UCGE Reports

Number 20276

Department of Geomatics Engineering

Improving Usability of Low-Cost INS/GPS Navigation Systems using Intelligent Techniques

(URL: http://www.geomatics.ucalgary.ca/links/GradTheses.html)

by

Christopher L. Goodall

January 2009

UNIVERSITY OF CALGARY

Improving Usability of Low-Cost INS/GPS Navigation Systems using Intelligent Techniques

by

Christopher L. Goodall

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF GEOMATICS ENGINEERING

CALGARY, ALBERTA

JANUARY, 2009

UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommended to the Faculty of Graduate Studies for acceptance, a thesis entitled “Improving Usability of Low-Cost INS/GPS Navigation Systems using Intelligent Techniques” submitted by Christopher L. Goodall in partial fulfillment of the requirements for the degree of doctor of philosophy.

_____

Supervisor, Dr. Naser El-Sheimy, Geomatics Engineering

_____

Dr. Elizabeth Cannon, Geomatics Engineering

_____

Dr. Aboelmagd Noureldin, Royal Military College of Canada

_____

Dr. Mark Petovello, Geomatics Engineering

_____

Dr. Simon Park, Mechanical & Manufacturing Engineering

_____

External Examiner, Dr. Chris Jekeli, Ohio State University

_____

Date

ABSTRACT

Integrated navigation systems using inertial and GPS signals are becoming popular in low-cost vehicle guidance systems as the limits of GPS-only navigators are pushed by consumers who operate in areas of poor signal coverage. Inertial sensors have been used successfully in many higher grade systems such as precision farming equipment and high accuracy road surveying. Their deployment in low-cost systems has been limited due to cost and performance tradeoffs which result in reduced usability for the user.

Low-cost inertial sensors, on the order of tens of dollars at the time of this publication, can be used to aid GPS when the signal quality is poor, but very low-cost inertial sensors, less than ten dollars, can introduce large error terms that greatly impact the performance of the integrated system if not properly dealt with by the designer. Unfortunately, the very low cost of the integrated systems prevents proper adjustment of these error terms before the unit is deployed, which can result in severely degraded performance. A number of filtering algorithms are used to combine GPS and inertial measurements, but regardless of the blending method there are *a priori* parameters used to define the error terms; these parameters need to be adjusted for individual systems to achieve optimal performance.

This research explores intelligent methods that are capable of adapting to these error parameters as the system is used. This maintains low cost while improving the performance over time to the level of a well tuned system. Intelligent methods are useful

since they can adapt and learn how to improve the performance based on absolute and relative measures.

Two methods are detailed for on-line performance improvement. The first uses two parallel filters to create a reference signal for state compensation of the navigation filter. This method takes dynamic and temporal inputs and is used to predict the position state error signals using an artificial neural network. An adaptive fuzzy inference system is used to control the application of the neural compensations to the filter states.

The second method uses relative performance measures to directly adapt the *a priori* error parameters. An absolute signal cannot be used since there is nothing known about the true parameter values. Reinforcement learning is applied so that performance can be evaluated over many different *a priori* parameters. This type of learning improves the performance by rewarding good parameters and punishing poor choices of parameters, by monitoring relative state improvements or degradations.

Both methods are applied in a general way to specific filtering algorithms, but it should be appreciated that the methods have been developed for any filter type that requires *a priori* information and provides position state estimation. The performance improvements largely depend on the quality of inertial sensors used in the integration, but improvements in position estimation of over 50% from the default manufacturer values have been obtained using two different low-cost sensor triads.

ACKNOWLEDGMENTS

From my experience, success in graduate school can largely be attributed to one's supervisor and a student's relationship to their supervisor. I can adamantly say that Dr. Naser El-Sheimy will be one of the most influential people in my life. He knew when to help me, knew when to push me, and knew when to encourage me. He understood me as a person and guided me in the correct path for success. He is an extremely well balanced person and has imparted much of his experience to me. Thank you for your wisdom, time and energy.

My family has also been a major source of encouragement. Thanks to my dad who challenged me on the practicality of my ideas, believed in me, and of course who helped me financially. Thanks to my mom who encouraged me to pursue my doctorate. I think I might have originally done all this just to make her happy. She was also there to push me to finish my thesis and helped in the editing process.

Finally, thank you Kristine for being there when I needed a rock. Your attitude of "you can do anything" really helped me realize of what I was capable. Without that extra reality check I might have never got to finishing this thesis and moving on to bigger and better things. With you my life has been busier yet more rewarding. I am better balanced, more efficient, harder working and most importantly happy. Thank you for everything with which you provide me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

Acronyms/Abbreviations

1D	One dimensional
2D	Two dimensional
3D	Three dimensional
Accel	Accelerometer
ADI	Analog Devices Incorporated
AI	Artificial intelligence
AKF	Adaptive Kalman filter
ANFIS	Adaptive neuro fuzzy inference system
ANN	Artificial neural network
ARW	Angular random walk
AV	Allan variance
b-frame	Body frame
BP	Backpropagation
C/A	Coarse acquisition
DCM	Direction cosine matrix
DGPS	Differential mode GPS
DoD	Department of Defense
DOF	Degree of freedom
DOP	Dilution of precision
DP	Dynamic programming
DR	Dead reckoning
ECEF	Earth centred earth fixed frame
EKF	Extended Kalman filter
ENU	East, North, Up
ESC	Electronic stability control
FIFO	First in first out
FIS	Fuzzy inference system
GNSS	Global navigation satellite system
GPS	Global positioning system
Gyro	Gyroscope
i-frame	Inertial frame
IAE	Innovation-based adaptive estimation
IMU	Inertial measurement unit
INS	Inertial navigation system
KF	Kalman filter
L1	L1 frequency band centred at 1575.42 MHz
L2	L2 frequency band centred at 1227.60 MHz
L2 C/A	New L2 unprotected civil signal
L5	L5 frequency band centred at 1176.45 MHz

LC	Loosely coupled
l-frame	Local level frame
LHU	Large heading uncertainty
LLF	Local level frame
LMS	Least mean squares
LPF	Low pass filter
MC	Monte Carlo method
MEMS	Micro-electro mechanical system
minFunc	Minimization function
MMAE	Multiple model adaptive estimation
MP	Markov process
MP ANN	Multilayer perceptron ANN
MSE	Mean squared error
N	Integer ambiguity term
n-frame	Navigation frame
NN	Nearest neighbour
OEM	Original equipment manufacturer
PCA	Principal components analysis
PRN	Pseudo random noise
PVA	Position, velocity and attitude
RBF	Radial basis function
RF	Radio frequency
RL	Reinforcement learning
RMS	Root mean square
RW	Random walk
SF	Scale factor
SGPS	Single point GPS
Std	Standard
std	Standard deviation
TC	Tightly coupled
TD	Temporal difference
US	United States
VRW	Velocity random walk
WAAS	Wide area augmentation system
WADGPS	Wide area differential GPS
WGS	World geodetic system
ε-greedy	Epsilon greedy

Symbols

V	Velocity
$\otimes$	Cross product
Ω	Rotation rate
d	Displacement
d_0	Initial displacement

v0	Initial velocity
$\int$	Integral
a	Acceleration
t	Time
dt	Differential time
ω_{ie}^{e}	Rotation from the e-frame, from ECEF to inertial
ω^{e}	The Earth rotation rate
N	Radius of curvature of the prime vertical
e	The first eccentricity of the Earth ellipsoid
h	Altitude
φ	Latitude
λ	Longitude
R_{l}^{e}	Rotation matrix from the l-frame to the e-frame
A	Azimuth
p	Pitch
r	Roll
Ω	Skew symmetric matrix
q	Quarternion parameter
I	The measured signal for either the raw accelerometer (f) or the raw gyroscope (w)
f	Error-free accelerometer signal
w	Error-free gyroscope signal
b	Bias offset term for either the accelerometer (a) or gyroscope (g)
b_n	Bias instability
b_T	Bias term due to thermal effects
S_1	Matrix representing the linear scale factor errors
S_2	Matrix representing the non-linear scale factor errors
S_T	Matrix representing the scale factor linear error due to a temperature change
ΔT	Temperature change
N	Matrix representing the non-orthogonalities of the individual triad axes. N does not consider between triad misalignment.
n	Vector representing sensor noise that drives ARW or VRW
δ_{AV}	Percentage error in Allan standard deviation
x	State vector
P	Covariance matrix of the state
Φ	Transition matrix
Q	Spectral density matrix
K	Kalman gain matrix
R	Matrix that defines the noise of the measurements
H	Matrix that defines the noiseless connection between the measurement and state vector
v	Innovation sequence
C_{vk}	Innovation sequence covariance matrix
w	Network weight
b	Network bias

f	Activation function
e	Error signal
γ	Discount value
R	Long term reward
r	Short term reward
E	Expectation operator
s	State
a	Action
$*$	Optimal policy
λ	Trace decay parameter
e	Eligibility trace
V	Value function
P	Radial basis function
c	Centre of radial basis function
σ	Variance
t	Target
f	Acceleration
g_b	Gyro signal in the body frame
w	White noise parameter
∂	Partial derivative
D	Average of maximum position drift errors over a defined time period
P	Average of predicted position drift errors over a defined time period
$\delta p(t)$	Positional error drift after time t
$\delta p(t_0)$	Initial position error at the start of the outage
$\delta v(t_0)$	Initial velocity error at the start of the outage
Δt	Time difference between the start of the outage and the current time
$\delta b_a(t_0)$	Accelerometer offset bias at the beginning of the outage
$\delta b_g(t_0)$	Gyroscope offset bias at the beginning of the outage
g	Local gravity constant
$\delta\theta_{r,p}(t_0)$	Nonorthogonality error due to roll or pitch errors
$\delta\theta_A(t_0)$	Nonorthogonality error due to azimuth errors
V	Average velocity during the outage
$\delta_{SFa}(t_0)$	Accelerometer scale factor error at the beginning of the outage
$\delta_{SFg}(t_0)$	Gyroscope scale factor error at the beginning of the outage
τ	Correlation time

CHAPTER 1: INTRODUCTION

Exploration is inherent to human nature. We have constantly evolved, grown and moved our knowledge forward. The desire to think, invent and create is what makes us unique. This inherent nature has also driven humans to explore the boundaries of Earth, and when adventuring to new places we strive to know where we are going. The science of navigation has been constantly evolving, but what is interesting is that two very old concepts have largely been used, and improved upon: relative and absolute navigation.

1.1 Navigation in the past

Navigation and guidance have, throughout history, pieced together information from a variety of sensors. The concept of fusing triangulation methods with dead reckoning has been used for hundreds of years, dating back to the time of tall ships and a perceived flat Earth (Sobel, 1995). A rope with knots tied at intervals was used to estimate distance travelled over time, in a relative sense. When combined with other sensors such as a compass, an hourglass and a sextant, navigators could get rough estimates of position, heading and speed.

A sextant and compass were both used as triangulation devices. A sextant could triangulate between the known position of the sun, from calendars, and the known position of the horizon. The sextant used simple trigonometry along with the angle between the sun, itself and the horizon to determine its own position, or the final point on the triangle. With a

compass and two known points on land a navigator could draw lines along the two headings to triangulate position. Furthermore, the navigator could determine the boat's position twice in a discrete time period and determine approximate boat speed.

Whether it was a triangulation system or relative positioning system, the introduction of high precision clocks, known as marine chronometers, enabled a level of accuracy not previously possible. The first high precision clocks, invented by John Harrison, were used to guide ships across the Atlantic Ocean to North America. Although the precision of these clocks was no better than quartz oscillators found in common wristwatches today, they were such a revolutionary breakthrough that the English Government paid an equivalent of 20 million US dollars to Mr. Harrison who solved the problem of East-West navigation across Longitude (Sobel, 1995). The chronometer was designed to keep the local time in London, then each day when the local sun was highest in the sky (noon), it could be determined exactly how far away a navigator was from London. For example, if the clock showed that it was midnight in London at high noon locally, then the navigator was half way around the world.

Before the invention of an accurate chronometer, dead reckoning only navigators were the cause of many shipwrecks and lost lives. Still, dead reckoning updates were useful since they were not affected by one's ability to see the sun or stars through clouds. By combining daily absolute updates with continuous dead reckoning, more accurate and reliable navigators were created.

1.2 Navigation today

History often repeats itself, and with the introduction of the Global Positioning System (GPS) and silicon inertial measurement units (IMU's) the debate between triangulation versus relative positioning seems to have resurfaced. GPS receivers on the ground require a clear line of sight to the satellites orbiting the Earth, similar to the requirement imposed on celestial navigation. Contrary to GPS, IMU's sense accelerations and turning rates to which are sent to an external processor to provide relative position, velocity and attitude (PVA) information. An inertial navigation system (INS) is similar to an IMU but contains built in navigation algorithms that output PVA. Similar to accumulated marine dead reckoning errors, inertial navigation errors also grow with time. So once again, even with the most technologically advanced navigators, humans are still stuck with the same underlying problems faced by navigators hundreds of years ago: precise timing and measurement. Both the GPS and IMU designers have attempted to improve upon their technologies, but the union of the two is the logical choice for vastly improved reliability and accuracy. The concept of multiple sensors worked 400 years ago for accuracies of one hundred kilometres; now, using a similar solution, positions are being determined on the level of several centimetres.

Portable navigation devices have become common items found within vehicles or cell phones, and the advances in their reliability, accuracy and usefulness continues to expand the market of these devices. The GPS has been the backbone of such systems, but with more demand come higher expectations. Users are taking these GPS units into areas where

they were not designed to operate, and when failure occurs the results can range from simply frustrating to disastrous, in the case of emergency vehicles or valued asset location.

Many GPS designers have been creating new methods of acquiring the satellite signals in weak or degraded environments. Some success has been achieved with high sensitivity receivers, but the reliability of these systems is questionable and based largely on the environment. Although GPS has revolutionized modern navigation it cannot be relied upon as a primary navigator. Fortunately, when GPS is used with other sensors, such as IMU's, a more reliable and accurate navigation solution can be provided.

1.3 Research objectives

This thesis brings together GPS and inertial technology for low-cost vehicle navigation applications. Current benchmark integrations of these technologies can obtain position accuracies on the order of 1 to 5 metres, even when driving a vehicle in an urban setting. The issue with IMU/GPS integrations is that low-cost IMU's display position drifts which can exceed several hundred metres in just a few minutes without GPS updates. The integration methodology and error estimation can minimize the level of this drift, which is ultimately limited by errors imposed by the manufacturing of the device. The union of these technologies is not new but the integration approach proposed herein will address the following questions:

1. How does cost affect the hardware that can be used?
2. What sort of reliability and accuracy can be obtained when using low-cost GPS receivers and IMU's?

3. What are the best algorithms that can be used for this integrated navigator?
4. What are the limitations of these algorithms?
5. How can the accuracy and reliability of the system be improved?
6. How can usability of the system be improved?

These questions are quite general but they address the underlying problems of all navigation systems. The first three are background questions that address the current system, but questions four to six form the novel aspects of this thesis. The general limitations of the current algorithm are centred on the fact that the integration method has limited intelligence to adapt itself to varying conditions and changing error characteristics. Therefore, artificial intelligence or intelligent adaptation techniques are used to improve the current benchmark integration algorithm which uses the Kalman filter, a minimum mean squared error approach that uses *a priori* modelling.

The physical constraints of the device are unavoidable, but by modelling the errors more accurately the integration algorithm can help make the PVA solution more accurate and reliable. The issue of reliability is almost more important at this stage of IMU/GPS development. If accuracies of 1 to 5 metres can be maintained in any environment then a plethora of everyday applications can be enabled such as indoor tracking, automated parking, smart advertising, and child monitoring, just to name a few.

1.4 State of the art

The art of INS/GPS navigation has changed due to advances in inertial manufacturing technology. This has created sensors that can be packaged with a GPS receiver/antenna for everyday use such as phone/vehicle location, asset tracking and even new applications such as simultaneous location and mapping (Mannings, 2008). These new manufacturing technologies are creating inertial sensors that are smaller and consume less power, but they also exhibit much larger errors than their higher priced counterparts. This has lead the research community to find new and more efficient ways of weighting the sensor signals with wireless GPS signals.

Kalman filtering has been the backbone of most successful INS/GPS applications. Furthermore, adaptive Kalman filters such as Basil et al. (2004), Hu et al. (2003) or Khosla,(2004), and advanced correlation methods, Nassar (2003), have shown to be useful when using medium grade inertial sensors combined with differential GPS. Tuning procedures using least squares have also shown some promise for off-line adjustment of the Kalman filter, see Akesson et al. (2007) and Bolognani et al. (2003).

Following the adaptive Kalman filter, the unscented Kalman filter (UKF) was introduced to aid with nonlinear dynamics and faster alignment (El-Sheimy et al., 2006). Results for land vehicles were found to be nearly identical to an extended Kalman filter, but alignment times have been shown to be significantly improved (Shin, 2005).

More recently, artificial intelligence has been used to estimate errors after deployment of the integrated system. This type of integration does not require a system model and is less dependent on designer input. Chiang (2004) and Forrest et al. (2000) replaced the Kalman filter with a neural network as the data integrator. Unfortunately, despite their ability to form a model on-line, neural networks require a learning stage and their estimates can only be as good as their training references. Ojeda et al. (2004) used an expert rule based system to fuse the INS/GPS data, but this system is limited to specific applications which have been well defined by the expert designer. For these reasons several researchers have suggested combining neural networks with Kalman filtering, see Haykin (2001), Abdel-Hamid (2005) and Goodall (2006). A Kalman/neural/fuzzy system is the first proposed integration method in this thesis, and is novel in the aspect that a fuzzy system is used to provide reliability to a neural/Kalman learning/estimation system. Chapters 4 and 5 cover the neural/Kalman and fuzzy systems respectively.

These intelligent systems can be used to compensate the state estimates of the Kalman filter, but a more direct solution can be taken: the *a priori* parameters used in the filter can be adjusted intelligently or through estimation theory. Korniyenko et al. (2005) used a neural network approach to tune the Kalman filter, while Pasadyn (2004) used off-line state estimation for the *a priori* quantities and Soloviev and Grass (2007) used a frequency based batch processing to remove additional error sources.

Intelligent tuning is one of the newest areas of research. Since tuning involves changing unknown error characteristics a truth signal cannot be used for learning, as is the case with

neural networks. Self-organizing maps, as in Kohonen (2001) and reinforcement learning, found in Sutton and Barto (1998) are two of the most recent approaches to artificial learning without a truth signal. McClelland and Rumelhart (1988) was one of the first references to deeply analyze the issue of machine control using reinforcement learning. Reinforcement learning was used by Buijtenen et al. (1998) to control the attitude of a satellite. A more obscure publication, Macias and Esposito (2006), used a form of competitive learning to self-tune a Kalman filter. The application of reinforcement learning for tuning a Kalman filter is an entirely new approach and is investigated in details in Chapter 6 of this dissertation.

Finally, it should be mentioned that INS/GPS integration can be improved by providing additional aiding signals or constraints. Attitude updates to the filter can help prevent divergence through poor observability (Skaloud, 1995). Attitude from GPS can be derived from multiple receivers, as in Lu (1995), and then used in the filter to update the inertial estimates (Strus et al., 2008). Velocity constraints, given in Niu (2007), heading constraints in Shin (2005) and zero velocity updates can all be used to further help observability and improve the filter estimates.

1.5 Contributions

Primary contributions of this work are towards the advancement and adoption of low-cost GPS and inertial navigation systems. Many error modelling simplifications are made by designers of low-cost solutions which have to be adjusted in an on-line and intelligent way

after the unit is deployed. Furthermore, the methods developed herein can be applied to a variety of integration techniques. The specific contributions are:

1. A neural method that compensates position state estimates to improve accuracy based on dynamic and temporal learning inputs.
2. A fuzzy reliability system that is used to control the neural compensations. This method of fuzzy reliability can be applied to any system employing neural networks as a control mechanism.
3. A reinforcement learning technique that adjusts *a priori* filter parameters based on relative improvements or degradations to position state estimates and consistency measures of the filter. This provides direct feedback to the filter, which is not possible with neural networks since there is no known solution of the *a priori* parameters.

1.6 Thesis outline and block diagram roadmap

The next chapter brings together the current benchmark integrated system that uses a Kalman filter. Chapter 2 also introduces the sensors used in this research and discusses advances in Micro-Electro Mechanical Systems (MEMS) IMU technology. The chapter finishes with a discussion of the advantages and disadvantages of the current Kalman filter and introduces adaptive Kalman filtering.

Chapter 3 moves into novel intelligent methods which address many of the disadvantages experienced with the current Kalman filter. This chapter introduces three distinct types of intelligent algorithms that are used extensively in later chapters: neural networks, fuzzy

inference systems and reinforcement learning. Specific details of how each algorithm is applied to the Kalman filter problem are discussed in later chapters.

The fourth chapter goes into details on a method to compensate residual Kalman filter errors during GPS outages using neural networks. This method compensates for errors after they have propagated through the Kalman filter and it requires training before it can be used reliably. It is a very non invasive approach to correcting the Kalman filter, but provides little insight into what is happening in the filter.

This issue of reliability leads the thesis into the fifth chapter which adds a fuzzy inference monitoring system to add reliability to the neural network compensations. Results of the entire system are provided at the end of chapter 5.

Chapter 6 changes the focus and attempts to solve the same problem but before the errors have been accumulated through the filter. Reinforcement learning is used to adapt the *a priori* stochastic parameters required by the Kalman filtering algorithm. Simulation results are shown first, followed by real examples using both loosely and tightly coupled extended Kalman filters. A baseline approach using a serial tuning method is also discussed for comparative purposes.

Chapter 7 summarizes the results from the previous three chapters so that conclusions and recommendations can be drawn in Chapter 8. Figure 1.1 provides a visual architecture of individual chapters and their contributions to the thesis.

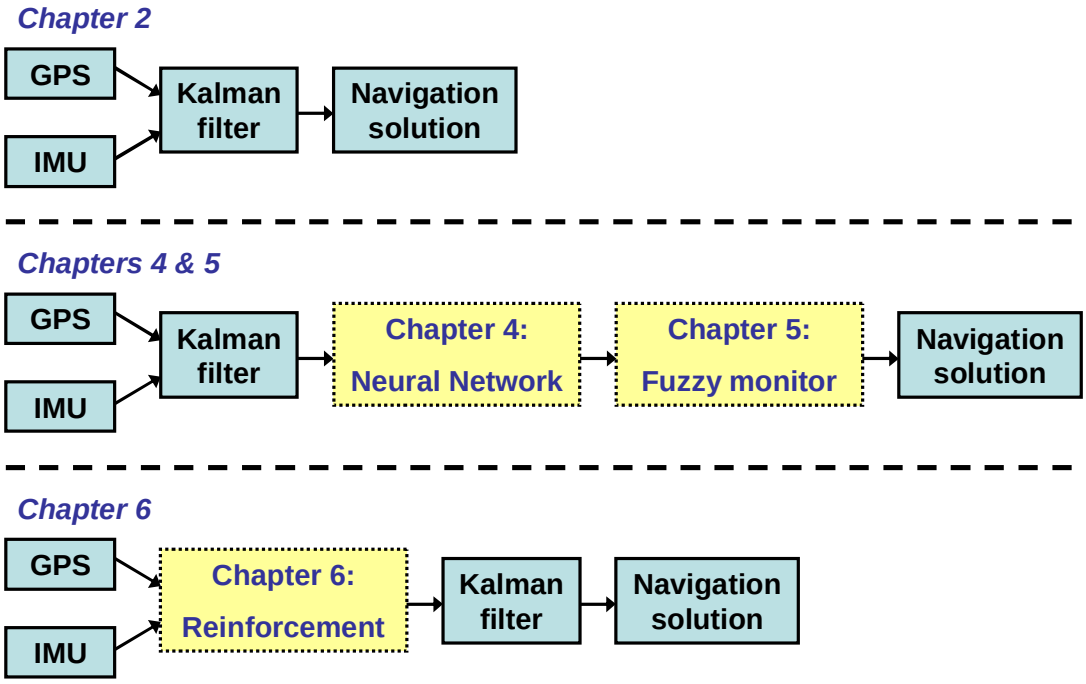

Figure 1.1: Block diagram roadmap

CHAPTER 2: NAVIGATION AND POSITIONING

Navigation comprises the methods and technologies to determine the time varying position and attitude of a moving object. Position, velocity and attitude, when presented as time variable functions, are called navigation states because they contain all necessary navigation information needed to georeference a moving object at any moment in time. In cases where only the position of the moving object is required, the term positioning, instead of navigation, is used. Positioning is commonly used in place of navigation since most modern navigators actually position or orient a platform at a discrete time, and from joining multiple discrete time periods navigation information can be generated. Global Navigation Satellite Systems (GNSS) are one such class of navigation system that provides absolute positioning information through trilateration from known points. IMU's also provide discrete updates, but these are relative and need to be mathematically integrated to provide positioning or orientation information in reference to a known starting point.

2.1 The global positioning system

The most successful implementation of a GNSS to date is GPS. In 1968 the US Department of Defense (DoD) issued requirements for worldwide location of its military forces. The response to this was the birth of NAVSTAR GPS in 1973 (Kaplan, 1996). With its full operational capability in July of 1995, GPS brought about large scale changes to the navigation community. It was initially adopted primarily for military use, but its usefulness quickly extended into the civil community. Vehicle and personal navigation,

asset tracking and weather prediction are just several of the numerous applications that have been revolutionized with the advent of GPS.

The GPS satellite constellation together with a receiver on the ground basically does the same thing as a sextant or compass, but it does so automatically, in real-time and with much greater precision due to accurate timing devices. GPS provides absolute positioning information to a receiver on the ground through trilateration of signals from satellites with known positions in space. In addition, each GPS satellite has a very precise timing device, such as an atomic clock, which enables speed and bearing information to be inferred. With a nominal constellation of over 20 satellites, GPS is capable of providing worldwide, continuous positioning. The concept of trilateration is similar to terrestrial positioning, but the satellites move through space so the geometry changes as a function of time. This means that to solve for a three dimensional position, four observations are needed since there will always be clock timing differences between the receiver and the satellites. The general concept of GPS trilateration, with 4 satellites, is shown in Figure 2.1.

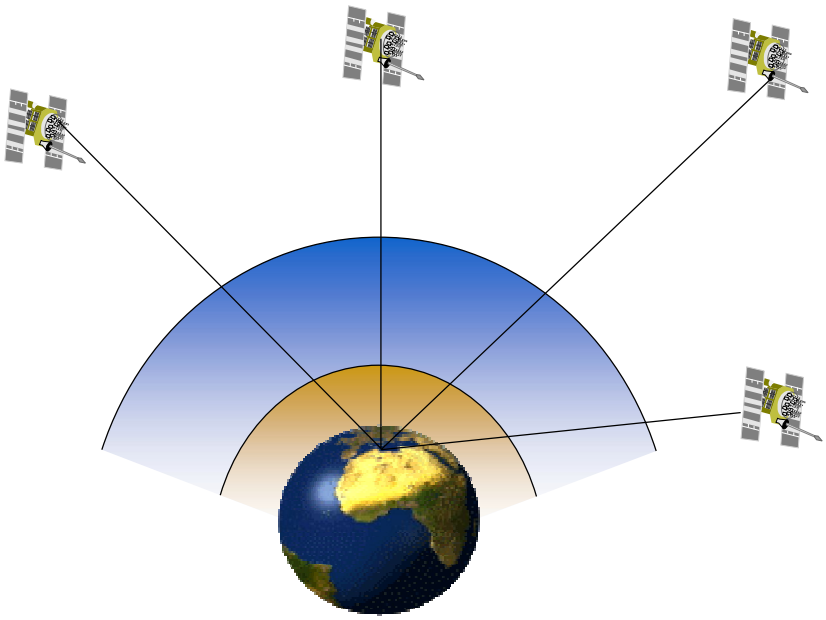

Figure 2.1: GPS trilateration

The GPS signal structure is important in understanding positioning accuracies. A carrier wave and a modulated Pseudo Random Noise (PRN) code are transmitted from each satellite. There are two different modulation codes used for pseudoranging: the C/A code, open to all users, and the P code, encrypted for US military use only. These are legacy signals, with newer signals, such as L2C and L5, soon to be realized. The chipping rate of the C/A and P codes is what distinguishes them. The C/A code has a chipping rate of 1.023 MHz and repeats every 1 ms, while the P code has a chipping rate of 10.23 MHz and repeats every 267 days. For this reason, and since the P code is encrypted, the C/A code is most often used for civil positioning applications (Misra, 2001).

There are also different frequencies of the carrier wave. The current signals being broadcast are the L1 (1575.42 MHz) and L2 (1227.60 MHz) carriers. The L5 (1176.45 MHz) carrier will be broadcast in the future along with L2C. Different frequencies are

useful for mitigating large ionospheric errors, thus improving the positioning accuracies available when using a single GPS receiver.

The carrier waves are processed to obtain more accurate positioning, while the codes provide rough position fixes. The term pseudoranging results from the use of these codes to get an approximate range measurement between a receiver and satellite. There are numerous errors inherent to the pseudorange signal. Orbital, satellite clock, receiver clock, ionospheric, tropospheric, noise and multipath errors are all unavoidable. The carrier phase observation has similar errors, but is also affected by a cycle ambiguity term (N) that is an integer number of unknown carrier wave cycles between the receiver and the satellite. Most very low-cost civil GPS receivers only process the C/A code as ambiguity resolution can be a difficult, time consuming and computationally intensive task. Carrier phase smoothing of pseudoranges has been shown to be an effective method of filtering pseudorange noise (Hatch and Knight, 1995).

Different modes of GPS can be used depending on the application and infrastructure available. GPS can be used in differential (DGPS), using two local receivers, or single point (SGPS) mode, which uses only a single local receiver. Differential mode uses differencing (single, double or triple) between a rover and a reference receiver to eliminate or reduce many of the errors inherent to single point mode carrier phase processing. **Error! Reference source not found.** summarizes these carrier phase differencing modes and lists the errors that are minimized (X) or reduced. Blank entries in this table indicate an unchanged error term.

Table 2.1: GPS error removal through differencing

		Error Source					
		Orbital	Atmosphere (ionosphere & troposphere)	Satellite clock	Receiver clock	N	Noise + multipath increase
Differencing mode	Between satellite SD				X		$\sqrt{2}$
	Between receiver SD	reduced	reduced	X			$\sqrt{2}$
	Between time SD					X	$\sqrt{2}$
	Satellite & receiver DD	reduced	reduced	X	X		2
	Triple difference	reduced	reduced	X	X	X	$2\sqrt{2}$

Although DGPS can remove many of the additional error sources, the practicality of this mode is limited to higher precision applications such as surveying or precision farming. The use of DGPS, using a local base station, in civil navigation applications is limited due to infrastructure and signal requirements. Therefore, SGPS is the mode of choice for large scale use leveraged to millions of users.

It should be noted here that the use of single point GPS is often augmented with Wide Area Differential GPS (WADGPS) systems. These differential updates are provided through a network of stations set up over thousands of kilometres and are sent to individual users with some time latency, typically on the order of several seconds. The reference stations are used to decompose the errors into satellite clock, ephemeris and ionosphere errors. The

corrections are then broadcast to individual receivers from geostationary satellites. These types of differential corrections are typically referred to as correction vectors, as opposed to scalar corrections in DGPS (Misra, 2001). These WADGPS systems do improve the positioning accuracy to within a metre or two, but they cannot provide centimetre level positioning as obtained in DGPS with two local receivers. Most new receivers are equipped with appropriate hardware to process these wide area corrections, with systems such as the United States Wide Area Augmentation System (WAAS) being the most popular in North America since no additional infrastructure imposed upon the user.

2.2 Inertial measurement units

The modernization of GPS, along with new GNSS platforms such as Galileo will bring about some important improvements to navigation. But these improvements do not change the fact that GNSS positioning requires line of sight between the satellites and the receivers. For this very reason, IMU's can be used to complement GNSS when the satellite signals are blocked, attenuated, reflected or interfered with in harsh environments (Li, 2008).

IMU's are also convenient since they can provide orientation information along with a platform position at a high data rate, whereas GPS is typically used for position fixing. So IMU's can be used to locate a point on a platform through dead reckoning, and they can also provide a time variable rotation component.

The basic principles of navigation using inertial sensors rely on Newton's laws of motion. The 1st law states: "*an object at rest tends to stay at rest and an object in motion tends to stay in motion with the same speed and in the same direction unless acted upon by an unbalanced force*" " (Lawrence, 1998). Accelerometers attempt to measure this unbalanced force through application of Newton's 2nd law: "*the acceleration of an object as produced by a net force is directly proportional to the magnitude of the net force, in the same direction as the net force, and inversely proportional to the mass of the object*" (Lawrence, 1998). A one dimensional accelerometer formed of a proof mass and two springs is shown in Figure 2.2. Acceleration along this dimension will move the proof mass in proportion to the acceleration.

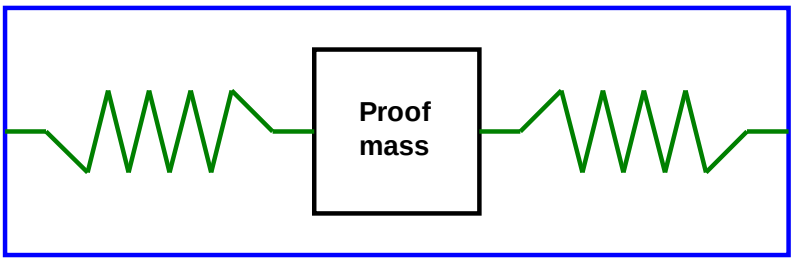

Figure 2.2: One dimensional accelerometer

Gyroscopes are used to measure angular rates, which provide angular changes when integrated. Nearly all low-cost gyroscopes use proof masses as vibrating mechanical elements (Nasiri, 2005). These types of gyroscopes sense the transfer of energy between two vibratory modes caused by Coriolis acceleration. Coriolis refers to an apparent acceleration that is proportional to the rate of rotation when viewed from a rotating frame of reference as shown in Figure 2.3.

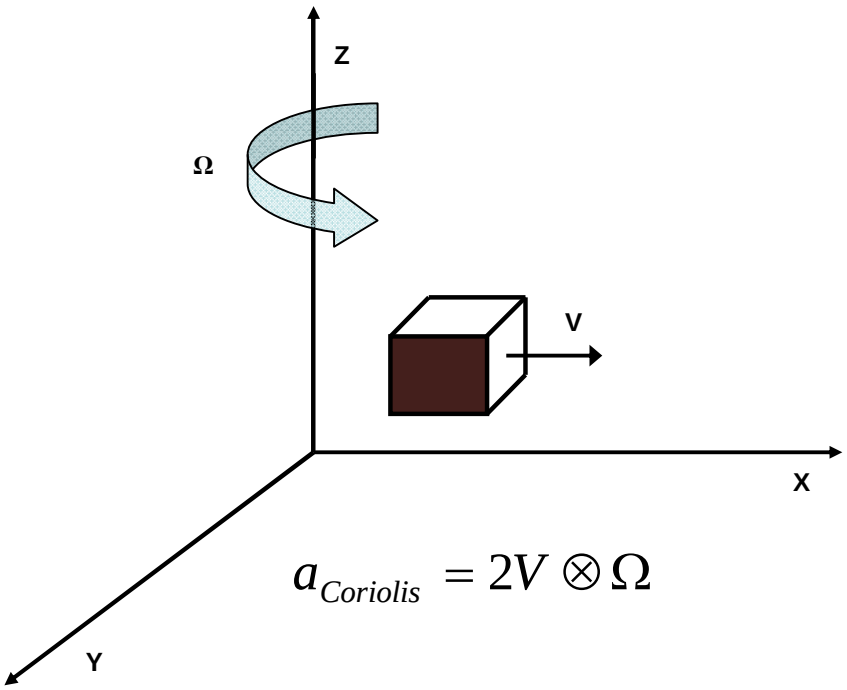

Figure 2.3: Coriolis acceleration

The frame of measurement is of key importance when using inertial sensors. When an accelerometer senses acceleration in the inertial frame it can simply be integrated twice to obtain an incremental displacement,

$$d = d_0 + v_0 t + \int\int a dt dt \quad \textbf{(1)}$$

In this equation,

d is the displacement

d_0 is the initial displacement

v_0 is the initial velocity

a is the acceleration

t is the elapsed time

Unfortunately this frame of reference is seldom useful since the platform position is typically located on the surface of the Earth, not that of the theoretical inertial frame which has no rotation or acceleration. This means these inertial measurements have to be transformed to the navigation frame for a strapdown system before integrating with respect to time.

The most common reference frames used in inertial navigation are the inertial frame, the Earth centred Earth fixed (ECEF) frame, the local-level frame (LLF) or navigation frame, and the body frame. The inertial frame (i-frame) is an idealized reference that directly follows Newton's 1st and 2nd laws of motion, having no rotation or acceleration, such as in a state of free fall. This ideal is not possible in reality so the common version of the i-frame has its origin at the centre of the Earth with non-rotating/accelerating axes with respect to distant parts of the universe. The i-frame has its z axis parallel to the rotation axis of the Earth (polar axis), its x axis in the direction of the mean vernal equinox, and a y axis that completes a right handed orthogonal frame.

The ECEF frame (e-frame) is a more commonly used frame of reference for navigation applications. The WGS-84 e-frame has its x axis in the equatorial plane pointing toward the Greenwich meridian, its y axis also in the equatorial plane but 90 degrees east of the Greenwich meridian, and its z axis along the Earth's polar axis. Since the e-frame origin is also at the centre of the Earth, the only angular velocity between the two frames is the

Earth's rotation. This rotation, given in the e-frame and rotated from ECEF to inertial, can be expressed as,

$$\omega_{ie}^{e} = (0,0,\omega^{e}) \quad \textbf{(2)}$$

The Earth rotation is a known constant and in WGS-84 is given as 7.292115147e-5 rad/sec (Misra, 2001). Considering Figure 2.4, the Cartesian coordinates of P (x,y,z) are given by,

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (N+h)\cos\varphi\cos\lambda \\ (N+h)\cos\varphi\sin\lambda \\ (N(1-e^{2})+h)\sin\varphi \end{bmatrix} \quad \textbf{(3)}$$

Where N is the radius of curvature in the prime vertical, e is the first eccentricity of the ellipsoid, h is the altitude, φ is the latitude, and λ is the longitude (Misra, 2001).

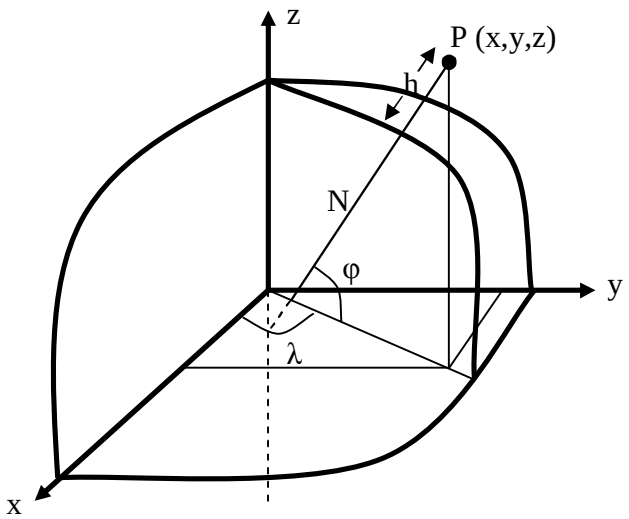

Figure 2.4: Cartesian reference frame

The navigation frame (n-frame) is also referred to as the local-level frame (l-frame). The origin of this frame is centred with the inertial hardware, which is typically found on the

surface of the Earth for a vehicle navigation system. The axes of this frame correspond to WGS-84 east, north and up (ENU) on the Earth ellipsoid. The transformation between the e-frame and l-frame involves two rotations. The first rotation aligns the up axis with the e-frame Z axis. The second rotation brings the east axis in line with the X axis. A rotation matrix, from the l-frame to the e-frame, can be formed as follows,

$$R_l^e = \begin{bmatrix} -\sin\lambda & -\sin\varphi\cos\lambda & \cos\varphi\cos\lambda \\ \cos\lambda & -\sin\varphi\sin\lambda & \cos\varphi\sin\lambda \\ 0 & \cos\varphi & \sin\varphi \end{bmatrix} \quad \textbf{(4)}$$

The body frame (b-frame) is common when using IMU's, since many low-cost civilian applications use strapdown sensors. Its origin is at the centre of the inertial hardware assembly. Its orientation is usually in line with that of the vehicle, but small misalignments are inevitable. The axes of the body frame are aligned with the pitch, roll and heading axes of the assembly.

The angular velocity between the l-frame and the b-frame is expressed through the pitch, roll and heading angles. The heading, pitch and roll angles are positive if they are eastward, rightward and upward respectively. To transform to the l-frame three successive rotations are needed:

$$R_b^l = R_3(A)R_2(r)R_1(p) \quad \textbf{(5)}$$

$$R_b^l = \begin{bmatrix} \cos A\cos r + \sin A\sin p\sin r & \sin A\cos p & \cos A\sin r - \sin A\sin p\cos r \\ -\sin A\cos r + \cos A\sin p\sin r & \cos A\cos p & -\sin A\sin r - \cos A\sin p\cos r \\ -\cos p\sin r & \sin p & \cos p\cos r \end{bmatrix}$$

The computation of this rotation matrix from measured angular velocity measurements is given in the next section after mechanization has been explained.

2.2.1 Integration equations

Mechanization refers to the integration equations required to transform the measured accelerations and angular rates to positions, velocities and attitudes. The measured accelerations are integrated with respect to time to provide changes in velocity and position. The gyroscopes are used to orient the accelerometers with respect to a defined frame. Mechanization can be performed in any frame of reference, but the most common is the l-frame since it outputs roll, pitch and azimuth directly; the horizontal errors are bound due to Schuler's effect, producing a Schuler loop (Groves, 2008).

In the l-frame the position of a moving platform is expressed in terms of φ, λ and h. The velocity components are expressed along the ENU directions. The attitude components are equated by solving the time derivative of the transformation matrix between the b-frame and l-frame. The solutions to the position, velocity and attitude mechanization equations, respectively, are (Abdel-Hamid, 2005):

$$\dot{r}^{l} = \begin{bmatrix} \dot{\varphi} \\ \dot{\lambda} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{(N+h)\cos\varphi} & 0 \\ \frac{1}{(M+h)} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V^{e} \\ V^{n} \\ V^{u} \end{bmatrix} \quad \textbf{(6)}$$

$$\dot{V}^{l} = R_{b}^{l} f_{ib}^{b} - (2\Omega_{ie}^{l} + + \Omega_{el}^{l})V^{l} + g^{l} \quad \textbf{(7)}$$

$$\dot{R}_b^l = R_b^l(\Omega_{ib}^b - \Omega_{il}^b) \quad \textbf{(8)}$$

In these equations, M is the meridian radius of curvature, f is the specific force measured by the accelerometers, g is the Earth's local gravity vector and Ω is a skew symmetric matrix.

To actually solve these equations, the rotation matrix between the b-frame and l-frame has to be parameterized. Two common transformations are direction cosines and quaternions. Quaternions are generally considered the better transformation since they are computationally simple and contain no singularities. Direction cosine matrices (DCMs) are still commonly used since the transformation matrix can be calculated directly and there are no singularities, but they are computationally expensive (Altmann, 1986).

The DCM representation has a linear relationship between the elements of the rotation matrix and the measured angular rates,

$$\dot{R} = R\Omega \quad \textbf{(9)}$$

Or, in recursive discrete form,

$$R_{t+1} = R_t e^{\Omega \Delta t} \quad \textbf{(10)}$$

This forms a system of nine linear differential equations for the DCM. In comparison, the quaternion representation requires the solution of only four linear differential equations. Quaternions use Euler's theorem, which states that the rotation matrix can be expressed by a single rotation about a fixed axis. Four parameters are used to determine the rotation matrix, one for the rotation angle and three to define the direction cosines of the relative axes. The quaternion parameters are,

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \frac{\theta_x}{\theta}\sin\frac{\theta}{2} \\ \frac{\theta_y}{\theta}\sin\frac{\theta}{2} \\ \frac{\theta_z}{\theta}\sin\frac{\theta}{2} \\ \cos\frac{\theta}{2} \end{bmatrix} \quad \textbf{(11)}$$

The rotation angle is,

$$\theta = \sqrt{\theta_x^2 + \theta_y^2 + \theta_z^2} \quad \textbf{(12)}$$

and the three direction cosines with respect to the local level frame are:

$$\frac{\theta_x}{\theta}, \frac{\theta_y}{\theta}, \frac{\theta_z}{\theta} \quad \textbf{(13)}$$

To compute attitude using quaternions they are considered functions of time with differential equations defined by,

$$\dot{q} = \frac{1}{2}\Omega(\omega)q \quad \textbf{(14)}$$

where ώ contains the x, y and z angular velocities of body rotation measured by the gyroscopes after removing the l-frame change of orientation and Earth rotation. Euler's method, also known as the constant slope method, can be used to solve the above differential equation (Spiegel, 1981). This method numerically solves the following first order differential equation,

$$\dot{y} = f(x, y) \quad \textbf{(15)}$$

In a numerical integration the starting point is known. So the question becomes: given the solution of the above differential equation is such that y is equal to c where x equals a, then

what is the value at x equals b? By integrating the equation with respect to x and replacing x with a,

$$y = c + \int_a^b f(x, y)dy \tag{16}$$

The constant slope method uses the approximation that n equal parts subdivide the space from x = a to x = b. Each interval is a fixed length h,

$$h = \frac{b - a}{n} \tag{17}$$

So now the integral becomes,

$$y = c + \int_a^{a+nh} f(x, y)dx \tag{18}$$

And if only one step is used (n = 1) and the slope is assumed constant the approximation is,

$$y = c + hf(a, c) \tag{19}$$

So returning to quaternions, the constant slope solution would be,

$$q_{k+1} = q_k + \left(\frac{1}{2}\Omega(w_k)q_k\right) \tag{20}$$

Finally, the rotation matrix can be determined using these quaternion parameters at a specific time,

$$R_b^l = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 - q_3q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_1q_2 + q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 - q_1q_4) \\ 2(q_1q_3 - q_2q_4) & 2(q_2q_3 + q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{21}$$

For more information on quaternions and attitude parameterization refer to Altmann (1986) and Shin (2005).

2.2.2 IMU errors

Any real measurement instrument is prone to errors. IMU's contain many possible error sources, with new errors and error models being a popular research topic; see Farrell (2007). In general, IMU errors can be separated into deterministic and stochastic. Different grades of inertial units contain different levels of these errors, but no sensor is immune to any of them. Common deterministic errors are the bias offset, scale factor error, non-orthogonality, gyro g-sensitivity, and scale factor non-linearity (Barshan, 1995). These errors are typically minimized by calibration before navigation, but special on-line methods can also be used; these are discussed in the following calibration section. The stochastic parts of inertial errors are short term instabilities of the sensor errors (noise), angular random walk (ARW), velocity random walk (VRW), and even bias and scale factor changes due to temperature effects. These stochastic errors are typically described by models such as random constant, random walk and the first order Gauss-Markov assumption (Gelb, 1974). Tables 2.2 and 2.3 give an extensive list of commonly used IMU errors, both deterministic and stochastic, for accelerometers and gyroscopes respectively.

Table 2.2: Accelerometer error terms

Accelerometer error term	Description and [typical units]
Bias offset	A systematic error obtained from a time average of the data (long-term). [g]
Velocity random walk	The velocity error build up with time that is due to integration of white noise in acceleration. A zero mean Gaussian stochastic process with a standard deviation that grows with the square root of time. [$m/s/hr^{1/2}$]
Bias instability	The noise on the bias offset. Also defined as the random variation of the bias offset over a time interval. [g]
Accelerometer triad misalignment from gyroscope triad	Assuming we have two perfectly orthogonal triads, we still might have a manufacturing misalignment between the triads. [deg]
Low-pass filter bandwidth	Typical hardware low pass filters the output. Any motion above the cutoff will be lost. [Hz]
Linear scale factor error	Linear change in the input to output ratio. [ppm]
Non-linear scale factor error	Non-linear change in the input to output ratio. [ppm of full-scale measurement]
Scale factor thermal drift	Linear change in the input to output ratio due to a temperature change. [ppm/˚C]
Bias thermal drift	The bias offset change due to a temperature change. [g/˚C]
Triad non-orthogonality	The misalignment between x, y and z axes of the triad of sensors in the body frame. [deg]
Measurement full-scale cutoff point	Maximum possible limit to measure an input signal. [g]

Table 2.3: Gyroscope error terms

Gyroscope error term	Description and units
Bias offset	A systematic error obtained from a time average of the data (long-term). [deg/hr]
Angular random walk	The angular error build up with time that is due to integration of white noise in angular rate. A zero mean Gaussian stochastic process with a standard deviation that grows with the square root of time. [deg/hr$^{1/2}$]
Bias instability	The noise on the bias offset. Also defined as the random variation of the bias offset over a time interval. [deg/hr]
Gyroscope triad misalignment from accelerometer triad	Assuming we have two perfectly orthogonal triads, we still might have a manufacturing misalignment between the triads. [deg]
Low-pass filter bandwidth	Typical hardware low pass filters the output. Any motion above the cutoff will be lost. [Hz]
Linear scale factor error	Linear change in the input to output ratio. [ppm]
Non-linear scale factor error	Non-linear change in the input to output ratio. [ppm of full-scale measurement]
Scale factor thermal drift	Linear change in the input to output ratio due to a temperature change. [ppm/˚C]
Bias thermal drift	The bias offset change due to a temperature change. [deg/hr/˚C]
Triad non-orthogonality	The misalignment between x, y and z axes of the triad of sensors in the body frame. [deg]
Measurement full-scale cutoff point	Maximum possible limit to measure an input signal. [deg/hr]
G-sensitivity	A gyroscope error response to an acceleration due to a mass unbalance from manufacturing. [deg/hr/g]

There are other IMU errors that expand upon this list but are often overlooked. Motion dependent degradations, such as g^2 sensitivity, rotation sensitivity and angular rate^2 sensitivity were commonly discussed decades ago, but are now less common when modelling IMU errors (Farrell, 2007). Thermal effects are receiving more attention from researchers and industry but aging and magnetic effects are rarely quantified. Another error source that has received little attention is vibratory motion errors. Despite vibrations

having a zero mean, they produce bias-like effects through rectification. These rectification errors add up in a consistent fashion when the dynamic conditions remain the same, but changes in the path or vibrations change the vector sum and in turn the effective bias contribution (Farrell, 2007). These unmodelled errors typically become attributed to other errors, such as time-varying biases, but their contribution is often unknown ahead of time or even after calibration.

There is even mention of errors that do not fall strictly into deterministic or stochastic models. Bandlimited drift is approximated by a constant for durations shorter than its autocorrelation time, but this only gives partial correction of the error source. A noise estimate is used outside the bands to limit error growth (Farrell, 2007).

Several researchers suggest state augmentation for error estimation. Unfortunately, due to the large number of possible error sources, the observability cannot be guaranteed. The system can become unstable with too many states and the real-time potential can be weakened (Nassar, 2003). States are very seldom directly observable due to limited vehicle dynamics, loss of GPS signal updates and limited data sets (Petovello, 2004). Models of the accelerometers and gyroscopes are often simplified to include the following terms (Yang, 2006),

$$I_f = (I + S_{a1} + S_{a2} f + S_{aT} \Delta T + N_a) f + b_a + b_{an} + b_{aT} + n_a \quad \textbf{(22)}$$

$$I_\omega = (I + S_{g1} + S_{g2} \omega + S_{gT} \Delta T + N_g) \omega + b_g + b_{gn} + b_{gT} + n_g \quad \textbf{(23)}$$

The terms used in these equations are defined as,

I	the measured signal for either the raw accelerometer (f) or the raw gyroscope (w)
f	the error-free accelerometer signal
w	the error-free gyroscope signal
b	the bias offset term for either the accelerometer (a) or gyroscope (g)
b_n	the bias instability
b_T	the bias term due to thermal effects
S_1	a matrix representing the linear scale factor errors
S_2	a matrix representing the non-linear scale factor errors
S_T	a matrix representing the scale factor linear error due to a temperature change
ΔT	the temperature change
N	a matrix representing the non-orthogonalities of the individual triad axes. N does not consider between triad misalignment.
n	a vector representing sensor noise that drives ARW or VRW

2.2.3 Calibration procedures

Calibration of inertial sensors involves the removal of as many error sources as possible before deploying the unit for navigation. Deterministic errors can be largely removed, while stochastic errors can be approximated using residual error models. To remove the deterministic errors the IMU outputs have to be compared to a reference. This type of reference can only be obtained by using special equipment such as a perfectly level table and a precision turn table. Techniques such as the six position static test, for

accelerometers, and angular rate tests, for gyroscopes, can then be applied to remove errors such as bias offsets and linear scale factor errors. These calibrations assume that aging is not an issue and the calibrated results apply for the life of the unit.

There are newer techniques that use more than six positions to calibrate the unit without specialized lab equipment. Shin (2001) developed a method that uses up to 26 different positions, of any orientation, that determines the bias, scale factor and non-orthogonality for a sensor triad. Regardless of the calibration technique used, time and processing are the main drawbacks to calibration of deterministic errors.

All further performance analyses of inertial systems are applicable to the remaining error after calibration (i.e. not accounted for), or due to time changing errors such as aging. Filtering techniques are employed in real-time to deal with such effects, which are often grouped into modelled errors such as a time varying sensor bias. Unfortunately, not all effects can be modelled by additional states (Titterton and Weston, 2004). These errors are often encompassed by stochastic error models, although they have deterministic properties to parameters that are not considered; this often results in pessimistic estimates of the stochastic processes.

2.2.4 Residual error models

Even if accurate techniques are used to determine the deterministic sensor errors, there still exist large uncertainties in many of the outputs (Shin, 2005). Sensor uncertainties need to be modelled to limit their growth and to predict filter performance; stochastic error models

are used for this purpose. The state vector of a navigation filter models some of these errors. Regardless of the filter used, these stochastic models are important aspects of the *a priori* information needed for the filter. Kalman filters, particle filters and least square estimators all need different forms of *a priori* information, with more accurate models yielding more accurate navigation results.

The inertial sensor bias and scale factors are often included in the state vector of Kalman filters. The choice of model for these states is dependant not only on the sensor performance, but also on the environment and operation time. For example, if the operation time is very short, on the order of several seconds, then a constant value for the state can be used, but if run for a long time, random walk and aging effects would change the value of the state.

The assumption of sensor errors having white noise characteristics is a common approximation. Another common model is that of random walk, whereby the white noise process is integrated. An INS does indeed integrate its signals, so the uncertainties of the velocities and attitudes increase and are described through the VRW and ARW terms.

Noise, VRW and ARW values are typically determined through Allan Variance analysis. Hou (2004) gives an excellent resource for this Allan Variance method. The underlying principles can be taken from Figure 2.5. The sensor noise, or bias instability, is found by the region of zero slope, while the random walks can be found where the Allan Variance slope is –1/2.

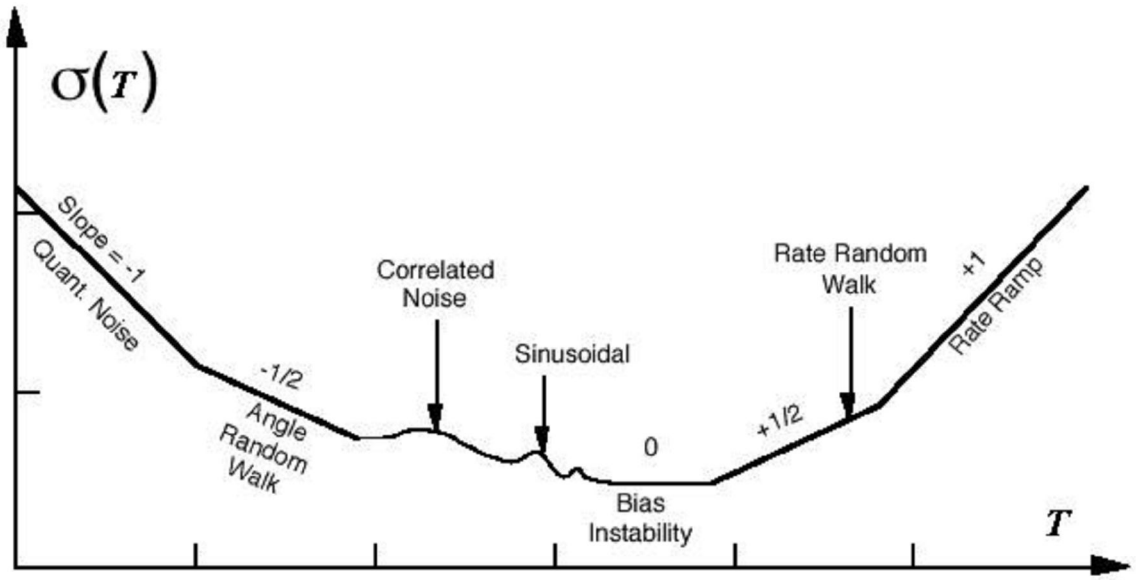

Figure 2.5: Allan variance analysis results (after IEEE Std 952, 1997)

The length of data is important when performing Allan Variance analysis. The number of clusters, or independent test results, combined with the total number of data points provides an estimate of the uncertainty in the measurements (Hou and El-Sheimy, 2003). IEEE Std 952 (1997) gives the following equation to estimate the percentage error in the Allan standard deviation,

$$\delta_{AV} = \frac{\sigma(T,M) - \sigma(T)}{\sigma(T)} = \frac{1}{\sqrt{2\left(\frac{N}{n} - 1\right)}} \tag{24}$$

Here, σ(T,M) is the estimate of the Allan standard deviation obtained from M independent clusters, and σ(T) is the theoretical value as M approaches infinity. The alternative form of the equation contains n, the number of data points in each cluster, and N, which is the total number of data points across all clusters. For 5% error the ratio of N divided by n would be

201. If n were large, say 5000, then the total number of data points required would be 1,005,000.

Another way to estimate how much data is required for accurate estimation of the noise levels is the autocorrelation function. Reliable determination of the autocorrelation also requires very long data collection periods. According to Brown and Hwang (1997) the accuracy can be represented as,

$$(accuracy)^2 \leq \frac{2T}{t_c} \quad \textbf{(25)}$$

T is the time constant of the process and t_c is the data collection time. If T is 1 hour and the required accuracy is 5% then the collection time would have to be greater than 200 hours.

2.2.5 Alignment

Initial orientation between the sensor body frame and the local level frame is defined as alignment. Alignment should be performed after calibration once the errors have been minimized since accurate alignment relies on error-free signals. In fact, since error-free signals can never be guaranteed, alignment can result in de-correlation errors for strapdown systems; these are discussed in Chapter 4. From the mechanization equations it can be seen that alignment refers to the initial definition of the R_b^l matrix. For very accurate inertial systems, such as tactical grade or better, with a gyroscope instability better than 10 degrees/hour, coarse and fine alignment methods can be applied (Brown and Lu, 2004). For lower grade sensors, the heading alignment (gyro compassing) cannot be performed

since the noise level of the gyroscopes is larger than the Earth's rotation signal (Li et al., 2008).

When the unit is static, accelerometer levelling uses the local gravity component to drive the output of the horizontal accelerometers to zero. Gyro compassing is performed after levelling so that the heading gyro, with its sensitive axis in the horizontal plane, can detect the North direction when its output is a maximum and a minimum when it points East.

For vehicle navigation systems, static alignment is often not possible due to constraints on the user. In-motion alignment uses the GPS derived velocity vector to perform a rough alignment. Roll is initialized to zero while pitch and heading are estimated as follows,

$$\theta = \tan^{-1}\left(\frac{-v_{down}}{\sqrt{v_{north}^2 + v_{east}^2}}\right) \quad \textbf{(26)}$$

$$\psi = \tan^{-1}\left(\frac{v_{east}}{v_{north}}\right) \quad \textbf{(27)}$$

In-motion alignment assumes the forward velocity of the vehicle is exactly parallel to the velocity vector. This assumption depends on how well the inertial unit is mounted within the vehicle. If poorly mounted, for example in the case of a cellular phone on a dash, the degradation can become severe. Syed et al. (2007) covers these degradations in details for a six degree of freedom MEMS inertial sensor assembly and Shin (2007) describes large heading uncertainty (LHU) methods for improved filtering results.

2.3 MEMS inertial sensors

The development of MEMS grade IMU's for navigation has brought about many new research and industrial developments. Their low-cost, small size and minimal power requirements make them excellent candidates for integration into small portable strapdown inertial navigation units. Their use as navigators has yet to fully penetrate the consumer market, but they are becoming more popular in vehicle and handheld units that often use GPS as a stand-alone navigator (Frost & Sullivan, 2006).

Within vehicle platforms, MEMS inertial sensors were made popular through airbag deployment. Recently, all new cars in the United States are required to be equipped with independent tire pressure sensors, which also use low-cost MEMS pressure sensors. Finally, MEMS gyroscopes are used in higher grade vehicles for advanced roll-over detection and vehicle all-wheel drive (AWD) and electronic stability control (ESC) systems.

The mobile handheld platform is also seeing an expansion of integrated MEMS sensors. Many new mobile phones are using MEMS accelerometers for mobile gaming and improved user interaction through screen orientation (portrait vs. landscape). Digital cameras and camcorders also use MEMS gyroscopes for image stabilization and these devices are now being integrated into smart phones. The applications for inertial sensors often overlap those of the traditional GPS dominated navigation units, with the next logical step being integration of these navigation units.

This integration has some serious hurdles that have to be overcome, such as calibration and alignment to the vehicle/platform frame. Alignment issues have been approached by many independent studies and is a topic of much current research; see Syed et al. (2007) or Chowdhary et al. (2007). But before alignment can be accurately tackled, calibration of the unit has to be performed. Calibration for these low-cost units is rarely done by the manufacturer. Very broad ranges are given for both deterministic and stochastic error sources, and fine-tuning is left to the designer. Unfortunately, when manufactured in mass quantity, it becomes too costly to calibrate individual MEMS inertial sensors. Poor calibration can severely degrade the performance of an integrated navigation unit (Grewal et al., 2001).

It is the intent of this thesis to explore intelligent methods that can adapt to the effects of poor calibration without input from the manufacturer, designer or user. Instead of imposing unrealistic constrictions on the user or manufacturer, calibration parameters are developed online while the unit is operating. Several different approaches will be considered and are discussed in the following chapters, with most emphasis going to the development of appropriate residual error model stochastic parameters. Of course residual deterministic errors are often combined into residual error modelling, but are removed over time through Kalman filter state estimation.

2.3.1 MEMS specifications

Some MEMS manufacturers provide stochastic information for the inertial error sources and these can be exploited when calibrating a unit. They can be used as a starting point for further calibrations and they can also be used as rough estimates of sensor performance when determining which sensors to choose since they represent averages for the hardware performances of thousands or millions of similar units. On average it would be expected that these units perform as stated by the manufacturer, although individual units might perform better or worse depending on the accuracy of individual calibrations.

Unfortunately, MEMS inertial sensor specifications lack the standards of higher grade sensors, so before this information can be exploited it first has to be understood. The vast array of terminology used by manufacturers is overwhelming, but when analyzed carefully these terms can be brought together to match the error sources defined in Section 2.2.2. An example of a MEMS gyroscope specification from Analog Devices Inc. (ADI) is given in Table 2.4.

Table 2.4: ADI gyroscope specification (ADXRS150)

		ADXRS150ABG			
Parameter	**Conditions**	**Min[1]**	**Typ**	**Max[1]**	**Unit**
SENSITIVITY	Clockwise Rotation Is Positive Output				
Dynamic Range[2]	Full-Scale Range over Specifications Range	±150			°/s
Initial	@25°C	11.25	12.5	13.75	mV/°/s
Over Temperature[3]	V_{CC} = 4.75 V to 5.25 V	11.25		13.75	mV/°/s
Nonlinearity	Best Fit Straight Line		0.1		% of FS
Voltage Sensitivity	V_{CC} = 4.75 V to 5.25 V		0.7		%/V
NULL					
Initial Null			2.50		V
Null Drift over Temperature[3]	Delta from 25°C			±300	mV
Turn-On Time	Power on to ±½°/s of Final		35		ms
Linear Acceleration Effect	Any Axis		0.2		°/s/*g*
Voltage Sensitivity	V_{CC} = 4.75 V to 5.25 V		1		°/s/V
NOISE PERFORMANCE					
Rate Noise Density	@25°C		0.05		°/s/√Hz
FREQUENCY RESPONSE					
3 db Bandwidth[4] (User Selectable)	22 nF as Comp Cap (See Applications section)		40		Hz
Sensor Resonant Frequency			14		kHz

In this table, sensitivity is another term for scale factor. The dynamic range refers to the measurement range, the initial scale factor is the nominal scale factor, over temperature refers to the temperature dependent scale factor error, non-linearity refers to the non-linear scale factor error, and the voltage sensitivity is just used for conversion purposes.

The term null refers to the bias parameters. The term initial refers to the nominal bias whereas over temperature refers to the bias change with respect to a temperature change. Turn-on-time is a parameter that refers to the time it takes for the bias to settle when the device is turned on. The linear acceleration effect is another term for g-sensitivity and the voltage sensitivity is used for conversions. The rate noise density defines the angular random walk and the frequency response defines the low-pass filter bandwidth (Oppenheim et al., 1997). Regardless of the manufacturer terminology the parameters can be converted to terminology or units as given in Table 2.3.

2.4 Kalman filtering

GPS and inertial navigators are complementary to one another; thus it makes logical sense to combine their measurements. An INS can provide a continuous solution at a high data rate since it is immune to jamming, but its errors are time dependent and it has no absolute time standard as a reference. GPS provides time-independent measurements and has an extremely accurate time standard, but GPS is sensitive to RF interference and signal blockage (Groves, 2008).

To minimize inertial error accumulation with time, GPS measurements can be applied. To optimally combine the two different measurements, a filtering technique needs to be employed. When combining sets of measurements, the Kalman filter is optimal in the minimum mean squared error sense and is unbiased. When combined with its real-time recursive and modelling capabilities, it is a natural choice for INS/GPS integration.

Dynamic inertial error state models can be derived for the various error states. Common error states include position, velocity, attitude, accelerometer bias, accelerometer scale factor, gyroscope bias, and gyroscope scale factor errors. These equations are used within the Kalman filter to define the dynamic model which is used to predict the next state. GPS measurements are then used to correct the prediction and constrain the growth of the inertial errors. There are of course errors in the GPS measurements, as discussed in Section 2.1, so the combination of the GPS updates and INS predictions have to be weighted

accordingly. Typically, GPS updates are coordinate updates, but derived GPS velocities and attitudes can also be used.

The Kalman filter equations are well documented, see for example Gelb (1974) or Brown and Hwang (1997), and will not be derived in detail here, but a short review will prove useful when discussing different forms of the filter. A basic recursive block diagram with equations is given in Figure 2.6.

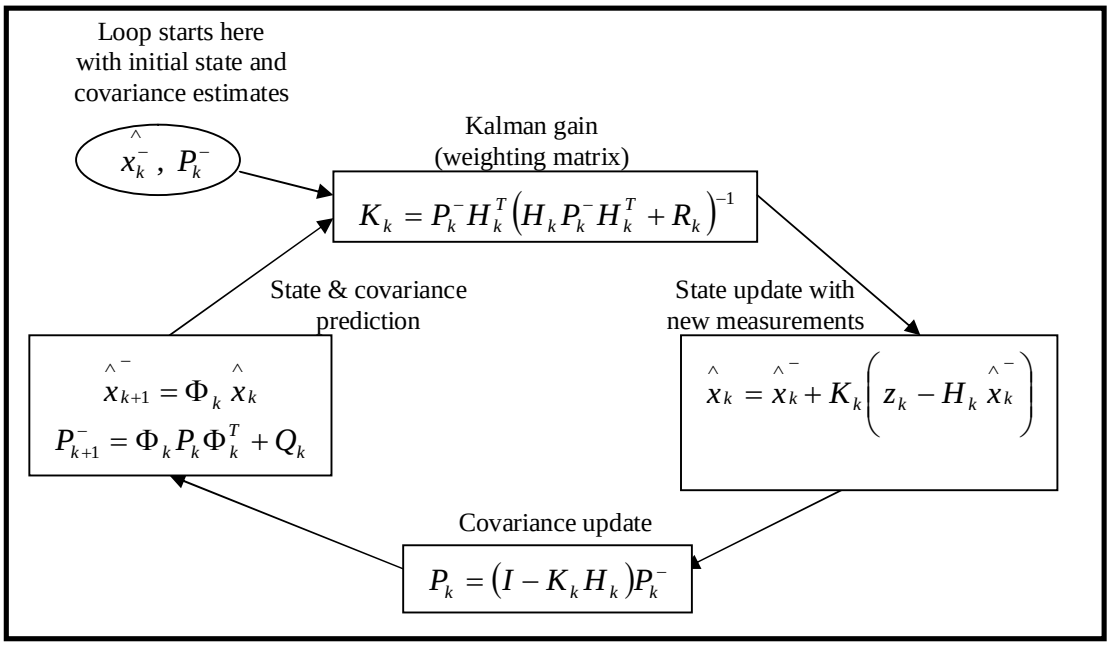

Figure 2.6: Kalman filter recursive equations (Brown and Hwang, 1997)

In the above equations, shown in Figure 2.6,

k is the time epoch

- is a superscript that denotes a prediction

ˆ denotes estimation

x is the state vector

P is the covariance matrix of the state

Φ is the transition matrix that uses the dynamic model to predict the next state

Q is the spectral density matrix that defines the noise of the dynamic model

K is the Kalman gain matrix that weights the predictions with the measurement noise

R is a matrix that defines the noise of the measurements, and

H is a matrix that defines the noiseless connection between the measurement and state vector

All noise terms are considered to be white sequences with known covariance. The dynamic system used for prediction is non-linear by definition, but the Kalman filter requires a linear set of differential equations to relate one state to another. An extended Kalman filter (EKF) performs a linearization of these equations about the predicted state vector. The EKF performs a Taylor series expansion of the non-linear measurement and system equations and truncates them to first-order approximations.

The state vector is in turn dependent on the architecture that is used to combine the GPS and inertial measurements; this is commonly referred to as coupling. Historically, there are three methods of coupling: loose, tight, and deep. The terms centralized and decentralized have also been used to refer to tight and loose coupling respectively (Gao et al., 1993). A loosely coupled architecture is the simplest to implement since the inertial and GPS

navigation solutions are generated independently before being weighted together in a separate Kalman filter known as the fusion Kalman filter. The advantages of this coupling strategy are that the INS errors are bounded by the GPS updates, the INS can be used to bridge GPS updates, and the GPS can be used to help calibrate the deterministic parts of the inertial errors online. Another advantage is that the loosely coupled architecture can be used to integrate existing GPS with available inertial systems, such as those currently found in vehicles or mobile phones, since it does not require access to the raw GPS signals. The loosely coupled integration strategy, using position and velocity updates from GPS, is shown in Figure 2.7. P stands for position, V for velocity, and A for attitude.

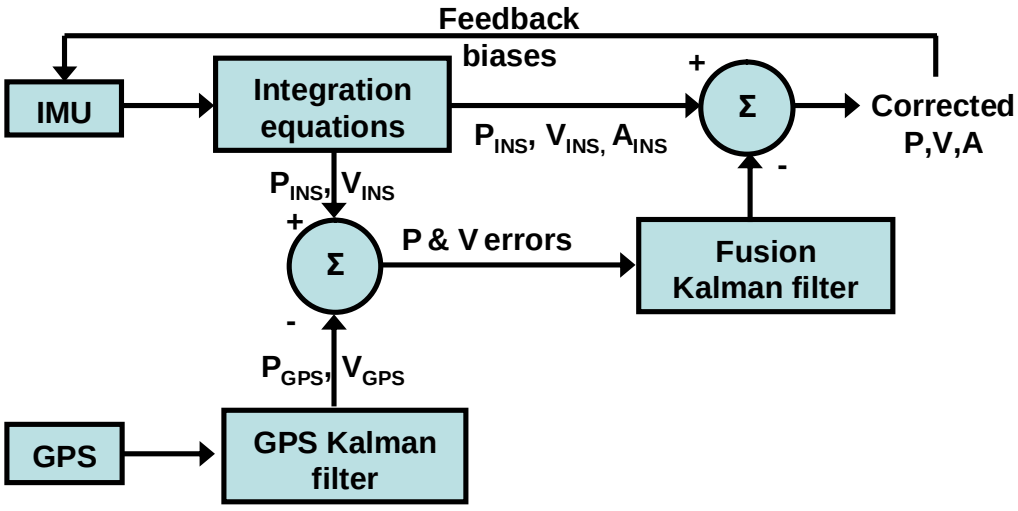

Figure 2.7: Loosely coupled architecture

The main drawback to a loosely coupled integration strategy is that it requires at least four satellites in view, or three for a horizontal solution. There are typically at least 6 to 8 GPS satellites in view at any time and anywhere on the Earth, but due to blockages such as trees or tall buildings, there could be significantly less at the antenna location. The tightly

coupled integration strategy alleviates this restraint by combining the integration into a single Kalman filter. Any number of GPS satellites can be used as measurements to limit the inertial errors from drifting. The comparison of parameters in the filter is different from the loosely coupled architecture. Instead of comparing positions and velocities, the tightly coupled architecture differences raw GPS pseudorange and Doppler measurements from those predicted by the inertial unit. The filter output is a correction to the inertial outputs as shown in block diagram format. The filter output can also be fed back to the GPS receiver in the form of INS derived velocity/Doppler information to aid the code and carrier tracking loops, see Petovello (2003) or Yang (2008).

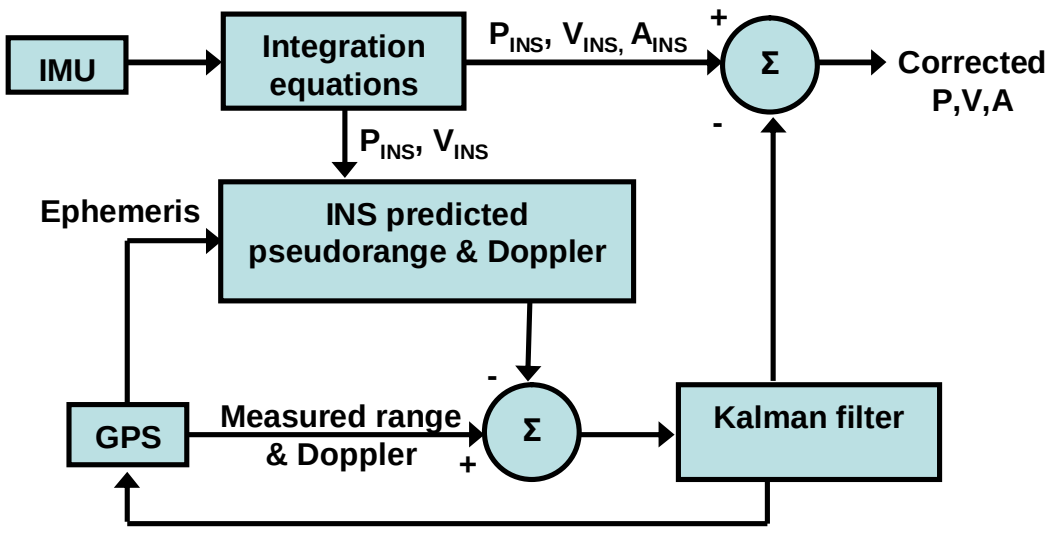

Figure 2.8: Tightly coupled architecture

Deep integration also combines measurements into a single Kalman filter, but combines GPS and inertial data at the earliest possible stage (Li, 2008). Immediately after the RF signal is shifted to some lower intermediate frequency (IF), inertial measurements are used

(Groves, 2008). The inertial data is used at this stage to provide a dynamic reference trajectory that aids the receiver correlators during signal integration. A replica GPS signal, both code and carrier, is generated by an integrated tracking and navigation Kalman filter (Soloviev et al., 2008).

Although the deeply coupled integration fuses inertial and GPS measurements at the earliest possible point, it suffers from implementation complications. A software receiver is needed to access the correlators, which is not yet a reality with most GPS receivers being hardware based. For tight integration, access to the raw pseudorange and Doppler is not as imposing, with many receivers providing these as standard outputs. The emphasis of this dissertation will be on calibration imperfections that can be corrected without the requirement of a software receiver. The loosely and tightly coupled integrations will be analyzed since they represent nearly all potential INS/GPS integrations within the current consumer vehicle and portable navigation markets.

2.4.1 *A priori* conditions

Although the Kalman filter is an optimal filter in the minimum MSE sense, it can only be optimal if certain conditions are met. The designer must first provide *a priori* deterministic and stochastic information. If this given information is incorrect or misleading then the filter is no longer guaranteed to give optimal results.

Dynamic and measurement models have to be supplied as deterministic data (Abeel et al., 2005). These are provided as the transition (Φ) and measurement design (H) matrices. The

filter also needs starting conditions, namely an initial state vector and an initial state covariance matrix P. These initial values do not affect the steady state optimality of the filter, but they do affect convergence time (Gelb, 1974). A large initial P matrix allows for large variability in the state estimation since P characterizes the uncertainty in the state. A large initial P essentially tells the filter that there is little confidence in the initial values of the state vector.

A priori stochastic data comes in the form of the measurement and system noise covariance matrices, R and Q respectively. R describes the noise on the measurements and the measurement model. Measurement modelling is often simplistic and usually ignores many of the non-white (eg. correlated) measurement errors. Correlated noise is often present when taking GPS measurements, especially in areas of large multipath (Kaplan, 1996). These unmodelled errors propagate in the Kalman filter and result in a state covariance (P) that is more optimistic than reality.

While the Kalman filter is only sensitive to initial conditions when converging, it can become suboptimal if there are errors in the design model or if key assumptions are violated, such as that of white measurement errors (Choukroun, 2003). The Q matrix defines how well the predictions can be trusted. The errors of the inertial sensors and the imperfections of the transition matrix are defined by white noise within the Q matrix. The Kalman filter does not restrict Q to a particular value to be optimal, but Q is critical for achieving practical results (Gelb, 1974). In an optimal solution it is desired to minimize MSE, but the filter also has to be consistent with results that are achieved in reality. This

means that the output of the P matrix should be sufficiently close to the actual errors the estimation is experiencing. The interaction of the R and Q matrices with that of the P matrix can be observed through equation (27), which is an alternate form of the equation given previously for the update of P.

$$P_k^{-1} = \left(P_k^-\right)^{-1} + H_k^T H_k \left(R_k\right)^{-1} \quad \textbf{(28)}$$

The updated state will be better than the previous state by the amount of information added by the measurement update, which is affected by R. The predicted state is affected by Q, so it can be seen that both R and Q determine the quality of the updated P, and to get practical results these matrices need to be set to values that are similar to those in reality.

Laboratory solutions for these parameters are sometimes possible but this involves an Allan Variance or autocorrelation analysis that can be very lengthy; several days or weeks is not uncommon (Hou, 2004). For low-cost MEMS these lab tests are not performed due to time and monetary restrictions; knowledge needed for tuning the dynamic model is almost never available in reality, especially for MEMS sensors (Vanicek and Omerbasic, 1999).

2.4.2 Adaptive Kalman filtering

There has been significant work done on adapting these noise matrices while the filter is used. One form of these adaptive Kalman filters (AKF) attempts to evolve the Q and R matrices by observing the whiteness of the filter innovation sequence (Mehra, 1970). This AKF is often referred to as the innovation-based adaptive estimation (IAE) AKF. The actual updates to R and Q are,

$$\hat{R}_k = \hat{C}_{vk} - H_k P_k^- H_k^T \quad \textbf{(29)}$$

$$\hat{Q}_k = K_k \hat{C}_{vk} K_k^T \quad \textbf{(30)}$$

where the innovation sequence and its covariance are,

$$v_k = z_k - H_k x_k^- \quad \textbf{(31)}$$

$$\hat{C}_{vk} = \frac{1}{N} \sum_{j=k-N+1}^{k} v_j v_j^T \quad \textbf{(32)}$$

The innovation sequence is simply the difference between the actual measurements and the predicted measurements based on the current, predicted, state vector. The covariance of the innovation sequence is used to detect abnormal behaviour by testing the assumptions of zero mean Gaussian white noise with a known covariance matrix (Grewal, 2001). N is simply a defined number of previous innovation samples which are used to form the covariance, determined by the designer of the AKF.

Unfortunately this approach requires a window length N, which is hard to define for real scenarios. For example, if a vehicle was driving in open sky and then quickly entered an urban canyon it would be expected to suffer from very different measurement conditions from the GPS. A long windowing period would tend to blend these changes into the previous open sky conditions, while a short window would change quickly to the conditions but would be prone to larger errors from blunders. This AKF also relies heavily on the initial Q and R values since bootstrapping is performed; the system noise covariance, measurement noise covariance, Q value and Kalman gain K are all functions of the previous state estimates (Mehra, 1970).

Another AKF, known as Multiple Model Adaptive Estimation (MMAE), runs a bank of Kalman filters in parallel, with each filter having different statistical parameters for R and/or Q (Brown and Hwang, 1997). Each filter provides its own estimate and *a posteriori* probabilities for each scenario. The final estimate is a weighted sum of all the filters based on the probabilities. The optimal solution in this case sets all weights to zero except the correct solution which is set to unity. The bank of filters does this by analyzing the sums of the weighted squared innovations.

The MMAE method is highly dependent on the number of filters that have to be run. If only two or three are run then real-time implementation would be possible. Unfortunately, with MEMS inertial sensors the unknowns cover such a wide range that many filters would need to be used to come close to optimality. This would not be possible for a real-time application.

AKF's have been shown to be superior to a conventional Kalman filter especially during high dynamics where the optimal R and Q parameters change. Unfortunately, the improvements rely upon proper choice of N in the innovation sequence covariance calculation and high quality DGPS data to form the innovation sequences. Also, the AKF adjusts the R and Q matrices, but often the actual dynamic model itself is in error.

The advantage of the MMAE method, over the IAE AKF, is that the MMAE does not perform bootstrapping and is less prone to instability. The Q and R values are independent

of the individual starting values since many filters are run in parallel. The tradeoff between real-time implementation and bootstrapping can be dealt with if the entire dataset is re-run using different *a priori* data over time. The methods outlined in the remainder of this dissertation deal with adapting either the state estimates of the Kalman filter or R and Q directly. In both cases intelligent adaptation methods are implemented that retain memory of previously learned scenarios. This avoids the need for a fixed window length, since memory from previous windows is maintained, and does not impose unrealistic real-time implementation conditions.

The next chapter will introduce the concepts necessary for various types of intelligent adaptation. Chapters 4 and 5 then develop methods for correcting the state estimates for poor stochastic parameters. Chapter 6 delves into more details and directly changes the R and Q matrices to provide better state estimates.

CHAPTER 3: ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) has become well accepted and documented with many real world examples such as intelligent gaming, weather prediction/modelling, and even vehicle manufacturing using automated robotic systems. The actual mathematics behind AI systems are not unlike that for other filtering algorithms; minimum mean squared errors are still key performance measures. Important differences are that intelligent algorithms often employ non-linear techniques that process many different tasks in parallel with feedback to one or more stages of the computation. AI has been developed to mimic human brains that seem to learn from many parallel interactions in a highly non-linear and complex manner (Haykin, 1994).

Biological neural networks are much further advanced than current artificial neural networks (ANN's). A simple example brings light to the difference. A bat uses sonar to detect flying insects. The bat's sonar and processing system is capable of detecting instantaneous velocity, size, azimuth and elevation of its target so accurately and quickly that few insects ever escape (Suga, 1990). Current radar systems that use highly complex parallel processing are still years away from the accuracy and capability of a bat's sonar. Yet a bat has a brain the size of a grape while the parallel computing environment would occupy a large room.

Although AI is still primitive in comparison to human intelligence, or even a bat, there are many interesting algorithms that apply to a wide variety of engineering applications. The specific algorithms developed herein do not represent new mathematical forms of AI, but their application to Kalman filtering for navigation applications is novel.

Neural networks are discussed first and used to compensate the Kalman filter state estimates in Chapter 4. They are used to form an input-output relationship using dynamic inputs and Kalman filter states as outputs. These states are another way to assess the performance of the system. They indirectly minimize the innovation sequence since the integrated drifts rely heavily on the last innovation before measurements are lost within the filter. Furthermore, this type of incremental learning algorithm is capable of adapting as new data is presented for real-time application.

Fuzzy inference systems will then be applied to improve reliability of the neural compensations in cases where the learning is not converged. Fuzzy logic is well suited to monitoring the accuracy of an ANN since the designer of the fuzzy system can include rules that govern the application of the neural learning.

Reinforcement learning (RL) will then be introduced as an alternative approach to the stochastic tuning problem where there are no truth signals that can be used to guide the AI learning. This is the case when direct manipulation of the parameters is desired. RL will be used to adapt the R and Q matrices in a similar manner as an adaptive Kalman filter, but without the conditions of a fixed window length for learning or multiple parallel filters. In

fact, it will be seen that this method can be compared to the MMAE method running in series instead of parallel with an infinite number of filter possibilities.

3.1 Adding NOT replacing

Before moving on to specific methods of intelligently aiding the Kalman filter it needs to be stressed that this work attempts to add information to a typical Kalman filter solution, not replace the Kalman filter with another estimation tool. All three block diagrams given in the first chapter make one fact clear: the Kalman filter is still the primary state estimation tool. AI is just being used to compensate for or adjust poor *a priori* information that the filter uses.

This is in contrast to several other researchers who have recommended replacing the Kalman filter as discussed in Chapter 1. A complete replacement is suggested to help overcome other Kalman filter limitations, such as incorrect dynamic modelling. These systems are typically adaptive or involve learning the input-output relationships needed for integration of INS and GPS.

Chiang (2004) replaced the Kalman filter structure entirely with intelligent algorithms in the form of ANN's. The idea was to limit the need for *a priori* knowledge of the mathematical process and allow the system to develop the model based on supervised learning. During GPS outages, the INS mechanization velocities and headings could be used to estimate the positional changes.

The primary goal of Chiang (2004) was to replace a tuning dependent algorithm with one that can form its own relationships. This intelligent navigator could also adapt itself over time making it applicable for different inertial sensors or varying test conditions. It was shown to be a better estimator than the EKF under several conditions, but if the assumptions made within the EKF are upheld then it can be argued that the EKF is truly an optimal estimator in an MSE sense. Unfortunately, in replacing the EKF entirely with ANN's, this optimality could no longer be guaranteed. In fact, ANN's can only be guaranteed to be as accurate as their training data. If the networks are trained on minimal or poor data then the estimation quality will be very poor. If they are trained on sufficient data and converge on this data then their estimates should approach those of a properly tuned and modelled EKF, within the noise level of the training reference.

3.2 Why is intelligence added to the Kalman filter?

The primary purpose of this work will be to improve upon the conventional Kalman filtering scheme using a combination of adaptive and optimal estimation tools. Based on the limitations of both the Kalman filter and intelligent methods the following will be investigated to help unite the two methods and benefit from the advantages of each:

1. **Develop an algorithm that is applicable for MEMS sensors using SGPS.**

   This method will have to adapt for individual MEMS sensor and GPS receiver errors. Calibration of the MEMS sensors cannot be performed due to cost requirements, thus the method will have to compensate for residual stochastic errors and in some cases larger deterministic errors that take significant time to converge.

These convergence times can be accommodated by increasing the noise levels within the R and Q matrices, while fine-tuning can be performed by later tightening the noise matrices to more realistic levels. This means the system has to be adaptive over time.

2. **Develop an algorithm that can be applied in real-time**

   Processor speed is often an issue for complex algorithms such as large neural networks. The developed algorithm should be capable of providing state estimates (PVA) in real-time at typical MEMS data rates (50-100Hz). The proposed method should also be applicable to real GPS outages, not just to simulated outages.

3. **Compensate for poorly tuned Kalman filters, or filters with dynamically changing parameters.**

   Similar to AKF's, this will involve some form of feedback to the filter in order to bring the filter closer to its optimal state. If the filter can reach this state during availability of GPS then the position drifts will be minimized when the filter operates in prediction mode using only inertial measurements. Since initial tuning accuracy cannot be known ahead of time for MEMS sensors, a learning method will be needed.

4. **Retain optimality of the Kalman filter.**

   When certain assumptions are met the Kalman filter is an optimal estimation tool in terms of minimum MSE. Assuming the correct conditions are met this optimality

criterion should be maintained even if using an intelligent method such as neural networks.

5. **Ensure reliability of the adaptive compensation method.**

   Intelligent methods, such as neural networks, involve a learning stage whereby the inner weightings of the mathematical model are adjusted based on training data. Once converged, these systems are very useful for adaptive estimations, but before they converge their estimates can be misleading and detrimental to overall estimation quality. There is also the extrapolation problem; neural networks are excellent interpolators, but poor extrapolators. A method for controlling the application of the intelligent compensations for these cases needs to be applied.

3.3 Neural networks

ANN's are capable of solving complex large-scale problems that were previously intractable using model based estimation tools. The networks are composed of many parallel layers of neurons with the neurons being connected by synaptic weights and biases. Figure 3.1 shows a single neuron and an example of two neurons connected in a layer. Three inputs (x) are shown, with three weights (w), a bias (b), an activation function (f) and an output (y). The activation function can be any transformation of the data so that the output is in a form required by the designer of the network.

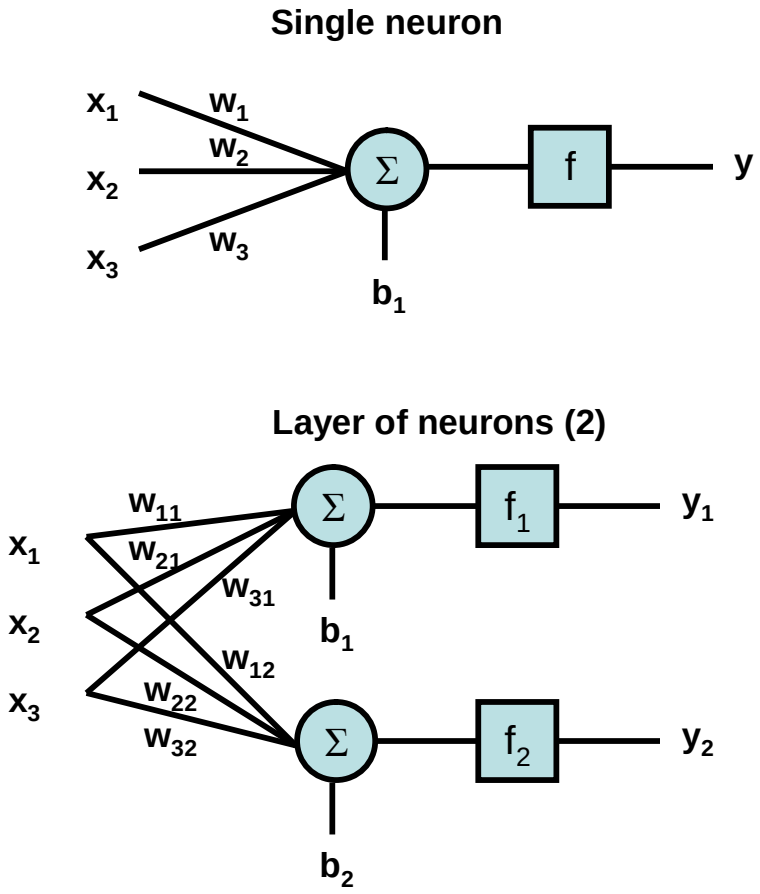

Figure 3.1: Neural network representation

It is the adjustment of these weights and biases that enables the network to change its output for given inputs. These neurons allow the network to adapt to non-linear problems and provide responses for interpolated inputs that have not been encountered. This ability to predict outputs from new inputs is a type of knowledge that makes ANN's different from other estimation algorithms.

An official definition of knowledge is given by Fischler and Firschein (1987) as "*stored information or models used by a person or machine to interpret, predict, and appropriately respond to the outside world*". Any AI system should be capable of doing three things with knowledge: (1) store knowledge, (2) apply knowledge, and (3) acquire new knowledge. These are commonly referred to as (1) representation, which is a structure used to store knowledge for a problem of interest, (2) reasoning, which is the AI's ability to solve a problem, and (3) learning, which is an improvement process to the current knowledge base (Haykin, 1994).

3.3.1 Error-correction learning

The learning process is the most interesting aspect of ANN's. Learning adjusts the weights and biases of the neurons as they are stimulated by inputs from the environment. Any form of learning adjusts the internal parameters so that the network responds in a new way to the environment. A weight between two neurons at time epoch n would be updated as follows,

$$w(n+1) = w(n) + \Delta w(n) \quad \textbf{(33)}$$

The delta step parameter is computed by the network in a specific way depending on the learning algorithm used and stimulation by the environment. Error-correction learning uses a truth target signal to form an error signal as the difference between the target and actual network response at iteration k,

$$error(k) = t\arg et(k) - response(k) \quad \textbf{(34)}$$

The error signal is then used to form a cost function which is used to solve the minimization problem. Specifically, the mean square value of the sum of squared errors is used.

$$Cost = E\left[\frac{1}{2}\sum_{k} e_k^2(n)\right] \quad \textbf{(35)}$$

E is the expectation operator, the ½ scale factor is used for simplification of future derivatives and k represents all the neurons in the output layer of the network. The method of gradient descent is achieved when this cost function is minimized. Realistically the method has no way of knowing the statistical characteristics of the underlying process, so an approximate solution is used, which is known as the instantaneous value of the sum of squared errors.

$$E(n) = \frac{1}{2}\sum_{k} e_k^2(n) \quad \textbf{(36)}$$

The solution to the optimization problem is then found by minimizing this simplified cost function with respect to the weights. A common solution, known as the least mean squares (LMS) algorithm, differentiates the instantaneous error and minimizes it by setting the derivative to zero (Widrow and Hoff, 1960). Using partial derivatives for both the weights and biases the solution for the updates can be given in the form,

$$w(n+1) = w(n) + \alpha e_k(n)x(n) \quad \textbf{(37)}$$

$$b(n+1) = b(n) + \alpha e_k(n) \quad \textbf{(38)}$$

In these equations α is a positive constant that defines the rate of learning and x is the input vector. This parameter also has impacts on the stability of the system. For instance, if α is too large the system might oscillate uncontrollably. In order to be stable, the learning parameter has to be larger than zero but smaller than the inverse of the largest eigenvalue of the input correlation matrix, defined as,

$$R = E\left[x(k)x(k)^T\right] \quad \textbf{(39)}$$

3.3.2 Backpropagation

The LMS algorithm is appropriate for the case of a single layer of linear neurons, but for multiple layers of neurons containing non-linear activation functions a more general algorithm has to be applied. The error backpropagation algorithm is the solution to this problem. The backpropagation algorithm consists of two passes through the network: a forward pass that activates the output of the network and a backward pass that adjusts the weights and biases in accordance with a generalized error-correction rule similar to that of LMS. The error signal is passed backwards through the network layers and this is from where the term backpropagation is derived.

Details of backpropagation can be found in Cichocki and Unbehauen (1993) and Haykin (1994) but the basic premise is to minimize the network error function,

$$\hat{F} = \left(t(k) - a(k)\right)^T \left(t(k) - a(k)\right) \qquad \textbf{(40)}$$

where t(k) is the target value for epoch k and a(k) is the output value at epoch k. The algorithm, for multiple neurons and layers, can be summarized as follows:

1. *Initialization*: define the network configuration and set all weights/biases to small random numbers uniformly distributed.
2. *Training inputs and forward computation*: propagate the new input forward through the network. The first input p_0 can be taken as the initial layer output a_0. The remaining q layer outputs are represented by a,

$$a^{m+1} = f^{m+1}\left(W^{m+1}a^m + b^{m+1}\right) \qquad \textbf{(41)}$$

for m = 0, 1, 2,…, q-1.

3. *Backward computation*: propagate the derivatives backward through the network

$$\frac{\partial \hat{F}}{\partial n^m} = -2F'^m\left(n^m\right)\left(W^{m+1}\right)^T \frac{\partial \hat{F}}{\partial \hat{n}^q} \quad \textbf{(42)}$$

for m = q-1, q-2,…, 2, 1. Where,

$$\frac{\partial \hat{F}}{\partial \hat{n}^q} = -2F^q\left(n^q\right)(t-a) \quad \textbf{(43)}$$

4. *Network update*: adjust the weights and biases using the approximated steepest descent algorithm.

$$W^m(k+1) = W^m(k) - \alpha \frac{\partial \hat{F}}{\partial n^m}\left(a^{m-1}\right)^T \quad \textbf{(44)}$$

$$b^m(k+1) = b^m(k) - \alpha \frac{\partial \hat{F}}{\partial n^m} \quad \textbf{(45)}$$

5. *Iterate*: return to step 2 with a new training input

The weights and biases are updated on a pattern by pattern basis. This means that for every new input the network is activated and updated. This is only an estimate of the true change that would occur if the parameters were modified all at once in batch mode using the cost function over the entire training set. However, for real-time applications that require constant adaptations, or for systems that would require too much memory to update in batch, the pattern mode of updating is more desirable.

3.3.3 Universal approximation

A multilayer network with non-linear sigmoidal activation functions trained with backpropagation is capable of estimating any continuous multivariate function to any desired degree of accuracy, provided that enough hidden neurons are available (Cybenko, 1988). Sigmoidal activation functions are continuously differentiable non-linear activation functions that are most commonly used due to the above theorem.

3.4 Fuzzy inference systems

Fuzzy inference systems (FIS) address the issue of precision when something mathematical has to be described vaguely. Sometimes precision cannot be entirely guaranteed since there will always be some level of uncertainty. Albert Einstein was quoted as saying, "*So far as the laws of mathematics refer to reality, they are not certain. And so far as they are not certain, they do not refer to reality.*". Classical mathematical approaches can have difficulties with these situations, so fuzzy set theory was introduced to bridge the gap between high precision and complexity of uncertainty (Kantardzic, 2001).

Fuzzy logic is quite simple. An input space is mapped to an output space through a list of weighted if/then rules created by the expert designer. The system can have any number of rules and all rules are evaluated in parallel. A typical FIS is made up of fuzzy sets, membership functions, logical operations, and rules.

A fuzzy set has elements that can be entirely or partially attributed to the set. This is in contrast to typical mathematics that says a defined element has to be entirely part of one set or another. If element *a* is part of set *X* then it would mathematically be represented by a value of 1. If *a* was not part of set *X* it would be valued at 0. A fuzzy set does not make this distinction and allows *a* to be part of *X* anywhere between 0 and 1. This is the first step in any FIS and is known as fuzzifying the inputs.

Membership functions are curves that define how each point in the input space is mapped to a value between 0 and 1. Commonly used membership functions are piecewise linear, Gaussian, and trapezoidal membership functions. The fuzzy set and membership functions define the fuzzy part of fuzzy logic. Adaptive fuzzy systems are capable of forming these membership functions from training data and are discussed in the next section.

The logic part is defined by basic OR, AND and NOT Boolean operators. To preserve the fuzziness of the system the OR operator is replaced with MAX, the AND operator with MIN and the NOT is equivalent to (1-A) where A is the first element. This conversion to retain the fuzziness is multivariate logic.

Applying if/then rules completes the FIS with an implication model. These rules form the conditional statements that allow the designer to achieve certain performance measures. An example would be: if the sky is grey, and the clouds are low, and it is daytime then it is likely to rain. The antecendent consists of fuzzifying the inputs and applying the fuzzy operators (if the sky is grey, and the clouds are low, and it is daytime). The implication

applies the result to the consequent (then it is likely to rain). This is a Mamdani fuzzy inference method, which is the most commonly used FIS and is the method used in this dissertation. The entire FIS can be summarized by the following transformations (Kaufmann and Gupta, 1985):

1. Inputs to membership functions
2. Membership functions to rules
3. Rules to output characteristics
4. Output characteristics to output membership functions
5. Output membership functions to a single valued output

Additional details on fuzzy theory and application of FIS can be found in Negoita et al. (2005) and Klir and Yuan (1994).

3.4.1 Adaptive neuro fuzzy inference systems (ANFIS)

In some cases the Mamdani FIS suffers from unknown membership function details. In this case the designer does not know how the membership functions should be formed since the expert knowledge is limited. An ANFIS system alleviates this issue by learning the membership functions using neural networks. Specifically, ANFIS adjusts the parameters of a predefined type of membership function. For example, the centre and variance of a Gaussian membership function can be adjusted. ANFIS can use existing data to estimate the membership functions, or more appropriately, adapt the membership functions as new data is collected. Most ANFIS algorithms use backpropagation for learning.

3.5 Reinforcement learning

ANN's learn by using a set of target values to generate a performance measure such as MSE. A reinforcement learning system improves performance by learning on the basis of any feedback supplied to the system by the environment. Specifically, in RL there is no teacher to provide target values so reinforcements from the environment are used to generate gradient information. Certain actions are tried, the environment provides reinforcement signals, and the system is adapted based on how good or bad the reinforcements were. The performance of the adaptation can really only be evaluated after many situations are tried and many different reinforcements signals are experienced. This type of learning is more relative than absolute, but given enough experience the system can learn things that would be impossible with an ANN.

In any RL problem there is always an interaction between an agent and its environment. Within these two, the RL problem contains a policy, a reward function, a value function and even a rough model of the environment. A policy explains how the agent behaves in a given scenario, or in other words, what actions should be taken in a given state. Policies are the most important element of the problem, and once learned they are all that is needed to determine appropriate behavior. A reward function associates individual state/action pairs of the environment to a number that describes the positive or negative reward. The reward function is typically used to alter the optimal policy depending on what is perceived to be good or bad by the agent. A value function defines what is good for long-term rewards. Typically, an RL problem attempts to maximize the value function over many

individual rewards so that the long-term success of the policy is satisfied. In essence, this type of learning needs to maximize long-term gain.

Policy iteration is important to RL theory. Two simultaneous processes interact with one another to converge on a solution. One process makes the value function consistent with the current policy, and the other makes the policy greedy with respect to the current value function. Greedy means that the option with maximal likelihood is chosen. Both of these processes alternate until convergence occurs on both the optimal value function and the optimal policy. Both processes stabilize together, and only when a policy has been found that is greedy with respect to its own value function. In more details, one process takes the policy as given and performs policy evaluation which changes the value function to be more similar to the true value function for the given policy. The other process takes the value function as given and performs improvement to the policy (Sutton and Barto, 1998).

Discounted returns are often used to relate newer and older rewards. Equation (45) gives the discounted reward R_t using all future rewards discounted by γ, which is always between 0 and 1. This equation looks ahead in time, but in causal systems the timing sequence would be reversed.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... + \gamma^n r_{t+n+1} \quad \textbf{(46)}$$

Finally, some RL systems use models of the environment (Barto et al., 1991). These models can be provided ahead of time, if sufficient information is available, or they can be learned through trial and error, and then incorporated into the policy. When tuning a KF the model is first developed through trial and error learning, and then applied to help guide

future tuning decisions; it is assumed that no prior knowledge is available besides a default starting point.

In general, the interaction between the agent and the environment can be visualized as in Figure 3.2 (Sutton and Barto, 1998). The agent acts on the environment, which in turn provides a reward back to the agent based on the current state. This continues for the next state/action pair in a recursive manner.

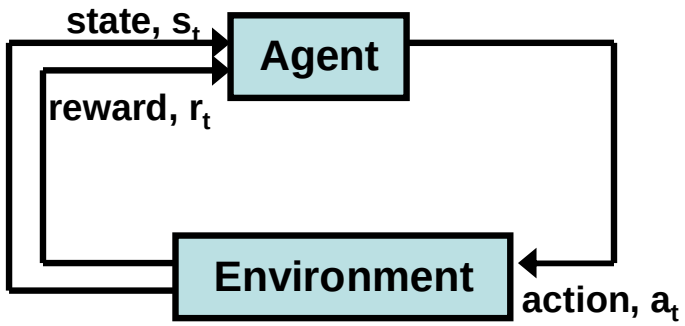

Figure 3.2: Agent-environment interaction (Sutton and Barto, 1998)

3.5.1 Basic RL theory

Since the RL algorithm estimates value functions and attempts to maximize these over future iterations, the definition of the value of taking action *a* in state *s* under policy *p* is,

$$Q^p(s,a) = E_p\{R_t \mid s_t = s, a_t = a\} \quad \textbf{(47)}$$

And the discounted reward can be inserted as,

$$Q^p(s,a) = E_p\{\sum_{n=0}^{\infty} \gamma^n r_{t+n+1} \mid s_t = s, a_t = a\} \quad \textbf{(48)}$$

This last equation says that as the number of states encountered reaches infinity, the average reward will approach the actual state value function. This form of learning is a type of Monte Carlo (MC) method. The most commonly used RL algorithms use a combination of modelling (dynamic programming) and trial and error learning (Monte Carlo methods).

In terms of dynamic programming (DP), RL learning follows particular recursive relationships, such as the Bellman optimality equations for Q*(s,a) shown here, starting from the previous equation, after Bellman (1957),

$$
\begin{aligned}
Q^{p}(s,a) &= E_{p}\{r_{t+1} + \gamma \sum_{n=0}^{\infty} \gamma^{n} r_{t+n+2} \mid s_{t} = s, a_{t} = a\} \\
Q^{*}(s,a) &= E\{r_{t+1} + \gamma \max_{a'}[Q^{*}(s_{t+1}, a')] \mid s_{t} = s, a_{t} = a\} \\
Q^{*}(s,a) &= \sum_{s'} P^{a}_{ss'}[R^{a}_{ss'} + \gamma \max_{a'}[Q^{*}(s_{t+1}, a')]]
\end{aligned}
\qquad \textbf{(49)}
$$

The * denotes the optimal estimate, while P represents the probability of the next state s' from the current state s and action a, and R gives the expected value of the reward.

The importance of the Bellman equation is that if the optimal Q is known then the agent does not even have to look ahead, all it has to do is pick the action that maximizes the state/action pair of Q. Unfortunately, explicitly solving the Bellman equation is similar to performing an exhaustive search, or having an accurate model in memory. RL theory attempts to approximate solutions to this optimality equation by 'experiencing' the transitions instead of actually knowing them ahead of time. This is how the Monte Carlo

and Dynamic Programming methods aid each other in finding a solution that neither could do alone.

This leads to the core RL algorithm used in this dissertation: temporal difference (TD) learning. TD can learn directly from experience, similar to Monte Carlo, and TD can also provide estimates based on previously learned estimates, similar to DP. For any RL algorithm, incremental learning is the key to updating a policy, as is the case for TD learning. The basic formula for this update is,

$$\text{New} = \text{Old} + \text{Step}[\text{Target} - \text{Old}] \tag{50}$$

The bracketed expression is an error in the old estimate, and is reduced by taking a step towards the target value; this reduction is imperative to avoid noise and oscillations. To understand how TD uses both MC and DP methods, consider the value functions for both cases. MC waits until the return is known, then uses the return as a target value for the value of that state,

$$V(s_t) = V(s_t) + \alpha\left[R_t - V(s_t)\right] \tag{51}$$

DP updates the value function using discounted returns and the optimal policy (accurate system model). The optimal solution would be found as follows,

$$\begin{aligned} V^{\pi}(s_t) &= E_{\pi}\left\{R_t \mid s_t = s\right\} \\ V^{\pi}(s_t) &= E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s\right\} \\ V^{\pi}(s_t) &= E_{\pi}\left\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \mid s_t = s\right\} \\ V^{\pi}(s_t) &= E_{\pi}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \mid s_t = s\right\} \end{aligned} \tag{52}$$

It can be seen from the final form of the optimal solution that it is impossible to evaluate this using DP since the optimal next return, $V^{\Pi}(s_{t+1})$, is not known. DP has an accurate model so the expected value is known, but it must use an estimate, $V(s_{t+1})$, instead of the unknown optimal next return.

$$V^{\pi}(s_t) = E_{\pi}\{r_{t+1} + \gamma V(s_{t+1}) \mid s_t = s\} \quad \textbf{(53)}$$

In comparison, MC uses an estimate of this last equation as a target. It is an estimate because MC assumes the expected value is not known, so a sample return is used instead of the real expected return.

The TD method attempts to unify these two approaches through their dualistic nature. TD uses sampling similar to MC, but does not have to wait until the end of an episode to update; it can be updated after each step. TD also uses modelling like DP but does not know the model ahead of time; the optimal policy is what TD is working towards. Instead, TD combines MC and DP theory into a combined method that performs bootstrapping (Dayan, 1992). Bootstrapping means the updated estimate is based on previous estimates. This is done as follows,

$$V(s_t) = V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \quad \textbf{(54)}$$

It can be noticed that the optimal policy is now no longer followed, but an estimate of this policy is used instead. This estimate is based on previous estimates that have been formed by sampling.

When learning Q values with TD, the update can be expressed in a similar form as,

$$Q(s_t,a_t)=Q(s_t,a_t)+\alpha[r_{t+1}+\gamma Q(s_{t+1},a_{t+1})-Q(s_t,a_t)] \quad \textbf{(55)}$$

In this equation, the step size is α and the target is expanded into the first two terms within the brackets. This method of updating the Q values is commonly referred to as SARSA since it uses the following information: s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}.

The definition of episodes can be important for certain RL problems. An episode is defined as the end of a learning process, but is not critical to the implementation of SARSA. For instance, in many games such as checkers there exist distinct episodes, such as the end of a game, and new games are played repeatedly. For some applications there may be only a single episode, known as a continuous episode, in which case the steps of the individual episode direct the learning. For the example of checkers a step would be a single move within the game.

The advantage of TD learning over Monte Carlo and DP methods are numerous from an applications perspective. TD methods learn estimates based on previous estimates, but unlike Monte Carlo methods they do not need to wait until the end of an episode, they can update after each step. In comparison to DP methods they do not require a model of the environment, but instead they can build this model through interactions. These two advantages are the main reasons why RL algorithms are better suited to on-line applications, such as tuning a KF incrementally as navigation data is collected.

3.5.2 Details of the learning strategy

A form of temporal difference learning can be used to tune a Kalman filter. The algorithm used is the SARSA method using on-policy control, eligibility traces and generalization. On-policy simply refers to the fact that the method attempts to improve the policy that is used to make decisions for the control problem. On-policy methods typically use ε-greedy methods, which usually choose an action that that has maximal estimated return value.

The general SARSA algorithm is quite simple and is shown as pseudocode in Figure 3.3, initially proposed by (Rummery and Niranjan, 1994).

```
Initialize the policy Q(s,a)
Define the number of episodes and steps
For each episode,
        Initialize the state s
        Choose action a from state s using ε-greedy Q
        For each step,
                Take action a
                Observe return r and new state s'
                Choose next action a' from next state s' using ε-greedy Q
                Update Q(s,a) = Q(s,a) + α[r + γQ(s',a') − Q(s,a)]
                Set s = s' and a = a'
```

Figure 3.3: Basic SARSA algorithm without eligibility traces and generalization

The actual learning process can be done sequentially with the SARSA method. This means that on-line adjustments can be made to the policy as new data is collected. Figure 3.4 shows the algorithm in expanded detail when applied to integrated navigation data.

1. Begin INS/GPS navigation using *a priori* data in current memory.
2. Simulate several GPS outages by operating the filter in INS prediction mode. The initial window length should be sufficiently long that many GPS outages can be simulated.
3. Run the SARSA algorithm on this window of data while retaining memory of past data.
    a. If the method has no knowledge of the model then begin with an arbitrary Q. Otherwise use the optimal policy from a previous tune or a similar sensor.
    b. Set the number of episodes and steps. A single episode can be used, but the use of multiple episodes will encourage the method to improve upon the process of tuning instead of simply reaching an optimal state. This optimal tuning strategy can be useful for leveraging to additional sensors.
    c. Begin an episode of tuning
        i. Initialize the state of the *a priori* data. If the tuning is just starting for a specific sensor then set the state to manufacturer values. If the tuning has already been performed on a previous window of data for this specific sensor then use the previously defined optimal state.
        ii. Choose a set of state changes as actions. Perform these actions on all of the *a priori* data in parallel; this means that a single action can change all of the tuning parameters at once. Use the ε-greedy method to make the choice of correct actions. Allow a small random chance for an exploratory action (i.e. one that does not follow the ε-greedy policy).
        iii. Begin steps within the episode
            1. Take the action a
            2. Observe the returned reward value and assign this as immediate reward.
            3. Update the state of the *a priori* data using the actions.
            4. Choose the next set of state changes as new actions. Perform these actions on all of the *a priori* data in parallel. Use the ε-greedy method to make the choice of actions. Allow a small random chance for an exploratory action.
            5. Update the policy using the SARSA equation.
            6. Set the new state and action values as the current state and action values.
        iv. Iterate until done all steps or if a step achieves a desired accuracy
    d. Iterate until done all episodes
4. Choose a new window of data and return to step 2. The new window is chosen by accepting the next simulated GPS outage and dropping the oldest outage.

Figure 3.4: Basic SARSA applied to navigation data

The moving window of data enables two important characteristics. Firstly, it allows the system to adapt to new data while still retaining knowledge of old data. The new data will provide some new knowledge of the system, but will likely not change the entire policy. This incremental knowledge adds to the optimal policy without changing other knowledge that has been already accumulated.

The moving window can also be used to balance adaptation and reliability. Adaptation should be performed quickly as new data is received, so a sliding window is used to accept new simulated GPS outages. However, to be reliable the window should encompass as many outages as possible to get a good estimate of the variance on the current state estimates. Based on memory constraints the window can be defined in a maximal sense, and can drop old outages and accept new outages in a first in first out (FIFO) manner.

The SARSA algorithm can guarantee that it adapts to changing models through continued exploration. This is a small probability that the algorithm ignores the ε-greedy policy and chooses another option. This is useful for exploring new state/action pairs or for checking previously visited pairs for any model changes. To fully converge to a solution the designer would have to slowly decrease this probability of a random choice, but in reality the model and *a priori* parameters are prone to change over time so continued exploration is preferable.

The stopping criteria within an episode can be defined in an absolute or relative sense. An absolute stopping criterion would set a hard threshold on the reward values and stop the

episode if any steps achieved this level. The difficulty with an absolute threshold is that this value cannot be known ahead of time. Another problem with such a threshold is that it takes a long time to achieve this level of performance, resulting in more steps needed for convergence.

An adaptive threshold was used to alleviate these issues. The initial threshold was defined as the average value of the individual rewards in the first episode. After this point the threshold was reduced by 10% for each episode that was stopped prematurely. Eventually the threshold was reduced to a level that could not be met and all steps were taken for each episode. This gave the algorithm a better threshold that did not require the need of designer input. Furthermore, if the model of the system were to change, the system would maintain memory of the previous best performance measure and would always try to do better.

3.5.3 Uniting TD and MC using eligibility traces

TD methods can be compared more closely with MC methods if eligibility traces are used. Eligibility traces keep memory of certain events, such as visiting a state and taking an action. The term trace refers to the memory parameters, while the term eligible means that these traces can undergo learning changes.

The use of eligibility traces creates a learning method that is somewhere between MC and TD methods. MC methods perform updates using the entire sequence of observed rewards whereas TD updates are performed based on every single next reward. Eligibility traces

perform updates based on an intermediate number of rewards. The TD method described in the previous section would be considered a one-step update, or TD(0).

The TD(0) method bootstraps using only the previous estimate, so in the case of a problem that is not 1st order Markov it may have convergence problems. MC methods are not concerned with Markov properties, but they do have to wait a long time until an update is performed. It is the use of eligibility traces that blends the advantages of each method into a more powerful TD method.

An acausal explanation is given for eligibility traces, but of course a causal implementation is used. The acausal, or forward view in time, is useful for understanding eligibility traces. In a one-step update the target is,

$$R_t^1 = r_{t+1} + \gamma V_t(s_{t+1}) \quad \textbf{(56)}$$

In this equation the superscript on R refers to the one-step. To extend this update to n steps a discounted return has to be applied, in a similar manner as it was applied to the immediate rewards,

$$R_t^n = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+1}) \quad \textbf{(57)}$$

The actual update occurs to the value function in an increment defined as,

$$\Delta V_t = \alpha[R_t^n - V_t(s_t)] \quad \textbf{(58)}$$

This equation is only applicable for states within the defined trace. For on-line methods the updates are done during the episode,

$$V_{t+1}(s) = V_t(s) + \Delta V_t(s) \quad \textbf{(59)}$$

The general n-step update is performed by averaging n updates, each proportional to λ^{n-1}, where λ is between 0 and 1. λ is commonly referred to as the trace decay parameter. The weights have to all sum to 1 to satisfy the averaging approach, so a normalization factor of 1-λ is used. The n-step updates provide a general return called the λ-return,

$$R_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}R_t^n \quad \textbf{(60)}$$

The 1-step return is given a weight of (1- λ), the 2-step return a weight of (1- λ) λ and so on. Two special cases of this equation result by setting λ to 1 and 0. If λ is 1, then the method turns into the MC method. If λ is 0 then the method is the same as the TD(0) algorithm. After a terminal state is reached all subsequent n-step returns are equal to R_t. If these are separated from the sum the form becomes,

$$R_t^{\lambda} = (1-\lambda)\sum_{n=1}^{T-(t+1)}\lambda^{n-1}R_t^n + \lambda^{T-(t+1)}R_t \quad \textbf{(61)}$$

In this form it can easily be seen that when λ equals 1,

$$R_t^{\lambda} = R_t \quad \textbf{(62)}$$

and when λ is zero,

$$R_t^{\lambda} = R_t^1 \quad \textbf{(63)}$$

Eligibility traces can only be applied to causal systems, so a backwards view has to be used to implement the theory. For a causal system the eligibility trace for each state s is defined by (Sutton and Barto, 1998),

$$e_t(s) = \gamma\lambda e_{t-1}(s) + 1 \quad \textbf{(64)}$$

if the state is part of the trace, otherwise the eligibility trace is,

$$e_t(s) = \gamma\lambda e_{t-1}(s) \quad \textbf{(65)}$$

As states are visited the eligibility trace accumulates. If states are not visited the eligibility trace fades to a minimum. λ is referred to as the trace decay parameter and is always between 0 and 1. From a practical viewpoint, these accumulating traces give an idea of how eligible a state is for learning changes, should an update occur. To apply eligibility traces in a backward view the following can be used to adjust the value function,

$$\Delta V_t(s) = \alpha\left[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)\right]e_t(s) \quad \textbf{(66)}$$

This theory is then easily extended to the SARSA algorithm (Rummery, 1995),

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\left[r_{t+1} + \gamma Q_t(s_{t+1},a_{t+1}) - Q_t(s_t,a_t)\right]e_t(s,a) \quad \textbf{(67)}$$

where the eligibility traces are,

$$e_t(s,a) = \gamma\lambda e_{t-1}(s,a) + 1 \text{ and } e_t(s,a) = \gamma\lambda e_{t-1}(s,a) \quad \textbf{(68)}$$

when the state is part of the trace and not part of the trace respectively. Eligibility traces can easily be added to the SARSA algorithm, but they do add memory requirements. A 2-step update requires double the memory. Fortunately, when generalization is applied the memory requirements are significantly reduced since traces are not applied to a discrete state set.

3.5.4 Generalizing from discrete to continuous RL

The policy developed using the basic SARSA algorithm, or SARSA with eligibility traces, is essentially a discrete look-up table that maps state/action pairs to rewards. This table is useful in the sense that it guides the learning towards the best *a priori* values, but it also suffers the fact that it can grow exponentially as more states or actions are added. To avoid the issue of increased memory, generalization can be used to make the table continuous, which also saves time and data needed to fill the table. Generalization expands experience

from a limited subset of the state space to provide good approximations over much larger subsets. By making the table continuous, fewer discrete steps need to be taken and the ranges on individual parameters can be expanded. Furthermore, without generalization eligibility traces add greatly to the computational task, but with generalization the computations are only doubled (Cichosz, 1995).

As an example, consider a discrete problem with eight *a priori* tuning parameters and five discrete steps for each parameter. The Q table would contain 390625 (5^8) discrete entries. These entries represent the reward transitions between different states using different actions. In order to guarantee that the optimal solution had been found, all 390625 tuning combinations would have to be attempted with rewards calculated for each. In a real-world example this number of tuning steps could never be attempted in an efficient manner.

Generalization applies the RL algorithm to states that have never been visited before. Supervised learning methods can be applied to approximate the value functions of states that have never been visited. In the discrete case a single table entry is moved towards the current value function returned by SARSA. But in the case of generalized function approximation using supervised learning, the updated value functions can be considered as desired input-output behaviours for the underlying function. In essence these are the training examples that form the function.

The performance measure for any supervised method is minimum MSE. Considering the states as inputs, the target function as the true value function V^{Π}, the distribution of the

inputs as P, and θ_t as the parameterized form of the table, then the MSE for an approximation V_t is,

$$MSE(\theta_t) = \sum_s P(s)[V^\pi(s) - V_t(s)]^2 \quad \textbf{(69)}$$

P distributes weights of errors across different states. It trades off better approximation at some states at the expense of worse approximation at others. Since ANN's are very good interpolators, but poor extrapolators, to gain uniform level of error across the entire state set training examples would have to be chosen uniformly distributed across the entire set. This is not often the case when the RL algorithm is left to learn on its own, but it is useful to understand what approximations are likely to be accurate and which will be useless; the P matrix provides this information.

Using minimum MSE as the performance measure, gradient descent methods can again be used (Widrow and Hoff, 1960). The parameter vector $\bar{\theta}$ is adjusted in the direction that most reduces the error of a sequential sample (Sutton and Barto, 1998). The step is adjusted in the direction of most improvement in terms of minimizing the MSE; the step in θ_t is proportional to the negative gradient of the example's squared error,

$$\bar{\theta}_{t+1} = \bar{\theta}_t - \frac{1}{2}\alpha\nabla_{\bar{\theta}_t}[V^\pi(s_t) - V_t(s_t)]^2 \quad \textbf{(70)}$$

When the partial derivative of the MSE is taken, using the chain rule, the equation becomes,

$$\bar{\theta}_{t+1} = \bar{\theta}_t + \alpha[V^\pi(s_t) - V_t(s_t)]\nabla_{\bar{\theta}_t} V_t(s_t) \quad \textbf{(71)}$$

The derivative of the optimal value function is zero by definition; it is a minimum or maximum with zero gradient. In these equations the gradient denotes a vector of partial derivatives. These derivatives simply represent the gradient of *f* with respect to the parameter vector,

$$\left(\frac{\partial f(\bar{\theta}_t)}{\partial \theta_t(1)}, \frac{\partial f(\bar{\theta}_t)}{\partial \theta_t(2)}, \ldots, \frac{\partial f(\bar{\theta}_t)}{\partial \theta_t(n)}\right)$$

In real situations the true value function quantity V^{Π} is not known. Instead we have to use the target output of the t[th] output. In the case of MC,

$$\bar{\theta}_{t+1} = \bar{\theta}_t + \alpha[R_t - V_t(s_t)]\nabla_{\bar{\theta}_t} V_t(s_t) \quad \textbf{(72)}$$

Or in the case of n-step TD the λ-return would be used. In acausal form this would be,

$$\bar{\theta}_{t+1} = \bar{\theta}_t + \alpha[R_t^{\lambda} - V_t(s_t)]\nabla_{\bar{\theta}_t} V_t(s_t) \quad \textbf{(73)}$$

In causal form, and considering eligibility traces combined with generalization the equation becomes,

$$\bar{\theta}_{t+1} = \bar{\theta}_t + \alpha[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)]\gamma\lambda\,\bar{e}_{t-1} + \alpha[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)]\nabla_{\bar{\theta}_t} V_t(s_t) \quad \textbf{(74)}$$

Since,

$$\bar{e}_t = \gamma\lambda\,\bar{e}_{t-1} + \nabla_{\theta_t} V_t(s_t) \quad \textbf{(75)}$$

and in more compact form,

$$\bar{\theta}_{t+1} = \bar{\theta}_t + \alpha[r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)][\gamma\lambda\,\bar{e}_{t-1} + \nabla_{\bar{\theta}_t} V_t(s_t)] \quad \textbf{(76)}$$

In the case of state/action pairs this equation can be rewritten as,

$$\bar{\theta}_{t+1} = \bar{\theta}_t + \alpha[r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)][\gamma\lambda \bar{e}_{t-1} + \nabla_{\bar{\theta}_t} Q_t(s_t, a_t)] \quad \textbf{(77)}$$

This is commonly referred to as gradient-descent SARSA(λ) (Rummery, 1995). Before implementing this update, the function approximation method must first be chosen.

3.5.5 Radial basis functions for approximation

Radial basis functions (RBF's) are a form of ANN and are especially suited to this type of function approximation problem. The RBF network consists of an input layer, a hidden layer and an output layer. The network performs a non-linear mapping from the input space to the hidden space, and then provides a linear mapping to the output space. A general RBF network is shown in Figure 3.5.

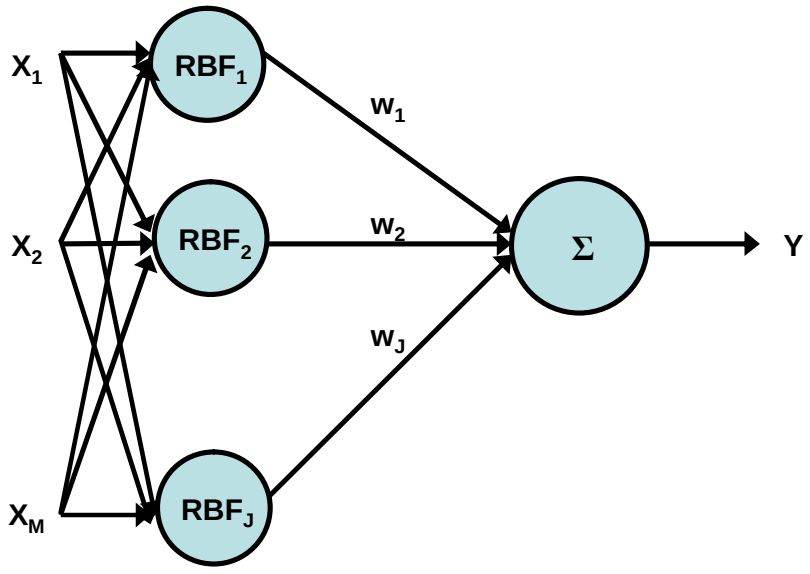

Figure 3.5: Radial basis function network

The output of the RBF is a simple summation of the non-linear functions in the hidden layer multiplied by the weights (Simon, 2001),

$$Y = \sum_{i=1}^{J} w_i \rho_i(\bar{x}) \quad \textbf{(78)}$$

ρ is simply the RBF used in the hidden layer. A commonly used function is the Gaussian, which can conveniently be used to estimate any differentiable non-linear function by adjusting the centres and variances. An example of a two dimension function approximation using Gaussians is shown in Figure 3.6.

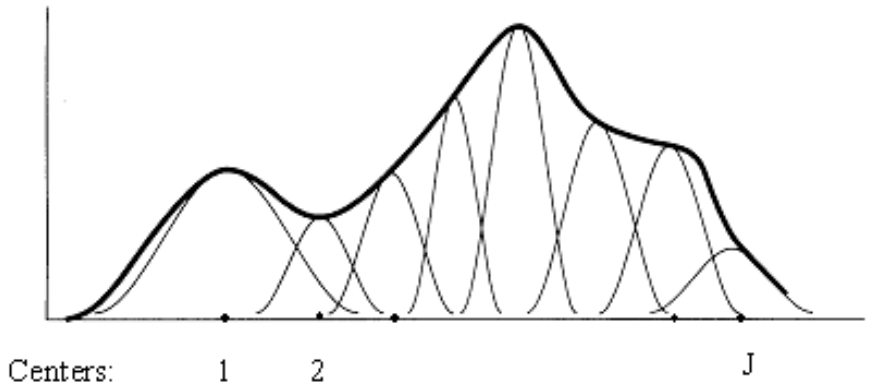

Figure 3.6: 2D Gaussian RBF approximation

The resulting output of the network using Gaussians would be,

$$Y = \sum_{i=1}^{J} w_i e^{-\|\bar{x}-c_i\|^2/\sigma_i^2} \quad \textbf{(79)}$$

There are three variable parameters within this equation: the centres, the variances and the weights. All these parameters can be adjusted using gradient descent and the LMS algorithm, but a more efficient method combines supervised and unsupervised learning methods. The unsupervised learning is performed on the centres and variances, and the supervised LMS algorithm is used last to adjust the weights.

The centre numbers must first be chosen, then the centres can be positioned using a clustering method such as K-means. Both the selection of the numbers of centres and their placement can be done in an iterative heuristic:

1. Define the maximum and minimum number of centres
2. Add one centre to the lower limit, and simultaneously subtract one centre from the upper limit.
3. Cluster the centres using K-means and stop adding/deleting centres if either of the following occur:
    a. By adding one centre there was minimal change
    b. By subtracting one centre there was a significant change

These changes are defined in a relative sense in comparison to a running average of observed changes from previously added/deleted centres. The K-means algorithm clusters the centres and updates their positions until no noticeable changes occur in the centre positions.

The variances of the individual RBF's can be determined using the p-nearest neighbour heuristic,

$$\sigma_i = \frac{1}{p}\sqrt{\sum_{j=1}^{p} \| c_i - c_j \|^2} \qquad \textbf{(80)}$$

The p-nearest neighbour calculates the average of centre position distances over p centres. The choice of p can be made as a percentage of the total number of centres, with a minimum value defined for p.

The weights of the network are chosen using minimum MSE as the training performance measure. If *t* is a vector of *N* target values then the MSE is,

$$MSE = \sum_{i=1}^{N} (t_i - y_i)^2 \tag{81}$$

The LMS algorithm can be used iteratively to update the weights,

$$\Delta w_i = -2\alpha[y(k) - \hat{y}(k)]e^{-\|\bar{x}-c_i\|^2/\sigma_i^2} \tag{82}$$

Or if training is performed in batch, all of the target values can be related to the weights and RBF's in linear matrix form,

$$\begin{bmatrix} t_1 \\ t_2 \\ \downarrow \\ t_N \end{bmatrix} = \begin{bmatrix} w_1\rho(\| x_1 - c_1 \|) + w_2\rho(\| x_1 - c_2 \|) + ... + w_J\rho(\| x_1 - c_J \|) \\ w_1\rho(\| x_2 - c_1 \|) + w_2\rho(\| x_2 - c_2 \|) + ... + w_J\rho(\| x_2 - c_J \|) \\ \downarrow \\ w_1\rho(\| x_N - c_1 \|) + w_2\rho(\| x_N - c_2 \|) + ... + w_J\rho(\| x_N - c_J \|) \end{bmatrix} \tag{83}$$

And in compact form,

$$t_{1xN} = \Phi_{NxJ} w_{Jx1} \tag{84}$$

The pseudo inverse can be used to solve directly for the weights where Φ is the matrix of RBF's separated from the weights,

$$\bar{w}^* = \Phi^t \bar{t}$$

where,

$$\Phi^t = \left(\Phi^T \Phi\right)^{-1} \Phi^T$$

In the case of RL, the weights are adjusted as the θ parameters.

3.5.6 Simplification for the linear case

RBF's contain a linear output layer. This means that the approximated function Q is a linear function of the parameter vector θ combined with the RBF Gaussian features ρ,

$$\begin{aligned} Q(s,a) &= \bar{\theta}_t^T \bar{\rho}_s \\ Q(s,a) &= \sum_{i=1}^{j} \theta_t(i)\rho_s(i) \end{aligned} \quad \textbf{(85)}$$

The partial derivative of Q with respect to θ is simply ρ.

$$\nabla Q(s,a) = \bar{\rho}_s \quad \textbf{(86)}$$

And as a Gaussian, the i[th] feature with the state denoted by s is,

$$\rho_s(i) = \exp\left(\frac{-\| s - c_i \|^2}{\sigma_i^2}\right) \quad \textbf{(87)}$$

So the gradient Q value is simply found by summing the j features, i.e.

$$\nabla Q(s,a) = \sum_{i=1}^{j} \exp\left(\frac{-\| s - c_i \|^2}{\sigma_i^2}\right) \quad \textbf{(88)}$$

3.5.7 Applying generalization

The purpose of the generalization is to replace the tabular SARSA with a parameterized form which contains significantly less parameters than table entries. The Q values depend completely on how θ varies with time. When using RBF's, the output of the network is the actual Q values and θ represents the variable parameters within the RBF network: the weights, centres and variances. Since unsupervised methods are used to control the centres

and variances, the parameter vector would only contain the weights. Changing several weights changes many more Q values. The inputs to the network are simply the state/action pairs. The complete algorithm is represented in Figure 3.7, initially proposed by Watkins (1989).

Initialize the θ arbitrarily, but uniformly across the state/action space
Set the eligibility trace vector to 0
Define the number of episodes and steps
For each episode,
Initialize the state s
For all the possible discrete actions a in state s,

$$Q_a \leftarrow \sum_{i=1}^{j} \theta(i)\rho_s(i)$$ summation over all j RBF features

Choose action a that maximizes Q_a (ε-greedy)
For each step,

$$\bar{e} \leftarrow \gamma\lambda \bar{e}$$

For all eligibility traces, which are also i in number,

$$e(i) \leftarrow e(i) + \rho_s(i)$$

Take action a
Observe return r and new state s'

$$\delta \leftarrow r - Q_a$$ (done here before Q_a updated)

For all the possible actions in state s',

$$Q_a \leftarrow \sum_{i=1_a}^{j} \theta(i)\rho_{s'}(i)$$

Choose action a' that maximizes Q_a (ε-greedy)

$$\delta \leftarrow \delta + \gamma Q_{a'}$$

$$\bar{\theta} \leftarrow \bar{\theta} + \alpha\delta \bar{e}$$

Set a = a'

Figure 3.7: Gradient-descent SARSA(λ)

The actions are still discrete for gradient-descent SARSA(λ), it is only the Q value function that is parameterized through RBF's. For each action in the current state, the $Q_a(s_t,a)$ value is computed and the action with the maximum Q return is chosen.

Each state is represented by all the RBF features since the Gaussian is continuous across the entire state space. Another important detail is that eligibility traces now correspond to components of θ, and not to individual states. This ensures that the traces now apply to many states in a weighted manner. Due to this fact, the number of eligibility traces can be decreased significantly. In fact, eligibility traces only apply to generalization since the discrete case is 1st order Gauss Markov.

3.5.8 Comparison to AKF

The RL algorithm performs iterative tuning. The tuning is expected to get better over time using only a single tuning strategy at a given time epoch. This is in contrast to the MMAE AKF which used multiple parallel filters run at the same time epoch. Within the MMAE method, if a sufficient number of filters were run then the solution would converge to the optimal filter.

The MMAE method can guarantee optimality but only with an infinite number of parallel filters. The RL algorithm is practical for real-time implementation since it only runs a single filter at a time, but it needs a long time to converge to a good solution. It is the intent of the RL algorithm to approach the optimality of the MMAE method as time and data increase, or as the unit is used. This is a more realistic implementation than the MMAE AKF and weights cost, usability and performance.

In comparison with the innovation sequence AKF, the RL holds memory of the tuning procedure that the innovation sequence could only do within a fixed window. The window length of the RL method does not impact memory of the system, it only affects reliability. The Q values retain memory of past tuning experiences, so if the optimal state of the system changes, the Q values can be used to quickly adapt the system to a new optimal state; the tuning method does not change even if the optimal state does.

The RL method of tuning can be thought of as a long-term approximation to the MMAE method. The ANN method performs a more short-term compensation for motion dependent errors, similar to the IAE AKF. If the error models change due to dynamics the ANN compensates the state estimates, but the RL learning still tries to get the underlying long-term model by averaging out short-term variations; this is done by learning over many data sets, as discussed in Chapter 6.

CHAPTER 4: COMPENSATION DURING OUTAGES

This chapter uses a learning algorithm to compensate additional filter state errors that are not dealt with properly by a loosely coupled filter. This requires the use of two parallel filters for online operation. One filter is used for navigation and the other for adapting the state estimates of the navigator. The use of two filters provides a reference signal that can be used as a target. The inputs to the system are temporal and dynamic. The assumption is that poor modelling of the inertial sensor errors creates amplified drifts during periods without measurement updates and these can be learned and removed from future scenarios.

4.1 Concept of using GPS outages

When GPS updates are not available the filter prediction errors accumulate with time due to the integrating nature of the inertial navigator. The extent of this drift over time and through different dynamics can be used to characterize how the filter is performing. Error models in the local level frame are used to understand how sensor errors propagate into state estimates, which are then accumulated during periods of prediction.

The error terms are time dependent, so they are represented by first-order differential equations. Starting with the position errors, they are primarily coupled to the velocity errors through the approximation,

$$\begin{bmatrix} \dot{\delta\varphi} \\ \dot{\delta\lambda} \\ \dot{\delta h} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{R+h} & 0 \\ \frac{1}{(R+h)\cos\varphi} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta V^e \\ \delta V^n \\ \delta V^u \end{bmatrix} \quad \textbf{(89)}$$

The velocity errors can also be approximated as,

$$\begin{bmatrix} \dot{\delta V^e} \\ \dot{\delta V^n} \\ \dot{\delta V^u} \end{bmatrix} = \begin{bmatrix} 0 & f^u & -f^n \\ -f^u & 0 & f^e \\ f^n & -f^e & 0 \end{bmatrix} \begin{bmatrix} \delta p \\ \delta r \\ \delta A \end{bmatrix} + R_b^l \begin{bmatrix} \delta a_{bx} \\ \delta a_{by} \\ \delta a_{bz} \end{bmatrix} \quad \textbf{(90)}$$

And the attitude errors as,

$$\begin{bmatrix} \dot{\delta p} \\ \dot{\delta r} \\ \dot{\delta A} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{R} & 0 \\ \frac{-1}{R} & 0 & 0 \\ \frac{-\tan\varphi}{R} & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta V^e \\ \delta V^n \\ \delta V^u \end{bmatrix} + R_b^l \begin{bmatrix} \delta g_{bx} \\ \delta g_{by} \\ \delta g_{bz} \end{bmatrix} \quad \textbf{(91)}$$

These equations show that the latitude error δφ depends mainly on the north velocity error, which in turn depends primarily on the pitch error, the gyroscope measurements in the parameterization matrix, the accelerometer bias errors and the gyroscope bias errors (Abdel-Hamid, 2005). Similarly, the longitude error depends on the east velocity error, which in turn depends primarily on the roll error, the gyroscope measurements and the inertial sensor bias errors. Both horizontal position errors also depend on the azimuth errors, but these have much less contribution in vehicle applications since the gravity signal contained in f^u is usually much stronger than the horizontal accelerations experienced through dynamics.

The modelling of the inertial sensor errors is typically done using first order Gauss Markov processes. For a general error term b,

$$\dot{b}(t) = -\frac{1}{\tau} b(t) + \sqrt{\frac{2}{\tau}\sigma^2}\, w(t) \qquad \textbf{(92)}$$

This model relies on appropriate values for the correlation time and process variance. Incorrect modelling can create poor estimates of the sensor error terms which can then propagate all the way upwards to eventually affect position estimates.

These stochastic parameters are usually given by the designer of the system as fixed values. When the designer changes these values to assess state performance the process is referred to as tuning. Different tuning parameters will result in different mean state drift errors for similar outage times and dynamics. The use of improper stochastic information can create deterministic errors on the output of the filter estimates.

An example of improper tuning could occur if the correlation time is too long and the variance level too small for a specific sensor. In this case the filter estimation would always be behind changes in the actual sensor bias estimation, leaving residual bias errors that could be large for a MEMS inertial sensor with considerable turn on biases.

Figure 4.1 displays two charts for an Analog Devices inertial unit; one plot is for the gyroscope bias versus temperature and another is for the accelerometer scale factor versus temperature. Regardless of the temperature range it can be seen that the standard deviation levels for each sensor are quite large. The turn-on gyroscope bias error can easily reach

900 deg/hr and the accelerometer scale factor error could be 2500 ppm. These are large values for a filter to have to estimate with any sort of precision. The manufactured level of instability for the gyroscope is 54 deg/hr. The initial turn on bias of 900 deg/hr could be estimated using calibration methods, but with low-cost MEMS sensors these calibrations are rarely performed. To converge on the true deterministic bias the filter would either have to wait a long time or increase the noise variance of the first order Gauss-Markov model. Unfortunately, if the noise is increased then future estimations will not be as accurate. Assuming a correlation time of 1 hour and a variance of 500 deg/hr, the level of precision after one hour of estimation would only be 180 deg/hr (36% of 500), which is still significantly larger than the instability of 54 deg/hr. This trade-off between correlation time and noise is an issue when calibration cannot be performed entirely or when the turn on biases are so large that convergence takes too long.

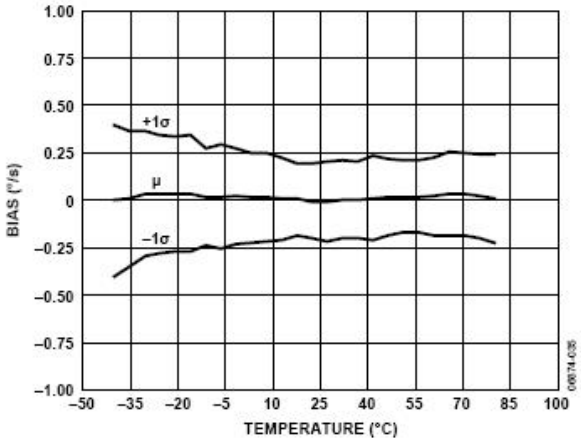

Figure 10. ADIS16355 Gyroscope Bias vs. Temperature

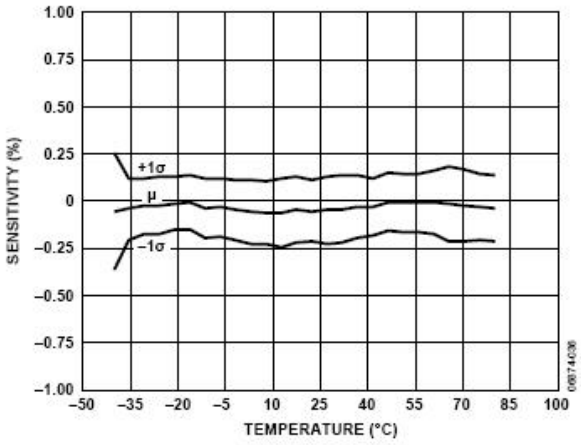

Figure 11. ADIS16355 Accelerometer Sensitivity vs. Temperature

Figure 4.1: ADIS16355 bias and scale factor versus temperature (ADIS16355 datasheet from: www.analog.com)

Since these *a priori* tuning parameters are deterministic in nature and inject the same noise over time into the system, their effect on state errors can be monitored over time and dynamics. Based on the coupling between error states certain inputs can be used to help

predict positional error states. Table 4.1 gives the inputs used to predict the three positional errors used in the ANN.

Table 4.1: Inputs to three positional networks

Error term (total #)	Inputs (#)
Latitude (total = 9)	• Time elapsed since last GPS measurement (1) • North velocity error (2) • Pitch error (3) • Accelerometer signal in the up direction (4) • Gyroscope angular rates in LLF (5, 6, 7) • Estimated accelerometer biases in north direction (8) • Estimated gyroscope biases in pitch direction (9)
Longitude (total = 9)	• Time elapsed since last GPS measurement (1) • East velocity error (2) • Roll error (3) • Accelerometer signal in the up direction (4) • Gyroscope angular rates in LLF (5, 6, 7) • Estimated accelerometer biases in east direction (8) • Estimated gyroscope biases in roll direction (9)
Height (total = 12)	• Time elapsed since last GPS measurement (1) • Up velocity error (2) • Roll error (3) • Pitch error (4) • Accelerometer signals in the east and north directions (5, 6) • Gyroscope angular rates in LLF (7, 8, 9) • Estimated accelerometer biases in up direction (10) • Estimated gyroscope biases in roll and pitch directions (11, 12)

The predictability of these position drifts is also driven by other errors that are amplified by dynamics and may have not been fully compensated by calibration,

- Residual non-orthogonalities (if calibration is not performed or is inaccurate).
- Misalignments between gyroscope and accelerometer triads.
- Vibration induced errors caused by road dynamics or engine vibrations. Rectification of the oscillatory motion creates a bias-like error. Many IMU signals

are products of angular rates or linear accelerations and if they are correlated to each other, say a vibration axis is in a skewed direction, then the average value of the product is not zero, as is the case of a purely oscillatory signal (Farell, 2007).

- Non-linear errors. These are typically negligible for vehicle applications but could be a potential source of additional error for pedestrian or aerial navigation. Vehicle navigation is typically very smooth compared to dynamics induced through walking or plane maneuvers (Shin, 2005).
- De-correlation errors. These errors are due to an alignment procedure that takes place in a certain orientation. Gyroscope and accelerometer biases are removed during the alignment, but due to the orientation of the strapdown system when this orientation changes the errors can actually be reinforced (Weston and Titterton, 2004).
- Lever arm errors. For low-cost systems the lever arms are often measured roughly with several centimetres of error. Bending errors, between the IMU and GPS antenna, can also create errors on the level of centimetres or decimeters (El-Mowafy, 1994). These errors propagate directly into the solution and can cause convergence issues if they become too large (Titterton and Weston, 2004).

Note that temperature induced errors have not been included in this analysis. Manufacturers are generating a better understanding of temperature induced errors as was apparent from the plots in Figure 4.1. However, if temperature has not been directly compensated from the bias and scale factor terms of the MEMS inertial sensors, then temperature can be added as an additional input to the system.

4.1.1 Training

The system used to perform estimation of the residual position errors is an ANN. The use of two parallel filters allows a target value to be created while the other filter simulates GPS outages by removing the GPS signals. These two filters are referred to as the navigation and simulation filters respectively (Goodall, 2006). The use of these filters allows the system to be trained incrementally by comparing the position outputs of the two filters to form target error curves as shown in Figure 4.2. Three distinct learning systems are employed for the three LLF position error terms.

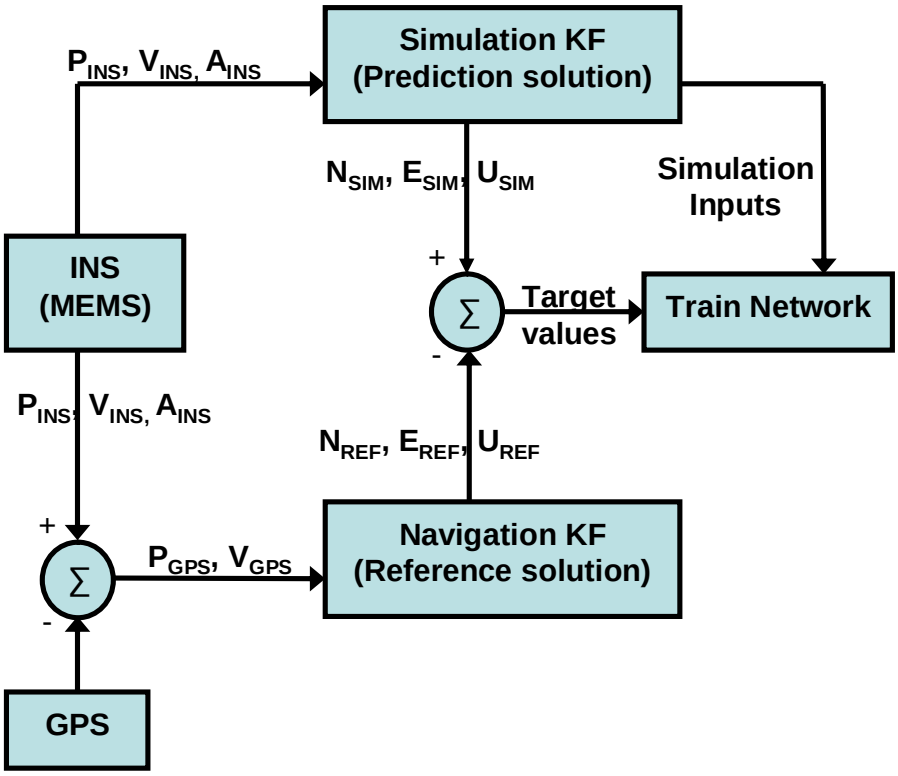

Figure 4.2: ANN training architecture

In the case of SGPS with MEMS grade sensors the training reference of the navigation filter is only precise to within a few metres. This in turn means that the target values used for training can also be in error of several metres. Therefore, very short outage periods, causing position drifts of only a few metres, cannot be expected to be improved using this training reference. The training errors become less noticeable as the length of the GPS outage times increase and the position errors grow. The purpose of this compensation is not for high accuracy, but for positioning to within several metres for applications such as vehicle navigation using a loosely coupled filter that loses GPS signals for longer than several seconds. For longer GPS outages the position drifts become a problem for such applications, but with intelligent compensation they can be minimized.

The training performance can be used as an indicator for future prediction performance. If there are no deterministic relationships between the inputs and target values of the system then the output of the ANN would simply inject zero mean noise into the system, with a variance equal to that of the reference training error. Chapter 5 deals with this situation and others related to ANN convergence issues and performance when combined with a Kalman filter. For now it is assumed that the filter converges on some deterministic error sources and predicts them within the noise level of the target signal.

4.1.2 Prediction

Once the networks are trained on a variety of dynamics and outage times they can be used to help compensate for real GPS outages occurring after the training has converged. During real outages only the navigation filter is used, without GPS, since the reference

cannot be generated when the signals are blocked. The inputs are sent to the three trained networks which then provide predictions of the position error. These predictions are then compensated from the EKF predictions to form the corrected positions. Based on the tuning and other residual errors this prediction can be quite helpful if there are significant left-over deterministic errors not estimated by the EKF. A general prediction architecture is given in Figure 4.3.

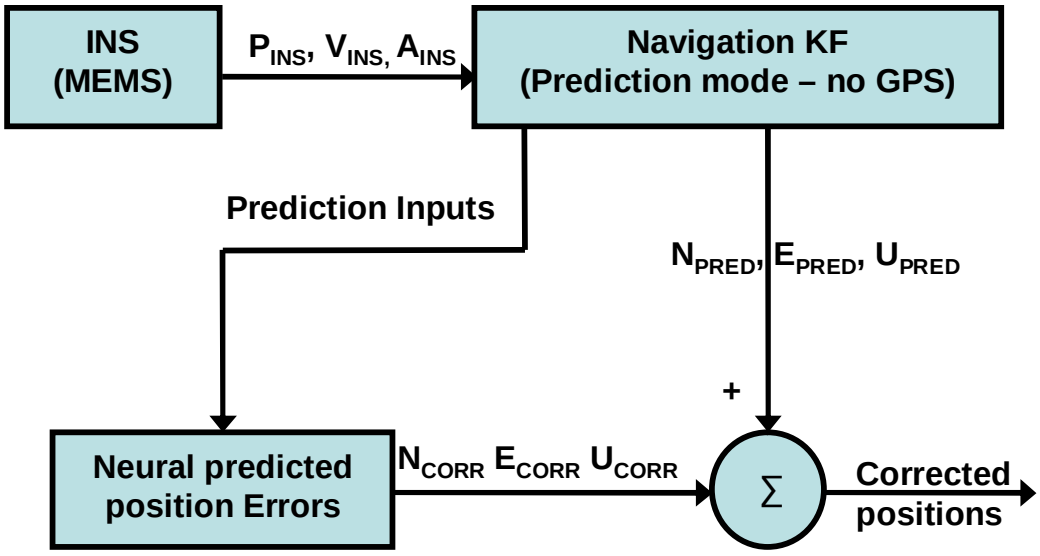

Figure 4.3: ANN prediction architecture

4.2 Network topology and details

The basic structure of the input/output relationship has been formed. Many designers then consider the specific details of a neural network to be less important, but as will be demonstrated there are several topologies that significantly enhance the performance of a system containing many inputs and a noisy reference signal.

4.2.1 Hidden layers

The most important aspect to designing a multilayer perceptron ANN is in the choice of the number of hidden layers. The universal approximation theory states that a single hidden layer is needed. But this theorem is only an existence proof (Haykin, 1994). The theorem makes two assumptions that are not practical for the system being estimated in this dissertation:

1. **The true continuous function is given**. This assumption is false since the reference signal is contaminated by noise and may be changing over time.
2. **A hidden layer of unlimited size is available**. This assumption is not practical from a processing point of view. Furthermore, the topology is fixed ahead of time.

An ANN containing a single hidden layer can have some serious estimation issues for complex problems due to global interactions between neurons in the hidden layer. The use of two hidden layers allows the first layer to extract local features and the second layer to deal with global features (Haykin, 1994). The input spaces can be segregated to get better approximation across both local and global features. This prevents the system from learning one global feature and forgetting another previously learned feature.

The use of two hidden layers can also help the system detect features that characterize input patterns, such as correlations between inputs. Dimensionality reduction is performed internally to the network without the need of a pre-design analysis tool such as Principal

Components Analysis (PCA). Fewer neural nodes are required in the second hidden layer due to this form of data compression formed in the first layer (DeMers and Cottrell, 1993).

The original design of the network contained only a single hidden layer with 50 hidden neurons. A revised network topology using 2 hidden layers with 10 neurons in the first layer and 5 in the second was found to approximate much more precisely and converge faster. Memory requirements were also decreased significantly since 35 fewer neurons were used. The number of parameters for the single and double hidden layers for the case of 9 inputs and a single output are given in Table 4.2.

Table 4.2: Parameter requirements for one and two hidden layers

Parameter	1 hidden layer (50 neurons)	2 hidden layers (15 neurons)
Input weights	500 (9*50)	90 (9*10)
Hidden #1 weights	50 (50*1)	50 (10*5)
Hidden #2 weights	0	5 (5*1)
Output layer weights	1	1
Biases	50	35
Total parameters	551	181

4.2.2 Activation function

The activation function of the hidden layers has to be non-linear and differentiable. The most commonly used activation function is the logsigmoid,

$$\log sig(n) = \frac{1}{1+e^{-n}} \qquad \textbf{(93)}$$

The resulting derivative of the logsigmoid is,

$$\partial \log sig(n) = \frac{e^{-n}}{\left(1+e^{-n}\right)^2} \quad \textbf{(94)}$$

which can be simplified for iterative use as,

$$\partial \log sig(n) = \log sig(n)\left[1 - \log sig(n)\right] \quad \textbf{(95)}$$

Although this activation function has a simple form of derivative used for backpropagation, it is generally found that the network can learn faster when the activation function is asymmetric (Russo, 1991). An activation function is asymmetric if,

$$f(n) = -f(n) \quad \textbf{(96)}$$

The most commonly used asymmetric function is the hyperbolic tangent,

$$a\tan(bv) = a\left[\frac{1-e^{-bv}}{1+e^{-bv}}\right] = \frac{2a}{1+e^{-bv}} - a \quad \textbf{(97)}$$

Commonly used values for a and b are (Guyon, 1991),

$$a = 1.716 \quad \textbf{(98)}$$

$$b = \frac{2}{3} \quad \textbf{(99)}$$

So the hyperbolic tangent becomes,

$$1.716\tan\left(\frac{2}{3}v\right) = \frac{3.432}{1+e^{-\frac{2}{3}v}} - 1.716 \quad \textbf{(100)}$$

The general derivative of the hyperbolic tangent is,

$$\partial a\tan(bv) = \frac{2abe^{-bv}}{\left(1+e^{-bv}\right)^2} \quad \textbf{(101)}$$

The derivative of the hyperbolic tangent can also be simplified as a function of the original activation function for iterative use. Consider a constant c, currently unknown,

$$c - \left[a\tan(bv)\right]^2 = c - \left[\frac{4a^2}{\left(1+e^{-bv}\right)^2} - \frac{4a^2}{\left(1+e^{-bv}\right)} + a^2\right] \quad \textbf{(102)}$$

Now let $c = a^2$,

$$c - \left[a\tan(bv)\right]^2 = \left[\frac{4a^2 - 4a^2\left(1+e^{-bv}\right)}{\left(1+e^{-bv}\right)^2}\right] = \frac{e^{-bv}}{\left(1+e^{-bv}\right)^2} \quad \textbf{(103)}$$

Now also introduce a scaling factor d,

$$d\left[c - \left[a\tan(bv)\right]^2\right] = \frac{de^{-bv}}{\left(1+e^{-bv}\right)^2} \quad \textbf{(104)}$$

So that,

$$d = 2ab \quad \textbf{(105)}$$

Equations (106) through (111) give a summary,

$$a\tan(bv) = \frac{2a}{1+e^{-bv}} - a \quad \textbf{(106)}$$

$$\partial a\tan(bv) = d\left[c - \left[a\tan(bv)\right]^2\right] \quad \textbf{(107)}$$

$$a = 1.716 \quad \textbf{(108)}$$

$$b = \frac{2}{3} \quad \textbf{(109)}$$

$$c = 2.944656 \quad \textbf{(110)}$$

$$d = 2.288 \quad \textbf{(111)}$$

The choice of variable a by Guyon (1991) relates to the desired response of input target values. The target value should be within the range of the activation function, or for

practical applications should be some small distance away from the limiting value of the activation function, namely a. If the small delta is set to 0.716 then the target values can range between ±1, which is convenient for scaling purposes. This is more practical than using $a = 1$ because the target inputs would rail the activation function.

4.3 Application of intelligent compensation during GPS outages

To test the applicability of this neural compensation, field test data was collected using low-grade MEMS accelerometers and gyroscopes combined with SGPS. Training was performed on one dataset and testing on another collected shortly after the first. Both trajectories are shown combined in red overlay on Figure 4.4. The test was performed in open sky conditions near the Springbank airport just west of the city of Calgary.

Figure 4.4: Open sky trajectory

The data was collected using a Chevrolet Express Cargo van 2500, shown in Figure 4.5. MEMS accelerometers and gyroscopes from Analog Devices Incorporated (ADI) were used along with GPS data from a NovAtel OEM4 receiver. The inertial assembly was designed by members of the MMSS research group at the University of Calgary with a system cost of approximately 150 US dollars. This low-cost unit is applicable for civil applications and is small enough to be embedded within a GPS receiver as shown in Figure 4.6.

Figure 4.5: Test vehicle

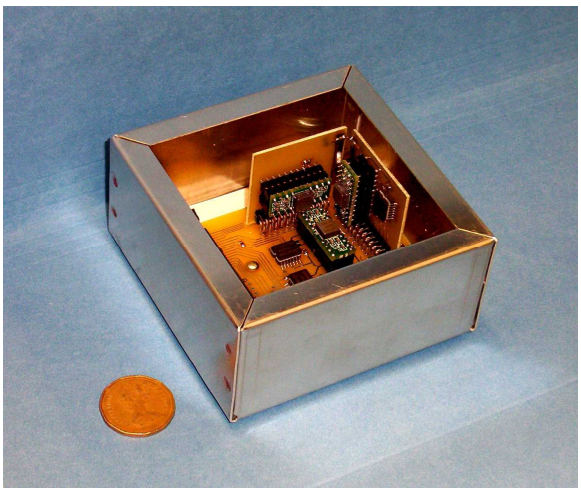

Figure 4.6: ADI inertial sensor assembly

A high-end tactical grade IMU using fiber optic gyroscopes was included in the field test to provide a quality reference for error analysis. The LN-200 INS from Litton was combined with DGPS from a NovAtel OEM4 receiver to provide a reference good to within a decimetre of the actual unknown true trajectory. This reference was possible due to the lack of any natural GPS outages and a minimum of seven satellites available throughout the test. The reference receiver located in Springbank is shown in Figure 4.7. The flat treeless terrain provided excellent GPS signal quality.

Figure 4.7: Base station for DGPS in Springbank

The inertial EKF was run in forward filter mode with GPS and without additional aiding, such as nonholonomic constraints. Nonholonomic constraints are velocity constraints on the vehicle that constrain lateral and vertical velocity. These have been shown to be very useful as velocity updates to the filter, see Niu et al. (2007), but were not considered in any of the following analyses due to their complex relationships with other vehicle error sources.

Although some small residual misalignment and temperature induced errors were already present in the MEMS IMU data, further errors were introduced to replicate an example of poor filter tuning. Since the temperature variations were small (from 10-20 degrees Celsius) and the non-orthogonalities were largely removed through a six position static test, these tuning induced errors dominated the residual errors.

The 21 state loosely coupled EKF required estimates of the accelerometer and gyroscope bias noise levels to optimally combine the inertial measurements with the GPS data. The ideal parameters for these values were developed through extensive pre and post test procedures. Static lab tests, using Allan Variance, were combined with intelligent tuning methods from Chapter 6 to generate the ideal *a priori* parameters. Deviations from these ideal parameters were then introduced by using a quarter the ideal values for both the gyro and accelerometer bias noise levels. By fixing the correlation times at one hour and using smaller than ideal tuning parameters, incorrect bias estimates were likely throughout the test which took about an hour of driving time. These incorrect bias estimates are likely since systematic errors on the innovation sequence are created by choosing variance values

too optimistically (Petovello, 2003). Figure 4.8 gives a plot of the accelerometer bias estimation during the test, which varied largely. An accelerometer residual error of 0.05 m/s^2 would be roughly 22 metres after 30 seconds of drift. This additional error could easily be injected with poor bias estimation.

Of course, non-compensated motion dependent errors can also contribute to poor bias estimation (Titterton, 2004). Misalignment between accelerometer and gyroscope triads can create an error on the bias estimation depending on the orientation of the strapdown platform. When this error is combined with lever-arm and bending effects, the overall error becomes less predictable. De-correlation and rectification errors also contribute as motion dependent terms. Rectification errors can be especially significant in vehicles that are subject to road or engine vibrations.

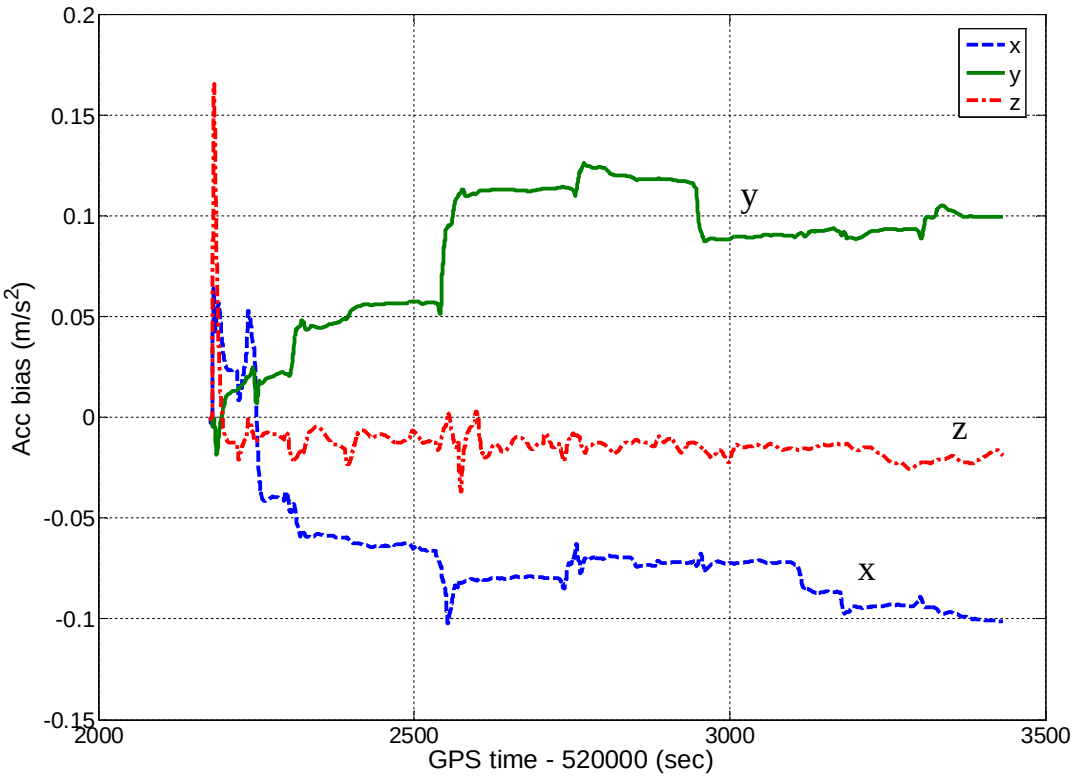

Figure 4.8: ADI accelerometer bias estimation

Based on 40 GPS signal outages, each lasting 30 seconds, these tuning parameters caused an average drift of about 55 metres in the horizontal direction. In comparison to the ideal case, which had drifts of 25 metres on average, the poorly tuned results contained an additional 55% in positional state errors.

4.3.1 Training Data

The training was performed sequentially as new inputs and targets arrived. The training was performed in real-time with a one outage delay. The network was updated in time for the next training outage since there was a minimum of 90 seconds between all the training

outages. Hence, the network could be considered updated with n-1 outages in real-time, given n real-time training outages.

A total of 40 simulated outages, each lasting 30 seconds, were used to train the networks. The length of the outages was chosen based on a typical time without GPS when navigating a vehicle amongst tall buildings. This of course depends on the receiver used and the environment. This length of outage also allows the EKF position errors enough time to diverge past the noise threshold of the training reference so that useful neural compensations can be made.

The results after the final batch training are shown in Figure 4.9. The target errors are the smooth drifts in black, while the trained network outputs are the noisy estimations in red. Although there are two outages, circled on the figure, which did not converge, the rest of the trained outputs are within several metres of the target values, which is as accurate as possible given the training reference that uses SGPS and MEMS inertial sensors.

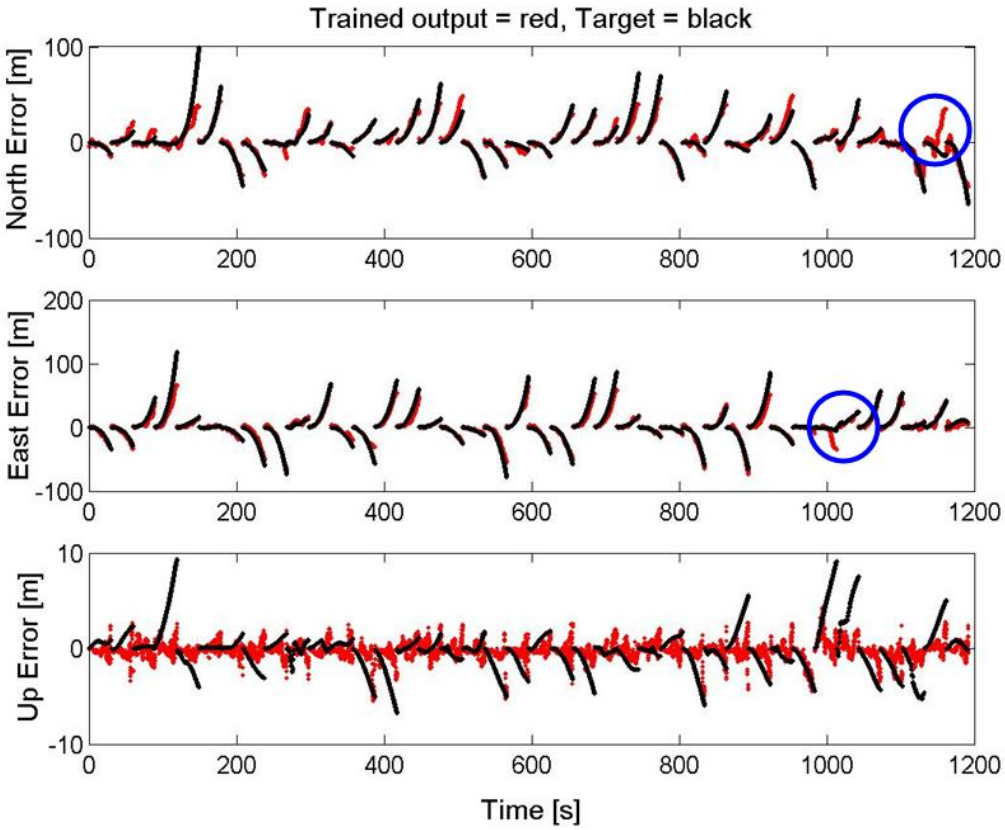

Figure 4.9: Training results on 40 outages

The east and north networks appear to have converged on appropriate values after the sequential outages, but the up network estimation is noisy. This noise level is on the magnitude of several metres and is a result of training on SGPS measurements combined with an MEMS grade IMU. This type of reference often exhibits an error that seems like noise due to the many contributing error terms of both the SGPS and IMU. The accuracy of the training results cannot be expected to be better than the reference, so this noise level is introduced into all three networks. Its effect is more apparent with the up network since the up errors have much smaller magnitudes than the horizontal errors, which are on the

level of 2 to 4 metres in the horizontal plane and 6 to 8 metres vertically with an average noise level of 3.7 metres horizontal and 6.9 metres vertical. These ranges are within the error level of the single point GPS reference.

4.3.2 Prediction Results

Since there were no natural GPS outages in this open sky field test, GPS test outages were created by removing the GPS data input to the EKF, similar to how the training outages were generated. The trained networks were then used to generate predictions of the drifts using the temporal and dynamic inputs during the test outages. The results for 11 test outages, each 30 seconds long, are given in Figure 4.10. These simulated outages were from the same dataset as the training data, but occurred during the last 20 minutes of driving, independent of the training data that was collected during the initial 40 minutes of the field test. The actual errors are shown as smooth black drifts while the neural predictions are shown as noisier estimations in purple.

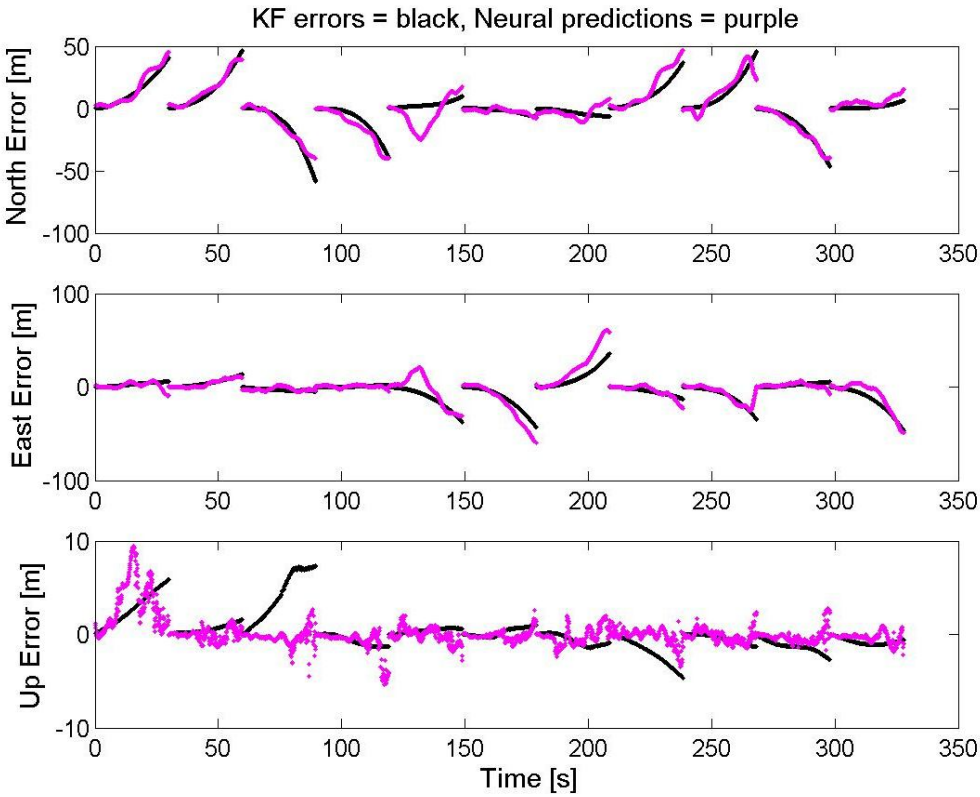

Figure 4.10: Prediction results on 11 test outages

The east and north network predictions show good correlation to the actual Kalman filter errors while the up network is mainly noise, as was expected from the training results. The neural predictions were differenced from the actual errors to form residual neural errors after compensation. The sum of the absolute errors at each epoch (100Hz) was used to compare the relative improvement between the Kalman filter uncompensated and the neural compensated cases. This performance measure was used since the neural compensated error signals contained significant noise which meant the largest errors did not necessarily occur at the end of an outage period. Furthermore, the accumulation of these noisy errors could pose a potential problem for some applications, so a cumulative sum of the errors

was needed to analyze all the errors over the outage period. This was done for the horizontal networks as shown in Figure 4.11.

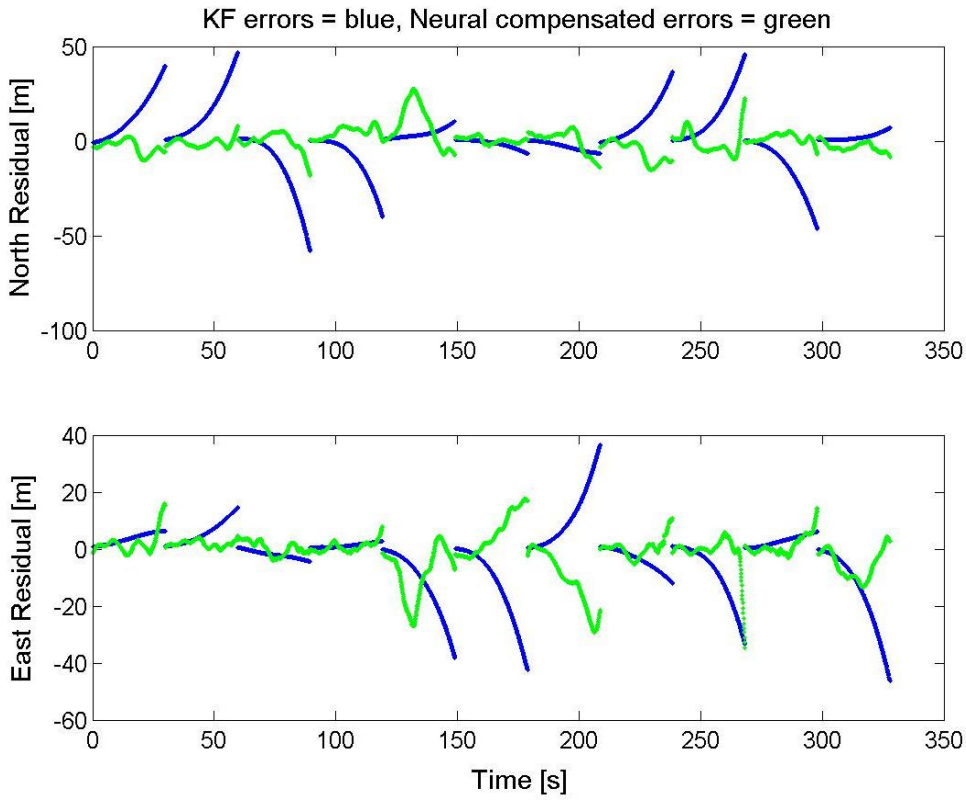

Figure 4.11: Prediction results after compensation

Table 4.3 gives the individual and overall improvements for each network after compensation of the predicted errors. Maximum values could not be used for a fair analysis of improvement since the neural compensations had noise levels that sometimes created the largest errors. The absolute sum of the errors was used over each outage as a more representative measure,

$$\%improvement = 100 * \left(\frac{\sum_{i=1}^{n} EKF\_error(i) - \sum_{i=1}^{n} Neural\_error(i)}{\sum_{i=1}^{n} EKF\_error(i)} \right) \quad \textbf{(112)}$$

Positive values indicate improvement after compensation while negative values indicate degradations. The north network displayed large improvements for 8 of the 11 outages. The three outages that were degraded were among the smaller drifts, all being less than 20 metres. The neural compensation cannot be expected to improve upon such small errors since their drifts are already close to ideal and approach the noise level of the predictions.

Table 4.3: Neural improvements after compensation

Outage	North (%)	East (%)	Up (%)
1	72	22	25
2	87	67	-2
3	76	-7	-4
4	52	-7	-67
5	-166	26	-5
6	25	46	-9
7	-76	2	-97
8	46	40	7
9	61	70	-33
10	84	17	-67
11	-61	57	-89
Overall	52	40	-11

The overall numbers in the last row represent a percentage improvement when equation (112) is used for *i* over all of the outage times. The overall improvement is not an average of the individual outage improvements since some improvements are worse or better, in absolute sense, depending on the magnitude of their original drift errors.

The east network performed well with an overall improvement of 40%. 9 of the 11 outages improved with neural compensation, while the two smallest outages degraded slightly due

to the noisy training reference and near zero EKF drifts. The up network was degraded overall, due to the noisy reference data and Kalman filter drift errors, less than 5 metres. To improve the up network the errors would have to drift beyond the noise threshold of the training reference, which would mean longer GPS signal outage times.

The overall improvements for the horizontal compensation networks agree well with the computed possible improvement of 55% when compared to an ideally tuned filter. The maximum average improvements should not exceed 55% but should come as close as possible to this value within the accuracy of the training reference and additional residual errors such as temperature and misalignment induced drifts.

4.4 Real versus simulated GPS outages

Testing of simulated outages proved that the neural compensation networks are capable of removing most of the deterministic errors introduced through poor tuning of the Kalman filter. Unfortunately, the behaviour of real GPS outages is more complex than simply losing the signals and then reacquiring them. In simulated outages the GPS quality before and after the signal outages was not altered, but during real outages the quality is often degraded. To analyze these characteristics a field test including downtown data and real GPS outages was conducted.

The test setup used to collect the data was exactly the same as with the simulated GPS outages. The same van was used along with the ADI and NovAtel OEM4. The reference for error analysis was again generated using a tactical grade INS and DGPS, but the

accuracy of the unit was degraded due to real GPS outages and additional reflected GPS signals in the form of multipath. The final accuracy of this reference was good to within 2 metres RMS.

The same 30 second simulated training outages were used along with additional 60 second outages. Since neural networks are very good interpolators but poor extrapolators, these additional 60 second outages were needed for longer real outage prediction times (Cichocki and Unbehauen, 1996).

4.4.1 Analysis of several real GPS outages

The quality of GPS before an outage period can largely impact the growth of errors during the outage. Real GPS signal outages in downtown environments typically display poor quality before an outage due to multipath. Depending on the magnitude of this multipath the Kalman filter can begin to diverge before the outage actually begins, which then impacts the usefulness of the neural predictions.

An example of this case is shown in Figure 4.12 through to Figure 4.17. The pink dots are GPS measurements, the black line is the tactical grade reference and the blue line shows the MEMS Kalman filter solution. Figure 4.12 gives an overview and numbers the stages of this multi-GPS outage period. This figure also shows the direction of travel and highlights the divergence of the original IMU/GPS solution in comparison to the east/west, north/south true solution.

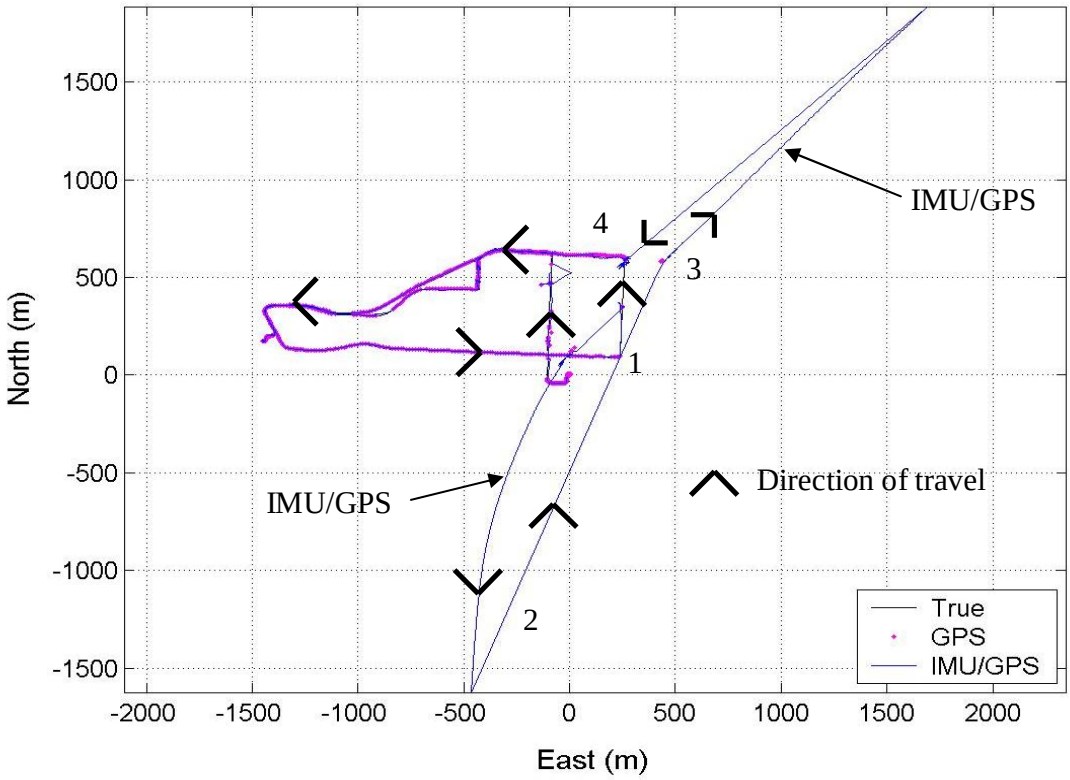

Figure 4.12: Original solution using poor GPS measurements

The first loss of GPS signals starts immediately after turning left onto the east-most street. The behaviour of the filtered horizontal solution in Figure 4.13 shows a minimal drift of less than 50 metres during this 40 second outage period. Several GPS measurements are then received while stopped at a light. These measurements bring the filtered solution back within 5 metres of the reference, but just after the vehicle starts moving the GPS signals are seriously contaminated by multipath, stronger than the direct signal, which results in the solution being pulled off course by over 250 metres.

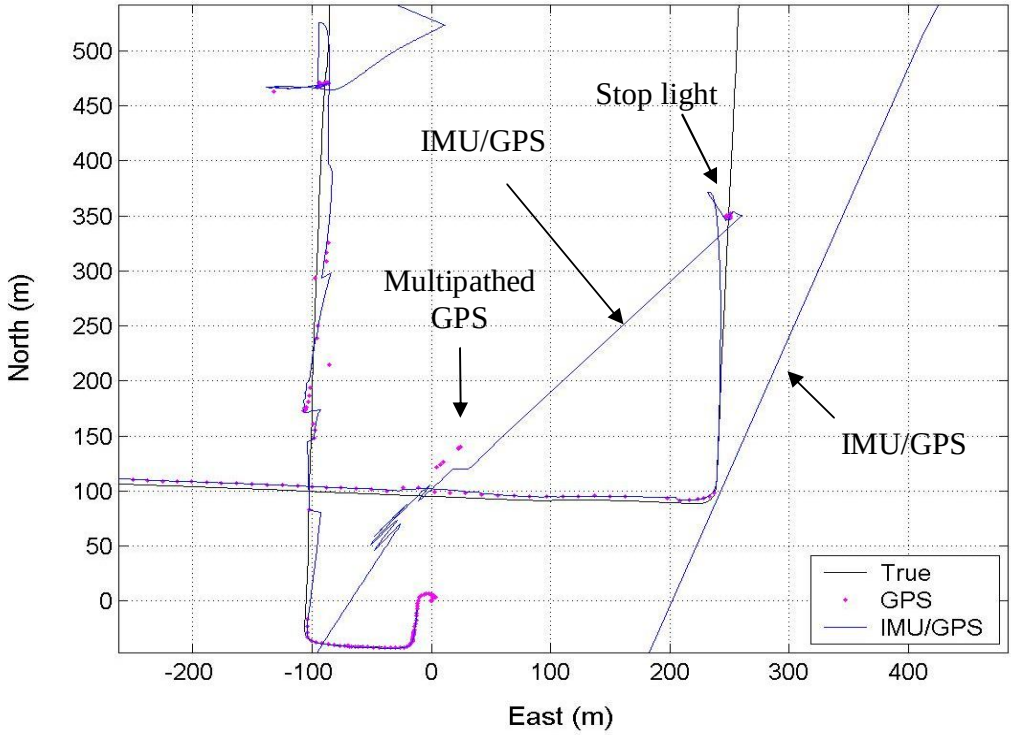

Figure 4.13: Zoom in #1 of original solution

This multipath is likely due to the tall buildings that surround the vertical arrow tip in Figure 4.14 which shows an overlay for this section of the trajectory. Since the GPS measurements are weighted strongly when using a low quality MEMS unit, it is difficult to prevent this situation.

Figure 4.14: Overlay of downtown buildings and test trajectory

After these poor quality GPS measurements, another outage period is started, but this time the filter drifts very quickly because of the poor initial GPS measurements. The solution drifts over 1500 metres in less than 1 minute (Figure 4.15) before further GPS measurements are received in Figure 4.16.

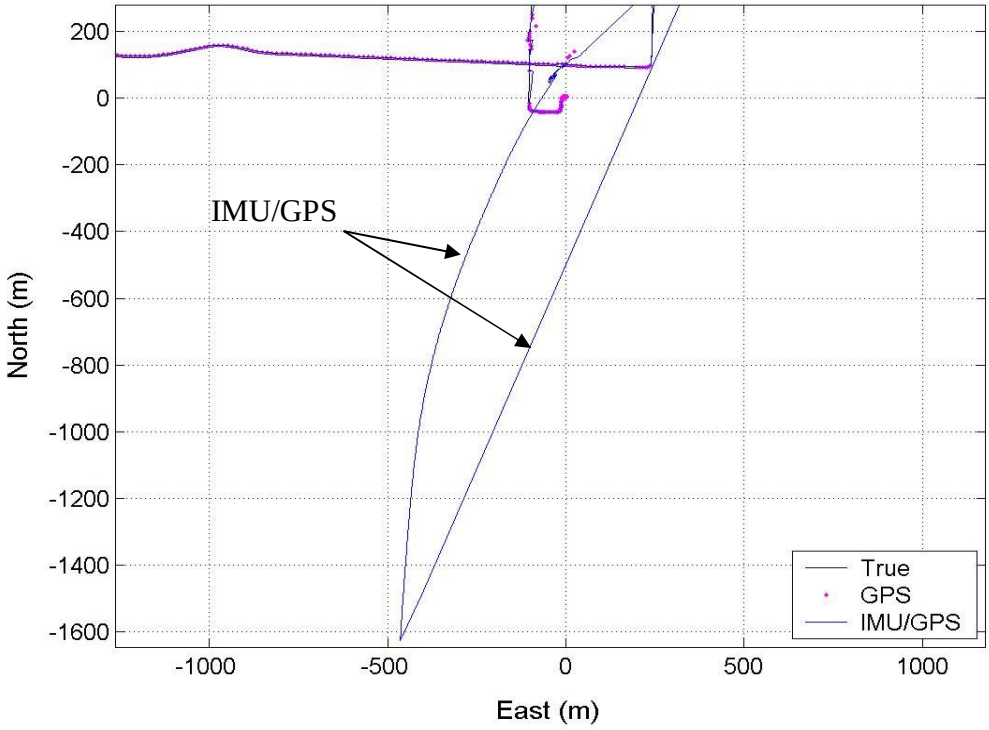

Figure 4.15: Zoom in #2 of original solution

These new signals are also affected by multipath, but are able to bring the solution to within 200 metres of the reference. These few updates are followed by a third GPS outage of 30 seconds. This time the filtered solution drifts in the north-east direction in error of nearly 1400 metres.

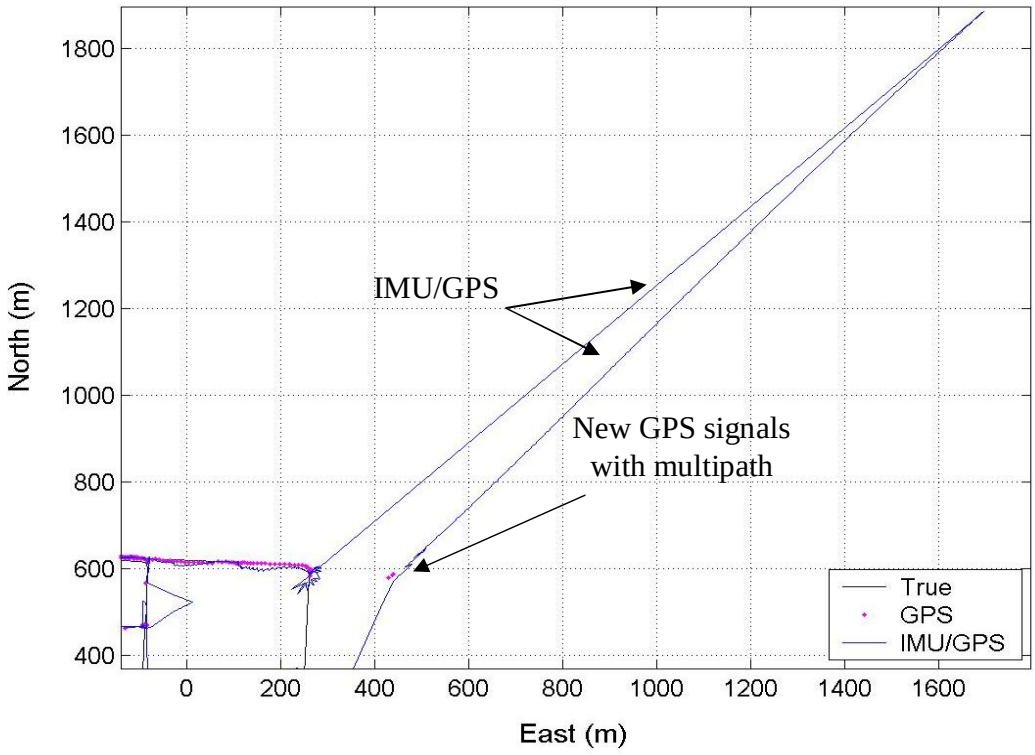

Figure 4.16: Zoom in #3 of original solution

When good GPS measurements are finally received in Figure 4.17 the solution takes some time to re-converge. This re-convergence time is typically much faster with simulated outages since the quality of GPS is less affected by multipath. This is evidenced in the next figure by the seesaw convergence behaviour of the IMU/GPS solution.

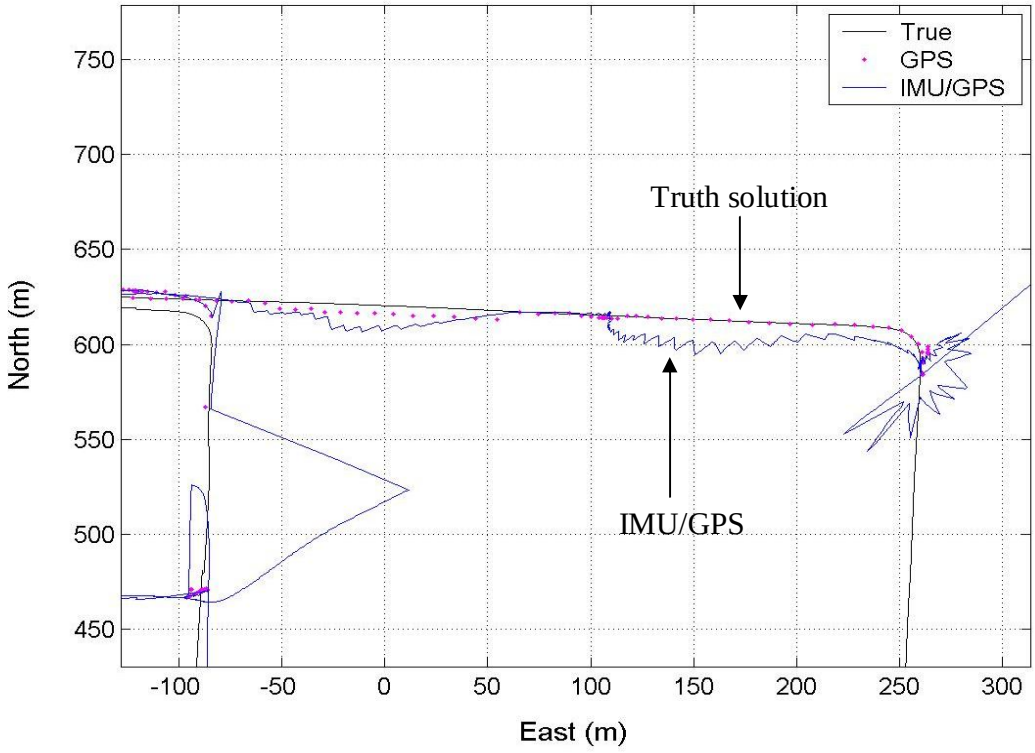

Figure 4.17: Zoom in #4 of original solution

The street view of this convergence period is shown in Figure 4.18. There are still very large buildings but the street is wider so additional GPS signals can be obtained.

Figure 4.18: Street and surrounding buildings during reacquisition of GPS

4.4.2 Filtering the GPS data

The large drifts shown in Figure 4.12 would not be well predicted by the trained networks using simulated outages. The first outage period that drifts only 50 metres would be similar to simulated outages, but the large biases created by the multipath in the other outages would not be considered when training the networks. To avoid these large biases and fast drift of the Kalman filter, innovation sequence testing and a velocity filter were used to remove poor GPS measurements.

The innovation sequence testing performed a global test to detect blunders in the latitude, longitude and height updates. The global test statistic used was (Mohamed, 1999),

$$T_k = \left(z_k - H_k x_k^-\right)^T \left[R_k + H_k P_k^- H_k^T\right]^{-1} \left(z_k - H_k x_k^-\right) \quad \textbf{(113)}$$

Or in more compact form,

$$T_k = v_k^T C_{vk}^{-1} v_k \quad \textbf{(114)}$$

Where v_k is the innovation sequence. If this test statistic exceeded a threshold level for a defined confidence level then all three GPS measurements were disregarded. The issue with this method was that at a low confidence level (eg. 95%) too many measurements were discarded and the filter diverged. When the confidence level was increased (99.5%) the filter reached a state of convergence, but there still remained many additional multipathed GPS measurements.

To deal with additional blunders a velocity filter was used on any measurements that passed innovation sequence testing. The velocity was derived by averaging the kinematic periods of the filtered solution over a 10 minute window. The standard deviation of the vehicle speed was also calculated over this same interval and the maximum vehicle velocity was set by adding three times the standard deviation to the mean. This ensured that very poor GPS measurements were ignored if it was found that the vehicle travelled an unreasonable distance based on previous velocities. For example, if a GPS measurement placed the vehicle 2 kilometres from its previous position after a minute from the last GPS measurement, this would mean the vehicle travelled 120 km/hr on average. If the previous driving was within an urban environment at 50-60 km/hr then this GPS measurement would

likely be ignored. The impact of filtering the occasional good measurement was deemed less important than rejecting bad measurements, so the 3 sigma threshold was used.

The results of innovation sequence testing and velocity filtering were dramatic, reducing the largest errors by nearly one kilometre. Figure 4.19 shows the new solution, with the largest drift being less than 400 metres. When this figure is compared to Figure 4.12, the result of minimizing poor GPS measurements is obvious.

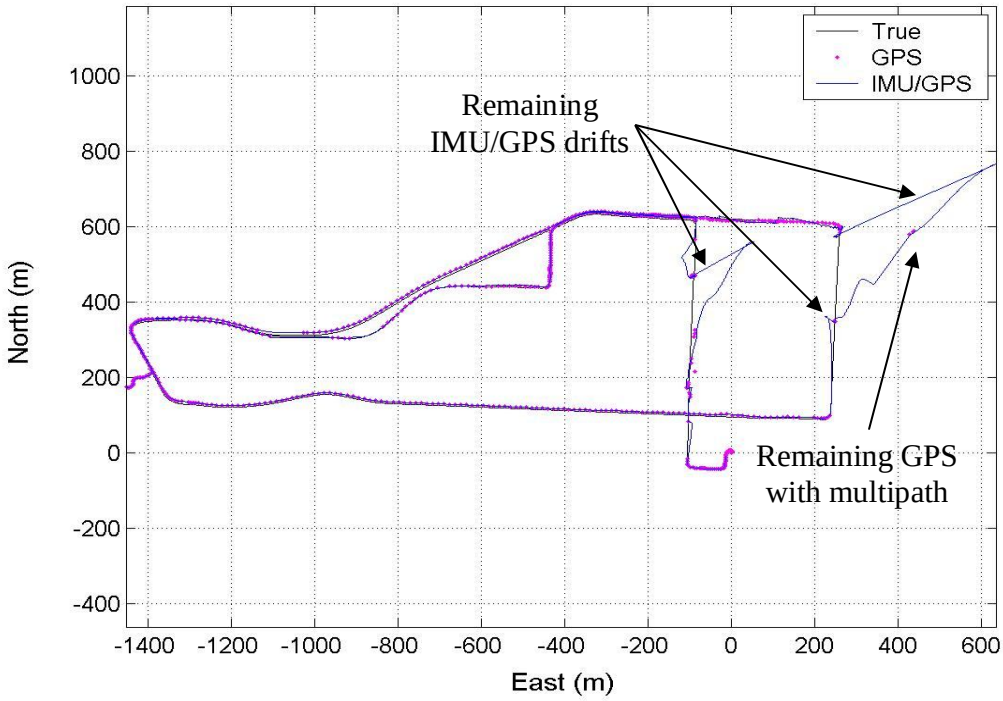

Figure 4.19: Filtered solution for EKF

There were a total of 21 largely biased GPS measurements in the original solution. The blunder detection scheme was able to remove 18 of these measurements with the remaining 3 contributing to the largest remaining bias offsets and resulting drifts. Unfortunately, there

is no logical software method for removing these last three measurements since it is possible that the vehicle did indeed move 300 metres within 1 minute and innovation testing was unable to remove these without filter divergence.

4.4.3 Prediction results using real outages

Using this GPS filtered solution, the neural compensation was applied to 5 real outages. The effects of two of these outages are quite apparent in Figure 4.19, with the other three being significantly smaller and similar to the simulated outage results. The length of each outage along with the maximum east and north errors are given in Table 4.4. The effect of multipath bias error is especially relevant to the fifth outage.

Table 4.4: Filter error during real GPS outages

Outage	Length (sec.)	Max. north drift (m)	Max. east drift (m)
1	40	78	138
2	14	35	29
3	42	17	22
4	50	24	83
5	22	180	365

The prediction results are shown in Figure 4.20, which also gives an idea of the initial biases due to GPS multipath for each outage.

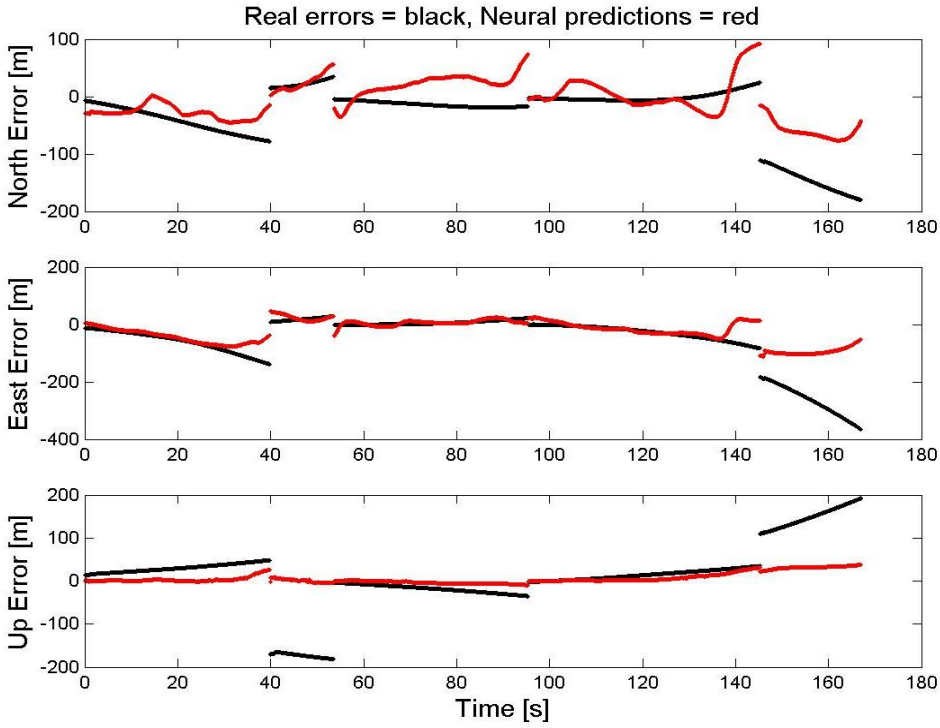

Figure 4.20: Prediction results during real GPS outages

Table 4.5 gives the overall and individual improvements for the 5 outages. 2 of the 5 predictions were very poor, but the overall errors were still decreased significantly. These improvements would have been significantly better had the initial biases been completely removed. Unfortunately, in a real situation these biases cannot be fully removed so the predictability of the outages would not be solely based on the Kalman filter tuning parameters. Outage drift rates would also have to account for GPS multipath induced biases, which depend on a variety of complex parameters such as environment geometry. Since these cannot be easily modelled, the best case would be to assume the biases are zero as was done in this analysis.

Table 4.5: Neural improvements during real GPS outages

Outage	North (%)	East (%)	Up (%)
1	48	81	3
2	32	34	3
3	-180	-8	22
4	-180	74	30
5	28	33	26
Overall	11	43	16

CHAPTER 5: RELIABILITY OF THE NETWORK

The neural compensation results shown in Chapter 4 require certain conditions to achieve their improvements over the uncompensated EKF errors. The inputs used for the predictions were similar to those used for training, and the networks were fully converged on the training data. In reality similar training data and the convergence of the networks cannot be guaranteed. In these cases the network predictions could diverge and create large degradations over the uncompensated EKF errors.

In a perfect system the neural compensations would only be applied once improvements can be guaranteed, without any degradation beyond the EKF uncompensated errors. This means that the neural compensations have to be better or equal to the EKF predictions over the entire length of the outage. By looking at Figure 5.1 (from Chapter 4, reproduced below) it can be seen that this is not generally the case. The EKF predictions are typically better towards the beginning of an outage due to the neural prediction noise. At the end of the outage the EKF errors drift significantly so the neural residual errors are smaller. Therefore, the optimal system would apply the neural compensations once the EKF drifts are larger than the neural residual noise level, and only if the input data during the prediction stage is similar to the inputs used for the converged training networks (i.e. representative training data).

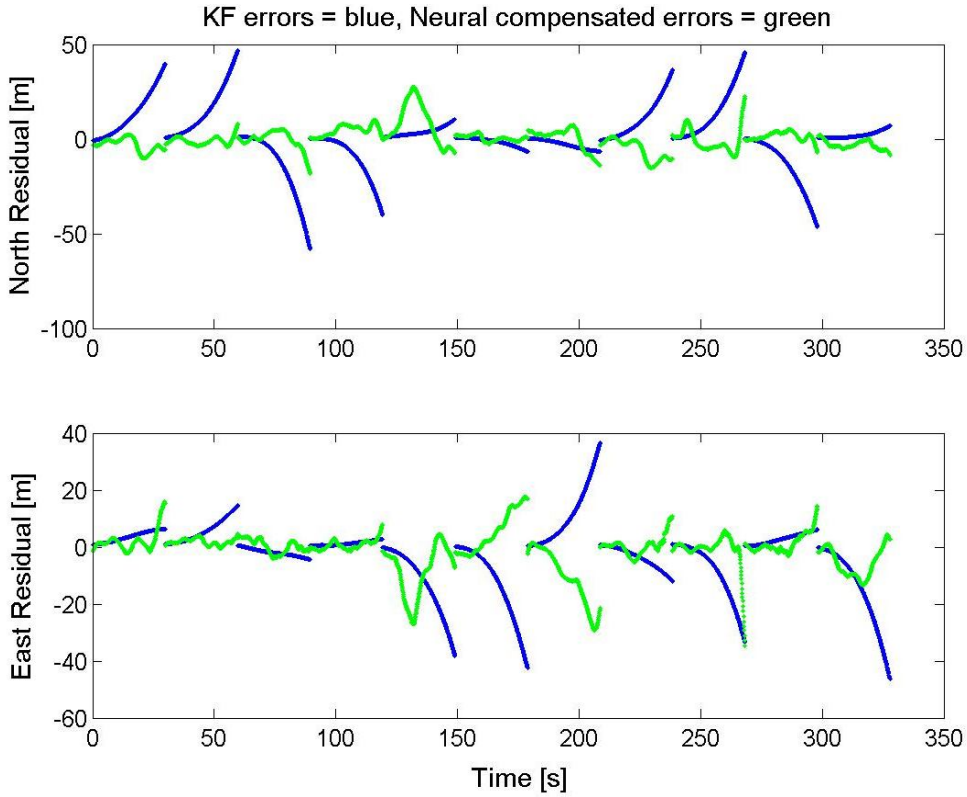

Figure 5.1: Neural compensations (from Chapter 4)

5.1 Fuzzy monitoring

A fuzzy inference system (FIS) is a logical choice for weighting the neural compensations with the original uncompensated EKF predictions. The system takes several inputs such as the neural training error, the outage time, and the differences between the input data used for training and predicting. Based on these inputs, and rules governing their values, the weighting of the EKF and neural predictions are changed. If the weighting is close to unity, then the EKF estimates are used with very little emphasis given to the neural predictions of these errors. If the weighting is close to zero, then the neural predictions are used in full to compensate the EKF predictions.

Weightings close to zero (strong neural weightings) should only be obtained for long GPS outages using a well trained neural compensation on prediction inputs within the range of the training inputs. If the network has not converged then more emphasis is given to the EKF predictions. Or if any of the prediction inputs are outside the range of the training inputs, then only minimal weighting is given to the neural compensations.

5.1.1 Fuzzy inputs, outputs and rules

These general heuristics fit nicely into an expert FIS system. The inputs to the system are:

1. The outage time
2. The current ANN training error
3. The differences between the prediction and training inputs for all network inputs.

The third set of inputs calculated the absolute differences between the prediction inputs and the closest training input. If the absolute differences were small then it was assumed that the ANN prediction would be capable of approximating well. If the difference was large then the ANN would not be expected to generalize well.

The output of the FIS was a simple weighting between the EKF and ANN compensated predictions. An output close to unity would weight the EKF predictions highly, while an output close to zero would give all the weight to the ANN compensated predictions. Six rules were made to relate the inputs to the output weighting factor:

1. If the outage time is short AND the neural training error is large AND any of the absolute differences in prediction and training inputs is large THEN give high weighting to the EKF predictions.
2. If the outage time is medium AND the neural training error is small AND all of the absolute differences in prediction and training inputs are small THEN give medium weighting to the EKF predictions.
3. If the outage time is long AND the neural training error is small AND all of the absolute differences in prediction and training inputs are small THEN give minimal weighting to the EKF predictions.
4. If the outage time is long AND the neural training error is small AND any of the absolute differences in prediction and training inputs are large THEN give high weighting to the EKF predictions.
5. If the outage time is very short THEN give very high weighting to the EKF predictions.
6. If the outage time is very long AND the neural training error is small AND all of the absolute differences in prediction and training inputs are small THEN give very minimal weighting to the EKF predictions.

5.1.2 Adaptive membership functions

A fuzzy system was useful due to the number of rules and uncertain nature of these rules. Assuming that the network is well trained and the prediction inputs are in the range of the trained inputs, the fuzzy system should still weight the early EKF predictions stronger than the late EKF predictions due to the neural prediction noise level. This weighting strategy

also depends on how well the Kalman filter has been tuned. If the filter has been tuned ideally, then the EKF predictions will drift more slowly compared with a poorly tuned EKF.

The actual definition of the outage time being short, medium or long is in question. The cutoff point for weighting between the filter and neural compensated predictions can change. Figure 5.2 gives a visual representation for two different tuning scenarios. The cutoff points for weighting between the EKF and neural compensated predictions would be different depending on the level of drift.

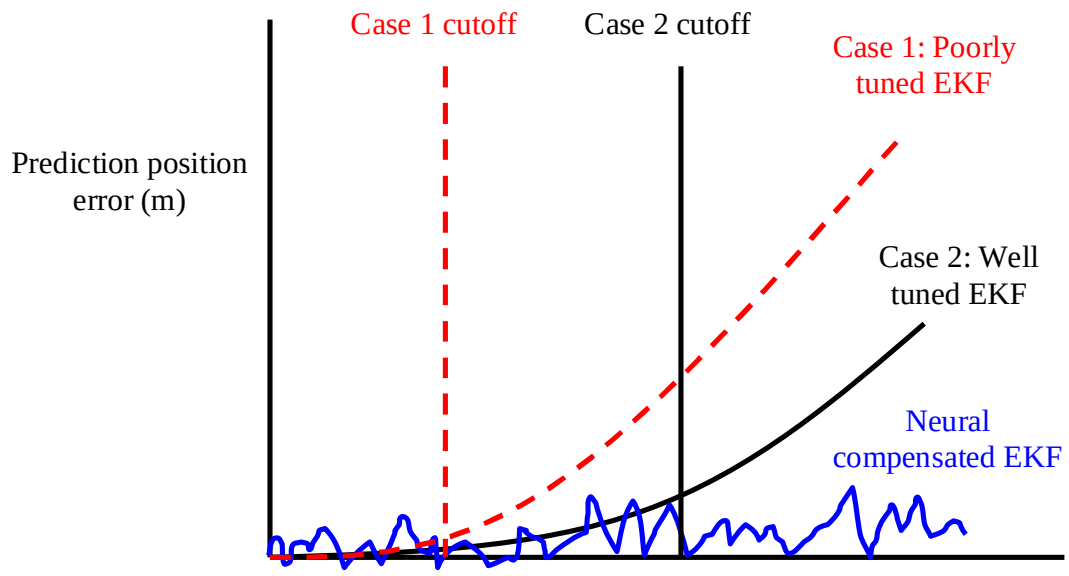

Figure 5.2: Drifts resulting from different *a priori* tuning

Since the tuning effectiveness cannot be known ahead of time when using MEMS grade sensors, an adaptive method is needed to characterize the time definitions. An adaptive neural fuzzy inference system (ANFIS) was used to adjust the membership functions of the

outage time input parameter. ANFIS is very similar to a regular FIS system, but the membership functions can be adjusted based on training data and a backpropagation learning algorithm.

The training inputs for the ANFIS were the outage times and the targets were discrete weightings, being either 1 or 0. A target value of 1 represented full EKF weighting while 0 gave full weight to the neural compensations. The targets were generated by setting the 0 or 1 value based on the case that contained the smallest residual absolute error during prediction. A 1 was assigned if the EKF error was smaller and a 0 if the neural compensated error was smaller.

The general behaviour of the weighting scheme for short, medium and long outages can be approximated by using evenly spaced triangular membership functions spanning the length of the possible outage times. Using outage length as the sole indicator of the weighting, the input-output relationship might look something like that shown in Figure 5.3. This result uses membership functions developed by manual adjustment based on expert tuning and an understanding of the expected drift characteristics; ANFIS is not applied here. Even by using knowledge of the desired input-output relationship there are still some odd characteristics of this figure. For instance, the increase in weighting near 25 seconds should not be possible, and neither should the increase near 3 seconds.

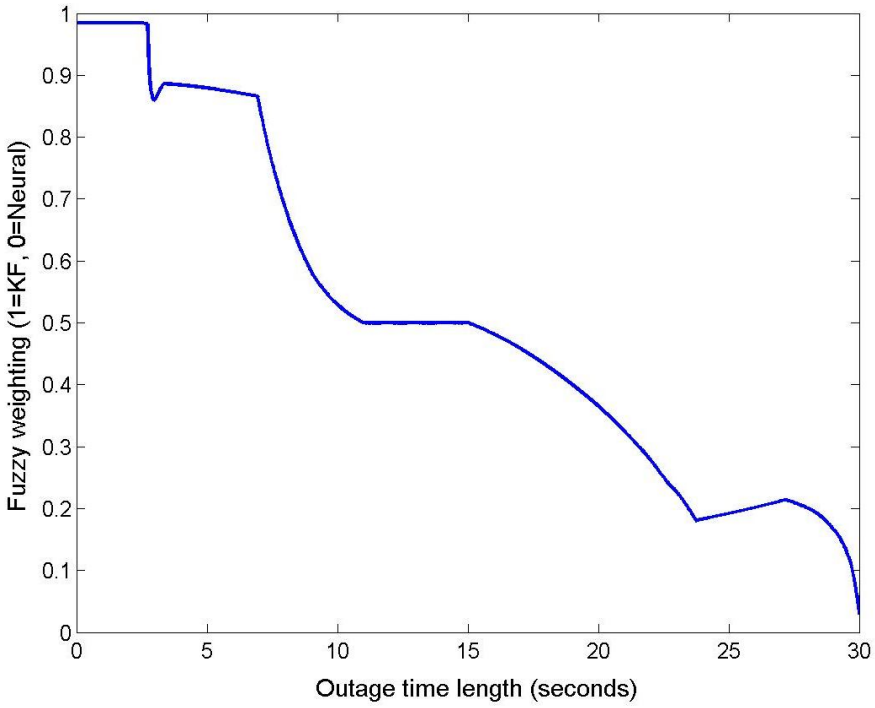

Figure 5.3: User developed weighting for outage time

These membership functions were used as initial values for the ANFIS tuning. Nine test outages were used from the training dataset to generate target values for adjustment. The result of adapting the membership functions using ANFIS is shown in Figure 5.4. The input-output relationship is more evenly spread across the input range, with a distinct linear trend towards the middle and less slope towards the extremes. This figure was trained on outages that always showed improvement using neural compensations; this is the reason the plot comes very close to zero after 30 seconds.

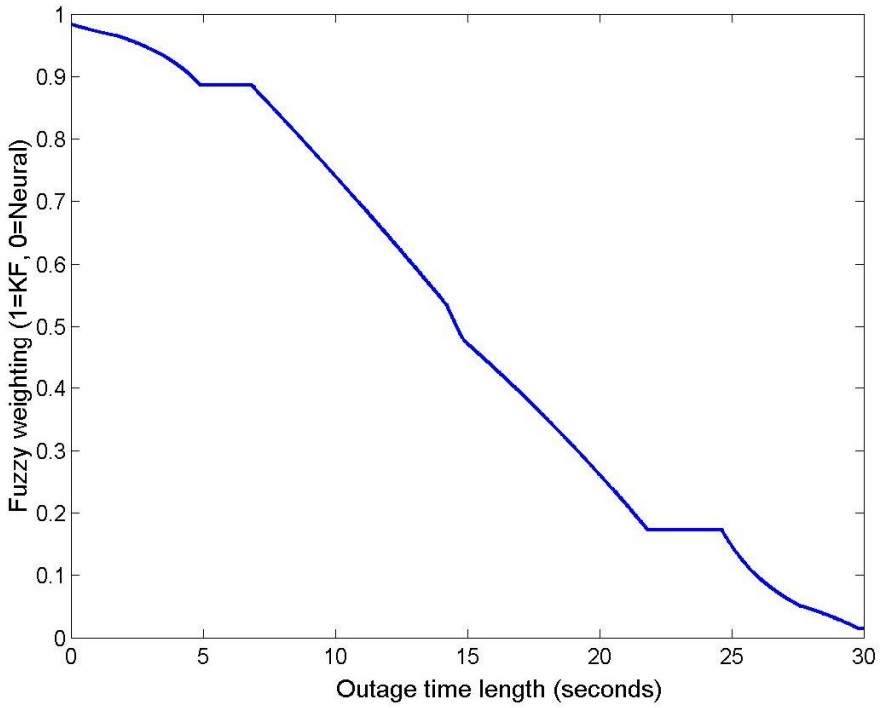

Figure 5.4: ANFIS developed weighting for outage time

5.2 Applying the fuzzy reliability monitor

The fuzzy weighting of the EKF and neural predictions ensures a reliable solution before the neural networks have converged and should also provide superior performance after the networks have converged. To analyze these situations, two different scenarios were constructed: a poorly trained case and a well trained case using results from Chapter 4.

5.2.1 A poorly trained case

In this scenario, neither east nor north networks were converged. In a real situation this could be due to lack of training data or time restrictions limiting the convergence of the

networks. As a result, the neural predictions are expected to be very poor and should be weighted low. The EKF predictions would be assumed optimal in this case.

The training results of the two networks that have not yet converged are shown in Figure 5.5. The training process typically involves several epochs before the networks converge, which is determined by the mean squared training error. The result in Figure 5.5 occurs if the training is stopped prematurely. The accuracy of the training data can be quickly found by differencing these two signals and calculating the RMS. If the RMS is found to be much larger than the accuracy of the known target signal RMS, then it can be deemed that training has not converged.

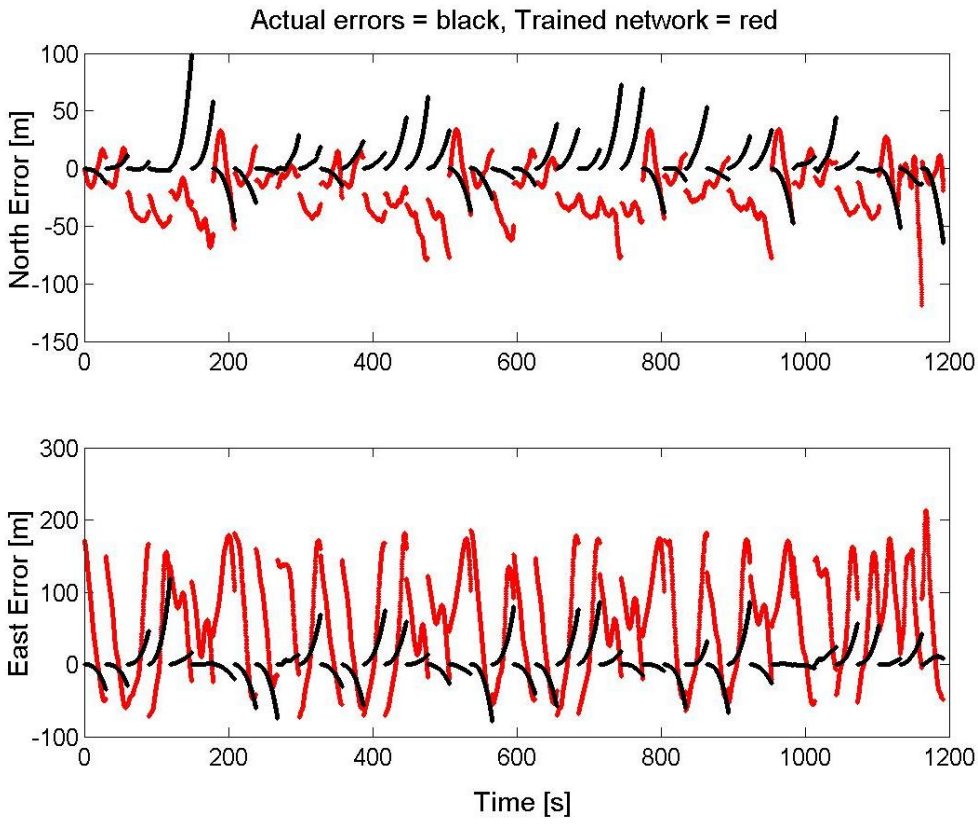

Figure 5.5: Neural training data that has yet to converge

The neural prediction errors using this training data are shown in Figure 5.6 in green, along with the EKF prediction errors in blue. Clearly, the neural compensated residual errors are large because of the poor training data.

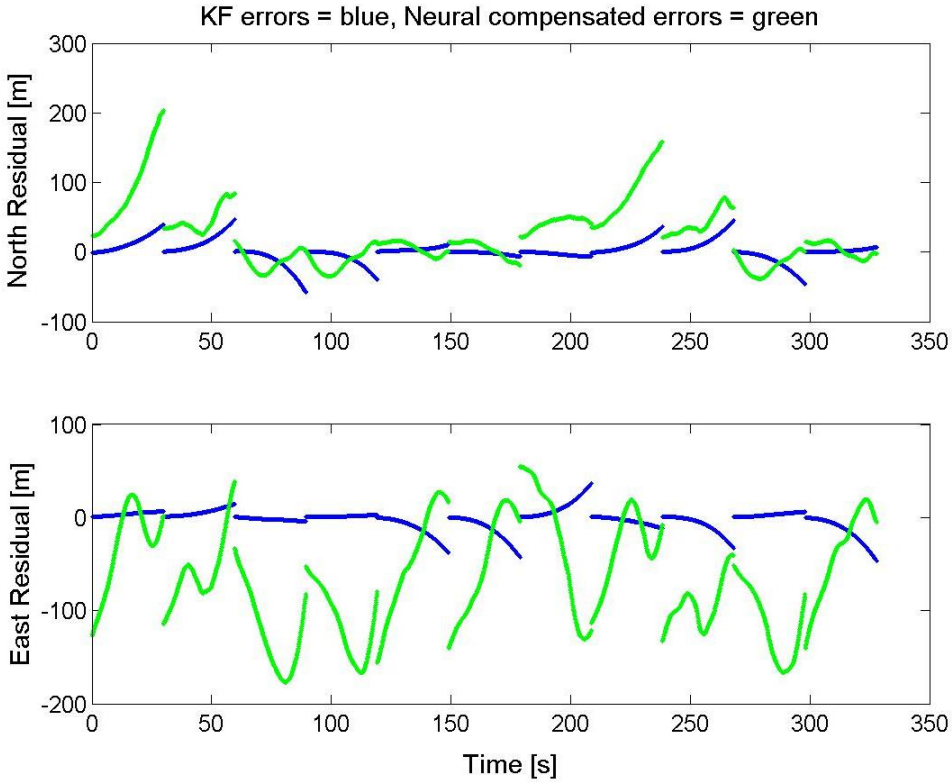

Figure 5.6: Prediction results using neural compensations that have not converged

The fuzzy weighted solution in Figure 5.7 is very close to the optimal EKF solution since the neural errors are very large and are rightfully avoided by the fuzzy reliability weighting. In this case the best solution is the EKF solution in blue which is well estimated by the fuzzy output in purple. The overall error difference between the fuzzy and EKF networks is less than 2%.

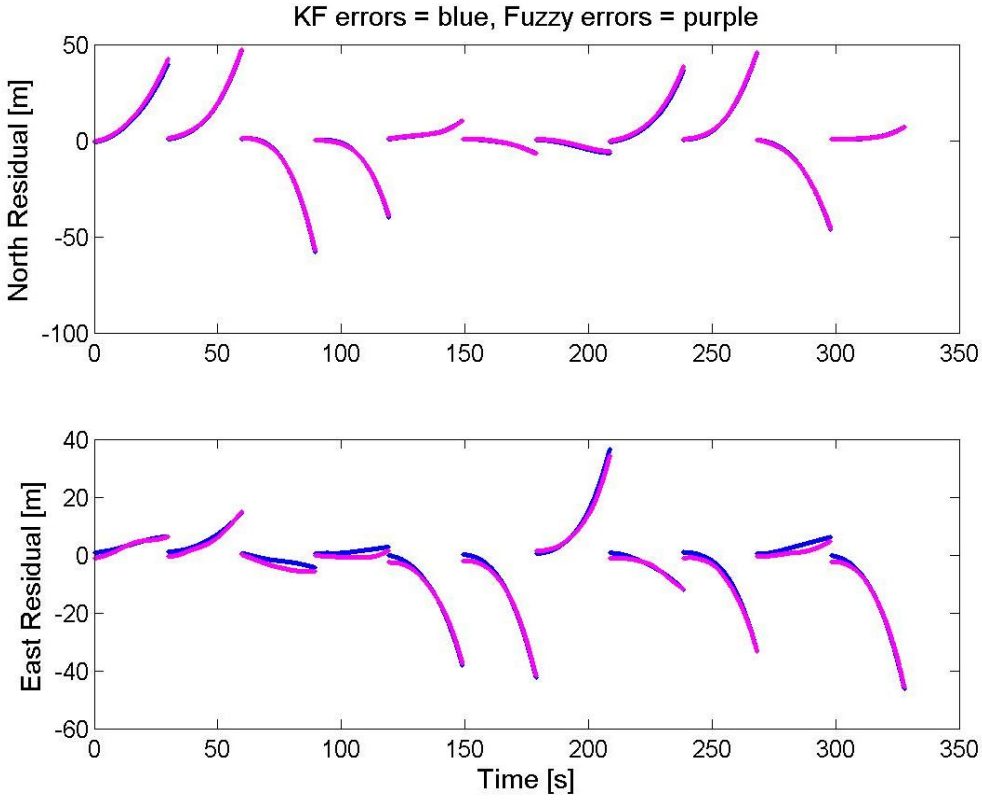

Figure 5.7: Solution of poor convergence case using fuzzy reliability monitoring

5.2.2 A well trained case

The next scenario assumes that the neural system has been well trained and the prediction inputs are within the range of the training inputs. The largest effect on weighting between the EKF and neural predictions is now based on the length of the outage. If the outage time is short then the EKF is given more weight, but if the outage continues then the neural compensations are applied with greater confidence. Figure 5.4 gives a general idea of the weighting for this case.

To compare the fuzzy results to the neural improvements previously attained, the same outages are once again considered using the well trained networks of Chapter 4. The only new information are the fuzzy errors shown in purple in Figure 5.8 along with the same EKF errors in blue and neural errors in green. It can be seen that the fuzzy errors are weighted between the EKF and neural errors depending on the length of each outage. This translates into good estimates throughout the length of the outage regardless of the noise level of the neural predictions or drift of the EKF errors. Towards the beginning of the outages the fuzzy errors are smooth and close to zero since they follow the EKF errors. As the outage time continues the fuzzy error weighting shifts from the EKF errors to the neural errors. This makes the fuzzy estimates noisier but also more accurate, avoiding the large EKF drifts.

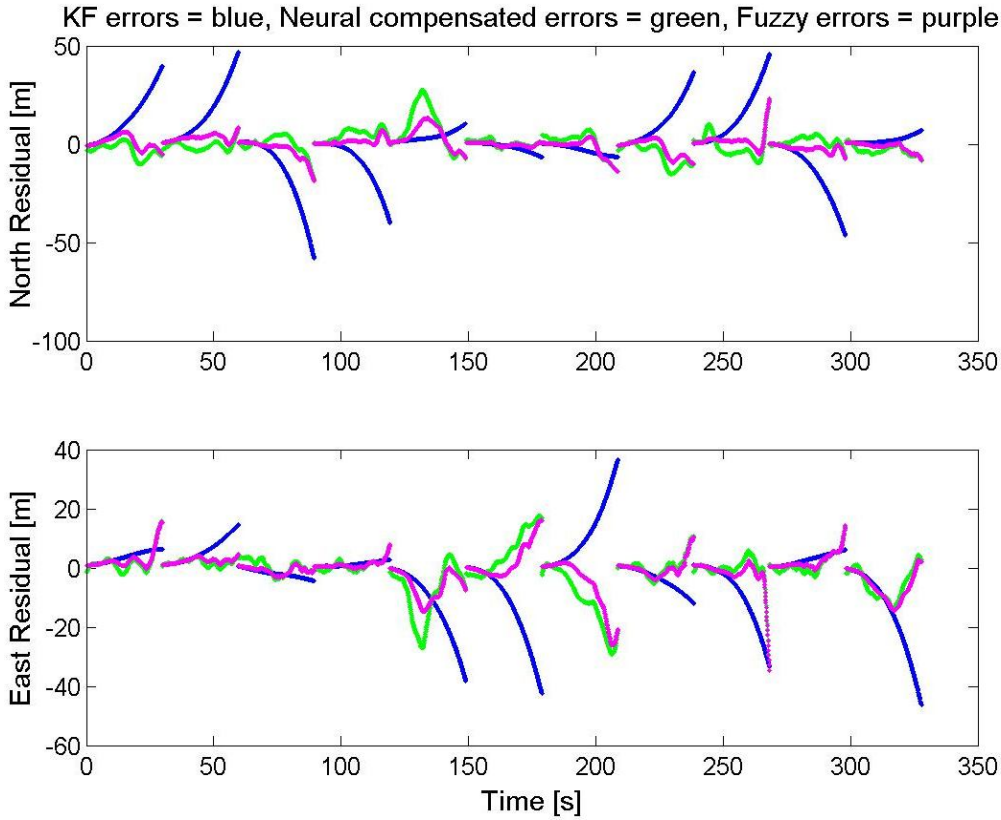

Figure 5.8: FIS results for a well trained network

The individual and overall improvements when using the fuzzy weightings are given in Table 5.1. Positive values represent percentage improvements while negative values indicate degradations. The overall improvements when using the fuzzy weighting compared to the EKF predictions were 70 and 52 percent in the north and east directions respectively. These are significant gains over the neural improvements shown previously in Table 4.5 of Chapter 4. To validate these additional gains using the fuzzy weighting, the individual improvements of the fuzzy weighting over the neural compensations are also given in Table 5.1. The additional improvements over the neural compensated EKF

predictions are 35 and 21 percent in the north and east directions respectively when using the fuzzy weighting between the EKF and neural predictions. This is largely due to the ANFIS weighting of outage times.

Table 5.1: Fuzzy improvements over neural and EKF

Outage	North % over KF	North % over neural	East % over KF	East % over neural
1	81	26	21	-3
2	79	-64	57	-32
3	78	3	43	47
4	85	68	7	13
5	-46	45	53	35
6	59	44	67	39
7	-5	40	38	36
8	65	35	50	16
9	80	47	70	-2
10	84	-5	21	3
11	25	53	57	-2
Overall	70	35	52	21

CHAPTER 6: INTELLIGENT TUNING ALGORITHM

The neural networks and fuzzy logic presented in Chapters 4 and 5 give practical methods to aid a general filter, but they do not provide insight about the actual filtering process. It is the intent of this chapter to use intelligent learning methods that feed back to the Kalman filter *a priori* inputs; in doing so the designer of the system can gain knowledge of the effects dominating the position state errors. Instead of compensating the outputs, this intelligent feedback system changes the inputs to the filter. This requires that the system have self-learning capabilities, which the RL system does through interaction with the environment.

6.1 Tuning a Kalman filter

The process of tuning a Kalman filter refers to the steps taken by the designer while changing the *a priori* data in an effort to improve the performance of the filter. The actual performance measures defined by individual designers can vary. Most INS/GPS designers would agree that the positional error drifts during periods without GPS updates are the most important measure. For a consumer product position is of utmost importance, so this is the primary measure.

These drifts relate directly back to the performance of the innovation sequence, which if white, is considered optimal (Mehra, 1970). If the last innovation sequence happens to be large then large prediction drifts would be expected. If the innovation sequence is small

then the opposite would occur. So the size of the drifts is governed by the size of the innovation sequence, which in turn if optimal, is driven by white noise.

The whiteness of the innovation sequence can be analyzed by looking at many time epochs when measurements are available. Another way of indirectly analyzing the innovation performance would be to look at the maximum values of the state drifts, since these depend largely on the last innovation sequence before prediction. These drifts also depend on the quality of the dynamic modelling, so if the average of an infinite number of these drifts was minimized then the filter would be performing optimal state estimation, in terms of accuracy.

Reliability of a Kalman filter is also very important. A Kalman filter can be considered reliable if its estimation of the position error drifts is close to the actual error drifts. This is commonly referred to as consistency of the filter (Crassidis and Junkins, 2004). In a real scenario, where GPS updates are lost it can be very important to reliably estimate the position error drift. An example of this would be map matching. If two parallel roads are 100 metres apart, and the error drift is only 20 metres then the display can lock the user to the correct road without displaying the 20 metres of error. If however, the Kalman filter incorrectly predicted this error to be over 50 metres then map matching could not be applied, or would be incorrectly applied.

In any estimation process accuracy and reliability should be of importance. If a system is accurate there should always be some quantity to maintain this accuracy. For this reason

the position drifts and Kalman predictions of these drifts were weighted equally in the development of a tuning performance measure. Equation (110) gives the minimization function, hereby referred to as 'minFunc',

$$\min\left(D+\frac{(D-P)^2}{D+P}\right) \tag{115}$$

D represents the average value of many state drifts, while P is the average of the Kalman filter predictions of these drifts. The drifts are obtained by simulating GPS outages when the GPS signal is available and allowing the inertial-only navigator to accumulate errors. The minFunc equation satisfies several important requirements that ensure a quality tuning result:

Case 1: a consistent filter, D-P = 0. In this case the optimization problem becomes minimize D.

Case 2: perfect state estimation, D = 0. For perfect state estimation the filter should predict no drift, so the optimization problem now becomes min P.

Case 3: the Kalman filter believes there should be no drifts, P = 0. In this case the optimization problem tries to match the first assumption by minimizing 2D, which is still proportional to minimizing D.

From a statistical viewpoint, an infinite number of GPS outages would have to be simulated to ensure that the solution is optimal. The central limit theorem states that the sum of an

infinitesimal number of random phenomenon approaches a Gaussian distribution, regardless of the individual distributions (Maybeck, 1982). Since it is not possible to collect and analyze the effects of *a priori* parameters over an infinite number of simulated outages, error bounds can be developed around certain solutions to provide insight about the level of fine-tuning that can be performed for a given set of data. Any method, even intelligent learning methods, will have limits to how well they can decrease these error limits. If a method could hold all past data in memory then the bounds would always decrease to the level of system noise. Even incremental intelligent methods can forget previous data and the requirement to hold all past data in memory is not possible. Therefore error bounds are applied to understand the limits of the obtainable accuracy.

6.1.1 Serial tuning procedure

It is worth discussing a method commonly used by designers to tune a Kalman filter in a manual way. This can be better described as a serial tune as will become apparent once the method is explained. The designer of an MEMS IMU typically starts from the manufacturer recommended values or some values that intuitively make sense. Unlike the Kalman filter first state, these starting values make a difference to the convergence of the final state when performing a serial tune, so it is important they are chosen carefully.

The second step is to collect some field test data or simulate test data. This data is then used to simulate GPS outages and analyze the performance measures defined by the designer; in this dissertation the optimization problem will remain the same, to minimize the minFunc equation. The collected data is crucial to the success of any tuning method.

The more data used the better, since the error bounds can then be approximated by the variance of the data.

The next step is to form a serial tuning order. This is typically done by performing a sensitivity analysis of the various parameters during GPS outages with respect to the time without measurements. The following equation, as a function of time, relates many of the inertial tuning parameters to position error in the horizontal plane.

$$\delta p(t) = \delta p(t_0) + \delta v(t_0)\Delta t + \delta b_a(t_0)\frac{\Delta t^2}{2} + \delta b_g(t_0)g\frac{\Delta t^3}{6} + \delta\theta_{r,p}(t_0)g\frac{\Delta t^2}{2} + \ldots$$
$$\ldots + \delta\theta_A(t_0)V\Delta t + \delta_{SFa}(t_0)\frac{\Delta t^2}{2} + \delta_{SFg}(t_0)g\frac{\Delta t^3}{6} \quad \textbf{(116)}$$

The terms in this equation are defined as,

$\delta p(t)$	the positional error drift after time t
$\delta p(t_0)$	the initial position error at the start of the outage
$\delta v(t_0)$	the initial velocity error at the start of the outage
Δt	the time difference between the start of the outage and the current time
$\delta b_a(t_0)$	the accelerometer offset bias at the beginning of the outage
$\delta b_g(t_0)$	the gyroscope offset bias at the beginning of the outage
g	the local gravity constant
$\delta\theta_{r,p}(t_0)$	the nonorthogonality error due to roll or pitch errors at the beginning of the outage

$\delta\theta_A(t_0)$	the nonorthogonality error due to azimuth errors at the beginning of the outage
V	the average velocity during the outage
$\delta_{SFa}(t_0)$	the accelerometer scale factor error at the beginning of the outage in specific force [m/s^2]
$\delta_{SFg}(t_0)$	the gyroscope scale factor error at the beginning of the outage [rad/s]

From this equation it can be seen that the residual gyroscope errors create the largest positional drifts with respect to time. Typically the gyro bias would be adjusted first, with parameters such as the gyroscope scale factor, accelerometer bias, angular random walk and velocity random walk coming after.

What is not shown in equation (111) is the effect of the GPS measurement updates. In a loosely coupled filter these come in the form of GPS position update noise estimation scale factors. The GPS Kalman filter provides estimates of these noise factors within the R matrix, but these can be scaled up or down depending on whether the GPS Kalman filter is pessimistic or optimistic. If the GPS Kalman filter was in a state of perfect consistency, the scale would be unity.

This R matrix scaling is very important when tuning a loosely coupled filter. Designers often put this parameter first or second in the tuning strategy. It should be considered at least as important as the gyro bias noise estimate since it affects the last innovation

sequence directly. The serial tuning results in this thesis always consider the R noise scaling factor first.

There can of course be other parameters such as misalignment and temperature noises, but these are often added into the gyroscope and accelerometer bias estimates since their exact values are difficult to obtain. This is a primary reason why the noise parameters of the biases are often optimistic when compared to real situations.

Once a priority order has been defined the tuning process can begin. The most sensitive parameter is allowed to vary while the remaining parameters stay fixed at their starting values. Once the best local value is found for this first tuning parameter, the value is fixed for the remainder of the tuning. The second value is then allowed to vary while all others stay fixed; the first parameter is fixed to its best local value and the others to their starting values. This process continues until all the tuning parameters have been allowed to vary.

6.1.2 Issues with the serial tuning method

Although the serial tuning method is simple and easy to implement in practice, there are many problems with tuning a Kalman filter using this approach. Optimality is compromised for the following reasons:

1. **Order of tuning**

   The sensitivity analysis and resulting order of serial tuning does not maintain optimality. Only one parameter is allowed to vary at a time and once it is fixed it is not adjusted again.

2. **Starting point**

   The tuning of initial parameters relies heavily on the values of the other parameters. If these parameters are chosen poorly then the entire tuning procedure can be compromised.

3. **Discrete tuning steps**

   Discrete steps are taken when tuning a parameter. Typically the value is scaled up or down by a constant multiple. This multiple will depend on how fine a tune is being performed. For MEMS this tuning approach would have to be performed several times with decreasing multipliers to enable a detailed tune**.**

4. **Range of tuning steps**

   A range for tuning has to be defined. The limits for MEMS sensors can be quite broad which would result in multiple tuning attempts using different discrete multipliers.

5. **Range of data**

   The data used to perform a serial tune typically comes from a single field test. This data is not guaranteed to apply to future tests, since the optimal values rely on testing across an infinite number of simulated outages. It cannot be expected to keep all of the data in memory, and the serial method has no way of accumulating memory without exceeding space requirements. Without a significant amount of data, the serial method cannot be expected to be very accurate. In contrast, an intelligent method learns sequentially and decreases the variance on the estimates as more data is accumulated in memory.

6.1.3 Bounding the tuning error

Tuning will never be perfect due to the fact that an infinite number of outages have to be tested. According to the central limit theorem, a Gaussian can be approximated, with a mean and variance, given that many outages are used. The variance on this data can also be used to constrain changes to the filter parameters if insignificant improvement or degradations are found. This analysis applies to any type of method that tunes the *a priori* parameters using position drifts during outages.

6.2 Learning through reinforcement

The *a priori* information should be adjusted online without input from either the user or the designer. This cannot be performed with an ANN since there is no target signal to use as reference. All that is available are changes in the Kalman filter state. When starting to tune it is unknown whether changes to certain parameters would be good or bad to the output of the Kalman filter. All that can be gained are relative changes to previously known input/output scenarios.

The desired method of tuning fits nicely into a reinforcement learning strategy. This type of tuning adjusts the parameters up or down depending on comparisons with previous tuning results. This allows multiple *a priori* tuning values to be updated during a single tuning step, making the method a parallel tuning approach.

The RL converges its solution around the best tuning procedure. Positive reinforcements are awarded for tuning steps that result in improved performance in terms of both accuracy and consistency. The RL policy actually learns how to tune the filter from any tuning state encountered through trial and error. This is important since the optimal state will result by forced improvement from whatever state the system is in. This also gives the system memory of the tuning procedure, which can be extended to other similar sensors, or updated sequentially as aging or environmental effects contaminate the optimal solution. In a case where the *a priori* parameters are static, learning can be used to accumulate data without having to store additional space in memory.

6.2.1 Leveraging the learning process

The usefulness of this type of adaptive learning can be used to extend the tuning results from one filter to another. Low-cost MEMS coming off the same production line can have many similarities, but can also be quite different. It is hard to know how the hardware differs until the sensors are actually used in an integrated system.

The idea of extending tuning results from one sensor to another is referred to as leveraging. The assumption is that an individual sensor will have similar error characteristics as another sensor of the same type, and that the similarities will be more influential than the differences. If this assumption is maintained, the learning will converge faster and more reliably than starting without knowledge. RL has the ability to begin with an initial model of the optimal policy. This policy can then be adapted as before, through a combination of trial and error and greedy exploitation.

The RL tuning strategy would typically start from the manufacturer's default values, and then take steps from this starting point towards a better solution. Other sensors of the same type would also start from the manufacturer's default values, so the models from previously tuned sensors could be applied as additional *a priori* information.

6.3 Simulation results

Simulation results are very important in analyzing the validity of the two tuning methods. Simulations allow the creation of defined stochastic processes without interference from additional unmodelled errors or contamination from incorrect error modelling. Both GPS and inertial sensor data were simulated. The GPS data was contaminated by time correlated and white noise errors, while the inertial error models are given in Table 6.1.

Table 6.1: Inertial error models

Error term	Model	Units
Gyroscope bias instability	1st order Gauss-Markov	deg/hr
Accelerometer bias instability	1st order Gauss-Markov	mGal
ARW	White noise	deg/√hr
VRW	White noise	m/s/√hr

Given these user defined parameters the Kalman filter can then be run with confidence of the true solution to the stochastic model. This allows a direct comparison to the true solution, whereas using real data the true solution can never be known.

The simulation results created the simulated inertial and GPS data through an inverse mechanization process using direction cosine matrices. See Yang (2008) for more details on the inertial simulator. The simulated signals were then passed through a loosely coupled extended Kalman filter to produce state estimates of the positions, velocities, attitudes, biases and scale factors. See Shin (2005) for full details on the EKF, which used quaternions to parameterize the attitudes. The actual error values used in the simulations are given in Table 6.2.

Table 6.2: Simulated error values

Error term	Value
Gyroscope instability (std)	100 deg/hr
Accelerometer instability (std)	600 mGal
ARW	1 deg/√hr
VRW	3 m/s/√hr

The random walk terms have no correlation times, but the bias standard deviations are modelled as 1st order Gauss-Markov so they require definition of correlation times for their autocorrelation functions,

$$R(t) = \sigma^2 e^{-\beta|\tau|} \tag{117}$$

The correlation time is,

$$t = \frac{1}{\beta} \tag{118}$$

In control engineering the variance term is considered the initial value since,

$$R(t) = \sigma^2 e^{-|\tau|/t} = \sigma^2 e^0 = \sigma^2 \tag{119}$$

The time constant is used to describe how quickly the exponential decays. When t equals τ the value decays by 37% of the original value,

$$R(t) = \sigma^2 e^{-|\tau|/t} = \sigma^2 e^{-1} = 0.37\sigma^2 \quad \textbf{(120)}$$

For a time constant equal to 5 times that of the correlation time the system has stabilized to less than 1% of the initial value (Dorf, 2000),

$$R(t) = \sigma^2 e^{-|\tau|/t} = \sigma^2 e^{-5} = 0.007\sigma^2 \quad \textbf{(121)}$$

This is typically defined as the stable point of the system. For a correlation time of 1 hour the stability would only really be guaranteed after 5 hours. Since this is not very practical, another way to minimize the noise is to reduce the initial value of the variance. This is what the tuning strategy does by keeping the correlation time fixed and adjusting the variance value.

If the time constant is fixed to a value that is too short, then the variance can be increased to allow the bias estimation to adapt more. If the time constant is fixed to a value that is too long the reverse can be performed by the tuning. All gyroscope and accelerometer time constants were set at one hour for all MEMS and simulated MEMS sensors.

The use of the loosely coupled EKF also allows a scaling factor to be applied on the R matrix estimations of the GPS position errors. Since the GPS data was simulated as white noise with a defined variance, the true value of this scaling term would be unity.

32 simulated outages were created with 90 seconds between outages for the filter to re-converge. Error bounds were created using the variance of the minFunc solutions. More

simulated outages meant that the error bounds decreased. Using the optimal stochastic solution, 30 simulated outages were used to generate error bounds of 27 metres, relative to a minFunc value of 102 metres, or approximately 26% of the minFunc value. When only 15 outages were used, the error bounds increased to 72 metres for a minFunc solution of 128 metres, or 56% of the minFunc value. By doubling the number of simulated outages the bounds on the optimal solution were halved. This clearly indicates the necessity of using as many simulated outages as possible to minimize the relative error on the generated solution, and the impracticality of using a fixed window approach without memory.

6.3.1 Serial tuning

The serial tuning was applied to the simulated data first to form a benchmark for comparisons of the RL tuning. Two serial tuning scenarios were performed:

1. The first case started from the optimal solution and used error bounds to constrain changes in the tuning solution. A new minFunc value could only be deemed better than the previous solution if it satisfied two conditions:
    a. The new minFunc value was smaller than the previous best minFunc minus the lower bound error.
    b. The new minFunc value plus the new upper error bound was smaller than the previous best minFunc value.
2. The second serial tuning approach did not apply the constraints of the error bounds and started from a random solution.

The first case was simulated to test the validity of the true stochastic solution combined with error bounds. The second serial tune was more realistic of a typical situation where the real solution is not known.

The order of serial tuning was kept the same for both scenarios. The tuning began with the R position standard deviation scale factor, the gyro bias standard deviation was tuned second, then the accelerometer bias standard deviation, the ARW and finally the VRW (Nassar et al., 2006).

The tuning plots for the first serial scenario are given in order of the serial tuning. The horizontal axes represent the discrete tuning values that were taken for each parameter. The vertical axes are the returned minFunc values that weighted the average of the 30 simulated position drifts with the consistency of the filter. The star denotes the starting point, which in the first scenario was the optimal solution. Each cross is a constant scale from the previous cross value on the horizontal axis. The error bounds for the optimal solution are shown as the vertical lines above and below the optimal solution.

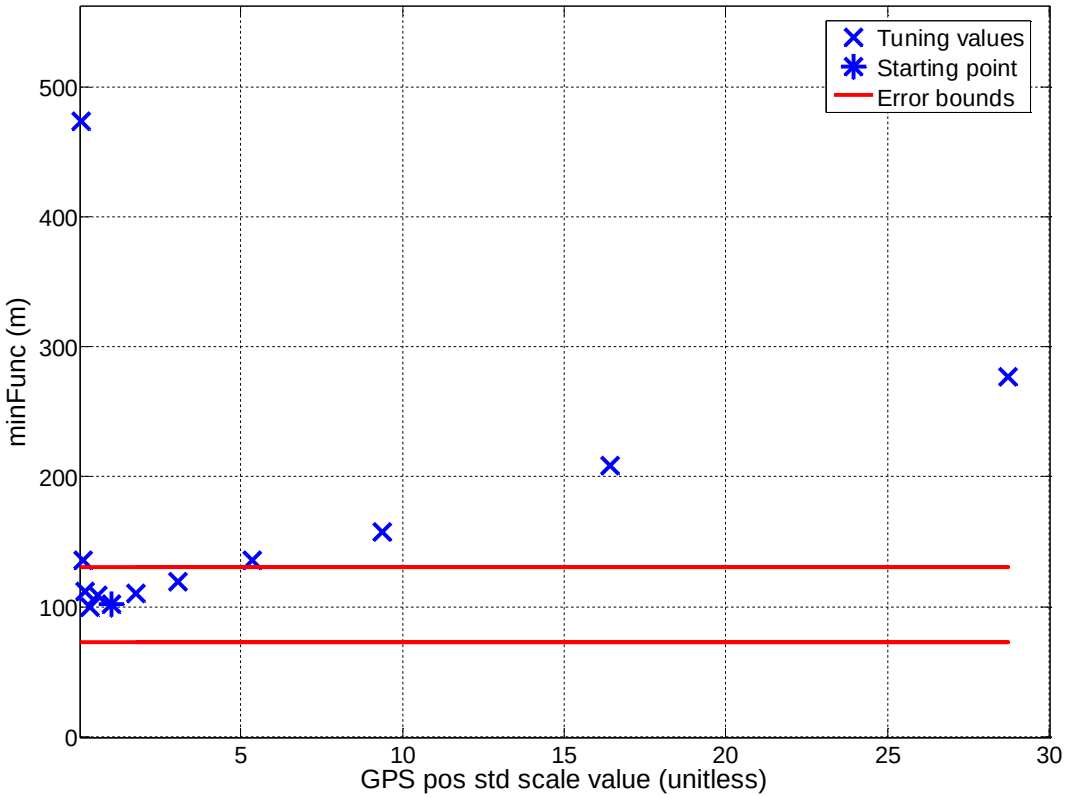

Figure 6.1: GPS position std scale results for serial tune of simulated data

For the GPS position standard deviation scale it can be noticed that smaller (optimistic) values diverge much more quickly than larger (pessimistic) values. This is due to the fact that the GPS noise variance was set to 5 metres, so when this value is scaled down the filter becomes too reliant on these values and contaminates the innovation sequence with additional noise on the level of several metres.

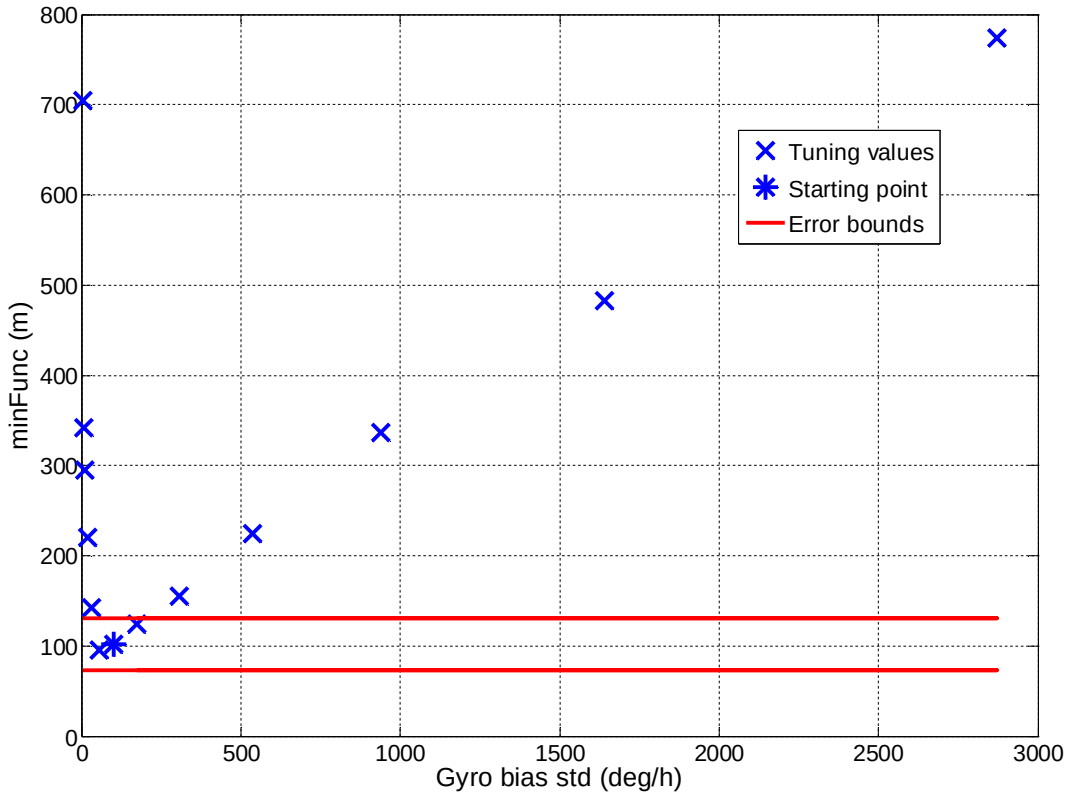

Figure 6.2: Gyroscope bias std results for serial tune of simulated data

The gyroscope bias is quite sensitive to changes in its standard deviation value. This was anticipated since a residual gyroscope bias has error proportional to time cubed when the filter operates without measurement updates.

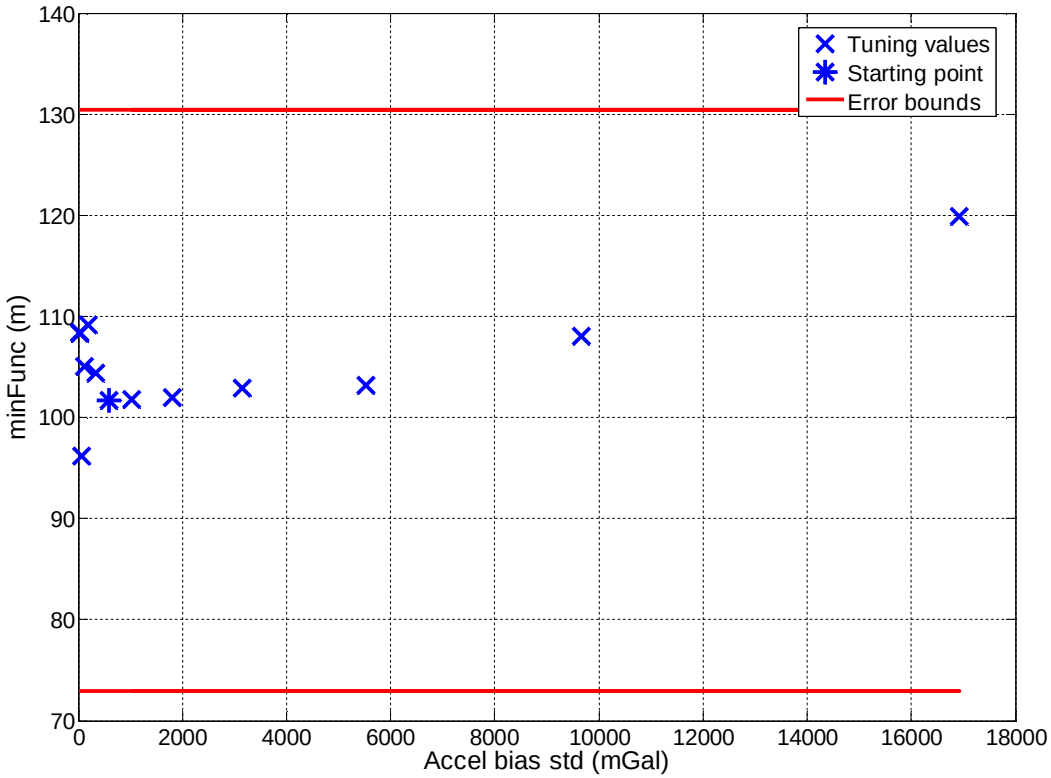

Figure 6.3: Accelerometer bias std results for serial tune of simulated data

The accelerometer bias standard deviation was not very sensitive to changes. None of the values surpassed the error bounds of the optimal solution, so the optimal solution of 600 mGal was maintained.

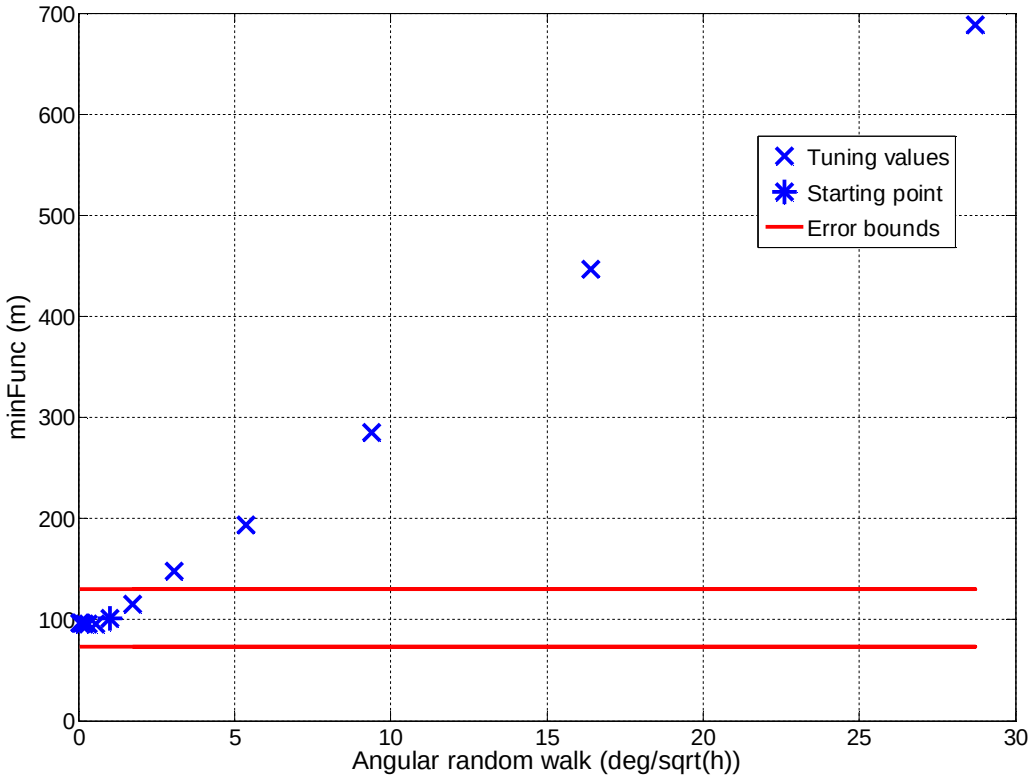

Figure 6.4: ARW results for serial tune of simulated data

The ARW values showed no improvement or degradation for small values, but some large degradations occurred for very large values. These large errors indicate that additional white noise errors would severely degrade the solution, while ignoring the ARW completely (i.e. very small values) does not degrade the solution significantly since the other stochastic bias errors dominate the solution during outages. The same explanation can be applied to the VRW values which show very similar characteristics, with slightly less sensitivity to the minFunc values.

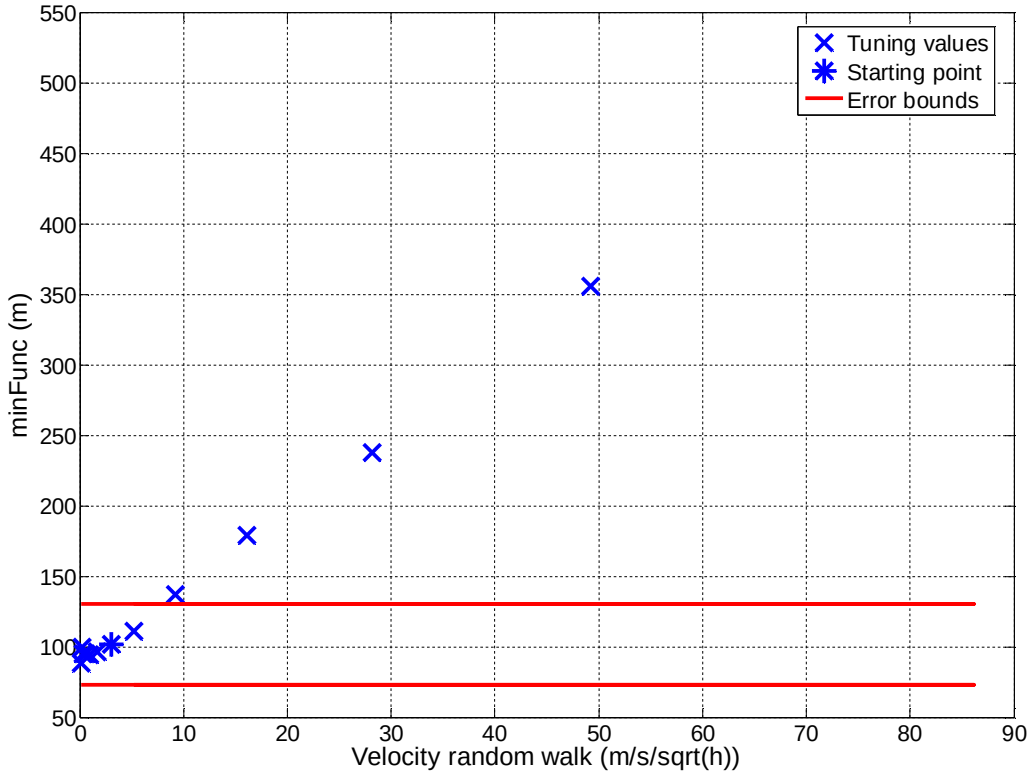

Figure 6.5: VRW results for serial tune of simulated data

The final solution using the serial tuning method, starting from the optimal solution and applying error bounds, converged to the true simulated solution. The average position error of the solution was 86 metres and P was slightly inconsistent at 111 metres. Unfortunately, this very contrived solution is not very practical. It really only proves that the simulator and loosely coupled EKF can be used together to produce an optimal solution within the errors created by using stochastic drift values as indicators of performance.

The second scenario is more representative of the performance that can be expected from a serial tuning method. Not all of the plots are reproduced, but those for the gyroscope bias

standard deviation and ARW are given. The starting solution was a random point within the defined ranges and the error bounds were not used to constrain the solution; the minimum minFunc values were chosen as optimal.

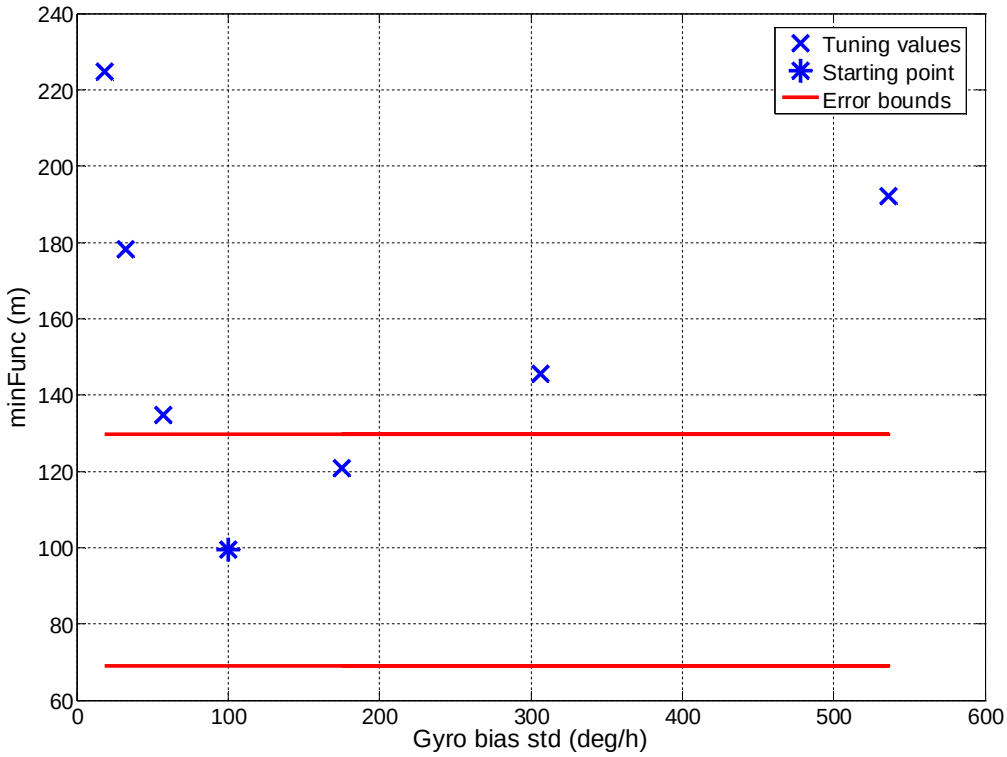

Figure 6.6: Serial tune, scenario 2, Gyro bias std

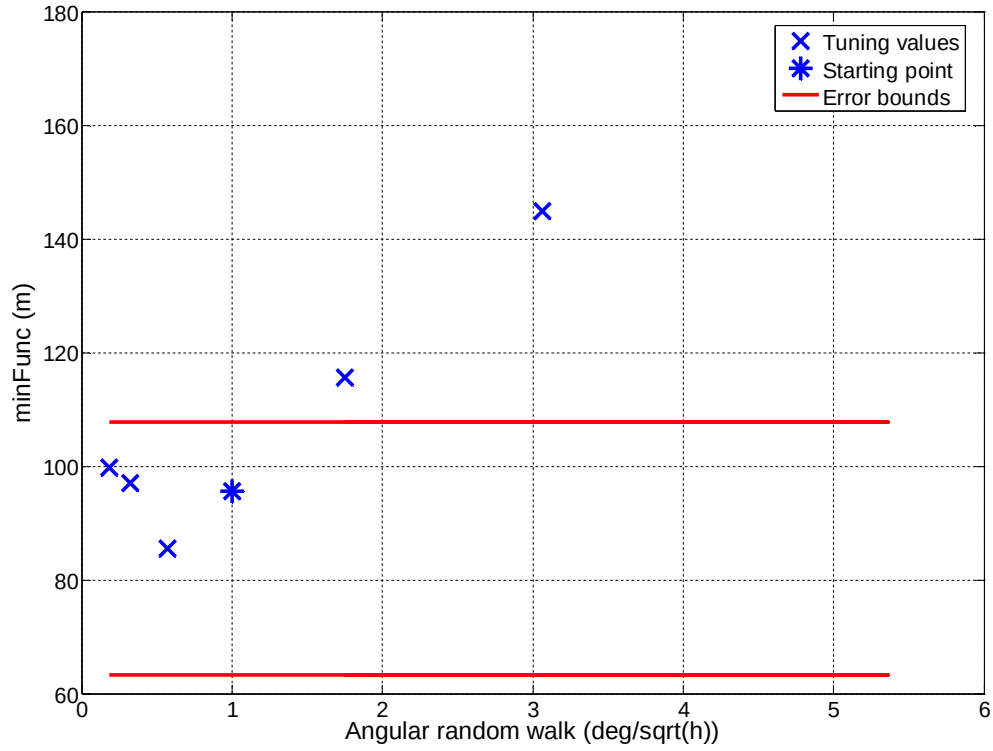

Figure 6.7: Serial tune, scenario 2, ARW

The serial tuning method came up with the values shown in Table 6.3. It can be seen that the values are no longer the optimal values. This result is from the serial tuning procedure combined with stochastic error that can occur when simply choosing the smallest minFunc values. From a practical viewpoint this is what is expected from the serial tuning method. With these stochastic parameters the average position drift was 69 metres and the filter was very consistent with a P estimation of 71 metres.

Table 6.3: Serial tuning, scenario 2 results

Error term (units)	Actual value	Serial tuned value	Absolute difference
GPS pos std scale (unitless)	1	0.32	0.68
Gyro bias std (deg/hr)	100	100	0
Accel bias std (mGal)	600	1030	430
ARW (deg/√hr)	1	0.57	0.43
VRW (m/s/√hr)	3	0.98	2.02

6.3.2 RL tuning

The RL tuning used the same performance criteria as the second example and also started from a random point within the same ranges. The primary differences between the two methods are that RL can tune the parameters in parallel and the state space is made continuous through RBF generalization.

Figure 6.8 shows the returned minFunc values over the first 35 iterations using parallel RL tuning. The horizontal axis represents the individual tuning steps taken. Each step was allowed to adjust the individual parameters up or down by discrete steps over the range of each parameter. This meant that any number of parameters could be changed at each step, allowing the method many more changes in 35 steps than available with 35 iterations of the serial method.

The tuning starts at a reasonable value for the parameters as indicated by the minFunc value just over 100 metres. The policy has not been formed at this time and learning occurs for

about the first 17 steps. In most reinforcement problems the solution gets worse before it gets better since the algorithm needs both good and bad examples of behaviour to converge upon positive reinforcements. The solution then slowly gets better until the end of the episode at 35 steps.

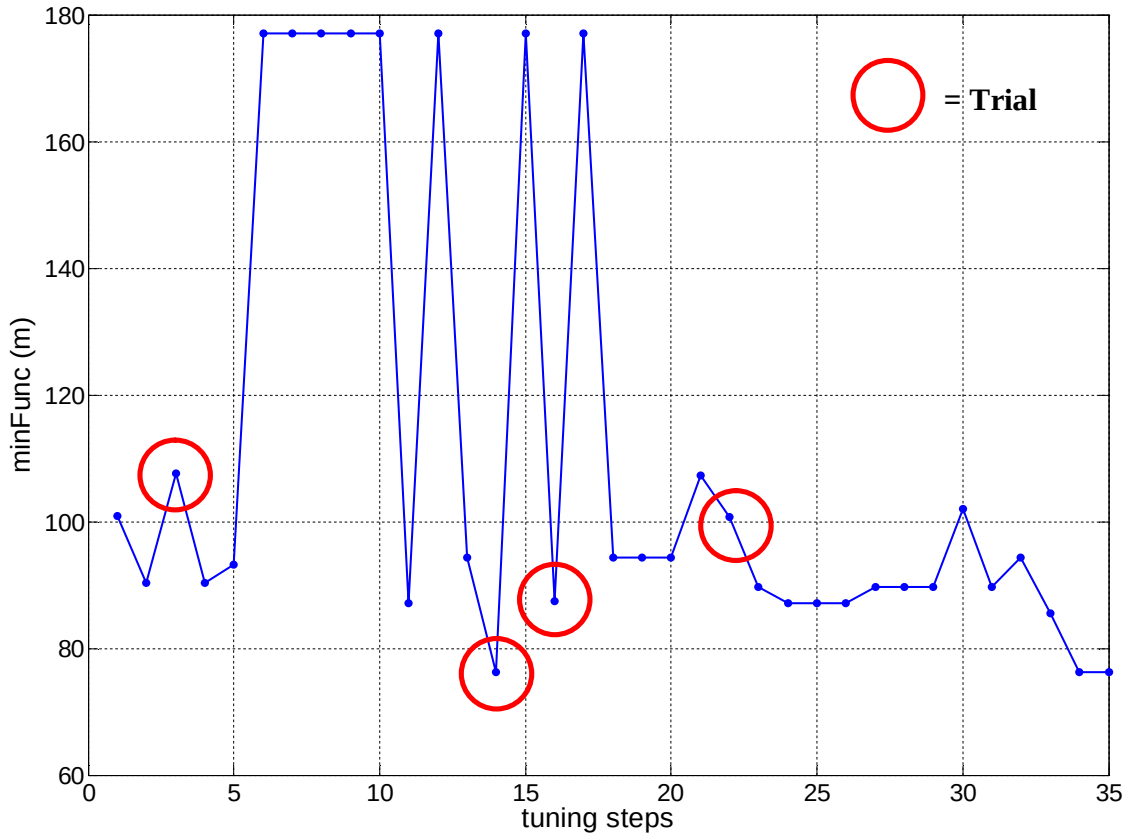

Figure 6.8: RL minFunc solution for simulated data

It is also important to analyze the trial steps during the learning process. The trials have been circled in Figure 6.8. Trials are important to help the algorithm explore new states with which it might otherwise not bother. This is evidenced by the trials at steps 14 and 16, which vastly improve the performance of the tuning. In fact, trial 14 is very close to the

best tuning state found in 35 steps; the algorithm just needed another 20 iterations to assure itself that this exploratory step was indeed positive reinforcement over the long-term.

Only a single episode was needed to solve this simplified problem, which converged to the solution shown in Table 6.4. The actual solution is closer than that of the serial tune. The actual values have not been found since the drifts are random and have a variance on them, which can only be improved with more data on which to learn.

Table 6.4: RL solution using simulated data

Error term (units)	Actual value	RL tuned value	Absolute difference
GPS pos std scale (unitless)	1	0.5	0.5
Gyro bias std (deg/hr)	100	100	0
Accel bias std (mGal)	600	400	200
ARW (deg/√hr)	1	1	0
VRW (m/s/√hr)	3	1	2

6.4 Real MEMS results on a loosely coupled extended Kalman filter

The same tuning methods were used with real MEMS sensors and SGPS data. MEMS gyroscopes and accelerometers from Analog Devices were used. The gyroscopes used were ADXRS150 and the accelerometers were ADXL105. The starting points for tuning were obtained from the datasheets, or if not available were extrapolated from experience of other similar MEMS sensors; both datasheets can be found in Appendices A and B. Ranges for the tuning parameters were also obtained from the datasheets, but were expanded to consider a broader range in case the datasheets were optimistic. A NovAtel

OEM4 receiver was used in single point GPS mode and was processed with GrafNav software for the loosely coupled EKF.

6.4.1 Serial tuning

The initial values and ranges for the tuning parameters are given in Table 6.5. Scale factor estimation has been added for the real data to make the tuning procedure complete for all tunable parameters of the loosely coupled EKF. The discrete steps were taken by multiplying or dividing the previous tuning value by the multiplier in the last column.

Table 6.5: Initial values and ranges for ADI

Error term	Initial value	Min value	Max value	# steps (+/-)	Multiplier
GPS pos std scale (unitless)	1	0.1	9.4	6	+/- 1.75
Gyro bias std (deg/hr)	2000	213	18757	6	1.75
Accel bias std (mGal)	10000	1066	93789	6	1.75
ARW (deg/√hr)	3	0.3	28.1	6	1.75
VRW (m/s/√hr)	0.21	0.04	3.75	6	1.75
Gyro SF std (ppm)	1000	107	9379	6	1.75
Accel SF std (ppm)	1000	107	9379	6	1.75

The best solution using error bound constraints is given in Table 6.6. These tuning parameters produced an average position drift error of 111 metres with an estimated P value of 215 metres. This solution is accurate but not very consistent. Actual figures of the tuning procedure are given in Appendix C.

Table 6.6: Serial tuning result for ADI

Error term	Initial value	Tuned value
GPS pos std scale (unitless)	1	0.19
Gyro bias std (deg/hr)	2000	213
Accel bias std (mGal)	10000	10000
ARW (deg/√hr)	3	3
VRW (m/s/√hr)	0.21	0.4
Gyro SF std (ppm)	1000	1000
Accel SF std (ppm)	1000	1000

6.4.2 RL tuning

The same initial starting point was chosen for the RL tuning from the datasheets, but instead of taking discrete steps across the state space, the space was made continuous. The ranges used for the space are given as maximum and minimum values in Table 6.7. The action space was discrete and discrete steps are given in the same table.

Table 6.7: RL initial start point and ranges

Error term	Initial value	Min value	Max value	Action (+/-)
GPS pos std scale (unitless)	1	0.5	2.5	0.5
Gyro bias std (deg/hr)	2000	500	2500	500
Accel bias std (mGal)	10000	4000	12000	2000
ARW (deg/√hr)	3	0.5	4.5	1
VRW (m/s/√hr)	0.21	0.01	0.81	0.2
Gyro SF std (ppm)	1000	500	4500	1000
Accel SF std (ppm)	1000	500	4500	1000

With seven tunable parameters and 5 discrete steps each the state space would have 78125 entries in the table. This is not practical, which is why generalization of the state space is so important. The state space is instead represented by 20 RBF features (as a maximum), which are adjusted each time an action is performed and a state is visited. The discrete choice of actions can also be changed over time to visit different states, but due to generalization this is not as important as in the discrete case.

Figure 6.9 shows the minFunc rewards over 200 steps and 10 episodes (20 steps/episode). To facilitate learning, the initial point of a new episode was set as the previously optimal solution. From this plot it can be noticed that the starting value from the datasheet was not very appropriate. Within five steps the RL algorithm quickly dropped the minFunc solution several hundred metres.

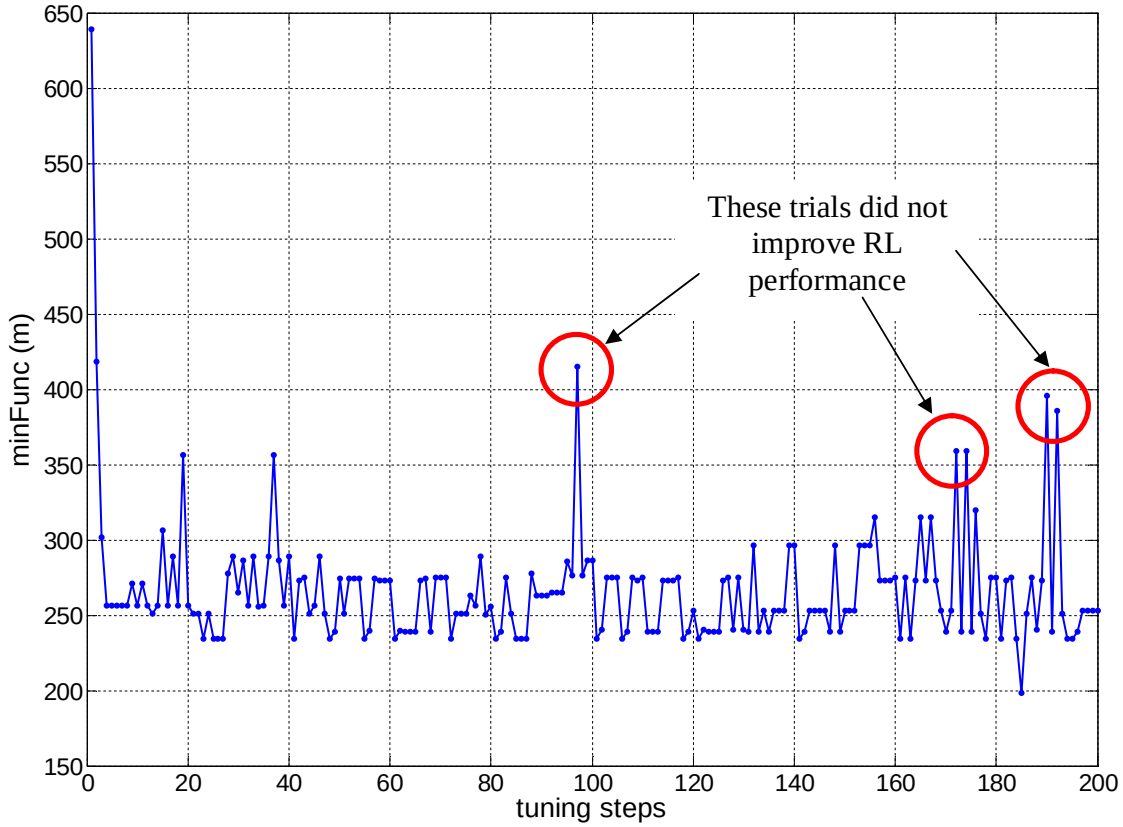

Figure 6.9: RL rewards over 200 steps for ADI

Trials occurred at steps 8, 17, 28, 71, 97, 118, 120, 172, 190, and 192. None of the trials helped the RL solution improve significantly. This is mainly due to chance and the fact that later trials occurred after the RL solution had converged. Trials 97, 172, 190 and 192 were all poor tuning strategies, but in return they were awarded negative reinforcements which led to large minFunc values as evidenced by the circles in Figure 6.9. These trials did not degrade the overall solution; in fact they were useful for the RL system to understand that it had indeed converged upon a satisfactory solution compared with previous examples.

The reason for the large drop in minFunc at the beginning of the tuning is due to the consistency of the filter. Figure 6.10 gives the same minFunc plot, but with the actual and predicted position errors as overlays. The actual position errors are quite good using the manufacturer start point, but the Kalman filter is not very consistent. The minFunc performance measure tries to minimize the difference between the two while also minimizing each individual signal. Emphasis is still given to position error minimization, but only if the filter is sufficiently consistent.

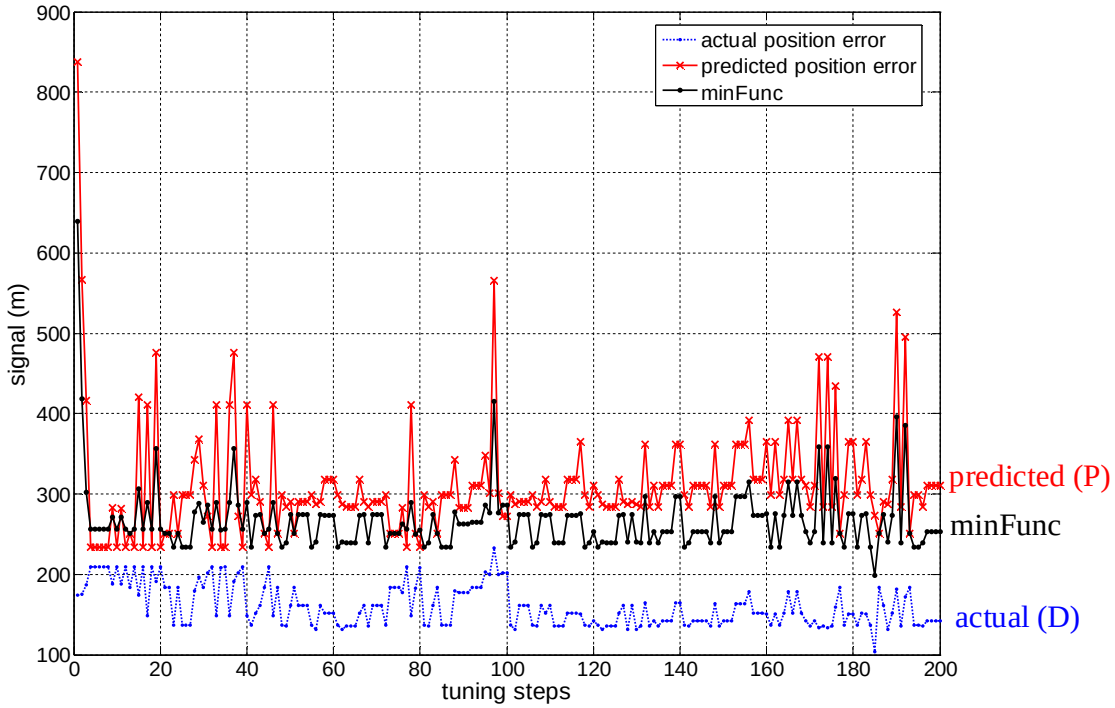

Figure 6.10: RL minFunc, position error, and predicted position error for ADI

During the first five tuning steps the filter is made more consistent, but less accurate. Then the accuracy is improved during the next fifteen steps, while consistency is sacrificed. Steps 20 to 40 try to re-balance the accuracy and consistency. It is only around step 100

(episode 5) that the RL solution begins to converge on what it believes to be the most accurate and consistent solution. Attempts to improve the accuracy are continued, while the consistency gets a little worse. It is the balancing of these two performance measures that creates slight oscillations in the RL minFunc solution.

To gain a better understanding of convergence, it is more useful to look at the average returns over each episode. Figure 6.11 shows that the RL solution begins to converge around the fourth or fifth episode. There will always be fluctuations after convergence due to the fact that the state space is still relatively unexplored and generalization is just providing an approximation to the state space. The poor trials also contribute to the average over each episode, and the effects of two poor trial steps are evidenced in the last episode.

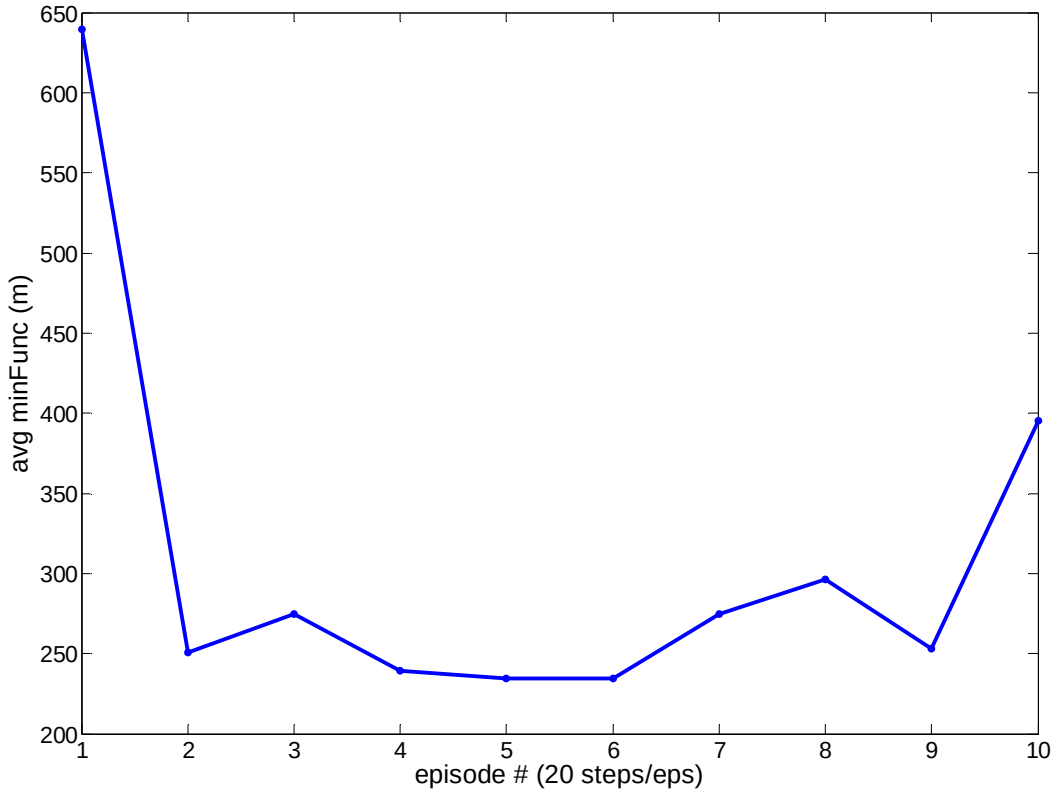

Figure 6.11: Average returns of minFunc over 10 episodes for ADI

The best optimal solution after 10 episodes is given in Table 6.8. A direct comparison cannot be made to the serial tuning result since the correct stochastic solution is no longer known. Only relative comparisons of the accuracy and consistency can be made between the two methods. The accuracy of the RL solution was 103 metres while the filter was pessimistic in predicting these errors at 270 metres. A more consistent solution could be obtained at the expense of accuracy. This is apparent from the beginning of Figure 6.10 when the accuracy and predicted accuracy are both near 210 metres. The minFunc performance measure significantly improves accuracy at the expense of consistency.

Table 6.8: RL tuning results for ADI

Error term	Initial value	Tuned value
GPS pos std scale (unitless)	1	0.54
Gyro bias std (deg/hr)	2000	498
Accel bias std (mGal)	10000	6202
ARW (deg/√hr)	3	2.56
VRW (m/s/√hr)	0.21	0.15
Gyro SF std (ppm)	1000	511
Accel SF std (ppm)	1000	1460

6.4.3 Leveraging to another unit

The best solution from the RL algorithm of the previous section was applied to a different triad of ADI inertial sensors. The tuning began from the optimal solution and the optimal policy found from the previous sensor triad. The point of leveraging is to show that even similar sensors coming off the same manufacturing line can have slightly different error characteristics, but have sufficiently enough similarities that the policy from one sensor can be applied to another sensor with some small adaptations.

Figure 6.12 shows the changes in minFunc and the position errors for 25 tuning steps. The initial solution is quite accurate which means that the solution from the previous sensor can be used as a good starting point. The minFunc value is 227 metres with an absolute position drift of 198 metres and a predicted drift of 240 metres.

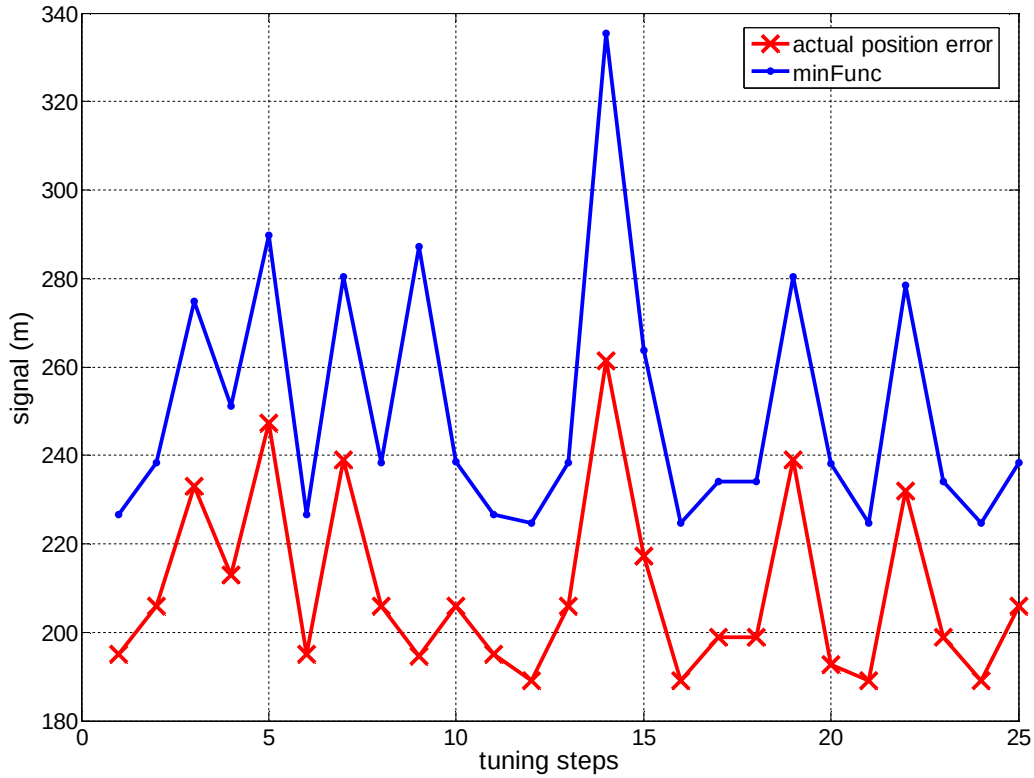

Figure 6.12: Leveraging tuning results to another ADI unit

The question now becomes: can the previous policy be adapted quickly to improve the overall solution? From the previous figure, it takes 12 tuning steps for the solution to adapt to a new minimum. The RL tuning then proceeds to try new tuning strategies, but the solution keeps returning to the same minimum value at steps 16, 21 and 24. This new minFunc value is 224 metres, while the actual position error is 186 metres and the predicted position error is 241 metres. The new solution is provided in Table 6.9. Small changes in all the parameters occurred with the largest occurring in the ARW and accelerometer scale factor standard deviation.

Table 6.9: Leveraged solution for new ADI triad

Error term	Initial value	Tuned value
GPS pos std scale (unitless)	0.54	0.5
Gyro bias std (deg/hr)	498	502
Accel bias std (mGal)	6202	5907
ARW (deg/√hr)	2.56	3.4
VRW (m/s/√hr)	0.15	0.15
Gyro SF std (ppm)	511	499
Accel SF std (ppm)	1460	2400

If this sensor were tuned without knowledge of the tuning policy and optimal solution from the previous sensor, the minFunc plot would look like that shown in Figure 6.13. It takes 33 tuning steps to get to the level of the leveraged solution. So by leveraging the solution has been found approximately three times faster. Furthermore, the leveraged solution varies less since it has memory of 225 tuning strategies, not just 33.

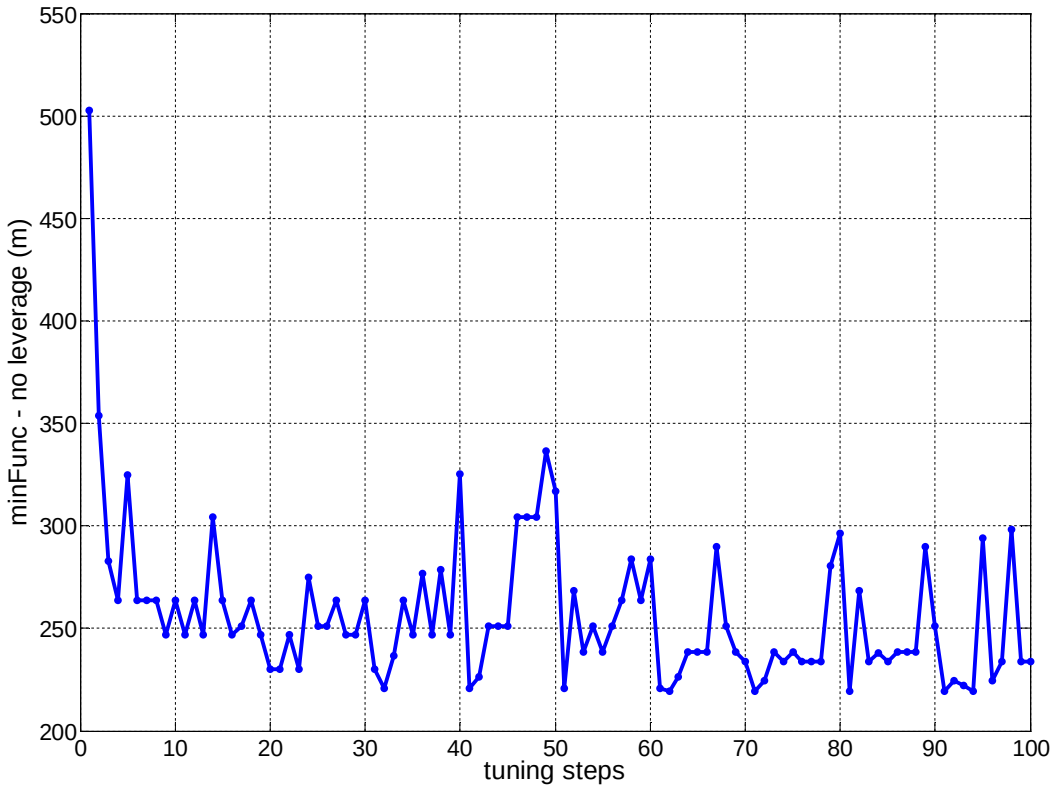

Figure 6.13: Second ADI tuning without leveraging

The solution without leverage is similar to that of using leverage, it has just taken longer to find. Table 6.10 gives the solution for the case without leveraging. The only parameters that changed significantly were the VRW and accelerometer scale factor standard deviation parameters. Due to their small significance during times without measurements, the large relative change in these parameters is not considered significant.

Table 6.10: Tuning results of second ADI without leveraging

Error term	Initial value	Tuned value	|Lev – No Lev|/Tuned %
GPS pos std scale (unitless)	1	0.51	2
Gyro bias std (deg/hr)	2000	500	0.4
Accel bias std (mGal)	10000	6003	1.5
ARW (deg/√hr)	3	3.45	1.4
VRW (m/s/√hr)	0.21	0.07	114
Gyro SF std (ppm)	1000	505	1.2
Accel SF std (ppm)	1000	4210	65

6.5 Real MEMS results on a tightly coupled extended Kalman filter

The tuning strategies employed in this chapter can be applied to a variety of different filter configurations. To show this, a tightly coupled extended Kalman filter was tuned with both the serial and RL methods. Furthermore, a different MEMS inertial unit was used. The BEI Motionpak II is a higher grade MEMS inertial device that has much better noise performance than the ADI sensors.

The tightly coupled Kalman filter uses the raw pseudorange and carrier phase measurements, and there is no longer a scaling factor on the GPS Kalman filter position estimates since the entire integration is centralized in one filter; the scaling can now be performed directly on the variance of the pseudorange measurements, not on the output of

another Kalman filter. The loss of one parameter and the addition of two more clock parameters gives a total of eight *a priori* values required for tuning.

The results of the serial tuning are provided in Table 6.11 with detailed plots in Appendix D. All tuning of the tightly coupled filter used zero satellites during simulated outages to prevent differences in dilution of precision when using less than four satellites. The effects of the serial tuning order is even more apparent with more parameters. Only the first two parameters change, while the last six remain fixed to their initial values.

Table 6.11: Serial tuning results for tightly coupled architecture

Error term	Initial value	Tuned value
Clock bias RW (m/√s)	5	0.17
Gyro bias std (deg/hr)	100	57
Accel bias std (mGal)	5000	5000
ARW (deg/√hr)	1	1
VRW (m/s/√hr)	0.4	0.4
Gyro SF std (ppm)	2000	2000
Accel SF std (ppm)	2000	2000
Clock drift RW (m/s/√s)	0.5	0.5

The serial tuned minFunc value was 109 metres with an average position drift of 81 metres and predicted position drift of 42 metres. This solution is much too optimistic due to the inability to change previously tuned values when performing the serial tune.

6.5.1 RL tuning

The same data was processed with the RL tuning algorithm. 300 steps were taken before convergence was reached. Significantly more steps were needed due to the increase in parameters and uncertainty on the parameter ranges which were left very broad due to the unknown nature of the clock parameters. 10 steps defined an episode, and the average minFunc value over the 30 episodes is shown in Figure 6.14. The general trend is a decreasing minFunc value, with some learning and trial periods. The best minFunc after 300 steps was 89 metres. This solution was actually found after 202 steps (20th episode), but could not be trusted until converged upon at the 30th episode, at which point the minFunc value dropped significantly.

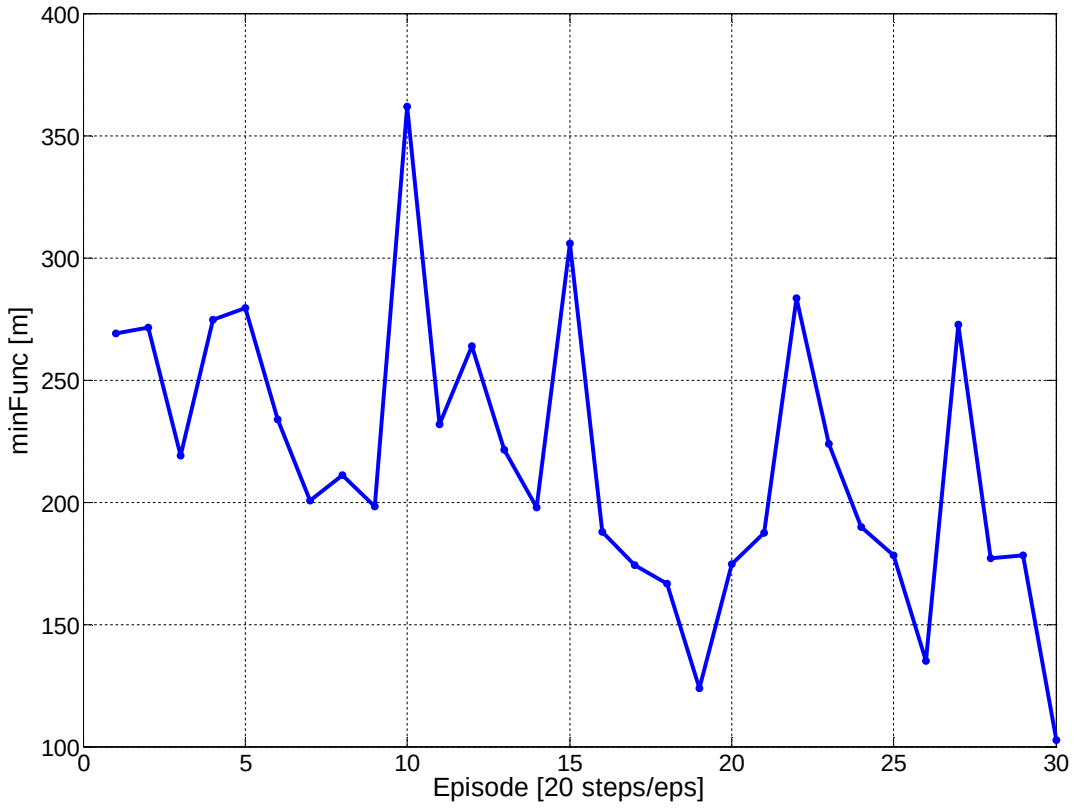

Figure 6.14: Average minFunc RL rewards for tightly coupled filter

The minFunc performance measure balances accuracy and consistency within the solution. This is evident from Figure 6.15 which shows the minFunc signal, the position error signal and the predicted position error signal. The best minFunc value returned an actual position error of 89 metres with a predicted error of 76 metres. It is clear that the RL method minimizes position error while also making the filter consistent. In this case the predicted error starts off too optimistic and is brought to the level of the true accuracy as the position errors are decreased; both signals converge towards each other.

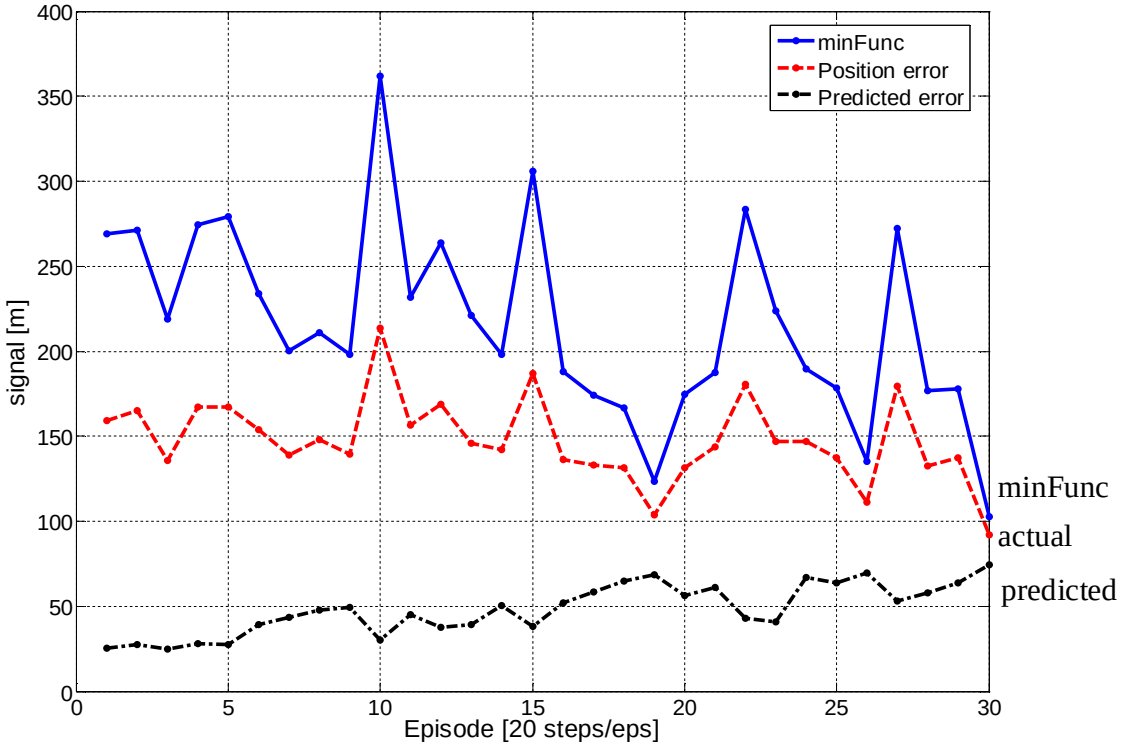

Figure 6.15: Tightly coupled minFunc, position error, and predicted error

The RL solution for the tightly coupled filter is given in Table 6.12. The values are very different than the serial solution since the RL has made the filter significantly more consistent with similar accuracy.

Table 6.12: RL solution for the tightly coupled filter

Error term	Initial value	Tuned value
Clock bias RW (m/√s)	5	10.1
Gyro bias std (deg/hr)	100	63
Accel bias std (mGal)	5000	2460
ARW (deg/√hr)	1	1.94
VRW (m/s/√hr)	0.4	0.11
Gyro SF std (ppm)	2000	4060
Accel SF std (ppm)	2000	9817
Clock drift RW (m/s/√s)	0.5	0.014

CHAPTER 7: CONCLUSIONS AND RECOMMENDATIONS

This chapter brings together the theoretical and practical results to create a unified view of the intelligence added to the filtering algorithm for INS/GPS integrations. All conclusions and recommendations are made in an attempt to improve usability and adoption of low-cost integrated systems for large scale consumer use.

7.1 Conclusions

Due to their symbiotic relationship, inertial and GPS systems can be very useful for improving navigation reliability and accuracy. There are many issues that need to be resolved before these systems can be used in very low-cost integrations. Some of the more important considerations are calibration, alignment and modelling.

Restrictions on the user prevent quality alignment before the device is used. Many newer techniques have been developed to deal with alignment, such as in-motion alignment using GPS or other external signals. Further work is being done to deal with misalignment between the body and vehicle frames, but additional filtering algorithms have been shown to perform well for a full six degree of freedom IMU (Syed, 2007), and for partial IMU configurations (El-Sheimy, 2008).

The issue of calibration and modelling is a serious hurdle that remains. Individual MEMS units are calibrated using general parameters for both deterministic and stochastic models.

Due to the large variance of these low-cost sensors the general parameters are usually far from optimistic, and the imposition of pre-deployment calibration and modelling for individual sensors is not cost effective. For these reasons, on-line intelligent methods were used to adapt the model parameters towards more realistic values.

There are two approaches to this problem that were presented in this thesis. The first is very non-invasive, and simply compensates the filter state estimates without adjustment of the internal filter. The actual error signal can be formed as a target by using two Kalman filters which are run in parallel. Using temporal and dynamic inputs, the deterministic part of the error signal is predicted and removed from the filter output. Results showed that positional state estimates could be brought back to the level of a well-tuned filter using these intelligent state compensations; in the case considered in this thesis that meant an improvement of horizontal positioning accuracy approaching 50%, after being initially degraded by 55%.

The issues with this type of state compensation were the injection of additional system noise, due to the noisy training reference, and the problem of applying predictions when the system was not capable of improving. A fuzzy reliability monitor was applied to properly weight both parallel outputs using expert knowledge that was adapted over time. This method used the uncompensated EKF predictions when the intelligent compensations were not reliable, and applied the intelligent compensations to larger outages once training convergence was reached. An adaptive FIS was used to further adapt certain membership functions to weight predictions across time.

Two separate results were demonstrated for the intelligent compensations combined with the fuzzy reliability monitor. The first set showed that in a worst case scenario the EKF filter predictions could be used with a minimal 2% degradation of the original predictions. As the intelligent compensations converged, the fuzzy system was used to apply weighting between the two sets of filters to improve the intelligent compensations by an additional 20% in horizontal positioning accuracy.

The second intelligent approach to solving the problem of calibrations and modelling was to use a form of adaptive feedback to the internal parameters of the filter based on relative differences of state estimates. A reinforcement learning algorithm was used to alter the *a priori* parameters of the system and measurement noise matrices within loosely and tightly coupled extended Kalman filters. The reinforcement learning used trial and error to develop a model of how to tune the parameters so as to minimize a performance measure that weighted accuracy and consistency of the filter position state estimates. This type of relative learning algorithm could form its own tuning model, and then exploit this model to obtain the best results in terms of positional state estimates.

Results applied to real MEMS data for a loosely coupled EKF showed that the RL method could quickly and efficiently tune many parameters in parallel, which made the filter more accurate and consistent than a manually tuned method using expert knowledge. Most importantly, the RL method required no designer input beyond what was already known from the datasheets. In comparison to the manufacturer suggested values, the tuning was

able to reduce the actual position error by over 50 metres. Furthermore, the filter consistency was improved by over 700 metres, demonstrating a significant improvement in reliability.

The tuning knowledge acquired from one MEMS unit was then applied to tune another similar MEMS unit. Convergence was obtained three times faster when using knowledge from a previous sensor, and the variations during learning were significantly reduced due to accumulated knowledge.

The RL method was also applied to a tightly coupled EKF that contained more *a priori* parameters. The method clearly displayed convergence properties between accuracy and consistency of the filter during tuning. In this case, the RL method improved positioning accuracy by over 55 metres and consistency by 75 metres. The algorithm was shown to be robust regardless of the type of filter being tuned; the same internal RL algorithm and state outputs were used, all that was changed were the *a priori* tuning inputs.

7.2 Recommendations

This work has combined existing filtering methods with newer mathematical methods that adapt their models over time or samples. The mathematical concepts are not new, but with the addition of learning techniques with filtering methods, usability of integrated devices can be improved without additional cost to the designer or manufacturer. The proposed intelligent methods are new in their application to INS/GPS data and much work is still to be done before adoption of these or similar methods.

Temperature and misalignment errors have not been included in the analyses. Manufacturers are beginning to compensate for temperature before their devices are used, but for very low-cost inertial systems temperature can still be an issue for bias and scale factor estimation. If temperature signals are available, it is recommended that these be added to the input of the ANN's used in Chapter 4.

Misalignment is another error that can be included as an input in terms of a noise parameter. This would mean addition of the deterministic misalignment error in the filter state estimation, which was not included in the EKF of this analysis. Nevertheless, this state augmentation can be made easily and the rigid body deformation could be modelled with a white noise term similar to other sensor errors.

If misalignment and temperature are added to the intelligent method of Chapter 4, then the RL and intelligent compensations could be used in a single system. The RL could be used for adjustment of *a priori* models while the ANN compensation could be used to correct errors due to temperature, misalignment and dynamics.

The separation of individual sensor signals for tuning should be performed. This analysis assumes a single error model for all three individual sensor types, for the gyroscopes and accelerometers. These should be separated for tuning purposes since individual sensors can have slight variances. This would increase the number of tuning parameters but not the overall states of the filter.

The minFunc equation given in Chapter 6 assesses only the position accuracy and prediction quality. For many applications it may be beneficial to also include the velocity and attitude signals into the minFunc equation. This can easily be done by extending this equation to include RMS values of all available EKF state errors and predictions. The designer could then scale these values based on importance or weight them all equally.

Finally, more data should be processed to fully assess the memory retention ability of the intelligent methods. This type of analysis could only be done with many sensors over many different data sets. The effect of leveraging and memory transfer can be fully assessed and made statistically significant with more data.

REFERENCES

Abdel-Hamid, W. (2005), *Accuracy Enhancement of Integrated MEMS-IMU/GPS Systems for Land Vehicular Navigation Applications*, PhD Thesis, The University of Calgary, UCGE Report 20207.

Abeel, P., Coates, A., Montemerlo, M., Ng, A.Y., and Thrun, S. (2005), *Discriminative Training of Kalman Filters*, Robotics; Science and Systems 2005 Proceedings.

Akesson, B.M., Jorgensen, J.B., and Poulsen, N.K. (2007), *A Tool for Kalman Filter Tuning*, 17th European Symposium on Computer Aided Process Engineering.

Altmann, S.L. (1986), *Rotations, Quaternions and Double Groups*, Oxford University Press, Oxford, New York.

Barto, A.G., Bradtke, S.J., and Singh, S.P. (1991), *Real-time learning and control using asynchronous dynamic programming*. Technical Report 91-57. Department of Computer and Information Science, University of Massachusetts, Amherst.

Barshan, B., and Durrant-Whyte, H.F. (1995), *Inertial Navigation Systems Mobile Robots*, IEEE Transactions on Robotics and Automation, Vol. 11, No. 3, June, pp. 328-342.

Basil, H., Ananthasayanam, M.R., and Puri, S.N. (2004), *Adaptive Kalman Filter Tuning In Integration of Low-Cost MEMS-INS/GPS*, AIAA Guidance, Navigation, and Control Conference and Exhibit, Providence, RI, August 2004.

Bellman, R.E. (1957), *Dynamic Programming*. Princeton University Press, Princeton.

Bolognani, S., Tubiana, L., and Zigliotto, M. (2003), *Extended Kalman Filter Tuning in Sensorless PMSM Drives*, IEEE Transactions on Industry Applications, Vol. 39, No. 6, December 2003, pp. 1741-174.

Brown, A.K., and Lu, Y. (2004), *Performance Test Results of an Integrated GPS/MEMS Inertial Navigation Package*, ION GNSS 2004, Long Beach, California, September 2004.

Brown, R.G., and Hwang, P.Y.C. (1997), *Introduction to Random Signals and Applied Kalman Filtering*, John Wiley & Sons, Inc., third edition.

Buijtenen, W.M., Schram, G., Babuska, R., and Verbruggen, H.B. (1998), *Adaptive Fuzzy Control of Satellite Attitude by Reinforcement Learning*, IEEE Transactions on Fuzzy Systems, Vol. 6, No. 2, May 1998.

Chiang, K-W. (2004), *INS/GPS Integration using Neural Networks for Land Vehicular Navigation Applications*, PhD Thesis, The University of Calgary, UCGE Report 20209.

Choukroun, D., (2003), *Novel Methods for Attitude Determination using Vector Observations*, PhD Thesis, Isreal Institute of Technology.

Chowdhary, M., Colley, J., and Chansarkar, M. (2007), *Improving GPS Location Availability and Reliability by Using a Suboptimal, Low-Cost MEMS Sensor Set*, ION GNSS 2007 conference proceedings, Fort Worth, Texas, September 2007.

Cichocki, A., and Unbehauen, R. (1993), *Neural Networks for Optimization and Signal Processing*, John Wiley & Sons.

Cichosz, P. (1995), *Truncating temporal differences: On the efficient implementation of TD(λ) for reinforcement learning*. Journal of Artificial Intelligence Research, Vol. 2, pp. 287-318.

Crassidis, J.L., and Junkins, J.L. (2004), *Optimal Estimation of Dynamic Systems*, Chapman & Hall/CRC.

Cybenko, G. (1988), *Approximation by Superpositions of a Sigmoidal Function*, Urbana: University of Illinois.

DeMers, D., and Cottrell, G. (1993), *Non-linear dimensionality reduction*, In Advances in Neural Information Processing Systems, Morgan Kauffman, San Mateo, CA, pp. 580-587.

Dayan, P. (1992), *The convergence of TD(λ) for general λ*, Machine Learning, 8:341-362.

Dorf, R.C., and Bishop, R.H. (2000), *Modern Control Systems*, Prentice Hall, NJ.

El-Mowafy, A. (1994), *Kinematic Attitude Determination From GPS*, PhD Thesis, The University of Calgary, UCGE Report 20074.

El-Sheimy, N. (2008), *Less is More: The Potential of Partial IMU's for Land Vehicle Navigation*, Inside GNSS, Spring 2008, pp. 16-25.

El-Sheimy, N., Shin, E-H., and Niu, X. (2006), *Kalman Filter Face-Off*, Inside GNSS, March 2006, pp. 48-54.

Farell, J.L. (2007), *Inertial Instrument Error Characterization*, Navigation Journal of the Institute of Navigation, Fall 2007, Volume 54, Number 3, pp. 169-176.

Fischler, M.A., and Firschein, O. (1987), *Intelligence: The Eye, the Brain, and the Computer*, Addison-Wesley, Reading, MA.

Forrest, M., Spracklen, T., and Ryan, N. (2000), *An Inertial Navigation Data Fusion System employing an Artificial Neural Network as the Data Integrator*, Institute of Navigation NTM 2000, Anaheim, CA, January 26-28, 2000.

Frost & Sullivan (2006), *World MEMS Sensors Markets: FA0A-32*, Palo Alto, CA, United States.

Gao, Y., Krakiwsky, E.J., and Abousalem, M.A. (1993), *Comparison and Analysis of Centralized, Decentralized, and Federated Filters*, Navigation: Journal of the ION, Vol. 40, No. 1, Spring 1993.

Gelb, A. (1974), *Applied Optimal Estimation, MIT Press, Fifth Edition.*

Goodall, C. (2006), *Intelligent Integration of a MEMS IMU with GPS using a Reliable Weighting Scheme*, ION GNSS 2006, Fort Worth, Texas.

Grewal, M.S., Weill, L.R., and Andrews, A.P. (2001), *Global Positioning Systems, Inertial Navigation, and Integration*, John Wiley & Sons.

Groves, P.D. (2008), *GNSS Solutions: What are the differences between the coherent and non-coherent versions of deep integration of combined inertial navigation GNSS systems (INS/GNSS)?*, Inside GNSS, January-February 2008, pp.26-37.

Groves, P.D. (2008), *Principles of GNSS, Inertial and Multisensor Navigation Systems*, Artech House.

Guyon, I. (1991), *Applications of neural networks to character recognition*, International Journal of Pattern Recognition and Artificial Intelligence, Vol 5. pp. 353-382.

Hatch, R.R., and Knight, J.E. (1985), *Method and Apparatus for Smoothing Code Measurements in a Global Positioning System Receiver*, United States Patent 5,471,217.

Haykin, S. (2001), *Kalman Filtering and Neural Networks*, John Wiley & Sons.

Haykin, S. (1994), *Neural Networks: A Comprehensive Foundation*, IEEE Press.

Hou, H. (2004), *Modeling Inertial Sensor Errors Using Allan Variance*, MSc Thesis, The University of Calgary, UCGE Report 20201.

Hou, H., El-Sheimy, N. (2003), *Inertial Sensors Errors Modeling using Allan Variance*, Institute of Navigation GPS/GNSS 2003 proceedings, Sept 9-12, Portland, OR.

Hu, C., Chen, W., Chen, Y., and Liu, D. (2003), *Adaptive Kalman Filtering for Vehicle Navigation*, Journal of Global Positioning Systems, Vol. 2, No. 1, pp. 42-47.

IEEE Standard 952-1997, *IEEE Standard Specification Format Guide and Test Procedure for Single –Axis Interferometric Fiber Optic Gyros*.

Kaufmann, A., and Gupta, M.M. (1985), *Introduction to Fuzzy Arithmetic*, V.N. Reinhold.

Klir, G.J., and Yuan, B. (1995), *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall.

Kohonen, T. (2001), *Self-Organizing Maps*, Springer, Information Sciences, third edition.

Korniyenko, O.V., Sharawi, M.S., and Aloi, D.N. (2005), *Neural Network Based Approach for Tuning Kalman Filter*, IEEE Conference on Electro Information Technology, May 2005, pp.1-5.

Kantardzic, M. (2001), *Data Mining: Concepts, Models, Methods, and Algorithms*, IEEE Press.

Kaplan, E.D. (1996), *Understanding GPS Principles and Applications*, Artech House Publishers.

Khosla, D. (2004), *Adaptive Kalman Filter Method For Accurate Estimation of Forward Path Geometry of an Automobile*, United States Patent, US 6718259 B1, April 6, 2004, Assignee: HRL Laboratories.

Lawrence, A. (1998), *Modern Inertial Technology: Navigation, Guidance, and Control*, Springer, Mechanical Engineering Series, second edition.

Li, Y., Mumford, P., and Rizos, C. (2008), *Seamless Navigation through GPS Outages: A Low-Cost GPS/INS Solution*, Inside GNSS, August 2008, pp. 39-44 & pp. 53.

Li, Y. (2008), *GNSS Solutions: How can a GPS receiver or MEMS (micro-electro mechanical systems) inertial sensor assembly sense a host platform's orientation? How can these sensor technologies be combined together?*, Inside GNSS, January-February 2008, pp.22-26.

Lu, G. (1995), *Development of a GPS Multi-Antenna System for Attitude Determination*, UCGE Report 20073.

Macias, J.A.R., and Esposito, A.G. (2006), *Self-Tuning of Kalman Filters for Harmonic Computation*, IEEE Transactions on Power Delivery, Vol. 21, No. 1, January 2006.

Mannings, R. (2008), *Ubiquitous Positioning*, Mobile Communications Series, Artech House, London.

Maybeck, P.S. (1982), *Stochastic Process and Estimation*, Vol. I and II, Academic Press, Inc., New York.

McClelland, J.L., Rumelhart, D.E. (1988), *Explorations in parallel distributed processing: A handbook of models, programs, and exercises*. Cambridge, MA: MIT Press.

Mehra, R.K. (1970), *On the Identification of Variances and Adaptive Kalman Filtering*, IEEE Transactions on Automatic Control, Vol. AC-15, No. 2, April 1970, pp. 175-184.

Misra, P. and Enge, P. (2001), *Global Positioning System: Signals, Measurements, and Performance*, Ganga-Jamuna Press.

Mohamed, A.H. (1999), *Optimizing the Estimation Procedure in INS/GPS Integration for Kinematic Applications*, PhD Thesis, The University of Calgary, UCGE Report 20127.

Nasiri, S. (2005), *A Critical Review of MEMS Gyroscopes Technology and Commercialization Status*, White Paper, InvenSense.

Nassar, S. (2003), *Improving the Inertial Navigation System (INS) Error Model for INS and INS/DGPS Applications*, PhD Thesis, The University of Calgary, UCGE Report 20183.

Nassar, S., Niu, X., Aggarwal, P., and El-Sheimy, N. (2006), *INS/GPS Sensitivity Analysis Using Different Kalman Filter Approaches*, ION NTM conference proceedings, January 2006.

Negoita, M., Palade, V., and Neagu, D. (2005), *Computational Intelligence: Engineering of Hybrid Systems*, Springer Verlag.

Niu, X., Nassar, S., and El-Sheimy, N. (2007), *An Accurate Land-Vehicle MEMS IMU/GPS Navigation System using 3D Auxiliary Velocity Updates*, Navigation Journal of the Institute of Navigation, Fall 2007, Volume 54, Number 3, pp. 177-188.

Ojeda, L., Reina, G. and Borenstein, J. (2004), *Experimental Results from FLEXnav: An Expert Rule-Based Dead-reckoning System for Mars Rovers*, IEEE Aerospace Conference 2004, Big Sky, MT, March 6-12, 2004.

Oppenheim, A.V., Willsky, A.S., and Nawab, S.H. (1997), *Signals & Systems*, Prentice Hall, Second Edition.

Pasadyn, A.J. (2004), *Kalman Filter State Estimation for a Manufacturing System*, United States Patent, US 6757579 B1, June 29, 2004, Assignee: Advanced Micro Devices Inc.

Petovello, M.G. (2003), *Real-time Integration of a Tactical-Grade IMU and GPS for High-Accuracy Positioning and Navigation*, PhD thesis, The University of Calgary, UCGE Report 20173.

Petovello, M.G., Cannon, M.E., and Lachapelle, G. (2004), *Benefits of Using a Tactical-Grade IMU for High-Accuracy Positioning*, Navigation: Journal of The ION, Vol. 51, No. 1, Spring 2004.

Rummery, G.A. (1995), *Problem Solving with Reinforcement Learning*, PhD Thesis, Cambridge University.

Rummery, G.A., and Niranjan, M. (1994), *On-line Q-learning using connectionist systems*, Technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University.

Russo, A.P. (1991), *Neural networks for sonar signal processing*, IEEE Conference on Neural Networks for Ocean Engineering, Washington, DC, Tutorial No. 8.

Shin, E. (2001), *Accuracy Improvement of Low cost INS/GPS for Land Application*, Msc Thesis, The University of Calgary, UCGE Report 20156.

Shin, E-H. (2005), *Estimation Techniques for Low-Cost Inertial Navigation*, PhD Thesis, The University of Calgary, UCGE Report 20219.

Shin, E-H., and El-Sheimy, N. (2007), *Unscented Kalman Filter and Attitude Errors of Low-Cost Inertial Navigation Systems*, Navigation Journal of the Institute of Navigation, Spring 2007, Volume 54, Number 1, pp. 1-9.

Simon, D. (2001), *Training radial basis neural networks with the extended Kalman filter*, Elsevier Science Ltd, Neucom, pp.1-21.

Skaloud, J. (1995), *Strapdown INS Orientation Accuracy with GPS aiding*, PhD Thesis, The University of Calgary, UCGE report 20079.

Sobel, D. (1995), *Longitude: The True Story of a Lone Genius Who Solved the Greatest Scientific Problem of His Time*, Walker Publishing Company Inc.

Soloviev, A., and Grass, F. (2007), *Batch-Processing of Inertial Measurements for Mitigation of Sculling and Commutation Errors*, Navigation Journal of the Institute of Navigation, Winter 2007, Volume 54, Number 4, pp. 265-276.

Soloviev, A., Gunawardena, S., and Graas, F. (2008), *Deeply Integrated GPS/Low-Cost IMU for Low CNR Signal Processing: Concept Description and in-Flight Demonstration*, Navigation Journal of the Institute of Navigation, Spring 2008, Volume 55, Number 1, pp. 1-13.

Suga, N. (1990), *Computations of velocity and range in the bat auditory system for echo location*, In Computational Neuroscience, Cambridge, MA, MIT Press, pp. 213-231.

Spiegel, M.S. (1981), *Applied Differential Equations*, Prentice-Hall Inc, Englewood Cliffs, New Jersey.

Strus, J.M., Kirkpatrick, M., and Sinko, J.W. (2008), *GPS/IMU: Development of a High Accuracy Pointing System for Maneuvering Platforms*, Inside GNSS, March/April 2008, pp. 30-37.

Sutton, R.S., and Barto, A. (1998), *Reinforcement Learning: An Introduction*, The MIT Press.

Syed, Z., Niu, X., Aggarwal, P., and El-Sheimy, N. (2007), *Accuracy Degradation of MEMS Integrated Navigation Systems Due to Poorly Aligned Inertial Sensors*, Institude of Navigation National Technical Meeting 2007, San Diego, California, January 22-24, 2007.

Vanicek, P. and Omerbasic, M. (1999), *Does a navigation algorithm have to use a Kalman filter?*, Canadian Aeronautical and Space Institute Journal, Vol. 45, 3, pp. 292-296.

Watkins, C.J.C.H. (1989), *Learning from Delayed Rewards*, PhD Thesis, Cambridge University.

Weston, J.L., and Titterton, D.H. (2004), *Strapdown Inertial Navigation Technology*, Peter Peregrinus Ltd.

Widrow, B., and Hoff, M.E. (1960), *Adaptive switching circuits*, 1960 WESCON Convention Record Part IV, pp. 96-104, Institute of Radio Engineers, New York. Reprinted in J.A. Anderson and E. Rosenfeld, *Neurocomputing: Foundations of Research*, pp. 126-134, MIT Press, Cambridge, MA, 1988.

Yang, Y. (2008), *Tightly Coupled MEMS INS/GPS Integration with INS Aided Receiver Tracking Loops*, PhD Thesis, The University of Calgary, UCGE Report 20270.

Yang, Y., Niu, X., El-Sheimy, N. (2006), *Real-Time MEMS Based INS/GPS Integrated Navigation Systems for Land Vehicle Navigation Application*, Institute of Navigation National Technical Meeting 2006, January 18-20, Monterey, California.

APPENDIX A: ADXRS150 DATASHEET

SPECIFICATIONS

@T_A = 25°C, V_S = 5 V, bandwidth = 80 Hz (C_{OUT} = 0.01 µF), angular rate = 0°/s, ±1g, unless otherwise noted.

Table 1.

Parameter	Conditions	ADXRS150ABG			Unit
		Min[1]	Typ	Max[1]	
SENSITIVITY	Clockwise rotation is positive output				
Dynamic Range[2]	Full-scale range over specifications range	±150			°/s
Initial	@25°C	11.25	12.5	13.75	mV/°/s
Over Temperature[3]	V_{CC} = 4.75 V to 5.25 V	11.25		13.75	mV/°/s
Nonlinearity	Best fit straight line		0.1		% of FS
Voltage Sensitivity	V_{CC} = 4.75 V to 5.25 V		0.7		%/V
NULL					
Initial Null			2.50		V
Null Drift over Temperature[3]	Delta from 25°C			±300	mV
Turn-On Time	Power on to ±½°/s of final		35		ms
Linear Acceleration Effect	Any axis		0.2		°/s/g
Voltage Sensitivity	V_{CC} = 4.75 V to 5.25 V		1		°/s/V
NOISE PERFORMANCE					
Rate Noise Density	@25°C		0.05		°/s/√Hz
FREQUENCY RESPONSE					
3 db Bandwidth[4] (User Selectable)	22 nF as comp cap (see the Applications section)		40		Hz
Sensor Resonant Frequency			14		kHz
SELF TEST					
ST1 RATEOUT Response[5]	ST1 pin from Logic 0 to 1, –40°C to +85°C	–400	–660	–1000	mV
ST2 RATEOUT Response[5]	ST2 pin from Logic 0 to 1, –40°C to +85°C	+400	+660	+1000	mV
Logic 1 Input Voltage	Standard high logic level definition	3.3			V
Logic 0 Input Voltage	Standard low logic level definition			1.7	V
Input Impedance	To common		50		kΩ
TEMPERATURE SENSOR					
V_{OUT} at 298°K			2.50		V
Max Current Load on Pin	Source to common			50	µA
Scale Factor	Proportional to absolute temperature		8.4		mV/°K
OUTPUT DRIVE CAPABILITY					
Output Voltage Swing	I_{OUT} = ±100 µA	0.25		V_S – 0.25	V
Capacitive Load Drive		1000			pF
2.5 V REFERENCE					
Voltage Value		2.45	2.5	2.55	V
Load Drive to Ground	Source		200		µA
Load Regulation	0 < I_{OUT} < 200 µA		5.0		mV/mA
Power Supply Rejection	4.75 Vs to 5.25 Vs		1.0		mV/V
Temperature Drift[3]	Delta from 25°C		5.0		mV
POWER SUPPLY					
Operating Voltage Range		4.75	5.00	5.25	V
Quiescent Supply Current			6.0	8.0	mA
TEMPERATURE RANGE					
Specified Performance Grade A		–40		+85	°C

TYPICAL PERFORMANCE CHARACTERISTICS

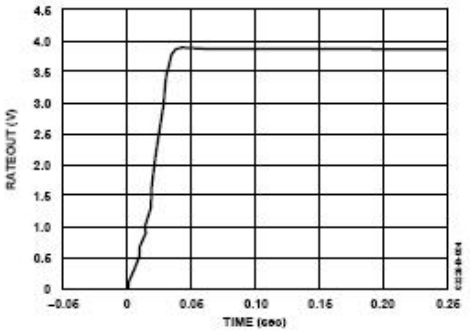

Figure 4. Rate Sensing Start-Up Time

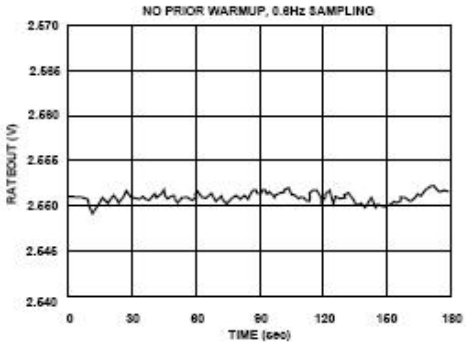

Figure 7. Null Settling Time

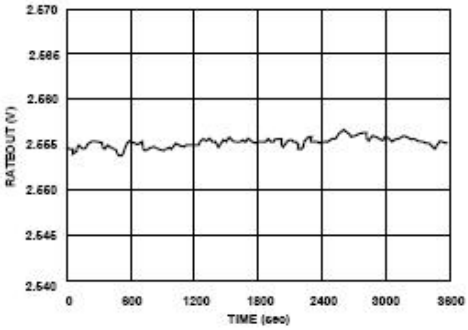

Figure 5. Null Stability for 1 Hour

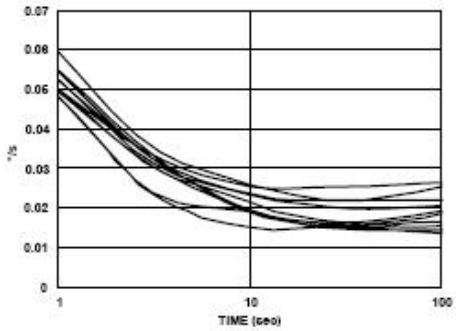

Figure 8. Root Allan Variance vs. Averaging Time

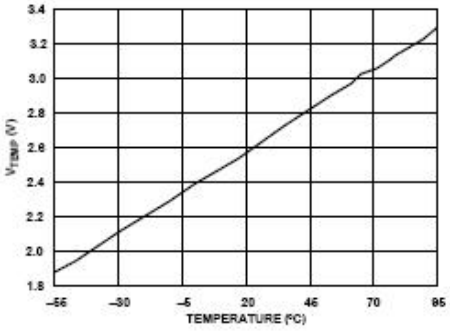

Figure 6. Temperature Sensor Output

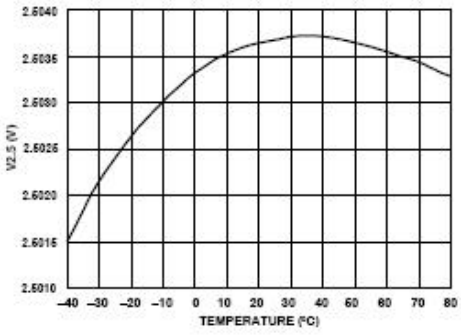

Figure 9. 2.5 V Voltage Reference vs. Temperature

APPENDIX B: ADXL105 DATASHEET

ADXL105–SPECIFICATIONS ($T_A = T_{MIN}$ to T_{MAX}, T_A = +25°C for J Grade Only, V_S = +5 V, @ Acceleration = 0 *g*, unless otherwise noted)

Parameter	Conditions	ADXL105J/A Min	Typ	Max	Units
SENSOR INPUT					
Measurement Range[1]		±5	±7		*g*
Nonlinearity	Best Fit Straight Line		0.2		% of FS
Alignment Error[2]			±1		Degrees
Cross Axis Sensitivity[3]	Z Axis, @ +25°C		±1	±5	%
SENSITIVITY[4] (Ratiometric)	At A_{OUT}				
Initial		225	250	275	mV/*g*
	V_S = 2.7 V	80	105	120	mV/*g*
vs. Temperature[5, 6]			±0.5		%
ZERO *g* BIAS LEVEL[7] (Ratiometric)	At A_{OUT}				
Zero *g* Offset Error	From +2.5 V Nominal	−625		+625	mV
vs. Supply		−20		+20	mV/V_{DD}/V
vs. Temperature[5, 7]			50		mV
NOISE PERFORMANCE					
Voltage Density[7]	@ +25°C		225	325	μ*g*/√Hz
Noise in 100 Hz Bandwidth			2.25		m*g* rms
FREQUENCY RESPONSE					
3 dB Bandwidth		10	12		kHz
Sensor Resonant Frequency		13	18		kHz
TEMP SENSOR[4] (Ratiometric)					
Output Error at +25°C	From +2.5 V Nominal	−100		+100	mV
Nominal Scale Factor			8		mV/°C
Output Impedance			10		kΩ
V_{MID}[4] (Ratiometric)					
Output Error	From +2.5 V Nominal	−15		+15	mV
Output Impedance			10		kΩ
SELF-TEST (Proportional to V_{DD})					
Voltage Delta at A_{OUT}	Self-Test "0" to "1"	100		500	mV
Input Impedance[8]		30	50		kΩ
A_{OUT}					
Output Drive	I = ±50 μA	0.50		V_S − 0.5	V
Capacitive Load Drive		1000			pF
UNCOMMITTED AMPLIFIER					
Initial Offset		−25		+25	mV
Initial Offset vs. Temperature			5		μV/°C
Common-Mode Range		1.0		4.0	V
Input Bias Current[9]			25		nA
Open Loop Gain			100		V/mV
Output Drive	I = ±100 μA	0.25		V_S − 0.25	V
Capacitive Load Drive		1000			pF
POWER SUPPLY					
Operating Voltage Range		2.70		5.25	V
Quiescent Supply Current	At 5.0 V		1.9	2.6	mA
	At 2.7 V		1.3	2.0	mA
Turn-On Time			700		μs
TEMPERATURE RANGE					
Operating Range J		0		+70	°C
Specified Performance A		−40		+85	°C

NOTES
[1]Guaranteed by tests of zero *g* bias, sensitivity and output swing.
[2]Alignment of the X axis is with respect to the long edge of the bottom half of the Cerpak package.
[3]Cross axis sensitivity is measured with an applied acceleration in the Z axis of the device.
[4]This parameter is ratiometric to the supply voltage V_{DD}. Specification is shown with a 5.0 V V_{DD}. To calculate approximate values at another V_{DD}, multiply the specification by V_{DD}/5 V.
[5]Specification refers to the maximum change in parameter from its initial value at +25°C to its worst case value at T_{MIN} to T_{MAX}.
[6]See Figure 3.
[7]See Figure 2.
[8]CMOS and TTL Compatible.
[9]UCA input bias current is tested at final test.

ADXL105–Typical Performance Characteristics

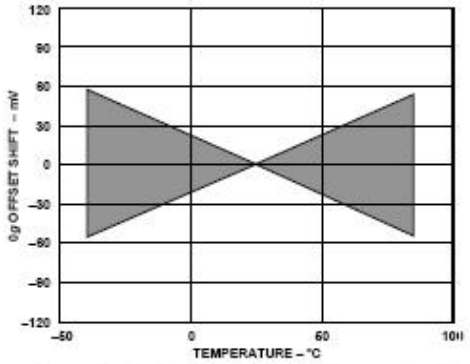

Figure 2. Typical 0 g Shift vs. Temperature*

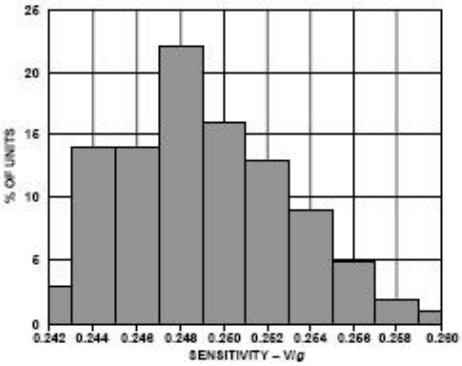

Figure 5. Sensitivity Distribution*

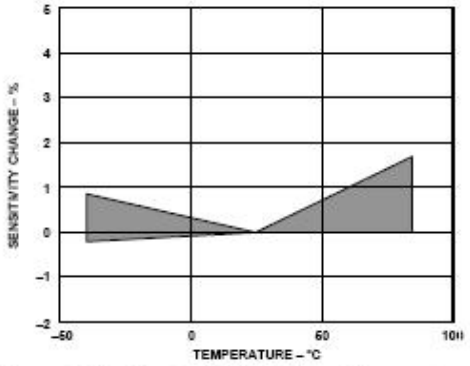

Figure 3. Typical Sensitivity Shift vs. Temperature*

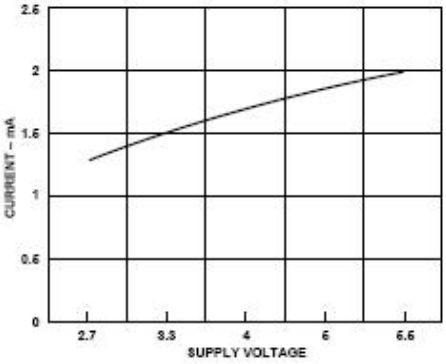

Figure 6. Typical Supply Current vs. Supply Voltage

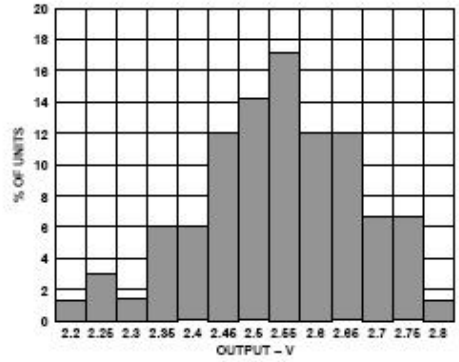

Figure 4. 0 g Output Distribution*

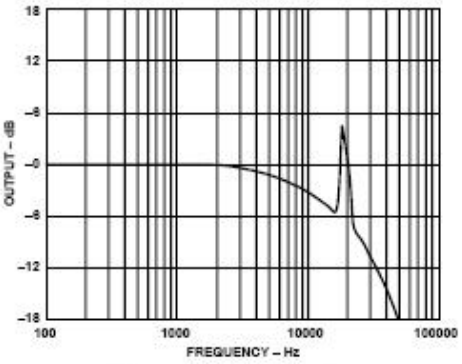

Figure 7. Noise Graph

*Data from several characterization lots.

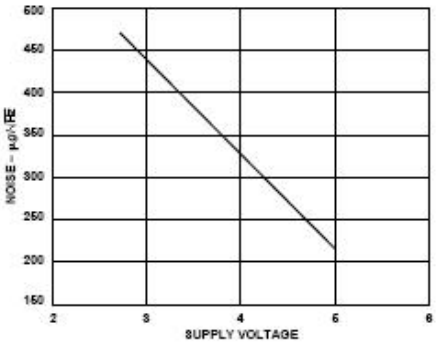

Figure 8. Typical Noise Density vs. Supply Voltage

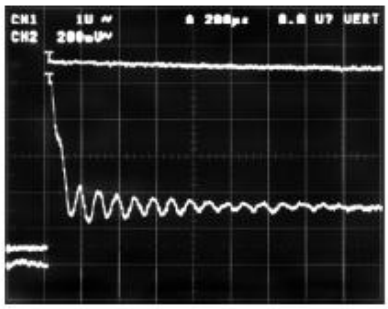

Figure 11. Typical Self-Test Response at V_{DD} = 5 V

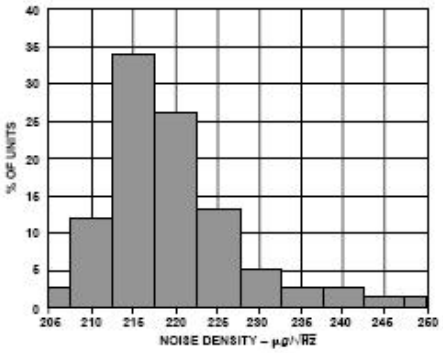

Figure 9. Noise Distribution*

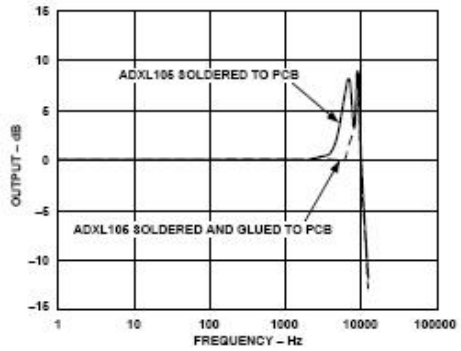

Figure 12. Frequency Response

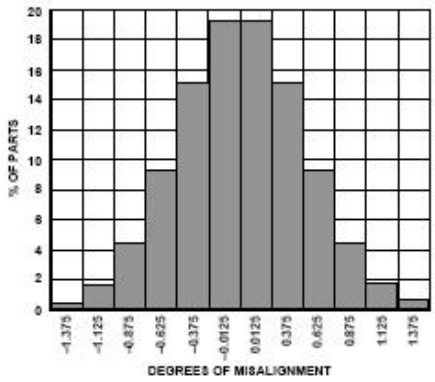

Figure 10. Rotational Die Alignment*

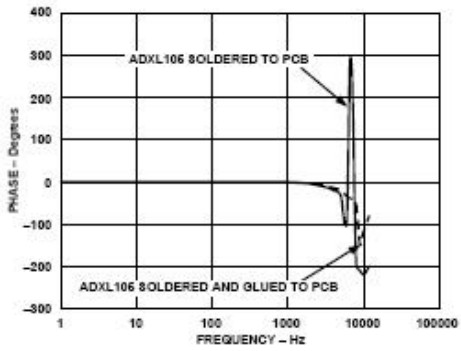

Figure 13. Phase Response

*Data from several characterization lots.

APPENDIX C: ADI SERIAL TUNING PLOTS

Error bounds were used to constrain the optimal solution of the ADI data for the serial tune. This is apparent in the last five plots which do not vary outside of the error bounds, so the solution stayed fixed.

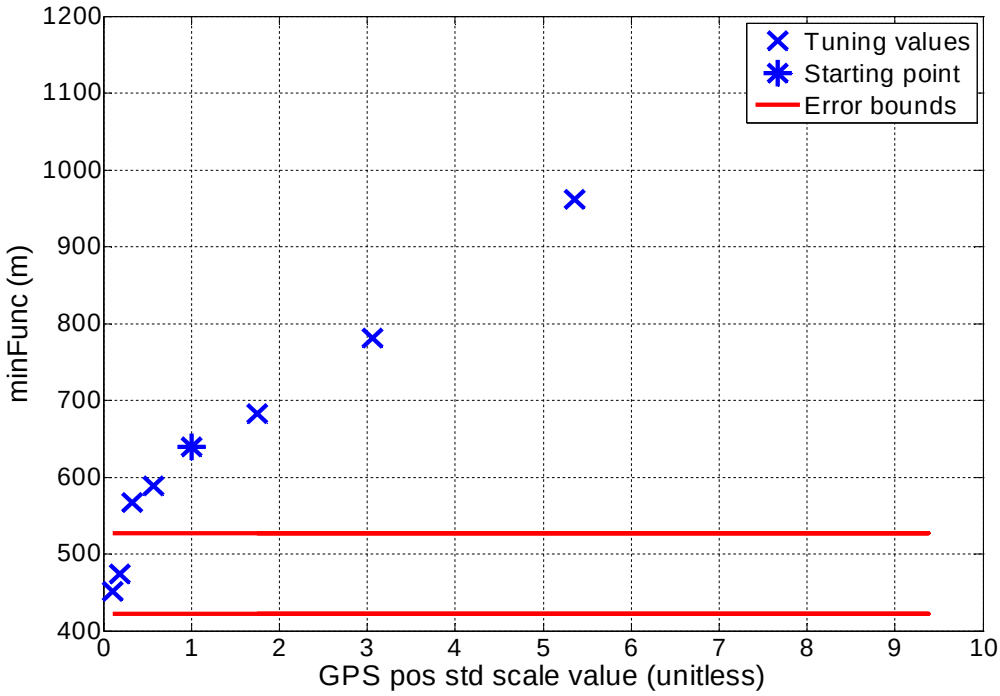

Figure C.1: GPS position std scale for ADI serial tune

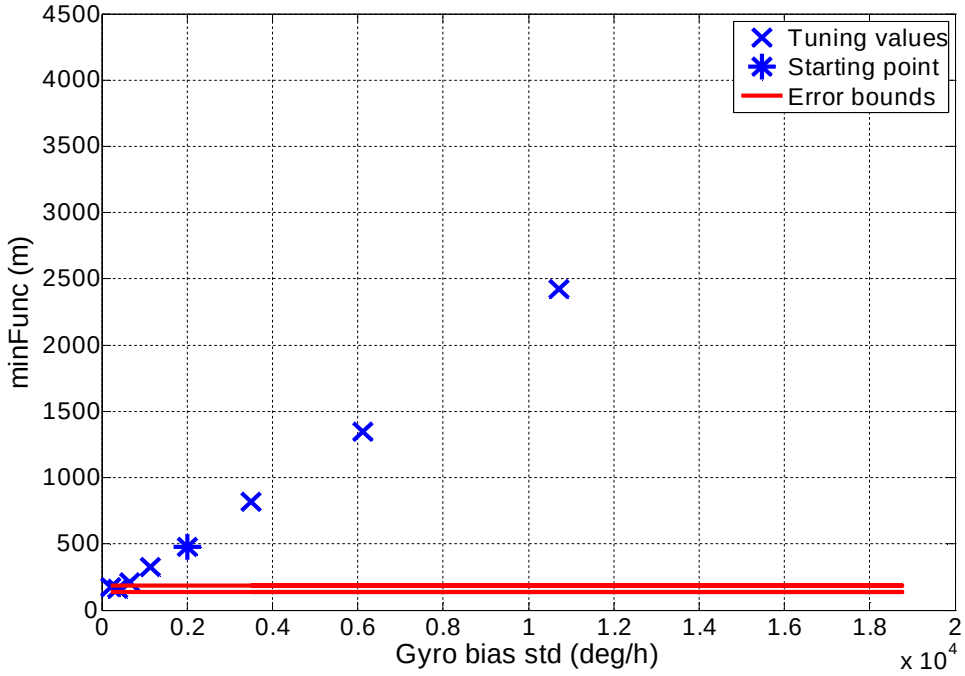

Figure C.2: Gyroscope bias std for ADI serial tune

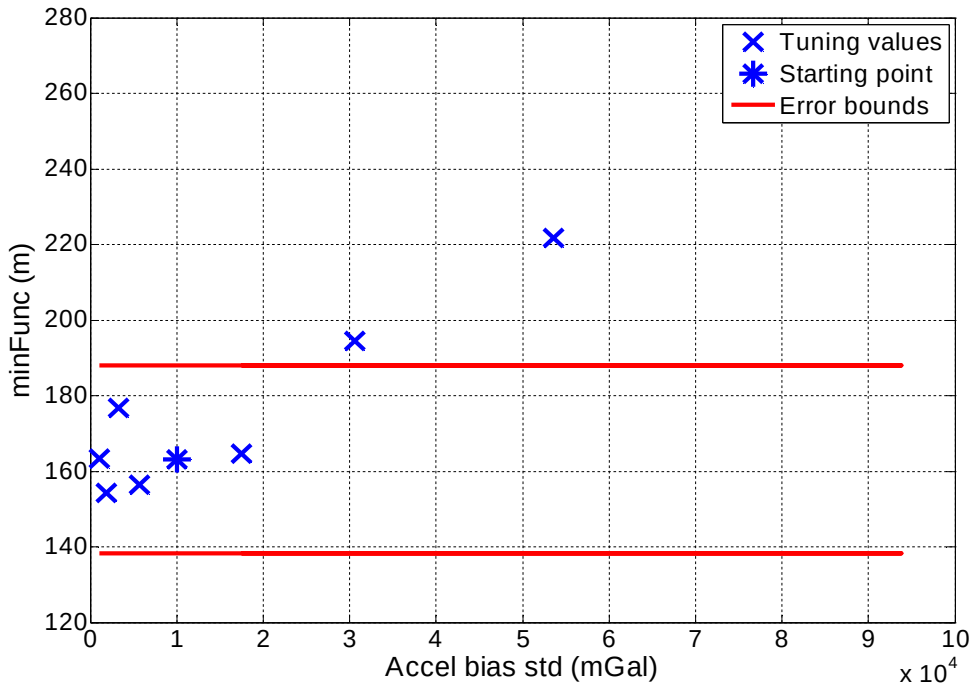

Figure C.3: Accelerometer bias std for ADI serial tune

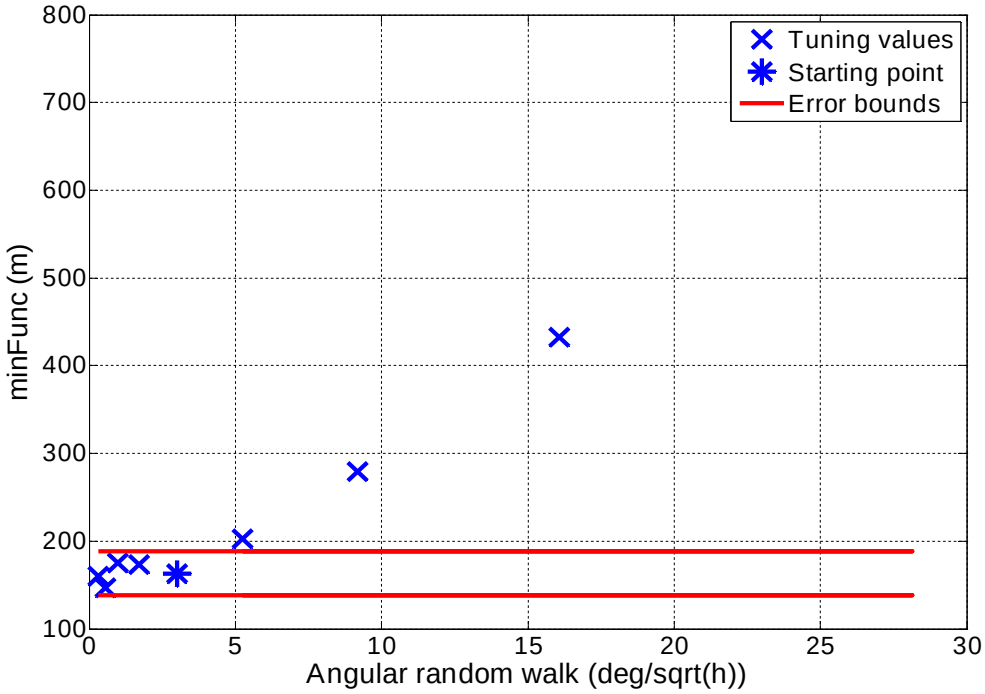

Figure C.4: ARW for ADI serial tune

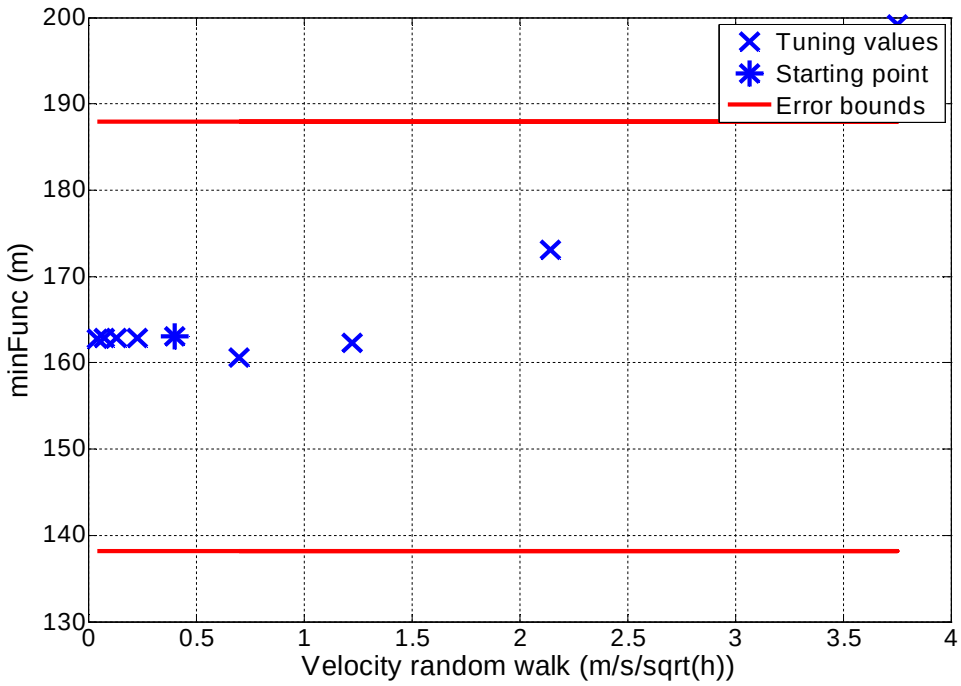

Figure C.5: VRW for ADI serial tune

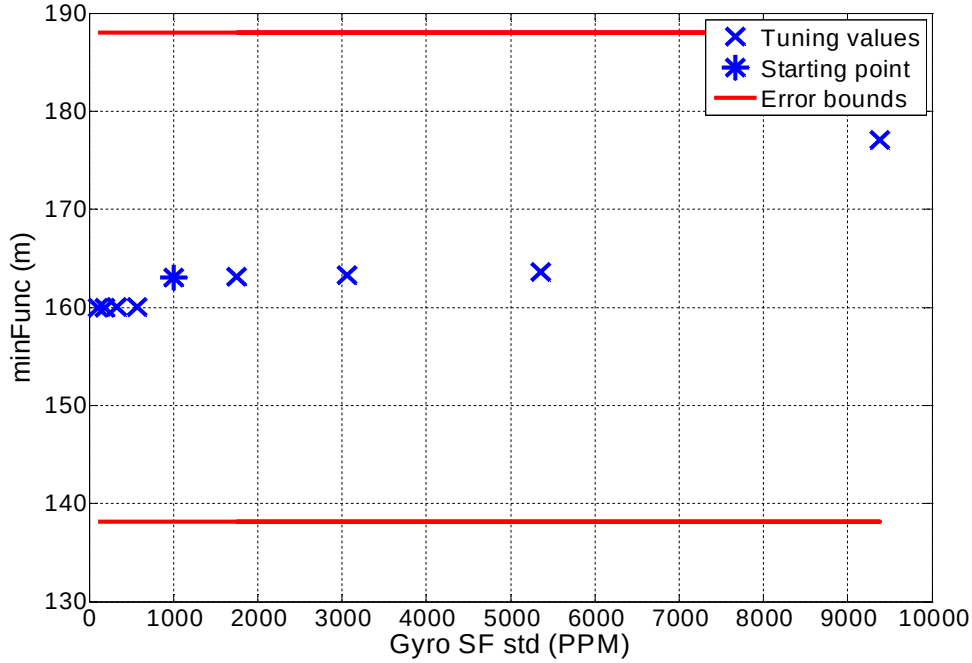

Figure C.6: Gyroscope scale factor std for ADI serial tune

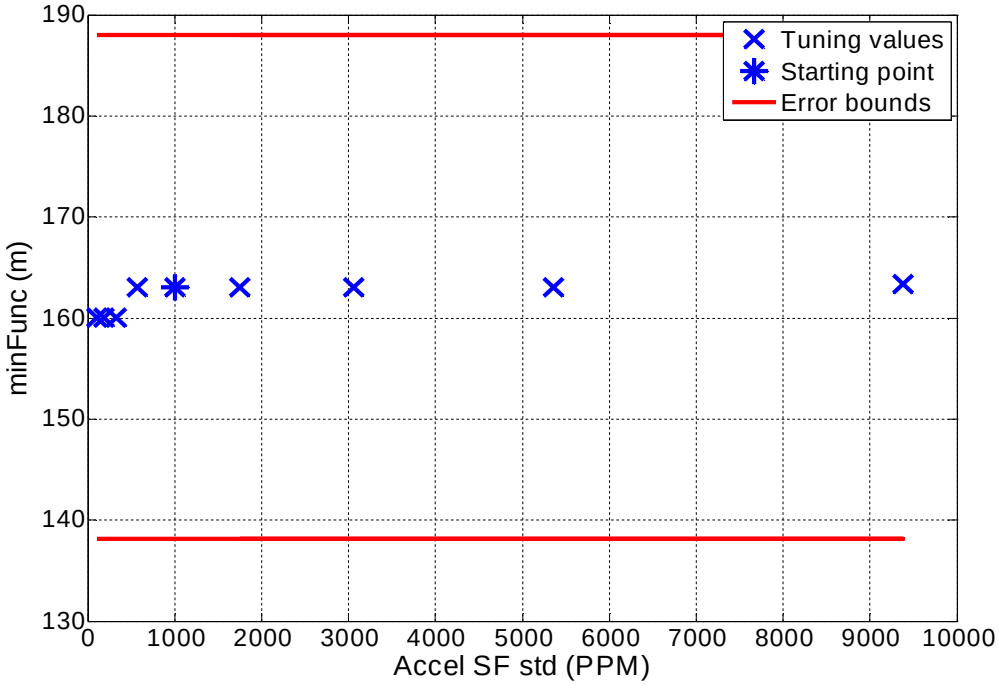

Figure C.7: Accelerometer scale factor std for ADI serial tune

APPENDIX D: SERIAL TUNING FOR TIGHTLY COUPLED FILTER

The tuning order for the tightly coupled filter was performed with the clock parameters first and last to test their relationship to positional state error. As is clear, their order made little difference to the overall results.

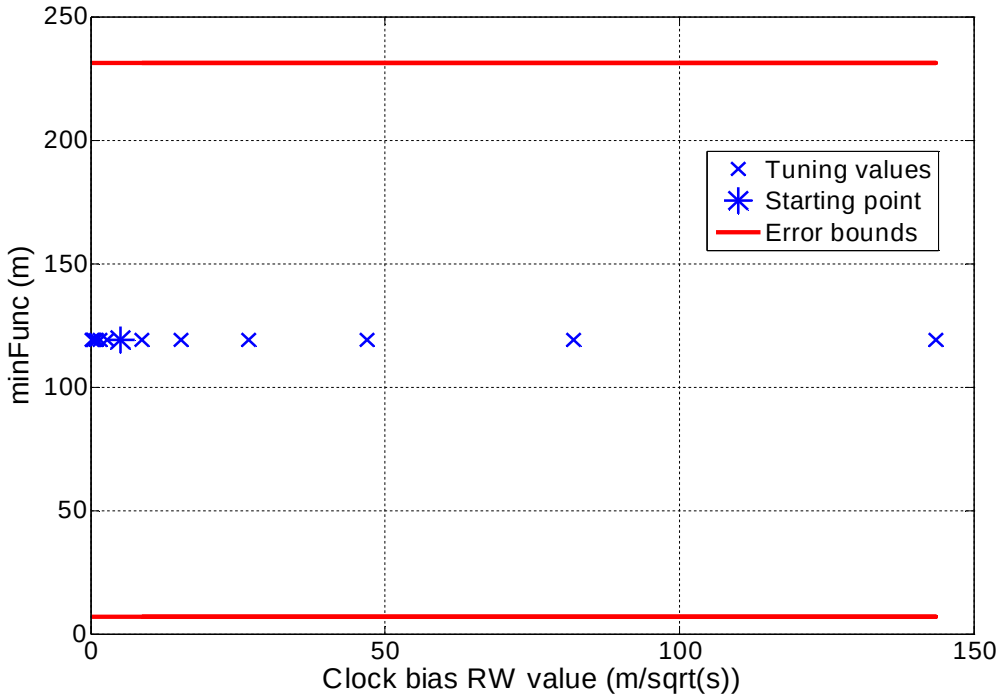

Figure D.1: Clock bias RW tuning for tightly coupled filter

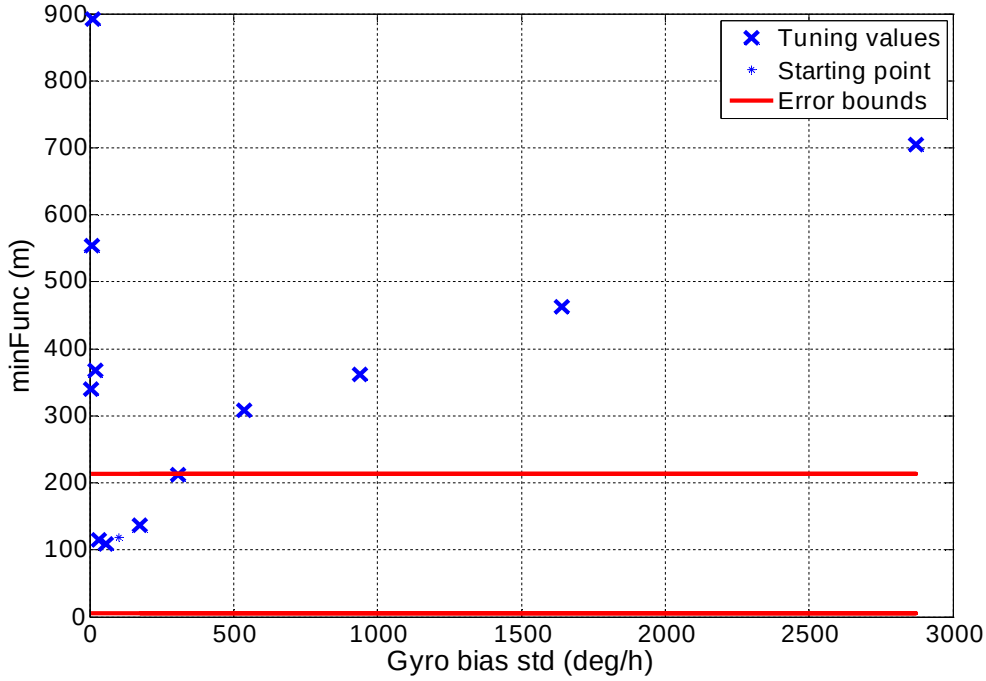

Figure D.2: Gyroscope bias std tuning for tightly coupled filter

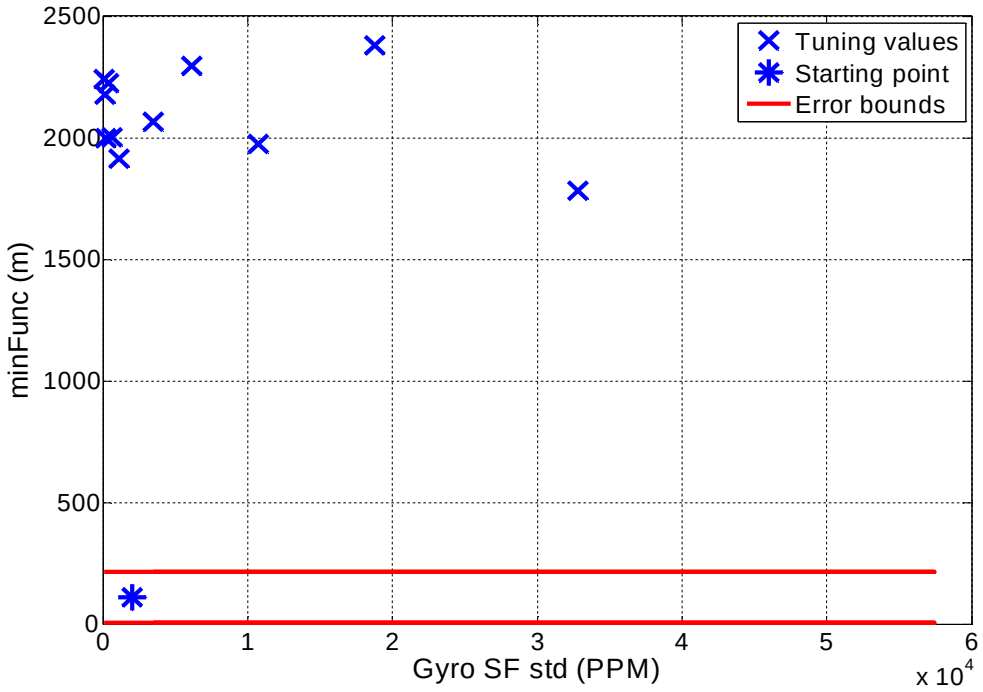

Figure D.3: Gyroscope scale factor tuning for tightly coupled filter

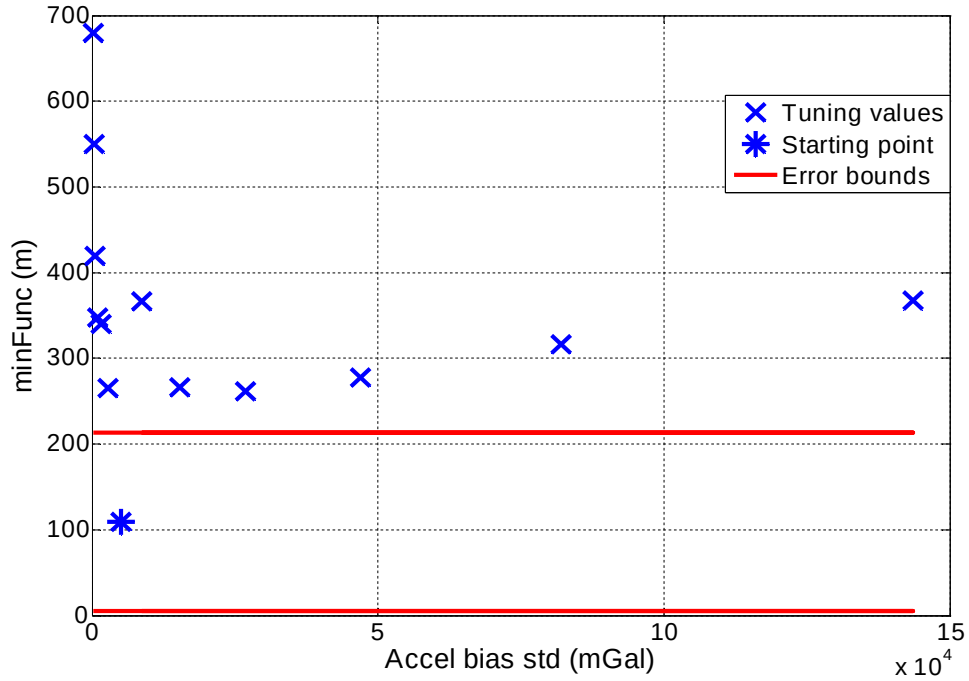

Figure D.4: Accelerometer bias std tuning for tightly coupled filter

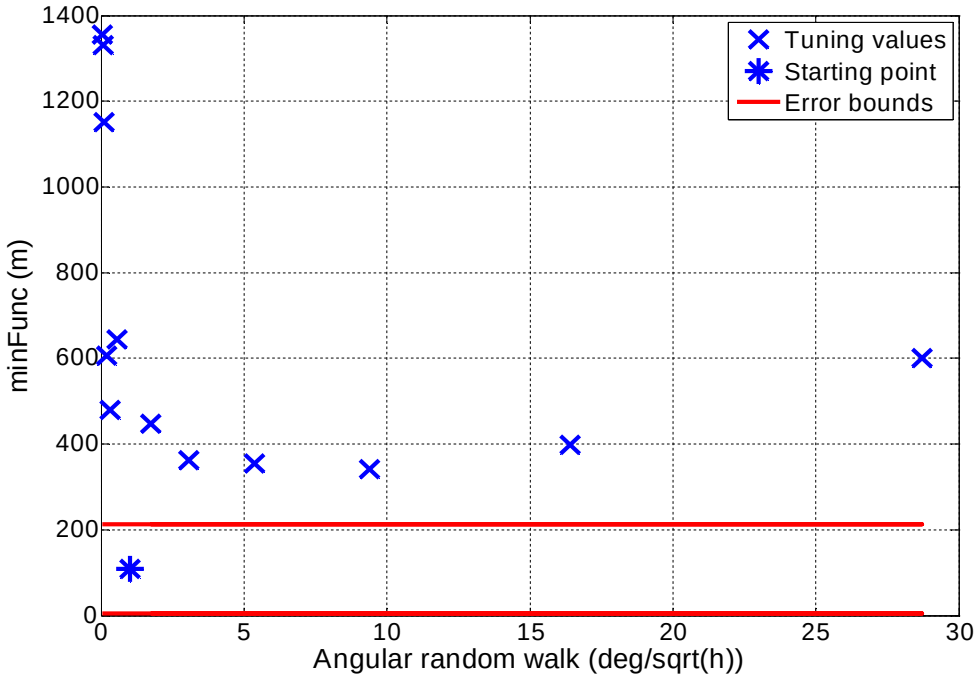

Figure D.5: ARW tuning for tightly coupled filter

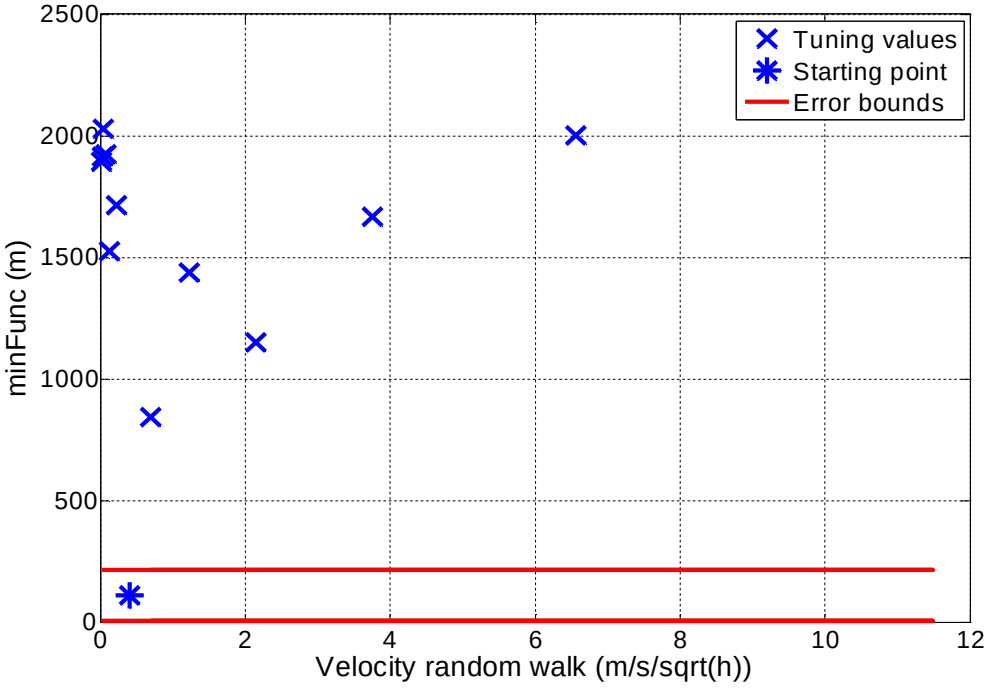

Figure D.6: VRW tuning for tightly coupled filter

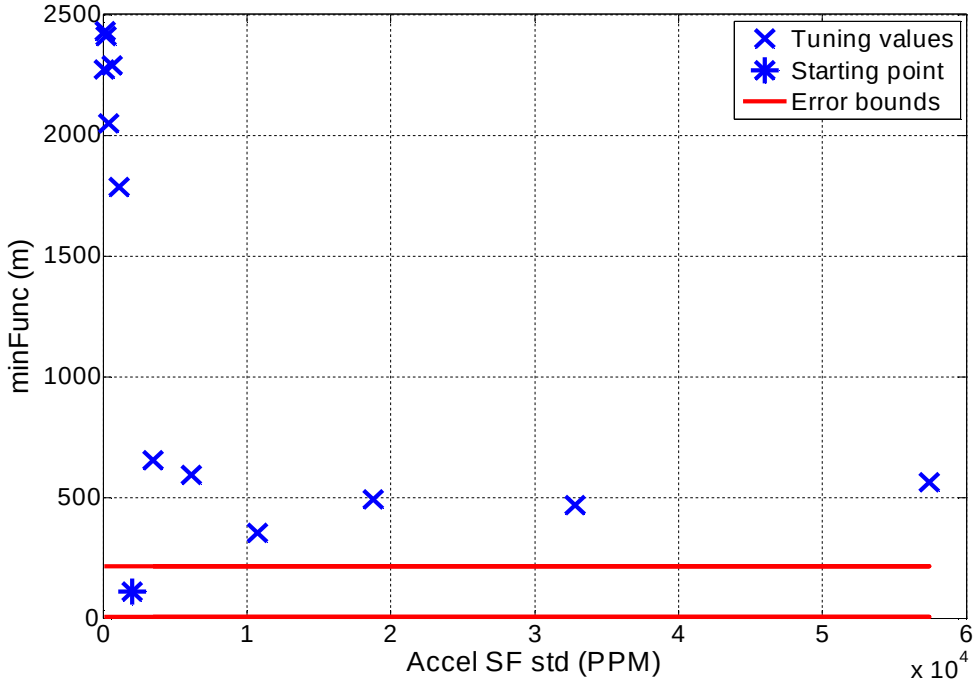

Figure D.7: Accelerometer scale factor std tuning for tightly coupled filter

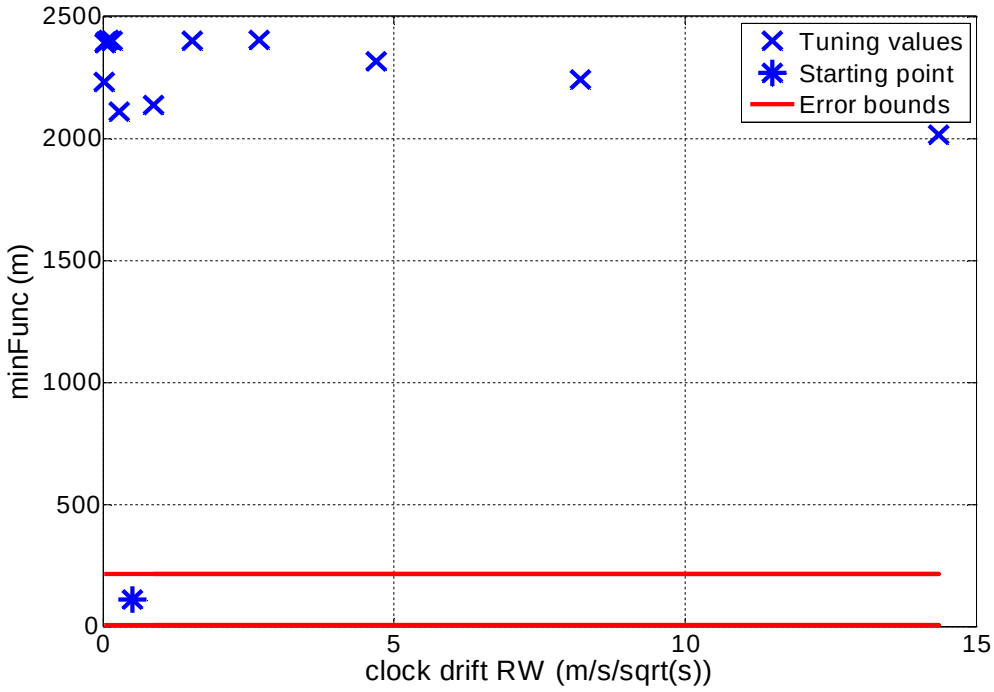

Figure D.8: Clock drift RW tuning for tightly coupled filter

If the clock drift and bias are tuned first, they do not contribute significantly to the overall change in minFunc value as evidenced by the flat response:

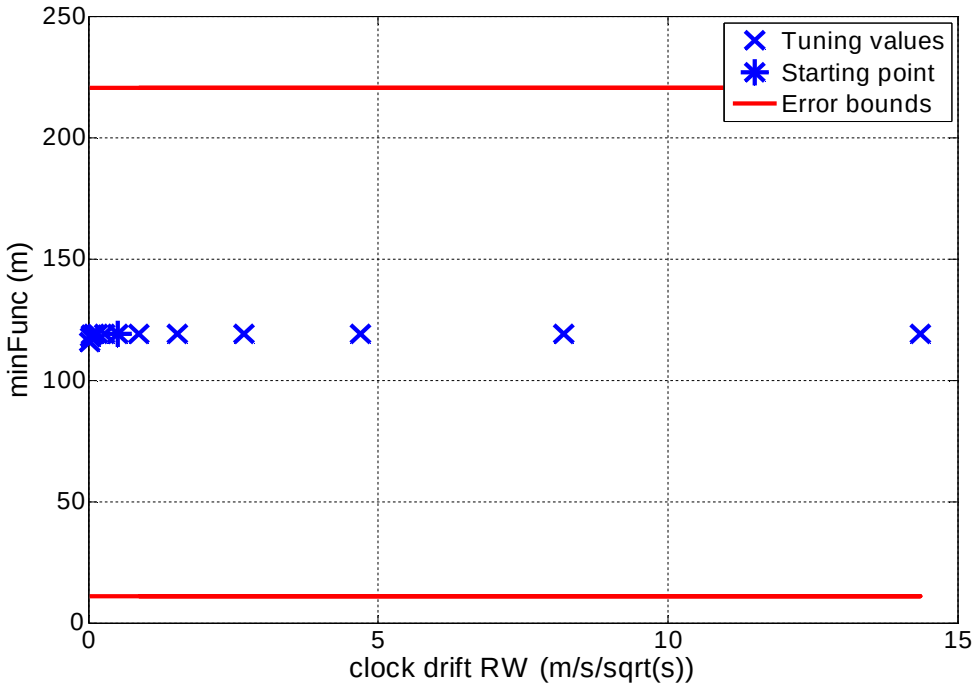

Figure D.9: Clock drift RW tuning done first in order

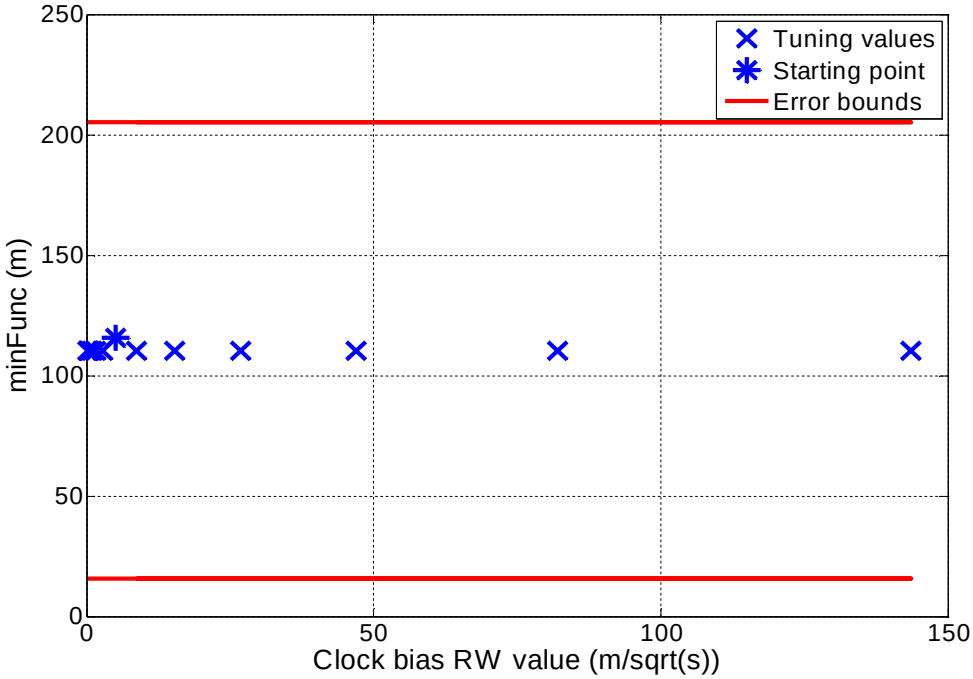

Figure D.10: Clock bias RW tuning done first in order