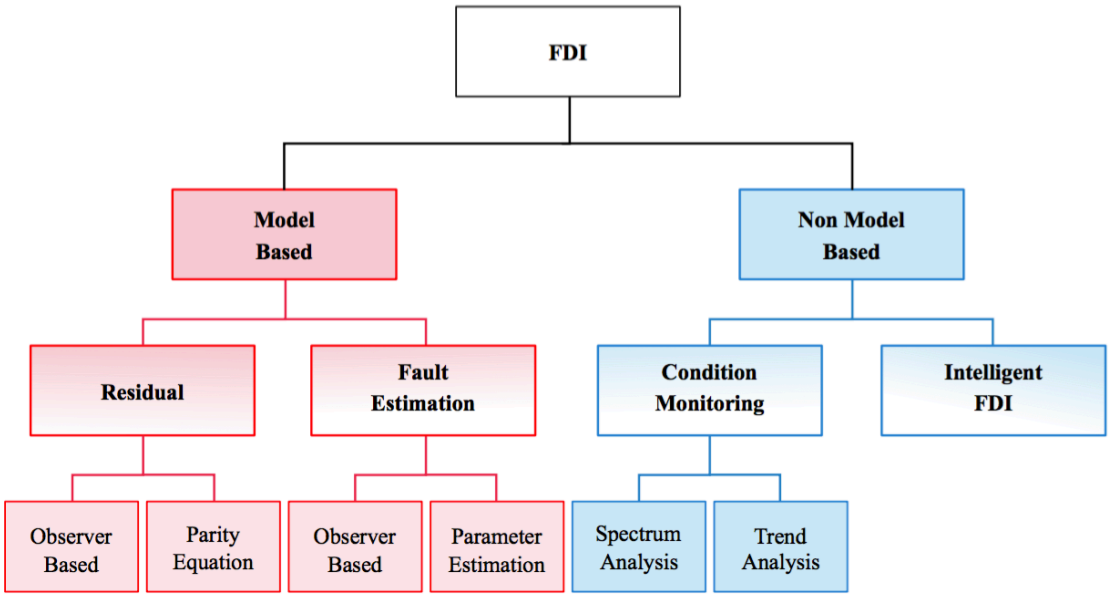



**Figure 2-12 Classification of methods in FDI**

(Kinnaert 2003; Isermann 2006) detail parity relations and parameter estimation.

Filtering techniques and fusion architectures applied to fault detection in navigation systems are discussed in (Marcos et al. 2005; Berdjag et al. 2010; Q. Cheng et al. 2008). The latter two sources also cover distributed filtering techniques as well. Other sources dealing specifically with distributed sensor fusion and FDI across multiple redundant navigation sensors include (Shim & Yang 2010; Waegli et al. 2008; Bancroft & Lachapelle 2011).

There are a number of comprehensive surveys of FDI methods, for example (Marzat et al. 2012).

Standard Kalman filters function as observers and can thus be used to detect faults by means of generating residual signals by comparing real and estimated outputs.

The fundamental Kalman filter concept has undergone modifications and spun off a number of variants, the extended Kalman filter for non-linear systems for example(Li & Olson 1991) and for parameter estimation which the parameters to be estimated are added on to the state vector as extra states.

The Kalman filter can also be composed into a bank of Kalman filters (Kobayashi & Simon 2003; Kobayashi & Simon 2004) or interacting multiple model Kalman filters (IMM-KF) (Rago et al. 1998; Zhang & Jiang 2001) with the aim of creating a residual which can be used for fault detection. The Kalman filter has also been hybridised with predictive control methods, which could potentially be applied to FDI.

Applying the principles concerned in the design of $H\infty$ controllers, observers have been designed as a tool for residual-based fault diagnosis (Marcos et al. 2005). The key concept here is to make the residual sensitive only to faults and insensitive to disturbances and model errors(Marcos et al. 2005). This is done by selecting the observer gain (for example by using linear matrix inequalities (LMIs) which minimise the $H\infty$ norm between the uncertainty and the residual signal.

Following residual generation, residual evaluation or change detection methods can be brought to bear, examples include:

- mean and variance estimation (Basseville & Nikiforov 1993)
- likelihood-ratio test (Marzat et al. 2012)
- Bayesian decision theory (Adams & MacKay 2007)

In the absence of a formal dynamical model, a description of the system has to be derived from real-time measurements, perhaps elaborated by a process history. Given such data, one of two strategies may be employed.

The first of these is classification, which starts of by enumerating classes built from a database. This can be done either in a supervised way (i.e., by first labelling the data) or in semi-supervised fashion (i.e., clustering data points by some criterion that measures their proximity to one another and having an expert apply labels to the identified classes). Subsequently a classification algorithm can be trained to assign freshly sampled data points to classes representing normal or anomalous behaviour.

A second strategy is regression, which is used to develop a statistical model that utilises redundancies in the process history to predict output values given inputs and thus generate residuals by comparing predicted against measured values.

In the event that no process history can be construed, the only information available regarding the monitored system is empirical expert knowledge. This can be used to construct expert systems. These comprise a series of if-then-else rules that endeavour to imitate human deductive reasoning by connecting premises to conclusions using chains of logical inference concerning events. If an illegitimate sequence of events is detected, this may be labelled as a fault.

Some of the disadvantages of this approach are the inherent loss of generality and its incapacity to deal with scenarios that have not been hardwired into the structure of the knowledge base (Angeli & Chatzinikolaou 2004).

Qualitative trend analysis attempts to resolve an observed signal into a sequence of known primitives (such as ‘stable’, ‘increasing’, or ‘decreasing’). This resolution may be achieved through analysing the signs of a sequence of derivatives of signals and putting them through a rule base, or by comparing patterns against a database storing samples of known primitives (Maurya et al. 2007). Both approaches involve careful formulation of heuristic rules. Faults are then established in the same way as in the case of expert systems.

In situations when a process model is discernible but its accuracy and its predictions are in some doubt, qualitative equations are an alternative solution method that can be used to model the variation of the process variables. Qualitative physics of this sort has the same aim as the above-enumerated techniques; that is, to prescribe the evolution of the process so as to detect anomalous behaviour (Panati & Dupré 2001).

A signed digraph (SDG) is another approach used to express causal relationships (Montmain & Gentil 2000). Regrettably qualitative modelling is rather limited in terms of its predictive ability, except in very basic cases.

When we have a process history on hand, fault diagnosis may be formulated as a pattern recognition problem, where freshly sampled observations are classified in predefined classes. Prior knowledge is encoded in the form of a database built up from observations of the monitored variables, which may for example be state variables or data parameters. As a first step, two off-line operations have to be performed: the data are sorted into classes and a decision rule learnt from the data. Classes are hence derived and vectors in the database are mapped to them.

To perform fault diagnosis, the classes adopted are: one to describe normal operation and the rest to cover all of the possible fault modes. An expert may perform the labelling, if such is available, otherwise an algorithm is used, such as k-means clustering (Patton et al. 1999). If the database contains only healthy measurements, another approach is to perform one-class classification (Shin et al. 2005; Mahadevan & Shah 2009; Chandola et al. 2009), though this will not be sufficient for fault isolation.

When the training data have all been labelled, a decision rule must be derived and adapted to classify new vectors in the appropriate classes. For this purpose, both parametric and non-parametric approaches are available.

Parametric classification seeks to determine explicit boundaries between classes using basis functions. The simplest case is one of linear binary classification, it also the method adopted for most algorithms of this type (Jain et al. 2000).

In the event of two classes, the aim is to discover a hyperplane that divides the data into two sets with respect to the predetermined labels. This boundary is optimally defined in accordance with some predefined cost function. A vector norm is chosen to calculate distance to the boundary, as well as a regularisation term to prevent over-fitting of the separator function.

When no linear boundary can be proposed, i.e., when the problem is non-linear, more complex functions (quadratic, cubic, etc.) could be adopted, though there are computational costs associated with this, in the way of tuning an expanding set of parameters.

An oft resorted-to solution to the problem of determining classification boundaries is to use neural networks. In this formulation, the difficulty of finding an optimal solution shifts from the choice of parameters to the selection of an activation function and the choice of structure of the network, i.e., the optimal number of layers and the quantity of neurons making up each of these.

Minimising the quadratic distance between the output of the network and the label of the required class necessitates tuning of the weights of the neurons, typically using the back-propagation algorithm, a local gradient algorithm that is known to become trapped in sub-optimal local optima. These methods have found wide application in FDI.

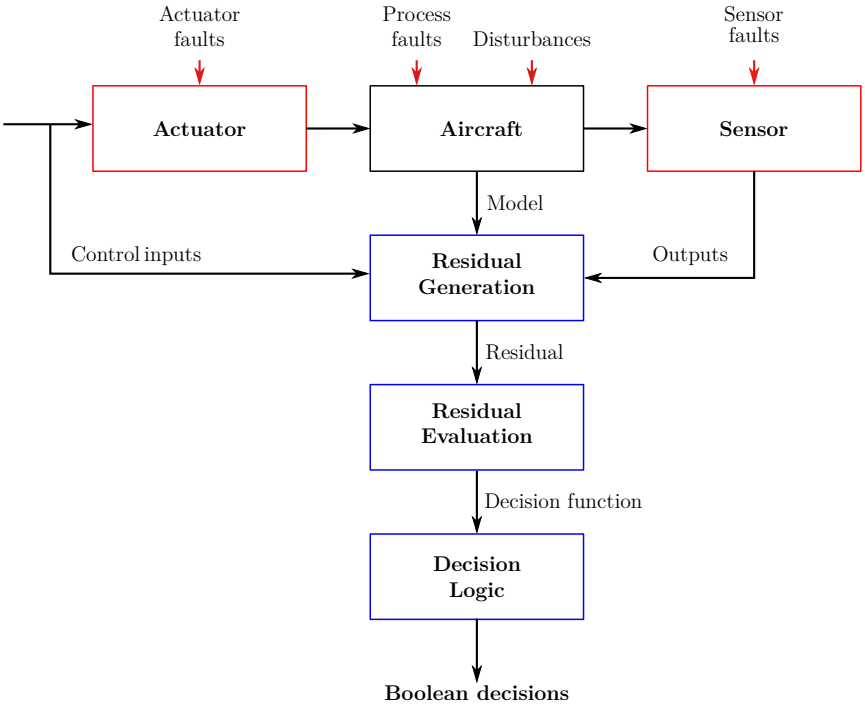



**Figure 2-13 Stages of fault detection**

Modern pattern recognition relies on two key concepts to build non-linear parametric boundaries: kernels and of sparsity. The kernel trick allows an algorithm to generalise linear methods by projecting the data into a high-dimensional feature space, typically a Hilbert space. Sparsity is necessary as it would otherwise be computationally expensive to associate weights with all samples, when they may not all be relevant. An appropriate design of the cost function can be a way to ensure sparsity.

# 3 KERNEL PARTIAL PCA FOR DETECTION AND ISOLATION OF IMU SENSOR FAULTS

Principal component analysis (PCA) is a multivariate statistical technique that has found wide application in fault monitoring, particularly in the processing industry. Its guiding principle is to extract linear structure from high-dimensional data. PCA ranks data systematically according to its variance. The PCA method is most famously a dimensionality reduction technique (Burges 2009) that can be used to compress information and data, retaining only the most salient information. In cases of systems with severe non-linearities, PCA is limited in its performance through its being bound to an assumption of linearity. Principal component analysis is by its nature a linear transformation, which naturally hinders its ability to handle non-linear systems. To get around this, a variety of non-linear extensions of PCA have been proposed, which can identify both linear and non-linear correlation among process variables.

Among the most popular non-linear extensions of standard PCA is kernel principal component analysis (kPCA), which was originally proposed by (Schölkopf 2002). It improves upon previous attempts at non-linear PCA in at least two ways: (i) it does not require non-linear function approximation, (ii) neither does it require a non-linear optimisation procedure. In spite of the many successes kPCA-based monitoring applications have met with, there are some attendant issues: the monitoring model is fixed and that can generate false alarms if the process is a time-varying one; fault isolation is a far more intractable problem in non-linear PCA than in its linear counterpart. Partial kernel PCA, which is presented in this chapter, is an attempt to find a workable solution to the latter problem. It has been shown that there is a strong duality between PCA and the well-known parity space FDI method, and this can be exploited to improve the isolation effectiveness of PCA (Huang et al. 2000). Section 3.1 contrasts PCA and structured parity relations. Section 3.2 develops the partial PCA technique; Section 3.3 - partial kernel PCA. Section 3.4 concludes this chapter validates the proposed technique on a bespoke data set.

## 3.1 Principal Component Analysis with Parity Space-like Isolation Structure

### 3.1.1 Structured parity relations

The parity space approach, also known as parity relations, has well-defined fault isolation properties (Yang & Shim 2007; Hagenblad et al. 2003). (Huang et al. 2000; Gertler et al. 1999) showed that there is a strong relationship between PCA and parity relations, a realisation which leads to the idea of 'isolation-enhanced PCA'. They proposed a 'partial' PCA method imparting the fault isolation properties of the structured parity relations onto PCA. (Gertler et al. 1999) gave a strict proof of the relationship, and proposed a direct algebraic method of generating a set of partial PCA models from the full PCA model.

The central idea of analytical redundancy is to compare the actual outputs to those predicted from the inputs by the model. Discrepancies, mathematically represented as residuals, point to the presence of faults. Because of noise and modelling errors, the residuals are not zero even in fault free situations, and, therefore, thresholds need to be established and the residuals tested against them.

In fault detection, oftentimes simplifying assumptions are made: e.g. that faults are additive sensor or actuator faults, variables are mean-centred, and each variable and its fault conforms to a zero-mean normal distribution. This can mean that techniques devised to perform fault diagnosis under such idealised conditions can come up short in more realistic settings. This goes some way towards explaining some of the limitations of standard PCA.

At time t a linear system has outputs $\mathbf{y}(t)=[y_1(t),...,y_m(t)]'$. The outputs relate to the observed inputs $\mathbf{u}(t)=[u_1,\ldots,u_k]'$, unknown faults $\mathbf{f}(t)=[f_1(t),\ldots,f_{m+k}]'$, and noise and disturbances $\mathbf{d}(t)$ as shown in the following equation:

$$\mathbf{y}(t) = \boldsymbol{A}\boldsymbol{u}(t) + \boldsymbol{B}\boldsymbol{f}(t) + \boldsymbol{C}\boldsymbol{d}(t) \qquad \textbf{(3-1)}$$

**A** and **B** are presumed known. Output and input faults are labelled as $\Delta\mathbf{y}(t)$ and $\Delta\mathbf{u}(t)$ respectively. Thus, we see that $\mathbf{f}(t)=[\Delta\mathbf{y}(t)'\ \Delta\mathbf{u}(t)']'$ and $\mathbf{B}=[\mathbf{I}\ -\mathbf{A}]$. Where **B** is

a block matrix consisting of sub-matrices **-A** and the identity matrix, **I**; or, alternatively, **B** is a concatenation of **I** and **–A**.

We define a 'primary' residual set as

$$\mathbf{e}(\mathrm{t}) = \boldsymbol{y}(t) - \boldsymbol{Au}(t) = \boldsymbol{Bf}(t) + \boldsymbol{Cd}(t) \quad \textbf{(3-2)}$$

where **e**(t) is the computational form by means of which we compute the primary residuals and the second is the fault-effect form that captures the direct relationship between the residuals and the faults. To enhance the usefulness of the residuals and thus improve their isolation properties, they can be transformed as follows

$$\mathbf{r}(\mathrm{t}) = \boldsymbol{We}(t) = \boldsymbol{WBf}(t) + \boldsymbol{WCd}(t) \quad \textbf{(3-3)}$$

where **W** is the transforming matrix. The components of **r**(t) are structured residuals (Huang et al. 2000).

Structured residuals are configured so that each residual is sensitive only to a specific fault or subset of faults. Therefore, each fault activates a different residual or subset of residuals. This pattern is known as a fault code, and provides a specific signature to allow for fault isolation. Taken together the fault codes form a scheme, whereby a set of residuals spiking uniquely locates a fault (Gertler et al. 1999).

A set of fault codes are organised into a residual structure as represented by an incidence matrix. The matrix rows correspond to residuals and the columns to faults. A '0' element indicates that a given residual has not been triggered by a particular fault, whilst a '1' indicates that it has been. The columns of the incidence matrix represent the Boolean fault codes generated by particular faults. The residual structure is said to be 'weakly isolating' if the columns differ from one another and none contain all zero elements, and is said to be 'strongly isolating' if, as well as being different, no one column can be transformed into a copy of another by turning '1''s into '0''s. A straightforward means of achieving strong isolation is by a column canonical structure, in which the same number of '1''s appear in all the columns, although in a different sequence. Such structures

were originally developed in coding theory (Gertler et al. 1999; Patton 2000). They are generally only amenable to isolation of single faults, thus precluding the possibility of isolating multiple simultaneous faults, though detection is still possible. With extended residual sets, it can be possible to design strongly isolating column canonical structures to handle double or triple faults.

Certain modified residual schemes have been proposed in the literature (Frank 1990) that fall within the framework of structured residuals. Included amongst these are residual sets where each residual is affected by

(i) all but one fault,
(ii) only one of the sensor faults,
(iii) only one of the actuator faults.

The first two cases correspond to the generalised observer scheme (GOS) and the dedicated observer scheme (DOS) respectively (Figures 3.1 & 3.2). In our work, since we are looking to isolate only sensor faults, the dedicated scheme is applicable and we adopt a structured residual set based on (ii), as laid out in Table 3.1, throughout this and subsequent chapters. The advantage of this scheme is that it enables the detection and isolation of multiple faults.

**Table 3-1 Residual scheme for detecting multi-sensor faults**

| | y1 | y2 | y3 | y4 | y5 | y6 |
|---|---|---|---|---|---|---|
| **r1** | 1 | 0 | 0 | 0 | 0 | 0 |
| **r2** | 0 | 1 | 0 | 0 | 0 | 0 |
| **r3** | 0 | 0 | 1 | 0 | 0 | 0 |
| **r4** | 0 | 0 | 0 | 1 | 0 | 0 |
| **r5** | 0 | 0 | 0 | 0 | 1 | 0 |
| **r6** | 0 | 0 | 0 | 0 | 0 | 1 |

Table 3.1 corresponds to the values of the incidence matrix denoted in Figures 3.5 & 3.6. The partial PCA component to the algorithm applies a transformation to break up the aggregate data matrix into 6 sub-matrices, each of which contains an output vector relating to a different sensor, while the input vectors bundled with each sub-matrix are identical, resulting in independent residuals.

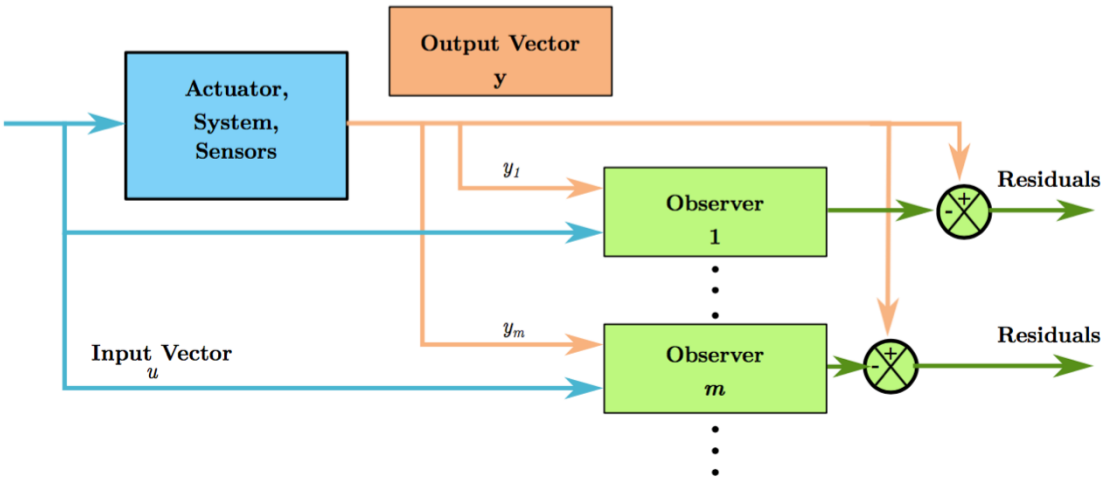



**Figure 3-1 Dedicated observer scheme**

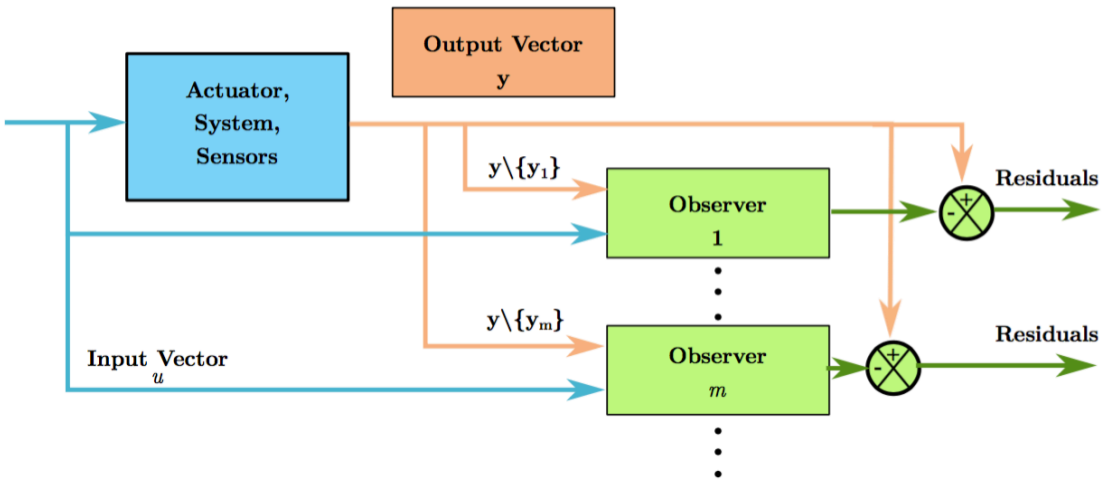



**Figure 3-2 Generalised observer scheme**

## 3.2 Partial PCA

### 3.2.1 Standard PCA

As standard PCA does not differentiate between inputs and outputs, we can conflate **y**(t) and **u**(t) from (3.1) into a single vector

$$\mathbf{x}(\mathrm{t}) = \begin{bmatrix} \boldsymbol{y}(t) \\ \boldsymbol{u}(t) \end{bmatrix} \tag{3-4}$$

and denote the corresponding sensor and actuator faults as

$$\Delta\mathbf{x}(\mathrm{t}) = \begin{bmatrix} \Delta\boldsymbol{y}(t) \\ \Delta\boldsymbol{u}(t) \end{bmatrix} \tag{3-5}$$

By definition, **y**(t) and $\Delta$**y**(t) are m-dimensional vectors, **u**(t) and $\Delta$**u**(t) are k-dimensional vectors, hence **x**(t) and $\Delta$**x**(t) are (m+k)-dimensional vectors. We can rewrite (3.2) and (3.3) in terms of **x**(t) and $\Delta$**x**(t)

$$\mathbf{e}(\mathrm{t}) = \boldsymbol{B}\boldsymbol{x}(t) = \boldsymbol{B}\Delta\boldsymbol{x}(t) \tag{3-6}$$

$$\mathbf{r}(\mathbf{t}) = \boldsymbol{W}\boldsymbol{e}(\boldsymbol{t}) = \boldsymbol{W}\boldsymbol{B}\Delta\boldsymbol{x}(t) \tag{3-7}$$

Assuming we have a training data set with N observations made under fault free conditions **X**=[ **x**(1),…, **x**(n)], performing a singular value decomposition (SVD) on **X** yields

$$\mathbf{X} = \sum_{i=1}^{m+k} \lambda_i \boldsymbol{p_i}\, \boldsymbol{v_i} \tag{3-8}$$

and $p_i$ and $v_i$ are the eigenvectors of **XX'** and **X'X** respectively, while $\lambda_i$ are the eigenvalues of **XX'.** Let there be *m* linearly independent relationships amongst the variables, then *m* of the eigenvalues are zeros, and **X** can be represented by the first *k* PCs, corresponding to the non-zero eigenvalues. However, absolute zero eigenvalues are rarely found in reality. Thus *m* has to determined heuristically, recovering only the eigenvectors with eigenvalues having relatively large magnitudes. The **X** matrix, and consequently **x**(t), can be estimated by the first *k* principal components associated with the *k* largest eigenvalues:

$$\mathbf{x}(\mathrm{t}) \cong \hat{\mathbf{x}}(\mathrm{t}) \tag{3-9}$$

$$= \sum_{i=1}^{k} \boldsymbol{p_i t}_i(t)$$

$$= \sum_{i=1}^{m+k} \boldsymbol{p_i} \left[\boldsymbol{p_i' x}(t)\right]$$

where $\mathbf{t}_i(t) = \boldsymbol{p_i' x}(t),\ i = 1, \ldots k$ are the principal component (PC) scores for the measurements at time t, and $\mathbf{p_i}$ are the PC loadings. Using the nominal fault-free data, we can calculate the PC loadings $\mathbf{p_i}$, evaluate the PC scores, $\mathbf{t_i}$, and approximate the data by (3.9). The sum of squared residuals provides a metric:

$$\epsilon(\mathrm{t}) = \|\boldsymbol{x}(t) - \hat{\boldsymbol{x}}(t)\| \tag{3-10}$$

This metric is known also as the squared prediction error or the Q statistic. Obtaining the PC loadings and the thresholds for $\epsilon(\mathrm{t})$ completes the modelling phase, and one can move on to the monitoring phase.

In the monitoring phase, new observations can be tested against the PCA model, and any $\epsilon(\mathrm{t})$ value which surpasses its threshold implies the occurrence of a fault. This, however, is no indication about where the fault is located.

### 3.2.2 Partial PCA for parity-like isolation

By making use of the concepts developed for parity relations, it is possible to generate structured residuals using the full PCA model. This method determines residuals using the principal components that are not used in the signal representation (H. Cheng et al. 2008). Out of these, new residuals are created through algebraic manipulation which are engineered to be selectively sensitive to different fault subsets. Such a set of residuals, constructed according to the relevant incidence matrix, will then ensure structured isolation for the particular faults concerned. This approach has been reported on in some detail by (Gertler et al. 1999).

If only sensor and actuator faults are of interest, then instead of algebraically manipulating the residuals, an approach that relies on partial PCA models can be used. Partial PCA is PCA carried out on a reduced vector, $\boldsymbol{x^l}(t)$, such that some variables in $\boldsymbol{x}(t)$ are absent. Thus, the residual is only responsive to faults related to the variables present in the reduced vector, $\boldsymbol{x^l}(t)$. Faults related to variables eliminated from the partial PCA will not cause the residuals to exceed their thresholds (Gertler & Cao 2005). The idea of partial PCA was first described by Gertler.

The idea of partial PCA is perhaps best understood through a single transformed residual in (3.11)

$$\mathrm{r}_l(\mathbf{t}) = \boldsymbol{w_l'Bx}(t) \qquad \textbf{(3-11)}$$

$$= \boldsymbol{w_l'}[\boldsymbol{B^{(l)}B^l}]\begin{bmatrix}\boldsymbol{x^l}(t)\\ \boldsymbol{x^{(l)}}(t)\end{bmatrix}$$

Here **B** and **x**(t) have been partitioned so that $\boldsymbol{x^{(l)}}(t)$ holds the variables (faults) to need to be dropped from the residual. The transformation matrix rows, **W**, $\boldsymbol{w_l'}$, are devised in such a way that

$$\boldsymbol{w_l'B^{(l)}} = 0 \qquad \textbf{(3-12)}$$

$$\mathrm{r}_l(\mathbf{t}) = \boldsymbol{w_l'Bx}(t) = \boldsymbol{w_l'B^lx^l}(t) \qquad \textbf{(3-13)}$$

We have for the nominal data

$$\mathrm{r}_l(\mathbf{t}) = \boldsymbol{w_l'B^lx^l}(t) = 0 \qquad \textbf{(3-14)}$$

Plainly, $\mathrm{r}_l(\mathbf{t})$ is not triggered by any fault related to $\boldsymbol{x^{(l)}}(t)$, and under certain conditions is only triggered by those related to $\boldsymbol{x^l}(t)$. To ensure that $\mathrm{r}_l(\mathbf{t})$ responds to faults related to $\boldsymbol{x^l}(t)$, certain rank (i.e. rank of a matrix) conditions have to be satisfied.

A partial PCA model denotes the PCA representation of the sub-system defined by (3.14). Having satisfied rank conditions, this sub-system is composed of *k+1* variables, amongst which there is one relationship. A PCA carried out on this

variable set will immediately establish this relationship (i.e., locate the k-dimensional sub-space where the variables are). Any fault related to the present variables is going to be detected, while those on the absent variables will be ignored.

The partial PCA procedure is equivalent in PCA terms to direct identification of sub-models in the parity space case. In relation to linear systems, it can be considered as an alternative to algebraic transformation for structured residual generation. Although, if the system happens to be non-linear, the algebraic transformation approach is no longer feasible (H. Cheng et al. 2008).

When data is assessed against a well-designed partial PCA subspace, the residual will be only sensitised to faults relating to the variables present in the reduced vector $\boldsymbol{x}^{\boldsymbol{I}}(t)$. Faults relating to variables absent from the partial PCA will not exceed the nominal subspace. Given the selectiveness of partial PCA to fault subsets, it is conceivable that an incidence matrix for such a set of partial PCAs could be devised, ending up with a structure with parity relations-like isolation properties.

## 3.3 Kernel PCA

### 3.3.1 The kernel trick

The kernel trick is a key notion in modern pattern recognition and machine learning. The kernel trick enables the generalisation of linear methods by mapping the data from its input space into some high-dimensional feature space (Marzat et al. 2012).

Working out an inner product in the feature space turns out to be the same as taking the inner product in the original input space and raising it to the power of *d*. This is an extremely attractive result computationally. A high number of dimensions is required to make the feature space sufficiently flexible to be useful. If calculated exhaustively, the computational cost of calculating the inner product in the feature space scales with the number of dimensions (Kung 2014).

A kernel function is defined by the following formalism:

$$\mathrm{k}(\mathbf{x},\mathbf{y}) \triangleq \langle \mathbf{\Phi}(\mathbf{x}), \mathbf{\Phi}(\mathbf{y}) \rangle = \langle \mathbf{x}, \mathbf{y} \rangle^{d} \quad \textbf{(3-15)}$$

In simple terms, kernels provide an efficient way of calculating inner products in high-dimensional feature spaces. At the same time, they provide a convenient non-linear generalisation of inner products. With the use of a kernel, it can be relatively straightforward to build non-linear variants of simple linear algorithms that are based on inner products. In the machine learning literature this is known as the kernel trick.

Applications of kernel methods in general and kernel PCA in particular to FDI have been reported but few are in aerospace (Marzat et al. 2012), though, according to the authors, kernel-based methods appear to be a promising way to enhance or carry out fault detection (Choi et al. 2005).

(Marzat et al. 2012) go on to say:

"*Moreover, the criteria used could be modified to perform regression. It would then become possible to use the same formalism to create a black-box model that can generate residuals by comparing its outputs and the measurements on the system to detect the faults. Finally, it should be pointed out that the choice of the kernel and cost function is crucial and far from trivial, and that adequacy to the data must be carefully checked*."

### 3.3.2 Non-linear PCA as a kernel eigenvalue problem

Kernel PCA (kPCA) extends standard PCA to non-linear settings. Assume a distribution consisting of *n* data points $\mathbf{x}_i \in \mathrm{R}^d$. Before performing a PCA, these data points are mapped into a higher-dimensional feature space *F*,

$$\mathbf{x_i} \rightarrow \mathbf{\Phi}(\mathbf{x_i}) \quad \textbf{(3-16)}$$

In this space, standard PCA is performed. The trick herein is that the PCA can be computed such that the vectors $\mathbf{\Phi}(\mathbf{x_i})$ appear only within scalar products . Thus, mapping (3.16) can be omitted. Instead, we only work with a kernel function $\mathrm{k}(\mathbf{x},\mathbf{y})$, which replaces the scalar product $\langle \mathbf{\Phi}(\mathbf{x}), \mathbf{\Phi}(\mathbf{y}) \rangle$. In kernel PCA,

an eigenvector **V** of the covariance matrix in *F* is a linear combination of points $\mathbf{\Phi}(\mathbf{x_i})$

$$\mathbf{V} = \sum_{i=1}^{n} \alpha_i \widetilde{\mathbf{\Phi}}(\mathbf{x_i}) \tag{3-17}$$

with

$$\widetilde{\mathbf{\Phi}}(\mathbf{x_i}) = \mathbf{\Phi}(\mathbf{x_i}) - \frac{1}{n}\sum_{r=1}^{n} \mathbf{\Phi}(\mathbf{x_r}) \tag{3-18}$$

The vectors $\widetilde{\mathbf{\Phi}}(\mathbf{x_i})$ are chosen such that they are centered around the origin in *F*. The $\alpha_i$ are the components of a vector $\boldsymbol{\alpha}$. It turns out that this vector is an eigenvector of the matrix $\widetilde{K}_{ij} = (\widetilde{\mathbf{\Phi}}(\mathbf{x_i}) \cdot \widetilde{\mathbf{\Phi}}(\mathbf{x_j}))$. The length of $\boldsymbol{\alpha}$ is chosen such that the principal components **V** have unit length: $\|\mathbf{V}\| = 1 \Leftrightarrow \|\boldsymbol{\alpha}\|^2 = 1/\lambda$, with $\lambda$ being the eigenvalue of $\widetilde{K}$ corresponding to $\boldsymbol{\alpha}$. To compute $\widetilde{K}$, we substitute $\widetilde{\mathbf{\Phi}}$ according to (3.18). This substitution gives $\widetilde{K}$ as a function of the kernel matrix $\widetilde{K}_{ij} = \mathrm{k}(\mathbf{x_i}, \mathbf{x_j})$:

$$\widetilde{K}_{ij} = K_{ij} - \frac{1}{n}\sum_{r=1}^{n} K_{ir} - \frac{1}{n}\sum_{r=1}^{n} K_{rj} - \frac{1}{n^2}\sum_{r,s=1}^{n} K_{rs} \tag{3-19}$$

Figure 3.1 illustrates the linearisation of the data in the high-dimensional feature space via the kernel trick, thus allowing for linear principal components to be applied to the transformed data. In other words, the original non-linear problem has been transformed into a linear one in feature space and is now tractable via application of standard linear PCA in this hyper-dimensional Hilbert space. This motivates the use of the kernel trick in the context of inertial navigation, given its non-linear nature, which precludes direct application of linear PCA. However, kernel PCA only gets us so far: a naïve implementation would only permit us to detect faults on a single sensor. In order to perform isolation across the INS sensor triads requires that we modify kernel PCA beyond its standard form. Section 3.3.5 outlines how to do this by kernelising the familiar partial PCA algorithm for linear systems, which is our main contribution in this chapter.

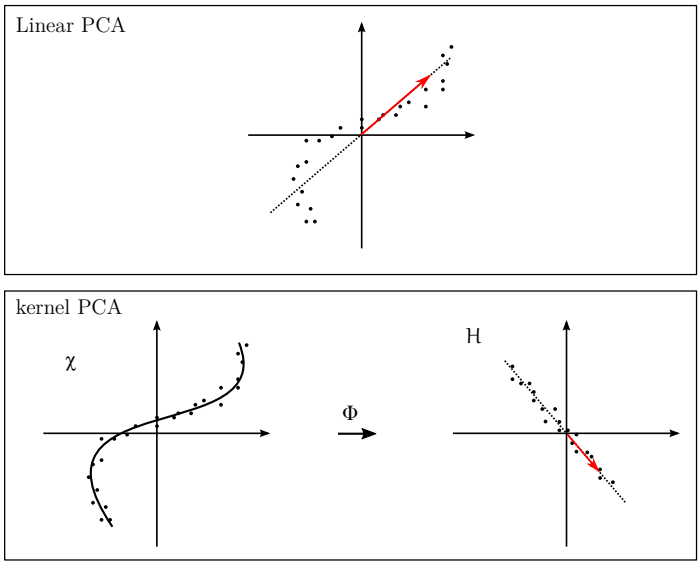



**Figure 3-3 Linear PCA and kernel PCA; input space into feature space**

### 3.3.3 Kernel PCA residuals based on reconstruction error

As novelty measure, we use the reconstruction error in feature space

$$p(\tilde{\boldsymbol{\Phi}}) = (\tilde{\boldsymbol{\Phi}} \cdot \tilde{\boldsymbol{\Phi}}) - (W\tilde{\boldsymbol{\Phi}} \cdot W\tilde{\boldsymbol{\Phi}}) \tag{3-20}$$

With no principal components, the reconstruction error reduces to a spherical potential field in feature space. All we need is the center of the data in *F*, $\boldsymbol{\Phi_0} = 1/\mathrm{n}\sum_{i=1}^{n}\boldsymbol{\Phi}(\mathbf{x_i})$. The potential of a point $\mathbf{z}$ in the original space is the squared distance from the mapping $\boldsymbol{\Phi}(\mathbf{z})$ to the centre $\boldsymbol{\Phi_0}$, i.e.,

$$p_s(\mathbf{z}) = \|\boldsymbol{\Phi}(\mathbf{z}) - \boldsymbol{\Phi_0}\|^2 \tag{3-21}$$

The squared magnitude can be written with kernel functions using the above expression for $\boldsymbol{\Phi_0}$

$$p_s(\mathbf{z}) = \mathrm{k}(\mathbf{z},\mathbf{z}) - \frac{2}{n}\sum_{i=1}^{n} k(\mathbf{z},\mathbf{x_i}) - \frac{1}{n^2}\sum_{i,j=1}^{n} k(\mathbf{x_i},\mathbf{x_j}) \tag{3-22}$$

All parts of this equation are known. The last term is constant, and can therefore be omitted. For RBF kernels, the first term is also constant, and the potential can be simplified to

$$p_s(\mathbf{z}) = \mathrm{k}(\mathbf{z},\mathbf{z}) - \frac{2}{n}\sum_{i=1}^{n} k(\mathbf{z},\mathbf{x_i}) - \frac{1}{n^2}\sum_{i,j=1}^{n} k(\mathbf{x_i},\mathbf{x_j}) \qquad \textbf{(3-23)}$$

Returning to the reconstruction error as defined in (3.20), $\widetilde{\mathbf{\Phi}}$ is a vector originating from the centre of the distribution in feature space, $\widetilde{\mathbf{\Phi}}(\mathbf{z}) = \mathbf{\Phi}(\mathbf{z}) - \mathbf{\Phi_0}$. Let *q* be the number of principal components. The matrix *W* contains the *q* row vectors $\mathbf{V}^l$. The index *l* denotes the $l_{\text{th}}$ eigenvector, with *l*=1 for the eigenvector with the largest eigenvalue (Hoffmann 2007).

We need to eliminate $\widetilde{\mathbf{\Phi}}$ in (3.20), and write the potential as a function of a vector $\mathbf{z}$ taken from the original space. The projection $f_l(\mathbf{z})$ of $\widetilde{\mathbf{\Phi}}$ onto the eigenvector $\boldsymbol{V}^l = \sum_{i=1}^{n} \alpha_i^l \widetilde{\mathbf{\Phi}}(\mathbf{x_i})$ can be readily evaluated using the kernel function k,

$$f_l(\mathbf{z}) = \left(\widetilde{\mathbf{\Phi}}(\mathbf{z}) \cdot \boldsymbol{V}^l\right) \qquad \textbf{(3-24)}$$

$$= \left(\left[\mathbf{\Phi}(\mathbf{z}) - \frac{1}{n}\sum_{r=1}^{n} \mathbf{\Phi}(\mathbf{x_r})\right] \cdot \left[\sum_{i=1}^{n} \alpha_i^l \mathbf{\Phi}(\mathbf{x_i}) - \frac{1}{n}\sum_{i,r=1}^{n} \alpha_i^l \mathbf{\Phi}(\mathbf{x_r})\right]\right)$$

$$= \sum_{i=1}^{n} \alpha_i^l \left[k(\boldsymbol{z},\boldsymbol{x_i}) - \frac{1}{n}\sum_{r=1}^{n} k(\boldsymbol{x_i},\boldsymbol{x_r}) - \frac{1}{n}\sum_{r=1}^{n} k(\boldsymbol{z},\boldsymbol{x_r}) + \frac{1}{n^2}\sum_{r,s=1}^{n} k(\boldsymbol{x_r},\boldsymbol{x_s})\right]$$

Here, the second equality uses Eq. (3). As a result, $p\left(\widetilde{\mathbf{\Phi}}\right)$ can be expressed as

$$p\left(\widetilde{\mathbf{\Phi}}\right) = \left(\widetilde{\mathbf{\Phi}} \cdot \widetilde{\mathbf{\Phi}}\right) - \sum_{\boldsymbol{l=1}}^{\boldsymbol{q}} f_l(\mathbf{z})^2 \qquad \textbf{(3-25)}$$

The scalar product $\left(\widetilde{\mathbf{\Phi}} \cdot \widetilde{\mathbf{\Phi}}\right)$ equals the spherical potential (3.23). Thus, the expression of the potential $p(\mathbf{z})$ can be further simplified

$$p(\mathbf{z}) = p_s(\mathbf{z}) - \sum_{\boldsymbol{l=1}}^{\boldsymbol{q}} f_l(\mathbf{z})^2 \qquad \textbf{(3-26)}$$

This is the desired form of the novelty measure in $R^d$.

The above computation of $f_l(\mathbf{z})$ requires *n* evaluations of the kernel function for each **z**. Since for all *l* components, the same kernels can be used, the total number of kernel evaluations is also *n* (Hoffmann 2007).

### 3.3.4 Kernel functions

The Kernel principal component analysis method has been shown to be effective for monitoring non-linear processes. However, its performance largely depends on the kernel function, and there is currently no general rule for kernel selection. Existing methods simply choose the kernel function empirically or experimentally from a given set of candidates. The kernel function plays a significant role in kPCA, and a poor choice of kernel may lead to significantly lowered performance. Two of the better known kernel functions are these:

- Polynomial kernel:

$$\mathrm{k}(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + \mathbf{1})^d \quad \textbf{(3-27)}$$

- Gaussian RBF kernel:

$$\mathrm{k}(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\|\mathbf{x_i} - \mathbf{x_j}\|^2 / 2\boldsymbol{\sigma}^2\right) \quad \textbf{(3-28)}$$

where $2\delta^2 = w$ is the width of the Gaussian kernel.

The above kernel functions can deliver similar results if parameters are chosen appropriately. The radial basis function is flexible in terms of the setting of its parameter: the width of the Gaussian kernel can be very small (< 1) or quite large (Chouaib et al. 2013).

The RBF kernel reconstruction-error decision boundary in the feature space wraps more tightly around the data than most other kernels and this gives a better description of the data (Figure 3.4).

For RBF kernels, $\mathrm{k}(\mathbf{x}, \mathrm{x})$ takes the same constant value for all **x**. Therefore, in *F*, all $\boldsymbol{\Phi}(\mathbf{x})$ lie on a hyper-dimensional sphere *S*. Figure 3.2 shows only three dimensions of *F*, but for RBF kernels, F is infinite-dimensional . However, this

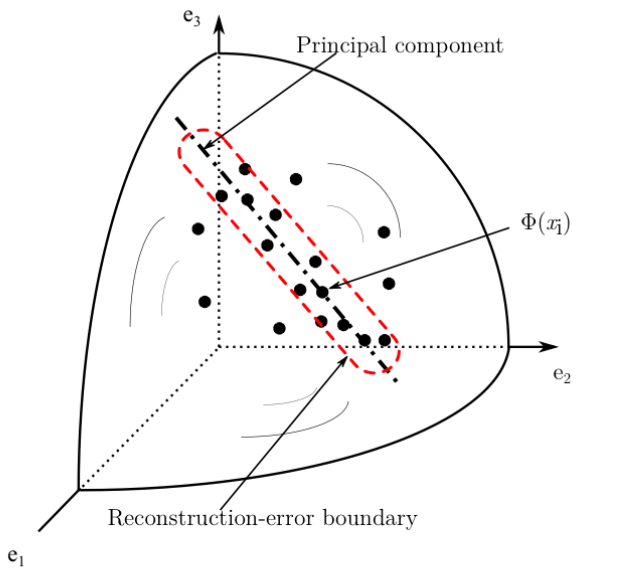
illustration is still meaningful, since *n* data points $\mathbf{\Phi}(\mathbf{x_i})$ can span only a finite space *U*, which is maximally *n*-dimensional if we include the origin in *F*. Due to the rotational invariance of the Euclidean norm, also in *U*, the data lie on a sphere that is embedded in *U* and centered at the origin.

In the direction of the principal subspace, also a boundary emerges since *S* is bending away from the principal subspace. This emerging boundary ensures that the total boundary is closed; this characteristic seems to be missing for polynomial kernels, where $\mathbf{\Phi}(\mathbf{x_i})$ is not restricted to a sphere. To conclude, compared to other kernel functions, for the same number of enclosed data points, the Gaussian RBF reconstruction-error boundary encloses a smaller volume in *S* (Hoffmann 2007).

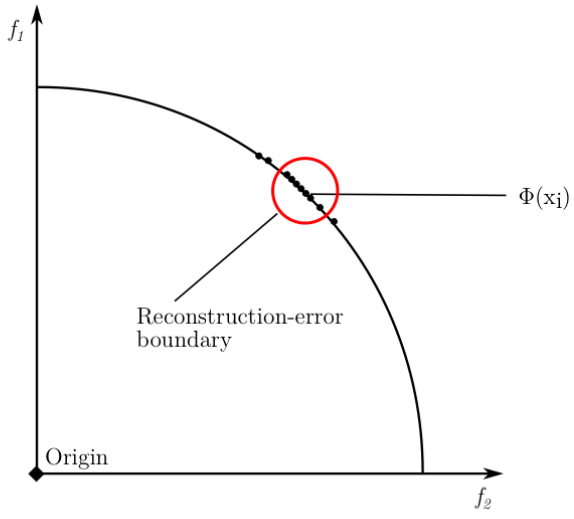
Figure 3.4 provides a schematic interpretation of the RBF kernel decision boundary in feature space. Although the data points are projected onto an infinite dimensional sphere, we can approximate this in three dimensions for visualisation purposes, as the data points in fact lie on a finite dimensional surface embedded in the infinite dimensional projection space. The dimensionality of this surface is determined by the number of the data points. Thus, if we project only three data points, then this sub-region would correspond to a three dimensional sphere like the one depicted in Figure 3.4(a).

The cross-sectional perspective of Figure 3.4(b) shows a planar view of the dispersal of the data points. Points lying inside the ring boundary represent sensor readings that are deemed ‘normal’, whereas points lying outside it are considered outliers and would translate to spikes in the residual. Equally, points contained within the tubular boundary in Figure 3.4(a) represent the 3-D analogue of ‘normal’ points – those without are the outliers.

As noted above, the adoption of different types of kernel functions in kernel PCA produces decision boundaries with different geometries. Thus the kernel selection problem effectively adds an extra parameter to be tuned in optimising the performance of the kPCA algorithm. There is no exact way to select the right kernel function and this choice varies with the specific target system being investigated.




(a)




(b)

**Figure 3-4 a) Decision boundary in the feature space of RBF kernel reconstruction error b) Cross-section view**

### 3.3.5 Partial kernel PCA

To achieve a structured set of partial kernel PCA residuals, each selectively sensitive to a single fault to the exclusion of the rest, as per the dedicated observer scheme, the following routine is observed:

1. Perform standard kernel PCA to determine the number of relations *m*.
2. Construct an incidence matrix, preferably with strong isolation properties.
3. Perform a set of partial kernel PCAs with each one implementing a row of the incidence matrix.
4. Determine the thresholds beyond which abnormality is indicated.

This routine is shown in Figure 3.5, where each partial kPCA model is composed of kPCA loading vectors and a threshold $\theta_i$ on the residual $\epsilon_i$.

After the structured partial kPCA set is obtained, it can be used in online monitoring and fault isolation. New observations are evaluated against the structured set as follows.

1. Run the observed data against each partial PCA subspace and compute the residuals.
2. Compare the residuals to appropriate thresholds and form the fault code $\eta$ according to:

$$\eta_i = \begin{cases} 0 & if\ \epsilon_i < \theta_i \\ 1 & otherwise \end{cases} \quad i = 1, \dots, L \tag{3-29}$$

3. Compare the fault code to columns of the incidence matrix to arrive at an isolation decision.

This routine is shown in Figure 3.6.

The complete partial kernel PCA algorithm captured in Figures 3.5-3.6 represents a forward leap with respect to the state-of-the-art, in that it combines the parity space isolation transferred to partial PCA with kernel PCA, thus providing a new hybrid algorithm that enables both detection and isolation in non-linear settings.

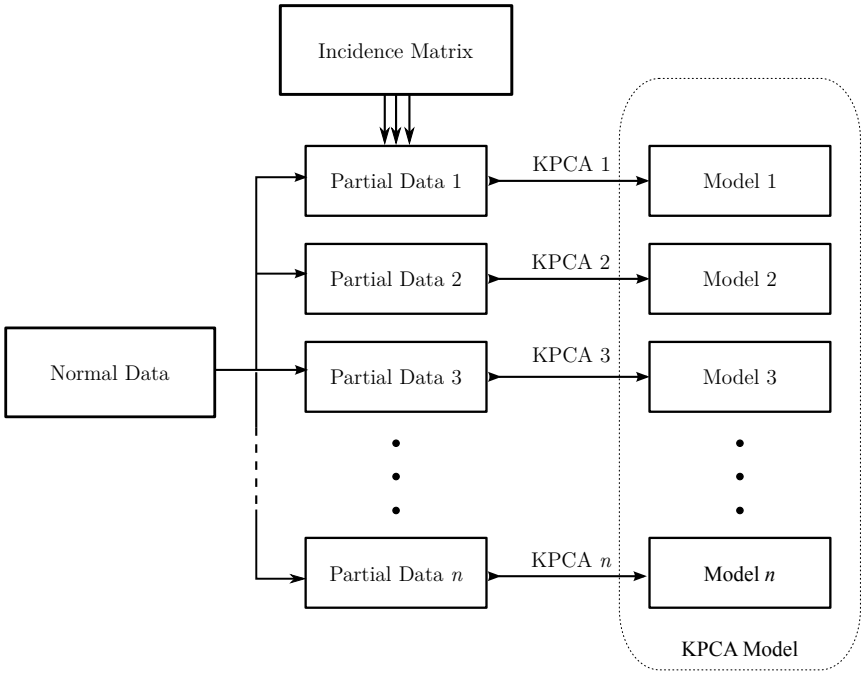



**Figure 3-5 Modelling process of partial kPCA**

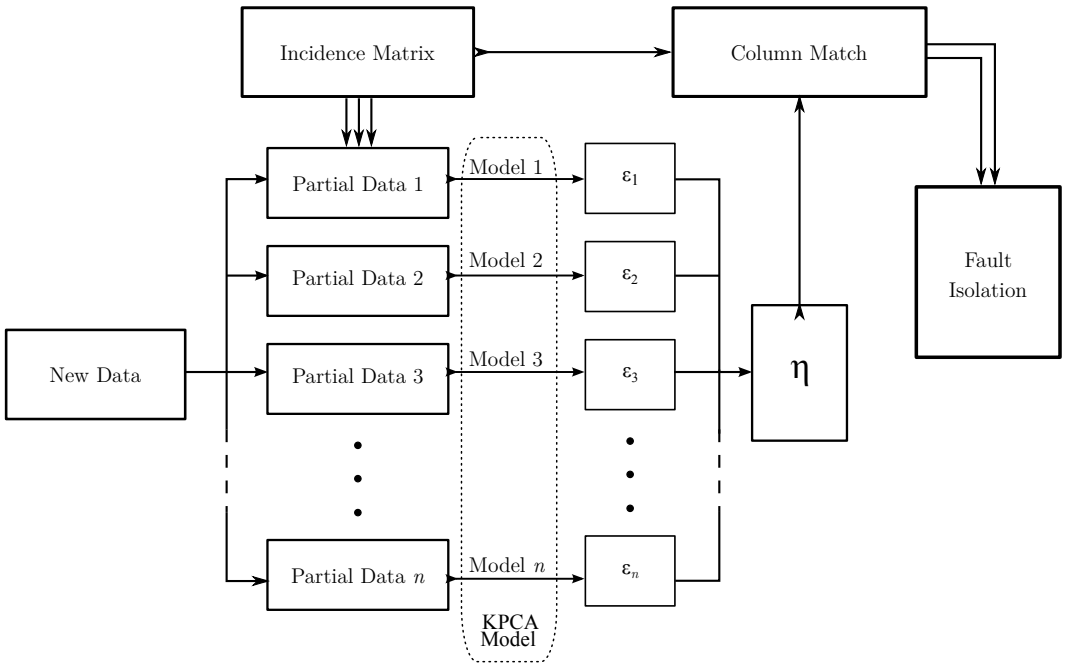



**Figure 3-6 Fault isolation process of partial kPCA**

## 3.4 Experimental Validation

### 3.4.1 Isolation results

Simulation experiments were conducted to assess the performance of kPCA. Synthetic data was generated through a Simulink model of a micro-quadrotor UAV incorporating an emulated MEMS IMU/INS unit. Noise processes and biases were simulated to make the resulting outputs as realistic as possible.

Inputs into the kPCA algorithm are in fact inputs to the simulated quadrotor's dynamic model. These input are fed through from the simulated vehicle's PID controller and control the behaviour of the actuators (i.e. the spin rates of the four propellers). These control signals are computed to have the vehicle maintain a predefined trajectory. The outputs are based on the accelerations and angular rates representing the elements of the IMU observation vector. Residuals are calculated on the reconstruction error principle outlined previously. The outputs used in the kPCA models are baselined against a ground truth signal set determined through the quadrotor dynamic model; that is, the outputs are error terms calculated via the difference between the vector of IMU readings and the projected values of position and orientation from the dynamica model. If the theoretical projections are accurate, this differencing should yield only the IMU error values + noise.

Controller inputs are not used in their raw state, but rather are converted to position information in the body frame and navigation frames. The inputs used are the tri-axis position velocities (U,V,W) – equivalent to the integrated accelerometer values - and the orientation Euler angles ($\varphi$,$\theta$,$\psi$) – equivalent to the integrated angular velocities. The inputs are normalised before being introduced to the kPCA models, as are the residuals, whose absolute values are also used. The input set remains constant across the six partial kernel PCA models trained on the data, what changes is the output.

Training and test data is drawn from a sequence of 10 data sets of a thousand data points each. The same fault scheme of a series of simulated incipient, intermittent and abrupt faults is applied to the output of each sensor in a sliding-

window manner (additive fault placement across the suite of sensors is presented in Table 3-2). Thus each dataset in the sequence is subject to the same fault distribution. Fault magnitude is scaled at 6 signal standard deviations, placing the signal response at the time of a fault occurrence well outside the 95% confidence interval.

Each 1000-point data set is further partitioned into 5 sub-sets or partitions of 200 points each. Duration of all faults is 100 time steps. A partition may contain a single fault or none at all, but there is never more than one fault per partition.

Two versions of kPCA are compared one using the Gaussian RBF kernel and another with the polynomial kernel. A summary of validation results is provided in Tables 3-3 & 3-4 below. Results are aggregated over 10 replications each, i.e. experiments are repeated ten-fold – each item with a different data set but constant fault distribution.

Receiver operating characteristic (ROC) curves are used to assess isolation performance, as well as to serve as a basis for comparison between algorithms; or, in this case, instantiation with different kernel functions. Aggregate receiver operating characteristic curves are presented that span all data sets used in validating performance.

Thresholds and detection/isolation rates are those at the ROC curve's operating point, whereas 'area under curve' assesses global performance – the closer the AUC value is to 1, the better the performance. Above 0.5 is considered competitive; below 0.5 suggests underperformance of the algorithm.

**Table 3-2 Sequence of simulated IMU faults**

| | Partition1 | Partition2 | Partition3 | Partition4 | Partition5 |
|---|---|---|---|---|---|
| **X-axis accelerometer** | INC | --- | --- | --- | ABR |
| **Y-axis accelerometer** | --- | INT | --- | --- | INC |
| **Z-axis accelerometer** | --- | --- | INT | INC | --- |
| **X-axis gyro** | --- | --- | INC | INT | --- |
| **Y-axis gyro** | --- | INT | --- | --- | INC |
| **Z-axis gyro** | INC | --- | --- | --- | ABR |

ROC curves are useful for comparing binary classifiers and as an aid to visualising their performance. They are increasingly being used in machine learning and data mining research. ROC graphs can also be used within a signal detection framework also - to capture the trade-off between true positive and false alarm rates of detection algorithms (Fawcett 2006).

Given an algorithm and data point, four outcomes are possible. A datum evaluates to positive and it is known to be positive - it is classed as a true positive; if it evaluates to negative, it is classed as a false negative. If the datum evaluates to negative and is known to be negative, it is classed as a true negative; if it evaluates to positive, it is counted as a false positive. Given a detection algorithm and test data set, a two-by-two detection matrix can be formed representing the different classification outcomes (Table 3.3). Figure 3.7 depicts diagrammatically the sequence of possible outcomes based expressed in the detection matrix. The labels applied are an alternative set to those in Table 3.3. Thus 'no fault detection' is a true negative; 'false alarm' is a false positive; 'miss detection' is a false negative; 'fault detection' is a true positive.

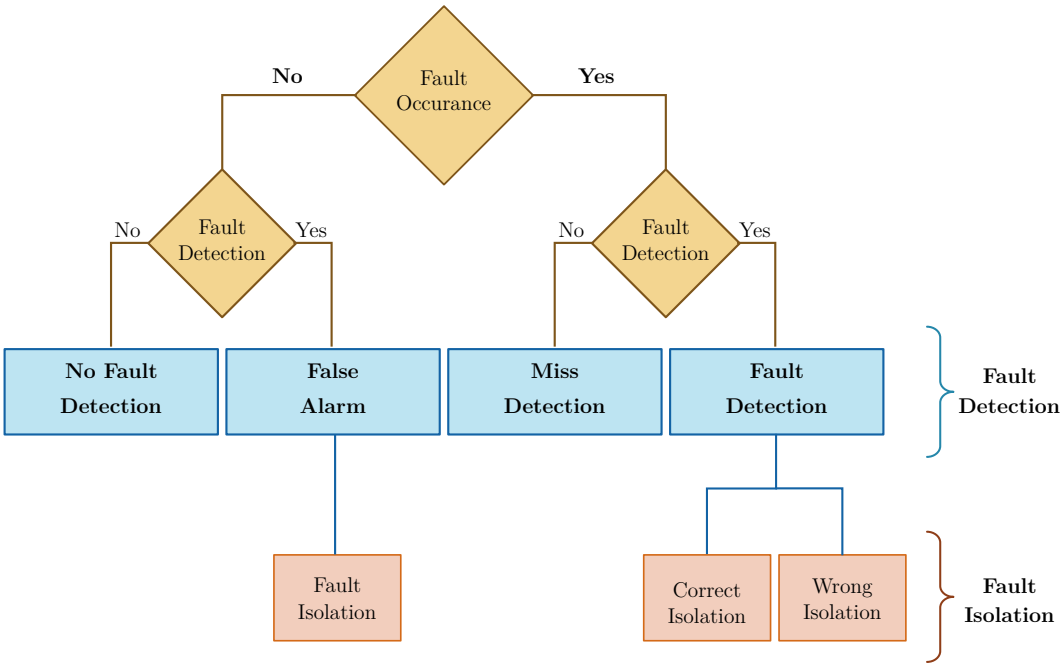



**Figure 3-7 Fault detection and isolation outcomes**

| | Fault | NonFault |
|---|---|---|
| **Detected** | TruePos | FalsePos |
| **Undetected** | FalseNeg | TrueNeg |

**Table 3-3 Detection matrix**

Figure 3.7 and Table 3.3 help to explain some of the detection metrics used in Table 3.5 in order to evaluate isolation rates. The IMU isolation rate being the average detection rate across the 6 sensors. As may be observable from Table 3.5, the true negative/false positive and the true positive/false negative rates are related.

**Table 3-4 Area under ROC curve and optimal threshold performance comparison**

| | kPCA_Gauss_AUC | kPCA_Poly_AUC | kPCA_Gauss_Thresh | kPCA_Poly_Thresh |
|---|---|---|---|---|
| **X-axis accelerometer** | 0.7 | 0.48 | 0.34 | 1.86 |
| **Y-axis accelerometer** | 0.57 | 0.49 | 0.22 | 3.93 |
| **Z-axis accelerometer** | 0.54 | 0.53 | 0.22 | 3.94 |
| **X-axis gyro** | 0.65 | 0.53 | 0.33 | 3.81 |
| **Y-axis gyro** | 0.65 | 0.49 | 0.33 | 3.8 |
| **Z-axis gyro** | 0.67 | 0.48 | 0.37 | 2.05 |
| **Average** | 0.63 | 0.5 | 0.3 | 3.23 |

As can be seen from Tables 3.4-3.5 the kPCA algorithm with the Gaussian kernel is rather more accurate than that employing the polynomial kernel, as might be expected.

**Table 3-5 False positive/negative & true positive/negative rates for RBF kPCA**

| | kPCA_Gauss_TruePos | kPCA_Gauss_FalsePos | kPCA_Gauss_TrueNeg | kPCA_Gauss_FalseNeg |
|---|---|---|---|---|
| **X-axis accelerometer** | 0.62 | 0 | 1 | 0.38 |
| **Y-axis accelerometer** | 0.44 | 0 | 1 | 0.56 |
| **Z-axis accelerometer** | 0.42 | 0 | 1 | 0.58 |
| **X-axis gyro** | 0.46 | 0.01 | 0.99 | 0.54 |
| **Y-axis gyro** | 0.46 | 0.01 | 0.99 | 0.54 |
| **Z-axis gyro** | 0.55 | 0.01 | 0.99 | 0.45 |
| **Average** | 0.49 | 0.01 | 0.99 | 0.51 |

The choice of operating point biases the threshold setting towards minimisation of false positives. This is the classic trade-off between ‘specificity’ and

'sensitivity'. To raise the true positive rate would incur an offsetting drop in the true negative rate and, thereby, increase in the false positive rate.

### 3.4.2 Residuals and ROC curves

Figures 3.10-3.15 display ROC curve comparisons for the partial kPCA algorithm in the cases of the kernel function being RBF or polynomial over the test set and across the different IMU sensors. Table 3.4 summarises the AUC values associated with the individual ROC curves. The closer to 1 the AUC value of a given curve, the better the detection performance. A high AUC value is reflected in the shape of a curve: an AUC value over 0.5 suggests a curve resides within a triangle whose sides are formed by the left-hand vertical axis, top-most horizontal axis and a diagonal drawn between the bottom-left and upper-right corners of the graph. Conversely for curves with AUV values below 0.5. Thresholds are optimised based on the ROC curves – typically the point on the curve closest to the top-left corner determines the value of the threshold.

The RBF partial kPCA residuals shown in Figure 3.8 have a notably low signal-to-noise ratio and low thresholds, showing good approximation of the time-varying behaviour of the various faults applied.

A low threshold can indicate increased rate of true negatives and lower rate of false positives, as is indeed the case with Gaussian kernel kPCA residual. The threshold is optimised for all the data sets used in the validation trial, but the residuals shown are generated over a single data set.

The polynomial kernel residuals in Figure 3.9 are notably indistinct and the high thresholds calculated from the ROC curve reflect the inferior performance of the RBF kernel, as borne out in the performance metrics in Table 3.5.

The AUC curves reveal the Gaussian kernel algorithm has clearly outperformed the polynomial kernel one, indicating that in an IMU fault detection scenario, the RBF kernel should be preferred. This result neatly circles back to the theoretical underpinnings of the choice of kernel function: the RBF kernel provides a tighter decision boundary; thus it is reasonable to expect it would yield a greater detection rate.

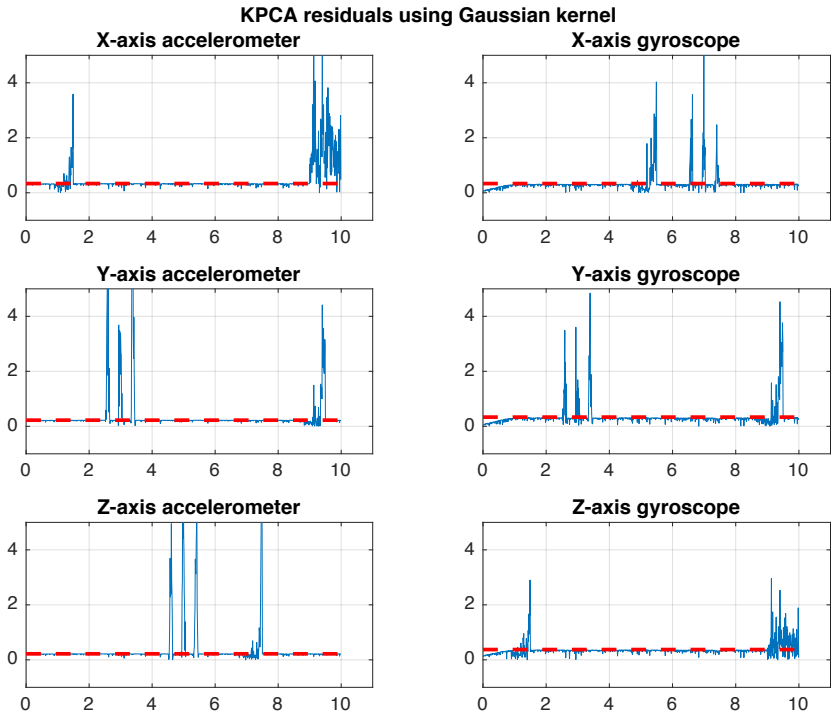



**Figure 3-8 RBF kernel residuals for a single data set**

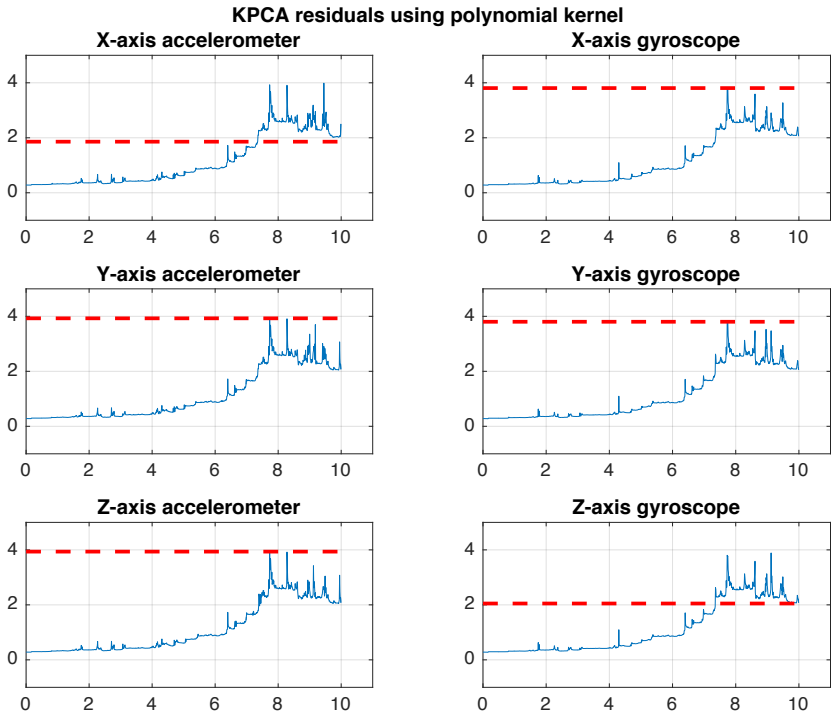



**Figure 3-9 Polynomial kernel residuals for a single data set**

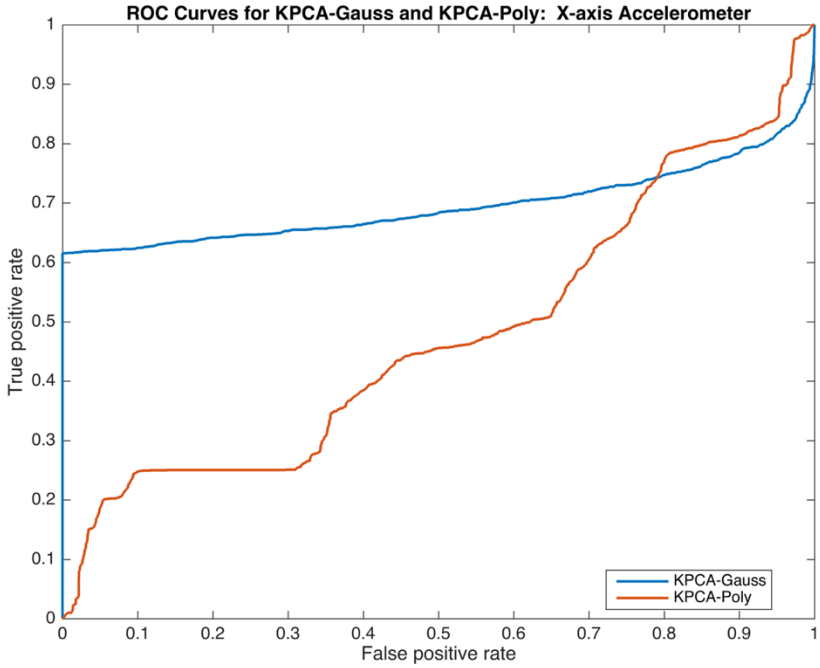



**Figure 3-10 X-axis accelerometer ROC comparison**

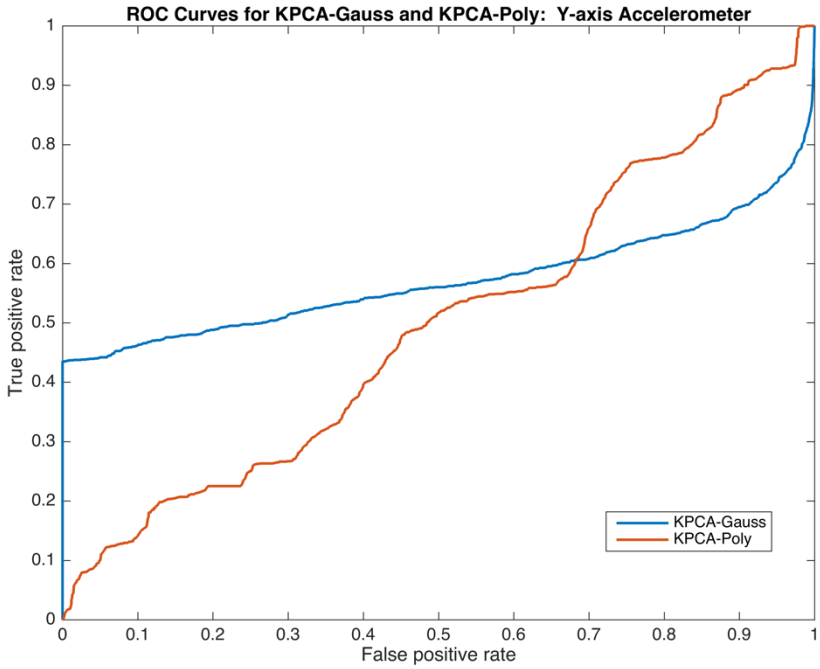



**Figure 3-11 Y-axis accelerometer ROC comparison**

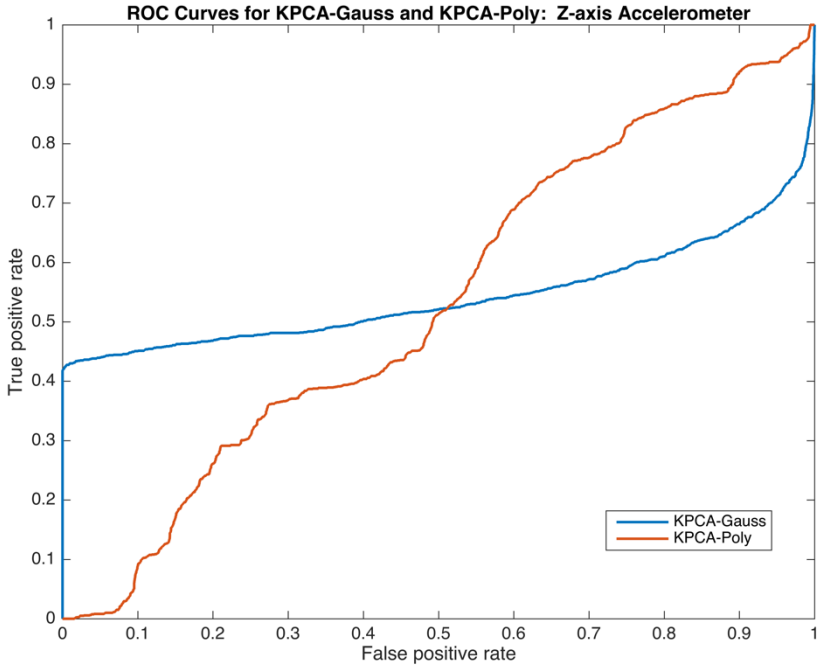



**Figure 3-12 Z-axis accelerometer ROC comparison**

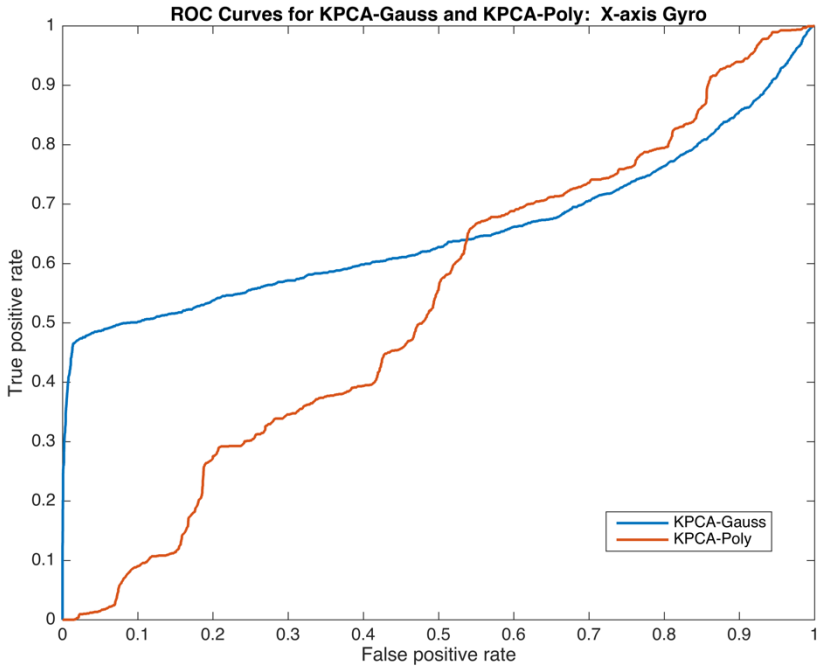



**Figure 3-13 X-axis gyro ROC comparison**

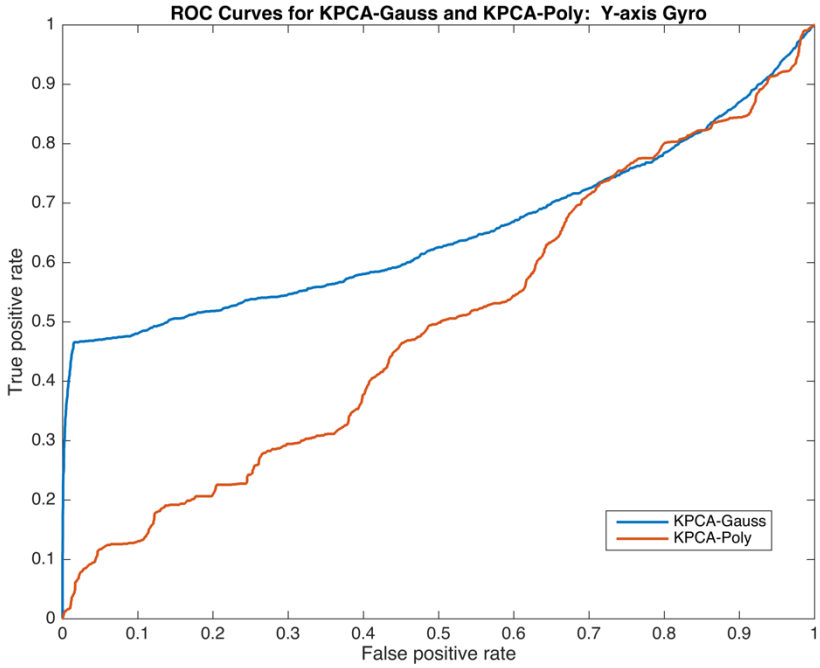



**Figure 3-14 Y-axis gyro ROC comparison**

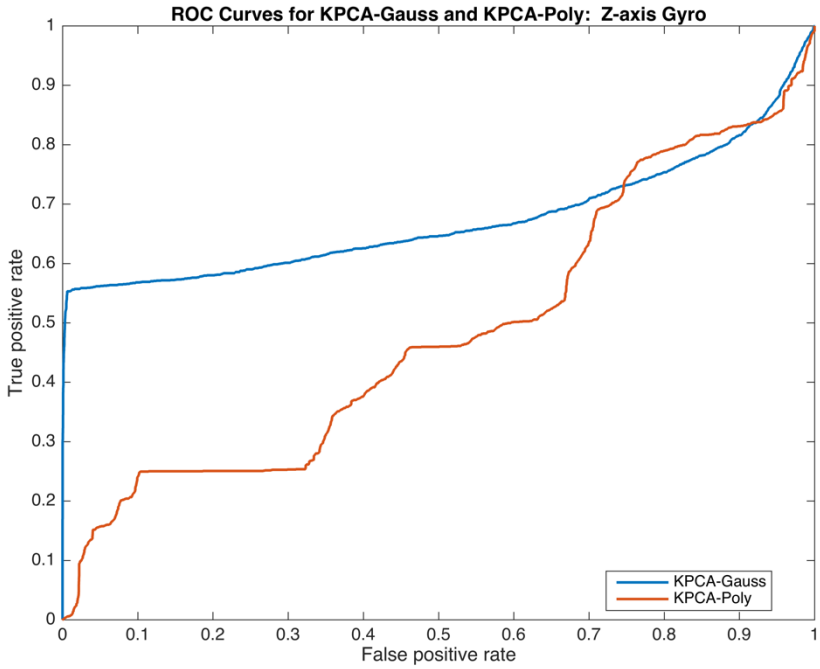



**Figure 3-15 Z-axis gyro ROC comparison**

## 3.5 Conclusion

This chapter has traced the development of a partial kernel PCA algorithm for the detection and isolation of sensor faults. It brings together elements from two related, established algorithms; namely, partial (linear) PCA and kernel PCA. The former exploits the duality known to exist between the parity space technique and linear PCA in order to transmit the well-established parity relations isolation structure onto an operationally equivalent variant of PCA. On the other hand, kernel PCA represents a non-linear extension of standard, linear PCA via the so-called kernel trick. Taken independently, neither technique is able to perform both isolation and detection in a non-linear setting, such as that posed by the IMU diagnostic scenario investigated in this thesis. Merging the two yields a novel solution that exhibits both non-linear modelling capability as well as fault isolation potential.

The developed partial kPCA algorithm is evaluated over a run of ten data sets of simulated IMU data representing the output of three MEMS gyroscopes and an equal number of accelerometers. Superimposed onto this data are fault signals corresponding to incipient, intermittent and abrupt faults. The algorithm's performance is evaluated using two configurations that differ in the choice of kernel function, i.e. RBF and polynomial kernels. The Gaussian (RBF) kernel configuration manages a respectable true positive rate of 0.49 (averaged across all IMU sensors) and a false positive rate of only 0.01, outclassing the polynomial kernel equivalent. An ROC graph has been used to select thresholds and derive performance metrics for the algorithm, where the bias in threshold selection has been towards minimising false positives, as these are deemed potentially more disruptive than a lowered rate of true positives – a trade-off implicit in threshold selection.

# 4 DETECTION AND ISOLATION USING GAUSSIAN PROCESS BAYESIAN FILTERS

The state space INS model, introduced in Chapter 2, forms the basis of the state estimation approach to analytical redundancy, which will be examined in the present chapter. Estimating the system state enables residual generation by comparing model-predicted signals against their measured values. We extend well-established linear residual generators to the non-linear domain using the sigma point transform and GP regression. Section 4.1 relates analytical redundancy and state estimation. Sections 4.2 and 4.3 cover linear and non-linear observers respectively. Section 4.4 elaborates on Gaussian process filtering and goes into the design details of the proposed filtering solutions. Section 4.5 contains simulation results and Section 4.6 concludes the chapter.

## 4.1 State Estimation and Analytical Redundancy

Model-based FDI schemes built around analytical redundancy are receiving increasing attention due to the reductions in size and cost that they can bring. The observer or filter-based FDI approach leverages explicit relations between system inputs and outputs. The error dynamics of the observer can then be treated as residual signals supplying fault signatures (Simon 2008).

The process dynamics are in a sense neutralised by the observer: making the observer sensitive only to disturbances (plant/model mismatch is typically subsumed into the disturbances) and faults. The observer should be designed with robustness in mind; its sensitivity to disturbances should be attenuated, whilst magnifying its sensitivity to faults (Siddiqui & Jiancheng 2012).

Filters achieve state estimation in a recursive procedure, wherein the existence of state and measurement noise is explicitly accounted for. Noise covariances are usually assumed to conform to known Gaussian probability distributions. When steady-state, fault-free conditions prevail, the filter innovation term is expected to be white noise with zero mean and known covariance.

## 4.2 The Kalman Filter

### 4.2.1 Linear Kalman filter

In this section we review the linear Kalman filter (KF) (Mehra & Peschon 1971) by way of establishing some common notions subsequently built upon in the discussion of non-linear filters. The standard Kalman filter is an effective state estimator, but one that is limited to linear systems. In fact, it is known to be optimal in a linear Gaussian setting, where it generates the smallest possible standard deviation of the estimation error. Put another way, the Kalman filter is a minimum variance estimator (Simon 2001).

Most real-world systems, however, are non-linear (D. Simon 2006b). In such cases, the classical Kalman filter does not find direct application. In practice, non-linear filters are utilised more often than linear filters, because real-world systems are generally non-linear. Indeed, the first time the KF was tried out in practice, it was reconstituted to non-linear form for the purposes of NASA's space programme in the 1960s (D. J. Simon 2006).

The discrete linear Kalman filter seeks to estimate the state, *x*, of a controlled process (in discrete time) governed by the following state space equations:

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k + \mathbf{w}_k \qquad \textbf{(4-1)}$$

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

The first equation is known as the state equation; the second is the called the observation or output equation. Between them, these two equations describe a discrete process of linear type. Equations (4.1) contain the following terms: *A*, *B*, and *H* are system matrices; *k* is the time index; *x*, as already mentioned, is the system state; *u* is a known input to the system (called the control signal); *z* (alternatively labelled as *y*) is the measured output; *w* and *v* are Gaussian noise terms - *w* is known as the process noise, and *v* is known as the measurement noise (Simon 2008).

In the case of our target system - the INS – the *x* and *u* terms correspond to a 9-variable state vector of and the 6-variable input vector.

In state estimation problems, we are interested in estimating *x* because it contains within itself all the requisite descriptive information we require of our system. The problem lies in that we are not in a position to measure *x* directly. Rather, we measure *z,* which is a measurement of *x* corrupted by noise *v*. Thus we use *z* in order to estimate *x*, but we do not typically take the value of *z* as is, because it is affected by noise. Matrix *A* maps the state at time *k+1* to the state at the previous time *k*. Matrix *B* maps the optional control input to the state *x*. Matrix *H* maps the state to the observation (Kim 2011).

As just observed, to estimate system state, we could just take $z_k$ to be our position estimate but for the fact that it is made imprecise by noise. The KF provides a more accurate estimate. This is because, aside from the measurement, $z_k$, the KF also uses the information contained in the state equation. The KF prediction equations can be written as follows:

$$\begin{aligned} \hat{x}_{k+1}^{-} &= A\hat{x}_k + Bu_{k+1} \\ P_{k+1}^{-} &= AP_kA^T + Q \end{aligned} \qquad \textbf{(4-2)}$$

The update equations then are:

$$\begin{aligned} K_{k+1} &= P_{k+1}^{-}H^T(HP_{k+1}^{-}H^T + R)^{-1} \\ \hat{x}_{k+1} &= \hat{x}_{k+1}^{-} + K_{k+1}(z_{k+1} - H\hat{x}_{k+1}^{-}) \\ P_{k+1} &= P_{k+1}^{-} - K_{k+1}HP_{k+1}^{-} \end{aligned} \qquad \textbf{(4-3)}$$

Where $\hat{x}$ is the estimate of *x*; *K* is the Kalman gain; *P* is the estimation error covariance matrix; *Q* is the covariance of the process noise, $w_k$, and *R* is the covariance of the measurement noise, $v_k$; *I* is an identity matrix. The goal of the KF is to find an a-posteriori state estimate as a linear combination of an a-priori estimate and a weighted difference between the measurement and predicted state. *K* is the gain factor that minimises the a-posteriori error covariance. The difference is called the innovation or residual , as expressed by the term:

$$(z_{k+1} - H\hat{x}_{k+1}^{-}) \qquad \textbf{(4-4)}$$

The KF recursively follows a prediction/update cycle, with $x_{k+1}$ and $P_{k+1}$ from the last iteration becoming $x_k$ and $P_k$ in the next one. The iterative nature of the KF

thus results in a permanent prediction/correction cycle. To set the cycle going, we need to start with an estimate of the state at the initial time. We also need to start with an initial estimation error covariance, $P_0$, which represents our uncertainty in our initial state estimate. The Kalman filter iterative cycle or loop is illustrated in Figure 4.1 below.

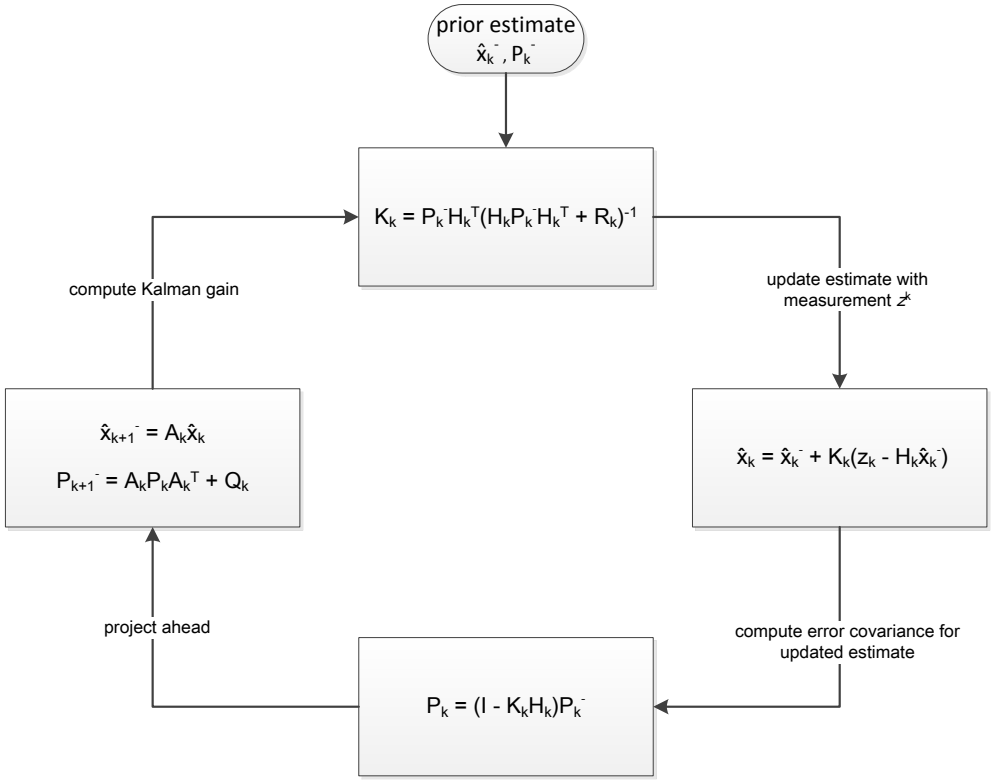



**Figure 4-1 Kalman filter loop**

The Kalman filter has been widely adopted for localisation and navigation tasks, particularly for INS/GPS fusion. An important issue in the application of the Kalman filter with inertial navigation systems in mind is the distinction between the direct and indirect forms, also known as the error-state form. In the total state (or direct) form, the measurements are INS outputs. In the error state (or indirect) form the errors in the vehicle pose measurement variables are either among or a totality or the variables being estimated. Each of the measurements in the error-state form comprises the difference between a particular INS variable and an external source (i.e. ground truth) (Roumeliotis 1999). Figures 4.2-3 depict the structural difference between error- and total-state forms.

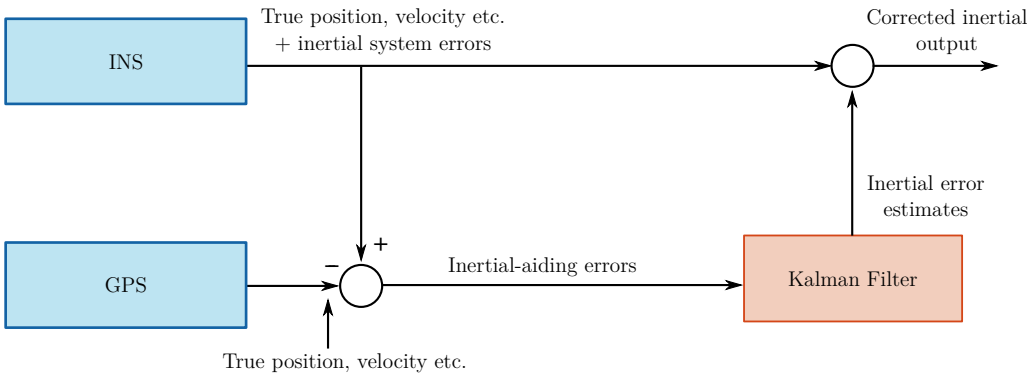



**Figure 4-2 Indirect Kalman filter**

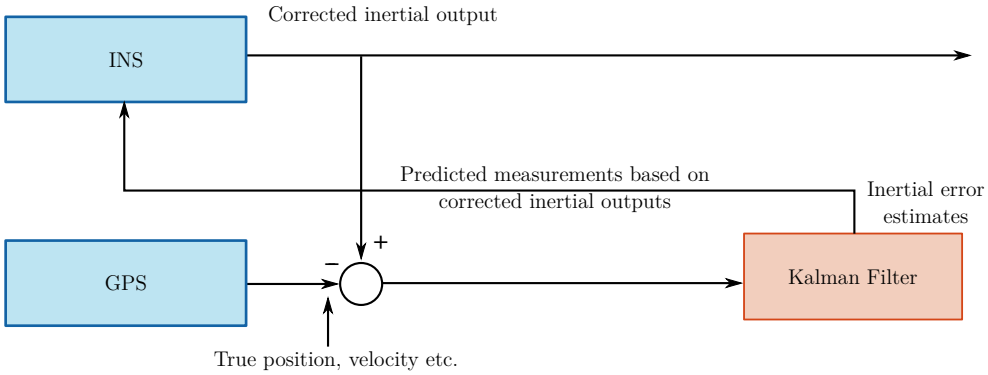



**Figure 4-3 Direct Kalman filter**

### 4.2.2 Kalman innovation filtering

Equation (4.4) defines the innovation term used to generate residuals with the KF. In detecting faults using the KF, we record successive values of this term to generate a set of residuals. When its value departs significantly from the average beyond an acceptable confidence level (i.e., exceeding a predefined threshold) this may be interpreted as a sign that a fault has occurred.

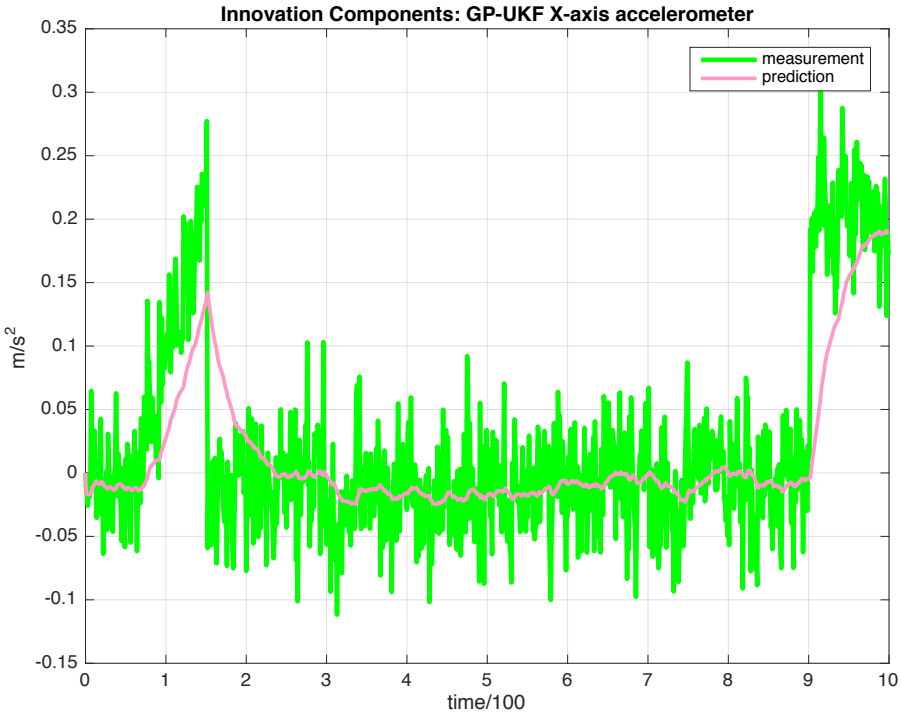
For a non-linear filter, the innovation term of (4.4) modifies to:

$$(z_{k+1} - h(\hat{x}_{k+1}^{-})) \tag{4-5}$$

Measurement innovations monitor the consistency between measurements and state estimates. Innovation filtering is particularly useful in detecting large discrepancies that occur instantaneously, while monitoring the entire innovation sequence enables smaller, more gradual discrepancies to be detected over time. Innovation filtering is also known as spike filtering, measurement gating, or pre-filtering (Groves 2008).

We can also define normalised innovations or residuals by dividing the innovation value by its standard deviation. In an idealised KF, normalised measurement innovations possess zero-mean unit-variance Gaussian distributions, and successive values are practically independent (Groves 2008). However, correlated process or measurement noise, differences between the true and modelled process and measurement noise covariances, neglected error sources and use of non-linear filters all cause deviations from this ideal. In practice, the statistics of normalised innovations need to be evaluated before having FDI algorithms make use of them.

Figure 4.4 provides a graphical description of the constituent signals that appear in the innovation term or residual as defined in (4.5). The plot is based on a single measurement and prediction from a dedicated GP-UKF filter (described in Section 4.4) associated with the X-axis accelerometer sensor of the MEMS-IMU. Smoothing in the form of a moving average filter is applied post-hoc to the raw IMU measurement in order to smooth out the residual.



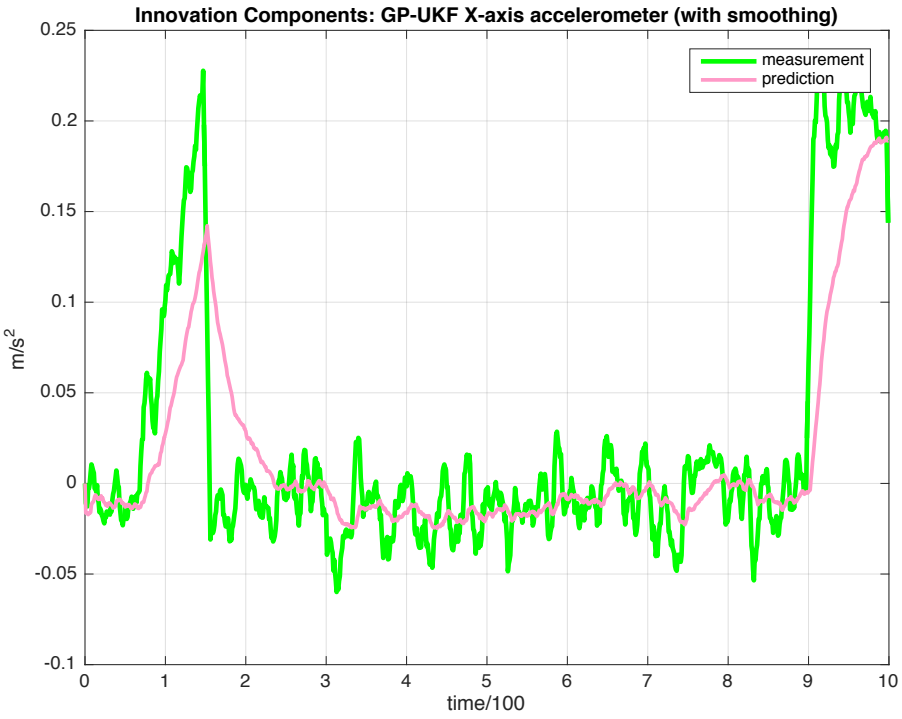




**Figure 4-4 Innovation components (with & without denoising)**

## 4.3 Non-linear Filters

### 4.3.1 The extended Kalman filter

As mentioned in Section 4.1, the KF is a linear filter that is, by implication, only applicable to linear systems. It can no longer provide adequate estimates when used for a system that exhibits non-linearities, even over a small operational range. In such cases, we need to resort to non-linear filters.

Non-linear filtering is not as readily tractable and well-understood as linear filtering, thereby making it altogether more complex to implement. Nevertheless, non-linear estimation techniques are widely used (D. J. Simon 2006). The most common of these is the extended Kalman filter, which, as the name suggests, is a non-linear extension of the Kalman filter. The EKF hinges on the principle that to linearise a non-linear system at certain points opens it up to analysis by linear estimation methods (such as the KF) allowing estimation of the states. In order to linearise a non-linear system, a Taylor series expansion is used to derive the Jacobian matrix. The EKF thus employs the Jacobian matrix to linearise a non-linear system model and hence derive *F* (i.e., *A*) and *H*. The EKF algorithm can be summarised as below, where (4.6) is the non-linear form of the state-space model.

$$\begin{aligned} x_{k+1} &= f(x_k, u_{k+1}) + w_k \\ z_k &= h(x_k) + v_k \end{aligned} \tag{4-6}$$

At each time step, we compute the following Jacobian matrices, evaluated at the current state estimate:

$$\begin{aligned} F_k &= f'(\hat{x}_k, u_{k+1}) \\ H_k &= h'(\hat{x}_k) \end{aligned} \tag{4-7}$$

We then evaluate the following EKF equations:

$$\begin{aligned} K_k &= P_k H_k{}^T (H_k P_k H_k{}^T + R)^{-1} \\ \hat{x}_{k+1} &= f(\hat{x}_k, u_k) + K_k (z_k - h(\hat{x}_k)) \\ P_{k+1} &= F_k (I - K_k H_k) P_k F_k{}^T + Q \end{aligned} \tag{4-8}$$

As can be seen from the EKF filter equations above, the overall flow of the algorithm is essentially the same as that for the linear Kalman filter, save for the non-linear state transition and observation functions and the terms of (4.7).

### 4.3.2 The unscented Kalman filter

The EKF and the unscented Kalman filter (UKF) both approximate a non-linear state-space as a linear system. The extended Kalman filter is the most commonly adopted state estimator for non-linear systems, though it has its downsides (D. Simon 2006a). It can be problematic to tune and may produce unreliable estimates when non-linearities become too pronounced. This is because the EKF relies on linearisation using a first order Taylor series expansion around the most recent estimate, whilst the UKF applies a more accurate, stochastic approximation, also known as the *unscented transform* (Julier & Uhlmann 1997) or the *sigma point transform*. Thus application of the UKF can deliver improvements in performance over the EKF (Thrun & Fox 2005; D. Simon 2006a; Julier & Uhlmann 1997).

To appreciate how the unscented transform is used, consider a random vector, $\boldsymbol{x}$, of dimension $n$ distributed according to a Gaussian with mean $\mu$ and covariance $P$. The aim is to estimate a Gaussian approximation of the distribution over $y = f(\boldsymbol{x})$, where $f$ is putatively a non-linear function. The unscented transform enables this procedure by extracting the so-called sigma points, $\chi$, from the Gaussian and propagating them through $f$. Typically these are found at the mean, $\mu$, and symmetrically along the main axes of the covariance $P$ (two per dimension).

Figures 4.5-4.6 show schematically how the sigma transform is utilised.

The $2n+1$ sigma points $\chi^{[i]}$ are chosen according to the following rule:

$$X^0 = \mu \qquad \textbf{(4-9)}$$

$$X^i := \mu \pm \left(\sqrt{(n+\lambda)P}\right)_i$$

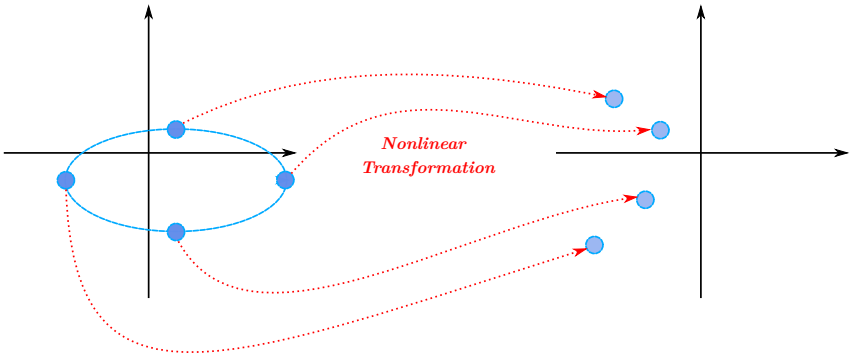



**Figure 4-5 Stochastic approximation via sigma transform**

Here $(\sqrt{(n+\lambda)P})_i$ is the *i*-th column of the matrix square root, and $\lambda$ is a scaling parameter that determines how far apart from the mean the sigma points are placed. The sigma points are next propagated through *f*, giving $f(\chi^{[i]})$. Then the mean and covariance of the function $y = f(\boldsymbol{x})$ can be computed as follows:

$$\mu' = \sum_{i=1}^{2n+1} W_i f(\chi^{[i]}) \qquad \textbf{(4-10)}$$

$$P' = \sum_{i=1}^{2n+1} W_i \{f(\chi^{[i]}) - \mu'\} \{f(\chi^{[i]}) - \mu'\}^T$$

Where the weights, $W_i$, are chosen appropriately. The UKF applies the unscented transform to the process model, *f*, and the measurement model, *h*. The steps of the UKF algorithm are given next. The inputs into the algorithm at each iteration are the mean and covariance of the estimate at time *k-1* along with the most recent control input, $u_k$, and observation, $z_k$.

Compute sigma points and weights:

$$(\chi^{[i]}, W_i) \leftarrow (\hat{x}_{k-1}, P_{k-1}, \kappa). \qquad \textbf{(4-11)}$$

Where $\kappa \in \Re$ is a scaling factor.

Predict next state and associated error covariance:

$$(\hat{x}_k^-, P_k^-) = UT(f(\chi^{[i]}, u_k), W_i, Q) \tag{4-12}$$

Predict measurement and its covariance:

$$(\hat{z}_k, P_z) = UT(h(\chi^{[i]}), W_i, R) \tag{4-13}$$

Evaluate Kalman gain:

$$P_{xz} = \sum_{i=1}^{2n+1} W_i \{f(\chi^{[i]}) - \hat{x}_k^-\} \{h(\chi^{[i]}) - \hat{z}_k\}^T \tag{4-14}$$
$$K_k = P_{xz} P_z^{-1}$$

Evaluate estimate and error covariance:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{z}_k) \tag{4-15}$$
$$P_k = P_k^- - K_k P_z K_k^T$$

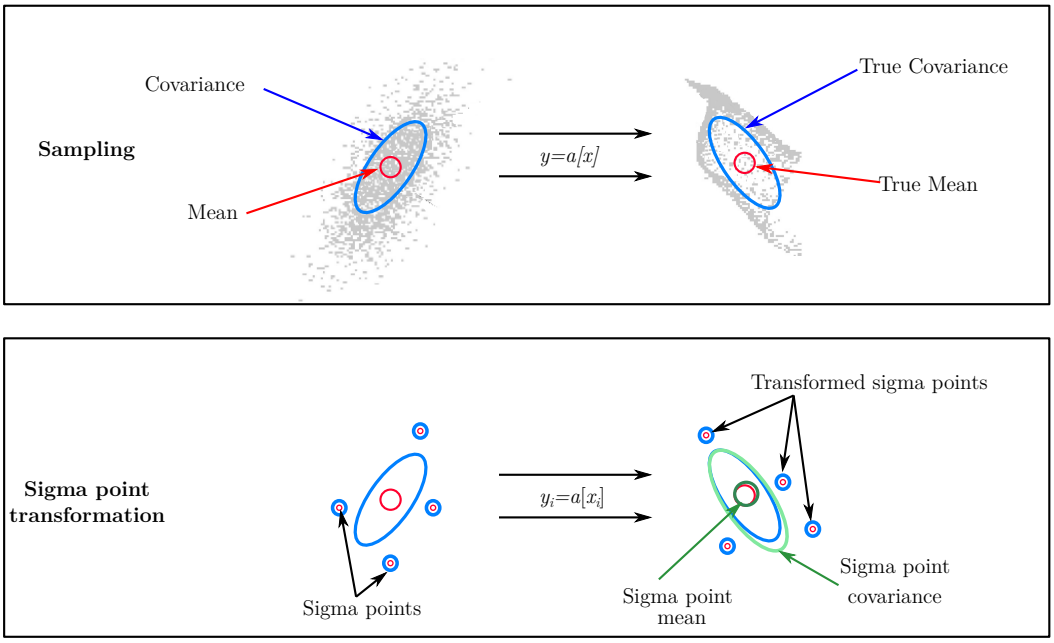



**Figure 4-6 Sampling via sigma point/unscented transform**

### 4.3.3 The extended/unscented H∞ filter

The EKF's universality is curtailed by the imposition of certain operating conditions. First, the stochastic noise terms are required to be zero mean. The expected value of the process noise, $w_k$, has to be zero; as does the expected value of the observation noise, $z_k$. The zero mean property has to be in effect for the duration of the process, as well as at each and every time instant. Second, the standard deviation of the noise terms is presumed to be known. The EKF uses the matrices $Q$ and $R$, i.e., the noise covariances, as design parameters. By implication, if we do not know $Q$ and $R$, it is not feasible to design a suitable Kalman filter (Shaked & Berman 1995). In addition, cross-correlations are required to be absent from the noise processes. The filter might diverge if these conditions are violated (Simon 2000).

An alternative to the EKF is the H∞ filter, also called the *minimax* filter, or the EH∞ (EHF) filter when extended to non-linear systems by embedding in the EKF update structure. The EH∞ filter is contrasted from the EKF in that it does not make any assumptions about the noise and minimises the worst-case estimation error via the H∞ norm (Einicke & White 1999).

The discrete-time, non-linear state space model used for the EH∞ filter is that discussed previously and used by the EKF and UKF (and indeed by all non-linear filters discussed in the present document).

The recursive form of the EHF equations is as follows:

$$\hat{x}_k^- = f(\hat{x}_{k-1}, u_k) \tag{4-16}$$

$$P_k^- = F_k P_{k-1} F_k^T + Q \tag{4-17}$$

$$\begin{aligned} \hat{x}_k &= \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-)) \\ K_k &= P_k^- H_k^T (R + H_k P_k^- H_k^T)^{-1} \end{aligned} \tag{4-18}$$

$$P_k = P_k^- - P_k^- (H_k^T I) R_{e,k}^{-1} (H_k^T I)^T P_k^- \tag{4-19}$$

Where $F_k$ and $H_k$ are the Jacobian matrices of the non-linear functions *f* and *h* computed at estimate $\hat{x}_k^-$. *I* is an identity matrix of corresponding dimension. The matrix $R_{e,k}^{-1}$ is given by:

$$R_{e,k} = \begin{bmatrix} R & 0 \\ 0 & -\gamma I \end{bmatrix} + \begin{bmatrix} H_k \\ I \end{bmatrix} P_k^- \begin{bmatrix} H_k^T & I \end{bmatrix} \quad \textbf{(4-20)}$$

Where *γ* is a scalar parameter, with *λ>0*.

The EHF is configured for non-linear systems with white noise processes. As the EHF takes on certain facets of the EKF, some of the disadvantages associated with the EKF inevitably 'rub off' on it. For example, the smoothness and mildly non-linear character of the non-linear functions that can be estimated and the computational errors due to the Jacobian matrices remain a challenge.

The unscented transform is an elegant way to approximate the filtering distribution by a Gaussian density, instead of linearising the non-linear functions as does the EKF. As already mentioned above, it is known that unscented transform-based estimates are accurate to the second order of the Taylor series expansion, as opposed to the first-order accuracy for the EKF, and hence attain greater accuracy than the EKF. Additionally, they are of approximately the same order of computational complexity. In view of its success in coping with non-linear state estimation problems, the unscented transform technique has been recently combined with the H∞ filter, producing the unscented H∞ (UHF) filter (Li & Jia 2010). On a side note, the choice of the disturbance tolerance level has not yet been investigated for this filter. This level cannot in general be predefined and should be chosen with care to ensure the existence of the H∞ filter, especially for different engineering applications.

The sigma points for the UHF are implemented in the same way as for the UKF, as are the predicted state, predicted measurement and their respective covariances, as detailed in expressions (3.10)–(3.15). The consolidated (filtered) estimates of the state and error covariance are evaluated as follows.

$$\hat{x}_k = \hat{x}_k^- + P_{xz}(R + P_z)^{-1}(z_k - \hat{z}_k) \qquad \textbf{(4-21)}$$
$$P_k = P_k^- - P_z\begin{bmatrix} P_{xz} & P_k^- \end{bmatrix} R_{e,k}^{-1} \begin{bmatrix} P_{xz} & P_k^- \end{bmatrix}^T$$

Where:

$$R_{e,k} = \begin{bmatrix} R + P_z & P_{xz}^T \\ P_{xz} & -\gamma I + P_k^- \end{bmatrix} \qquad \textbf{(4-22)}$$

It should be noted that the level $\gamma$ must be set carefully to guarantee the existence of the UHF; this setting is typically application-dependent. The variable $\gamma$ can be adjusted adaptively to its minimum at each iteration thus:

$$\gamma_k = \alpha \max\{eig((P_k^-)^{-1} + H_k^T R^{-1} H_k)^{-1}\} \qquad \textbf{(4-23)}$$

Where α is a scalar larger than one.

### 4.3.4 Filter bank architecture

A single, stand-alone filter is not really suited to the task of fault isolation in non-linear systems, which, unlike FDI for linear systems, is an area which has not been very thoroughly investigated and within which there are many open problems remaining. A lone filter could conceivably be used in a fault detection task (without isolation, fault estimation, etc.), but this may be of limited use in some applications.

We therefore implement all filtering methods as a bank of filters in order to be able to not only detect but also isolate faults, in accordance with the DOS or dedicated observer scheme described earlier in this work. This architecture dedicates a specific filter to each of the six IMU sensors (three gyroscopes and three accelerometers).

This effectively breaks up the measurement vector of the stand-alone filter, resolving it into six scalar values: individual position accelerations ($a_x$, $a_y$, $a_z$) and angular rates ($p$, $q$, $r$). We can then use the dedicated filters to generate six decoupled residuals, each one monitoring the behaviour of one of the six sensors for presence of faults through the outcome of the innovation term of the respective filter coupled to that sensor. For non-linear filters, innovations are computed per expression (4.5).

The filter bank architecture is presented in Figure 4.7 overleaf.

Applying a residual evaluation function to each residual, consisting of suitably assigned thresholds, yields the binary fault vector $\boldsymbol{\varepsilon} = [\varepsilon_1\ \varepsilon_2\ \varepsilon_3\ \varepsilon_4\ \varepsilon_5\ \varepsilon_6]$. Here each $\varepsilon_i$ evaluates to a '1' or '0', indicating the detection or non-detection respectively of a fault on sensor $i$.

We are typically interested in the false positive/negative rate for a particular filter as a performance measure, as well as the accuracy (lack of divergence and prediction/measurement mismatch) and responsiveness to faults of the residuals. For the filter bank as a whole, we can average across the true/false positive/negative rates for the individual observers to arrive at a single measure.

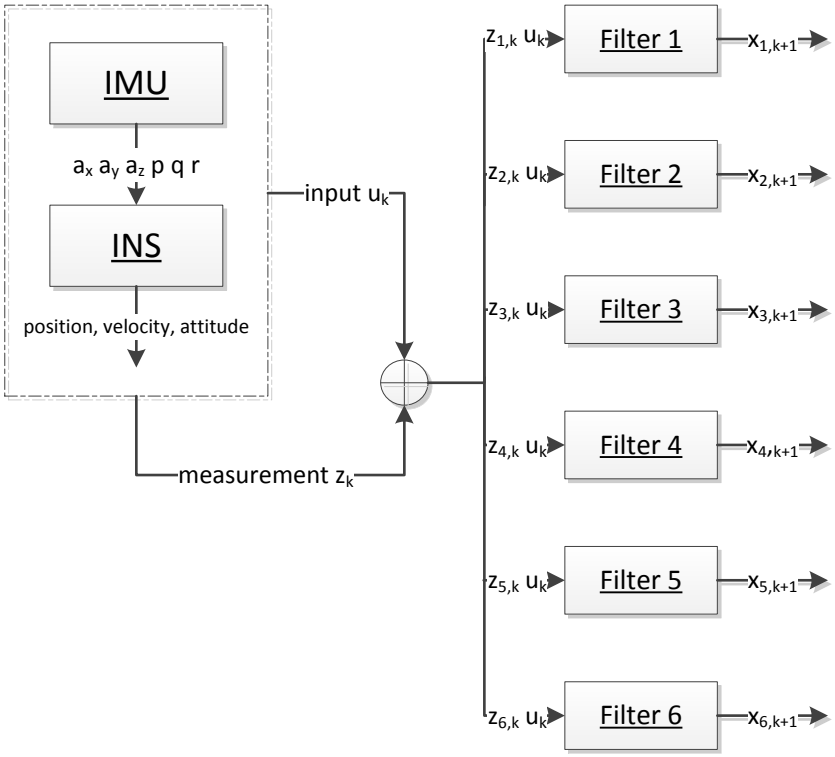



**Figure 4-7 DOS bank of filters architecture**

## 4.4 Gaussian Process Filters for FDI

### 4.4.1 Gaussian processes

This section considers the use of Gaussian process (GP) models for sequential state estimation and on to fault detection and isolation.

The Gaussian process is intimately linked to the normal, bell-shaped or Gaussian distribution (Figure 4.8) from which it takes its name. The Gaussian process is in basic terms a multivariate Gaussian distribution over functions – as opposed to a random variable or a collection of outcomes. Gaussian processes have received much attention in recent years as a powerful Bayesian non-parametric technique for regression and classification (Kocijan 2012).

Clearly, the quality of observer-based FDI is contingent upon the quality of the state estimator. In particular, an IMU process model can be learnt from sample data using Gaussian Process (GP) regression. An enhanced-GP model can also be learnt that complements a parametric or analytical model when one is available for a particular system (Ko et al. 2007b).

The enhancement consists in learning a stochastic component to the existing deterministic model and combining both to produce more accurate predictions, as well as to learn noise covariances from the data (which otherwise would need to be tuned as parameters). This enhanced model can then be embedded into an existing filter.

The state estimator that best showcases the implementation of this approach is the unscented Kalman filter, some familiarity with which has been provided already.

Our implementation is based on the work of (Ko et al. 2007b) and for GP error modelling of INS/GPS systems on the work of (Atia et al. 2011). Where we add novelty is in applying this algorithm for fault diagnosis in accordance with the filter bank architecture of Section 4.3.4. We also develop a GP-UHF filter which and apply it in the same framework (i.e. the filter bank architecture).

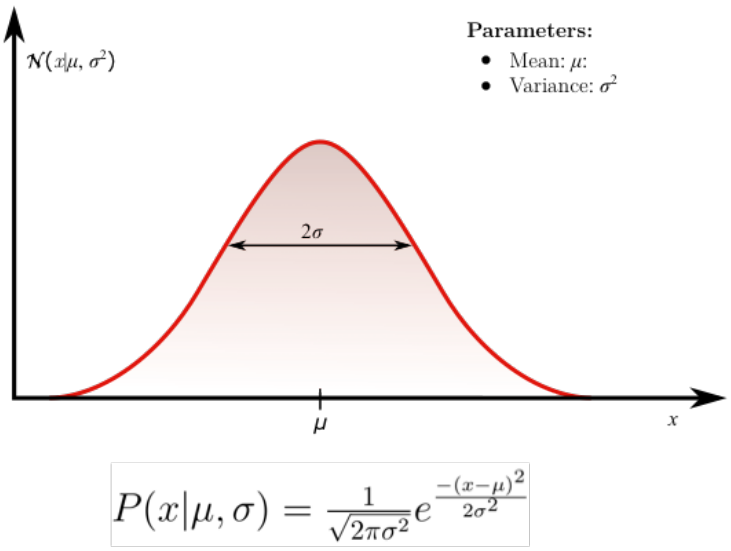



**Figure 4-8 Univariate Gaussian distribution**

The hybrid GP-UKF algorithm ought to have certain advantages over the regular UKF, as a result for example of the fact that GPs consider both the noise in the system and the uncertainty in the model. Hence, the filter can adjust its uncertainty estimate depending on how much data it has sampled in a particular region of the state space. Besides providing uncertainty estimates, other advantages of GPs include their modelling flexibility and ability to learn noise and smoothness parameters from training data (Ko et al. 2007b).

(Reece & Roberts 2010) argue that a GP can be viewed as a special case of the KF. Consequently, practitioners' intuitions about the KF, often acquired after years of experience with KF implementations, apply in equal measure to GPs.

### 4.4.2 Regression modelling with Gaussian processes

A GP is usually said to be a '*Gaussian distribution over functions*' (Rasmussen & Williams 2006). It can be regarded as an extension of a Gaussian distribution over a finite vector space to an infinite-dimensional space. As a Gaussian is fully specified by its mean and covariance matrix, so a Gaussian process is fully described by its mean and covariance function, *K*. Even though the mean and

covariance functions are infinite-dimensional, GPs predict function values over a finite set of prediction points from observed data (Ko et al. 2007b).

In regression problems, we take a sample $S = \{(x_1, y_1),(x_2, y_2),...,(x_n, y_n)\}$ in which $x_i$ denotes the $i^{th}$ input and $y_i$ is the corresponding output value. The relationship between $x_i$ and $y_i$ is formulated as:

$$y_1 = f(x_i) + \varepsilon(x_i) . \quad \textbf{(4-24)}$$

That is, a function *f* describes the relationship of the input vector $x_i$ to the true output, which, being corrupted by noise $\varepsilon(x_i)$, is measured as $y_i$. Gaussian Process Regression (GPR) is a non-parametric model that assumes

$$F = [f(x_1), f(x_2),..., f(x)_n]^T \sim N(0, K) , \quad \textbf{(4-25)}$$

where *K* is the covariance matrix.

The *n* observations in an arbitrary data set $y = \{y_1,...,y_n\}$ can be thought of as a single point sampled from some multivariate (of dimension *n*) Gaussian distribution. So this data set can be represented by a GP.

Oftentimes it is assumed that the mean of the GP is zero everywhere. In such instances the 'glue' holding together the different observations is only the covariance function, *k*(*x*,*x*'). The most common choice of covariance function is the so-called 'squared exponential',

$$k(x, x') = \sigma_f^2 \exp\left[\frac{-(x - x')^2}{2l^2}\right], \quad \textbf{(4-26)}$$

for which the greatest permissible covariance is given as $\sigma_f^2$ - functions that span a wide stretch of the y-axis should have this value high.

If $x \approx x'$, then $k(x, x')$ approaches the maximum, implying *f*(*x*) is almost perfectly correlated with *f*(*x*'). This means that for a function to look smooth, neighbouring points must closely related. If, however, *x* is distant from *x*', we would have that

$k(x,x') \approx 0)$, i.e., the two points exert a negligible effect on each other. Thus when predicting a new value, distant ones will not be taken into account. The separation effect will depend on the length parameter, *l*, so (4.26) is quite flexible.

Regression can be thought as function fitting or searching for a model to approximate *f*(*x*). The first step in GP regression is to compute the covariance function (which is a measure of the 'closeness' between inputs) over all the points as in:

$$K = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \cdots & k(x_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \cdots & k(x_n,x_n) \end{bmatrix} \quad \textbf{(4-27)}$$

$$K* = [k(x*,x_1) \;\; k(x*,x_2) \;\; \cdots \;\; k(x*,x_n)] \qquad K** = k(x*,x*). \quad \textbf{(4-28)}$$

As the main assumption in GP modelling is that the data is represented as a sample from a multivariate Gaussian distribution, we have that

$$\begin{bmatrix} y \\ y* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K & K*^T \\ K* & K** \end{bmatrix}\right). \quad \textbf{(4-29)}$$

We want to know the conditional probability $p(y*|\boldsymbol{y})$, i.e., given the data, how likely is the prediction for *y**, assuming the probability is drawn from a Gaussian distribution?

As we assume non-correlated noise from one sample to the next in the data, so the noise term only increases the values along the diagonal of *K*, resulting in a new covariance for noisy observations having the form

$$V(x,x) = K(x,x) + \sigma^2 I \quad \textbf{(4-30)}$$

where ***I*** is the identity matrix and $\sigma^2$ is a hyper parameter representing the noise variance.

To compute the Gaussian process posterior distribution at some new input value, x*, we initially consider the joint distribution of the observed data (consisting of *x* and respective values *y*) modified by *x** and *y**,

$$p\left(\begin{bmatrix} y \\ y* \end{bmatrix}\right) = N\left(\begin{bmatrix} \mu(x) \\ \mu(x*) \end{bmatrix}, \begin{bmatrix} K(x,x) & K(x,x*) \\ K(x*,x) & k(x*,x*) \end{bmatrix}\right) \quad \textbf{(4-31)}$$

where **K**(**x**,*x**) is the column vector formed from k($x_1$,*x**),…,k($x_n$,*x**) and **K**(*x**,**x**) is its transpose.

After some rearranging, we discover that the posterior distribution over *y** is Gaussian with mean and variance given by

$$m* = \mu(x*) + K(x*,x)K(x,x)^{-1}(y - \mu(x)), \quad \textbf{(4-32)}$$

$$\sigma_*^2 = K(x*,x*) - K(x*,x)K(x,x)^{-1}K(x,x*). \quad \textbf{(4-33)}$$

This result can be extended to evaluate the Gaussian process at a different set of points to our inputs, *x**, and to evaluate the posterior distribution of y(*x**). This latter is easily obtained by augmenting the equations above and using known techniques for multivariate Gaussians. The posterior mean and variance are evaluated as

$$p(y*) = N(m*, C*). \quad \textbf{(4-34)}$$

Where,

$$m* = \mu(x*) + K(x*,x)K(x,x)^{-1}(y(x) - \mu(x)), \quad \textbf{(4-35)}$$

$$C = K(x*,x*) - K(x*,x)K(x,x)^{-1}K(x*,x). \quad \textbf{(4-36)}$$

### 4.4.3 1-D GP learning example

This example illustrates the use of a Gaussian process model for 1-D regression problems. That is, a problem where the input and output are both scalar. The GP inertial measurement unit process models we embed in GP filters are 6-D; that is, each one has 6-inputs. Therefore, it is not possible to visualise them, as the data occupies a 7-dimensional space (6 inputs and one output). However, we can do this for regression problems that are up to 2-dimensional.

This example is based on training data consisting of 20 points drawn from a Gaussian process. A plot of the points is contained in Figure 4.9 below.

We next compute the predictions using a Gaussian process at 201 test points evenly distributed in the interval [-7.5, 7.5].

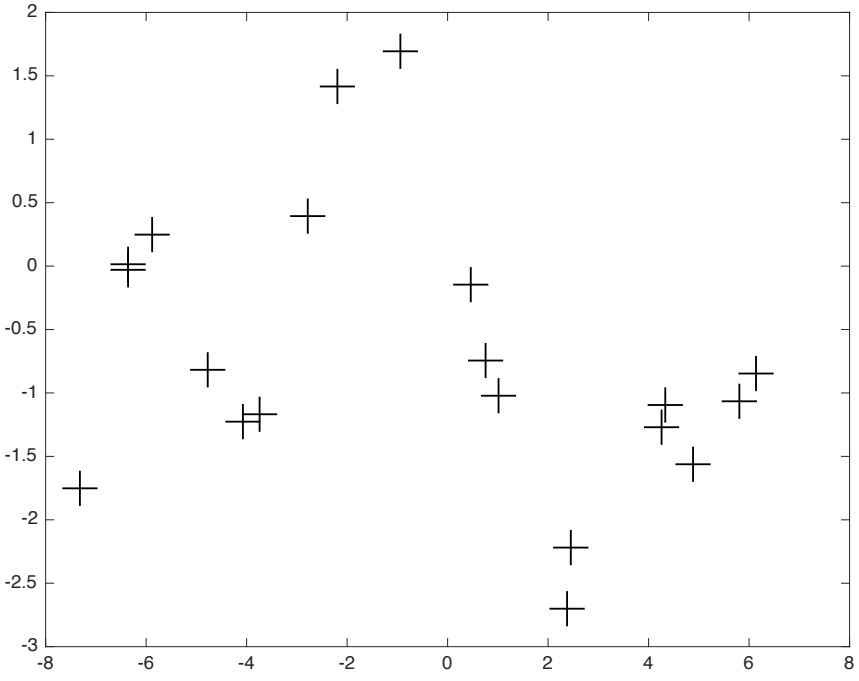



**Figure 4-9 GP example training data**

In this simple example, we use a covariance function whose functional form matches the covariance function which was used to generate the data. In this case, this was a squared exponential (SE) covariance term with added independent noise.

The squared exponential covariance function is equivalent to the Gaussian kernel detailed in Chapter 3. It should be noted that a sum of two or more GP covariance functions is also a GP covariance. So the noise and squared exponential can be summed into a single covariance.

We have now specified the functional form of the covariance function but we still need to specify values of the parameters of these covariance functions, also called hyperparameters. In this case we have 3 hyperparameters. These are: a characteristic length-scale for the squared exponential (SE) contribution, a signal magnitude for the SE contribution, and the standard deviation of the noise. The logarithm of these hyperparameters, $\theta$, is specified as follows:

$$\log(\theta) = \log\begin{bmatrix}1.0\\1.0\\0.1\end{bmatrix} \quad \textbf{(4-37)}$$

thus specifying a unit length scale, unit magnitude and a noise variance of 0.01 (corresponding to a standard deviation of 0.1). For numerical reasons, the hyperparameters are converted to their logarithmic values before being set in the GP model.

We then use Gaussian process regression to make predictions for the test points, and we plot these by showing the predictive mean (solid line) and two standard error (95% confidence), noise free, point-wise error-bars as shown in Figure 4.10.

Note that since we are interested in the distribution of the function values and not the noisy examples, we subtract the noise variance, which is stored in hyperparameter number 3, from the predictive variance $\sigma^2$. Alternatively, the mean and error-bar range can be displayed in gray-scale.

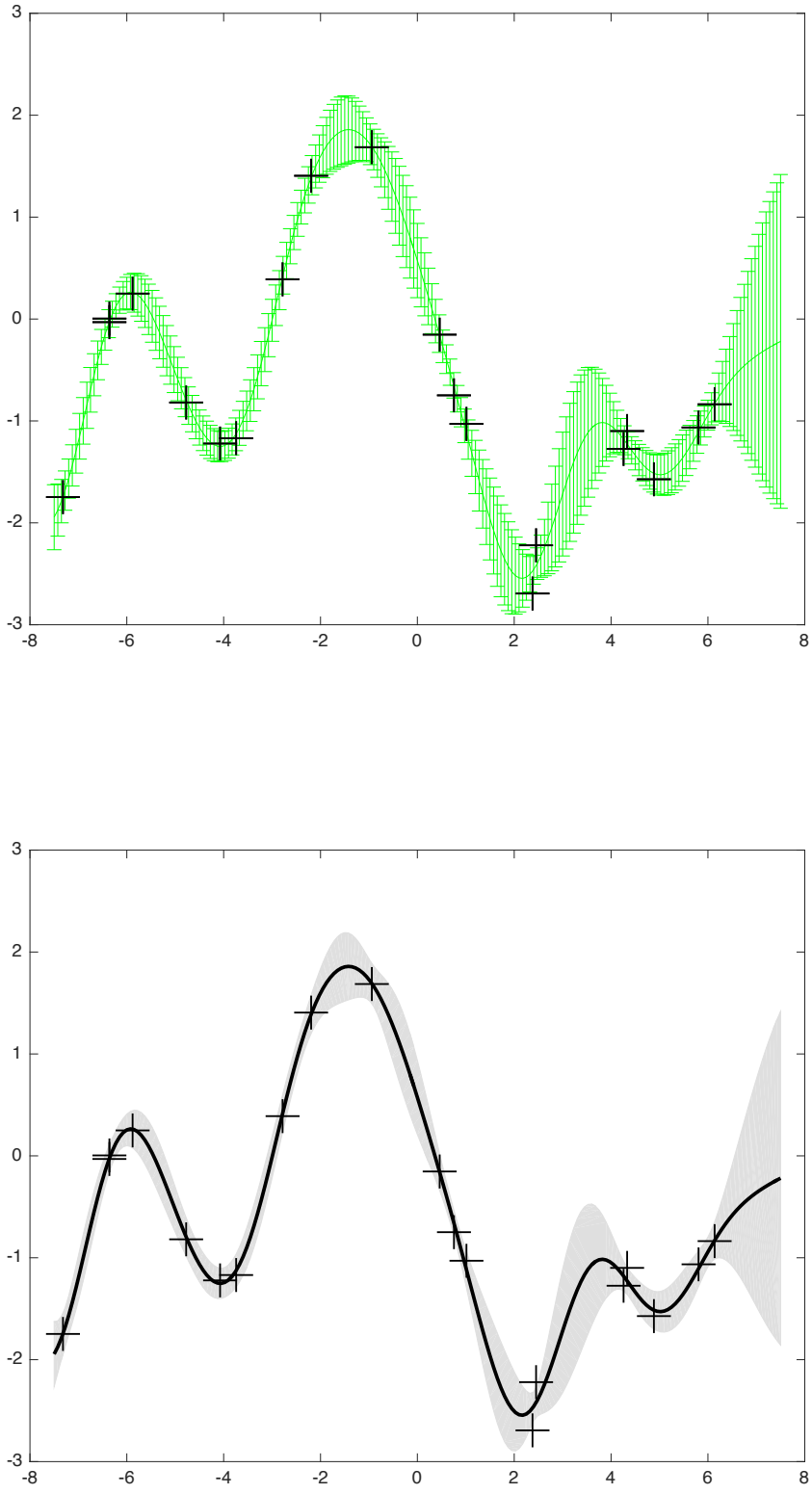



**Figure 4-10 GP predictions of test points using SE covariance**

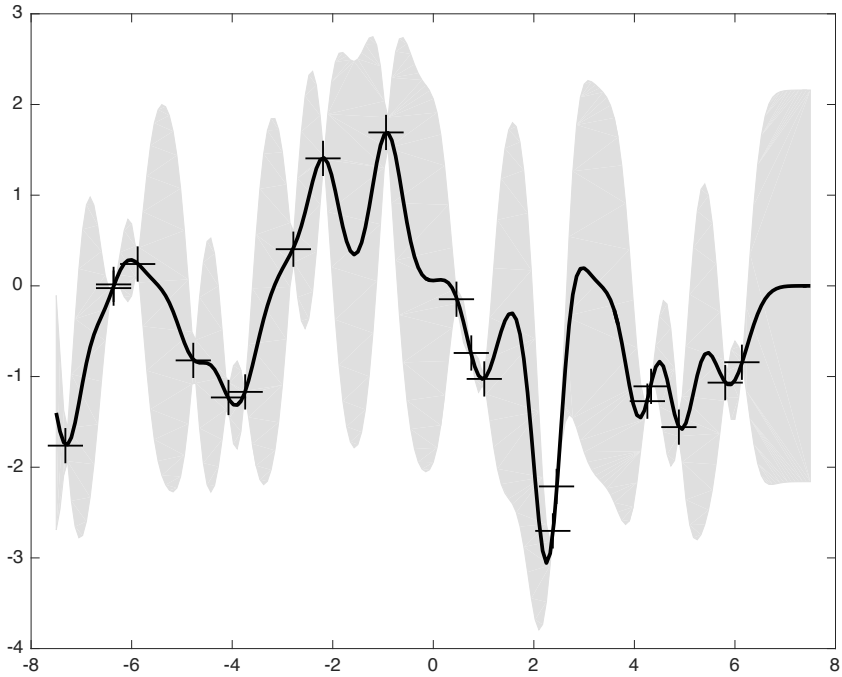



**Figure 4-11 GP regression using new set of hyperparameters**

We next investigate changing the hyperparameters to have the values:

$$\log(\theta) = \log\begin{bmatrix} 0.3 \\ 1.08 \\ 5e-5 \end{bmatrix} \tag{4-38}$$

The length-scale is now shorter (0.3) and the noise level is much reduced, so the predicted mean almost interpolates the data points. Notice that the error bars grow rapidly away from the data points due to the short length-scale. The resulting plot is shown in Figure 4.11.

Alternatively, we can change the hyperparameters to have the values:

$$\log(\theta) = \log\begin{bmatrix} 3.0 \\ 1.16 \\ 0.89 \end{bmatrix} \tag{4-39}$$

The length-scale is now longer than initially and the noise level is higher. Thus the predictive mean varies more slowly than before, as seen in Figure 4.12.

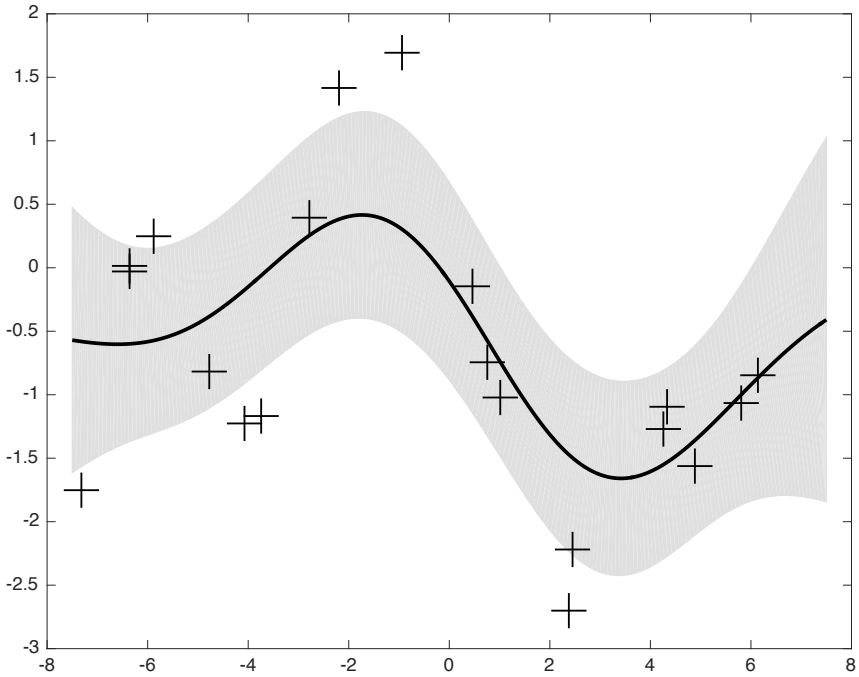



**Figure 4-12 GP regression using another set of hyperparameters**

The hyperparameters can also be learned by maximising the marginal likelihood. The hyperparameters are initialised to:

$$\log(\theta) = -\begin{bmatrix}1\\1\\1\end{bmatrix} \tag{4-40}$$

We use 100 function evaluations and end up with the following learned hyperparameters:

$$\exp\left[\log(\theta)\right] = \begin{bmatrix}1.3659\\1.5462\\0.1443\end{bmatrix} \tag{4-41}$$

Note that the hyperparameters learned here are close, but not identical to the parameters [1.0, 1.0, 0.1] used when generating the data. The discrepancy is partially due to the small training sample size, and partially due to the fact that we only get information about the process in a very limited range of input values.

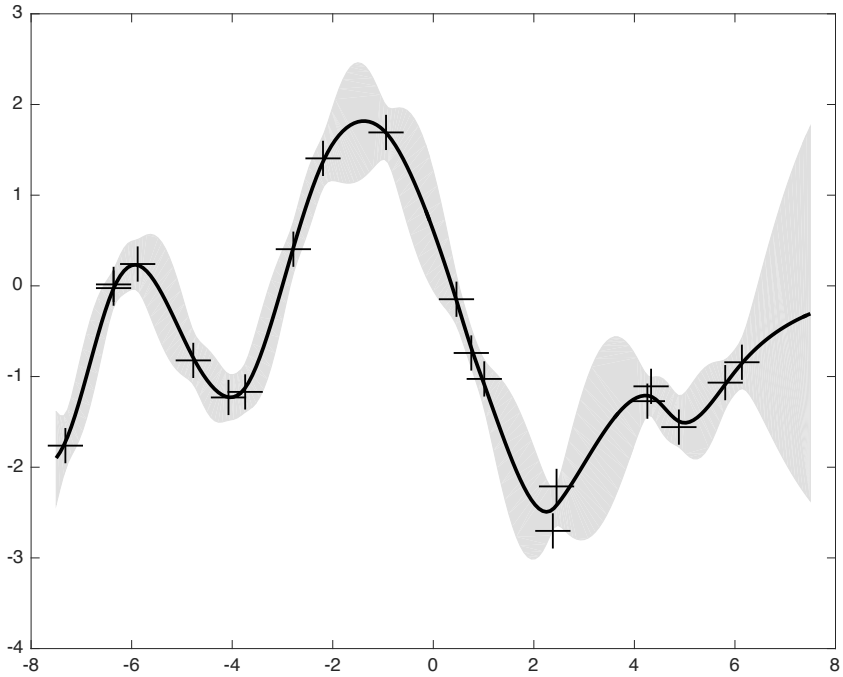



**Figure 4-13 GP regression using learned hyperparameters**

Repeating the experiment with more training points distributed over wider range leads to more accurate estimates. Finally, we compute and plot the predictions using the learned hyperparameters (Figure 4.13), showing a reasonable fit, with a relatively tight confidence region.

Note, that so far in this example we have used the same functional form of the covariance function as was used to generate the data. In practice things are seldom so simple, and one may have to try different covariance functions. Next we explore how a Matern form covariance function, with a shape parameter of 3/2, does on the test data. Hyperparameters are again learned after first being initialised to (4.27).

Comparing the value of the marginal likelihoods for the two models gives -15.6 for SE and -18.0 for Matern, shows that the SE covariance function is approximately *exp*(18.0-15.6)=11 times more probable than the Matern form for these data.

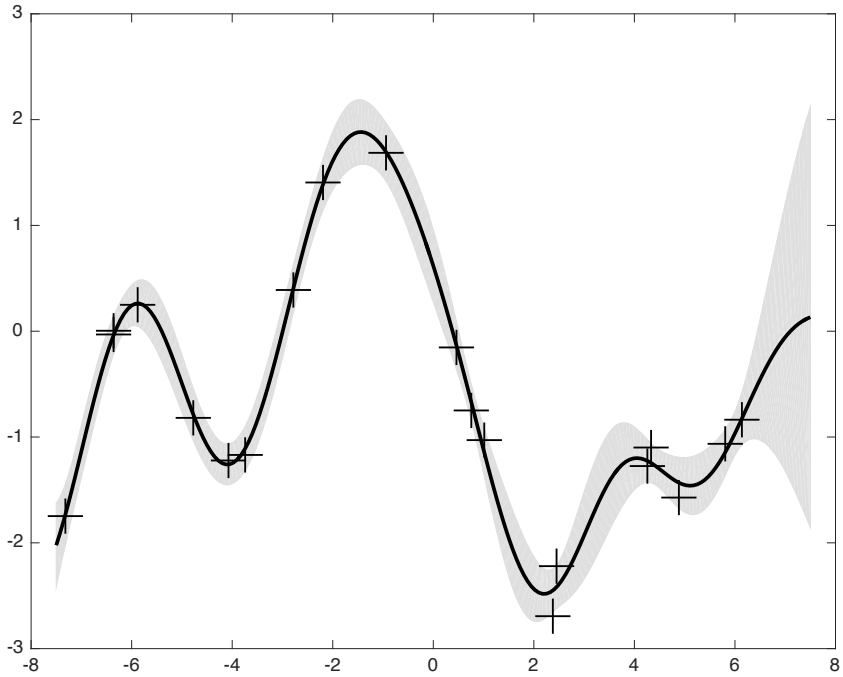



**Figure 4-14 GP predictions of test points using Matern form covariance**

This is in agreement with the data generating process. The predictions from the Matern-based model are plotted in Figure 4.14.

It is notable that the uncertainty grows more rapidly in the vicinity of the data points, reflecting the property that the sample paths for the Matern class of functions with a shape parameter of 3/2 don't have second derivatives (and are thus much less smooth than the SE covariance function).

### 4.4.4 Learning GP IMU process models

GP Bayesian filters such as the GP-EKF and GP-UKF replace analytical or parametric process and observation models with non-linear, non-parametric regression models based on GPs. That is, Gaussian processes are used for learning such models from training data (Ko & Fox 2008).

The input to the IMU model is the current vehicle state according to predictions made by the quadrotor UAV dynamic model and the output is the IMU sensor error. The GP learns a mapping between these two datasets.

The input data sets dimensions are the six states encompassing the three attitude Euler angles and three vehicle velocities. Only vehicle velocity and attitude are included in the input data set because their values are bounded, whereas position has no bounded range, which makes function approximation by GPs infeasible.

The output data set is one dimensional and consists of the gyroscope and accelerometer deviations. The errors in the IMU navigation information are determined by taking the difference between the IMU readings and a ground truth provided by the quadrotor dynamic model, which provides the analytical redundancy. The way that these are simulated, the gyroscopes along each of the three axes have the same error characteristics as one another, and the same holds for the accelerometers. Thus there are two GP models enclosed in the filter process model: one that models accelerometer error and the other gyroscope error. We do not need to learn a GP observation model, since the observation model is straightforward and can be summarised as an identity or unit matrix.

Our GP error modelling approach is based on (Atia et al. 2012), although they model a MEMS-based INS, not IMU, deviations for the purposes of bridging GPS outages. In terms of the GP Bayesian filters, our approach follows the work of (Ko et al. 2007b; Ko et al. 2007a)

To learn either of our GP models, the procedure followed is the same. We first check the scaling of the input and target variables. We might be concerned if the standard deviation is very different for different input dimensions. In the event, it was, so we had to carry out rescaling.

We use Gaussian process regression with a squared exponential covariance function, and allow a separate length-scale for each input dimension, as in (Rasmussen & Williams 2006). These length-scales (and the other

hyperparameters $\sigma_f$ and $\sigma_n$) are adapted by maximising the marginal likelihood w.r.t. the hyperparameters.

We next move on to training the GP by optimising the hyperparameters. The hyperparameters are first initialised to 0.

We can plot the negative marginal likelihood as a function of the number of line-searches of the optimisation routine. Figure 4.15 displays plots for both the gyro and accelerometer GP error models.

To assess the training process, we use the following metrics: the mean squared error and the mean predictive log likelihood.

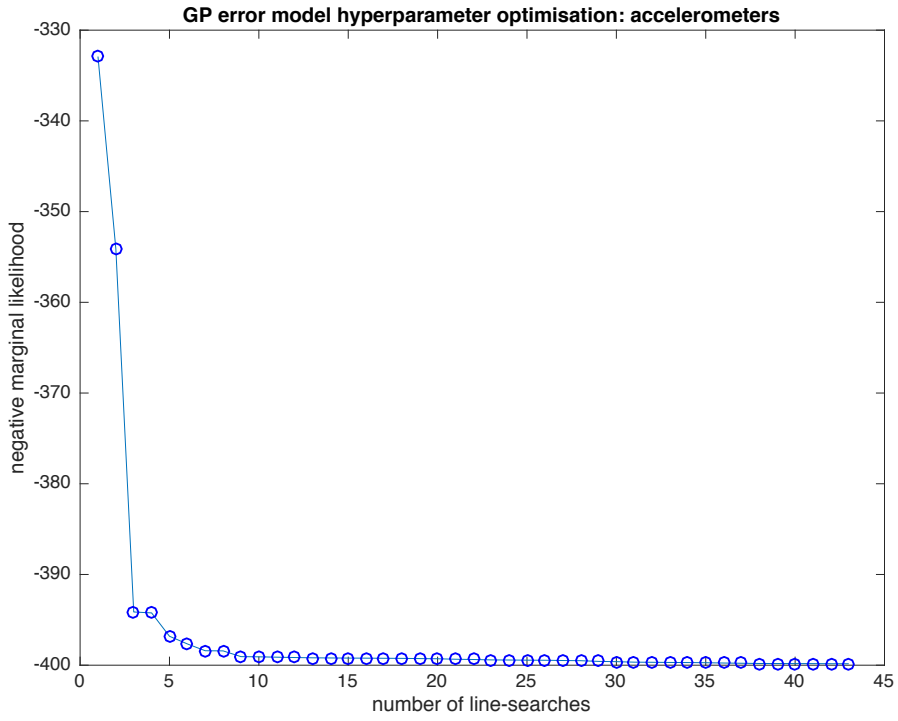
For the GP error model of the accelerometers:

- The mean squared error is 0.0012
- and the mean predictive log likelihood is 1.9125.

For the GP error model of the gyroscopes:

- The mean squared error is 0.0005
- and the mean predictive log likelihood is 2.3929.

The training set used to build the GP models contains 200 points taken from a single data set of 1000 points.



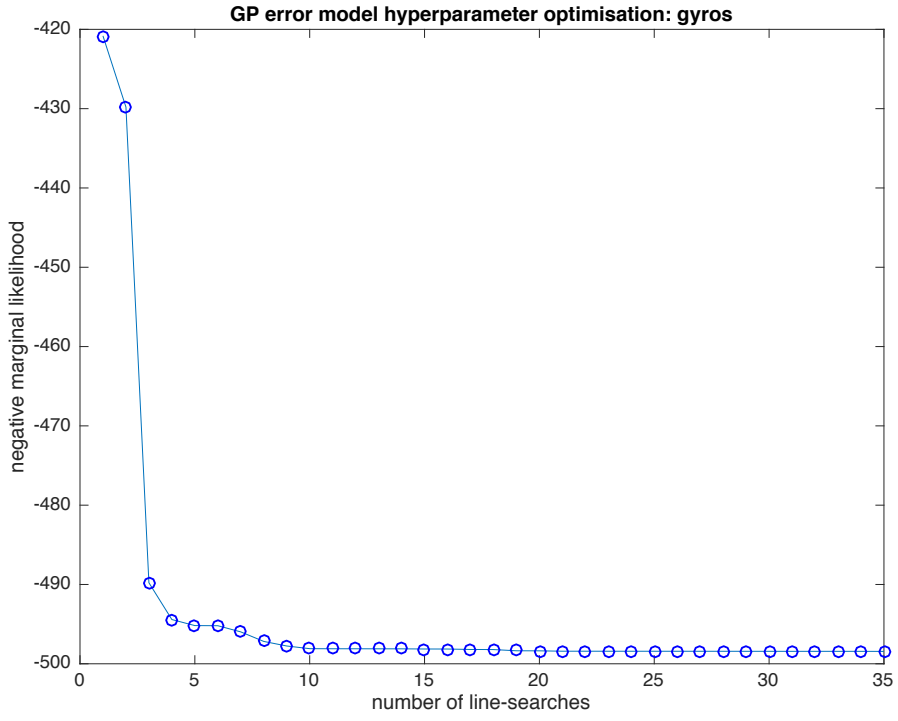




**Figure 4-15 Negative marginal likelihood as a function of no. of line-searches**

### 4.4.5 GP-enhanced UKF/UHF

The training data for each GP is a set of input-output mappings. The transition model relates the state and control ($x_k$,$u_k$) to the state transition $\delta x_k = x_{k+1} - x_k$. The *k+1* state can be found by summing the *k* state with the state transition. The GP transition function (i.e. process model is the following

$$\mathbf{x}_k = \mathrm{GP}_{\mu}^{f}\left([\mathbf{x}_{k-1}, \mathbf{u}_k], \mathrm{D}_f\right) + \boldsymbol{\varepsilon}_k \qquad \textbf{(4-42)}$$
$$\boldsymbol{\varepsilon}_k = \mathcal{N}(0, \mathrm{GP}_{P}^{f}\left([\mathbf{x}_{k-1}, \mathbf{u}_k], \mathrm{D}_f\right))$$

Where the $x_k$ expression is the mean prediction of the Gaussian process model and the second expression is the stochastic prediction by the GP. The only difference between this implementation and the standard UKF/UHF as described in Section 4.3 is that instead of passing the sigma points through *f*(x), they are passed through (4.29) and $\varepsilon_k$ is used to generate $Q_k$.

The transition function training data set is defined as follows:

$$\mathrm{D}_f = \{(X, U), X'\} \qquad \textbf{(4-43)}$$

Where *X* is the matrix of ground truth states, and $X' = [\delta x_1, \delta x_2, ..., \delta x_k]$ is a matrix containing transitions made from those states after applying the controls stored in *U* (Ko & Fox 2008; Ko et al. 2007b).

By incorporating GP regression, GP-UKFs are able to learn enhanced models, as well as noise processes, from training data. In addition, the noise models of the filter adapt automatically to the system state in relation to the density of training data around the current state. So that if less training data is available, the GP-UKF returns higher uncertainty estimates.

By embedding a Gaussian process process model into the unscented H-infinity filter, we have proposed a new variant Gaussian process Bayesian filter, which previously did not exist, and can bring to bear some of the advantages of the H-infinity update structure on the GP Bayesian filter formulation, such as robustness.

## 4.5 Experimental Validation

### 4.5.1 Isolation results

The experimental trials described in this section were carried out in the same way, under the same conditions and using the same data as those for the kPCA algorithm of Chapter 3. Using the same setup throughout should allow for a fair and balanced comparison between the proposed algorithms.

As in the last chapter, results are gathered over a series of ten data sets, producing ten replications of the same experiment, i.e. each time applying the same simulated faults on a different data set. Residual plots in Figures 4.16-17 provide just a snapshot, visualising the residuals for a single data set, typically the first in the sequence.

Results tables 4.1-4.3 reveal some differences between the Bayesian filters and kPCA: the average isolation rate for both filters is slightly lower than Gaussian kPCA, even as the average AUC is around 20 points higher. It should be recalled that the area under an ROC curve is a performance metric that measures performance across the range of all possible thresholds – each point on the ROC curve represents a different threshold instantiation. The threshold applied is chosen optimal to some performance criterion – in our case minimising the false positive rate; this implies that we select the optimal point as the point on the ROC curve nearest the top-left corner of the ROC graph. Hence the explanation for the lopsided results between kPCA and GP-UKF/UHF: kPCA demonstrates superior isolation performance at its operating point (i.e. the threshold selected as optimal), whilst GP-UKF/UHF best it across the entire operating envelope.

The UHF filter performs marginally better than the UKF. This is mirrored in both the average true positive rate and AUC score being higher for the GP-UHF, whilst the optimised thresholds are seen to be lower. Though lower thresholds cannot be taken as an absolute measure of isolation performance, they certainly are a fair indication. Thus the kPCA (RBF) average threshold is far lower than those for GP-UKF/UHF and this matches the isolation results.

**Table 4-1 Area under curve and optimal thresholds**

| | GP_UKF_AUC | GP_UHF_AUC | GP_UKF_Thresh | GP_UHF_Thresh |
|---|---|---|---|---|
| **X-axis accelerometer** | 0.8 | 0.83 | 0.87 | 0.92 |
| **Y-axis accelerometer** | 0.83 | 0.84 | 2.41 | 1.66 |
| **Z-axis accelerometer** | 0.83 | 0.82 | 2.48 | 1.94 |
| **X-axis gyro** | 0.86 | 0.86 | 1.7 | 1.72 |
| **Y-axis gyro** | 0.86 | 0.87 | 1.92 | 1.65 |
| **Z-axis gyro** | 0.85 | 0.86 | 0.93 | 1.52 |
| **Average** | 0.84 | 0.85 | 1.72 | 1.57 |

**Table 4-2 GP-UKF isolation rates**

| | GP_UKF_TruePos | GP_UKF_FalsePos | GP_UKF_TrueNeg | GP_UKF_FalseNeg |
|---|---|---|---|---|
| **X-axis accelerometer** | 0.59 | 0.13 | 0.87 | 0.41 |
| **Y-axis accelerometer** | 0.2 | 0.01 | 0.99 | 0.8 |
| **Z-axis accelerometer** | 0.19 | 0.01 | 0.99 | 0.81 |
| **X-axis gyro** | 0.44 | 0.04 | 0.96 | 0.56 |
| **Y-axis gyro** | 0.41 | 0.02 | 0.98 | 0.59 |
| **Z-axis gyro** | 0.69 | 0.14 | 0.86 | 0.31 |
| **Average** | 0.42 | 0.06 | 0.94 | 0.58 |

**Table 4-3 GP-UHF isolation rates**

| | GP_UHF_TruePos | GP_UHF_FalsePos | GP_UHF_TrueNeg | GP_UHF_FalseNeg |
|---|---|---|---|---|
| **X-axis accelerometer** | 0.65 | 0.13 | 0.87 | 0.35 |
| **Y-axis accelerometer** | 0.41 | 0.03 | 0.97 | 0.59 |
| **Z-axis accelerometer** | 0.3 | 0.01 | 0.99 | 0.7 |
| **X-axis gyro** | 0.48 | 0.02 | 0.98 | 0.52 |
| **Y-axis gyro** | 0.52 | 0.02 | 0.98 | 0.48 |
| **Z-axis gyro** | 0.49 | 0.07 | 0.93 | 0.51 |
| **Average** | 0.47 | 0.05 | 0.95 | 0.53 |

### 4.5.2 Residuals and ROC curves

The Gaussian process filter residuals in Figures 4.16-17 are both qualitatively and quantitatively different from those of kPCA, the peaks are wider and less pronounced and there is some evidence of overshoot. Which is something typically associated with Kalman-type filters, because the measurement and prediction can get out of phase, which then causes residuals to spike after a lag.

The ROC graphs (Figures 4.18-23) show the GP-UHF generally better, although in some regions of the plots, the GP-UKF curve overtakes the GP-UHF in being further to the left. Clearly, the closer a curve is to the horizontal ceiling and vertical left side of the ROC graph, the higher the AUC is going to be.

Overall the GP-UHF edges out its counterpart – the GP-UKF, which may have something to do with the fact that the UHF filter is more adept at dealing with coloured noise, which there may well be some of in the IMU error terms. Indeed, the UHF is specially designed for this purpose: to deal with systems exhibiting non-white, non-Gaussian noise.

It is also noteworthy that the average AUC values for both filters at 0.84 and 0.85 are extremely high, suggesting that the GP process models in the filters have generalised well to the test data sets and are capable of decent extrapolation. The GPs are trained on a 200-point subset from a single data set. The data set that the training set points are drawn from is by default the first in the sequence of ten. Thus the next nine data sets can be treated exclusively as test sets.

During the practical trials, it became apparent that the Gaussian process models appear to possess better generalisation capability than kPCA, since they require fewer and less widely distributed training points than the latter, although this comes at the cost of a much higher computational overhead for Gaussian process algorithms. Indeed, the GP-UKF and GP-UHF are perhaps an order of magnitude slower than their non-GP equivalents.

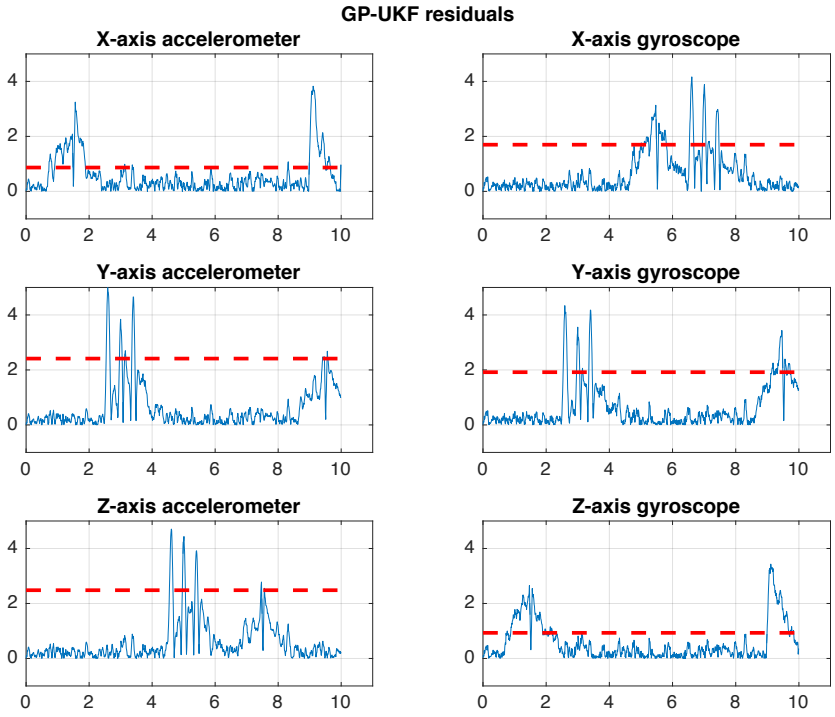



**Figure 4-16 GP-UKF residuals**

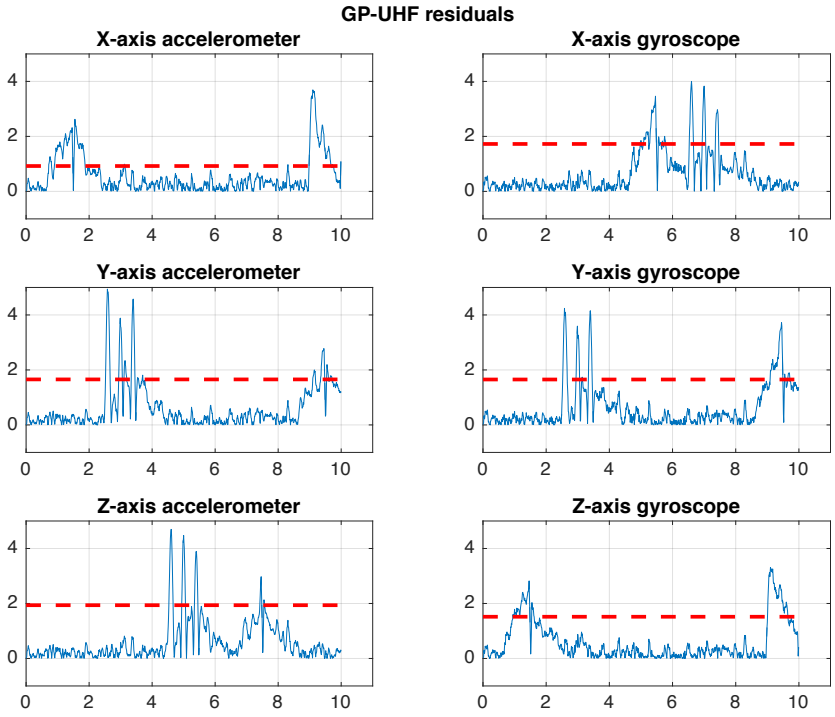



**Figure 4-17 GP-UHF residuals**

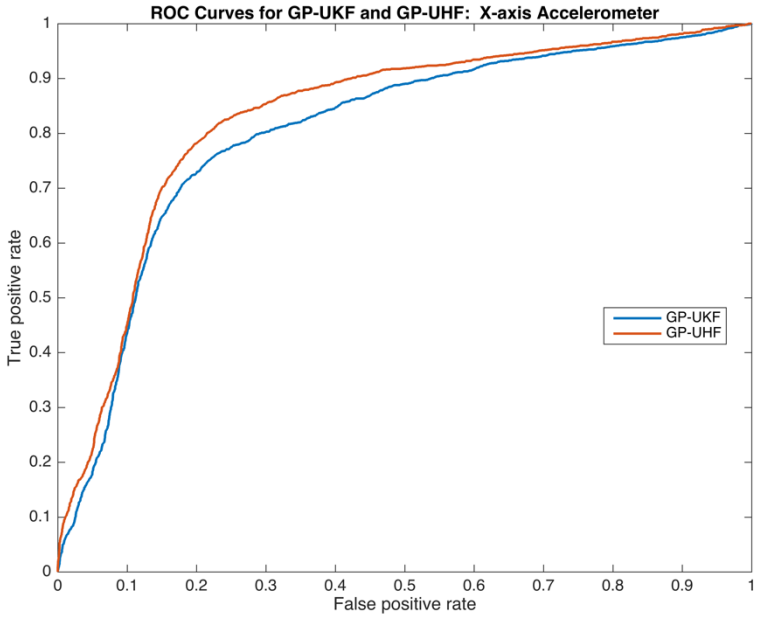



**Figure 4-18 X-axis accelerometer ROC curves**

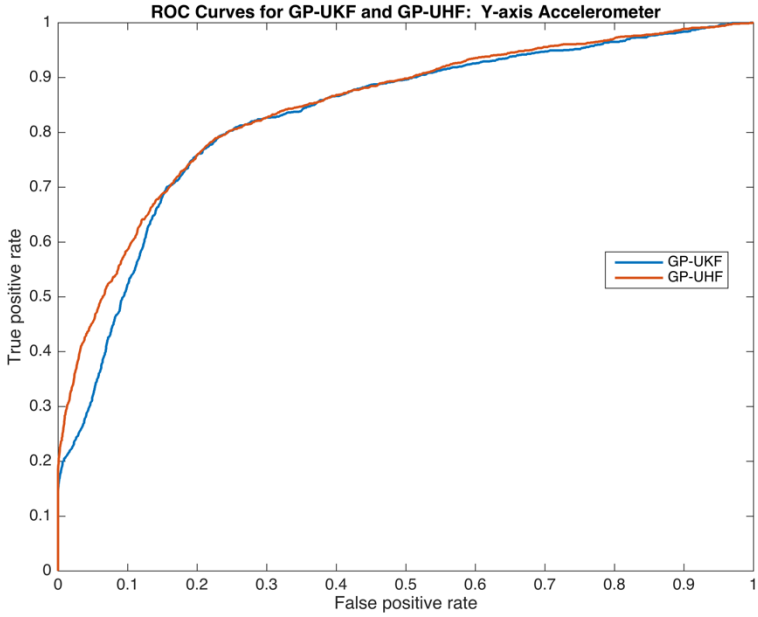



**Figure 4-19 Y-axis accelerometer ROC curves**

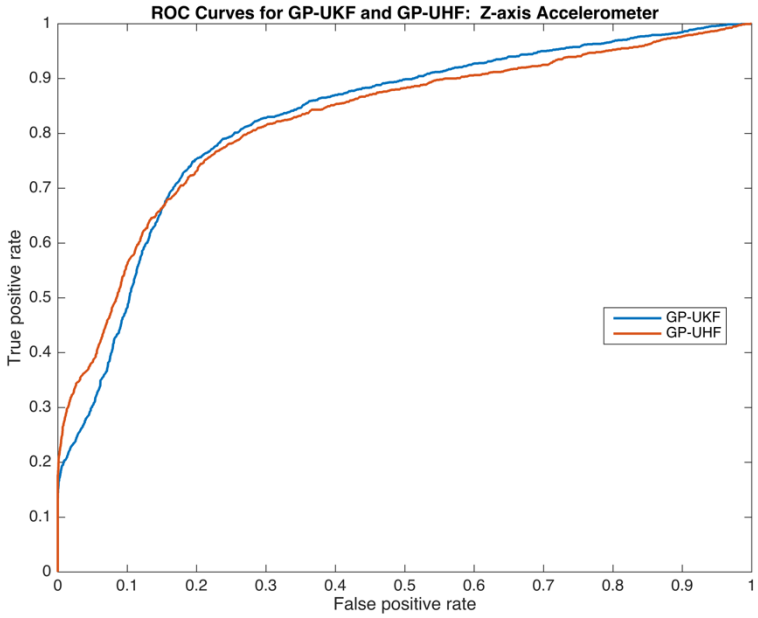



**Figure 4-20 Z-axis accelerometer ROC curves**

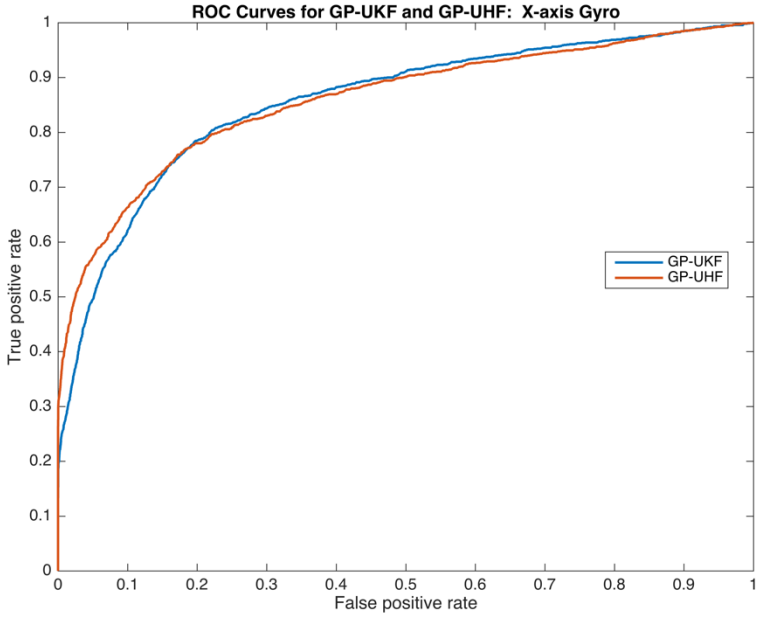



**Figure 4-21 X-axis gyro ROC curves**

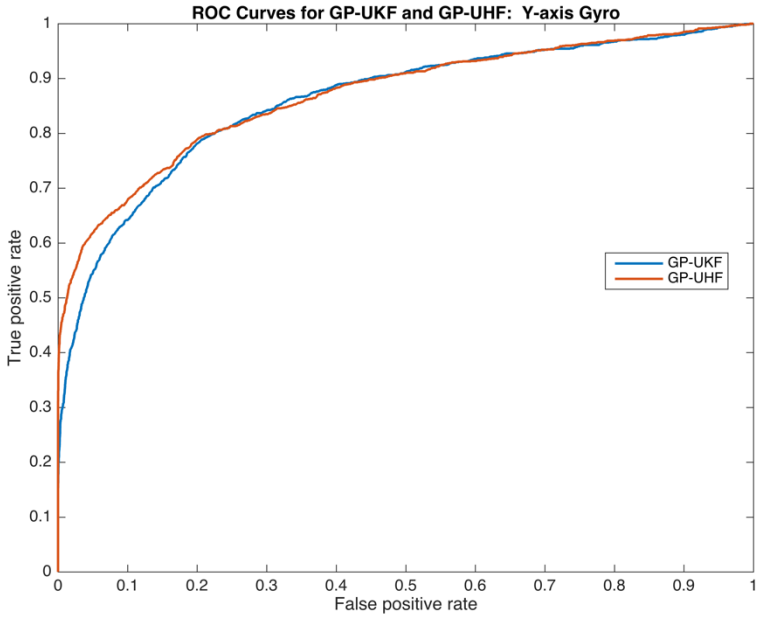



**Figure 4-22 Y-axis gyro ROC curves**

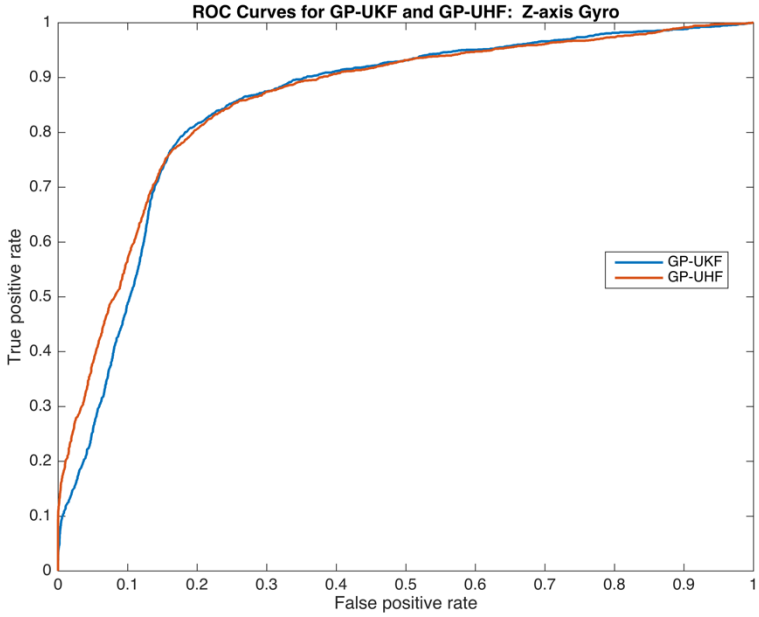



**Figure 4-23 Z-axis gyro ROC curve**

## 4.6 Conclusion

This chapter has presented the use of stochastic observers in the form of Bayesian filters for diagnosing IMU sensor faults. First the theory behind the Kalman filter was developed, leading on to an exposition of non-linear Kalman filter extensions: the extended and unscented Kalman filters. The theory behind Gaussian process regression is subsequently woven into the discussion. The main contributions of the chapter are next developed: the adaptation of an unscented Kalman filter featuring a Gaussian process transition model to the dedicated observer scheme, or DOS, for fault diagnosis by means of banks of observers; and the introduction of an unscented H-infinity filter with Gaussian process transition model. Whereas the GP-UKF is not a new filter, its adaptation to perform fault diagnosis is novel. The GP-UHF, on the other hand, is an entirely new filter design. Even though a UHF filter has been proposed previously, it did not incorporate Gaussian process models, nor was it applied to a fault diagnosis task.

The performance statistics and ROC curves generated from the IMU case study indicate that the GP-UHF has a slight edge performance-wise to the GP-UKF.

# 5 GAUSSIAN PROCESS EXTENSIONS: DISTRIBUTED FILTERING AND SENSOR FUSION

Sensor fusion methods find use across a wide number of applications - in areas ranging from target tracking to vehicle navigation, image processing, and signal processing in general. A working definition could be stated as: how to optimally obtain useful information from multiple sensor streams.

Sensor fusion is most useful in multi-sensor systems with complementary sensors, such as an INS/GPS system. In terms of fault detection, this opens up additional avenues to further layers of analytical redundancy and fault mitigation.

A popular method for fusing multiple sensor signals is by application of an information filter – a key design framework in distributed estimation. The Kalman filter, and its variants presented in the previous chapter, is clearly amenable to the processing of multi-sensor data streams, and is often resorted to in practice. However, the KF can incur a high computational overhead in distributed settings. In contrast, information filtering, which is fundamentally a Kalman filter formulated using the inverse of the covariance matrix, posits some advantages over the standard Kalman filter. Namely, its update structure is computationally less demanding and complicated than that of the standard KF, and is more easily initialised (Lee 2008). The information form is also known as the inverse covariance form of the Kalman filter (Bar-Shalom et al. 2001).

In this chapter, we extend the GP-UKF algorithm detailed in Chapter 4 to distributed settings by introducing two Gaussian process-enhanced unscented information filter (UIF) variants: the Gaussian process centralised unscented information filter (GP-CUIF) and the Gaussian process federated unscented information filter (GP-FUIF). To the best of the author's knowledge, a filter design conflating Gaussian processes with information filtering does not occur elsewhere in the literature, and so represents a novel implementation.

A key difference between the centralised and federated architectures in an FDI setting is that it is considerably more difficult to uncouple a failed IMU sensor using the centralised information filter (CIF) as opposed to the federated Kalman filter, where the sensors are more loosely coupled. By corollary, centralised filter designs run a greater risk of failure because a faulty sensor measurement having gone undetected can straight away jeopardise the entire navigation solution. As with the GP Kalman filter variants from the last chapter, we monitor the innovation term to detect and isolate a failed sensor.

This chapter is laid out as follows. Section 5.1 presents the presents the relevant multi-sensor fusion architectures which underlie the centralised and federated approaches. This is followed by Section 5.2, which overviews the classical form of the centralised and federated information filters (FIF) used, as well as introducing the proposed extensions utilising GPs and the unscented transform. Section 5.3 evaluates the performance of the proposed algorithms through simulation experiments. Section 5.4 concludes the chapter.

## 5.1 Multi-Sensor Fusion

### 5.1.1 Sensor fusion architectures

Multi-sensor fusion deals with the extraction of information from observations gathered by a collection of sensors. The number of sensors involved can be two or greater. The object is that the integration of data from multiple sensors should lead to the acquisition of information that is more precise than would be possible otherwise.

A number of approaches to multi-sensor fusion are available, prominent amongst these is the Kalman filter and its information form. The key frameworks for sensor fusion based on the Kalman filter are measurement fusion and state-vector fusion. These have been the subject of numerous studies over the last two decades (Mosallaei & Salahshoor 2008). In measurement fusion, sensor measurement sets are fused directly to arrive at a weighted or integrated measurement set. Following this, a lone Kalman filter is used for state estimation - updated by the fused measurement. On the other hand, in state-

vector fusion, a bank of local Kalman filters is utilised, providing estimates based on individual sensors. The individual state estimates are subsequently fused to arrive at a joint state estimate.

In general, measurement fusion is deemed to offer superior estimation performance overall, whereas state-vector fusion entails a lower computational load and lower communication cost. It mirrors the features of a parallel implementation and exhibits the same advantages, including fault-tolerance (Gan & Harris 2001). State-vector fusion approaches come with the caveat that their effectiveness is lessened unless used with Kalman filters that are consistent. In areas of application like navigation and target tracking, the processes concerned are typically non-linear and noisy, leading to non-linear Kalman filters that employ approximations, like for example the Jacobian linearisation found in the EKF. Linearisation may introduce modelling errors, causing inconsistency. Hence, measurement fusion is more commonly adopted over state-vector fusion by researchers in the context of KF-based multi-sensor fusion. Both sensor fusion architectures are classified, in a general sense, as centralised architectures (Mosallaei & Salahshoor 2008).

Measurement fusion can be performed in two principal ways. One way (Method I) brings the multi-sensor data together en bloc, which expands the dimensionality of the filter's measurement vector. A second way (Method II) integrates the multi-sensor data by estimating the minimum mean square error, which leaves the dimensionality of the measurement vector unaltered. It is arguable that since Method I leverages all of the raw measurement information, it should exceed the performance of Method II. However, it is also the case that Method II bears a lower computational burden. Furthermore, Method II may not always be inapplicable, such as in situations involving different sensors that have observation matrices of varying size.

### 5.1.2 Measurement fusion

Let us consider a system whose dynamics and sensor outputs are modelled by the standard discrete-time state space model introduced previously. As before, the measurement noise is assumed to be independent and identically distributed.

*N* sensor models can be conflated into a single model as follows:

$$\mathbf{y}(\mathrm{k}) = C(k)\boldsymbol{x}(k) + \boldsymbol{w}(k) \tag{5-1}$$

Where $\boldsymbol{w}(k) \sim N(\mathbf{0}, \mathbf{R}(\mathrm{k}))$. In the following, we describe the two measurement fusion methods (Gan & Harris 2001).

- Measurement fusion method I

Fusion Method I fuses sensor readings from various sensors by modifying the observation vector as follows:

$$\mathbf{y}(\mathrm{k}) = \mathbf{y}^{(\mathrm{I})}(\mathrm{k}) = [\mathbf{y}_1(\mathrm{k}) \cdots \mathbf{y}_{\mathrm{N}}(\mathrm{k})] \tag{5-2}$$

$$\mathrm{C}(\mathbf{k}) = \mathrm{C}^{(\mathbf{I})}(\mathbf{k}) = [\mathrm{C}_{\mathbf{1}}(\mathbf{k}) \cdots \mathrm{C}_{\mathbf{N}}(\mathbf{k})] \tag{5-3}$$

$$\mathrm{R}(\mathrm{k}) = \mathrm{R}^{(\mathrm{I})}(\mathrm{k}) = [\mathrm{R}_1(\mathrm{k}) \cdots \mathrm{R}_{\mathrm{N}}(\mathrm{k})] \tag{5-4}$$

- Measurement fusion method II

Fusion Method II weights the various measurement sets in order to fuse multi-sensor data:

$$\mathbf{y}(\mathrm{k}) = \mathbf{y}^{(\mathrm{II})}(\mathrm{k}) = \left[\sum_{j=1}^{N} R_j^{-1}(k)\right]^{-1} \sum_{j=1}^{N} R_j^{-1}(k)\, \mathbf{y}_j(k) \tag{5-5}$$

$$\mathrm{C}(\mathbf{k}) = \mathrm{C}^{(\mathbf{II})}(\mathbf{k}) = \left[\sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} \boldsymbol{R_j^{-1}(k)}\right]^{-1} \sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} \boldsymbol{R_j^{-1}(k)}\, \mathrm{C}_{\boldsymbol{j}}(\boldsymbol{k}) \tag{5-6}$$

$$\mathrm{R}(\mathbf{k}) = \mathrm{R}^{(\mathbf{II})}(\mathbf{k}) = \left[\sum_{\boldsymbol{j=1}}^{\boldsymbol{N}} \boldsymbol{R_j^{-1}(k)}\right]^{-1} \tag{5-7}$$

For the state space model given in (5.1), the Kalman filter delivers an unbiased, optimal state estimate, i.e. it is a minimum mean-square error estimator for the system under consideration.

The information filter, i.e. the Kalman filter in information form, is functionally the same as the Kalman filter, but brings a lower computational overhead. This is particularly the case in multi-sensor fusion, wherein the innovation covariance $[C(k)P(k|k-1)C^T(k) + R(k)]$ is typically a non-diagonal, high-dimensional matrix. The critical difference that sets apart the stand-alone Kalman filter from its distributed counterpart, the information filter, is to be found in the measurement update.

The measurement update of the information filter is simplified since the gain matrix, $K(k)$, in the regular Kalman filter is more elaborate than the corresponding term in the information filter, $C^T(k)R^{-1}(k)$. In multi-sensor fusion the gain, $K(k)$, takes the form of the matrix inverse $[C(t)P(k|k-1)C^T(k)+R(k)]^{-1}$, which becomes computationally burdensome with increasing size and dimensionality.

Contrasting (5.2-5.4) and (5.5-5.7), it can be seen that the two approaches to measurement fusion are substantially different. Nevertheless, there does exist a certain duality or functional equivalence amongst the two approaches, as represented in the below theorem.

THEOREM *If the N sensors used for data fusion, with different and independent noise characteristics, have identical measurement matrices, i.e.,* $C_1(k) = C_2(k) = \cdots = C_N(k)$*, then the measurement fusion Method I is functionally equivalent to the measurement fusion Method II* (Gan & Harris 2001).

When using the Kalman filter, the functional equivalence of the respective measurement fusion approaches can be ascertained by simply verifying that the terms $K(k)C(k)$ and $K(k)y(k)$ of Method I are in fact functionally equivalent to those of Method II. Conversely, when using the information filter, the functional equivalence is established by comparing the terms $C^T(k)R^{-1}(k)C(t)$ and $C^T(k)R^{-1}(k)y(k)$ in both cases.

## 5.2 Information Filters

Section 5.1 introduced the principal paradigms for achieving sensor fusion as it relates to distributed Kalman or information filters in a centralised context. That is, where there is a central processing unit or fuser, which combines either measurements (measurement fusion) or estimated states (state-vector fusion). It should be noted that fully decentralised schemes for sensor fusion are also available (Kim & Hong 2003) that have been applied to non-linear state estimation as well (Bae & Kim 2010). Some authors have contended that in real-time multi-sensor systems the decentralised architecture can outperform centralised architectures (Kim & Hong 2003). In the present work, however, we only consider centralised fusion mechanisms. Indeed, from the architectures already described, measurement fusion (Method II) forms the basis of the centralised information filter (see Section 5.2.1) and state-vector fusion maps onto the federated information filter (see Section 5.2.2).

In a distributed filtering scheme, sensors are treated individually and at least some data processing occurs locally, followed, in the case of centralised systems, by propagation of the resulting signals to a data fusion centre that achieves a global estimate. Decentralised algorithms essentially parallelise the Kalman filter equations and produce a global estimate whilst resorting only to local estimates, with information being exchanged in a network of dedicated filter nodes attached to given sensors, removing the need for any central processor or fusion block (Durrant-Whyte 2001).

As previously stated, the information filter is basically a Kalman filter expressed in terms of information measures concerning state estimates and the corresponding covariances. Hence the name sometimes used for this filter: the inverse covariance Kalman filter. In spite of its relevance to multi-sensor systems, it has not been used extensively and is sparsely covered in the literature (Kim & Hong 2003).

The information filter algorithm was fully derived by (Mutambara 1998) as a variant of the Kalman filter stated in terms of information-theoretic variables,

which measure the quantity of information gathered about states being estimated.

Aside from the standard centralised filter architecture based on measurement fusion, a federated structure has also been proposed for multi-sensor data fusion. The motivation behind its emergence lies in the fact that a federated filter excels over other distributed filtering techniques in terms of simplicity and fault-tolerance (Kim & Hong 2003); the latter being a particularly valuable trait in an FDI context. Put another way, the federated filtering technique leverages information-sharing mechanisms and ensures close-to-globally-optimal or optimal estimation precision alongside a a fault-tolerant capability, stemming from the ease of decoupling sensors found to be faulty. The federated filter architecture leverages local filter nodes dedicated to individual sensors and a master filter to obtain a global state estimate through fusion of individual filter estimates.

The federated Kalman/information filter has certain features that bring it closer to a decentralised filter structure and is treated by some authors as a special case of the decentralised Kalman/information filter (Kim & Hong 2003). Other authors classify the state-vector fusion architecture it is based upon as centralised information sharing. Effectively, it is a compromise between the centralised and decentralised modes of distributed state estimation, and can be viewed either way. In our work, we adopt what appears to be the consensus definition in the literature; namely, we treat it as a member of the family of centralised information filters.

Though providing higher estimation quality than the EKF, the UKF and UHF introduced in Chapter 4 have some drawbacks, including a higher computational cost. The federated information filter, with its inherent fault-tolerance capability and ability to perform fault diagnosis and global estimation side-by-side, offers an avenue to improved performance as compared to the UKF, when the sigma point transform is embedded within its update structure. Additional benefits accrue as a consequence of the overall information paradigm: i.e. simplified estimation updating and greater decentralisation. As

with the information filter there is no gain or innovation covariance matrix, it is more efficient in multi- sensor systems.

### 5.2.1 Centralised information filter

The centralised information filter is applicable to systems with multiple measurements that aim to estimate global states optimally. Though the centralised Kalman filter, as distinct from the centralised information filter – which features information updates, purports to deliver an optimal solution to the estimation problem, the multiplicity of its states often places exorbitant computational demands that cannot be maintained in practical real-time applications. What is more, by definition, the estimate is a function of all previous measurement updates. This makes it difficult to extract the sensor data of a failed sensor from the estimate. Thus, concurrent (distributed) filtering solutions are often regarded as delivering better FDI capability, improved redundancy management and lower costs of system integration.

A solution that introduces additional degrees of decentralisation is the federated information filter (Lee 2003), described in Section 5.2.2. The federated information filter differs from the centralised information filter in that each measurement is processed by a local filter, and then the estimates of the individual filters are combined in a master filter. The local filters operate completely independently of each other and do not share information - in order to prevent contamination of solutions in case of sensor failure. A filter with a fault is simply ignored by the master filter.

The chief disadvantage of the federated information filter is that it does not equal the centralised filter performance-wise. Further, the federated filter requires greater processing power to implement the local filters. It can be shown that with respect to information sharing, the information filter with federated structure is equal to the centralised information filter (Kim & Hong 2003).

Filter accuracy depends on a-priori assumptions about system models and noise statistics. In practical applications, a-priori knowledge can be misleading. Estimation accuracy will degrade from the theoretical prediction. (Lee 2008)

proposes an adaptive filter to reduce discrepancies through a feedback strategy. Applications to fault detection and isolation on navigation systems of information filters include (Magrabi & Gibbens 2000; Lee 2003)

We apply the centralised and federated information filters to the IMU FDI scenario which is used throughout this thesis and will already be familiar from previous chapters. In place of the filter bank architecture of Chapter 4, we adopt the distributed structure of information filters, where the gyroscopes and accelerometers of the IMU are treated as separate sensor nodes, in a similar manner to the dedicated observer scheme's use of filters as that architecture's building blocks. Figure 5.1 provides a graphical illustration of the centralised filter structure specified through the equations that make up the filtering algorithm given in (5.8).

$$
\begin{aligned}
&\hat{x}_c(k/k-1) = \Phi_c(k-1)\hat{x}_c(k-1) \\
&P_c(k/k-1) = \Phi_c(k-1)P_c(k-1)\Phi_c^T(k-1) + Q_c \\
&P_c(k)^{-1} = P_c(k/k-1)^{-1} + \sum_{i=1}^{N} H_i^T R_i^{-1} H_i \\
&P_c(k)^{-1}\hat{x}_c(k) = P_c(k/k-1)^{-1}\hat{x}_c(k/k-1) + \sum_{i=1}^{N} H_i^T R_i^{-1} z_i(k)
\end{aligned}
\qquad \textbf{(5-8)}
$$

Where:

- $\hat{x}_c(k/k-1) \in \Re^{n_c}$ is an a-priori estimate of x(k) in CKF,
- $Q_c \in \Re^{n_c \times n_c}$ is a covariance matrix of system noise in CKF,
- $\hat{x}_c(k) \in \Re^{n_c}$ is an a-posteriori estimate of x(k) in CKF,
- $P_c(k/k-1) \in \Re^{n_c \times n_c}$ is an a-priori covariance of estimation errors in CKF,
- $P_c(k) \in \Re^{n_c \times n_c}$ is an a-posteriori covariance of estimation errors in CKF.

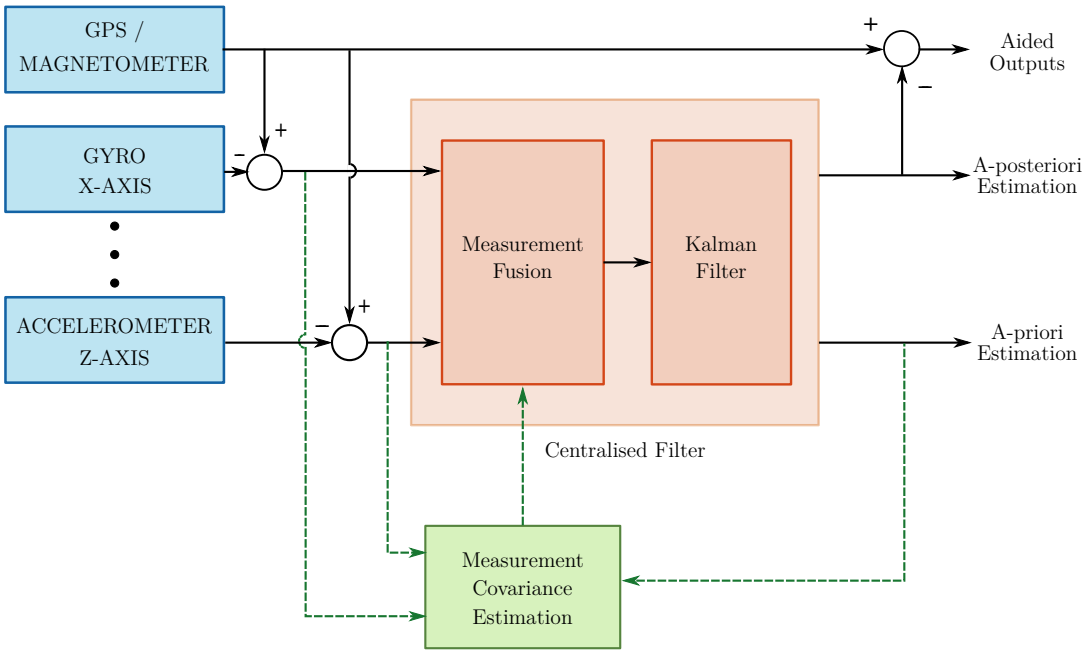



**Figure 5-1 Centralised information filter**

### 5.2.2 Federated information filter

In contrast to the centralised information filter, the federated filter fuses not the measurements from the sensor nodes, but the estimates of the local filters attached to each of the nodes. This mirrors the differing characteristics of the state-vector and measurement fusion architectures (Method II) introduced in Section 5.1. Because of the sensor nodes being served by dedicated filters, fault detection and isolation can occur at the local level, allowing easy decoupling of any faulty sensor and allowing a global estimate to be obtained.

The federated filter was suggested by Carlson (Waegli et al. 2008) and is considered a milestone in decentralised filter design. Federated filtering can be seen as a two-tier data processing solution, with the local filters occupying the lower level and the master filter above. The local filters run concurrently, independent from each other; their estimates are periodically fused by the master filter, yielding a global solution. A federated filter can detect and isolate a faulty local filter as soon as this filter fails. Thus, immediate reconstitution of the overall filter can eliminate any contamination from a faulty local filter.

More precisely, a federated filter in which local filters operate independently is a no-reset mode federated filter. This is one of a variety of design formulations that the federated filtering procedure can be applied to; each formulation reflects a different information sharing scheme or mode.

In federated no-reset mode, information feedback does not occur: each local filter maintains its process information locally, hence the master filter holds none of the fused data and the global fused estimate has no impact on any of the local estimates. There are pros and cons associated with each of the resetting modes.

Generally, the federated filter operated in reset mode is expected to provide better estimation accuracy, whilst in no-reset mode - a better tolerance of sensor faults.

The equations of the federated no-reset information filter are summarised in (5.9-5.10) below. The no-reset mode is adopted for the unscented federated filter design described in the next section.

Figure 5.2 depicts the no-reset federated information filter architecture. The option of adding complementary sensors in an integrated navigation system, such as a magnetometer and GPS is shown in the diagram, which could potentially provide further sources of analytical redundancy. This also illustrates how the federated information filter design is readily extensible; there being little difficulty in adding further sensors without necessitating too much disruption or re-design of an existing filter setup.

*Local filter* (*i*=1,2,…,N)

$$\hat{x}_i(k/k-1) = \Phi_i(k)\hat{x}_c(k-1) \qquad \textbf{(5-9)}$$
$$P_i(k/k-1) = \Phi_i(k)P_i(k-1)\Phi_i^T(k) + Q_i$$
$$P_i(k)^{-1} = P_i(k/k-1)^{-1} + H_i^T R_i^{-1} H_i$$
$$P_i(k)^{-1}\hat{x}_i(k) = P_i(k/k-1)^{-1}\hat{x}_i(k/k-1) + H_i^T R_i^{-1} z_i(k)$$

*Master filter*

$$P_f(k)^{-1} = \sum_{i=1}^{N} P_i(k)^{-1} \qquad \textbf{(5-10)}$$

$$P_f(k)^{-1}\hat{x}_f(k) = \sum_{i=1}^{N} P_i(k)^{-1}\hat{x}_i(k)$$

Where

- $\hat{x}_i(k/k-1) \in \Re^{n_i}$ is an a-priori estimate of x(k) in the i[th] local filter,
- $Q_i \in \Re^{n_i \times n_i}$ is a covariance matrix of system noise in the i[th] local filter,
- $\hat{x}_i(k) \in \Re^{n_i}$ is an a-posteriori estimate of x(k) in the i[th] local filter,
- $P_i(k/k-1) \in \Re^{n_i \times n_i}$ is an a-priori covariance of estimation errors in in the i[th] local filter,
- $P_i(k) \in \Re^{n_i \times n_i}$ is an a-posteriori covariance of estimation errors in the i[th] local filter,
- $P_f(k) \in \Re^{n_f \times n_f}$ is a fused covariance in the master filter,
- $\hat{x}_f(k) \in \Re^{n_f}$ is a fused estimate in the master filter.

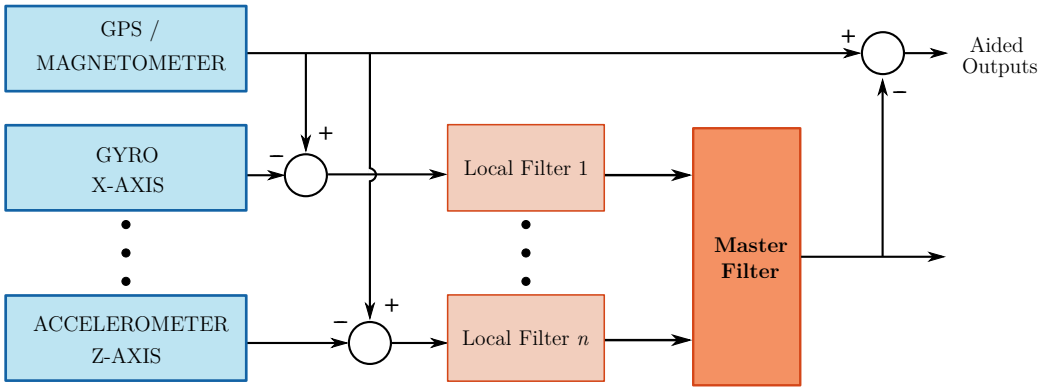



**Figure 5-2 Federated information filter**

### 5.2.3 GP unscented information filters

In this section, the centralised and federated unscented information filter algorithms are developed by embedding the unscented transformation into the linear information filter update structure outlined previously, thereby extending this framework to non-linear settings. The information filter definition contains new quantities, the Fisher information matrix, $P_i(k)^{-1}$, and the information state vector, $P_i(k)^{-1}\hat{x}_i(k)$.

The UIF is developed based on the unscented transformation and information filtering approach. Initially, the time update of the UIF algorithm is performed to propagate the state estimate and covariance from one measurement time to the next. To propagate from time step (k-1) to k, a set of sigma points is selected based on the current estimate of the mean and covariance. In the state propagation step, the a-priori state estimate and error covariance are evaluated by means of the propagated sigma points. The information prediction equations are then determined based on the a-priori estimates.

Next the measurement update is carried out. The update equations for the information matrix and the information state vector are the same as those given in (5.8-5.10), except the $H_i^T R_i^{-1} z_i(k)$ and $H_i^T R_i^{-1} H_i$ terms in (5.8-5.9) are replaced by the information state update, $i_k$, and its associated information matrix, $I_k$. These are defined as

$$\mathrm{i}_k = P_i(k/k-1)^{-1} P_{xz} R_i^{-1} z_i(k) \quad \textbf{(5-11)}$$

$$\mathrm{I}_k = P_i(k/k-1)^{-1} P_{xz} R_i^{-1} z_i(k) {P_{xz}}^T P_i(k/k-1)^{-1} \quad \textbf{(5-12)}$$

$P_{xz}$ and $P_z$ are defined as per the UKF update structure and the Gaussian process prediction and observation models are embedded exactly as they were for the GP-UKF/GP-UHF and have identical properties. Thus, the additional terms and information measures presented in this section (associated with the unscented transform) coupled to GP models are used to modify the centralised and federated structures of (5.8-5.10). GP centralised and federated unscented information filters are the end product.

## 5.3 Experimental Validation

### 5.3.1 Isolation results

The same experimental setup used in Chapters 3 and 4 is replicated to obtain comparative results for the Gaussian process centralised unscented information filter and the Gaussian process federated unscented information filter. The test data sets used are identical to those used in the simulation trials described in the aforementioned chapters. Namely, the IMU fault profiles designated in Table 3.2 overlaid onto a series of ten synthetic data sets, representing different segments of a spiral trajectory of an autonomous quadrotor vehicle with an on-board MEMS IMU. As before, fault detection and isolation is performed across the tri-axis gyroscopes and accelerometers of the IMU unit, giving rise to six residuals in all, generated through the innovation terms of the GP-CUIF and GP-FUIF.

From tables 5.1-5.3 it is seen that the results for the two algorithms are virtually identical, though these figures belie the fact that there do exist some very slight disparities in performance, albeit too slight to register an effect on the average performance values. This goes against the grain of expectations, since the CIF is thought to perform better than the FIF in most instances. There was also no noticeable difference in execution time between the two algorithms, though this may be due to the equalising effects of the GP models, which carry by far the highest computational load of all the filter update stages. To wit, the computational expense due to the GPs is so much greater than that due to other updates that they have little impact on the overall execution time. Thus the GP component blurs any processing time disparities between the two algorithms.

The GP-FUIF being able to equal the performance of GP-CUIF is an intriguing prospect, since in other ways it is the more versatile filter, with greater fault tolerance and decoupling ability. Further trials would need to be carried out to verify the statistical significance of these results and to establish whether the two filters are on a parity across different application scenarios. One downside of the GP-based UIFs is their relative long processing times.

**Table 5-1 Area under ROC curve and optimised thresholds**

| | GP_CUIF_AUC | GP_FUIF_AUC | GP_CUIF_Thresh | GP_FUIF_Thresh |
|---|---|---|---|---|
| X-axis accelerometer | 0.79 | 0.79 | 0.92 | 0.92 |
| Y-axis accelerometer | 0.83 | 0.83 | 2.67 | 2.67 |
| Z-axis accelerometer | 0.83 | 0.83 | 2.51 | 2.51 |
| X-axis gyro | 0.85 | 0.85 | 1.71 | 1.71 |
| Y-axis gyro | 0.86 | 0.86 | 1.9 | 1.9 |
| Z-axis gyro | 0.85 | 0.85 | 0.9 | 0.9 |
| Average | 0.83 | 0.83 | 1.77 | 1.77 |

**Table 5-2 GP-centralised unscented information filter isolation rates**

| | GP_CUIF_TruePos | GP_CUIF_FalsePos | GP_CUIF_TrueNeg | GP_CUIF_FalseNeg |
|---|---|---|---|---|
| X-axis accelerometer | 0.55 | 0.13 | 0.87 | 0.45 |
| Y-axis accelerometer | 0.17 | 0 | 1 | 0.83 |
| Z-axis accelerometer | 0.19 | 0.01 | 0.99 | 0.81 |
| X-axis gyro | 0.44 | 0.04 | 0.96 | 0.56 |
| Y-axis gyro | 0.41 | 0.02 | 0.98 | 0.59 |
| Z-axis gyro | 0.69 | 0.14 | 0.86 | 0.31 |
| Average | 0.41 | 0.05 | 0.95 | 0.59 |

**Table 5-3 GP-federated unscented information filter isolation rates**

| | GP_FUIF_TruePos | GP_FUIF_FalsePos | GP_FUIF_TrueNeg | GP_FUIF_FalseNeg |
|---|---|---|---|---|
| X-axis accelerometer | 0.55 | 0.13 | 0.87 | 0.45 |
| Y-axis accelerometer | 0.17 | 0 | 1 | 0.83 |
| Z-axis accelerometer | 0.19 | 0.01 | 0.99 | 0.81 |
| X-axis gyro | 0.44 | 0.04 | 0.96 | 0.56 |
| Y-axis gyro | 0.41 | 0.02 | 0.98 | 0.59 |
| Z-axis gyro | 0.69 | 0.14 | 0.86 | 0.31 |
| Average | 0.41 | 0.05 | 0.95 | 0.59 |

### 5.3.2 Residuals and ROC curves

The residuals in Figures 5.3-4 are not too dissimilar to those of the GP-UKF and GP-UHF. As for the latter, there is some evidence of overshooting in the residual, where the prediction and measurement that are differenced to obtain the residual are to some degree out of phase. This can cause the residual to spike irrespective of whether a fault has occurred or not.

The ROC curves in Figures 5.5-10 overlap for the most part, which is why there appears to be only a single curve, i.e. they are overlaid. The performance of the GP information filters is a little lower than for the GP-UKF and GP-UHF, but this is more than made up for by a lighter computational burden and, in the federative information filter's case, the ability to decouple faulty sensors and perform estimation, fusion and fault isolation at the same time.

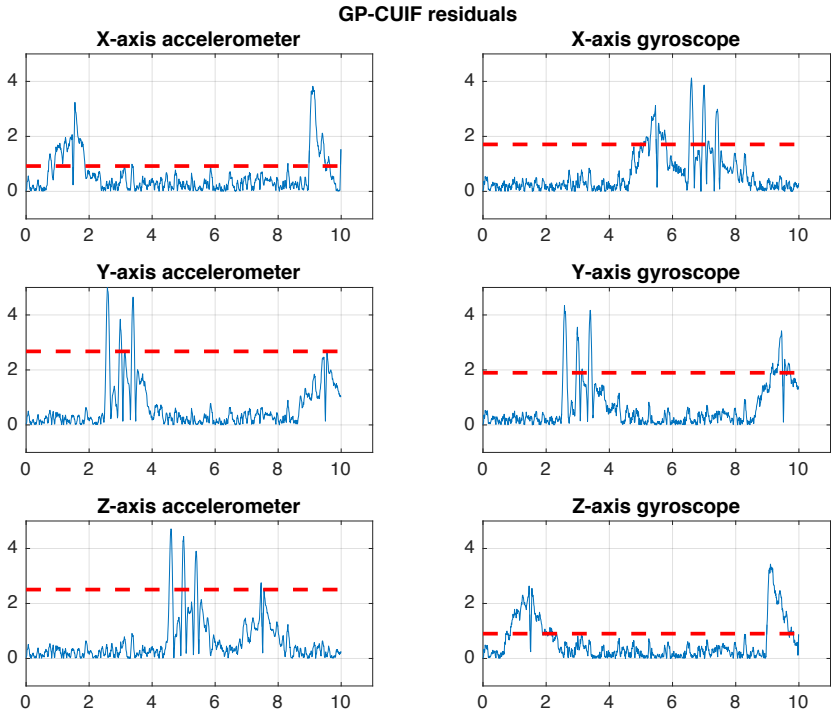



**Figure 5-3 GP-CUIF residuals**

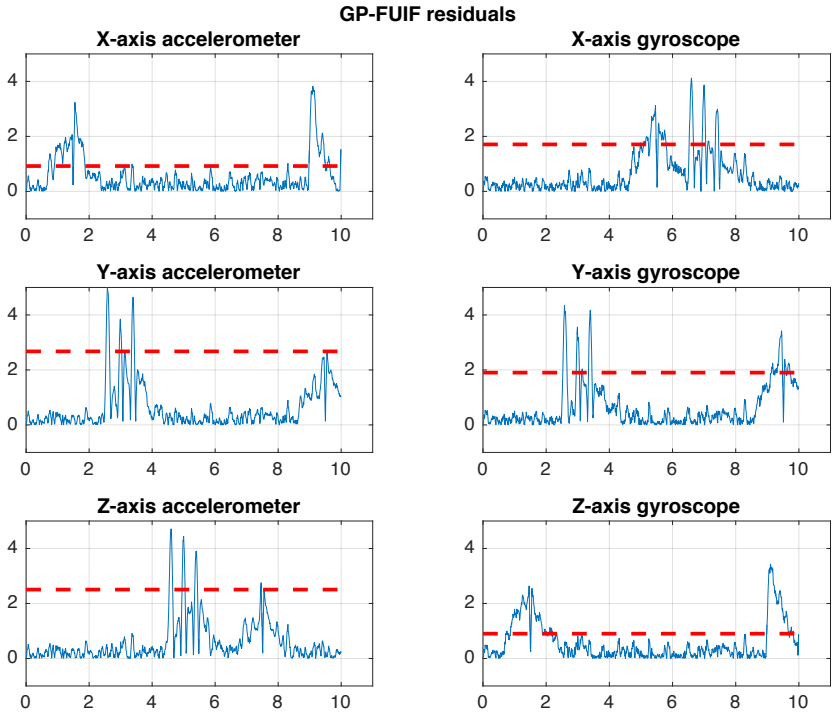



**Figure 5-4 GP-FUIF residuals**

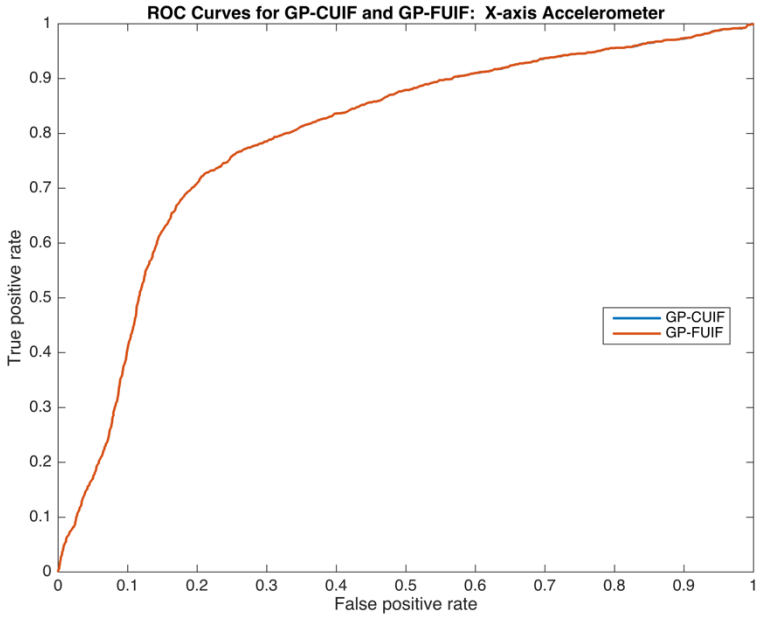



**Figure 5-5 X-axis accelerometer ROC curves**

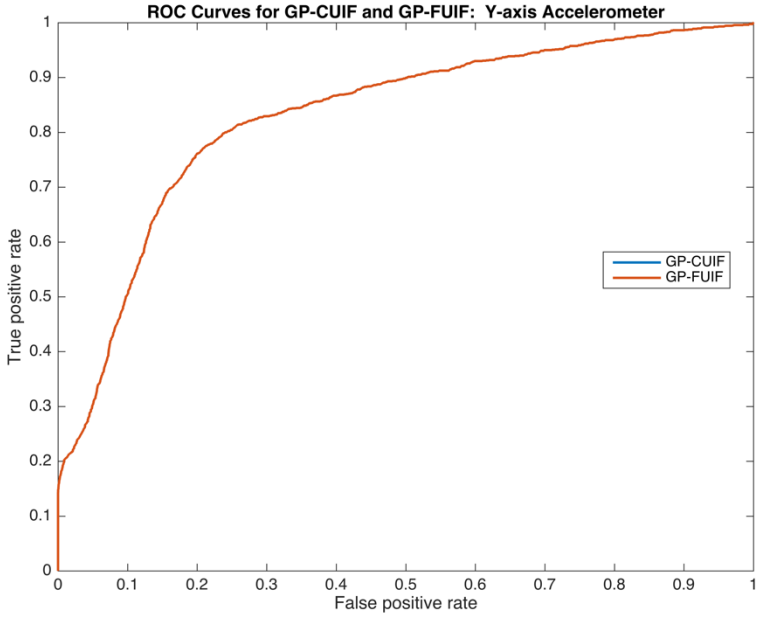



**Figure 5-6 Y-axis accelerometer ROC curves**

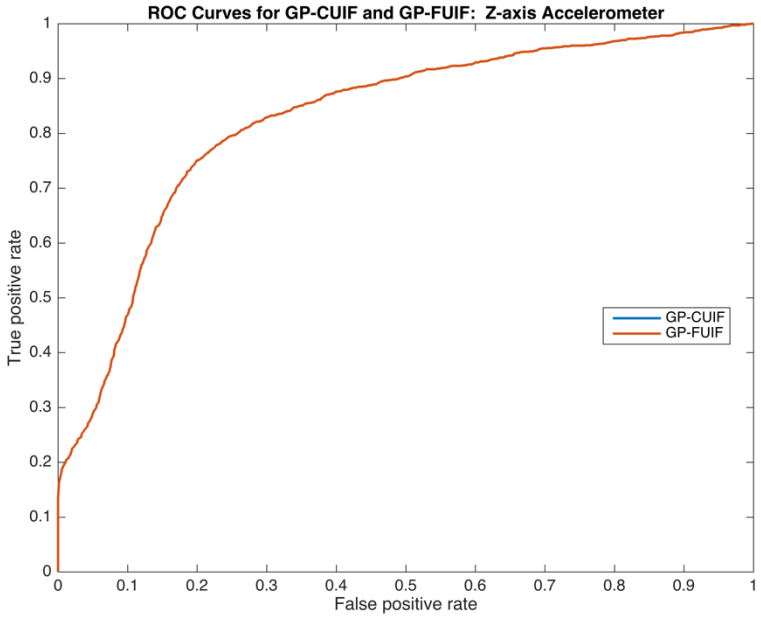



**Figure 5-7 Z-axis accelerometer ROC curves**

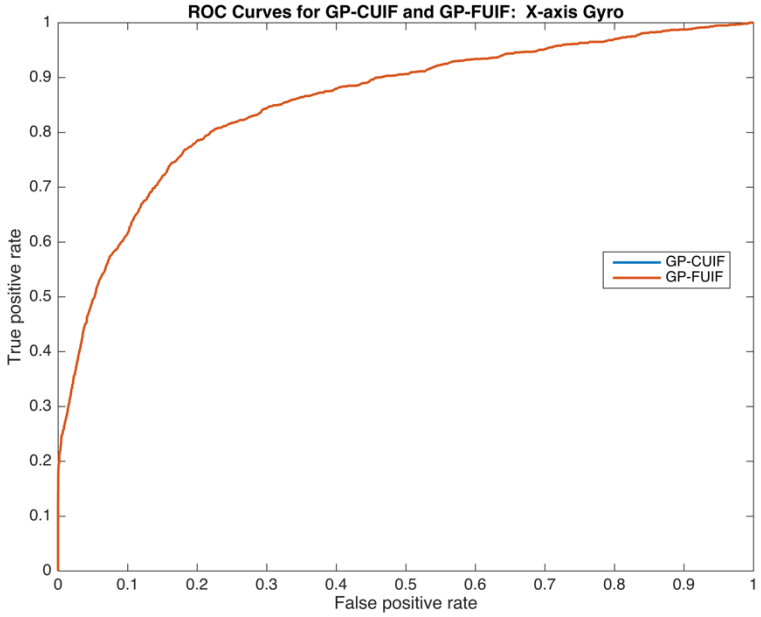



**Figure 5-8 X-axis gyro ROC curves**

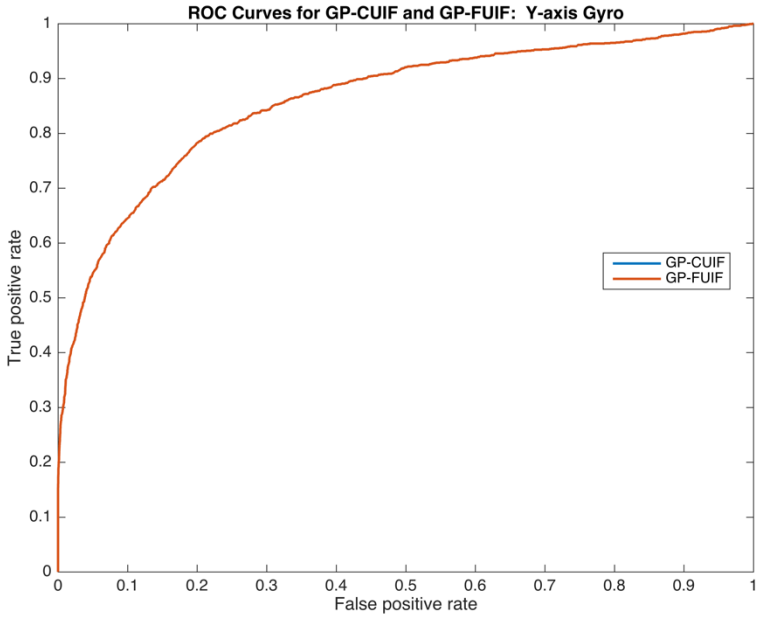



**Figure 5-9 Y-axis gyro ROC curves**

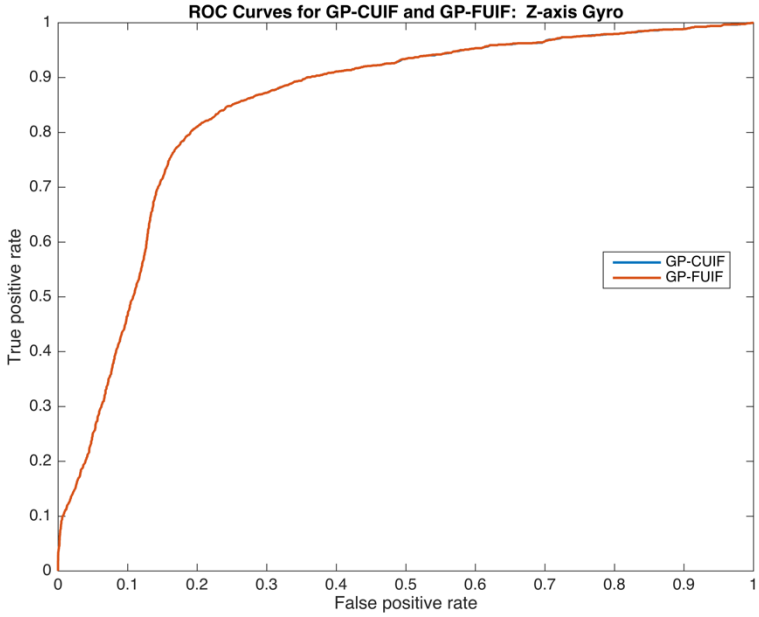



**Figure 5-10 Z-axis gyro ROC curves**

## 5.4 Conclusion

The chapter has sought to extend the Bayesian filtering FDI paradigm to decentralised and distributed settings. The dominant methodology in this research space is a decentralised dual form of the Kalman filter known as the information filter. It confers certain advantages over a stand-alone Kalman filter in terms of lower computational cost, simpler initialisation and, in some of its architectural forms, the ability to perform sensor data fusion and fault diagnosis at the same time. Additionally, the more decentralised information filter architectures can have the ability to decouple faulty sensors once they have been diagnosed, thus avoiding contamination of the filtered solution.

We explore and extend two forms of the information filter: the centralised and federated; or, more specifically, their non-linear forms, where the linearisation is performed based on the unscented or sigma point transform introduced in Chapter 4. We embed GP transition models as in Chapter 4 to develop two completely new information filter variants for FDI – a clear contribution to the state-of-the-art.

The evaluation results on the IMU data reveal that the federated and centralised approaches perform on a par with each other, though the federated approach has the additional merit of being capable of sensor decoupling as well as sensor fusion coincident with FDI.

# 6 DISCUSSION AND CONCLUSION

## 6.1 Summary and Discussion

Airborne navigation sensors are critical to a flight vehicle's ability to negotiate airspace. This is particularly the case with micro-rotorcraft that have limitations placed upon them as regards the payload they can bear, and hence come equipped with the cheapest and least resilient gyroscopes, accelerometers and magnetometers, i.e. MEMS devices, no different than those found in mobile telephones. Such devices are highly fault- and error-prone and require remedial action in the way of sensor fusion to offset imbalances and fault detection to prevent systemic failures. Investigating suitable algorithms to meet this need has been the focus of this thesis.

The approaches that have been explored belong to the paradigm of analytical redundancy, and, more specifically, to the branch of it concerned with model-based residual generators; that is, stochastic observers and parity relations – the most commonly used FDI techniques in the aerospace industry. It has been our aim to shine a new light on these techniques by coupling them with kernel-based non-parametric Bayesian regression and dimensionality reduction, i.e. Gaussian processes and kernel PCA respectively. The use of non-linear observers based around the unscented or sigma-point transform has been another unifying thread that runs through this thesis.

Thesis contributions made are contained in Chapters 3-5, which provide technical and theoretical coverage of the proposed algorithms.

In Chapter 3 we developed the kernel partial PCA technique, which was shown to be capable of effective IMU fault detection and isolation on a bespoke synthetic data set. This technique resulted from combining a linear partial PCA algorithm with non-linear PCA (in the form of kernel PCA). We thereby arrive at a 'best of both worlds' type scenario, ending up with an algorithm that inherits the isolation properties of partial PCA and non-linear approximation ability of kernel PCA. A further contribution is the introduction of kernel PCA into the aerospace realm. Even though there is a strong duality between PCA and the

parity relations technique, the former is seldom used in the aerospace sector, but widely adopted in the chemical processing industry, where it has proved to be something of a mainstay of FDI solutions. Our experiments revealed that partial kPCA can work very well on the IMU detection task and is much faster than the observer-based approaches that we have proposed, but is sensitive to the choice of kernel function and to numerical instabilities. Also, it seems to require a larger training set than the GP models.

In Chapters 4 & 5, we presented hybrid non-linear observers combined with Gaussian process models. The first of these – the GP-UKF – is a known filter, but to the author's knowledge has not seen application in FDI before. We cast this filter in the dedicated observer scheme, making the necessary adaptations for it to function in a filter bank. A second contribution is the introduction of an entirely novel filter design: the GP-UHF. This filter slightly outperformed the GP-UKF on the IMU diagnosis task. Two further contributions consist in combining the GP formalism and unscented transform with two distributed estimation architectures based around the Kalman filter: the federated and centralised information filters. While slightly below par in terms of their performance as compared to the GP-UKF and GP-UHF, the federated design brings ancillary benefits such as the ability to perform sensor fusion and fault detection at the same time.

The procedures developed in Chapters 4 & 5 were markedly slower than kPCA, but showed more robust performance and less proneness to numerical instabilities, whilst performing somewhat better in terms of the area under ROC curve performance metric.

## 6.2 Future Work

One possible extension of the work lies in the direction of adaptive filters. The object of adaptive measurement fusion is automatic isolation and recovery from some sensor failures - a feature lacking in most regular state observers - in addition to core monitoring capability. Such an adaptive filter has been proposed by (Lee 2003). Since the GP process model already provides an

estimate of the process covariance, it could form the basis fo such a filter design.

An adaptive filter which tunes the measurement covariance would virtually eliminate the need for parameter tuning and make the filter more responsive to fluctuations in its environment.

A further direction of future work could be to use sparsification on the GP process models, in order to lighten their computational load. Another way to achieve the same end would be to turn to a sequential or online GP implementation. Multi-output GP models are another interesting new development, which could help improve the filters described in this work.

# REFERENCES


Abid, M., 2010. *Fault detection in nonlinear systems: An observer-based approach*. University of Duisburg-Essen.

Adams, R.P. & MacKay, D.J., 2007. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*.

Angeli, C. & Chatzinikolaou, A., 2004. On-Line Fault Detection Techniques for Technical Systems: A Survey. *IJCSA*, 1(1), pp.12–30.

Atia, M.M., Noureldin, A. & Korenberg, M., 2012. Enhanced Kalman Filter for RISS/GPS Integrated Navigation using Gaussian Process Regression. In *Proceedings of the 2012 International Technical Meeting of The Institute of Navigation*. pp. 1148–1156.

Atia, M.M., Noureldin, A. & Korenberg, M., 2011. Gaussian process regression approach for bridging GPS outages in integrated navigation systems. *Electronics Letters*, 47(1), p.52.

Bae, J.B.J. & Kim, Y.K.Y., 2010. Nonlinear estimation for spacecraft attitude using decentralized unscented information filter. In *Control Automation and Systems (ICCAS), 2010 International Conference on*. pp. 1562–1566.

Bancroft, J.B. & Lachapelle, G., 2011. Data fusion algorithms for multiple inertial measurement units. *Sensors*, 11(7), pp.6771–6798.

Bar-Shalom, Y., Li, X.-R. & Kirubarajan, T., 2001. *Estimation with applications to tracking and navigation*, New York: Wiley.

Basseville, M. & Nikiforov, I. V, 1993. *Detection of abrupt changes: theory and application*, Englewood Cliffs, N.J.: Prentice Hall.

Berdjag, D. et al., 2010. Fault detection and isolation for redundant aircraft sensors. In *Control and Fault-Tolerant Systems (SysTol), 2010 Conference on*. pp. 137–142.

Burges, C.J.C., 2009. *Dimension Reduction: A Guided Tour*,

Chandola, V., Banerjee, A. & Kumar, V., 2009. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3), p.15.

Chatfield, a. B., 1997. *Fundamentals of high accuracy inertial navigation*, AIAA.

Cheng, H., Nikus, M. & Jämsä-Jounela, S.L., 2008. Evaluation of PCA methods with improved fault isolation capabilities on a paper machine simulator. *Chemometrics and Intelligent Laboratory Systems*, 92(2), pp.186–199.

Cheng, Q., Varshney, P.K. & Belcastro, C.M., 2008. Fault detection in dynamic systems via decision fusion. *Aerospace and Electronic Systems, IEEE Transactions on*, 44(1), pp.227–242.

Choi, S.W. et al., 2005. Fault detection and identification of nonlinear processes based on kernel PCA. *Chemometrics and Intelligent Laboratory Systems*, 75(1), pp.55–67.

Chouaib, C., Mohamed-Faouzi, H. & Messaoud, D., 2013. Adaptive kernel principal component analysis for nonlinear dynamic process monitoring. In *2013 9th Asian Control Conference (ASCC)*. pp. 1–6.

Chow, E.Y. & Willsky, a. S., 1984. Analytical redundancy and the design of robust failure detection systems. In *IEEE Transactions on Automatic Control*. pp. 603–614.

Construit, A.E.T. et al., 2009. *Trajectory Determination and Analysis in Sports by Satellite and Inertial Navigation*. ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE.

Durrant-Whyte, H., 2001. *Multi Sensor Data Fusion*,

Einicke, G.A. & White, L.B., 1999. Robust extended Kalman filtering. *Signal Processing, IEEE Transactions on*, 47(9), pp.2596–2599.

Fawcett, T., 2006. An introduction to ROC analysis. *Pattern Recognition Letters*, 27, pp.861–874.

Frank, P.M., 1990. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy - A survey and some new results.

*Automatica*, 26(3), pp.459–474.

Gan, Q. & Harris, C.J., 2001. Comparison of two measurement fusion methods for Kalman-filter-based multisensor data fusion. *Aerospace and Electronic Systems, IEEE Transactions on*, 37(1), pp.273–279.

Gertler, J. et al., 1999. Isolation enhanced principal component analysis. *AIChE Journal*, 45(2), pp.323–334.

Gertler, J. & Cao, J., 2005. Design of optimal structured residuals from partial principal component models for fault diagnosis in linear systems. *Journal of Process Control*, 15, pp.585–603.

Groves, P.D., 2008. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, Boston: Artech House.

Hagenblad, A., Gustafsson, F. & Klein, I., 2003. Fault Detection: the Parity Space Approach and Principal Components Analysis. In *13th IFAC Symposium on System Identification 2003*.

Hoffmann, H., 2007. Kernel PCA for novelty detection. *Pattern Recognition*, 40(3), pp.863–874.

Huang, Y., Gertler, J. & McAvoy, T., 2000. Sensor and actuator fault isolation by structured partial PCA with nonlinear extensions. *Journal of Process Control*, 10, pp.459–469.

Hwang, I. et al., 2010. A survey of fault detection, isolation, and reconfiguration methods. *Control Systems Technology, IEEE Transactions on*, 18(3), pp.636–653.

Isermann, R., 2006. *Fault-Diagnosis Systems*, [New York]: Springer-Verlag Berlin Heidelberg.

Isermann, R., 1997. Supervision, fault-detection and fault-diagnosis methods—an introduction. *Control engineering practice*, 5(5), pp.639–652.

Jain, A.K., Duin, R.P.W. & Mao, J., 2000. Statistical pattern recognition: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*,

22(1), pp.4–37.

Jia, H., 2004. *Data fusion methodologies for multisensor aircraft navigation systems*. Cranfield University.

Julier, S.J. & Uhlmann, J.K., 1997. New extension of the Kalman filter to nonlinear systems. In *AeroSense'97*. pp. 182–193.

Kim, P., 2011. *Kalman Filter for Beginners: with MATLAB Examples*,

Kim, Y.-S. & Hong, K.-S., 2003. Decentralized information filter in federated form. In *SICE 2003 Annual Conference (IEEE Cat. No.03TH8734)*. pp. 2176–2181.

Kinnaert, M., 2003. Fault diagnosis based on analytical models for linear and nonlinear systems-a tutorial. In *Proceedings of IFAC Safeprocess*. pp. 37–50.

Ko, J. et al., 2007a. Gaussian Processes and Reinforcement Learning for Identification and Control of an Autonomous Blimp. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*. pp. 742–747.

Ko, J. et al., 2007b. GP-UKF: Unscented Kalman filters with Gaussian process prediction and observation models. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. pp. 1901–1907.

Ko, J. & Fox, D., 2008. GP-BayesFilters: Bayesian filtering using Gaussian process prediction and observation models. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*. pp. 3471–3476.

Kobayashi, T. & Simon, D.L., 2003. *Application of a bank of Kalman filters for aircraft engine fault diagnostics*, DTIC Document.

Kobayashi, T. & Simon, D.L., 2004. *Evaluation of an enhanced bank of Kalman filters for in-flight aircraft engine sensor fault diagnostics*, DTIC Document.

Kocijan, J., 2012. Dynamic GP Models: An Overview and Recent Developments. In *Proceedings of the 6th International Conference on Applied Mathematics, Simulation, Modelling*. pp. 38–43.

Kung, S.Y., 2014. *Kernel Methods and Machine Learning*,

Lee, D.J., 2008. Nonlinear Estimation and Multiple Sensor Fusion Using Unscented Information Filtering. *Ieee Signal Processing Letters*, 15(2), pp.861–864.

Lee, T., 2003. Centralized Kalman Filter with Adaptive Measurement Fusion : its Application to a GPS / SDINS Integration System with an Additional Sensor. *International Journal of Control, Automation, and Systems*, 1(4), pp.444–452.

Li, R. & Olson, J.H., 1991. Fault detection and diagnosis in a closed-loop nonlinear distillation process: application of extended Kalman filters. *Industrial & Engineering Chemistry Research*, 30(5), pp.898–908.

Li, W. & Jia, Y., 2010. H infinity filtering for a class of nonlinear discrete time systems based on unscented transform. *Signal Processing*, 90(12), pp.3301–3307.

Lopes, H.J.D., 2011. *Attitude Determination of Highly Dynamic Fixed-wing UAVs*. Technical University of Lisbon.

Magrabi, S.M. & Gibbens, P.W., 2000. Decentralised fault detection and diagnosis in navigation systems for unmanned aerial vehicles. In *IEEE 2000 Position Location and Navigation Symposium*. pp. 363–370.

Mahadevan, S. & Shah, S.L., 2009. Fault detection and diagnosis in process data using one-class support vector machines. *Journal of Process Control*, 19(10), pp.1627–1639.

Marcos, A., Ganguli, S. & Balas, G.J., 2005. An application of $\mathcal{H}\infty$ fault detection and isolation to a transport aircraft. *Control Engineering Practice*, 13, pp.105–119.

Marzat, J. et al., 2012. Model-based fault diagnosis for aerospace systems: a survey. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 226(10), pp.1329–1360.

Maurya, M.R., Rengaswamy, R. & Venkatasubramanian, V., 2007. Fault diagnosis using dynamic trend analysis: A review and recent developments. *Engineering Applications of Artificial Intelligence*, 20(2), pp.133–146.

Mehra, R.K. & Peschon, J., 1971. An innovations approach to fault detection and diagnosis in dynamic systems. *Automatica*, 7(5), pp.637–640.

Montmain, J. & Gentil, S., 2000. Dynamic causal model diagnostic reasoning for online technical process supervision. *Automatica*, 36(8), pp.1137–1152.

Mosallaei, M. & Salahshoor, K., 2008. Comparison of Centralized Multi-Sensor Measurement and State Fusion Methods with an Adaptive Unscented Kalman Filter for Process Fault diagnosis. In *2008 4th International Conference on Information and Automation for Sustainability*. IEEE, pp. 519–524.

Mutambara, A.G.O., 1998. *Decentralized Estimation and Control for Multisensor Systems*, CRC Press.

Nemra, A., 2011. *Robust airborne 3D visual simultaneous localisation and mapping*. PhD. Shrivenham: Cranfield University.

Nemra, A. & Aouf, N., 2010. Robust INS/GPS Sensor Fusion for UAV Localization Using SDRE Nonlinear Filtering. *IEEE Sensors Journal*, 10(4), pp.789–798.

Noureldin, A. et al., 2009. Performance Enhancement of MEMS-Based INS / GPS Integration for Low-Cost Navigation Applications. In *IEEE Transactions on Vehicular Technology*. pp. 1077–1096.

Panati, A. & Dupré, D.T., 2001. Causal simulation and diagnosis of dynamic systems. In *AI* IA 2001: Advances in Artificial Intelligence*. Springer, pp. 135–146.

Patton, R.J., 2000. *Issues of Fault Diagnosis for Dynamic Systems*, London; New York: Springer-Verlag London.

Patton, R.J., Lopez-Toribio, C.J. & Uppal, F.J., 1999. Artificial intelligence approaches to fault diagnosis. In *Condition Monitoring: Machinery, External Structures and Health (Ref. No. 1999/034), IEE Colloquium on*. pp. 5–1.

Rago, C. et al., 1998. Failure detection and identification and fault tolerant control using the IMM-KF with applications to the Eagle-Eye UAV. In *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*. pp. 4208–4213.

Rasmussen, C.E. & Williams, C.K.I., 2006. *Gaussian processes for machine learning*, Cambridge, Mass.: MIT Press.

Reece, S. & Roberts, S., 2010. An introduction to Gaussian processes for the Kalman filter expert. In *Information Fusion (FUSION), 2010 13th Conference on*. pp. 1–9.

Roumeliotis, S.G.S. and G.A.B., 1999. Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization. In *IEEE International Conference on Robotics and Automation*. Detroit, MI.

Schaefer, R., 2003. *Unmanned aerial vehicle reliability study*,

Schölkopf, B., 2002. *Learning with kernels*, MIT Press.

Schumacher, A., 2006. *Integration of a gps aided strapdown inertial navigation system for land vehicles*. MSc. Sweden: Royal Institute of Technology.

Shaked, U. & Berman, N., 1995. H∞ nonlinear filtering of discrete-time processes. *Signal Processing, IEEE Transactions on*, 43(9), pp.2205–2209.

Shim, D.-S. & Yang, C.-K., 2010. Optimal configuration of redundant inertial sensors for navigation and FDI performance. *Sensors*, 10(7), pp.6497–6512.

Shin, H.J., Eom, D.-H. & Kim, S.-S., 2005. One-class support vector machines—an application in machine fault detection and classification.

*Computers & Industrial Engineering*, 48(2), pp.395–408.

Siddiqui, S.M. & Jiancheng, F., 2012. Robust Integrated Navigation of a Low Cost System. , pp.633–636.

Simon, D., 2008. A comparison of filtering approaches for aircraft engine health estimation. *Aerospace Science and Technology*, 12(4), pp.276–284.

Simon, D., 2000. From here to infinity. *Embedded Systems Programming*, (July 2000), pp.2–9.

Simon, D., 2001. Kalman Filtering. *Embedded System Programming*, (June), pp.72 –79.

Simon, D., 2006a. *Optimal state estimation: Kalman, H [infinity] and nonlinear approaches*, Hoboken, N.J.: Wiley-Interscience.

Simon, D., 2006b. Using nonlinear Kalman filtering to estimate signals. *Embedded Systems Design*, 13(1), pp.83–86.

Simon, D.J., 2006. Using nonlinear Kalman filtering to estimate signals. *Embedded Systems Design*, 19(7), pp.38–53.

Tedaldi, D., 2013. *IMU calibration without mechanical equipment*. University of Padova.

Thrun, S. & Fox, D., 2005. *Probabilistic robotics*, Cambridge, Mass.: MIT Press.

Vitanov, I. & Aouf, N., 2014a. Fault detection and isolation in an inertial navigation system using a bank of unscented H-infinity filters. In *2014 UKACC International Conference on Control, CONTROL 2014 - Proceedings*. pp. 250–255.

Vitanov, I. & Aouf, N., 2013. Fault detection and isolation in inertial navigation systems with SDRE non-linear filter. In *5th European Conference for Aeronautics and Space Sciences (EUCASS)*.

Vitanov, I. & Aouf, N., 2014b. Fault diagnosis and recovery in MEMS inertial navigation system using information filters and Gaussian processes. In

*2014 22nd Mediterranean Conference on Control and Automation, MED 2014*. pp. 115–120.

Vitanov, I. & Aouf, N., 2014. Fault diagnosis for MEMS INS using unscented Kalman filter enhanced by Gaussian process adaptation. In *Hardware and Systems (AHS), 2014 NASA/ ….*

Waegli, A., Guerrier, S. & Skaloud, J., 2008. Redundant MEMS-IMU integrated with GPS for Performance Assessment in Sports. In *Position, Location and Navigation Symposium, 2008 IEEE/ION*. pp. 1260–1268.

Woodman, O., 2007. *An introduction to inertial navigation*,

Yang, C. & Shim, D., 2007. Double Faults Isolation Based on the Reduced-Order Parity Vectors in Redundant Sensor Configuration. *International Journal*, c.

Zhang, Y. & Jiang, J., 2001. Integrated active fault-tolerant control using IMM approach. *Aerospace and Electronic Systems, IEEE Transactions on*, 37(4), pp.1221–1235.

Zolghadri, A., 2011. The challenge of advanced model-based fdir techniques for aerospace systems: the 2011 situation. *Progress in Flight Dynamics, Guidance, Navigation, Control, Fault Detection, and Avionics*, 6, pp.231–248.