CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF ELECTRICAL ENGINEERING
DEPARTMENT OF MEASUREMENT

Use of Adaptive Filtering Methods in Inertial Navigation Systems

DOCTORAL THESIS

2010 **Michal Reinštein**

CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF ELECTRICAL ENGINEERING
DEPARTMENT OF MEASUREMENT

Use of the Adaptive Filtering Methods in Inertial Navigation Systems

DOCTORAL THESIS

Ph.D. Programme: *Electrical Engineering and Information Technology*
Branch of study: *Air Traffic Control*
Supervisor: *doc. Ing. Karel Draxler CSc.*
Supervisor-Specialist: *Ing. Jan Roháč Ph.D.*

2010 **Michal Reinštein**

ABSTRACT

This dissertation explores the use of adaptive filtering methods in inertial navigation systems (INS). A navigation system was developed for a unique quadruped robot to provide complete orientation and position information. The design of the navigation algorithm covers three crucial aspects: the inertial sensors error modelling, inertial sensor output data de-noising and data fusion using Kalman filtering. Chosen adaptive filters (the extended Kalman filter and the unscented Kalman filter) were exploited for error estimation and inertial data fusion with legged odometry data. The actual data fusion was based on a novel concept of combining a complete full-state error model of the INS with a data-driven approach to legged odometry based on processing of self-motion cues. Verification of the navigation algorithm was performed on more than one hundred navigation experiments with precise trajectory reference ensured using video tracking. In the final, the proposed navigation system has proved to be sufficiently robust even to slippage and fully autonomous since it is independent of any external signals.

ACKNOWLEDGEMENT

First of all, I give my big thanks to my beloved Stanka, my mother and my uncle for supporting me on the long journey to unravel the mysteries of science and working on my doctoral thesis.

My great thanks also belong to my supervisor doc. Karel Draxler and supervisor specialist Jan Roháč Ph.D., for their patience and wisdom, as well as to everybody from the Department of Measurement, especially my colleagues from the Laboratory of Aircraft and Space Systems for supporting me and helping me in the friendliest manner all the years.

Special thanks go to Matěj Hoffmann, my fellow-scientist and a friend since childhood, who offered me the great opportunity to spend wonderful time working at the Artificial Intelligence Laboratory, University of Zurich, lead by Prof. Rolf Pfeifer. It was Matěj who introduced me to the branch of robotics and provided me with the robotic platform originally created by Prof. Fumiya Iida (ETH Zurich). Our cooperation showed me the beautiful synergy between the philosophy of bio-inspired robotics and the true science. I appreciate Matěj's invaluable advices and inspiring ingenuity during the development of the navigation system as well as when carrying out more than one hundred of experiments and processing all the measured data.

I have to thank Prof. M. S. Grewal (California State University, Fullerton) for giving me deep insight into the Kalman filtering during his amazing lectures in June 2007. Regarding Kalman filtering, I also thank to Miloš Soták Ph.D. for his kind attitude and willingness to answer any of my questions.

Last but not least, I would like to thank also Prof. Vladimír Haasz for reading my thesis and giving me valuable advice regarding the structure and the formal issues.

This work was partially supported by the research program no. MSM6840770015 "Research of Methods and Systems for Measurement of Physical Quantities and Measured Data Processing" of the CTU in Prague sponsored by the Ministry of Education, Youth and Sports of the Czech Republic, and partially by the Czech Science Foundation project 102/09/H082. The robotic platform used was part of the project "From locomotion to cognition", grant no. 200020-122279/1, supported by Swiss National Science Foundation

CONTENTS

ABSTRACT III
ACKNOWLEDGEMENT IV
CONTENTS V
LIST OF SYMBOLS VIII
LIST OF ABBREVIATIONS IX
1 INTRODUCTION 1
2 STATE OF THE ART 2

2.1 Inertial Navigation 2
2.2 Estimation Methods for Inertial Navigation 3
2.2.1 The Adaptive MMSE Filtering Approach 3
2.2.2 The Sampling-Based Approach 4
2.2.3 The Artificial Intelligence Approach 5

2.3 Dead Reckoning for Legged Robots 6
2.3.1 Legged Odometry 7
2.3.2 Quadruped Robotic Platform 8
2.3.3 Webots™ Simulation Environment 8

3 AIMS OF DISSERTATION 10
4 THEORY AND METHODOLOGY 11

4.1 Introduction to State Space Modelling 11
4.2 Inertial Sensors 13
4.2.1 Inertial Sensors Selection Criteria 13
4.2.2 Inertial Sensor Errors 14
4.2.3 Inertial Sensor Errors Analysis using Power Spectral Density 15
4.2.4 Inertial Sensor Errors Analysis using Allan Variance 20

4.3 Procedure for Sensor Output Data De-noising 26
4.3.1 Discrete Wavelet Transform 26
4.3.2 Wavelet Multi-Resolution Analysis 27
4.3.3 Wavelet Thresholding 29

4.4 Structure of the Inertial Navigation Systems 30
4.4.1 Attitude Representation 31
4.4.2 Coordinate Frames for INS 35
4.4.3 Initial Alignment Algorithms for INS 39
4.4.4 Strapdown Mechanization Scheme 41
4.4.5 Compensation of Deterministic Sensor Errors 42
4.4.6 Aiding Options and Integration Schemes for INS 44

4.5 Adaptive Filtering Methods for Inertial Navigation 48

4.5.1 The Linearized Kalman Filter 50
4.5.2 The Extended Kalman Filter 51
4.5.3 The Second Order Extended Kalman Filter 52
4.5.4 The Unscented Kalman Filter 53
4.5.5 Smoother for Kalman Filter 59

4.6 Introduction to Performance Analyses 60
4.7 Introduction to Suboptimal Modelling Techniques 61

5 SMC-INS REALIZATION 62

5.1 SMC-INS Integration Scheme 63
5.2 INS Mechanization Design 64
5.2.1 Numerical Integration 65
5.2.2 Velocity Update 66
5.2.3 Position Update 68
5.2.4 Attitude Update 70

5.3 SMC-INS Error Model Design 71
5.3.1 Perturbation Equations 71
5.3.2 Position Error Dynamics 72
5.3.3 Velocity Error Dynamics 73
5.3.4 Attitude Error Dynamics 73
5.3.5 Error Model Implementation 74
5.3.6 Measurement Model 75
5.3.7 Kalman Filter Tuning 77

5.4 SMC Mechanization Design 78
5.4.1 Gait Characteristics 78
5.4.2 Legged Odometer Development 79

5.5 SMC-INS Enhancement by Nonholonomic Constraints 85
5.6 SMC-INS Overview 86
5.6.1 Covariance Analysis 89
5.6.2 Observability Analysis 91

6 EXPERIMENTS RESULTS 92

6.1 Inertial Sensor Modelling and Analysis 92
6.2 De-noising of the Inertial Sensor Output Data 95
6.2.1 Results: WMRA Positioning Test 95
6.2.2 Results: WMRA Dynamic Field-test 97

6.3 Evaluation of the Adaptive Filtering Methods for INS 99
6.3.1 Results: Attitude Stability Evaluation 100
6.3.2 Results: Trajectory Evaluation by Simulation 102
6.3.3 Results: Trajectory Evaluation by Field-test 108

7 CONCLUSIONS AND RECOMMENDATION 115

7.1 Summary of the Achievements 115
7.2 Conclusions 116
7.3 Issues for Further Research 117

8 APPENDICES 118

8.1 3DM-GX2 Attitude and Heading Reference System 118
8.2 AHRS M3 Attitude and Heading Reference System 119
8.3 MTi-OEM and MT9 Xsens Attitude and Heading Reference Systems 120
8.4 System for Positioning Tests: Rotational Tilt Platform 121
8.5 MATLAB Codes 123

List of Tables 129
List of Figures 130
REFERENCES 133
RELATED WORK AND PUBLICATIONS 139

LIST OF SYMBOLS

$\boldsymbol{F}$	*State transition matrix for continuous time system*
$\boldsymbol{G}$	*Process noise coupling matrix*
$\boldsymbol{H}$	*Measurement sensitivity matrix*
$\boldsymbol{C}$	*Control matrix*
$\boldsymbol{K}$	*Kalman gain matrix*
$\boldsymbol{P}$	*Covariance matrix of state estimation uncertainty*
$\boldsymbol{Q}$	*Covariance matrix of process noise*
$\boldsymbol{R}$	*Covariance matrix of observational uncertainty*
$\boldsymbol{\Phi}$	*State transition matrix for discrete time system*
$\boldsymbol{\Phi}_k^{[1]}$	*Jacobian*
$\boldsymbol{\Phi}_{k-1}^{[2,i]}$	*Hessian*
$\boldsymbol{I}$	*Unit matrix*
$\boldsymbol{f}$	*Nonlinear system function*
$\boldsymbol{h}$	*Nonlinear measurement function*
$\boldsymbol{x}$	*System state vector*
$\boldsymbol{z}$	*Measurement vector*
$\boldsymbol{u}$	*Control input vector*
$\boldsymbol{x_k}$	*State vector (k^{th} step) of a discrete linear system*
$\hat{\boldsymbol{x}}$	*Estimated state vector*
$\hat{\boldsymbol{x}}_{\boldsymbol{k}}(-)$	*A priori estimate of x_k (measurement at time t_k not included)*
$\hat{\boldsymbol{x}}_{\boldsymbol{k}}(+)$	*A posteriori estimate of x_k (measurement at time t_k included)*
$\dot{\boldsymbol{x}}$	*Derivation of $\boldsymbol{x}$ with respect to time*
$\boldsymbol{w}$	*Process noise vector*
$\boldsymbol{v}$	*Measurement noise vector*
$\overline{\boldsymbol{x}}$	*Mean value of the state vector*
φ	*Latitude*
λ	*Longitude*
θ	*Pitch angle*
ρ	*Roll angle*
ψ	*Yaw angle*

LIST OF ABBREVIATIONS

ACF	*Autocorrelation Function*
AHRS	*Attitude and Heading Reference System*
AI	*Artificial Intelligence*
ANFIS	*Adaptive Neuro-Fuzzy INference System*
AR	*Autoregresive*
ARMA	*Autoregresive Moving Average*
ARW	*Angle Random Walk*
AVAR	*Allan Variance*
BF	*Body Frame*
CWT	*Continuous Wavelet Transform*
DCM	*Direction cosine matrix*
DGPS	*Differential GPS*
DWT	*Discrete Wavelet Transform*
ECEF	*Earth-centred-Earth-fixed frame*
ECI	*Earth-centred Inertial frame*
EKF	*Extended Kalman Filter*
FIR	*Finite Impulse Response*
GM	*Gauss Markov*
GPS	*General Positioning System*
IIR	*Infinite Impuls Response*
IMU	*Inertial Measurement Unit*
INS	*Inertial Navigation System*
KF	*Kalman Filter*
LD	*Levinson-Durbin*
LKF	*Linear Kalman Filter*
LMA	*Levenberg-Marquardt Algorithm*
MEMS	*Microelectromechanical Systems*
MMSE	*Minimum Mean-Square Error*
NED	*North-East-Down navigation frame*
PSD	*Power Spectral Density*
RMSE	*Root-Mean Square Error*
RRW	*Rate Random Walk*
RTS	*Rauch-Tung-Striebel*
SEKF	*Second order EKF*
SINS	*Stand-alone INS*
SLAM	*Simultaneous Localization And Mapping*
SMC	*Self-motion Cues*
SP	*Sigma Point*
UKF	*Unscented Kalman Filter*
UKS	*Unscented Kalman smoother*
UT	*Unscented Transformation*
WMRA	*Wavelet Multi-Resolution Analysis*
Y-W	*Yule-Walker*

1 INTRODUCTION

Rapid advances in the field of inertial sensors, which improve their precision, cost-effectiveness and availability, make the inertial sensors increasingly more popular. Development of signal processing methods aimed at enhancing their performance has to reflect this advancement. With the recent proliferation of low-cost inertial sensors based on Microelectromechanical systems (MEMS) technology it has become viable to construct inexpensive strapdown navigation systems of any kind. Therefore, the inertial sensors can be employed in many areas of research and industrial applications:

1) *Terrestrial, aerospace, submarine and space navigation*: data fusion of inertial sensors, as concluded in [1], for attitude [2] and displacement measurements [3] with appropriate aiding, such as GPS [4], [5], [6] magnetometer or more complex azimuth-level detector [7], IR optical sensors, airborne lasers range scanners [8] and finders [9], odometers based on wheel encoders [10], ultrasonic sensors [11], vision and optic flow sensors such as appearance based visual compasses [12] etc.
2) *Robotics*: use of inertial sensors to control locomotion [13], provide information regarding orientation in space [14], processing of self-motion cues that leads to path integration and tracking [15], and the Simultaneous Localization And Mapping (SLAM) where environmental map is created [16].
3) *Industry*: platform stabilization and stability issues in general.
4) *Monitoring and safety systems*: vibration measurements for diagnostics and fault detection purposes.
5) *Smartphones*: user interface enhancement and gaming.
6) *Biology*: prosthetics development and motion sensing.
7) *Automotive systems*: airbag systems [17], land vehicle navigation [18], [19], usually enhanced with nonholonomic constraints [20].
8) *Movie and gaming industry*: motion-capture technology for special effects [21].
9) *Intelligent buildings*: in-home localization regarding assistive technologies for people with cognitive disability [22].

One of the most popular areas of research regarding inertial navigation is the dead reckoning for autonomous aerial and robotic platforms. Although robots may vary almost indefinitely according to the fantasy of their makers, reliability and precision of navigation is a common issue for all of them. Nowadays, the legged robots especially come to foreground and hence resolving the autonomous navigation using appropriate sensor suit and signal processing methods is a hot topic. Finding a low-cost, reliable, and sufficiently robust solution is a challenge, but the inertial sensors offer the opportunity to beat it in many possible ways.

Therefore, this dissertation focuses on the autonomous navigation for legged robots where inertial sensors will be exploited for sensing the motion dynamics and combined with additional motion-cues sensors. Adaptive filtering methods will be used for the data fusion and estimation of the robot's orientation in space and distance travelled.

2 STATE OF THE ART

This dissertation explores the use of adaptive filtering methods in the inertial navigation systems (INS) based on traditional concepts described in [23], [24], [25], [26], and [27]. This chapter covers the state of the art regarding the principles of inertial navigation, the most common estimation methods for inertial navigation, and the dead reckoning for legged robots.

2.1 INERTIAL NAVIGATION

The operation of an inertial navigation system follows the laws of classical mechanics as formulated by Newton. With ability to measure specific force using an accelerometer it is possible to calculate a change in velocity and position by performing successive integration of the acceleration with respect to time. A conventional inertial navigation system usually contains three accelerometers and three angular rate sensors, placed perpendicularly to each other in order to create an orthogonal measurement frame as described by Titterton [23 pp. 25-29]. Therefore, each accelerometer is able to detect the specific force that is defined as the time rate of change of velocity relative to local gravitational field [28 p. 2]. To navigate with respect to Earth frame, it is important to track directions in which the accelerometers are pointing. Rotational motion of the navigated object can be sensed by angular rate sensors or gyroscopes. These sensors are used to determine the orientation of the accelerometers at any moment. Given this information, it is possible to project the accelerations into the reference frame before the integration process takes place.

This principle is used in inertial measurement units (IMU), comprising of usually 3 accelerometers and 3 angular rate sensors (or gyros), which provide good high-frequency information but poor low-frequency data due to long term system drift and sensor errors [29]. The INS that uses such IMU then estimates the velocity, position and attitude values by integrating and double integrating the acceleration data, respectively, which results in the unbounded estimation error growth over time [24], [30]. This flaw can be overcome by periodical correction feedback to the IMU based on information obtained from other sensors generally named as an aiding [26]. The most conventional procedure used for fusion of inertial measurements with aiding measurements is an adaptive filtering approach called the Kalman filter (KF) [31]. The KF has been widely used in INS state estimation in many variations of the original algorithm [32], [33], [34], [35]. It combines all available measurements with prior knowledge about the sensing device and the system to produce an optimal state estimation that statistically minimizes the error [26]. Since in the real life, the exact model of a system and measurements are not available, and furthermore, the statistical characteristics of the process and measurement noise are difficult to determine, the major task, when developing a navigation algorithm, is to seek the best type of the estimation methods available.

2.2 ESTIMATION METHODS FOR INERTIAL NAVIGATION

In general, *adaptive filtering* or *adaptive signal processing* concerns the design of time-varying (adaptive) digital filters that would tune themselves to optimally process non-stationary signals in non-stationary environments [36 p. 169]. Recently, there exist a number of tutorials and overviews regarding adaptive filtering available on the internet. The summary by Hayes [36], presented as the "best of web", provides brief conclusion and evaluation. Regarding the INS, as described by Shin [32 p. 2], there are three different approaches to the research of adaptive filtering and estimation methods; these methods can be categorized as follows:

1. The minimum mean-square error (MMSE) based methods, such as the **linearized Kalman filter** (LKF) or the **extended Kalman filter** (EKF); both being the modification of the conventional Kalman filtering algorithm enhanced for the nonlinear system application, as described by Grewal and Andrews [26].
2. The sampling-based methods, such as the **unscented Kalman filter** (UKF) as described by Shin [32] and **particle filters** as described by Bergman [33].
3. The artificial intelligence based methods (AI), such as the artificial neural networks or the **Adaptive Neuro-Fuzzy Inference Systems** (ANFIS) as described by Abdel-Hamid [34].

All of the three approaches are built on the principle of Kalman filtering and the KF equations can be found in each approach with slight modifications. In general the KF has the following properties, which has to be taken into account [26] (+ merit, - flaw, ± ambiguous):

(±) The KF is "optimal" if the mathematical model reflects the real world behaviour. Filter precision and performance are hence directly proportional to the filter optimality.

(+) The KF is recursive, there is no need for data storing during filtering procedure.

(±) The KF develops its own covariance analysis automatically; however it is only as precise as the model itself.

(–) The KF algorithm is straightforward in description; however its implementation requires much of an insight regarding filter numerical stability, memory requirements and computational complexity.

(+) The KF is easy to modify due to its model based conception. Model changes can be easily implemented and validated.

(±) The KF is always linear; modifications to the KF algorithm are required for nonlinear systems.

(±) The KF is based on matrix calculus; hence, mathematical operations and KF equations are independent on model complexity.

2.2.1 THE ADAPTIVE MMSE FILTERING APPROACH

For long time, linearized/extended KF has been playing a key role in navigation software design. In principle, the extended Kalman filter simply applies the Taylor series expansion on the nonlinear system equations and takes the first order terms, where the state probability density function is approximated by a Gaussian distribution. Such a distribution can completely be parameterized by its mean and covariance. Although this approach provides suitable solution to cope to some extent with nonlinearities, several significant drawbacks arise in practice [37], [38]:

1. The derivation of the Jacobian matrices for both the system and observation equations is complicated and leads to significant limitation in the actual implementation.
2. The EKF is strictly dependent on the actual error model used. The choice of an appropriate error model for the specific application is a crucial issue since only small errors are allowed to be delivered to the EKF. Because of the first order approximations, large error values can cause biased solutions and inconsistency of the covariance update [32]. In other words, "*Linearization can produce highly unstable filters if the assumptions of local linearity are violated"* [38 p. 2].

A way to cope with the EKF drawbacks was sought by Nasser [39] by increasing the order of approximation. Higher order approximation approach was tested, evaluated and it proved its suitability. However, for some complex error models the computational cost proved to be unbearably high as well. Nowadays, the EKF approach is the most widely used one [26] and for carefully chosen suboptimal models it provides reliable solution. Detailed description of the KF can be found in [26], the higher order approximation approach, e.g. the Second order KF (SEKF), in [39].

2.2.2 THE SAMPLING-BASED APPROACH

When considering the sampling-based methods, two of them come to foreground; both providing solution without the necessity of computing derivations, as the EKF requires.

Firstly, it is the group of Sequential Monte Carlo filters, known as the particle filters, which have been originally developed for nonlinear processes based on Bayesian filtering theory. On the eve of the 20th century these methods were used mainly for practical problems such as in econometrics, computer vision, or target tracking for radar application. However, they were abandoned due to the lack of computing power for the real-time implementation. With the rapid development in the field of embedded systems and signal processors, particle filters were evaluated for terrain navigation purposes by Bergman in 1999 [33]. Bergman has proved, that the particle filters provide numerical accuracy superior to other filtering methods since they do not make any assumptions on the probability distribution function. Still, when high-dimensional systems with a high update rate are considered, implementation of particle filters is almost impossible unless parallel processing is employed. That is due to generation of enormous number of particles (samples) and their further transformations. For more details on the Monte Carlo methods for signal processing see the thorough overview by Doucet [40].

Secondly, it is the unscented Kalman filtering method introduced by Julier and Uhlmann in 1996 [37], [38], that has the same advantage over the EKF as the particle filters do. It approximates the state distribution by a Gaussian random variable rather than trying to approximate an arbitrary nonlinear function or transformation by a Taylor series. On the contrary to the particle filters, the Gaussian random variable is specified using a minimal set of carefully chosen sample points, called sigma points (SPs), which capture the true mean and covariance of the probability distribution function. If these SPs are propagated through the true nonlinear system, by means of unscented transformation, the obtained transformed mean and covariance are accurate up to a second order for any nonlinearity. Such UKF should provide more accurate nonlinearity approximation, compared to the EKF, as well as bearable computational load due to the significantly reduced number of deterministically chosen samples, compared to the particle filters. This conclusion is presented in [32], [37], and [38].

Obviously a question that arises in the first place is the performance comparison of the two most suitable estimating methods, the EKF and UKF, regarding the inertial navigation systems. An in depth analysis was carried out regarding the UKF by Shin [32] in his dissertation. Further relevant analyses of the EKF and UKF performance comparisons can be found in [41], [42], and [43], where similar conclusions are presented as those derived by Shin. In [42 p. 2] the following flaws of EKF are highlighted: difficulties of determining the Jacobians, errors introduced by linearization and the ability to deal with systems with multimodal or asymmetric probability density functions. Conclusions presented by Shin [32 pp. 160 -164] will be the key for describing the state of art regarding the UKF and INS. Shin contributed the following new developments to the field of inertial navigation [32 p. 161]:

1. A general system model was developed for a quaternion based UKF for an aided inertial navigation system [32 pp. 86 -130].
2. When comparing the EKF and UKF it was proven that the UKF converges faster than the EKF when initial attitude uncertainties are large; hence, the UKF can recover faster than the EKF during GPS outages or aiding signal outages in general [32 pp. 140 - 155].
3. Quaternion-based backward UKF was developed - the unscented Kalman smoother (UKS) [32 pp. 126 - 130] - and it was proven that its performance is as good as of the classical Rauch-Tung-Striebel smoother algorithm for the EKF.

2.2.3 THE ARTIFICIAL INTELLIGENCE APPROACH

When comparing the above mentioned approaches, the conventional LKF/EKF estimation method and the sampling based methods have much more in common than the artificial intelligence based approach. Whereas the nonlinear system function and the state distribution function are approximated by a mathematical model in both the EKF and the UKF, the AI based methods do not use any mathematical model of system dynamics or measurements at all. AI based methods make advantage of relatively straightforward design procedures, indifferent to target application. The actual estimation process is then rather defined by the quality of the learning process of the neural networks and their real-time adaptability. As described by Shin [32] AI based methods have two serious drawbacks:

1. AI based methods, on the contrary to the covariance analysis of the KF, do not use any statistical information as input and hence they do not provide any. Such statistical information is crucial for post-processing in order to obtain the best estimate. Both the EKF and the UKF operating as “smoother” outperforms the AI based methods in this way.
2. Mathematical models of vehicle dynamics and sensor error models are generally well known. However, in AI based methods these models cannot be used since the system model is created by neural network trained on given data sets. Hence, the AI based methods are critically sensitive to the character of the training data set. If such training data set does not cover the whole spectrum of the possible system dynamics, estimated outputs may be heavily biased. On-line training is required; this increases the hardware cost and computational load.

The only case, when AI based methods do actually provide superior results, is during the long term outage of the measurement updates. Inertial navigation systems are multi-sensor systems with opened options for various aiding and data fusion to compensate such measurement outages.

For vehicle applications, GPS signal may be supplemented by odometers, for UAV applications precise triaxial magnetometer aiding is possible solution. More details regarding comparison of AI based estimation methods and a conventional KF can be found in [34]; the major points are concluded in the following table [34 p. 190]:

Point of Comparison	AI Approach	Conventional KF
Pre-knowledge of the system	Huge training dataset required	System dynamics model required
Appropriate system model	No model required	Model optimality is essential
Short-term accuracy	No guarantee to provide better estimates than KF	Usually superior short term accuracy (~5-10s)
Long-term accuracy	No accuracy degradation over time	Time correlated degradation in precision
Measurement updates	Not required	Required to avoid accuracy degradation
Statistical analysis	Unavailable – no error model to predict error propagation	Errors can be propagated and covariance matrix is estimated
Range of input data	Restricted to training data ranges	Not restricted

Table 2-1 Comparison of AI-based approach (ANFIS) to the conventional KF [34 p. 190]

2.3 DEAD RECKONING FOR LEGGED ROBOTS

This dissertation is partially connected to the research project called *From Locomotion to Cognition*[1], which concerns primarily the principles of development of artificial intelligence (AI) with respect to the embodiment of an agent, usually a robot. The main idea behind this project is described by Pfeifer in [44]. The whole concept is described and analysed in [45] and updated according to the recent scientific achievements and research results in [46]. A quotation regarding this project from a review article by Pfeifer [44] is in place:

"Robotics researchers increasingly agree that ideas from biology and self-organization can strongly benefit the design of autonomous robots. Biological organisms have evolved to perform and survive in a world characterized by rapid changes, high uncertainty, indefinite richness, and limited availability of information. Industrial robots, in contrast, operate in highly controlled environments with no or very little uncertainty. Although many challenges remain, concepts from biologically inspired (bio-inspired) robotics will eventually enable researchers to engineer machines for the real world that possess at least some of the desirable properties of biological organisms, such as adaptivity, robustness, versatility, and agility". [44 p. 1088]

The self-organization and embodiment means the reciprocal and dynamical coupling among brain (control), body, and environment [44 p. 1088]. To study and develop a real artificial intelligence, a robot with appropriate physical body has to be designed in the first place. The goal of the project is to work out the principles of biological systems and transfer them to robot design in order to create effective embodiment based on clever morphology and use of material properties

[1] Grant Nr. 200020-122279/1, supported by Swiss National Science Foundation, University of Zurich

[44 p. 1088]. If it is properly applied, embodiment can lead to surprising insights and finally, to creation of autonomous adaptive systems.

2.3.1 LEGGED ODOMETRY

An important question arises: how is the project *From Locomotion to Cognition* related to the adaptive filtering methods in inertial navigation? When studying the actual locomotion and embodiment, especially in mammals, interesting results were discovered regarding navigation. It is the process of navigation that is the desired outcome of locomotion; quoting a review article by Etienne [15 p. 180]: *"It is often assumed that navigation implies the use, by animals, of landmarks indicating the location of the goal. However, many animals (including humans) are able to return to the starting point of a journey, or to other goal sites, by relying on self-motion cues only. This process is known as path integration, and it allows an agent to calculate a route without making use of landmarks."* In other words, Etienne proposes that instead of remembering landmarks for navigation, the agent, i.e. the navigated object, estimates its position in a continuous manner with respect to a reference point by relying on signals that it derives from its own locomotion [15 p. 180]. This process of estimation is called *path integration* and is defined as: *a process that takes place by the addition of successive small increments of movement onto a continually updated representation of direction and distance from the starting point* [15 p. 180]. Path integration relies only on the locomotion and internal signals that are called, in case of mammals, *self-motion cues* (SMC). These cues are derived from several sensory sources, including visual (linear and radial optic flow), vestibular (translational and rotational accelerations from extra-vestibular gravity receptors), and proprioceptive (feedback information from muscles, tendons, and joints) [15 p. 180]. Navigation based on SMC targets especially the bio-inspired systems, for example see [13], [47], [48], [49], and [50], which can imitate the motion of insects and legged animals to achieve high mobility on rough terrain, even without using vision [51].

The whole concept of path integration in mammals can be considered equivalent to the *dead reckoning*, i.e. the navigation based on data fusion of inertial sensors with odometer-based sensors (called also *leg pose system* [52], [53], [54]), or with sensors providing optic-flow information and area mapping. This data fusion inevitably leads to state estimation and adaptive filtering [55] such as Kalman filtering. In robotics, a standard solution to navigation and dead reckoning, that concerns wheeled robots, is based usually on INS, GPS for absolute position, magnetometer for heading, and wheel encoders based odometry for speed determination. Multiple-sensor data fusion for navigation purposes in legged robots, e.g. walking humanoid robots [56] and quadruped robots [57], is recently a very popular area of research and yet it is not so well explored [14 p. 753]. According to [50] the KF based estimators are considered currently the best option. In [50] two dimensional navigation is addressed and comparison of different estimators is presented. To my knowledge, the navigation problem solution without external reference aiding in a legged robot has been addressed only by Pei-Chun Lin et al. in [52], [53], where strain-based leg sensor was used to estimate the kinematic configuration of the legs of a hexapod robot. From the configuration of the legs, the pose of the whole robot was estimated; however, turning was not addressed and the system proved to be sensitive to slipping.

2.3.2 QUADRUPED ROBOTIC PLATFORM

In order to provide real data a quadruped robotic platform named ALAN[2] was chosen as a mobile platform for dynamic navigation field-tests. Due to its embedded multi-sensor system and its morphology it became a perfect candidate for exploring the use of adaptive filtering in the inertial navigation systems. The multi-sensor system (see **Figure 2-1**) provided desired signals for INS aiding and data fusion. The variability of all the possible motion gaits provided unique testing conditions due to the wide range of possible dynamics.

ALAN was developed at the AI Laboratory at University of Zurich by Iida (for more details see [58]). It is driven by two pairs of servo-motors to control active hip and shoulder joints; knee joints are passive with springs attached. The springs add compliance to the system which is capable of dynamic running with minimalistic open-loop control architecture, also due to self-stabilizing properties. In principle, ALAN was built according to a spring–mass model, which is widely used in biomechanics. It is based on the hypothesis that animal's leg can easily be approximated by a spring-loaded inverted pendulum. [58].

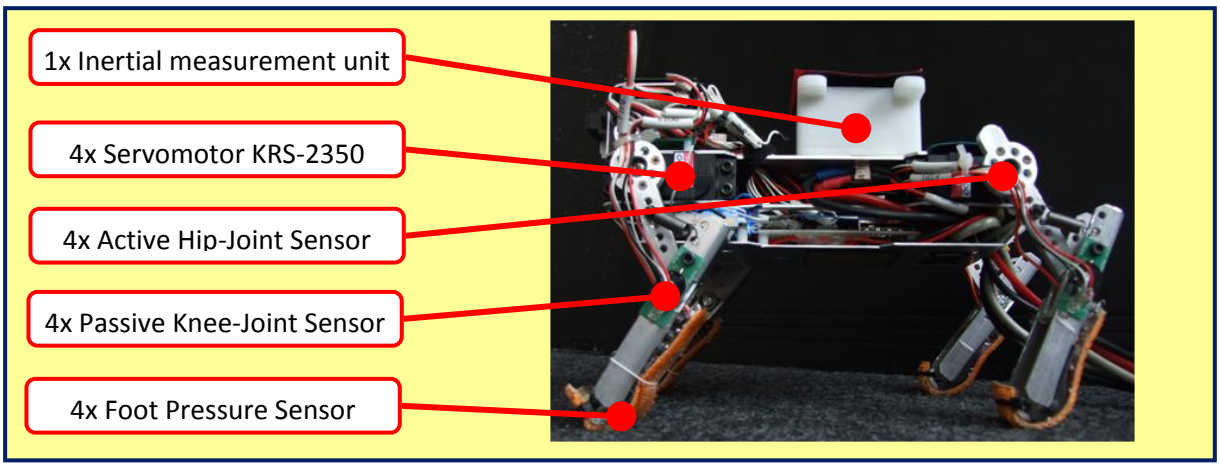


Figure 2-1 Platform for dynamic tests: quadruped robot
(platform developed by F. Iida and M. Hoffmann, University of Zurich, AI Laboratory)

2.3.3 WEBOTS™ SIMULATION ENVIRONMENT

Webots™ is a development environment usable for modeling, programming and simulating of mobile robots of any kind; for details see Webots Reference Manual [59]. Complex robotic setups can be designed in Webots™ even with a number of different robots sharing the same environment under variable physical conditions. The simulation includes sensors and actuators operating in the robot's body frame as well as robot controllers that can be fully programmed. Due to the physically realistic worlds all the developed controllers can be transferred to the commercially available robots as well. Webots™ is used by over 650 universities and research centers worldwide [59]. The technology used has been co-developed by the Swiss Federal Institute of Technology in Lausanne, thoroughly tested, well documented and continuously maintained for over 10 years [59]. The key features of Webots™ are [59]:

[2] Platform developed by Fumiya Iida and Matej Hoffmann, University of Zurich, AI Laboratory

- Complete library of sensors and actuators,
- Built-in 3D world and robot editor with VRML import capability,
- Robot controller programming languages: C/C++, Java, Python, URBI™, MATLAB™ or interface with third party software through TCP/IP,
- Support of multiple platforms: Windows, Mac OS X and Linux,
- ODE (Open Dynamics Engine) library for accurate physics simulation,
- Transfer of controllers to real mobile robots: e-puck™, Nao™, Katana™, Hoap-2™, etc.,
- Creation of AVI or MPEG simulation movies for the web or for public presentations.

As shown in **Figure 2-2** the Webots™ environment was used to model the real robot ALAN in order to evaluate its gaits – simulated ALAN named SIMALAN was produced this way. The simulation was used to emulate the inertial sensors and then to tune and evaluate the algorithms developed in this dissertation. Although limited by the approximations of the real world, this simulation proved to be still useful due to precise reference of navigation data provided by the supervisor controller and due to fully controllable environment including sensor noise parameters.

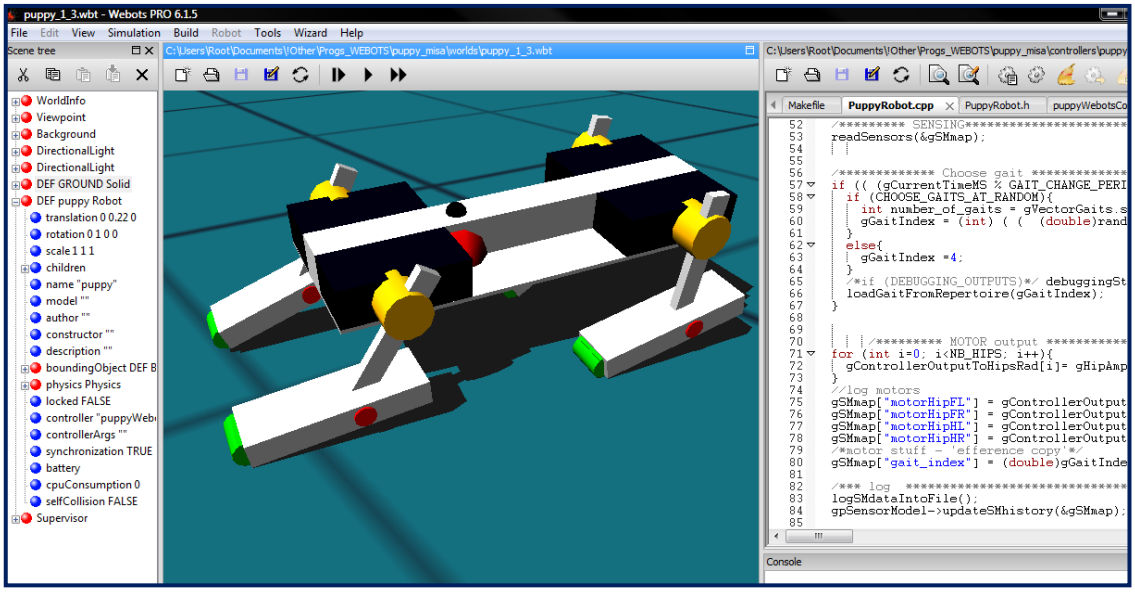


Figure 2-2 Webots 6.1.5 modelling environment: simulated model of robot ALAN, the SIMALAN (model developed by M. Hoffmann, University of Zurich, AI Laboratory)

3 AIMS OF DISSERTATION

Based on the conclusions drawn from the current state of the art the adaptive filtering approach, especially the EKF and the UKF algorithms, were chosen to be the subject of research among all the estimation methods for inertial navigation.

The main aim was to evaluate the usability of these adaptive filtering methods in the INS, such that a complete navigation algorithm was developed in a way to enable its implementation for terrestrial and possibly for aerial navigation. The development process included three crucial aspects: *the inertial sensors error modelling*, *inertial sensor output data de-noising* and *data fusion using Kalman filtering*. Verification of the developed navigation algorithm was performed both by simulation for long-term data and by using dynamic robotic platform for real navigation data with precise reference obtained by video tracking. The detailed structure of the partial aims of this dissertation is hence as follows:

1. **Inertial sensors modelling and analysis**
   Use the available navigation units 3DM-GX2 (MicroStrain, Appendix 8.1), AHRS M3 (Innalabs, Appendix 8.2), MTi-OEM and MT9 (both Xsens, Appendix 8.3) and their embedded inertial sensors to identify, analyse and evaluate the most crucial inertial sensors errors.
2. **De-noising of the inertial sensor output data**
   Investigate and evaluate a suitable procedure for optimal inertial sensor data de-noising.
3. **Evaluation of adaptive filtering methods for inertial navigation**
   Develop, implement and evaluate fully autonomous self-motion cues based inertial navigation system (SMC-INS) that is independent of any external signals. The SMC-INS has to allow comparison and evaluation of both the EKF and the UKF algorithms. This includes the following:
   - compensation of sensor errors identified by calibration,
   - implementation of an initial alignment procedure,
   - development of INS mechanization algorithm for conversion of raw inertial data into position, velocity, and attitude angles,
   - proposal of an integration scheme for desired data fusion,
   - development of legged odometer (SMC mechanization algorithm),
   - development of an error model for data fusion from SMC mechanization and INS mechanization algorithms,
   - development of nonholonomic constraints according to covariance and observability analysis of the proposed error model,
   - evaluation of the SMC-INS regarding attitude stability, precision of position estimation, and robustness to slippage on both the real data using ALAN platform and using SIMALAN simulation in the Webots 6.5,
   - creation of an INS Toolbox for MATLAB with all the developed algorithms available for further research.

4 THEORY AND METHODOLOGY

This chapter concludes all the important theoretical and methodological background and is structured in such a way to cover the aims of this dissertation entirely. In order to provide theory necessary for SMC-INS design, the classical INS/GPS concept is discussed throughout the thesis to provide common basis and analogy; similarities and differences to the proposed SMC-INS are then highlighted in the next chapter, which deals with the implementation issues. All the algorithms and implementation details are presented with emphasis on MATLAB development environment, which was used extensively.

4.1 INTRODUCTION TO STATE SPACE MODELLING

One of the most common approaches to mathematical modelling of systems and their behaviour is based on a state space modelling. This approach is described in [26 pp. 77 -78] and will be adopted throughout the whole dissertation. Following state space description is assumed:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}\boldsymbol{x}(t) + \boldsymbol{G}\boldsymbol{w}(t) + \boldsymbol{C}\boldsymbol{u}(t), \boldsymbol{w}(t) \sim N(0, \boldsymbol{Q}_t) \tag{4-1}$$

$$\boldsymbol{z}(t) = \boldsymbol{H}\boldsymbol{x}(t) + \boldsymbol{v}(t), \boldsymbol{v}(t) \sim N(0, \boldsymbol{R}_t) \tag{4-2}$$

where:

- $\boldsymbol{F}$ *State transition matrix for continuous time system*
- $\boldsymbol{G}$ *Process noise coupling matrix*
- $\boldsymbol{H}$ *Measurement sensitivity matrix*
- $\boldsymbol{C}$ *Control matrix*
- $\boldsymbol{Q}$ *Covariance matrix of process noise*
- $\boldsymbol{R}$ *Covariance matrix of observational uncertainty*
- $\boldsymbol{\Phi}$ *Transition matrix for discrete time system (the discrete counterpart of* $\mathbf{F}$*)*
- $\boldsymbol{x}$ *System state vector*
- $\boldsymbol{z}$ *Measurement vector*
- $\boldsymbol{u}$ *Control input vector*
- $\boldsymbol{w}$ *Zero-mean uncorrelated "plant noise" process*
- $\boldsymbol{v}$ *Zero-mean uncorrelated "measurement noise" process*

Matrices $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{C}$, and $\boldsymbol{H}$ are assumed variable in time; n is the number of states.

In order to convert this continuous time-domain description to the desired discrete time-domain, simplified discretization (without proof) is as follows [60 p. 28]:

$$\boldsymbol{\Phi}_k = \exp(\boldsymbol{F}\Delta t) \approx \boldsymbol{I} + \boldsymbol{F}\Delta t \tag{4-3}$$

$$\boldsymbol{Q}_k = \mathrm{E}\left(\boldsymbol{w}_k \boldsymbol{w}_k^T\right) \approx \boldsymbol{\Phi}_k \boldsymbol{G} \boldsymbol{Q}_t \boldsymbol{G}_k^T \boldsymbol{\Phi}_k^T \Delta t \tag{4-4}$$

where $\boldsymbol{I}$ is the unit matrix of appropriate size, $\mathrm{E}(\cdot)$ is the expected mean value operator.

The implementation of the above approach is called the Van Loan discretization method (for MATLAB implementation see **Matlab 8-1** in Appendix 8.5). It is proposed as follows:

1. Consider a simplified state space description of a system:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}\boldsymbol{x} + \boldsymbol{G}\boldsymbol{w}(t), \boldsymbol{w}(t) \sim N(0, \boldsymbol{Q}_t) \tag{4-5}$$

2. Create the matrix **M** of size [2n×2n] in such a way where Δt is the sampling period:

$$\boldsymbol{M} = \begin{bmatrix} -\boldsymbol{F} & \boldsymbol{G}\boldsymbol{Q}(t)\boldsymbol{G}^T \\ 0 & \boldsymbol{F}^T \end{bmatrix} \Delta t \tag{4-6}$$

3. Use the function *expm* implemented in MATLAB obtain the matrix exponential **N** such that:

$$\boldsymbol{N} = \text{expm}(\boldsymbol{M}) = \begin{bmatrix} - - - & \boldsymbol{\Phi}^{-1}\boldsymbol{Q}_k \\ 0 & \boldsymbol{\Phi}^T \end{bmatrix} \tag{4-7}$$

4. By the transposition of the right lower part of the matrix $\boldsymbol{N}$ the discrete system transition matrix $\boldsymbol{\Phi}_k$ is to be obtained.
5. By left multiplication of the upper right part of the matrix $\boldsymbol{N}$ by the $\boldsymbol{\Phi}_k$ the $\boldsymbol{Q}_k$matrix is to be obtained.
6. Using $\boldsymbol{Q}_k$ and $\boldsymbol{\Phi}_k$ the discrete-time system description is defined for $t_k = k \cdot \Delta t$ as follows:

$$\boldsymbol{x}_k = \boldsymbol{\Phi}_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1}, \boldsymbol{w}_k \sim N(0, \boldsymbol{Q}_k) \tag{4-8}$$

Such a system description has to include the measurement equation that binds the actual measured data and the inner states of the system:

$$\boldsymbol{z}_k = \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{v}_k, \boldsymbol{v}_k \sim N(0, \boldsymbol{R}_k) \tag{4-9}$$

Assume a state space model of a system is to be expanded by another state space model of a system called shaping filter given by a state vector $\boldsymbol{x}_{SF}(t)$. Let $\boldsymbol{v}_2(t)$ be zero-mean white Gaussian noise and let the measurement noise be denoted $\boldsymbol{v}_1(t)$; such that the shaping filter is modelled by:

$$\dot{\boldsymbol{x}}_{SF}(t) = \boldsymbol{F}_{SF}(t)\boldsymbol{x}_{SF}(t) + \boldsymbol{G}_{SF}(t)\boldsymbol{v}_2(t) \tag{4-10}$$

$$\boldsymbol{v}_1(t) = \boldsymbol{H}_{SF}(t)\boldsymbol{x}_{SF}(t) \tag{4-11}$$

where index *SF* denotes the matrices for the state space description of the shaping filter.

Since both systems are driven by white Gaussian noises, the shaping filter can be augmented into the original state space description, as follows [26]:

$$\begin{bmatrix} \dot{\boldsymbol{x}}(t) \\ \dot{\boldsymbol{x}}_{SF}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}(t) & 0 \\ 0 & \boldsymbol{F}_{SF}(t) \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{x}_{SF}(t) \end{bmatrix} + \begin{bmatrix} \boldsymbol{G}(t) & 0 \\ 0 & \boldsymbol{G}_{SF}(t) \end{bmatrix} \begin{bmatrix} \boldsymbol{w}(t) \\ \boldsymbol{v}_2(t) \end{bmatrix} \tag{4-12}$$

$$\boldsymbol{z}(t) = [\boldsymbol{H}(t) \quad \boldsymbol{H}_{SF}(t)] \begin{bmatrix} \boldsymbol{x}(t) \\ \boldsymbol{x}_{SF}(t) \end{bmatrix} \tag{4-13}$$

This process of augmentation can be used formally to incorporate state space models of any random processes introduced in the next chapter into the state space description of a system.

4.2 INERTIAL SENSORS

Inertial navigation systems (INS) contain gyros (meaning both the angular rate sensors and gyroscopes) to measure inertial angular rotation [61], i.e. changes in angular orientation, and accelerometers to measure changes in velocity (acceleration) of the navigated object. More precisely stated gyros actually measure angular rate relative to non-rotating inertial space. Angular rate relative to the Earth (or relative to any rotating body) is then calculated as the gyro output minus the inertial rotation rate of the Earth or the body [62]. A similar concept can be observed when considering the accelerometers. With the input axis pointed downwards, the accelerometer output would be minus 1 g. Observations of the accelerometers output behaviour in the Earth's gravitational field have led to the conclusion that an accelerometer does not measure gravity; it actually measures the *component of total acceleration minus gravity* along its input axis [28], or in other words *the specific force*, which is the time rate of change of velocity relative to local gravitational space. Local gravitational space is then defined as the same space occupied by the accelerometer but which contains a virtual mass with zero applied specific force [28]. The specific force is hence a basic absolute quantity that is measured directly by the accelerometer. Gravity, on the other hand, is a property of space whose value at a particular location can only be determined relative to the value of gravity at another location [62].

4.2.1 INERTIAL SENSORS SELECTION CRITERIA

According to Grewal and Andrews [26], the KF theory assumes that the type, number and noise characteristics of sensors are defined statistically in advance and the noisy information from all sensors is combined optimally. Based on this theory, there are some generalities regarding the sensor selection that can be stated:

- The more sensors used and the lower the noise-level, the smaller is the estimation error.
- The frequency response of the sensor should be compatible with the corresponding signal represented by a state variable.
- The sensor noise spectrum should be separable from the system state-spectrum if possible.
- Sensors should be chosen to cause observability of the system, i.e. system states should be determinable from measurements.
- The physical quantity measured by each sensor should match the corresponding state-variable.
- Differentiation of sensor output, which is corrupted by noise, should be avoided, since it causes a large increase in the estimation error.
- KF covariance analysis and observability analysis should be carried out for each combination of sensors to determine the overall "cost" (economic, power requirements, degradation, reliability, and weight), and to optimize KF performance.

4.2.2 INERTIAL SENSOR ERRORS

The overall performance of INS is determined by the errors of individual inertial sensors involved in the system. The possible inertial sensor errors according to [34 p. 15] are as follows:

- **Drift:** A sensor *drift* (or *drift rate* for angular rate sensors) is a result of manufacturing imperfections of the sensors. Sensor drift has a systematic component, called *bias*, which can be quantified by calibration, and a stochastic component, which can be approximated as random process and modelled in that way. Bias is measured over a specified time and has no correlation to the input quantity. Drift is functionally independent to the input as well and refers to the rate at which the error of an inertial sensor accumulates with time. Complete structure of drift / drift rate components according to IEEE Std. 528-2001 [63 p. 5] is concluded in **Figure 4-1**.
- **Scale Factor:** Scale factor is the ratio of a change in output to a change in input intended to be measured. It is generally evaluated as the slope of the straight line that can be fitted by the method of least squares to input-output data [63 p. 17]. The scale factor depends on the system dynamics and the temperature variation. Under normal operational conditions and low dynamics it remains constant. Scale factor errors have to be compensated by calibration.
- **Output Stability:** The output stability of a sensor is defined by the run-to-run variation of the sensor drift and bias, i.e. it is a measure of the ability of a specific mechanism or performance coefficient to remain invariant when continuously exposed to fixed operating condition [63 p. 18]. The run-to-run stability can be evaluated from the scatter in the mean output value for each run out of given number of runs.
- **Thermal Sensitivity:** Thermal sensitivity refers to range of variation of the sensor bias and scale factor errors with the change in temperature. Nature of thermal sensitivity is measured and expressed in the form of compensation tables. Modern sensors, such as ADIS16355, have a built-in thermal compensation.

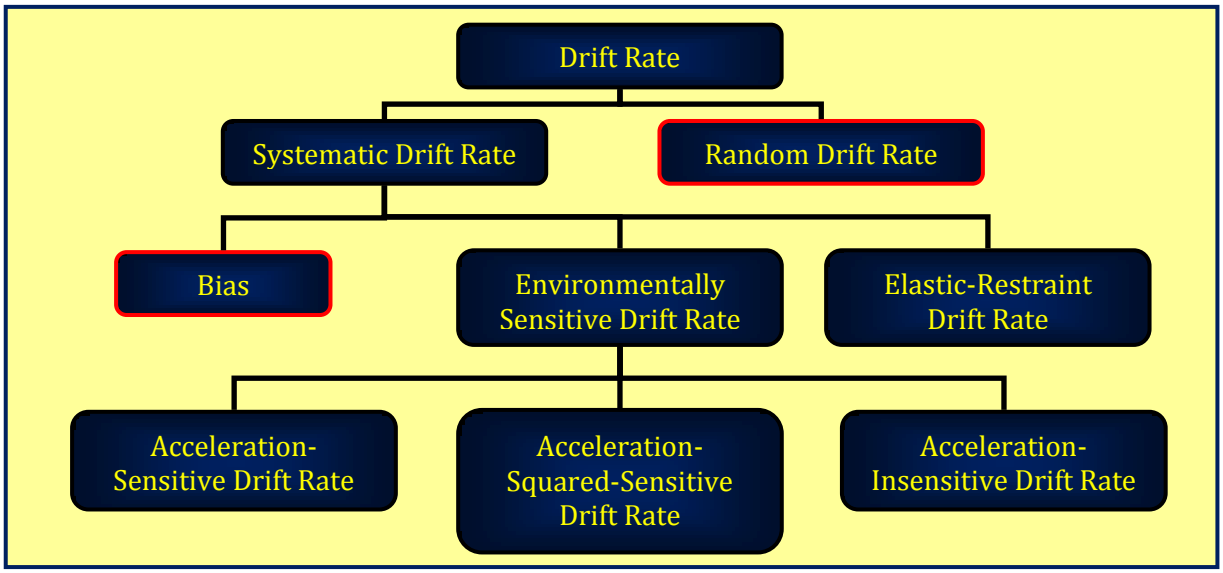


Figure 4-1 Components of gyro drift rate according to IEEE Std 528-2001 [63 p. 5]

- **Sensing axis misalignment**: Axis misalignment is the result of mounting imperfections of the multi-sensor unit in attachment to the measuring platform. It causes a non-orthogonality of the axes that define a sensor coordinate system. As a result, each sensor axis can be affected by the originally perpendicular actuating variable. Sensing axis misalignment has to be compensated by calibration.
- **Sensor noise:** Sensor noise represents the overall uncertainty in the sensor model given by effects that are time correlated and cannot be deterministically modelled. Therefore, sensor noise has to be *modelled by random processes* with parameters derived from analysis of both static and dynamic noise measurements (see next chapter).
- **Magnetic Sensitivity:** Magnetic sensitivity refers to the influence of external magnetic field on the drift characteristics of the sensor. The sensitivity of the bias on the strength and orientation of an applied magnetic field is determined by laboratory tests.
- **Shock Survivability:** The purpose of shock tests is to measure the sensor resilience to large acceleration over a very short period of time. This resilience to shock impulse is defined by variation in the mean bias value for both static and dynamic cases.
- **Vibration Effect:** Evaluating the effect of vibration includes following:
  1. determining the acceleration squared - bias dependency of the sensor,
  2. examining the sensor resilience in specific vibratory environment,
  3. determining the sensor resonant frequency and response magnitude,
  4. evaluating the variation in the output noise characteristics during vibration load.
- **Aging:** Aging is examined by a series of laboratory tests to determine the sensor quality deterioration over long time periods.

4.2.3 INERTIAL SENSOR ERRORS ANALYSIS USING POWER SPECTRAL DENSITY

For real world systems encountered in practice, it is not justified to assume all noises are white Gaussian; however, the white noise is crucial for the KF best performance. Using power spectral density (PSD) data and appropriate differential equations, the solution is to develop a noise model that will shape the white noise input accordingly to represent the real noise spectrum. Such a noise model is called a *shaping filter.* A proof can be found in [26] and shows that a linear time-invariant system (shaping filter) driven by wide sense stationary white noise (stationary with respect to 1st and 2nd moments) provides such a noise model. This model can be then augmented into the state space model for the KF. Random process for a continuous time state space model or a random sequence for a discrete time state space model can be identified according to its PSD as a *white noise, random walk (Wiener process), random constant, exponentially correlated noise (1st order Gauss-Markov (GM) process), harmonic noise* or any *combination* of these [30], see **Figure 4-2**. Each of these random processes has specific characteristics [26]:

- White noise PSD is flat and constant.
- Random walk noise PSD decreases with the value of -20dB/decade; considered stationary on time interval τ.
- Random constant is an unpredictable random quantity with a constant value.
- Exponentially correlated noise PSD flattens near DC component. The correlation among data samples decreases with the increase of the time shift between samples (zero at $\tau = \infty$).
- Harmonic noise PSD has bump where the spread depends on damping frequency.

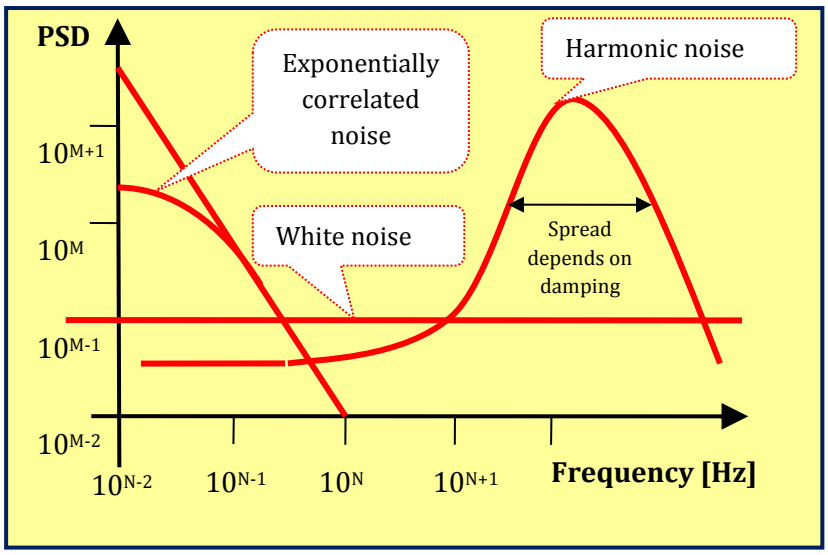


Figure 4-2 Power spectral density characteristics of various noise types

In general, inertial sensors are usually approximated by 1st order GM process. GM processes of higher order may be used as well; however, by increasing the order, the number of KF error states for each sensor increases as well and this may possibly cause the KF being unstable. Any GM process of any order can be represented using an autoregressive (AR) process of appropriate order [30]. AR process of order p can be described using a pole-zero transfer function $H(z)$ where $X(z)$ is the z-transform of the input $x(k)$, $Y(z)$ is the z-transform of the output $y(k)$ and $\alpha_1, \alpha_2, \ldots, \alpha_p$ and β_0 are the AR process parameters, with k being the discrete time step [39 p. 34]:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\beta_0}{1+\sum_{n=1}^{p} \alpha_n z^{-n}} \quad \text{or} \quad y(k) = -\sum_{n=1}^{p} \alpha_n y(k-n) + \beta_0 x(k) \tag{4-14}$$

It is obvious that two special cases exist [30], [39]: The first one is the zero-order GM process; the process value at any time does not depend on any past values, i.e. no time correlation, and therefore it can be regarded as a white noise process. The second special case exists on the contrary when the order of the GM process is very high, i.e. $p \to \infty$, which yields a constant PSD of σ^2, and hence the process can be regarded as random constant. The approximation by p^{th} order AR process is necessary when PSD exhibits more complex trend.

The definitions of shaping filters for each random process are given by their PSD as follows (see **Table 4-1**) [26 pp. 79 - 83]. Let $x(t)$ be a zero-mean scalar stationary random process with autocorrelation function (ACF) $\psi_{x(\tau)}$ and time difference τ. The ACF and PSD of $x(t)$ are defined by using Fourier transform [64], and [65]:

$$\psi_x(\tau) = \mathrm{E}[x(t)x(t+\tau)], \qquad \Psi_x(\omega) = \int_{-\infty}^{\infty} \psi_x(\tau) e^{-j\omega\tau} \mathrm{d}\tau \tag{4-15}$$

Noise Type	Autocorrelation Function ψ_x Power Spectral Density Ψ_x	State Space Models	
White noise	$\psi_x(\tau)=\sigma^2\delta^2(\tau)$ $\Psi_x(\omega)=\sigma^2$	Always treated as measurement noise	
Random walk	$\psi_x(\tau)=(undefined)$ $\Psi_x(\omega)\approx\sigma^2/\omega^2$	$\dot{x}=w(t)$ $\sigma_x^2(0)=0$	$x_k=x_{k-1}+w_{k-1}$ $\sigma_x^2(0)=0$
Random constant	$\psi_x(\tau)=\sigma^2$ $\Psi_x(\omega)=2\pi\sigma^2\delta(\omega)$	$\dot{x}=0$ $\sigma_x^2(0)=\sigma^2$	$x_k=x_{k-1}$ $\sigma_x^2(0)=\sigma^2$
Harmonic	$\psi_x(\tau)=\sigma^2\cos(\omega_0\tau)$ $\Psi_x(\omega)=\pi\sigma^2\delta(\omega-\omega_0)$ $+\pi\sigma^2\delta(\omega+\omega_0)$	$\dot{x}=\begin{bmatrix}0 & 1\\ -\omega_0^2 & 0\end{bmatrix}x$	$P(0)=\begin{bmatrix}\sigma^2 & 0\\ 0 & 0\end{bmatrix}$
Exponentially correlated	$\psi_x(\tau)=\sigma^2e^{-\alpha\lvert\tau\rvert}$ $\Psi_x(\omega)=\frac{2\sigma^\alpha\alpha}{\omega^2+\alpha^2}$	$\dot{x}=-\alpha x+\sigma\sqrt{2\alpha}w(t)$ $\sigma_x^2(0)=\sigma^2$	$x_k=e^{-\alpha}x_{k-1}$ $+\sigma\sqrt{1-e^{-2\alpha}}w_{k-1}$ $\sigma_x^2(0)=\sigma^2$
p[th] order GM process	$\psi_x(\tau)=\sigma^2e^{-\alpha_p\lvert\tau\rvert}\sum_{n=0}^{p-1}\frac{(p-1)!(2\alpha_p\lvert\tau\rvert)^{p-n-1}(p+n+1)!}{(2p-2)!n!(p-n-1)!}$		

Table 4-1 Shaping filter equations with σ being standard deviation, δ Kronecker delta, $1/\alpha$ correlation time, $\boldsymbol{P}$ covariance matrix and k discrete time step; definitions according to [26 p. 80] and [39 p. 20]

Since the most commonly used random process for inertial sensor error modelling is the 1st order Gauss-Markov process, it is important to specify its one parameter, the correlation time. Technique used to identify the correlation time of a random process is based on the ACF of the static measurement noise, [39] as in the **Figure 4-3**:

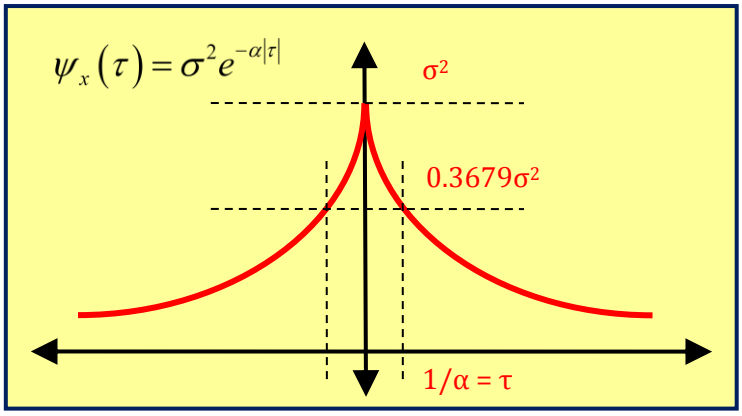


Figure 4-3 Autocorrelation function of the 1st order Gauss-Markov process

However, to determine the parameters of higher order GM processes, the ACF approach is not sufficient and other estimation technique has to be used to provide AR representation as in equation (4-14). Several methods have been reported to estimate the values of the α_n parameters in equation (4-14) by fitting AR model to the input data: the Yule-Walker (Y-W) method [39 p. 36], the Covariance method [39 p. 40], and the Burg's method [39 p. 41]. All three methods are implemented in *Signal Processing Toolbox* for MATLAB, each of them providing different features. The Y-W method can be used effectively only on long data records because of the limitation due to the data windowing process. The Covariance method provides more accurate estimates; however, it can lead to unstable AR models. The Burg's method overcomes these drawbacks and provides both stable models and high resolution even for short data records [39 p. 41]. In principle, the Burg's method uses the measured data by defining both forward and backward prediction error terms. The cost function to be minimized is in this case the sum of both the forward and the backward prediction errors. Regarding implementation, the forward and the backward prediction errors are expressed recursively. These recursion formulae form the basis of what is called a Lattice (or Ladder) realization of a prediction error filtering [39 pp. 41-43]. Shaping filter with states $\boldsymbol{x}_{SF}$ for such AR model has the following state space description [39 p. 48]:

$$\begin{pmatrix} x_{SF,k-p+1} \\ \vdots \\ x_{SF,k-3} \\ x_{SF,k-2} \\ x_{SF,k-1} \\ x_{SF,k} \end{pmatrix} = \begin{pmatrix} 0 & 1 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 \\ -\alpha_p & -\alpha_{p-1} & \cdots & -\alpha_3 & -\alpha_2 & -\alpha_1 \end{pmatrix} \begin{pmatrix} x_{SF,k-p} \\ x_{SF,k-p+1} \\ \vdots \\ x_{SF,k-3} \\ x_{SF,k-2} \\ x_{SF,k-1} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \beta_0 \end{pmatrix} w_k \qquad (4\text{-}16)$$

All of the random processes introduced in this chapter can be simulated and hence used for noise generation in various testing scenarios or as driving noise to the KF. The MATLAB code developed for such noise generation and simulation is provided in Appendix 8.5 (**Matlab 8-2**). The corresponding noise characteristics are presented as examples in **Figure 4-4**, **Figure 4-5**, **Figure 4-6**, and **Figure 4-7**.

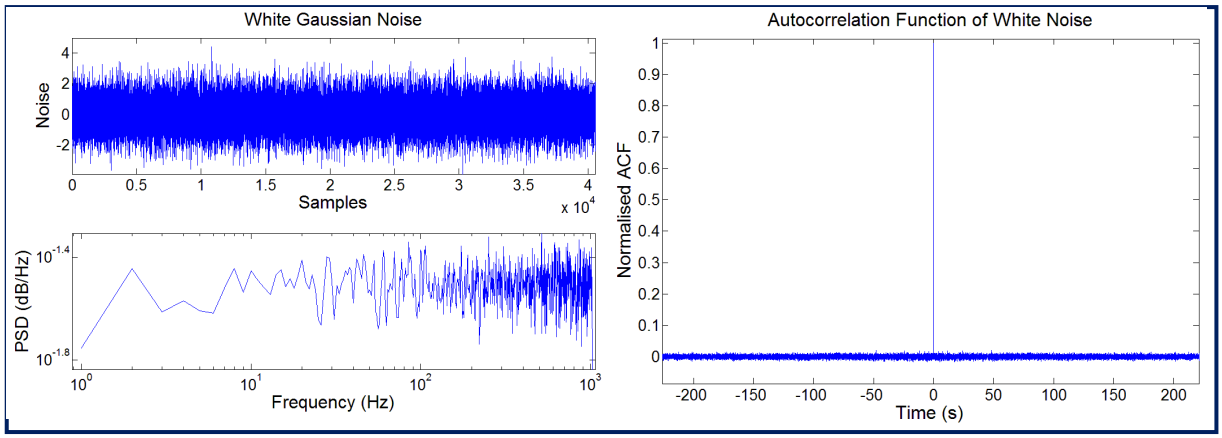


Figure 4-4 White noise simulation; its power spectral density and autocorrelation function

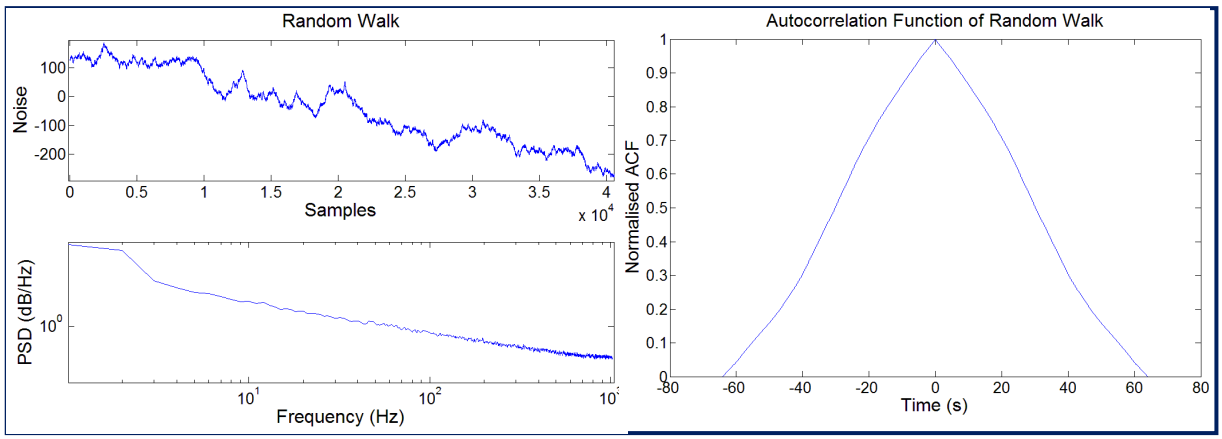


Figure 4-5 Random walk simulation; its power spectral density and autocorrelation function

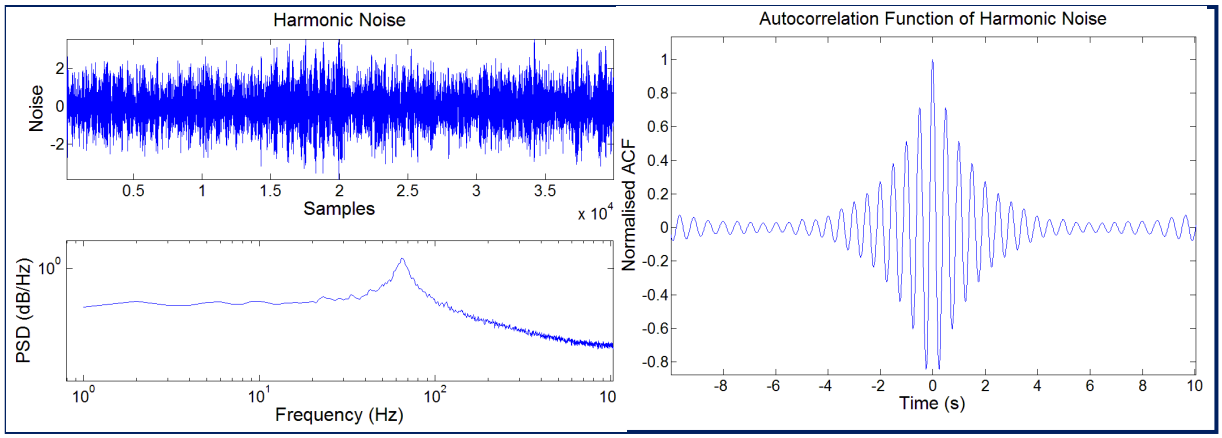


Figure 4-6 Harmonic noise simulation; its power spectral density and autocorrelation function

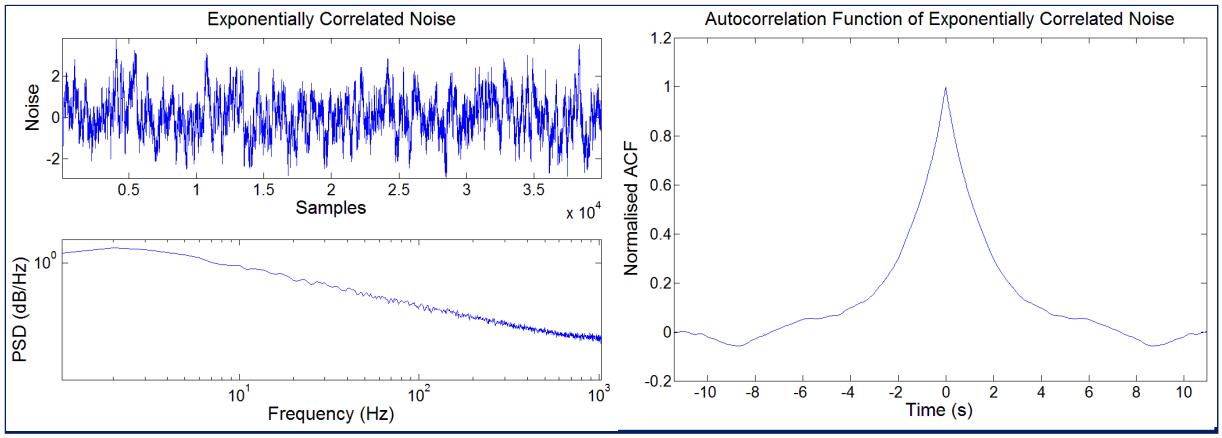


Figure 4-7 Exponentially correlated noise simulation; its power spectral density and autocorrelation function

Since the technique proposed in this chapter uses the PSD / ACF analysis to approximate the sensor errors by random processes according to the PSD / ACF trend, it is viewed as a combined time and frequency domain analysis.

4.2.4 INERTIAL SENSOR ERRORS ANALYSIS USING ALLAN VARIANCE

Allan variance (AVAR) is a time domain analysis technique that has been accepted into an IEEE Std. 647-2006 standard for testing of laser gyros [66]. This method was initially developed by David Allan [67] of the National Bureau of Standards to quantify the error statistics of a Caesium beam frequency standard employed as the U.S. Frequency Standards in 1960's [68]. Allan variance technique can be used to determine the character of the underlying random processes [69] that give rise to the data noise in a similar way as it was introduced using the PSD in the chapter 4.2.3. AVAR analysis finds its use in many different areas of measurements, such as inertial sensors performance analysis [68], [70], [71] and [72] or Earth rotation time series analysis [73].

The key attribute of this method is that it allows for more precise characterization and identification of error sources and their contribution to the overall noise statistics [68]. It can be used to characterize various types of noise terms in the inertial sensor data by performing certain operations on the entire length of data [69]. Allan's definition was originally related to six basic noise terms [68]. These were actually expanded to seven by introducing the correlated noise [70], which is rather difficult to identify this way and usually is approximated by exponentially correlated random process, the GM process. The seven noise terms are: *quantization noise*, *angle / velocity random walk* for angular rate sensors / accelerometers (ARW), *harmonic (sinusoidal) noise*, *bias instability*, *rate/acceleration random walk* (RRW), *drift rate ramp*, and *correlated noise*, e.g., exponentially correlated noise (for typical AVAR plot see **Figure 4-8**).

The AVAR technique provides several significant advantages over the others. Traditional approaches, such as computing the sampled mean and variance from a measurement set, do not reveal the underlying error sources [68]. Although the combined PSD / ACF approach provides a complete description of all basic error sources, the results are difficult to interpret. PSD is ideal for identifying either narrowband harmonic components or broadband sources in general; however extracting other contributing components such as bias instability, angle / velocity random walk, and quantization noise parameters is complicated [68].

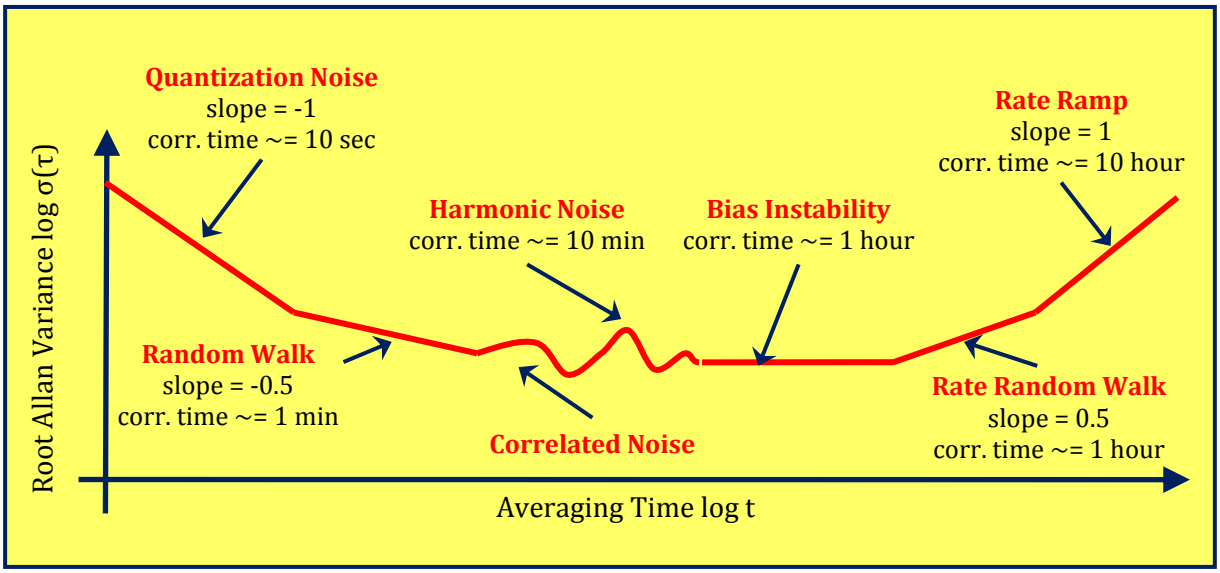


Figure 4-8 Allan Variance sample plots with typical correlation times [74 p. 115]

The error sources have slopes between ±1 and these slopes identify the different contributing sources of the accelerometer and angular rate sensor noise [68]. Each component is given by a typical correlation time according to appropriate scales. It is important to mention, that the error sources considered as the most important are in practice usually only the bias instability and the random walks (both ARW and RRW) [69], [72]. Therefore, determining the parameters of these error sources is the desired output of the AVAR analysis.

AVAR ANALYSIS IMPLEMENTATION USING CLUSTER METHOD

AVAR analysis is based on a method of cluster analysis. The principle of the cluster analysis can be concluded as follows [68 pp. 2-3]:

1. The measured data are divided into clusters of specified length – the clusters. Assume the data of length N, hence $K = N/M$ is the number of clusters formed.
2. The average value of each cluster is then computed for $k = 1, \ldots, K$.

$$\bar{\omega}_k(M) = \frac{1}{M}\sum_{k=1}^{M} \omega_{(k-1)M+1} \tag{4-17}$$

   where ω_k is the measured rate value at time step k.
3. A two point, unbiased sample Allan variance estimate is computed from the successive cluster averages for the specified correlation time $\tau_M = M / f_s$:

$$\sigma_A^2 \cong \frac{1}{2(K-1)}\sum_{k=1}^{K-1}\left(\bar{\omega}_{k+1}(M) - \bar{\omega}_k(M)\right)^2 \tag{4-18}$$

4. Different cluster length or correlation time for each AVAR computation can be chosen to obtain the AVAR as a function of correlation time. When plotted in logarithmic scales, the discrimination of different contributing error sources is done simply by examining the slope of the AVAR plot [68]. To extract information on a specific source of error one particular value of correlation time is selected and the corresponding value of root AVAR is obtained from the plot. The root AVAR value is then used to compute the desired parameter in the noise model equation.

The implementation of this algorithm can be done either in batch processing mode [68 p. 4], taking advantage of vector based computation for speed, or in recursive mode [68 p. 5], advantage for real time processing due to lower data storage space requirements. The batch mode was chosen for implementation in this dissertation since AVAR analysis was done by the means of post-processing.

DESCRIPTION OF ERROR SOURCES USING AVAR

In order to clarify the relation of AVAR and noise source characterisation it is important to express the AVAR in the frequency domain; for proof see [68 pp. 24-25]:

$$\sigma_{\omega}^{2}(\tau) = 4\int_{0}^{\infty} \Psi_{\omega}(f)\frac{\sin^{4}(\pi f\tau)}{(\pi f\tau)^{2}}\,\mathrm{d}f \tag{4-19}$$

where τ is a correlation time, and $\Psi_{\omega}(f)$ is two-sided PSD of the measured rate / acceleration noise data, which should obey the definition of a stationary process, i.e. the ACF is only function of τ.

As proven by Allan in [67] different error sources can be expressed in various powers of correlation time, i.e. AVAR for each error source can be obtained from its PSD when substituted into equation (4-19) and integrated. AVAR is then proportional to the total error power of gyro or accelerometer output when passed through a filter with transfer function of $\frac{\sin^{4}(\pi f\tau)}{(\pi f\tau)^{2}}$, that is a band-pass determined by the cluster τ. By varying the cluster time τ different types of random processes can be constructed.

Quantization noise: This noise is due to the discrete nature of the sensors output signal and represents the minimum resolution level of the sensor; for angular rate sensors usually expressed in μrad, for accelerometers in mg. It is introduced into an analogue signal by encoding it in digital form when rounding the analogue value to the digital output value. The AVAR for the quantization noise is given as [67], [69]:

$$\sigma_{Q}^{2}(\tau) = \frac{3Q^{2}}{\tau^{2}} \tag{4-20}$$

where Q is the quantization noise coefficient with theoretical limit $S/\sqrt{12}$ for S being the sensor scale factor [68].

If the root AVAR, the Allan deviation, is plotted vs. τ in log-log scale, the quantization noise is represented by the slope of -1 and the parameter Q can be obtained for $\tau = \sqrt{3}$ s. Quantization noise is characterised by a short correlation time, i.e. wide bandwidth. Since vehicle motion has usually low bandwidth, this noise can be filtered out and is not considered a major source of error [68].

Angle / Velocity Random Walk: The ARW is result of integrating a wideband PSD noise and it is characterised by a small varying drift with variance increasing with time [68]. It is defined as: *the angular/velocity error build-up with time that is due to white noise in angular rate/acceleration signal* [63 p. 14]. Typically ARW has a bandwidth less than 10 Hz and therefore it can be considered a major source of error for most INS. The AVAR is for the ARW is defined [67], [69]:

$$\sigma_{\mathrm{ARW}}^{2}(\tau) = \frac{N^{2}}{\tau} \tag{4-21}$$

where N is the ARW coefficient usually expressed in deg/h/$\sqrt{\mathrm{Hz}}$ (i.e. deg/$\sqrt{\mathrm{h}}$) for angular rate sensors and in (m/s)/$\sqrt{\mathrm{h}}$ for accelerometers.

If root AVAR is plotted in log-log scale vs. τ, the ARW is represented by the slope of -0.5. Hence, the value of N is obtained by reading the slope line at τ = 1 s [68]. For example, consider a navigation unit that has the ARW measurement of 0.2 deg/$\sqrt{\mathrm{h}}$. This means that after 1 hour the standard deviation of the orientation error will be 0.2 deg, after 2 hours it will be $\sqrt{2}$*0.2 = 0.28 deg and etc.

Bias Instability: Bias instability is described as the low frequency bias fluctuations in the measured rate data caused usually by electronics or components susceptible to random "flickering" [68]. The AVAR for the bias instability is defined as follows:

$$\sigma_{BI}^2(\tau) = \frac{2B^2}{\pi}\left[\ln 2 - \frac{\sin^3(\pi f_0\tau)}{2(\pi f_0\tau)^2}(\sin(\pi f_0\tau) + 4\pi f_0\tau\cos(\pi f_0\tau)) + C_i(2\pi f_0\tau) - C_i(4\pi f_0\tau)\right]$$
$$\sigma_{BI}^2(\tau) = (0.664B)^2 \; for \; \tau \gg \frac{1}{f_0} \tag{4-22}$$

where B is the bias instability coefficient, f_0 is the 3 dB cut-off frequency and C_i is the cosine-integral function.

The value of B can be obtained by reading the root AVAR vs. τ log-log plot where the slope is zero. The units of the bias instability are deg/s (or deg/h) for angular rate sensors and mg for accelerometers. Bias instability is usually specified as a 1σ value with units of deg/h, or deg/s. Over time bias instability creates a random walk in gyro or accelerometer bias, whose standard deviation grows proportionally to the square root of time. For this reason bias instability can be modelled as bias random walk [70]:

$$\mathrm{BRW}\left(^\circ/\sqrt{h}\right) = \frac{\mathrm{BS}(^\circ/h)}{\sqrt{t(h)}} \tag{4-23}$$

where t is the time span for which the bias instability is defined.

Harmonic Noise: Harmonic noise is characterised by a number of distinct frequencies in the PSD. Thus, the log-log scale of the AVAR plot would indicate this harmonic behaviour with successive peaks. It is described using AVAR as follows [68]:

$$\sigma_H^2(\tau) = \omega_0^2\left(\frac{\sin^2(\pi f_0\tau)}{\pi f_0\tau}\right)^2 \tag{4-24}$$

where ω_0 is the rate amplitude.

The log-log plot of the root AVAR vs. τ would exhibit sinusoidal behaviour with successive peaks of slope of ±0.5 [68].

Rate/Acceleration Random Walk: The RRW noise is a result of integrating wideband acceleration PSD noise and is usually associated with the limiting case of an exponentially correlated noise with a very long correlation time [68]. For gyros the AVAR is defined as *the drift rate error build-up with time that is due to white noise in angular acceleration* [63 p. 14]:

$$\sigma_{\mathrm{RRW}}^2(\tau) = \tau\frac{K^2}{3} \tag{4-25}$$

where K is the RRW coefficient.

The RRW is represented by a slope of +0.5 in the log-log AVAR vs. τ plot and the parameter K can be obtained for τ = 3 s. The units of K are given in deg/h$^{1.5}$ for gyros and in m/s/h$^{1.5}$ for accelerometers.

Rate Ramp: The rate ramp noise is rather deterministic than random in nature. Its influence on measured data is slow and monotonic with comparable effect such as a very slow acceleration of the navigated vehicle in the same direction over hours long period of time [68]. The AVAR expression for rate amp is as follows:

$$\sigma_{\mathrm{RR}}^2(\tau) = \frac{R^2\tau^2}{2} \tag{4-26}$$

where R is the rate ramp coefficient.
The rate ramp is represented by a slope of+1 in the log-log AVAR vs. τ plot and the parameter R can be obtained for $\tau = \sqrt{2}$ s.

Correlated Noise: By the correlated noise it is usually meant the exponentially correlated noise described by a GM process (see chapter 4.2.3). The correlated noise is identified by AVAR and is parameterized by the noise amplitude q_c and finite correlation time T_c as follows:

$$for\ \tau \gg T_c: \sigma_{CN}^2(\tau) = \frac{(q_c T_c)^2}{\tau}, for\ \tau \ll T_c: \sigma_{CN}^2(\tau) = \frac{q_c^2}{3}\tau \tag{4-27}$$

Based on the AVAR definitions presented in this chapter, the root AVAR definitions can be obtained by taking the square root as follows [68]:

Noise Types	Units (Gyro)	Slope	Root Allan Variance (deg/s)	Parameter of interest
Quantization noise	μrad	**-1**	$\sigma_Q(\tau) = \frac{\sqrt{3}Q}{\tau}$	Q
Angle/Velocity Random Walk	deg/$\sqrt{\mathrm{h}}$ resp. m/s/$\sqrt{\mathrm{h}}$	**-0.5**	$\sigma_{\mathrm{ARW}}(\tau) = \frac{N}{\sqrt{\tau}}$	N
Bias Instability	deg/s resp. m/s	**0**	$\sigma_{\mathrm{B}}(\tau) = 0.664B$	B
Harmonic noise	deg/s	**±0.5**	$\sigma_{\mathrm{H}}(\tau) = \omega_0 \frac{\sin^2(\pi f_0 \tau)}{\pi f_0 \tau}$	ω_0
Rate/Acceleration Random Walk	deg/h$^{1.5}$ resp. m/s/h$^{1.5}$	**+1**	$\sigma_{\mathrm{RRW}}(\tau) = K\sqrt{\frac{\tau}{3}}$	K
Rate Ramp	deg/s^2	**0.5**	$\sigma_{\mathrm{RR}}(\tau) = \frac{R\tau}{\sqrt{2}}$	R
Correlated Noise	deg/$\sqrt{s}$	**±1**	$for\ \tau \gg T_c: \sigma_{CN}^2(\tau) = \frac{(q_c T_c)^2}{\tau}$, $for\ \tau \ll T_c: \sigma_{CN}^2(\tau) = \frac{q_c^2}{3}\tau$	q_c, T_c

Table 4-2 Summary of the root AVAR definitions for different errors sources

EXTRACTION OF ERROR SOURCES FROM AVAR

Depending on the type of sensor, its construction, principle of operation and on the environment, where measurement data are obtained, in general, any combination of the random noise components concluded in **Table 4-2** can be present in data. If assumed that these sources are statistically independent, the overall AVAR can be expressed [68]:

$$\sigma^2_{\text{total}} = \sigma^2_{\text{Q}} + \sigma^2_{\text{ARW}} + \sigma^2_{\text{BIAS}} + \sigma^2_{\text{H}} + \sigma^2_{\text{RRW}} + \sigma^2_{\text{RR}} + \sigma^2_{\text{CN}} \tag{4-28}$$

Hence, the root AVAR, the Allan deviation, can be computed using coefficients A_n obtained by a least mean squares estimation method:

$$\sigma_{\text{A}}(\tau) = \sqrt{\sigma^2_{\text{total}}} = \text{f}\left(\sigma_{\text{Q}}, \sigma_{\text{ARW}}, \sigma_{\text{BIAS}}, \sigma_{\text{H}}, \sigma_{\text{RRW}}, \sigma_{\text{RR}}, \sigma_{\text{CN}}\right) = \sum_{n=-2}^{2} A_n \, \tau^{n/2} \tag{4-29}$$

As proven in [69] and [72], it is the bias instability, the angle/velocity random walk and the rate/acceleration random walk that are the most influential from the above mentioned six error sources. There is a quick direct approach for extraction of the N and B noise parameters that does not require the least mean square estimation procedure; given as follows [68]:

$$N = \left(\frac{\sigma_A(\tau_R)\sqrt{\tau_R}}{60}\right)(deg/hr) \tag{4-30}$$

$$B = \frac{\sigma_A(\tau_B)}{0.664}(deg) \tag{4-31}$$

where the τ_R, τ_B represent the time on the root AVAR versus τ plot where the parameters N and B are to be evaluated.

The extracted AVAR values of the angle/velocity random walk are used to design the covariance matrix of observational uncertainty $\boldsymbol{R}$; the AVAR values of the rate/acceleration random walk are used to design the covariance matrix of process noise $\boldsymbol{Q}$; both the $\boldsymbol{R}$ and $\boldsymbol{Q}$ being used in the KF algorithm [50]. In accordance to the random walk definition (see **Table 4-1**) the actual implementation to the KF can be expressed using following equations [50],

$$\dot{B}_{k+1} = \dot{B}_k + \sigma^2_{\text{RRW}_k} \tag{4-32}$$

where B is the bias error value, σ^2_{RRW} is the rate/acceleration random walk value.

4.3 PROCEDURE FOR SENSOR OUTPUT DATA DE-NOISING

The KF based integration of INS with other aiding sources such as GPS leads to error compensation within a limited bandwidth. This depends upon the rate of aiding update, vehicle dynamics, and inertial sensor performance characteristics (see 4.2.2). In general, inertial sensor errors can be classified into two categories according to their spectral signature [34]:

- *long-term errors* such as biases and drifts that can be partially compensated by aiding,
- *short-term errors* that require special threshold specification to be separated from motion dynamics.

According to [39], better performance can be expected if the short term inertial noise is suppressed prior to the integration by applying optimal low-pass filter beyond the motion dynamics bandwidth. However, short-term errors actually do overlap with the motion dynamics, so the pre-filtering has to be extended to frequencies which are not separable by external aiding. The approach of data de-noising using Discrete Wavelet Transformation (DWT), as mentioned in [34], is truly an option that can lead to significant INS precision improvement.

4.3.1 DISCRETE WAVELET TRANSFORM

Wavelet techniques are based on analyzing a signal through windowing process, but with variable windows size [39]. This provides an advantage over other signal processing techniques because of the capability of performing local analyses. This is possible since wavelets allow the use of narrow windows (short-time intervals) in cases when high frequency information is needed and wide windows (long-time intervals) if low frequency information is required. Hence, the wavelet transformation can be applied on the discrete signal sequence to decompose the signal into lower and higher frequency components. The DWT definition utilized in this dissertation is deduced from the Continuous Wavelet Transform (CWT) of a time domain signal $x(t)$ as described in [34 p. 84], [39] and [64]:

$$\mathrm{CWT}(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t) \Psi \left(\frac{t-b}{a} \right) \mathrm{d}t, \tag{4-33}$$

where a and b are the scaling and shifting parameters of the wavelet function $\Psi(t)$ respectively. Each scale a and position b results in a particular scaled and shifted version of a wavelet function that is to be multiplied by the time domain signal; see **Figure 4-9**.

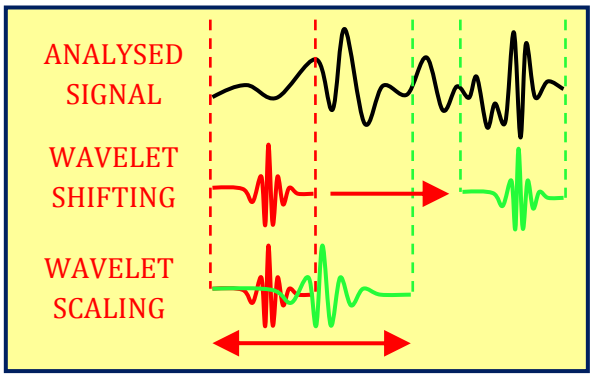


Figure 4-9 Wavelet shifting and scaling in the discrete wavelet transformation

Scaling the wavelet means stretching or compressing it in the time domain. The smaller the scale, the more the wavelet will be compressed. The larger the scale, the more the wavelet will be stretched instead. The low wavelet scales are hence suitable for analysis of high frequency signal components – the rapidly changing signal *details* - referred to as the signal "details" [34], [39]. The high wavelet scales on the other hand allow the analysis of low frequency signal components – the slowly changing features that actually roughly *approximate* the signal behaviour - referred to as the signal "approximations" [34], [39].

The actual integration over time gives the CWT coefficients corresponding to a and b. These coefficients are considered to be a measure of correlation between the used wavelet function and the signal itself for different scales and different time locations of the wavelet. The wavelet $\Psi(t)$ is called the basis function and it is required to have following properties to ensure the integration is finite: *short and oscillatory, zero average value, converge rapidly to zero at both ends* [34]. The DWT definition is hence as follows, assuming $x(n)$ to be discrete time sequence [34 p. 85] and [64]:

$$C_{j,k} = 2^{(-j/2)} \sum_{n} x(n)\Psi\left(2^{-j} - k\right), \tag{4-34}$$

where $\Psi(n)$ is the wavelet function (the basis function) and $2^{(-j/2)}\,\Psi(2^{-j} - k)$ are the shifted and scaled versions of $\Psi(n)$, based on the values of j (the scaling parameter) and k (the shifting parameter). The shifted and scaled wavelet function is usually denoted $\Psi_{j,k}(n)$; $C_{j,k}$ then represents the corresponding wavelet coefficients.

The decomposed signal is reconstructed by applying the inverse DWT (IDWT) on its computed wavelet coefficients. It is done by passing the coefficients of selected approximation level through the IDWT low-pass filter and resetting the coefficients of all subsequent details to zero before passing them through the IDWT high-pass filters. The original discrete signal $x(n)$ can therefore be generated from the corresponding wavelet [64]:

$$x(n) = \sum_{j} \sum_{k} C_{j,k} \Psi_{j,k}(n) \tag{4-35}$$

4.3.2 WAVELET MULTI-RESOLUTION ANALYSIS

For the vast majority of signals, the low frequency component is the one of interests since it gives the signal its identity [34]. On the other hand, the high frequency component usually corresponds to the signal noise, vibration, and partially to the system high dynamics. In the de-noising implementation of the DWT, wavelet coefficients of a signal are computed by passing such a signal through two complementary half-band filters: a low-pass filter and a high-pass filter. To obtain better resolution in frequency components of a specific signal, the signal is broken down into many lower-resolution components by repeating this DWT decomposition. This procedure is called the Wavelet Multi-Resolution Analysis (WMRA) and may be represented by the level of decomposition (LOD) tree; see **Figure 4-10**.

By using WMRA, the signal can be represented by a finite sum of components with different resolutions. Each component is processed adaptively, depending on the application. Beside the choice of the wavelet type, another question arises when determining the LOD for the data de-noising procedure. Two cases have to be distinguished according the mode of operation of the inertial sensor: the *static* and *dynamic data*.

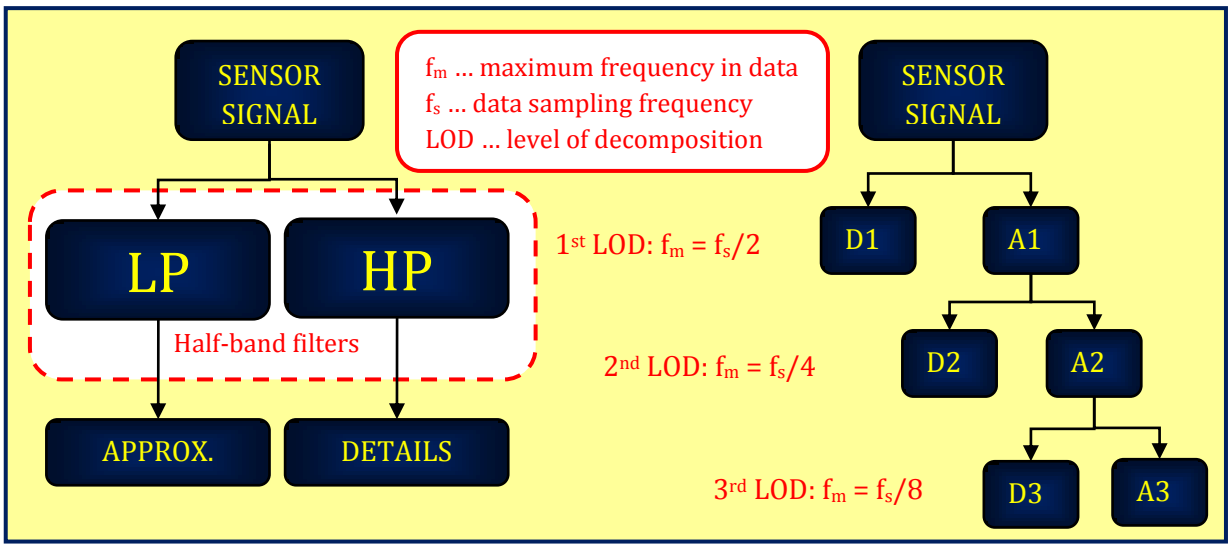


Figure 4-10 Signal decomposition principle and the level of decomposition tree [39 pp. 69, 71]

For static inertial data, sensory outputs contain signals such as *the Earth gravity components*, *the Earth rotation rate components* and *sensors long-term errors.* These signals have very low frequency, and therefore they can be separated easily from the high frequency noise components [39 p. 72]. To select an appropriate LOD, several decomposition levels are applied and the standard deviation is computed for each obtained component. The most suitable LOD is the one for which the standard deviation reaches its minimum value.

Concerning the dynamic inertial data de-noising, the output of the sensors contains both effects of *the actual vehicle motion dynamics* and *the sensor noise* as well as some other undesirable effects such as *vibration* [39 p. 73]. Before applying the WMRA on the dynamic data, it must be assured, that the decomposition or de-noising process does not remove any actual motion information. Therefore, a spectral analysis should be performed first. Then, the appropriate LOD is selected in such a way that the decomposition process will remove only the components that have frequencies higher than the detected motion frequency range. Such procedure is called the *coefficients thresholding* and plays key role for successful application of the WMRA. WMRA can easily be tested due to its implementation in *Wavelet Toolbox* for MATLAB. However, MATLAB implementation is not meant for real time processing. An alternative real time algorithm is required; such as the one proposed by Hamid in [34 pp. 88 - 90].

4.3.3 WAVELET THRESHOLDING

The process of wavelet thresholding is a procedure necessary for cases of noise interference within the frequency band of the motion dynamics [34], [39]. Since the signal disturbances caused by for example noisy environment or vibration are usually given by a single frequency and its harmonics, the aim is to separate it without degrading the signal of interest. Successful removal of this sinusoidal interference is given by the accuracy in the threshold estimation.

The wavelet thresholding technique is in principle a procedure that adjusts the coefficients, which contain sharp transition details of the monitored dynamics, such that the noise interference is cancelled out without distortion to these transition details [34]. There are two most commonly used thresholding operators; the *soft* and *hard* thresholding. In the case of hard thresholding, any wavelet coefficient with an absolute value below the threshold is replaced by zero. Coefficients with values above are kept without modification. In the case of soft thresholding, coefficients with absolute value above the selected threshold are replaced and reduced in value by the actual threshold value. Coefficients with values below are replaced by zero as in the case of hard thresholding. Soft and hard thresholding can be described by following equations [34]:

$$Threshold\_hard = \begin{Bmatrix} y \; if \; |y| > threshold \\ 0 \; if \; |y| < threshold \end{Bmatrix} \tag{4-36}$$

$$Threshold\_soft = \begin{Bmatrix} y - threshold * \text{signum}\{threshold\} \; if \; |y| > threshold \\ 0 \; if \; |y| < threshold \end{Bmatrix} \tag{4-37}$$

The type of the actual thresholding does not bring any significant difference to the quality of de-noising process; however, the implementation of thresholding itself plays a crucial role and is an integral part of the WMRA process. Example of WMRA application (with different LOD) on accelerometer measurements obtained during navigation field-tests is shown in **Figure 4-11**:

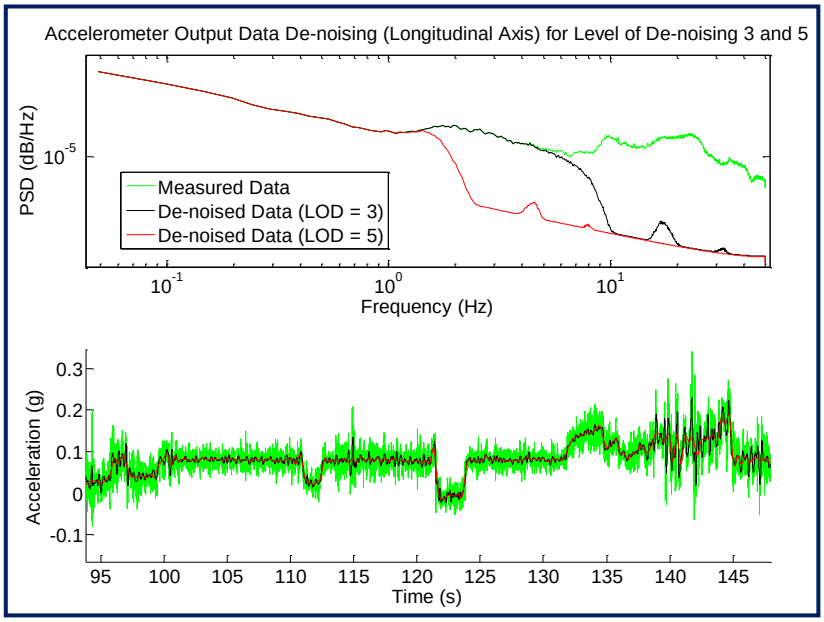


Figure 4-11 Accelerometer output data de-noising by the WMRA procedure

4.4 STRUCTURE OF THE INERTIAL NAVIGATION SYSTEMS

The main aim of any navigation process utilizing the inertial sensors is to fuse the information provided by the IMU and an aiding system, such as GPS, to accurately estimate the vehicle trajectory and orientation in space [25]. The conventional and proven way how to do so is to use the Kalman filter. The classical scheme for KF based INS/GPS integration is shown in **Figure 4-12**, where the compensation of both deterministic and random errors is highlighted. To estimate the INS errors, raw inertial measurements have to be processed first by the INS mechanization (or sometimes called strapdown mechanization) to obtain the integrated position, velocity, and attitude information, which is still biased with random errors. Then a dynamic state space error model has to be implemented into the KF and used to estimate the random errors. Usually, feedback loop is employed for the compensation of errors. Therefore, the quality of the navigation system depends upon the quality of the error estimates, which is strongly given by [23], [26], [34]:

- the quality of the measurements that can be improved by various de-noising procedures,
- the complexity and reliability of the error model chosen to suit the application.

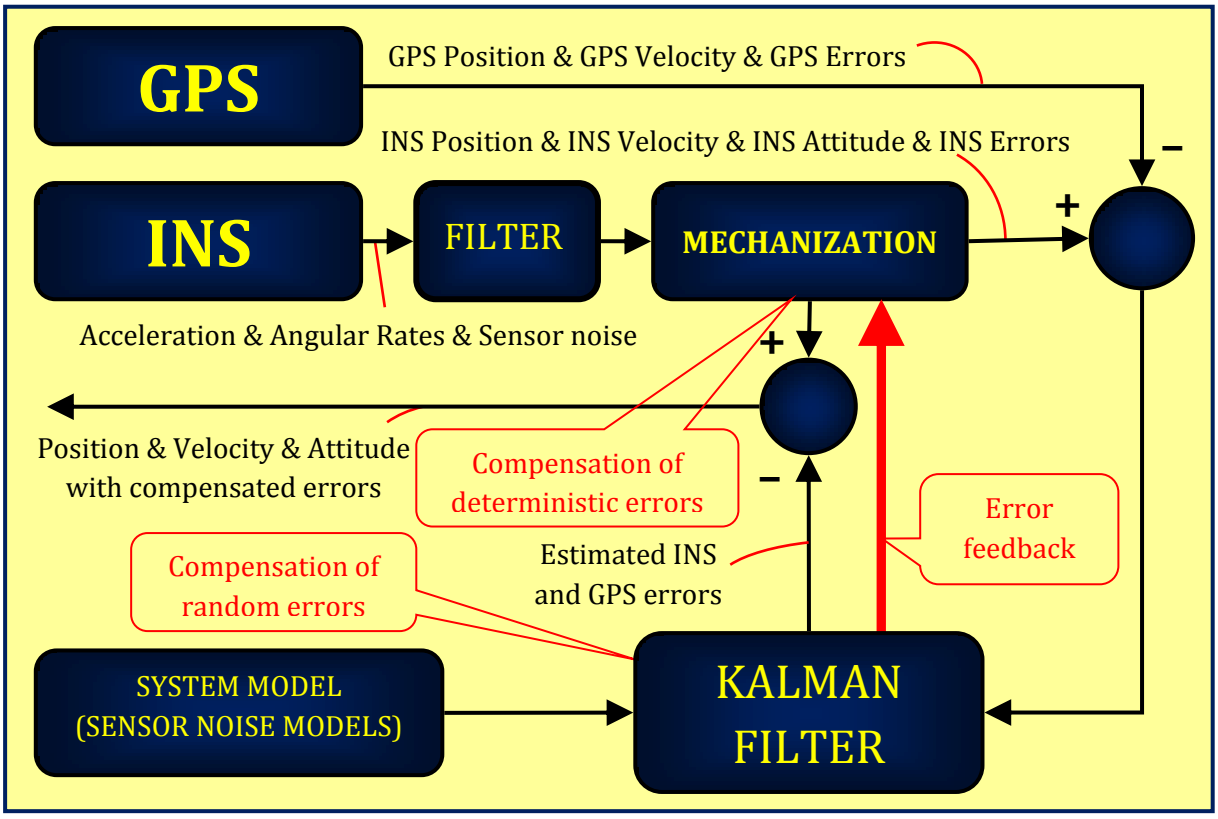


Figure 4-12 Conventional scheme for INS/GPS integration using a Kalman filter

4.4.1 ATTITUDE REPRESENTATION

The most commonly used and implemented approaches to the attitude representation for the purpose of inertial navigation are described in detail by Savage in [75] and [76]. Savage proposed an INS mechanization algorithm operating with the four following attitude representations:

- Euler angles (pitch, roll and yaw),
- Direction Cosine Matrix (DCM),
- Quaternion algebra,
- Rotation vector representation.

The same attitude representation is used by both Salychev [25] and Shin [32], although Salychev uses different axes orientation in the navigation frame. In this chapter, mathematical background necessary to work with all the four representations will be presented [77]. Regarding notation, general frames a and b will be used in this chapter. Hence, let the rotation vector, quaternion and DCM corresponding to the transformation from the b-frame to the a-frame be denoted as $\mathbf{\Pi}$, $\mathrm{q}_{\mathrm{b}}^{\mathrm{a}}$, and $\mathbf{C}_{\mathrm{b}}^{\mathrm{a}}$ respectively.

EULER ANGLES

Euler angles were originally developed by Leonhard Euler to describe the orientation of a rigid body in 3-dimensional Euclidean space. To give an object a specific orientation it may be subjected to a sequence of three rotations defined by the Euler angles. In other words, a rotation matrix, which describes the change in orientation, can be decomposed as a product of three elemental rotations defined by yaw, pitch, and roll angles, also known as Tait–Bryan rotations. These rotations are given by a specific sequence of Euler angles that are very often used in aerospace applications to define the relative orientation of the navigated object, i.e. the rotation between navigation frame and vehicle-fixed or body-fixed frame.

Consider a body frame, which is fixed to the vehicle such that the origin of the system is located at the centre of gravity, the x-axis points forward, the y-axis points to the right of the vehicle, and the z-axis points downwards to form an orthogonal right-handed system. Consider a local horizontal and local vertical reference frame (the navigation frame) that shares the same origin as the fixed frame but is always aligned with x-axis pointing in the direction of true north, y-axis pointing to true east, and the z-axis pointing down towards the centre of gravity of the earth. It will be shown without proof – for proof see [78 p. 33] - that the following matrix equation composed of three rotation matrices defined by the Euler angles provides the transformation from this navigation frame to the body frame:

$$\boldsymbol{r}^b = \boldsymbol{C}_n^b \boldsymbol{r}^n = \\ = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\rho) & \sin(\rho) \\ 0 & -\sin(\rho) & \cos(\rho) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{r}^n \tag{4-38}$$

where $\boldsymbol{r}^b = [x_b \; y_b \; z_b]$ is the position vector in the body frame, $\boldsymbol{r}^n = [x_n \; y_n \; z_n]$ is the position vector in navigation frame, ρ is roll angle, θ is pitch angle, ψ is the yaw angle, $\boldsymbol{C}_n^b$ is the direction cosine matrix (DCM) performing the transformation from navigation frame to the body frame.

There are also other possibilities how to define the DCM; however, as proposed by Titterton [23 p. 44], this is the optimal one for inertial navigation regardless the nature of navigated subject.

QUATERNION APPROACH

Quaternion approach is an alternative to the Euler angles representation in expressing the transformations between frames of reference. The quaternion is a four-dimensional vector composed of a scalar part *s* and a vector part $\boldsymbol{v}$. For a quaternion performing transformation from *b*-frame to *a*-frame, the inverse operation and the quaternion multiplication rule for computing the transcending transformation is defined as follows [32 p. 11], :

$$\boldsymbol{q}_b^a = \begin{bmatrix} s \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{4-39}$$

$$(\boldsymbol{q}_b^a)^{-1} = \begin{bmatrix} s \\ -\boldsymbol{v} \end{bmatrix} \tag{4-40}$$

$$\begin{gathered} \boldsymbol{q}_c^a = \begin{bmatrix} s_1 \\ \boldsymbol{v}_1 \end{bmatrix}, \boldsymbol{q}_b^c = \begin{bmatrix} s_2 \\ \boldsymbol{v}_2 \end{bmatrix} \\ \boldsymbol{q}_b^a = \boldsymbol{q}_c^a * \boldsymbol{q}_b^c = \begin{bmatrix} s_1 s_2 - \boldsymbol{v}_1^T \boldsymbol{v}_2 \\ s_1 \boldsymbol{v}_2 + s_2 \boldsymbol{v}_1 + \boldsymbol{v}_1 \times \boldsymbol{v}_2 \end{bmatrix} \end{gathered} \tag{4-41}$$

The transformation from the *b*-frame to *a*-frame using the transcending *c*-frame (transformation from *b*-frame to *c*-frame is followed by the transformation from *c*-frame to *a*-frame) can be expressed in a similar way using appropriate direction cosine matrices (DCM) and the matrix multiplication rule:

$$\boldsymbol{C}_b^a = \boldsymbol{C}_c^a \boldsymbol{C}_b^c \tag{4-42}$$

To obtain an inverse transformation matrix, simple transposition operation can be performed instead of matrix inversion since every DCM matrix is orthogonal (without proof) [32]:

$$\boldsymbol{C}_b^a = \left(\boldsymbol{C}_a^b\right)^T \tag{4-43}$$

There is a relationship between the quaternion representation and the DCM that was chosen for the transformation between the vehicle-fixed body frame and the navigation frame. The expression of the DCM in terms of a quaternion is according to [32 p. 14]:

$$\boldsymbol{C}_b^a = \begin{bmatrix} (q_1^2 + q_2^2 - q_3^2 - q_4^2) & 2(q_2 q_3 - q_1 q_4) & 2(q_2 q_4 + q_1 q_3) \\ 2(q_2 q_3 + q_1 q_4) & (q_1^2 - q_2^2 + q_3^2 - q_4^2) & 2(q_3 q_4 - q_1 q_2) \\ 2(q_2 q_4 - q_1 q_3) & 2(q_3 q_4 + q_1 q_2) & (q_1^2 - q_2^2 - q_3^2 + q_4^2) \end{bmatrix} \tag{4-44}$$

The expression of the quaternion from the DCM is not so straightforward. An example of robust algorithm for such conversion can be concluded as follows [79 pp. 882 - 890], [32 p. 15]:

$$\begin{gathered} P_1 = 1 + tr(\boldsymbol{C}_b^a), \;\; P_2 = 1 + 2c_{11} - tr(\boldsymbol{C}_b^a) \\ P_3 = 1 + 2c_{22} - tr(\boldsymbol{C}_b^a), \;\; P_4 = 1 + 2c_{33} - tr(\boldsymbol{C}_b^a) \end{gathered} \tag{4-45}$$

$$
\begin{aligned}
&If\ P_1 = \max(P_1, P_2, P_3, P_4)\\
&\quad q_1 = 0{,}5\sqrt{P_1},\ q_2 = \frac{c_{32} - c_{23}}{4q_1}, q_3 = \frac{c_{13} - c_{31}}{4q_1}, q_4 = \frac{c_{21} - c_{12}}{4q_1}\\
&If\ P_2 = \max(P_1, P_2, P_3, P_4)\\
&\quad q_2 = 0{,}5\sqrt{P_2},\ q_3 = \frac{c_{21} + c_{12}}{4q_2}, q_4 = \frac{c_{13} + c_{31}}{4q_2}, q_1 = \frac{c_{32} - c_{23}}{4q_2}\\
&If\ P_3 = \max(P_1, P_2, P_3, P_4)\\
&\quad q_3 = 0{,}5\sqrt{P_3},\ q_4 = \frac{c_{32} + c_{23}}{4q_3}, q_1 = \frac{c_{13} - c_{31}}{4q_3}, q_2 = \frac{c_{21} + c_{12}}{4q_3}\\
&If\ P_4 = \max(P_1, P_2, P_3, P_4)\\
&\quad q_4 = 0{,}5\sqrt{P_4},\ q_1 = \frac{c_{21} - c_{12}}{4q_4}, q_2 = \frac{c_{13} + c_{31}}{4q_4}, q_3 = \frac{c_{32} + c_{23}}{4q_4}
\end{aligned}
\tag{4-46}
$$

where *tr* stands for the *trace* of a matrix, which is the sum of the terms on the main diagonal.

The quaternion can be expressed in the terms of Euler angles as well [23 p. 50], [32 p. 18], [79]:

$$
\boldsymbol{q}_b^a = \begin{bmatrix}
\cos\left(\frac{\rho}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\rho}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right)\\
\sin\left(\frac{\rho}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) - \cos\left(\frac{\rho}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right)\\
\cos\left(\frac{\rho}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\rho}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right)\\
\cos\left(\frac{\rho}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) - \sin\left(\frac{\rho}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right)
\end{bmatrix}
\tag{4-47}
$$

Another representation of rotation is using the rotation vector. Rotation vector specifies the direction of rotation by the direction of the vector from the origin which coincides with the axis of rotation (perpendicular to the plane of rotation, right hand rule). The magnitude of the rotation is then given by Euclidean norm of the rotation vector [25]. The relative rotation of two frames, given by an angular rate, can be expressed as a time derivative of a rotation vector using Bortz equation [25], [75 p. 23], which is in its simplified form:

$$
\begin{aligned}
\dot{\boldsymbol{\Pi}} &= \boldsymbol{\omega}_{ab}^b + \frac{1}{2}\boldsymbol{\Pi} \times \boldsymbol{\omega}_{ab}^b + \frac{1}{\|\boldsymbol{\Pi}\|^2}\left(1 - \frac{\|\boldsymbol{\Pi}\|\sin(\boldsymbol{\Pi})}{2\left(1 - \cos(\boldsymbol{\Pi})\right)}\right)\boldsymbol{\Pi} \times \left(\boldsymbol{\Pi} \times \boldsymbol{\omega}_{ab}^b\right)\\
&\approx \boldsymbol{\omega}_{ab}^b + \frac{1}{2}\boldsymbol{\Pi} \times \boldsymbol{\omega}_{ab}^b + \frac{1}{12}\boldsymbol{\Pi} \times \left(\boldsymbol{\Pi} \times \boldsymbol{\omega}_{ab}^b\right)
\end{aligned}
\tag{4-48}
$$

where $\dot{\boldsymbol{\Pi}} = \frac{d\boldsymbol{\Pi}}{dt}$ is the time change of the rotation vector $\boldsymbol{\Pi}$, $\boldsymbol{\omega}_{ab}^b$ is the desired angular rate vector of the *b*-frame rotating with respect to the *a*-frame, $\|\cdot\|$ is the Euclidean norm of a vector.

If a rotation is given in angular rates, the approach using rotation vector is the most straightforward one. The quaternion approach was chosen to represent all the rotations used

throughout this thesis due the numerical stability. Therefore, the conversion of a rotation vector - given by an angular rate - to a quaternion has to be stated as well. According to [25], and [32 p. 12] the conversion is as follows:

$$\boldsymbol{q}_b^a = \begin{bmatrix} cos\|0.5\boldsymbol{\Pi}\| \\ \frac{sin\|0.5\boldsymbol{\Pi}\|}{\|0.5\boldsymbol{\Pi}\|}0.5\boldsymbol{\Pi} \end{bmatrix} is\ the\ direct\ (positive) form\ or$$
$$\boldsymbol{q}_a^b = \begin{bmatrix} cos\|0.5\boldsymbol{\Pi}\| \\ -\frac{sin\|0.5\boldsymbol{\Pi}\|}{\|0.5\boldsymbol{\Pi}\|}0.5\boldsymbol{\Pi} \end{bmatrix} is\ the\ inverse\ (negative)\ form \tag{4-49}$$

$$where\ \ \|\boldsymbol{\Pi}\| = \sqrt{\Pi_1^2 + \Pi_2^2 + \Pi_3^2} \tag{4-50}$$

In order to save the computational load, the geometric functions can be expressed using a number of the first terms of the Taylor series according to the desired precision. According to [32 p. 14] and [79] a rotation vector can be extracted from the quaternion values using following equations:

$$\boldsymbol{\Pi} = \frac{1}{f}[q_2 \quad q_3 \quad q_4]^T \tag{4-51}$$

$$where\ f \equiv \frac{\sin\|0.5\boldsymbol{\Pi}\|}{\|\boldsymbol{\Pi}\|}\ \ and\ \|0.5\boldsymbol{\Pi}\| = \frac{\sqrt{q_2^2 + q_3^2 + q_4^2}}{q_1} \tag{4-52}$$

All of the presented equations and transformations introduced in this chapter were implemented in MATLAB (see **Matlab 8-3** in Appendix 8.5) and used further on in the INS mechanization algorithm.

4.4.2 COORDINATE FRAMES FOR INS

There are in total five frames of reference that are considered throughout this thesis. Definitions used in this document coincide with the ones used in general; such as described by Titterton [23], Salychev [25] and Sotak [78], and as concluded in the **Figure 4-13** below:

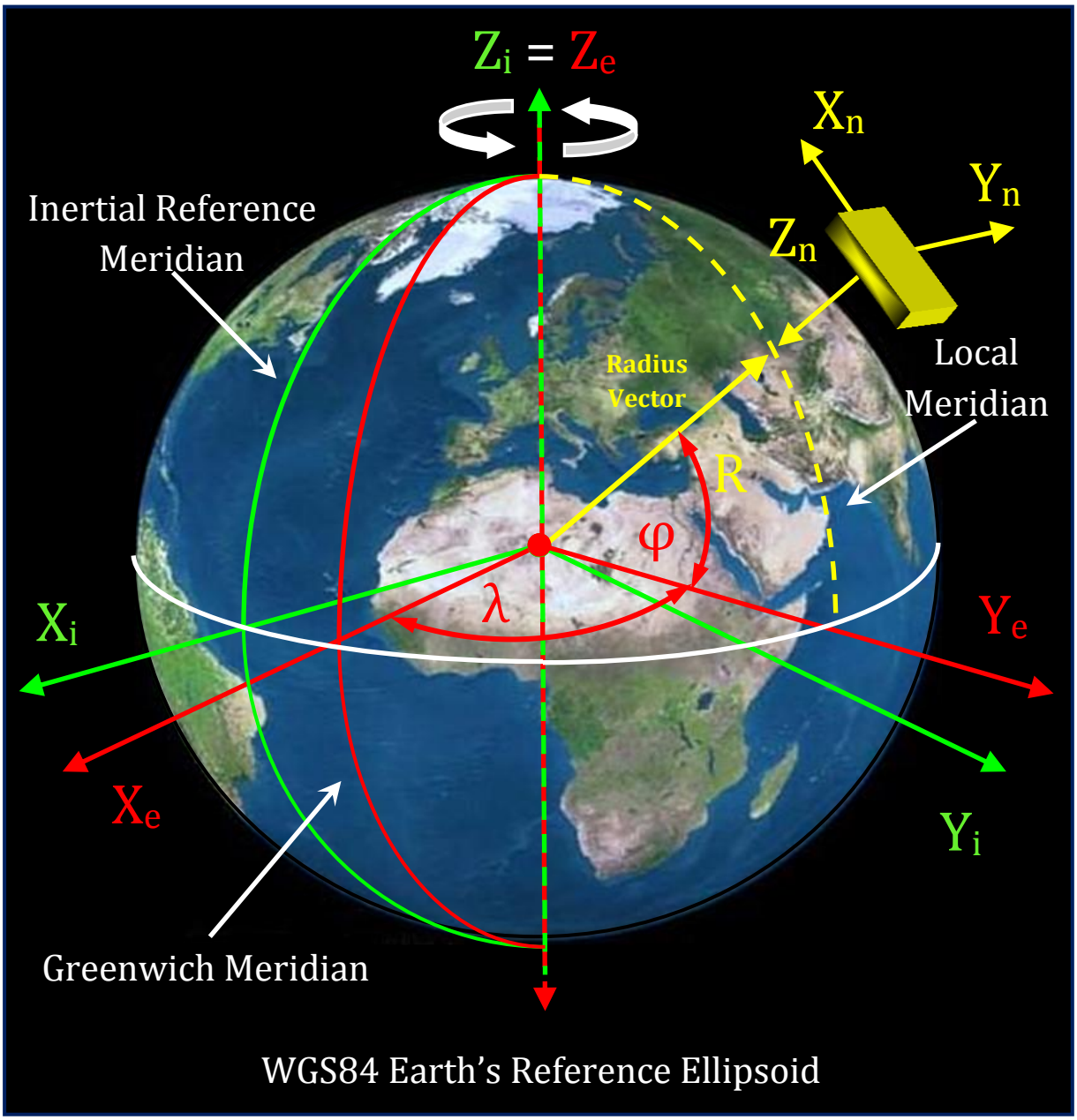


Figure 4-13 Frames of reference used for the aircraft navigation (ECI, ECEF, NED)

EARTH-CENTRED INERTIAL (ECI) FRAME

According to Newton, an Earth-centred inertial (ECI) frame is such a one that neither rotates nor accelerates. In fact, this is an easy approach to be realized in theory but impossible to achieve in practice. An approximation is usually made using distant stars as points of reference and assuming their position does not change [25]. In many surveying application a right ascension system is used. As described by Salychev [25], such system precesses and nutates at the rate of less than $3.6 \cdot 10^{-7}$ rad/s which is well below the noise level in inertial sensors. This approximation of the inertial coordinate frame is defined in [25], [78], see **Figure 4-13**:

- Origin – the centre of the Earth,
- X_i – axis directed towards mean vernal equinox,
- Y_i – axis completes a right handed system,
- Z_i – axis directed towards the north celestial pole.

EARTH-CENTRED-EARTH-FIXED (ECEF) FRAME

In order to use the Earth-centred Earth-fixed (ECEF) frame for navigation, it is necessary to distinguish the geocentric and geodetic latitudes. The geocentric latitude on the Earth's surface is defined by the angle subtended by the radius vector from the Earth's centre to the surface point and the equatorial plane. The geodetic latitude (further on only latitude) on the Earth's surface is defined by the angle subtended by the surface normal vector and the equatorial plane [23]. The ECEF is a frame that revolves around the Sun and rotates at a rate of $7.292115 \cdot 10^{-5}$ rad/s. It is defined as follows [25], [78]; see **Figure 4-13**:

- Origin – the centre of the Earth,
- X_e – axis directed towards the Greenwich meridian in the equatorial plane,
- Y_e – axis 90° east of Greenwich meridian in the equatorial plane,
- Z_e – axis is the axis of rotation of the reference ellipsoid.

The ECEF coordinates can be transformed to the inertial frame by a negative rotation about the Z-axis by the amount of the Greenwich Mean Sidereal Time (GMST) [25]. The reference ellipsoid used for computation is the WGS 84 with following parameters: semi-major axis r_e = 6378137.0 m and semi-minor axis r_p = 6356752.3 m.

To define angular rates, notation of ω_{ab}^{c} states that the b-frame rotates with respect to the a-frame, the rotation rate is expressed in the c-frame and has the magnitude of ω_{ab}^{c} in rad/s. According to this notation, the rotation rate of the Earth is:

$$\boldsymbol{\omega}_{ie}^{e} = \begin{bmatrix} 0 \\ 0 \\ \omega_e \end{bmatrix} \tag{4-53}$$

$$\omega_e = 7{,}2921158 \times 10^{-5} \frac{\text{rad}}{\text{s}} \tag{4-54}$$

The position vector in the ECEF is then expressed in terms of geodetic latitude φ, longitude λ and height h above the surface (altitude):

$$\boldsymbol{r}^{e} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \equiv \begin{bmatrix} (R_N + h)\cos(\varphi)\cos(\lambda) \\ (R_N + h)\cos(\varphi)\sin(\lambda) \\ (R_N(1-\varepsilon^2) + h)\sin(\varphi) \end{bmatrix} \tag{4-55}$$

where R_N is the radius of curvature in the prime vertical and ε is the eccentricity of the reference ellipsoid.

The reference ellipsoid is the approximation of the shape of the Earth since Earth is not a homogeneous sphere. This reference ellipsoid is given by the radius of curvature in the prime vertical R_N and by the meridian radius of curvature R_M, which both depend on the latitude, all according to the following equations [23 p. 54]:

$$R_M = \frac{r_e(1-\varepsilon^2)}{\sqrt{\left(1-\varepsilon^2\sin^2(\varphi)\right)^3}} = r_e\left[1+\varepsilon^2\left(\frac{3}{2}\sin^2(\varphi)-1\right)\right]$$
$$R_N = \frac{r_e}{\sqrt{1-\varepsilon^2\sin^2(\varphi)}} = r_e\left[1+\frac{\varepsilon^2}{2}\sin^2(\varphi)\right] \quad (4\text{-}56)$$

$$\varepsilon = \sqrt{1-\frac{r_p^2}{r_e^2}} \quad (4\text{-}57)$$

where r_e is the distance from the centre of the Earth to the Equator - the length of the semi-major axis - and r_p is the distance from the centre of the Earth to the North/South Pole - the length of the semi-minor axis [23 p. 53].
If the Earth is modelled using this reference ellipsoid given by R_M and R_N, these parameters are to be used to obtain the rates of change of latitude and longitude, especially when computing the transport rate, i.e. the rotation rate of the navigation frame with respect to the ECEF frame.

NAVIGATION (NED) FRAME

The navigation frame used in this dissertation is defined as north-east-down (NED) system and it is a non-inertial system with its origin fixed at the navigated object's centre of mass (COM). It is a local geodetic frame such that its axes are oriented along the geodetic directions defined by the Earth's surface (see **Figure 4-13**):

- The X_N-axis points north, parallel to the geoids surface in the polar direction.
- The Y_N-axis points east, parallel to the geoids surface along a latitude curve.
- The Z_N-axis points downward, towards the Earth's centre.

A problem arises with the NED frame when navigating at poles, where the NED is singular and north direction cannot be determined [78]. The *wander azimuth* frame has to be used instead [25]. The DCM describing the transformation from NED to ECEF is expressed in terms of geodetic latitude and longitude as follows [23], [25]:

$$\boldsymbol{C}_n^e = \begin{bmatrix} -\sin(\varphi)\cos(\lambda) & -\sin(\lambda) & -\cos(\varphi)\cos(\lambda) \\ -\sin(\varphi)\sin(\lambda) & \cos(\lambda) & -\cos(\varphi)\sin(\lambda) \\ \cos(\varphi) & 0 & -\sin(\varphi) \end{bmatrix} \quad (4\text{-}58)$$

where $\boldsymbol{C}_n^e$ is the transformation matrix from NED to ECEF.

The quaternion equivalent to the $\mathbf{C}_{\mathrm{n}}^{\mathrm{e}}$ matrix is then [32 p. 23]:

$$\boldsymbol{q}_n^e = \begin{bmatrix} cos(-\pi/4 - \varphi/2)cos(\lambda/2) \\ -sin(-\pi/4 - \varphi/2)sin(\lambda/2) \\ sin(-\pi/4 - \varphi/2)cos(\lambda/2) \\ cos(-\pi/4 - \varphi/2)sin(\lambda/2) \end{bmatrix} \tag{4-59}$$

To obtain the influence of rotation of the Earth projected into NED, following transformation equation applies [23 p. 52]:

$$\boldsymbol{\omega}_{ie}^n = \boldsymbol{C}_e^n \boldsymbol{\omega}_{ie}^e = \begin{bmatrix} \omega_e \cos(\varphi) \\ 0 \\ -\omega_e \sin(\varphi) \end{bmatrix} \tag{4-60}$$

Another important rate that needs to be defined is the *transport rate;* the rate of rotation of the NED with respect to the ECEF and expressed in NED. This transport rate is given by the velocities of motion in northern v_N and eastern v_E direction and the radii of curvature of the Earth [23 p. 52]:

$$\boldsymbol{\omega}_{en}^n = \begin{bmatrix} v_E/(R_N + h) \\ -v_N/(R_M + h) \\ -v_E \tan(\varphi)/(R_N + h) \end{bmatrix} \tag{4-61}$$

BODY FRAME

The body frame (BF) is defined as the frame coinciding with the sensors sensing axes, i.e. the frame in which the accelerations and angular rates are generated by the inertial sensors. If the BF is not fixed in both origin and orientation to the rigid navigated vehicle, another frame has to be introduced, usually noted as v-frame (vehicle frame) [32]. The orientation of the body coordinate axes is fixed in the shape of body. For example assume the body is an aircraft:

- The X-axis points forward through the nose of the aircraft.
- The Y-axis points to the right of the X-axis, perpendicular to the X-axis.
- The Z-axis points down through the bottom of the aircraft, perpendicular to the X-Y plane and satisfying the right-hand rule (hence, in case of stationary horizontal alignment the accelerometer measures negative gravity value in the Z-axis).

The DCM describing the transformation of the measured accelerations and angular rates from BF to the NED is as follows [23 p. 45]:

$$\boldsymbol{C}_b^n = \begin{bmatrix} \cos(\theta)\cos(\psi) & -\cos(\rho)\sin(\psi) + \sin(\rho)\sin(\theta)\cos(\psi) & \sin(\rho)\sin(\psi) + \cos(\rho)\sin(\theta)\cos(\psi) \\ \cos(\theta)\sin(\psi) & \cos(\rho)\cos(\psi) + \sin(\rho)\sin(\theta)\sin(\psi) & -\sin(\rho)\cos(\psi) + \cos(\rho)\sin(\theta)\sin(\psi) \\ -\sin(\theta) & \sin(\rho)\cos(\theta) & \cos(\rho)\cos(\theta) \end{bmatrix} \tag{4-62}$$

The rotations defined in BF obey the right-hand rotation rule: positive roll with respect to (w.r.t.) x-axis for y-axis moving downwards, positive pitch w.r.t. y-axis for x-axis moving upwards, positive yaw w.r.t. z-axis for turning clockwise.

4.4.3 INITIAL ALIGNMENT ALGORITHMS FOR INS

INS initial alignment is a mathematical procedure for determining the initial attitude information between the body frame and the navigation frame [23], [25]. There exist various algorithms for initial alignment, each suitable under different circumstances. These algorithms can be divided into [32]; see **Table 4-3**:

- **coarse / fine** alignment according to the amount of attitude error that has to be dealt with,
- **static / in-motion** alignment according to the dynamics of the navigated object.

	In-motion Alignment	Static Alignment
Coarse Alignment	Velocity matching	Levelling/gyro compassing
	EKF with large heading uncertainty model	Analytic
Fine Alignment	EKF with small heading uncertainty model	

Table 4-3 Different initial alignment procedures for INS

For example the algorithm for static coarse alignment is described by Britting [80] and Shin [81]. It is an analytic approach based on solving a two-vector measurement problem exploiting gravity and the Earth rotation measurements in one step according to gyro-compassing equation introduced in [81 pp. 68-72]:

$$\boldsymbol{C}_b^n = \begin{bmatrix} \frac{-\tan(\varphi)}{g} & \frac{1}{\omega_e \cos(\varphi)} & 0 \\ 0 & 0 & \frac{-1}{g\omega_e \cos(\varphi)} \\ \frac{-1}{g} & 0 & 0 \end{bmatrix} \begin{bmatrix} \left(\boldsymbol{a}^b\right)^T \\ \left(\boldsymbol{\omega}_{ib}^b\right)^T \\ \left(\boldsymbol{a}^b \times \boldsymbol{\omega}_{ib}^b\right)^T \end{bmatrix} \tag{4-63}$$

where $\boldsymbol{a}^b = [a_x\ a_y\ a_z]^T$ is the accelerometer measurements vector, g is the gravity (i.e. g^{nz}), ω_e is the Earth's rate (i.e. $\boldsymbol{\omega}_{ie}^{ez}$) and φ is latitude.

Coarse alignment is a straightforward method that requires gyroscopes or angular rate sensors with small biases and high signal to noise ratio. Low cost MEMS sensors are not suitable in this case and hence this alignment procedure cannot be performed in the stationary mode [32 p. 42]. Another flaw of this method is that the transformation matrix obtained usually does not satisfy the orthogonality and normality condition; hence, resolution using Euler angles needs to be implemented as well [81 p. 69].

During static alignment (levelling) only roll and pitch values can be determined from the static accelerometer measurements; external heading measurements using magnetic compass or a velocity matching alignment technique has to be implemented [81]. This static alignment by means of levelling is given by the following equations [32 p. 43]:

$$\rho = \text{sign}(a_z)\tan^{-1}\left(\frac{a_y}{a_z}\right) \tag{4-64}$$

$$\theta = -\text{sign}(a_z)\sin^{-1}\left(\frac{a_x}{g}\right) \tag{4-65}$$

where function $sign(\cdot)$ gives the sign value.

If the actual heading information is unknown and unavailable, then the EKF with a large heading uncertainty model is used for a coarse alignment. To further enhance the alignment precision, EKF with small heading uncertainty model is used at last – the fine alignment. If in-motion alignment is required, a consistent and robust KF has to be developed, such as proposed in [82]. To further improve the precision of alignment, data de-noising procedure, such as the WMRA introduced in chapter 4.3, can be applied and significant improvement in precision can be expected [83].

The actual implementation of the initial alignment used in this dissertation is based on both the *coarse alignment* for the case, when the initial heading is unknown, i.e. solving the matrix equation (4-63), and on the *static alignment* for the case, when the initial heading is at disposal. The static alignment was implemented by solving the following matrix equation with the equation solver from the *Symbolic Math Toolbox* for MATLAB (see **Matlab 8-4** in Appendix 8.5). Pitch and roll values are the computed variables, the yaw angle value is obtained from an external aiding source; referential gravity value is determined as well. The equation is as follows:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \boldsymbol{C}_n^b \begin{bmatrix} 0 \\ 0 \\ g^{nz} \end{bmatrix} = \begin{bmatrix} g^{nz}\sin(\theta) \\ -g^{nz}\sin(\rho)\cos(\theta) \\ -g^{nz}\cos(\rho)\cos(\theta) \end{bmatrix} \tag{4-66}$$

where $\boldsymbol{g}^n = f(latitude,\ altitude)$, ρ is roll angle, θ is pitch angle, $\boldsymbol{C}_n^b$ can be obtained by transposition of $\boldsymbol{C}_b^n$ and a_x, a_y, a_z are the body frame accelerometer measurements, ideally mean values of a static set of acceleration measurements.

4.4.4 STRAPDOWN MECHANIZATION SCHEME

Strapdown mechanization (or INS mechanization) is a process of determining the navigation states (position, velocity and, attitude) from the raw inertial measurements (accelerations $\boldsymbol{a}_b$ and angular rates $\boldsymbol{\omega}_{ib}^{b}$) by solving differential equations describing the system motion dynamics [34 p. 59]. The raw inertial data are measured with respect to the inertial frame (ECI) and expressed as seen by an observer in body frame. Each approach to INS mechanization is characterized by a set of differential equations for which the solution is sought usually in the local level navigation frame (NED in this case); the ECEF frame is used as a transition frame for the position update. INS mechanization algorithm developed in this dissertation is based on the theoretical proposal in [32 pp. 29 - 36], however modified and optimized for modular and easy implementation. The differential equations describing the INS mechanization are as follows [32 p. 29]:

$$\dot{\boldsymbol{v}}^n = \boldsymbol{C}_b^n \boldsymbol{a}^b + \boldsymbol{g}^n - (2\boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n) \times \boldsymbol{v}^n \tag{4-67}$$

$$\dot{\boldsymbol{C}}_n^e = \boldsymbol{C}_n^e (\boldsymbol{\omega}_{en}^n \times) \tag{4-68}$$

$$\dot{h} = -v_D \tag{4-69}$$

$$\dot{\boldsymbol{C}}_b^n = \boldsymbol{C}_b^n (\boldsymbol{\omega}_{ib}^b \times) - (\boldsymbol{\omega}_{in}^n \times) \boldsymbol{C}_b^n \tag{4-70}$$

$$where\ \boldsymbol{\omega}_{in}^n = \boldsymbol{\omega}_{ie}^n + \boldsymbol{\omega}_{en}^n \tag{4-71}$$

Although Savage [75] suggests two-speed realization of the INS mechanization algorithm, single speed implementation was developed due to high performance hardware and low sampling rate inertial sensors. The digital output of all attitude and heading reference systems tested, i.e. 3DM-GX2, AHRS M3, MT9, and MTi-OEM, was at 100Hz. The quaternion approach was chosen due to its computational stability and reliability [25], [75], and [76]. The classical conceptual structure of the algorithm is given by the scheme in **Figure 4-14**; for the actual step-by-step implementation see chapter 5.2.

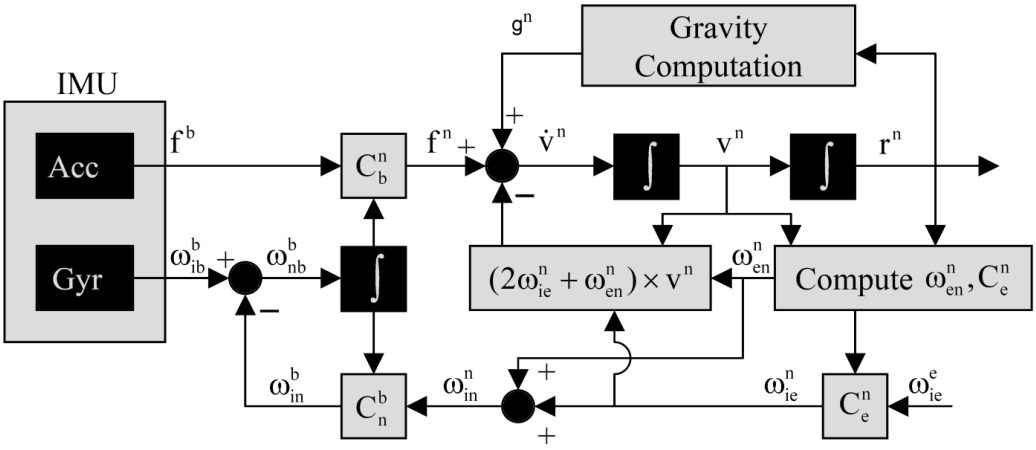


Figure 4-14 Classical conceptual scheme of INS mechanization defined for local navigation frame [81 p. 25]

4.4.5 COMPENSATION OF DETERMINISTIC SENSOR ERRORS

The inertial sensor outputs are distorted by errors that have two parts in general (for more details see chapter 4.2.2): the static (deterministic) part and the varying (temperature dependant and random) part. The deterministic sensor errors (the static part of the sensor bias, scale factor, and sensing axis misalignment) can be compensated during the INS mechanization procedure according to an *observation equation*. The random errors have to be modelled and estimated using the KF. The importance of bias compensation arises when considering an unaided INS. The estimate in position in the navigation frame is obtained after three integration steps in time t: the first to compute the Euler angles from angular rate measurements, and the second and third when double integrating the acceleration measurements to obtain position. Therefore, the error in position estimate due to any uncompensated bias of the angular rate sensors will be proportional to t^3 [20 p. 733] and in the accelerometers to t^2. Additionally, this error will drift randomly in accordance to the sensor noise type present in the IMU readings; typically a random walk, which is proportional to $\sqrt{t}$. The detailed description of the deterministic sensor error compensation is given in [84 pp. 79 - 109] and [85]. The mentioned observation equation for accelerometers is defined as [85]:

$$\boldsymbol{a}^p = \boldsymbol{C}_a^p \boldsymbol{SF}_a(\boldsymbol{a}^a - \boldsymbol{b}^a) = \begin{pmatrix} 1 & -\alpha_{yz} & \alpha_{zy} \\ \alpha_{xz} & 1 & -\alpha_{zx} \\ -\alpha_{xy} & \alpha_{yx} & 1 \end{pmatrix} \begin{pmatrix} SF_{ax} & 0 & 0 \\ 0 & SF_{ay} & 0 \\ 0 & 0 & SF_{az} \end{pmatrix} \left(\begin{pmatrix} a_{ax} \\ a_{ay} \\ a_{az} \end{pmatrix} - \begin{pmatrix} b_{ax} \\ b_{ay} \\ b_{az} \end{pmatrix} \right) \qquad (4\text{-}72)$$

where: $\boldsymbol{a}^p = [a_{px} \quad a_{py} \quad a_{pz}]^T$ is the vector of acceleration estimates in the orthogonal sensor frame called the *platform frame*, $\boldsymbol{a}^a = [a_{ax} \quad a_{ay} \quad a_{az}]^T$ is the vector of measured accelerations in the *accelerometer frame*, $\boldsymbol{SF}_a$ is the diagonal matrix containing the scale factors $SF_{ax}, SF_{ay}, SF_{az}$, and $\boldsymbol{b}^a = [b_{ax} \quad b_{ay} \quad b_{az}]^T$ is the vector of accelerometer biases. The matrix $\boldsymbol{C}_a^p$ provides transformation from the non-orthogonal accelerometer frame to the orthogonal platform frame with the non-diagonal terms α_{ij} (for $i \neq j$) being the axis misalignment coefficients as defined in **Figure 4-15**, [85 p. 2].

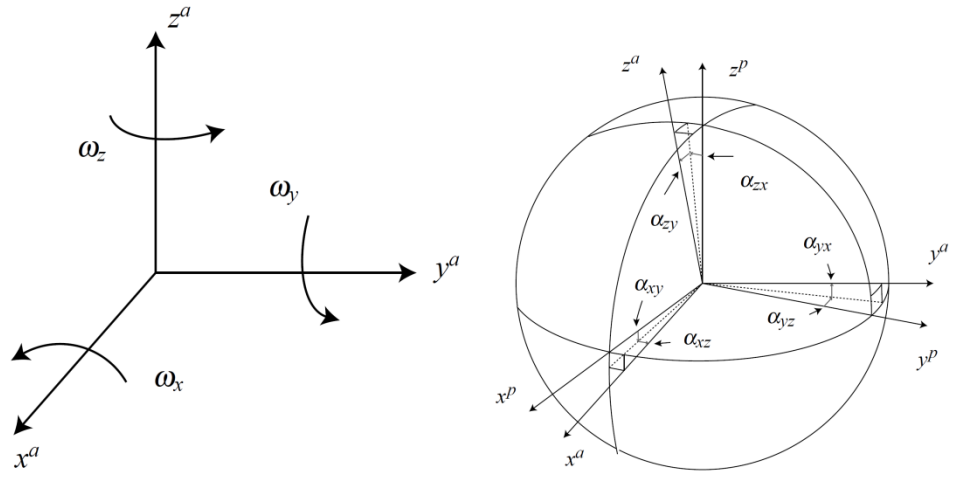


Figure 4-15 Definition of the misalignment angles between the non-orthogonal and the orthogonal sensor frame, i.e. the platform frame (right); definition of accelerometer and gyros sensing axes (left) [85 p. 2].

The values of α_{ij} correspond to the rotation of the i^{th} accelerometer sensitivity axis along the j^{th} platform axis. By defining the platform coordinate system so that the platform coordinate axis x_p coincides with the x_a accelerometer sensitivity axis, and so that the y_p axis is lying in the plane spanned by x_a and y_a, the angles of $\alpha_{xz}; \alpha_{xy}; \alpha_{yx}$ become zero. Hence, the transformation matrix is reduced to:

$$\boldsymbol{C}_a^p = \begin{pmatrix} 1 & -\alpha_{yz} & \alpha_{zy} \\ 0 & 1 & -\alpha_{zx} \\ 0 & 0 & 1 \end{pmatrix} \tag{4-73}$$

In a similar way, the observation equation for angular rate sensor errors compensation is defined using the same approach as for deriving the observation equation for accelerometers [85]:

$$\boldsymbol{\omega}_{ip}^p = \boldsymbol{C}_o^p \boldsymbol{C}_g^o \boldsymbol{SF}_g \left(\boldsymbol{\omega}_{ig}^g - \boldsymbol{b}^g\right) = \boldsymbol{C}_o^p \begin{pmatrix} 1 & -\alpha_{yz} & \alpha_{zy} \\ 0 & 1 & -\alpha_{zx} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} SF_{ax} & 0 & 0 \\ 0 & SF_{ay} & 0 \\ 0 & 0 & SF_{az} \end{pmatrix} \left(\begin{pmatrix} a_{ax} \\ a_{ay} \\ a_{az} \end{pmatrix} - \begin{pmatrix} b_{gx} \\ b_{gy} \\ b_{gz} \end{pmatrix} \right) \tag{4-74}$$

where: $\boldsymbol{\omega}_{ip}^p$ is the vector of angular rate estimates in the orthogonal system, $\boldsymbol{\omega}_{ig}^g$ is the vector of measured angular rates in the *non-orthogonal gyro frame*, $\boldsymbol{SF}_g$ is the diagonal matrix containing the scale factors $SF_{gx}, SF_{gy}, SF_{gz}$, and $\boldsymbol{b}^g = [b_{gx} \quad b_{gy} \quad b_{gz}]^T$ is the vector of angular rate sensor biases. The matrix $\boldsymbol{C}_g^o$ provides transformation from the non-orthogonal gyro frame to the orthogonal gyro frame.

Additional transformation is necessary in the case of angular rates when comparing the equation (4-74) to the observation equation for accelerometers (4-72). This transformation is represented by a directional cosine matrix $\boldsymbol{C}_o^p$ that transforms the angular rates from the orthogonal gyro frame to the platform frame as defined for accelerometers [85].

To find out the desired deterministic sensor errors, more specifically the sensor bias, scale factor and axis misalignment angles, i.e. angles of non-orthogonality [25], system calibration needs to be performed prior to the navigation process. There are various calibration methods each based on slightly different approach. Most of the approaches define a calibration model and according to a selected criterion they estimate its optimal parameters such that the variance of the calibrated data is minimized, usually in a nonlinear least squares sense. The most straightforward method for implementation is the *Thin Shell* method [78 pp. 205 - 210] and a calibration method based on the *Levenberg-Marquardt Algorithm* (LMA) [86 pp. 24 - 29]. There exist other approaches as well; for example the calibration procedure for low-cost IMU evaluated by Skog in [85], procedure for calibration of a miniature AHRS with magnetometer developed by Jurman [87], or a procedure proposed by Salychev [25 pp. 122 - 132] based on combined raw data indications of a SINS and velocity indications in the navigation mode. Calibration results used throughout this dissertation are based on the LMA approach implemented according to the pseudo-code proposed in [86 p. 37].

Details regarding the actual implementation are not subject of this dissertation since the MT9 and MTi-OEM (Xsens) navigation units, used for navigation field-testing, already provide raw data calibrated according to the compensation model introduced in this chapter. Both the AHRS M3 unit (Innalabs) and 3DM-GX2 unit (MicroStrain) were calibrated using the LMA for the purpose of sensor noise modelling and data de-noising.

4.4.6 AIDING OPTIONS AND INTEGRATION SCHEMES FOR INS

To design and evaluate the estimation methods for INS it is crucial to plan the basic architecture in the first place and then define the concept on which the system model will be constructed. This is given by the scheme for systems integration and data fusion, such as the conventional INS/GPS [88 p. 226], or by using any of alternative approaches as in [89]. INS aiding is usually categorised according to the type of systems integration [34]. In general, there are two basic types: first depends on the architecture of the system and second is given by the actual method of data fusion [34]. For the purpose of explanation of various integration schemes, INS integration with GPS will be assumed as referential example. In practice one can encounter various types of aiding based on:

- magnetometers [50] and more complex azimuth-level detectors [7],
- radio navigation systems and radio frequency identification tags (RFID) [22 p. 104],
- odometers based on wheel encoders,
- ultrasonic sensors for vehicle navigation [11],
- database aiding using terrain map matching,
- nonholonomic constraints for land vehicles [20],
- visual aiding using image processing methods and optic flow measurements,
- laser range scanners [8] and finders [9] - aiding based on surroundings mapping.

Furthermore, the integration approach can be categorized by the extent to which data from each component aid the others: the loosely-coupled and the tightly-coupled integration approaches.

LOOSELY–COUPLED INTEGRATION APPROACH

Loosely–coupled integration is characterized by following features [25 pp. 194 - 196], [34 p. 66]:

- independent generation of navigation solutions (position, velocity, attitude) by both the GPS and the INS alone; solutions are subsequently combined in the KF to provide filtered results,
- the differences between the GPS and inertial solutions are fed back in form of estimated errors to carry out the recalibration,
- state vector is smaller when compared to tightly-coupled integration allowing faster signal processing and lower demands on the hardware [78 p. 328],
- it is usually implemented with higher quality inertial sensors (navigation or tactical grade) if the GPS outages are expected to be long in duration.
- Lower quality inertial sensors (consumer or automotive grade) are suited for applications where GPS outages are infrequent and short in duration.
- One of the benefits of loosely coupled integration is that the integrated navigation solution tends to have higher bandwidth and better noise characteristics than the GPS solution.
- Usually it is implemented as decentralized, error model based EKF, such that the measurement vector for EKF consists of differences in position and velocity vectors of the INS and position and velocity vectors of the GPS.
- The main disadvantage is the cascade realization of the filters that causes the measurement error to possibly become time correlated [78 p. 328]; hence, allowing only suboptimal solution.

TIGHTLY–COUPLED INTEGRATION APPROACH

Tightly–coupled system integration has the following features [25 p. 197], [34 p. 67]:

- GPS system provides pseudoranges, Doppler or carrier phase measurements that are fused directly with the navigation solution provided by inertial sensors.
- Real-time feedback of INS velocities to the GPS receiver enables an accurate prediction of GPS pseudorange and phase at the following epoch, thus allowing a smaller bandwidth of the receiver's tracking loop in a high-dynamic environment with subsequent increase in accuracy.
- More complex concept implies larger state vector and hence higher hardware demands.
- It is more accurate than loosely-coupled since the basic GPS observables are not possibly as correlated as the position and velocity solutions used in the loosely-coupled integration.
- Offers possibility to implement fault detection and isolation scheme for verification of the quality of pseudorange or Doppler measurements.

CENTRALIZED/DECENTRALIZED PROCESSING APPROACH

Centralized processing approach is usually associated with tightly-coupled system integration since all the raw sensor output data are combined preferably using one central processor [34]. On the other hand, decentralized processing approach is based on sub sequential combination of results obtained from a number of individual systems, each with a different degree of optimality. If both approaches are constructed and implemented correctly by the means of error propagation, in the final, same solutions should be obtained. For the cases of system fault detection, isolation, and correction decentralized approach is preferable due to its computational simplicity [34]. Comparison of the centralized and decentralized architecture of INS/GPS integration schemes is concluded in the **Table 4-4** [34 p. 71]:

Decentralized INS/GPS Integration	Centralized INS/GPS Integration
GPS receiver, inertial sensors, Kalman filter	GPS receiver, inertial sensors, Kalman filter
Position and velocity from GPS are updated every 1 – 10 seconds	Position and velocity from GPS are updated from each satellite at 1 HZ
No solution if less than 4 GPS satellites are visible	Optimal use of the number of satellites available
Easier to jam	Due to reduced tracking bandwidth better jamming resistance
Low performance in case of high dynamics (receiver may lose lock and take long time to relock on phase)	Reliable tracking in case of high dynamics (fast relock on phase)
Feasible multi-sensor aiding	Difficult augmentation with other sources
Smaller filter size	Larger filter size

Table 4-4 Centralized and decentralized INS/GPS integration schemes [34 p. 71]

OPEN/CLOSED LOOP STATE ESTIMATION APPROACH

Beside the type of integration architecture and the processing approach type, there exists another way how to differentiate the INS aiding – the way of estimating the state vector values. The state vector estimation can be implemented either in a closed loop or open loop [34], depending whether or not there is a error feedback providing corrections to the measurements, i.e. to the INS mechanization. The open loop is usually implemented using the LKF where the INS output is the nominal trajectory used for linearization [78 p. 325]. On the other hand, the closed loop is implemented using the EKF where the linearization is performed with respect to the last estimate used to provide the feedback [78 p. 325]. In general, the closed-loop is meant to provide better performance [34].

The actual simplified schemes representing different approaches to INS/GPS integration in a more straightforward manner are presented in figures: **Figure 4-16**, **Figure 4-17**, **Figure 4-18**, and **Figure 4-19**. An overall conclusion regarding the INS/GPS fusion, showing the major advantages and disadvantages, is presented in **Table 4-5**, according to [35 pp. 27-2]. For more details on different concepts of INS aiding see [25], [90] and [91].

INS	GPS	INS/GPS
High position and velocity accuracy over short term	High position and velocity accuracy over long term	High position and velocity accuracy over long term
Accuracy decreasing with time	Uniform accuracy, independent of time	High data rate
Affected by gravity	Not sensitive to gravity	Navigation output during GPS signal outages
High measurement output rate	Low measurement output rate	Precise attitude determination
Autonomous	Non-autonomous	

Table 4-5 Characteristics of INS, GPS and integrated INS/GPS systems [35 pp. 27-2]

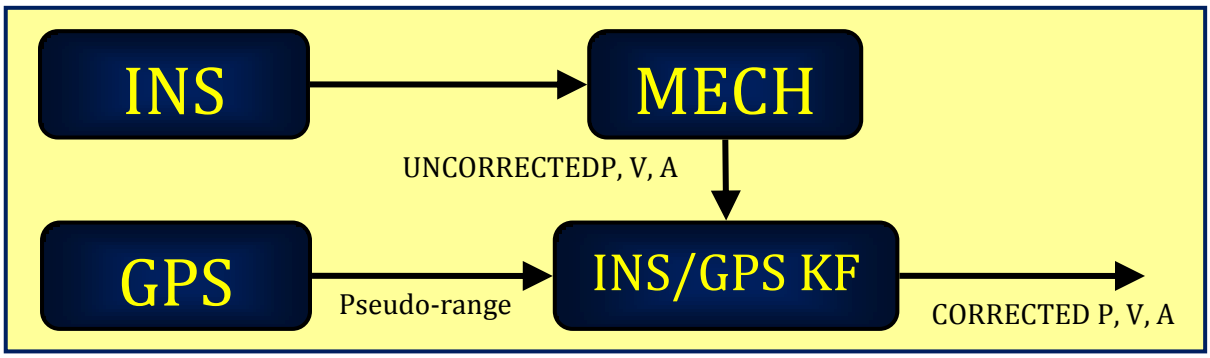


Figure 4-16 Centralized INS/GPS integration – open loop [34 p. 69]

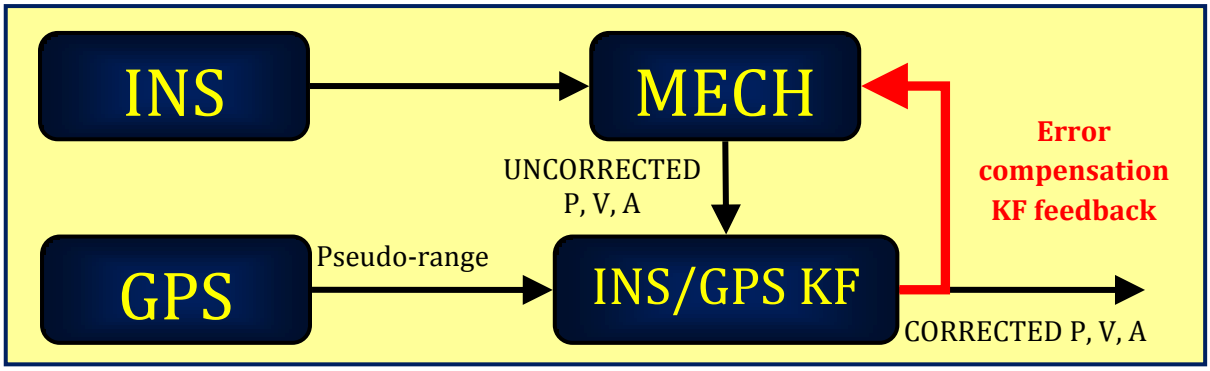


Figure 4-17 Centralized INS/GPS integration – closed loop [34 p. 70]

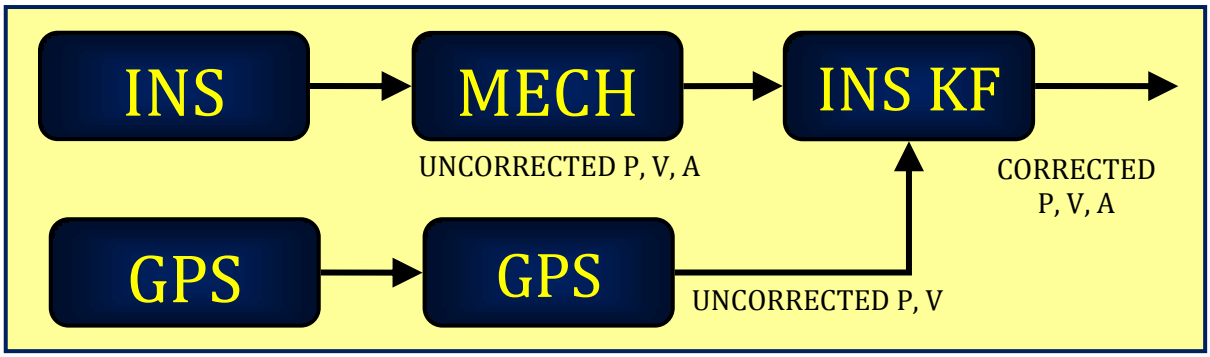


Figure 4-18 Decentralized INS/GPS integration – open loop [34 p. 70]

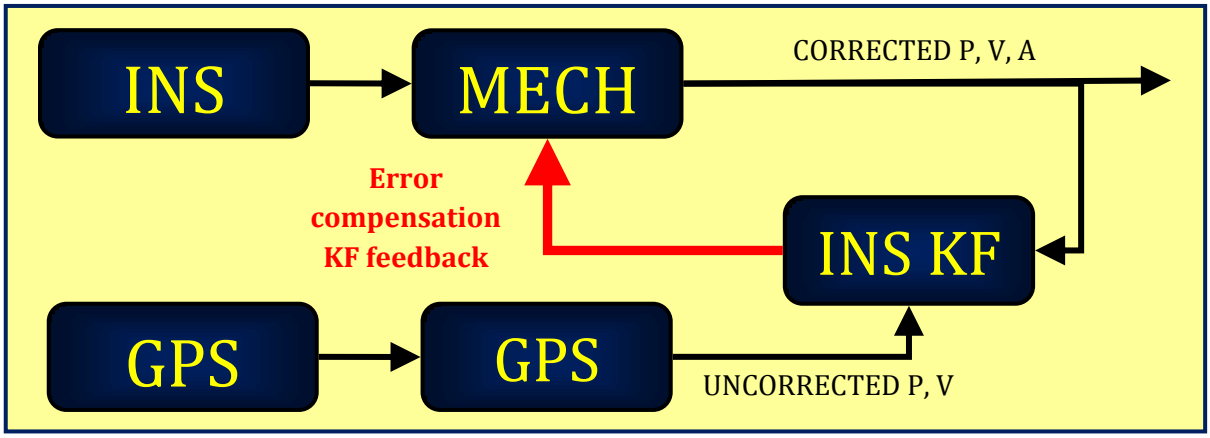


Figure 4-19 Decentralized INS/GPS integration – closed loop [34 p. 71]

4.5 ADAPTIVE FILTERING METHODS FOR INERTIAL NAVIGATION

Kalman filtering is primarily a procedure for combining noisy sensor outputs to estimate the state vector of a system with dynamics that needs to be modelled as precisely as possible [26]. The system state vector includes any system variables as well as inner variables for modelling of time-correlated noise sources. The KF is also a tool for finding an optimal combination of sensors for a system since it can be used to determine the suitability of each sensor based on the covariance analysis. Covariance analysis can be performed even without real data, based just on the sensor noise parameters given by the manufacturer and the proposed state space model. Seeking optimal solution, the KF is however "optimal" if and only if the physical world and the mathematical model coincide to each other [26]. Major task of a KF designer is to deal with the model disparities. There are two major factors that have to be considered:

Firstly, it is the *computational cost,* which grows as cube of the number of state variables and linearly with sampling rate [26]. KF performance and optimization are therefore important issues that inevitably lead to suboptimal modelling techniques (see chapter 4.7).

Secondly, it is the *numerical stability* of the KF. The more complex the filter is (more states), the more grows the risk. When dealing with numerical stability it is always a question of either bad modelling or bad implementation of the model. There are various procedures, for e.g. Square Root filtering [26 p. 238] or UD Filters [26 p. 238], to solve the implementation problems by modifying the KF algorithm. The actual error modelling is and always will be case specific. In general, there exist three possible ways how the KF can be exploited (see **Figure 4-20**), and all of them respect the basic block scheme for the discrete KF [26 p. 122] (see **Figure 4-21**):

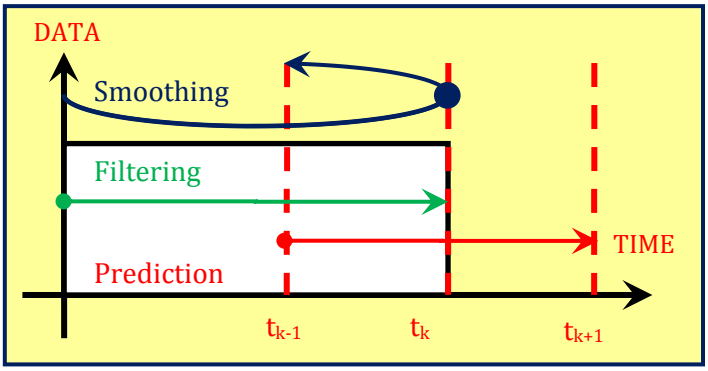


Figure 4-20 Time diagram showing prediction, filtering and smoothing using the Kalman filter

1. **Prediction –** during the process of prediction only a priori system observations are used for the system states estimation, i.e. the only data used are those measured in time-steps prior to the observation update [26 p. 116].
2. **Filtering –** during the process of filtering both a priori system observations as well as observations at the time of the actual estimation are used. Availability of measured data at the time of estimation is necessary condition for the real-time filtering [26 p. 116].
3. **Smoothing –** it is a method based on a posteriori system observations and hence usually applied as post-processing. In practice it consists of two KF algorithms operating in opposite directions in the time domain to provide smoothed estimates [26 p. 116].

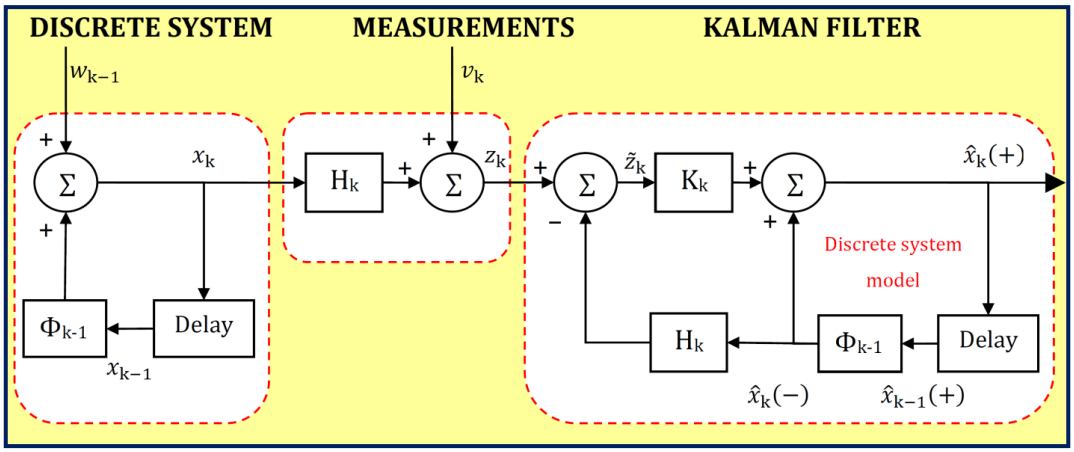


Figure 4-21 Block scheme of the discrete linearized Kalman filter operation [26 p. 122]

The KF equations that are used in this dissertation are based on the KF definition introduced by Andrews and Grewal [26 p. 116]. The KF algorithm will be given without proof as follows [26 p. 121], with notation given in chapter 4.1:

- **The discrete-time system model and measurement model:**

$$\boldsymbol{x}_k = \boldsymbol{\Phi}_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1}\,,\ \boldsymbol{w}_k \sim N(0, \boldsymbol{Q}_k) \tag{4-75}$$

$$\boldsymbol{z}_k = \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{v}_k\,,\ \boldsymbol{v}_k \sim N(0, \boldsymbol{R}_k) \tag{4-76}$$

- **Initial conditions:**

$$\hat{\boldsymbol{x}}_0 = \mathrm{E}\langle \boldsymbol{x}_0 \rangle \quad \boldsymbol{P}_0 = \mathrm{E}\langle \boldsymbol{x}_0 \hat{\boldsymbol{x}}_0{}^T \rangle \tag{4-77}$$

- **Assumption of independence:**

$$\mathrm{E}\langle \boldsymbol{w}_k \boldsymbol{v}_j{}^T \rangle = 0 \; for\ all\ k\ and\ j \tag{4-78}$$

- **Extrapolation of the state estimation:**

$$\hat{\boldsymbol{x}}_k(-) = \boldsymbol{\Phi}_{k-1}\hat{\boldsymbol{x}}_{k-1}(+) \tag{4-79}$$

- **System state estimation:**

$$\hat{\boldsymbol{x}}_k(+) = \hat{\boldsymbol{x}}_k(-) + \boldsymbol{K}_k[\boldsymbol{z}_k - \boldsymbol{H}_k\hat{\boldsymbol{x}}_k(-)] \tag{4-80}$$

- **A priori covariance matrix estimation:**

$$\boldsymbol{P}_k(-) = \boldsymbol{\Phi}_{k-1}\boldsymbol{P}_{k-1}(+)\boldsymbol{\Phi}_{k-1}{}^T + \boldsymbol{Q}_{k-1} \tag{4-81}$$

- **A posteriori covariance matrix estimation:**

$$\boldsymbol{P}_k(+) = [\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k]\boldsymbol{P}_k(-) \tag{4-82}$$

An alternative to the covariance estimate is called the *Joseph form* and can be used in KF/LKF/EKF algorithms to yield more stable solution due to the guaranteed symmetry of $\boldsymbol{P}$ [26 p. 120]. Such a solution is less vulnerable to numerical errors:

$$\boldsymbol{P}_k(+) = [\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k]\boldsymbol{P}_k(-)[\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k]^T + \boldsymbol{K}_k\boldsymbol{R}_k\boldsymbol{K}_k^T \tag{4-83}$$

- **Kalman gain computation:**

$$\boldsymbol{K}_k = \boldsymbol{P}_k(-)\boldsymbol{H}_k{}^T\left[\boldsymbol{H}_k\boldsymbol{P}_k(-)\boldsymbol{H}_k{}^T + \boldsymbol{R}_k\right]^{-1} \tag{4-84}$$

The four-step Kalman filter algorithm can be implemented as follows (see **Matlab 8-5** in Appendix 8.5)**:**

1. Compute $\boldsymbol{P}_k(-)$ from the values of $\boldsymbol{P}_{k-1}(+)$, $\boldsymbol{\Phi}_{k-1}$ and $\boldsymbol{Q}_{k-1}$.
2. Compute Kalman gain $\boldsymbol{K}_k$ from the values of $\boldsymbol{P}_k(-)$ (step 1), $\boldsymbol{H}_k$ and $\boldsymbol{R}_k$.
3. Compute $\boldsymbol{P}_k(+)$ from the values of the Kalman gain $\boldsymbol{K}_k$(step 2) and $\boldsymbol{P}_k(-)$(step 1).
4. Compute successive values of $\widehat{\boldsymbol{x}}_k(+)$ recursively using the computed values of the Kalman gain $\boldsymbol{K}_k$(step 3), given the initial estimate of $\widehat{\boldsymbol{x}}_0$, and the filter input data $\boldsymbol{z}_k$.

4.5.1 THE LINEARIZED KALMAN FILTER

There are two most common modifications of the conventional KF algorithm that can deal with nonlinearities, the linearized KF (LKF) and the extended KF (EKF). The LKF is optimal for applications, where values of state variables are fairly known and only small perturbations occur due to process noise and measurement noise. For these applications, the estimation problem can effectively be linearized about this nominal data trajectory. Hence, Kalman gain can be pre-computed to save the computational load. Assuming functions $\boldsymbol{f}$ and $\boldsymbol{h}$ are continuously differentiable, the LKF algorithm according to [26 p. 179] is formulated:

- **Non-linear model of the nominal trajectory:**

$$\boldsymbol{x}_k^{nom} = \boldsymbol{f}_{k-1}(\boldsymbol{x}_{k-1}^{nom}) \tag{4-85}$$

- **Linearized model:**

$$\delta\boldsymbol{x}_k = \boldsymbol{x} - \boldsymbol{x}^{nom} \tag{4-86}$$

$$\delta\boldsymbol{x}_k \approx \frac{\partial \boldsymbol{f}_{k-1}}{\partial \boldsymbol{x}}_{\boldsymbol{x}=\boldsymbol{x}_{k-1}^{nom}} \delta\boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1}\,,\ \boldsymbol{w}_k \sim N(0, \boldsymbol{Q}_k) \tag{4-87}$$

- **Non-linear measurement model:**

$$\boldsymbol{z}_k = \boldsymbol{h}_k(x_k) + \boldsymbol{v}_k\,,\ \boldsymbol{v}_k \sim N(0, \boldsymbol{R}_k) \tag{4-88}$$

- **Extrapolation of the system state estimation:**

$$\widehat{\delta\boldsymbol{x}}_k(-) = \boldsymbol{\Phi}_{k-1}^{[1]}\widehat{\delta\boldsymbol{x}}_{k-1}(+) \tag{4-89}$$

$$Jacobian\ computation\text{:}\ \boldsymbol{\Phi}_{k-1}^{[1]} \approx \frac{\partial \boldsymbol{f}_{k-1}}{\partial \boldsymbol{x}}_{\boldsymbol{x}=\boldsymbol{x}_{k-1}^{nom}} \tag{4-90}$$

$$Jacobian\ computation\text{:}\ \boldsymbol{H}_k^{[1]} \approx \frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{x}}_{\boldsymbol{x}=\boldsymbol{x}_k^{nom}} \tag{4-91}$$

- **A priori covariance matrix estimation:**

$$\boldsymbol{P}_k(-) = \boldsymbol{\Phi}_{k-1}^{[1]}\boldsymbol{P}_{k-1}(+)\boldsymbol{\Phi}_{k-1}^{[1]\mathrm{T}} + \boldsymbol{Q}_{k-1} \tag{4-92}$$

- **Kalman gain computation:**

$$\boldsymbol{K}_k = \boldsymbol{P}_k(-)\boldsymbol{H}_k^{[1]\mathrm{T}}\left[\boldsymbol{H}_k^{[1]}\boldsymbol{P}_k(-)\boldsymbol{H}_k^{[1]\mathrm{T}} + \boldsymbol{R}_k\right]^{-1} \tag{4-93}$$

- **System state estimation:**

$$\widehat{\delta\boldsymbol{x}}_k(+) = \widehat{\delta\boldsymbol{x}}_k(-) + \boldsymbol{K}_k\left[\boldsymbol{z}_k - \boldsymbol{h}_k(x_k^{nom}) - \boldsymbol{H}_k^{[1]}\widehat{\delta\boldsymbol{x}}_k(-)\right] \tag{4-94}$$

- **A posteriori covariance matrix estimation:**

$$\boldsymbol{P}_k(+) = \left[\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k^{[1]}\right] \boldsymbol{P}_k(-) \tag{4-95}$$

4.5.2 THE EXTENDED KALMAN FILTER

The nominal data trajectory, as mentioned in 4.5.1, can also be determined "on the fly" as the current best estimated of the actual trajectory. This is the principle of the extended KF (EKF). The EKF has the advantage that the perturbations include only the state estimation errors, which are in principle smaller than perturbations from any predefined nominal trajectory and hence better conditioned for linear approximation [26]. However, this advantage is compensated by additional computational cost due to real-time linearization about an unpredicted trajectory. The EKF algorithm according to [26 p. 180] is formulated:

- **Nonlinear model of the system:**

$$\boldsymbol{x}_k = \boldsymbol{f}_{k-1}(\boldsymbol{x}_{k-1}) + \boldsymbol{w}_{k-1}\,,\, \boldsymbol{w}_k \sim N(0, \boldsymbol{Q}_k) \tag{4-96}$$

- **Nonlinear measurement model:**

$$\boldsymbol{z}_k = \boldsymbol{h}_k(\boldsymbol{x}_k) + \boldsymbol{v}_k\,,\, \boldsymbol{v}_k \sim N(0, \boldsymbol{R}_k) \tag{4-97}$$

- **Extrapolation of the system state estimation:**

$$\hat{\boldsymbol{x}}_k(-) = \boldsymbol{f}_{k-1}\big(\hat{\boldsymbol{x}}_{k-1}(+)\big) \tag{4-98}$$

- **Measurement prediction:**

$$\hat{\boldsymbol{z}}_k = \boldsymbol{h}_k\big(\hat{\boldsymbol{x}}_k(-)\big) \tag{4-99}$$

- **Linearized equations** (with respect to the previous system state)**:**

$$Jacobian\ computation\text{:}\ \boldsymbol{\Phi}_{k-1}^{[1]} \approx \frac{\partial \boldsymbol{f}_{k-1}}{\partial \boldsymbol{x}}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1}(-)} \tag{4-100}$$

$$Jacobian\ computation\text{:}\ \boldsymbol{H}_k^{[1]} \approx \frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{x}}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k(-)} \tag{4-101}$$

- **A priori covariance matrix estimation:**

$$\boldsymbol{P}_k(-) = \boldsymbol{\Phi}_{k-1}^{[1]} \boldsymbol{P}_{k-1}(+) \boldsymbol{\Phi}_{k-1}^{[1]\mathrm{T}} + \boldsymbol{Q}_{k-1} \tag{4-102}$$

- **Kalman gain computation:**

$$\boldsymbol{K}_k = \boldsymbol{P}_k(-) \boldsymbol{H}_k^{[1]\mathrm{T}} \left[\boldsymbol{H}_k^{[1]} \boldsymbol{P}_k(-) \boldsymbol{H}_k^{[1]\mathrm{T}} + \boldsymbol{R}_k\right]^{-1} \tag{4-103}$$

- **System state estimation:**

$$\hat{\boldsymbol{x}}_k(+) = \hat{\boldsymbol{x}}_k(-) + \boldsymbol{K}_k[\boldsymbol{z}_k - \hat{\boldsymbol{z}}_k] \tag{4-104}$$

- **A posteriori covariance matrix estimation:**

$$\boldsymbol{P}_k(+) = \left[\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k^{[1]}\right] \boldsymbol{P}_k(-) \tag{4-105}$$

The EKF predictions are approximated as the function of the prior mean value for estimates, i.e. no expectations are taken [92 p. 39]. The covariances are determined by linearizing the dynamic equations; the posterior covariance matrices are determined analytically. Hence, in the EKF the state distribution is approximated by a Gaussian random variable, which is propagated analytically

through the "first-order" linearization of the nonlinear system, i.e. EKF provides first order approximations to the optimal terms [92 p. 39].

When judging the EKF performance, the state approximations can introduce large errors in the true posterior mean and covariance of the transformed (Gaussian) random variable [26]. This may lead to suboptimal performance and sometimes divergence of the filter [32]. Solution can be sought by propagating the Gaussian random variables (estimates of the distributions) through non-liner system equations, i.e. for example as implemented in the UKF [38]. Regarding the proposed EKF implementation see **Matlab 8-6** in Appendix 8.5.

4.5.3 THE SECOND ORDER EXTENDED KALMAN FILTER

The Second order extended Kalman filter (SEKF) is an extension to the EKF developed to reduce the linearization error of the EKF. In SEKF the Taylor series expansion of the system (or process) function $\boldsymbol{f}_k(\boldsymbol{x}_k)$ and measurement function $\boldsymbol{h}_k(\boldsymbol{x}_k)$ is performed up to second order. Hence, the KF equations have to be adjusted accordingly.

The discrete time SEKF algorithm presented in this chapter is based on the theory described in [93 pp. 413-417] and adjusted according to the recently presented issues concerning the SEKF design [94] and performance comparison to the EKF [95]. In general, the SEKF provides superior results to the EKF as expected, however, the actual implementation can cause serious complications for larger complex models. Hessian matrices have to be derived analytically and the computational load increases dramatically due to the increased number of matrix operations. For completeness, the SEKF algorithm is presented such that the extension of the algorithm compared to conventional EKF is highlighted in red:

- **Nonlinear model of the system:**

$$\boldsymbol{x}_k = \boldsymbol{f}_{k-1}(\boldsymbol{x}_{k-1}) + \boldsymbol{w}_{k-1}\,,\ \boldsymbol{w}_k \sim N(0, \boldsymbol{Q}_k) \tag{4-106}$$

- **Nonlinear measurement model:**

$$\boldsymbol{z}_k = \boldsymbol{h}_k(\boldsymbol{x}_k) + \boldsymbol{v}_k\,,\ \boldsymbol{v}_k \sim N(0, \boldsymbol{R}_k) \tag{4-107}$$

- **The prediction step –** the computation of the predicted mean and covariance of the state**:**

$$\hat{\boldsymbol{x}}_k(-) = \boldsymbol{f}_{k-1}\big(\hat{\boldsymbol{x}}_{k-1}(+)\big) + \frac{1}{2}\sum_{i=1}^{n} e_i \operatorname{trace}\left\{\boldsymbol{\Phi}_{k-1}^{[2,i]}\boldsymbol{P}_{k-1}(+)\right\} \tag{4-108}$$

$$\boldsymbol{P}_k(-) = \boldsymbol{\Phi}_{k-1}^{[1]}\boldsymbol{P}_{k-1}(+)\boldsymbol{\Phi}_{k-1}^{[1]\mathrm{T}} + \frac{1}{2}\sum_{i,j=1}^{n} e_i e_j^{\mathrm{T}} \operatorname{trace}\left\{\boldsymbol{\Phi}_{k-1}^{[2,i]}\boldsymbol{P}_{k-1}(+)\boldsymbol{\Phi}_{k-1}^{[2,j]}\boldsymbol{P}_{k-1}(+)\right\} + \boldsymbol{Q}_{k-1} \tag{4-109}$$

where n is the dimension of the state vector and i is the corresponding vector state index, $e_i = [0 \ldots 0\ 1\ 0 \ldots 0]^{\mathrm{T}}$ is a unit vector in direction of the coordinate axis i (i.e. equals 1 at position i and 0 otherwise), the product $e_i e_j^{\mathrm{T}}$ is an $n \times n$ matrix whose elements are all zero except for the element in i^{th} row and j^{th} column, the operator trace{·} stands for computation of a trace of a matrix, $\boldsymbol{\Phi}_{k-1}^{[1]}$ is the *Jacobian* and $\boldsymbol{\Phi}_{k-1}^{[2,i]}$ is the *Hessian* of function $\boldsymbol{f}_{k-1}^{[i]}$ which represents the i^{th} element of $\boldsymbol{f}_{k-1}(\boldsymbol{x}_{k-1})$. The Hessian $\boldsymbol{\Phi}_{k-1}^{[2,i]}$ is computed according to:

$$\boldsymbol{\Phi}_{k-1}^{[2,i]} = \frac{\partial^2 \boldsymbol{f}_{k-1}^{[i]}}{\partial \boldsymbol{x}^2}_{\,x=\hat{\boldsymbol{x}}_{k-1}(-)} \tag{4-110}$$

- **The update step –** the computation of the Kalman gain and the measurement residual (innovation) at time t_k in order to update the mean and covariance of the state**:**

$$\boldsymbol{S}_k = \boldsymbol{H}_k^{[1]} \boldsymbol{P}_k(-) \boldsymbol{H}_k^{[1]\mathrm{T}} + \frac{1}{2} \sum_{i,j=1}^{m} e_i e_j^{\mathrm{T}} \operatorname{trace}\left\{ \boldsymbol{H}_k^{[2,i]} \boldsymbol{P}_k(-) \boldsymbol{H}_k^{[2,j]} \boldsymbol{P}_k(-) \right\} + \boldsymbol{R}_k \tag{4-111}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_k(-) \boldsymbol{H}_k^{[1]\mathrm{T}} \boldsymbol{S}_k{}^{-1} \tag{4-112}$$

$$\hat{\boldsymbol{x}}_k(+) = \hat{\boldsymbol{x}}_k(-) + \boldsymbol{K}_k \left[\boldsymbol{z}_k - \boldsymbol{h}_k\left(\hat{\boldsymbol{x}}_k(-)\right) - \frac{1}{2} \sum_{i=1}^{m} e_i \operatorname{trace}\left\{ \boldsymbol{H}_k^{[2,i]} \boldsymbol{P}_k(-) \right\} \right] \tag{4-113}$$

$$\boldsymbol{P}_k(+) = \boldsymbol{P}_k(-) - \boldsymbol{K}_k \boldsymbol{S}_k \boldsymbol{K}_k{}^{\mathrm{T}} \tag{4-114}$$

where m is the dimension of the measurement vector, $\boldsymbol{H}_k^{[1]}$ is the *Jacobian* and $\boldsymbol{H}_k^{[2,i]}$ is the *Hessian* of function $\boldsymbol{h}_k^{[i]}$ which represents the i^{th} element of $\boldsymbol{h}_k(\boldsymbol{x}_k)$. The Hessian $\boldsymbol{H}_k^{[2,i]}$ is computed as follows:

$$\boldsymbol{H}_k^{[2,i]} = \frac{\partial^2 \boldsymbol{h}_k^{[i]}}{\partial \boldsymbol{x}^2}_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k(-)} \tag{4-115}$$

4.5.4 THE UNSCENTED KALMAN FILTER

As introduced in chapter 2.2, the Unscented Kalman filter (UKF) is a new sampling based approach for dealing with nonlinear systems. The fundamental component of this filter is the unscented transformation (UT) which uses a set of carefully chosen weighted points to parameterise the mean and covariance of probability distributions. According to Julier and Uhlman [38 p. 5]: *"The unscented transformation is a new, novel method for calculating the statistics of a random variable which undergoes a nonlinear transformation. It is founded on the intuition that it is easier to approximate a Gaussian distribution than it is to approximate an arbitrary nonlinear function or transformation."*

The UT is used for transforming the sampling points of given mean and covariance of the probability distribution directly through the nonlinear system function. The transformed mean and covariance are then constructed from the transformed points, called sigma points (SP). To develop and implement the UKF, the UT has to be proposed first. UT may then be extended to a recursive estimation problem using the conventional KF equations. When the UKF is used for the INS/GPS integration, it seems to deal with large and small attitude errors seamlessly [32]. However, for attitude expression in Euler angles singularities can possibly occur the same way as in EKF. Therefore, it is necessary to develop UKF based on quaternion attitude representation to resolve this problem [32]. Both the UKF and the UT are thoroughly described and derived by Julier and Uhlman in [37], [38]. Experimental review is provided in [35] where the UKF is compared to particle filter and in [41] where the UKF is compared to the EKF and evaluated on four examples from industrial practice. For thorough description of the UKF algorithm and its modification regarding constraints handling in the state estimation see [42].

Concluded, the UKF can be viewed as a new extension of the Kalman filter to nonlinear equations. For the target tracking purposes (radar applications), it is well proven that this UKF approach outperforms the classical EKF, especially in very noisy environments [92 p. 58]. The major merit of the UKF is its robustness to both process noise and measurement noise [92 p. 57].

UNSCENTED TRANSFORMATION

The UT is an algorithm for obtaining set of weights W_i and sigma points $\boldsymbol{\chi}_i$ from a given mean $\bar{\boldsymbol{x}}$ and covariance $\boldsymbol{P}$, as described in [32 pp. 88-93]. Following conditions must be satisfied for p being the number of generated sigma points:

$$\sum_{i=0}^{p-1} W_i = 1 \tag{4-116}$$

$$\sum_{i=0}^{p-1} W_i \boldsymbol{\chi}_i = \bar{\boldsymbol{x}} \tag{4-117}$$

$$\sum_{i=0}^{p-1} W_i (\boldsymbol{\chi}_i - \bar{\boldsymbol{x}})\,(\boldsymbol{\chi}_i - \bar{\boldsymbol{x}})^T = \boldsymbol{P} \tag{4-118}$$

According to the first UT proposal (as described in [32 p. 88] and firstly introduced by Julier and Uhlman [37]), $p = 2n+1$ of sigma points are generated, where n is the number of system states. Sigma points and weights are then obtained in the following way:

$$\boldsymbol{\chi}_i = \begin{Bmatrix} \bar{\boldsymbol{x}}, i = 0 \\ \bar{\boldsymbol{x}} + \sqrt{n+\kappa}\boldsymbol{\sigma}_i, & i = 1,\dots,n \\ \bar{\boldsymbol{x}} - \sqrt{n+\kappa}\boldsymbol{\sigma}_{i-n}, & i = n+1,\dots,2n \end{Bmatrix} \tag{4-119}$$

$$W_i = \begin{Bmatrix} \dfrac{\kappa}{n+\kappa}, & i = 0 \\ \dfrac{1}{2(n+\kappa)}, & i = 1,\dots,2n \end{Bmatrix} \tag{4-120}$$

where κ is a scaling parameter for adjusting the effects of fourth and higher moments of probability distribution during given nonlinear transformations, and σ_i is the i^{th} column of the square-root of the covariance matrix $\boldsymbol{C}$ calculated by the Cholesky decomposition such that:

$$\boldsymbol{CC}^T = \boldsymbol{P} \tag{4-121}$$

The optimal scaling parameter configuration is $\kappa = 3 - n$ for single-state Gaussian probability distribution systems and $0 < n + \kappa < 3$ for multi-state systems [37]. Julier and Uhlman [38] also proved that for systems with very high sampling frequency the number of sigma points can be reduced to $n + 2$ in order to ease the computational burden. Points are chosen to match the first two moments and minimizing the third moment (skew). Resulting sigma points are called the minimal skew simplex points [38]. However, for multi-dimensional systems the radius of the bounding *hyper-sphere* – the set of *n*-dimensional points with equal Euclidean norm – increases rapidly with the number of states $2^{n/2}$ and SPs are sampled beyond uncertainty level [38]. The original concept of SP generation was hence expanded by introducing the *spherical simplex* sigma points and the spherical simplex UT [32 p. 90], as follows:

THE SCPHERICAL SIMPLEX UNSCENTED TRANSFORMATION:

1. The 0^{th} weight is chosen freely $0 \le W_0 \le 1$. All other weights have the same value:

$$W_i = \frac{(1 - W_0)}{(n+1)} \text{for } i = 1, \dots, n+1 \tag{4-122}$$

2. The 0th point is the same as the mean and all other points lie on a hyper-sphere centred at the mean. The vector sequence of sigma points is initialized as follows:

$$\boldsymbol{\chi}_{u,0}^{1} = [0], \qquad \boldsymbol{\chi}_{u,1}^{1} = \begin{bmatrix} -1 \\ \sqrt{2W_1} \end{bmatrix} \text{ and } \boldsymbol{\chi}_{u,2}^{1} = \begin{bmatrix} 1 \\ \sqrt{2W_1} \end{bmatrix} \tag{4-123}$$

3. The vector sequence of sigma points is then expanded such that *j* denotes the dimension of the vector and *i* is the index in the SP sequence:

$$\boldsymbol{\chi}_{u,i}^{j} = \begin{Bmatrix} \begin{bmatrix} \boldsymbol{\chi}_{u,0}^{j-1} \\ 0 \end{bmatrix} & \text{for } i = 0 \\ \begin{bmatrix} \boldsymbol{\chi}_{u,i}^{j-1} \\ -1/\sqrt{j(j+1)W_1} \end{bmatrix} & \text{for } i = 1, \dots, j \\ \begin{bmatrix} \mathbf{0}^{j-1} \\ j/\sqrt{j(j+1)W_1} \end{bmatrix} & \text{for } i = j+1 \end{Bmatrix} \tag{4-124}$$

4. The SP for an arbitrary mean and covariance are then obtained:

$$\boldsymbol{\chi}_i' = \overline{\boldsymbol{x}} + \boldsymbol{C}\boldsymbol{\chi}_{u,i} \tag{4-125}$$

For high dimensional systems involving attitude estimation the bounding hyper sphere radius expansion may cause serious problem even if the spherical simplex SPs are implemented. For example: for a system with n = 12 and w_0 = 0.5 the radius becomes $\sqrt{n/(1-w_0)}$ = 4.899. That means, if the heading uncertainty is approximately 20°, then the heading will be sampled in the range of ±98° and hence the nonlinearities outside the region may affect the solution. As described in [32 p. 94] this was the reason Julier and Uhlman [38] have developed the concept of a scaled UT. Algorithm for scaling the SPs from the first UT proposal and the spherical simplex UT is as follows:

THE SCALED UNSCENTED TRANSFORMATION ALGORITHM:

1. Scaling factor α, a small positive number of value $10^{-4} \leq \alpha \leq 1$, is implemented into the equations:

$$\boldsymbol{\chi}_i = \begin{Bmatrix} \overline{\boldsymbol{x}}, i = 0 \\ \overline{\boldsymbol{x}} + \alpha\sqrt{n + \kappa}\boldsymbol{\sigma}_i, i = 1, \dots, n \\ \overline{\boldsymbol{x}} - \alpha\sqrt{n + \kappa}\boldsymbol{\sigma}_{i-n}, i = n+1, \dots, 2n \end{Bmatrix} \tag{4-126}$$

Hence, sigma points for an arbitrary mean and covariance are obtained:

$$\boldsymbol{\chi}_i' = \overline{\boldsymbol{x}} + \alpha C\boldsymbol{\chi}_{u,i} \tag{4-127}$$

2. Weights for the mean are adjusted to:

$$W_i^m = \begin{Bmatrix} \frac{(W_0 - 1)}{\alpha^2} + 1 \, , i = 0 \\ \frac{W_i}{\alpha^2} \, , \quad i \neq 0 \end{Bmatrix} \tag{4-128}$$

3. Weights for the covariance are adjusted to:

$$W_i^c = \begin{Bmatrix} \frac{(W_0 - 1)}{\alpha^2} + 2 + \beta - \alpha^2 \, , i = 0 \\ \frac{W_i}{\alpha^2} \, , \quad i \neq 0 \end{Bmatrix} \tag{4-129}$$

where β is a parameter to reduce high order effects; $\beta = 2$ is optimal for Gaussian distributions [32 p. 95]. For scaling the spherical simplex SPs, the scaling parameter α should be chosen as the reciprocal value of the radius of the bounding hyper-sphere so the SPs are sampled in the range of $\pm 1\sigma$. There exist two ways of implementing the UKF given by the character of the process noise:

1. The system model is a nonlinear function $\boldsymbol{f}$ of system state and the process noise is additive:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{w}_k, \boldsymbol{Q}_k = \mathrm{E}\langle \boldsymbol{w}_k \boldsymbol{w}_k^T \rangle \tag{4-130}$$

2. The system model is nonlinear function $\boldsymbol{f}$ of both system state and process noise:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{w}_k), \boldsymbol{Q}_k = \mathrm{E}\langle \boldsymbol{w}_k \boldsymbol{w}_k^T \rangle \tag{4-131}$$

   In this case, system noise vector has to be augmented with the state vector into the system process model to form new state vector:

$$\boldsymbol{x}^a = \begin{bmatrix} \boldsymbol{x} \\ w \end{bmatrix} \text{ and hence } \boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k^a) \tag{4-132}$$

In this chapter implementations for additive system noise will be derived in details, modifications for noise vector augmentation will be mentioned, both without proof. For simplicity that does not bias the filter final performance, following assumptions are made:

- System and measurement noise are uncorrelated (if not, noise de-correlation using Cholesky decomposition is essential and has to be performed at first).
- The nonlinear measurement model is described as:

$$\boldsymbol{z}_{k+1} = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{v}_k, \boldsymbol{R}_k = \mathrm{E}\langle \boldsymbol{v}_k \boldsymbol{v}_k^T \rangle \tag{4-133}$$

UNSCENTED KALMAN FILTER ALGORITHM

Assuming the system with additive noise, the UKF derived for such system will work in three main phases [32 pp. 104-125]: initialization, prediction and measurement update.

INITIALIZATION:

1. The state vector and its covariance are initialized:

$$\hat{\boldsymbol{x}}_{0|0}, \boldsymbol{P}_{0|0} = \mathrm{E}\left\langle \left(\boldsymbol{x}_{0|0} - \hat{\boldsymbol{x}}_{0|0}\right)\left(\boldsymbol{x}_{0|0} - \hat{\boldsymbol{x}}_{0|0}\right)^{T} \right\rangle \tag{4-134}$$

   (NOTE: The first index in the above equation stands for current time, the second stands for the time of the last measurement used during the update.)
2. A set of weights and sigma points $\{W_i, \boldsymbol{\chi}_i\}$ is generated using spherical simplex UT algorithm followed by the scaling process.
3. Scaled SP are computed with the use of the Cholesky decomposition:

$$\boldsymbol{\chi}'_{i,0|0} = \hat{\boldsymbol{x}}_{0|0} + \alpha \boldsymbol{C}_{0|0} \boldsymbol{\chi}_{u,i} \tag{4-135}$$

PREDICTION:

1. The SPs are transformed through the system model:

$$\boldsymbol{\chi}'_{i,k|k-1} = \boldsymbol{g}\left(\boldsymbol{\chi}'_{i,k-1|k-1}\right) \tag{4-136}$$

2. The mean and covariance are computed from the transformed SPs:

$$\hat{\boldsymbol{x}}_{k|k-1} = \sum_{i=0}^{p-1} W_i^m \, \boldsymbol{\chi}'_{i,k|k-1} \tag{4-137}$$

$$\boldsymbol{P}_{k|k-1} = \sum_{i=0}^{p-1} W_i^c \left(\boldsymbol{\chi}'_{i,k|k-1} - \hat{\boldsymbol{x}}_{k|k-1}\right)\left(\boldsymbol{\chi}'_{i,k|k-1} - \hat{\boldsymbol{x}}_{k|k-1}\right)^{T} + \boldsymbol{Q}_{k-1} \tag{4-138}$$

3. The scaled SPs are computed using the Cholesky decomposition to obtain $\boldsymbol{C}_{k|k-1}$:

$$\boldsymbol{\chi}'_{i,k|k-1} = \hat{\boldsymbol{x}}_{k|k-1} + \alpha \boldsymbol{C}_{k|k-1} \boldsymbol{\chi}_{u,i} \tag{4-139}$$

MEASUREMENT UPDATE:

1. The SPs are transformed through the measurement model:

$$\boldsymbol{z}_{i,k|k-1} = \boldsymbol{h}\left(\boldsymbol{\chi}'_{i,k|k-1}\right) \tag{4-140}$$

2. The predicted measurements are computed from the transformed SPs:

$$\hat{\boldsymbol{z}}_{k|k-1} = \sum_{i=0}^{p-1} W_i^m \, \boldsymbol{z}_{i,k|k-1} \tag{4-141}$$

3. The covariance between the states and the measurements is computed:

$$\boldsymbol{P}_{xz,k} = \sum_{i=0}^{p-1} W_i^c \left(\boldsymbol{\chi}'_{i,k|k-1} - \hat{\boldsymbol{x}}_{k|k-1}\right)\left(\boldsymbol{z}_{i,k|k-1} - \hat{\boldsymbol{z}}_{k|k-1}\right)^T \tag{4-142}$$

(NOTE: The above equation can be interpreted as $\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k{}^T$ in the EKF algorithm.)

4. The covariance of the innovation sequence is computed:

$$\boldsymbol{P}_{vv,k} = \sum_{i=0}^{p-1} W_i^c \left(\boldsymbol{z}_{i,k|k-1} - \hat{\boldsymbol{z}}_{k|k-1}\right)\left(\boldsymbol{z}_{i,k|k-1} - \hat{\boldsymbol{z}}_{k|k-1}\right)^T + \boldsymbol{R}_k \tag{4-143}$$

(NOTE: The first term on the right hand side can be interpreted as $\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k{}^T$ in the EKF.)

5. The KF update equations are as follows:

$$\boldsymbol{K}_k = \boldsymbol{P}_{xz,k}\boldsymbol{P}_{vv,k}^{-1} \tag{4-144}$$

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k\left(\boldsymbol{z}_k - \hat{\boldsymbol{z}}_{k|k-1}\right) \tag{4-145}$$

$$\boldsymbol{P}_{k|k} = \boldsymbol{P}_{k|k-1} - \boldsymbol{K}_k\boldsymbol{P}_{vv,k}\boldsymbol{K}_k^{-1} \tag{4-146}$$

6. The scaled SPs are computed using the Cholesky decomposition to obtain $\boldsymbol{C}_{k|k}$:

$$\boldsymbol{\chi}'_{i,k|k} = \hat{\boldsymbol{x}}_{k|k} + \alpha\boldsymbol{C}_{k|k}\boldsymbol{\chi}_{u,i} \tag{4-147}$$

For the case with the non-additive process noise the UKF implementation equations are the same with the exception that state vector augmentation and covariance augmentation are carried out. This is done at every time step before the actual Cholesky decomposition has proceeded in both prediction and update phases:

- **Prediction phase augmentation:**

$$\hat{\boldsymbol{x}}_{k|k-1}^a = \begin{bmatrix}\hat{\boldsymbol{x}}_{k|k-1}\\ \boldsymbol{0}\end{bmatrix}, \boldsymbol{P}_{k|k-1}^a = \begin{bmatrix}\boldsymbol{P}_{k|k-1} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{Q}_k\end{bmatrix} \tag{4-148}$$

- **Update phase augmentation:**

$$\hat{\boldsymbol{x}}_{k|k}^a = \begin{bmatrix}\hat{\boldsymbol{x}}_{k|k}\\ \boldsymbol{0}\end{bmatrix}, \boldsymbol{P}_{k|k}^a = \begin{bmatrix}\boldsymbol{P}_{k|k} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{Q}_k\end{bmatrix} \tag{4-149}$$

Considering the equations presented in this chapter the proposed UKF implementation is given in **Matlab 8-7** in Appendix 8.5.

4.5.5 SMOOTHER FOR KALMAN FILTER

The main purpose of using smoothers is to find an optimal estimate that utilizes all past, current and future measurements [32 p. 83]. Smoothing can be applied in three distinct ways: fixed-point [26 p. 163], fixed-lag [26 p. 164], and fixed-interval [26 p. 162]. Fixed-point smoothing is a real-time approach that utilizes new measurements to improve the estimates relative to specific fixed points in time [32 p. 83]. Fixed-lag smoothing is the enhanced version of fixed point smoothing extended from single point to a short time interval. It is usable in case the delay between the estimates does not affect the desired performance of the estimator. This approach finds its use mainly in communication and telemetry [32 p. 83]. The fixed-interval smoothing cannot be used in real-time since it uses all the data measured. However, it offers the most precise estimation results. Fixed-interval smoothing is mostly useful for any surveying application and post-processing to obtain the best trajectory estimates.

Generally, to operate the smoother for the KF, it is necessary to run the algorithm both in the forward and backward direction, i.e. for the forward solution utilize updates from t_k to t_{k+1}, for the backward solution utilize updates from t_k to t_{k-1} (see **Figure 4-20**). According to Shin [32 p. 83] forward and backward solutions can be combined using following equations:

$$\boldsymbol{P}_{sm} = (\boldsymbol{P}_f^{-1} + \boldsymbol{P}_b^{-1})^{-1} \tag{4-150}$$

$$\hat{\boldsymbol{x}}_{sm} = \boldsymbol{P}_{sm}\left(\boldsymbol{P}_f^{-1}\hat{\boldsymbol{x}}_f + \boldsymbol{P}_b^{-1}\hat{\boldsymbol{x}}_b\right) = \hat{\boldsymbol{x}}_f + \boldsymbol{P}_{sm}\boldsymbol{P}_b^{-1}\left(\hat{\boldsymbol{x}}_f - \hat{\boldsymbol{x}}_b\right) \tag{4-151}$$

where *sm, f,* and *b* denotes the *smoothed, forward* and *backward* solutions.

The most widely used implementation of the fixed-interval smoother was derived by H. Rauch, K. Tung, and C. Striebel in 1965. It is known as the Rauch-Tung-Striebel (RTS) algorithm. On the contrary to the above equations, it does not require full-scale backward filter, although it is equivalent to combination of both forward and backward solutions [32 p. 84]. The RTS algorithm is a two-pass smoother: the first forward pass uses the KF but saves the intermediate results $\hat{\boldsymbol{x}}_k(-)$, $\hat{\boldsymbol{x}}_k(+)$,$\boldsymbol{P}_k(-)$, and $\boldsymbol{P}_k(+)$ at each measurement time t_k. The second pass runs backward in time in a sequence from the time t_N of the last measurement, computing smoothed state estimate from the intermediate results stored during the first pass [26 p. 162]. The smoothed estimate, subscript [s], is initialized as follows:

$$\hat{\boldsymbol{x}}[s]N = \hat{\boldsymbol{x}}_n(+) \tag{4-152}$$

The recursive RTS algorithm is then as follows [26 p. 162]:

$$\hat{\boldsymbol{x}}_{[s]k} = \hat{\boldsymbol{x}}_k(+) + \boldsymbol{A}_k(\hat{\boldsymbol{x}}_{[s]k+1} - \hat{\boldsymbol{x}}_{k+1}(-)) \tag{4-153}$$

$$\boldsymbol{A}_k = \boldsymbol{P}_k(+)\boldsymbol{\Phi}_k^T\boldsymbol{P}_{k+1}^{-1}(-) \tag{4-154}$$

$$\boldsymbol{P}_{[s]k} = \boldsymbol{P}_k(+) + \boldsymbol{A}_k(\boldsymbol{P}_{[s]k+1} - \boldsymbol{P}_{k+1}(-))\boldsymbol{A}_k^T \tag{4-155}$$

This RTS algorithm can be applied to both the EKF and the UKF to provide smoothed results [32]. Detailed theory with proof regarding smoother for both the EKF and the UKF is well documented in [32 pp. 126-140]. An example of RTS implementation to solve the random walk estimation and smoothing problem is presented in **Matlab 8-8** in Appendix 8.5.

4.6 INTRODUCTION TO PERFORMANCE ANALYSES

One of the most important implementation issues regarding performance of signal processing algorithms is computational load. In case of the KF the computational load is given by memory requirements and sampling frequency, which are both hardware specific [26 pp. 316-326]. Memory requirements for conventional KF are given in number of data words [26]:

$$\#data\_words = 4n^2 + 2l^2 + 3nl + 2n + l \tag{4-156}$$

where n is the total number of state variables and l is the number of measured state variables.

Length of a data word determines the precision of the estimates as well as filter numerical stability. To ease the computational load, order of the filter, i.e. the number of states, may be decreased by implementing suboptimal filtering techniques, which are for example well analyzed in [26 pp. 299-309]. In special cases, depending upon the matrix shape, data arrays can be replaced entirely by algorithms to compute matrix elements directly. This can spare up to half the operations when dealing for example with upper triangular matrices. Further memory savings can be achieved by eliminating data redundancy by reusing of temporary arrays or by introducing a suitable indexing method according to the data structures [26 p. 322]. These matrix structures given by indices *i* and *j* can be converted to one dimensional array given by index *k* according to the following indexing schemes as proposed in [26 p. 323]:

MATRIX STRUCTURE (n x n matrix)	MINIMUM MEMORY #(data words)	INDEXING k(i,j)
Symmetric	n(n+1)/2	i+j(j-1)(2n-1)(i-1)/2+i
Upper Triangular	n(n+1)/2	i+j(j-1)/2
Unit Upper Triangular	n(n+1)/2	i+(j-1)(j-2)/2
Diagonal	n	i
Toeplitz	n	i+j-1

Table 4-6 Alternative indexing schemes for memory saving purposes [26 p. 323]

The performance of the KF can then be evaluated by its *throughput;* defined as how many updates can be processed in a unit of time [26 p. 325]:

$$throughput\left(\frac{updates}{s}\right) = \frac{host\ processor\ speed\ (flops/s)}{computational\ complexity\ of\ application\ (flops/update)} \tag{4-157}$$

There exist several methods to the KF performance evaluation that usually lead to suboptimal modelling techniques introduced in the next chapter; the methods are as follows:

1. The *error-budgeting* is used to validate the contribution of different errors to the overall performance by testing and analyzing various error models [96].
2. *Observability analysis* is method for examining if a state estimation problem is solvable with respect to measurements used (see chapter 5.6.2), [24 pp. 85-87].
3. *Covariance analysis* is used to determine the overall sensitivity to modelling errors, to analyse possible filter divergence and to evaluate conditional observability over a given time interval (see chapter 5.6.1), [24 pp. 217-223].

4.7 INTRODUCTION TO SUBOPTIMAL MODELLING TECHNIQUES

Sometimes it can be heard that the KF is robust against modelling errors and disparities in the system model only to some extent. Therefore, to deal with these disparities the theory of suboptimal filtering has been postulated [26 p. 299]. Optimal filters usually have undesirable effects such as heavy computer load caused in most cases by the complexity of the models used. Solution is to reduce this implementation complexity. Unfortunately, main disadvantage of suboptimal filtering is that covariance matrix may not represent the actual estimation error and the estimates may be biased.

Modifying the Kalman gain is one of the techniques of suboptimalization [26 p. 300]: If the Kalman gain is time varying but quickly reaches constant nonzero value, this constant value then can be used for approximating the Kalman gain. For a non-constant or zero value of Kalman gain, piecewise constant function may be used for such approximation instead. However, the use of the modified gain usually results in poorer transient response and slower filter convergence.

Another technique is based on controlled filter model modification that can be realized in many different ways [26 pp. 303-316]:

- pre-filtering to attenuate some states, usually by adding a low-pass filter [26 p. 303],
- ignoring "redundant" states that exhibit only small effects during covariance analysis,
- decoupling states with weak bonds into separate subsystems,
- state spectrum simplification based on frequency domain approximations,
- state merging (combination of two unobservable states with identical propagation into one state),
- structural modelling (omitting the uncertain parameters by keeping only the structure, e.g. approximation of random processes such as sensor drift by random walk).

Based on an engineering insight, it is up to the designer to propose and evaluate these techniques. Suboptimal filters are in practice evaluated by *dual-state analysis* [26 p. 305]. Real system model is constructed as close to truth as possible, including all known phenomena that may influence the estimator performance [26]. Suboptimal filter model is then compared by means of estimation accuracy to the real system model. Such comparison requires simultaneous combination of both models into one dual-state vector; for more details see [26 pp. 305-309].

5 SMC-INS REALIZATION

Based on the state of the art described in chapter 2 and all the theory and methodology in chapter 4, this chapter presents the actual realisation of the SMC-INS designed in MATLAB. The emphasis is put on the realization using the adaptive filtering methods introduced so far, especially the EKF and the UKF.

To aid the basic research regarding the *From locomotion to cognition* project, a quadruped robot named ALAN (see chapter 2.3.2) was selected as an experimental platform for investigation of locomotion and dead reckoning using adaptive filtering methods. ALAN was equipped with appropriate inertial sensors, more specifically the MTi-OEM (Xsens) inertial measurement unit, to provide angular rates and accelerations. These inertial signals were processed by an **INS mechanization algorithm** (see chapter 4.4.4) that was developed and by data de-noising procedure (see chapter 4.3), to provide position, velocity, and attitude information ready for further data fusion with self-motion cues.

Generally, mathematical kinematic models of legged robots can provide velocity estimates. Using such a kinematic model is a straightforward solution for stable robots with stiff, position-controlled limbs. The advancement of the centre of mass can be computed through a rigid body transformation from the leg joint positions. With the correct model and provided no slipping occurs, this approach always works. However, in reality, this is rarely the case and dynamic locomotion poses even more difficulties, especially if an aerial phase is involved. The fact, that ALAN is underactuated with passive compliant knee joints, whose trajectory thus cannot be commanded, made such explicit kinematic modelling inappropriate. Therefore the proposed aiding using the self-motion cues is based on an untraditional and novel data-driven **SMC mechanization algorithm**. This algorithm analyses the passive knee-joint position measurements, the active hip-joint position measurements, the motor control signals, the feet pressure sensors, and the gait signal, in order to provide information about the change in velocity vector. The SMC mechanization algorithm can also be referred to as *legged odometer*. It is based on the idea that the speed of a legged robot moving with periodical gaits can in turn be obtained from its *frequency*stride length*. While the frequency of the movement of the legs is always easily accessible to the robot - it is a motor parameter - the stride length cannot be obtained directly since it is influenced by the environment (e.g. friction). In addition, the robot needs to be able to track its heading such that it can integrate its path. Hence, *the stride length* and heading, or more precisely *the change in heading over the gait period – the delta heading*, are the desired outputs of the developed legged odometer.

For the data fusion and elimination of sensors errors, the Kalman filter according to the EKF and the UKF (see chapter 4.5) approaches was designed. The design was based on an appropriate **SMC-INS error model**. The actual realisation of the SMC-INS navigation algorithm described in this chapter has the following structure:

1. the data integration scheme is presented in chapter 5.1,
2. the INS mechanization algorithm is documented in chapter 5.2,
3. the INS error model for the KF is described in chapter 5.3,
4. the SMC mechanization and the legged odometer design is presented in 5.4,
5. enhancements using nonholonomic constraints are introduced in chapter 5.5,
6. SMC-INS covariance and observability analyses are given in chapter 5.6.

5.1 SMC-INS INTEGRATION SCHEME

Due to the non-linear character of the SMC-INS system, the initial assumption for using primarily the UKF and the EKF was made. The LKF was omitted due to the expected poor performance caused by nonlinearities; the SEKF was omitted due to the analytical complexity of the Hessian derivations and due to the high computational load originating from increased number of necessary matrix operations. The scheme for SMC-INS integration using both the EKF and the UKF obeys the classical concept of complementary filtering and in principle it coincides with the generally used INS/GPS integration scheme introduced in chapter 4.4 (see **Figure 4-12**). The main reason to use the same concept is due to the character of the legged odometer errors. The legged odometer errors evaluated over one gait period are bounded and during the period they are not time-correlated. Still, the legged odometer cannot provide absolute information regarding position as provided by the GPS, because the error in heading can still accumulate with increasing number of gait periods. **Figure 5-1** [97 p. 244] shows the implemented integration scheme, i.e. how the errors enter the KF in the form of a measurement vector:

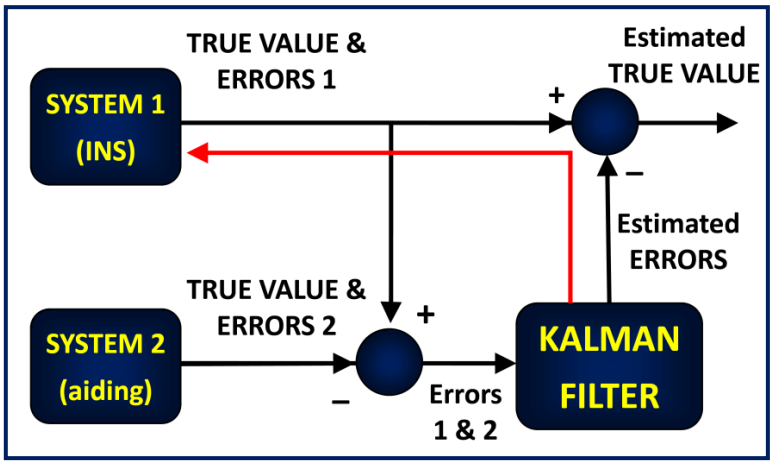


Figure 5-1 Principle of the SMC-INS integration using a KF (red arrow for feedback) [97 p. 244]

The KF estimates the errors using an error model (see chapter 5.3) and the estimated error values can be used to correct the inertial output either in a *feed-forward way* (INS mechanization algorithm is unaware of the error correction) or in a *feedback way* [81 p. 45]. Since the error values can grow large with time, the feedback approach is essential for the UKF and necessary for the EKF implementation [88]. Therefore, the feedback option was implemented such that the INS mechanization algorithm can use the corrected states for further computation. This way the error growth is controlled. According to **Figure 5-1** the SMC-INS system can be divided into three parts:

1. the **INS mechanization algorithm** to process the raw sensor data to obtain position, velocity and attitude values (see SYSTEM 1 in **Figure 5-1**),
2. the **SMC-INS error model** enhanced by nonholonomic constraints for error estimation and data fusion using the EKF/UKF algorithms (see KALMAN FILTER block in **Figure 5-1**),
3. the **SMC mechanization algorithm** based on the principle of legged odometry that provides the aiding signals (denoted SYSTEM 2 in **Figure 5-1**).

5.2 INS MECHANIZATION DESIGN

Using the mathematical apparatus introduced in chapters 4.4.1, 4.4.2 and 4.4.3 an algorithm for INS mechanization was implemented in MATLAB to operate in real time. It is based primarily on the theory proposed by Titterton [23 pp. 29-39] and Savage [75], [76]. Some important implementation issues originate from Salychev [25 pp. 61-75] and Shin [32 pp. 28-36], especially the successive extrapolation and interpolation approach to computation.

The conceptual scheme for INS mechanization implementation was introduced in chapter 4.4.4. In this chapter the actual step-by-step algorithm is presented without proof; for the theoretical proof and for more details see [23], [32], [25], [75], and [76]. The algorithm is structured according to the following simplified scheme in **Figure 5-2**:

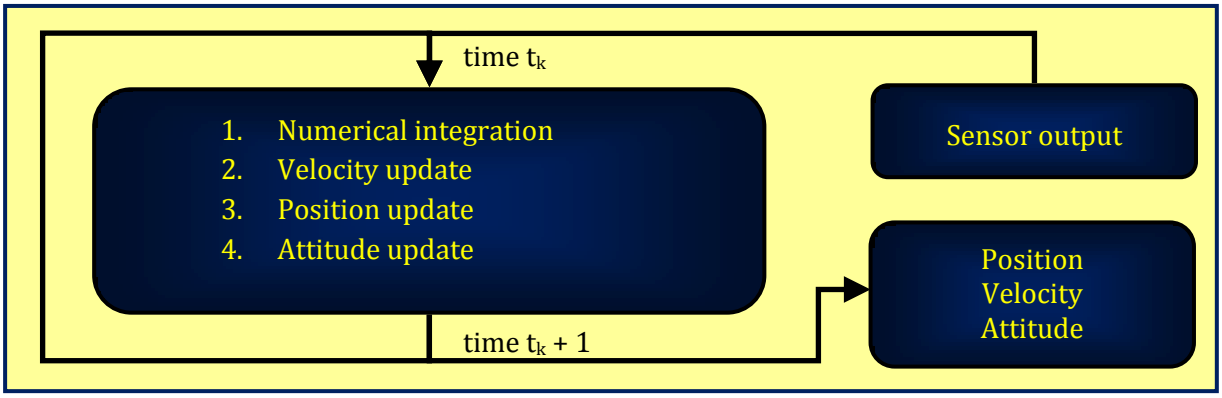


Figure 5-2 Simplified conceptual INS mechanization scheme as implemented

As presented in **Figure 5-2**, after the numerical integration proceeds, three steps of updates follow: *velocity*, *position* and *attitude* update. However, before the first measurements are processed, the initial latitude, longitude and altitude values have to be known to determine the gravity value according to the WGS84 model. The Euler angles (roll, pitch, and yaw) have to be known as well to determine the initial orientation. Usually, the initial position is obtained from a source of absolute position reference such as GPS; the initial Euler angles are usually computed using one of the possible initial alignment algorithms such as proposed in chapter 4.4.3. These initial conditions are then used to compute all the corresponding quaternions and direction cosine matrices that are required in the next computation steps.

5.2.1 NUMERICAL INTEGRATION

Numerical integration is the first step to take after the compensation of deterministic sensor errors is resolved (see chapter 4.4.5) – not implemented for the Xsens sensors since they already provide calibrated measurements. Numerical integration is a common approach to obtain the actual increments in angle and velocity from angular rate and acceleration measurements, respectively, using equations [23]:

$$\Delta\boldsymbol{\theta}_k = \int_{t_{k-1}}^{t_k} \widetilde{\boldsymbol{\omega}}_{ib}^b \mathrm{d}t \tag{5-1}$$

$$\Delta\boldsymbol{v}_k^b = \int_{t_{k-1}}^{t_k} \widetilde{\boldsymbol{a}}^b \mathrm{d}t \tag{5-2}$$

where $\widetilde{\boldsymbol{\omega}}_{ib}^b$ and $\widetilde{\boldsymbol{a}}^b$ are the vectors of measured sensor outputs provided by the angular rate sensors and accelerometers triads respectively, but compensated for the sensor errors (see chapter 4.4.5), t_k is the discrete time step given by the sampling period.

The most straightforward, but sufficient, numerical integration technique is the second order Runge-Kutta method that obeys the following equation [25]:

$$\Delta\boldsymbol{x}_k = \frac{1}{2}\big(\boldsymbol{x}(t_k) + \boldsymbol{x}(t_{k-1})\big)\Delta t \tag{5-3}$$

where $\boldsymbol{x}(t_k)$ is the actual measured value, $\Delta\boldsymbol{x}_k$ is the desired increment at actual time step and Δt is the sampling period.

To further investigate the influence of different numerical integration schemes on the strapdown mechanization precision, see the analysis published by Thong [98].

5.2.2 VELOCITY UPDATE

Before introducing the next individual steps of the INS mechanization, two important terms have to be clarified:

Coning effect in gyroscopes: "*Coning effect is the apparent drift rate caused by motion of an input axis in a manner that generally describes a cone. This usually results from a combination of oscillatory motions about the gyro principal axes. The apparent drift rate is a function of the amplitudes and frequencies of oscillations present and the phase angles between them, and is equal to the net solid angle swept out by the input axis per unit time,*" [63 p. 4].

Sculling effect in accelerometers: "*Sculling effect is the apparent acceleration resulting from the combined inputs of linear vibration along one axis, and angular oscillation at the same frequency around a perpendicular axis. The magnitude of the effect depends on the amplitudes and relative phase of these inputs, and appears on the axis perpendicular to both inputs,*" [63 p. 17].

All the three updates of the INS mechanization are structured into 17 main steps according to the latest implementation:

1. **Approximation of the rotational and sculling motion** effect on the velocity increment in BF is computed (2nd and 3rd term of the following equation respectively):

$$\Delta \boldsymbol{v}_{acc,k}^{b(k-1)} = \Delta \boldsymbol{v}_{acc,k}^{b} + \frac{1}{2}\Delta \boldsymbol{\theta}_k \times \Delta \boldsymbol{v}_{acc,k}^{b} + \frac{1}{12}\left(\Delta \boldsymbol{\theta}_{k-1} \times \Delta \boldsymbol{v}_{acc,k}^{b} + \Delta \boldsymbol{v}_{acc,k-1}^{b} \times \Delta \boldsymbol{\theta}_k\right) \tag{5-4}$$

The computation of the rotation vector of the NED frame at time t_k with respect to the attitude of the NED_{k-1} frame requires computation of the extrapolated values of $\boldsymbol{\omega}_{\text{ie,k}-1/2}^{n}$ and $\boldsymbol{\omega}_{\text{en,k}-1/2}^{n}$ obtained from midway positions and velocities for time interval (t_{k-1}, t_k). The position and velocities at time t_k are obtained using extrapolation since they are not available at the time t_k.

2. **Extrapolation of height (altitude)** for the time $t_{k-1/2}$, where $v_{D,k-1}$ is the velocity in NED in the downwards direction extracted from vector v_{k-1}^{n} from previous epoch:

$$h_{k-1/2} = h_{k-1} - v_{D,k-1}\Delta t_k/2 \tag{5-5}$$

3. **Extrapolation of the longitude and latitude** using quaternion approach:

3.1. Computation of the $\boldsymbol{\zeta}_{k-1/2}$ NED frame half interval $t_{k-1/2}$ rotation vector:

$$\boldsymbol{\zeta}_{k-1/2} = \left(\boldsymbol{\omega}_{ie,k-1}^{n} + \boldsymbol{\omega}_{en,k-1}^{n}\right)\Delta t_k/2 \tag{5-6}$$

3.2. Computation of the $\boldsymbol{\xi}_{k-1/2}$ ECEF frame half interval $t_{k-1/2}$ rotation vector:

$$\boldsymbol{\xi}_{k-1/2} = \boldsymbol{\omega}_{ie}^{e}\Delta t_k/2 \tag{5-7}$$

3.3. Half interval quaternions $\boldsymbol{q}_{n(k-1/2)}^{n(k-1)}$ and $\boldsymbol{q}_{e(k-1)}^{e(k-1/2)}$ are computed from the above $\boldsymbol{\zeta}_{k-1/2}$ and $\boldsymbol{\xi}_{k-1/2}$ rotation vectors respectively using apparatus derived in equation (4-49). The quaternion for the NED frame is computed using direct (positive) form conversion, the ECEF frame quaternion is computed using the inverse (negative) form.

3.4. Extrapolated NED to ECEF quaternion $\boldsymbol{q}_{n(k-1/2)}^{e(k-1/2)}$ is computed using quaternion multiplication chain rule:

$$\boldsymbol{q}_{n(k-1/2)}^{e(k-1)} = \boldsymbol{q}_{n(k-1)}^{e(k-1)} * \boldsymbol{q}_{n(k-1/2)}^{n(k-1)} \tag{5-8}$$

$$\boldsymbol{q}_{n(k-1/2)}^{e(k-1/2)} = \boldsymbol{q}_{e(k-1)}^{e(k-1/2)} * \boldsymbol{q}_{n(k-1/2)}^{e(k-1)} \tag{5-9}$$

3.5. Extrapolated longitude and latitude values are extracted from $\boldsymbol{q}_{n(k-1/2)}^{e(k-1/2)}$. The $\boldsymbol{q}_{n(k-1/2)}^{e(k-1/2)}$ quaternion can be first converted into corresponding DCM using equations (4-44), (4-58).

4. **Extrapolation of velocity** using the increments in NED velocity and gravity obtained from previous epoch (if k = 2 use initial values instead):

$$\boldsymbol{v}_{k-1/2}^{n} = \boldsymbol{v}_{k-1}^{n} + \frac{1}{2}\left(\Delta\boldsymbol{v}_{acc,k-1}^{n} + \Delta\boldsymbol{v}_{g/cor,k-1}^{n}\right) \tag{5-10}$$

5. **Update of the rates** $\boldsymbol{\omega}_{ie,k-1/2}^{n}$ and $\boldsymbol{\omega}_{en,k-1/2}^{n}$ using extrapolated longitude and latitude substituted into equations (4-60) and (4-61) respectively.
6. **Re-computation of the NED rotation vector** using the extrapolated rates:

$$\boldsymbol{\zeta}_k = \left(\boldsymbol{\omega}_{ie,k-1/2}^{n} + \boldsymbol{\omega}_{en,k-1/2}^{n}\right)\Delta t_k \tag{5-11}$$

7. **Velocity update** using velocity increment in NED due to specific force and velocity increment due to gravity and Corriolis force.

7.1. Transformation of the velocity increment from BF to NED:

$$\Delta\boldsymbol{v}_{acc,k}^{n} = [\boldsymbol{I} - (0.5\boldsymbol{\zeta}_k \times)]\boldsymbol{C}_{b(k-1)}^{n(k-1)}\Delta\boldsymbol{v}_{acc,k}^{b(k-1)} \tag{5-12}$$

where $(\boldsymbol{\alpha} \times)$ is the conversion of a vector into skew-symmetric matrix according to:

$$\left(\begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\end{bmatrix}\times\right) \equiv \begin{bmatrix}0 & -\alpha_3 & \alpha_2\\ \alpha_3 & 0 & -\alpha_1\\ -\alpha_2 & \alpha_1 & 0\end{bmatrix} \tag{5-13}$$

7.2. Computation of the gravity/Corriolis increment based on normal gravity value $\boldsymbol{g}_{k-1/2}^{n}$ obtained with respect to the extrapolated latitude and longitude position on the reference ellipsoid WGS84 [78 pp. 73-78], [81 p. 24]. This approach contains the correction for the centripetal acceleration due to the Earth rotation. The increment is obtained as follows:

$$\Delta\boldsymbol{v}_{g/cor,k}^{n} = \left[\boldsymbol{g}_{k-1/2}^{n} - \left(2\boldsymbol{\omega}_{ie,k-1/2}^{n} + \boldsymbol{\omega}_{en,k-1/2}^{n}\right) \times \boldsymbol{v}_{k-1/2}^{n}\right]\Delta t_k \tag{5-14}$$

7.3. Superposition of the velocity increments:

$$\boldsymbol{v}_{k}^{n} = \boldsymbol{v}_{k-1}^{n} + \Delta\boldsymbol{v}_{acc,k}^{n} + \Delta\boldsymbol{v}_{g/cor,k}^{n} \tag{5-15}$$

5.2.3 POSITION UPDATE

Position update is the part of the INS mechanization algorithm where the actual position is computed based on the velocity update obtained in the previous step.

8. **Re-computation by means of interpolation of the true half interval NED velocity** from the updated velocity and velocity from previous epoch:

$$\boldsymbol{v}_{k-1/2}^{n} = 0.5(\boldsymbol{v}_{k-1}^{n} + \boldsymbol{v}_{k}^{n}) \tag{5-16}$$

9. **Interpolation of height (altitude)** for the time t_k using recomputed NED velocity:

$$h_k = h_{k-1} - v_{D,k-1/2}\Delta t_k \tag{5-17}$$

10. **Interpolation of the longitude and latitude** using quaternion approach:
    10.1. Using the interpolated half interval NED velocity $\boldsymbol{v}_{k-1/2}^{n}$ the NED rotation vector can be recomputed as well by firstly updating the transport rate $\boldsymbol{\omega}_{en,k-1/2}^{n}$ using equation (4-61) and then substituting into:

$$\boldsymbol{\zeta}_k = (\boldsymbol{\omega}_{ie,k-1/2}^{n} + \boldsymbol{\omega}_{en,k-1/2}^{n})\Delta t_k \tag{5-18}$$

    10.2. Re-computation of the ξ_k ECEF rotation vector for the time t_k:

$$\boldsymbol{\xi}_k = \boldsymbol{\omega}_{ie}^{e}\Delta t_k \tag{5-19}$$

    10.3. Quaternions $\boldsymbol{q}_{n(k)}^{n(k-1)}$ and $\boldsymbol{q}_{e(k-1)}^{e(k)}$ are computed from the above $\boldsymbol{\zeta}_k$ and $\boldsymbol{\xi}_k$ rotation vectors respectively using apparatus derived in equation (4-49). The quaternion for the NED frame is computed using direct (positive) form conversion, the ECEF frame quaternion is computed using the inverse (negative) form equation.
    10.4. The NED to ECEF quaternion $\boldsymbol{q}_{n(k)}^{e(k)}$ for the rotation at time t_k is computed using quaternion multiplication chain rule:

$$\boldsymbol{q}_{n(k)}^{e(k-1)} = \boldsymbol{q}_{n(k-1)}^{e(k-1)} * \boldsymbol{q}_{n(k)}^{n(k-1)} \tag{5-20}$$

$$\boldsymbol{q}_{n(k)}^{e(k)} = \boldsymbol{q}_{e(k-1)}^{e(k)} * \boldsymbol{q}_{n(k)}^{e(k-1)} \tag{5-21}$$

    10.5. Updated longitude and latitude values are extracted from $\boldsymbol{q}_{n(k)}^{e(k)}$. The $\boldsymbol{q}_{n(k)}^{e(k)}$ quaternion may be first converted into a corresponding DCM using equation (4-44).
11. **Position update:** At this point the updated position in longitude and latitude is obtained. However, the increments in NED position are needed and can be computed using the difference in the ECEF coordinates when compared to the previous epoch at time t_{k-1}. This computation is biased by the approximation of the value of Earth radius at the given longitude and latitude. Two other approaches to obtain the NED position increments are proposed:
    11.1. The direct NED velocity integration approach – the same computation as used to update the height:

$$\boldsymbol{r}_{k}^{n} = \boldsymbol{r}_{k-1}^{n} + \boldsymbol{v}_{k}^{n}\Delta t_k \tag{5-22}$$

    11.2. The quaternion approach based on computing the quaternion corresponding to the position change from t_{k-1} to t_k:

$$\boldsymbol{q}_{\delta r^n} = \left(\boldsymbol{q}_{n(k-1)}^{e(k-1)^{-1}}\right) * \boldsymbol{q}_{n(k)}^{e(k)} \tag{5-23}$$

11.3. The rotation vector $\delta \boldsymbol{r}^n$is extracted from the quaternion $\boldsymbol{q}_{\delta r^n}$ using equations (4-51) and (4-52)

11.4. Increments in the NED position due to east δr_E and north δr_N can be extracted from the $\delta \boldsymbol{r}^n(1,1)$ and $\delta \boldsymbol{r}^n(2,1)$ values respectively, as given by equation derived from the equation (4-61):

$$\boldsymbol{\delta r}^n = \begin{bmatrix} \delta r_E/(R_N + h_k) \\ -\delta r_N/(R_M + h_k) \\ -\delta r_E \tan(\varphi_k)/(R_N + h_k) \end{bmatrix} \tag{5-24}$$

11.5. Since the height is already updated in equation (5-17), following integration scheme updates the NED position in eastern and northern coordinates.

$$\boldsymbol{r}_k^n = \boldsymbol{r}_{k-1}^n + \begin{bmatrix} \delta r_N \\ \delta r_E \\ 0 \end{bmatrix} \tag{5-25}$$

12. **Actualization of the transport rate $\boldsymbol{\omega}_{en}^n$ and earth rate in NED $\boldsymbol{\omega}_{ie}^n$** using updated longitude and latitude extracted from the quaternion $\boldsymbol{q}_{n(k)}^{e(k)}$ computed in equation (5-21). Updated longitude and latitude are substituted into equations (4-61) and (4-60).
13. **Actualization of the normal gravity value** due to the updated longitude and latitude position using approximation as proposed by Sotak [78 pp. 73-78].

5.2.4 ATTITUDE UPDATE

Attitude update is the part of the INS mechanization algorithm where the actual values of Euler angles are computed based on the rotations of the BF with respect to the NED frame.

14. **Correction of the interpolated values of longitude and latitude** using true updated positions:

$$\begin{bmatrix} \varphi_{k-1/2} \\ \lambda_{k-1/2} \\ h_{k-1/2} \end{bmatrix} = \frac{1}{2}\left(\begin{bmatrix} \varphi_{k-1} \\ \lambda_{k-1} \\ h_{k-1} \end{bmatrix} + \begin{bmatrix} \varphi_{k} \\ \lambda_{k} \\ h_{k} \end{bmatrix} \right) \tag{5-26}$$

15. **Correction of the NED rotation vector $\boldsymbol{\zeta}_k$** using the corrected half interval longitude, latitude and height to update the transport rate $\boldsymbol{\omega}^n_{en,k-1/2}$ and the $\boldsymbol{\omega}^n_{ie,k-1/2}$ NED earth rate by substituting into (4-61) and (4-60) respectively. These corrected rate values can be then used to re-compute the equation (5-18).
16. **Approximation of the BF rotation vector $\boldsymbol{\phi_k}$** with compensation of the coning effect in the angle increment in BF (2nd equation term):

$$\boldsymbol{\phi}_k = \Delta\boldsymbol{\theta}_k + \frac{1}{12}\Delta\boldsymbol{\theta}_{k-1} \times \Delta\boldsymbol{\theta}_k \tag{5-27}$$

17. **Attitude update** using quaternion multiplication chain rule:
    17.1. Quaternion $\boldsymbol{q}^{b(k-1)}_{b(k)}$ is computed by substituting the BF rotation vector value $\boldsymbol{\phi}_k$ into the direct form of equation (4-49).
    17.2. Quaternion $\boldsymbol{q}^{n(k)}_{n(k-1)}$ is computed by substituting the NED rotation vector value $\boldsymbol{\zeta}_k$ into the inverse form of equation (4-49).
    17.3. If $\boldsymbol{q}^{b(k-1)}_{b(k)} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ the following equivalent substitution $\boldsymbol{q}^{b(k-1)}_{b(k)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ is done instead to avoid division by zero.
    17.4. Quaternion chain rule is applied as follows:

$$\boldsymbol{q}^{n(k-1)}_{b(k)} = \boldsymbol{q}^{n(k-1)}_{b(k-1)} * \boldsymbol{q}^{b(k-1)}_{b(k)} \tag{5-28}$$

$$\boldsymbol{q}^{n(k)}_{b(k)} = \boldsymbol{q}^{n(k)}_{n(k-1)} * \boldsymbol{q}^{n(k-1)}_{b(k)} \tag{5-29}$$

    17.5. Quaternion $\boldsymbol{q}^{n(k)}_{b(k)}$ contains the actualized attitude information. The values of the Euler angles can be extracted (see the sample code **Matlab 8-3** in Appendix 8.5). Conversion of the quaternion $\boldsymbol{q}^{n(k)}_{b(k)}$ into the DCM $\boldsymbol{C}^{n(k)}_{b(k)}$ is necessary using equation (4-44) since it has to be substituted into (5-12) during the next cycle to provide compensation for the rotation of the accelerometers sensing axes.

5.3 SMC-INS ERROR MODEL DESIGN

To develop the SMC-INS error model theory regarding the sensor error propagation as described by Grewal in [27 pp. 172-178] was considered and the proposed error model was simplified up to a certain level of desired optimality. As described in chapter 4.6, such suboptimal error model assures that the KF performance is sufficiently fast and more stable. The suboptimal error model proposed in this chapter respects the classical 15-state concept [81 pp. 35-41], [60 p. 20]; however, it is modified to enable the data fusion for the SMC-INS and to include nonholonomic constraints. Details regarding derivation of both the first order and the second order SINS error models for EKF/SEKF can be found in [39 pp. 83-97]. Furthermore, simplified error models, such as *vertical error channel* and *lateral error channel,* can be found in [24 pp. 397-399].

To compare the performance of the classical EKF and the UKF, the same model was used in both cases. Another approach, as proposed by Shin [32 p. 116], could have been taken for the UKF. Shin proposes that the process model would include the whole strapdown mechanization algorithm (chapter 4.4.4) and the sigma points would propagate in the same manner as the raw sensor data do. This requires expansion of the state vector by the rotation quaternion errors and special averaging treatment during the computation [32 pp. 119-120]. Unfortunately, when this approach was combined with the SMC mechanization, computational load of this quaternion UKF proved to be enormous and the stability was questionable. Also the comparison to the EKF is inadequate since the quaternion UKF proceeds with absolute state values instead of error states – a completely different integration scheme would be required. Hence, this option was omitted for now.

5.3.1 PERTURBATION EQUATIONS

Differential equations that describe the error behaviour of the INS can be divided into equations for propagation of position, velocity and attitude errors [60 p. 24]. The translational errors – errors in velocity and position – and the attitude errors are not bound to each other and hence an uncoupled approach can be taken. The classical *perturbation analysis* [26] approach was performed, as proposed in [81], to linearize the system of differential equations (4-67), (4-68), (4-69), and (4-70). In this analysis the perturbation equations were substituted into the non-linear equations describing the system dynamics. The equations were then expanded and errors of the second and higher order were neglected [60]. The results of multiplication of two error values was assumed to be negligible. Resulting differential equations were expressed in terms of errors and formed the desired linearized error model. These equations in both the continuous-time form and the matrix form will be introduced in this chapter without proof since the derivation of the error model equations is not the subject of this dissertation and is well documented in [39 pp. 142-148], [60 pp. 24-30], and [81 pp. 35-41]. The perturbation equations were chosen as follows [60], [81]:

$$\hat{\boldsymbol{r}}^n = \boldsymbol{r}^n + \delta\boldsymbol{r}^n \tag{5-30}$$

$$\hat{\boldsymbol{v}}^n = \boldsymbol{v}^n + \delta\boldsymbol{v}^n \tag{5-31}$$

$$\hat{\boldsymbol{C}}_b^n = (\boldsymbol{I} - \boldsymbol{E}^n)\boldsymbol{C}_b^n \tag{5-32}$$

$$\hat{\boldsymbol{g}}^n = \boldsymbol{g}^n + \delta\boldsymbol{g}^n \tag{5-33}$$

where $\hat{x}$ in general is the computed value of x and δx is the error value of x.

The attitude errors are given as follows:

$$\boldsymbol{E}^n = (\boldsymbol{\epsilon}^n \times) \equiv \begin{bmatrix} 0 & -\epsilon_D & \epsilon_E \\ \epsilon_D & 0 & -\epsilon_N \\ -\epsilon_E & \epsilon_N & 0 \end{bmatrix} \tag{5-34}$$

Therefore, the error state vector for the proposed error model is as follows:

$$\boldsymbol{x} = [\delta \boldsymbol{r}^n \quad \delta \boldsymbol{v}^n \quad \boldsymbol{\epsilon}^n \quad \delta \boldsymbol{acc} \quad \delta \boldsymbol{gyr}]^T \tag{5-35}$$

where $\delta \boldsymbol{r}^n$ is the vector of errors in position, $\delta \boldsymbol{v}^n$ is the vector of errors in velocity, $\boldsymbol{\epsilon}^n$ is the vector of errors that define the error matrix for attitudes, $\delta \boldsymbol{acc}$ and $\delta \boldsymbol{gyr}$ are general error states corresponding to the sensor noise models defined in **Table 4-1**.

5.3.2 POSITION ERROR DYNAMICS

The position dynamics equations are functions of position and velocity; hence, the position error dynamics equation can be computed using the partial derivatives according to [81 p. 28]:

$$\delta \dot{\boldsymbol{r}}^n = \boldsymbol{F}_{rr} \delta \boldsymbol{r}^n + \boldsymbol{F}_{rv} \delta \boldsymbol{v}^n \tag{5-36}$$

where:

$$\boldsymbol{r}^n = \begin{bmatrix} \varphi \\ \lambda \\ h \end{bmatrix} \tag{5-37}$$

$$\boldsymbol{v}^n = \begin{bmatrix} v_N \\ v_E \\ v_D \end{bmatrix} \tag{5-38}$$

$$\boldsymbol{F}_{rr} = \begin{bmatrix} \frac{\partial \dot{\varphi}}{\partial \varphi} & \frac{\partial \dot{\varphi}}{\partial \lambda} & \frac{\partial \dot{\varphi}}{\partial h} \\ \frac{\partial \dot{\lambda}}{\partial \varphi} & \frac{\partial \dot{\lambda}}{\partial \lambda} & \frac{\partial \dot{\lambda}}{\partial h} \\ \frac{\partial \dot{h}}{\partial \varphi} & \frac{\partial \dot{h}}{\partial \lambda} & \frac{\partial \dot{h}}{\partial h} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \frac{-v_N}{(R_M + h)^2} \\ \frac{v_E \sin \varphi}{(R_N + h) \cos^2 \varphi} & 0 & -\frac{v_E}{(R_N + h)^2 \cos \varphi} \\ 0 & 0 & 0 \end{bmatrix} \tag{5-39}$$

$$\boldsymbol{F}_{rv} = \begin{bmatrix} \frac{\partial \dot{\varphi}}{\partial v_N} & \frac{\partial \dot{\varphi}}{\partial v_E} & \frac{\partial \dot{\varphi}}{\partial v_D} \\ \frac{\partial \dot{\lambda}}{\partial v_N} & \frac{\partial \dot{\lambda}}{\partial v_E} & \frac{\partial \dot{\lambda}}{\partial v_D} \\ \frac{\partial \dot{h}}{\partial v_N} & \frac{\partial \dot{h}}{\partial v_E} & \frac{\partial \dot{h}}{\partial v_D} \end{bmatrix} = \begin{bmatrix} \frac{1}{R_M + h} & 0 & 0 \\ 0 & \frac{1}{(R_N + h) \cos \varphi} & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{5-40}$$

and R_M, R_N are defined in equations (4-56), (4-57) respectively.

5.3.3 VELOCITY ERROR DYNAMICS

The velocity dynamics equation (4-67) can be perturbed using the perturbation equations (5-30), (5-31), (5-32), and (5-33) as derived in [81 pp. 29-32]. The gravity vector in NED can be approximated by the normal gravity vector, i.e. its component g^{nz} in the vertical axis, in the same way as in the velocity update of the INS mechanization algorithm (see chapter 5.2.2). Implementation of the normal gravity value computation using the WGS84 gravity model, for both the error model and the INS mechanization, was done according to [78 pp. 73-78], [81 p. 24] and is presented in **Matlab 8-9** in Appendix 8.5. The velocity error dynamics equation can be rewritten in the same way as the position error dynamics equation according to [60 p. 26] and [81 p. 32]:

$$\delta\dot{\boldsymbol{v}}^n = \boldsymbol{F}_{vr}\delta\boldsymbol{r}^n + \boldsymbol{F}_{vv}\delta\boldsymbol{v}^n + (\boldsymbol{f}^n\times)\boldsymbol{\epsilon}^n + \boldsymbol{C}_b^n\delta\boldsymbol{f}^b \tag{5-41}$$

$$\boldsymbol{F}_{vr} = \begin{bmatrix} -2v_E\omega_e\cos\varphi - \dfrac{v_E^2}{(R_N+h)\cos^2\varphi} & 0 & \dfrac{-v_Nv_D}{(R_M+h)^2} + \dfrac{v_E^2\tan\varphi}{(R_N+h)^2} \\ 2\omega_e(v_N\cos\varphi - v_D\sin\varphi) + \dfrac{v_Ev_N}{(R_N+h)\cos^2\varphi} & 0 & \dfrac{-v_Ev_D}{(R_N+h)^2} - \dfrac{v_Nv_E\tan\varphi}{(R_N+h)^2} \\ 2v_E\omega_e\sin\varphi & 0 & \dfrac{v_E^2}{(R_N+h)^2} + \dfrac{v_N^2}{(R_M+h)^2} - \dfrac{2g^{nz}}{\sqrt{R_MR_N}+h} \end{bmatrix} \tag{5-42}$$

$$\boldsymbol{F}_{vv} = \begin{bmatrix} \dfrac{v_D}{R_M+h} & -2\omega_e\sin\varphi - \dfrac{2v_E\tan\varphi}{R_N+h} & \dfrac{v_N}{R_M+h} \\ 2\omega_e\sin\varphi + \dfrac{v_E\tan\varphi}{R_N+h} & \dfrac{v_D + v_N\tan\varphi}{R_N+h} & 2\omega_e\cos\varphi + \dfrac{v_E}{R_N+h} \\ -\dfrac{2v_N}{R_M+h} & -2\omega_e\cos\varphi - \dfrac{2v_E}{R_N+h} & 0 \end{bmatrix} \tag{5-43}$$

5.3.4 ATTITUDE ERROR DYNAMICS

The attitude dynamics equation (4-70) can be perturbed in the same manner as the velocity dynamics equation to obtain the attitude error dynamics equation as in [81 pp. 33-35]:

$$\delta\dot{\boldsymbol{\epsilon}}^n = \boldsymbol{F}_{er}\delta\boldsymbol{r}^n + \boldsymbol{F}_{ev}\delta\boldsymbol{v}^n - (\boldsymbol{\omega}_{in}^n\times)\boldsymbol{\epsilon}^n + \boldsymbol{C}_b^n\delta\boldsymbol{\omega}_{ib}^b \tag{5-44}$$

$$\boldsymbol{F}_{er} = \begin{bmatrix} -\omega_e\sin\varphi & 0 & \dfrac{-v_E}{(R_N+h)^2} \\ 0 & 0 & \dfrac{v_N}{(R_M+h)^2} \\ -\omega_e\cos\varphi - \dfrac{v_E}{(R_N+h)\cos^2\varphi} & 0 & \dfrac{v_E\tan\varphi}{(R_N+h)^2} \end{bmatrix} \tag{5-45}$$

$$\boldsymbol{F}_{ev} = \begin{bmatrix} 0 & \dfrac{1}{R_N+h} & 0 \\ -\dfrac{1}{R_M+h} & 0 & 0 \\ 0 & -\dfrac{\tan\varphi}{R_N+h} & 0 \end{bmatrix} \tag{5-46}$$

5.3.5 ERROR MODEL IMPLEMENTATION

To implement the INS error model into the KF, all the differential error equations had to be converted into a matrix state-space description (see chapter 4.1). The actual implementation in this thesis is given by:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}_t\boldsymbol{x} + \boldsymbol{w}(t), \boldsymbol{w}(t) \sim N(0, \boldsymbol{Q}_t) \tag{5-47}$$

where $\boldsymbol{F}$ is the transition matrix augmented together with the accelerometers and angular rate sensors noise models such that they become part of the state vector $\boldsymbol{x}$. The augmentation was performed as described in chapter 4.1. The augmented $\boldsymbol{F}_t$ transition matrix is as follows:

$$\boldsymbol{F}_t = \begin{bmatrix} \boldsymbol{F}_1 \\ \boldsymbol{F}_2 \end{bmatrix} \tag{5-48}$$

$$\boldsymbol{F}_1 = \begin{bmatrix} \boldsymbol{F}_{rr} & \boldsymbol{F}_{rv} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} \\ \boldsymbol{F}_{vr} & \boldsymbol{F}_{vv} & (\boldsymbol{f}^n \times) & \boldsymbol{C}_b^n & \boldsymbol{0}_{3x3} \\ \boldsymbol{F}_{er} & \boldsymbol{F}_{ev} & -(\boldsymbol{\omega}_{in}^n \times) & \boldsymbol{0}_{3x3} & -\boldsymbol{C}_b^n \end{bmatrix} \tag{5-49}$$

$$\boldsymbol{F}_2 = \begin{bmatrix} \boldsymbol{0}_{3x9} & \boldsymbol{F}_{acc_{3x3}} & \boldsymbol{0}_{3x3} \\ \boldsymbol{0}_{3x9} & \boldsymbol{0}_{3x3} & \boldsymbol{F}_{gyr_{3x3}} \end{bmatrix} \tag{5-50}$$

where $\boldsymbol{F}_{acc_{3x3}}$ is the accelerometer noise model triad and $\boldsymbol{F}_{gyr_{3x3}}$ is the angular rate sensor noise model triad given by appropriated model equations concluded in **Table 4-1**; the actual dimensions varies according to the selected noise model. Parameters of the noise model equations are chosen according to either *PSD analysis* (see chapter 4.2.3) or *Allan variance analysis* (see chapter 4.2.4). Eventually, if the $\boldsymbol{F}_2$ matrix is chosen as all zeros $\boldsymbol{0}_{6x15}$ sensor noise is approximated by white noise for all sensors.

The covariance matrix of the process noise $\boldsymbol{Q}_t$, sometimes called the spectral density matrix, is defined by the variances in output of accelerometers and angular rate sensors, which form its diagonal elements as follows:

$$\boldsymbol{Q}_t = diag\begin{pmatrix} \sigma_{accx}^2 & \sigma_{accy}^2 & \sigma_{accz}^2 & \sigma_{gyrx}^2 & \sigma_{gyry}^2 & \sigma_{gyrz}^2 \end{pmatrix} \tag{5-51}$$

Because strapdown inertial systems are implemented with high-rate sampled data [81], the state space model has to be converted into discrete time, i.e. $\boldsymbol{F}_t$ is substituted by $\boldsymbol{\Phi}_k$ and $\boldsymbol{Q}_t$ is substituted by $\boldsymbol{Q}_k$ using for example the Van Loan discretization method (chapter 4.1). For more details regarding the relationship between continuous-time and discrete-time state space descriptions see [26 pp. 153-154]. The abbreviated notation of the discrete-time form is then:

$$\boldsymbol{x}_k = \boldsymbol{\Phi}_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1}, \boldsymbol{w}_k \sim N(0, \boldsymbol{Q}_k) \tag{5-52}$$

In order to account for any correlations between components of the driving noise that can develop over the sampling period due to integration, $\boldsymbol{Q}_k$ has to be approximated as [81 p. 38]:

$$\boldsymbol{Q}_k \approx \boldsymbol{\Phi}_k\boldsymbol{G}\boldsymbol{Q}_t\boldsymbol{G}^T\boldsymbol{\Phi}_k^T\Delta t \tag{5-53}$$

5.3.6 MEASUREMENT MODEL

The multi-sensor data fusion is performed by defining the measurement matrix $\boldsymbol{H}_k$ and the corresponding measurements vector $\boldsymbol{z}_k$. This measurement model specifies which state variables are used for the measurements to enter the KF. This is done using the differences in measurements from at least two different sensors; usually it is the difference in INS position and the position given by the aiding source. If the state variables are observable from the measurements, i.e. their covariance converges during the estimation process, the correct errors are estimated for the whole state vector. The measurement model is given by the following observation equation [26]:

$$\boldsymbol{z}(t) = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{v}(t), \boldsymbol{v}(t) \sim N(0, \boldsymbol{R}_t) \tag{5-54}$$

The position-velocity data fusion as implemented for the SMC-INS is described by the following measurement model:

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{r}^n_{INS} - \boldsymbol{r}^n_{AID} \\ \boldsymbol{v}^n_{INS} - \boldsymbol{v}^n_{AID} \end{bmatrix} \quad \boldsymbol{H} = \begin{bmatrix} \boldsymbol{I}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} \\ \boldsymbol{0}_{3x3} & \boldsymbol{I}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} \end{bmatrix} \tag{5-55}$$

where *AID* stands for measurements provided by the aiding source; in this case the SMC mechanization algorithm, but generally the GPS, magnetometer, odometer, radar etc.

Unfortunately, this model would cause numerical instabilities in calculating the Kalman gain vector because the state variables containing the errors in latitude and longitude are in radians and hence very small [60 p. 28], [81 p. 40]. Therefore, rather the following modification of the above equation is used:

$$\boldsymbol{z} = \begin{bmatrix} (R_M + h)(\varphi_{INS} - \varphi_{AID}) \\ (R_N + h)\cos\varphi(\lambda_{INS} - \lambda_{AID}) \\ h_{INS} - h_{AID} \\ \boldsymbol{v}^n_{INS} - \boldsymbol{v}^n_{AID-} \end{bmatrix} \tag{5-56}$$

$$\boldsymbol{H} = \begin{bmatrix} (R_M + h) & 0 & 0 & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} \\ 0 & (R_N + h)\cos\varphi & 0 & & & & \\ 0 & 0 & 1 & & & & \\ \multicolumn{3}{c}{\boldsymbol{0}_{3x3}} & \boldsymbol{I}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} \end{bmatrix} \tag{5-57}$$

If only the position measurements are available, the measurement model becomes simplified:

$$\boldsymbol{z} = \begin{bmatrix} (R_M + h)(\varphi_{INS} - \varphi_{AID}) \\ (R_N + h)\cos\varphi(\lambda_{INS} - \lambda_{AID}) \\ h_{INS} - h_{AID} \end{bmatrix} \tag{5-58}$$

$$\boldsymbol{H} = \begin{bmatrix} (R_M + h) & 0 & 0 & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} \\ 0 & (R_N + h)\cos\varphi & 0 & & & & \\ 0 & 0 & 1 & & & & \\ \multicolumn{3}{c}{\boldsymbol{0}_{3x3}} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} & \boldsymbol{0}_{3x3} \end{bmatrix} \tag{5-59}$$

Once the errors are estimated, it is necessary to provide corrections to the mechanization algorithm. These corrections depend on the actual integration scheme as proposed in chapter 4.4.6.

Corrections implemented in this dissertation were provided using the feedback method (see **Figure 5-1**) given by the following equations derived from the perturbation equations (see chapter 5.3.1):

$$\boldsymbol{r}^n = \hat{\boldsymbol{r}}^n - \delta\boldsymbol{r}^n \tag{5-60}$$

$$\boldsymbol{v}^n = \hat{\boldsymbol{v}}^n - \delta\boldsymbol{v}^n \tag{5-61}$$

$$\boldsymbol{C}_b^n = (\boldsymbol{I} + \boldsymbol{E}^n)\widehat{\boldsymbol{C}}_b^n \tag{5-62}$$

To bind the KF and the INS mechanization algorithms together correctly, following implementation rules were proposed and had to be obeyed:

- If mechanization is implemented using the quaternion approach, the update of the $\boldsymbol{q}_b^n$ quaternion has to be performed using the $\boldsymbol{C}_b^n$ DCM matrix when the feedback proceeds.
- If at least one KF update has occurred and if sufficient time has passed to build up the errors, feedback to mechanization and correction of errors can proceed.
- Each time the feedback occurs and the errors are corrected, the state vector has to be reset to zero, otherwise the errors in the state vector would grow unbounded [24].
- Usually, the sampling frequency of the inertial sensors is much higher than the sampling frequency of the aiding data source. Until the aiding measurements are available, the KF has to run in the prediction regime accumulating and estimating the errors. With the measurements available the KF update occurs asynchronously. (NOTE: This rule was implemented by polling the measurements for zero values. If it was zero, the Kalman gain was reset to zero.

The actual implementation of the KF feedback to the mechanization algorithm according to the above rules is presented in **Matlab 8-10** in Appendix 8.5 with the KF algorithm developed according to chapter 4.5).

5.3.7 KALMAN FILTER TUNING

After both the system and the measurement models have been developed, it was necessary to proceed with the KF tuning to obtain optimal performance and to prevent the KF from diverging. The KF tuning consisted of adjusting the parameters of the process covariance matrix $\boldsymbol{Q}_k$ and the covariance matrix of observational uncertainty $\boldsymbol{R}_k$. In general, the process covariance matrix $\boldsymbol{Q}_k$ can be set according to the value of its norm [25], [81 p. 38]:

- If the norm of $\boldsymbol{Q}_k$ is larger than the real one, the KF trusts measurements more than the system model and resulting estimates will be significantly noisy due to free passage of the measurement noise.
- If the norm of $\boldsymbol{Q}_k$ is smaller than the real one, time delay is present in the estimated data.
- If the norm of $\boldsymbol{Q}_k$ is significantly smaller than the real one, the KF usually diverges and becomes numerically instable.

In case of low-cost inertial sensors the matrix $\boldsymbol{Q}_k$ should be chosen rather small such that the KF can account for all the sensor errors and imperfections. The KF was tuned such that $\boldsymbol{Q}_k$ was increased continuously until the filter reached stable state and the estimated navigation trajectory followed the one provided by the source of aiding [81]. To improve the tuning, an iterative approach was taken such that the matrix $\boldsymbol{Q}_k$ was adapted according to the RMSE between the estimated trajectory and the reference trajectory acquired from the video.

The covariance matrix of the observation uncertainty $\boldsymbol{R}_k$ is a diagonal matrix as follows:

$$\boldsymbol{R}_k = diag\left(\sigma_\varphi^2 \quad \sigma_\lambda^2 \quad \sigma_h^2 \quad \sigma_{vn}^2 \quad \sigma_{ve}^2 \quad \sigma_{vd}^2\right). \tag{5-63}$$

The elements of the matrix $\boldsymbol{R}_k$ are usually initialized as follows [25], [81 p. 41]:

- The **position uncertainty** is given by the standard deviations according to the external aiding source - in case of GPS it is the standard deviation in GPS position solution, in case of SMC, it was given by the statistics determined for each gait.
- For stationary initialization, which occurs in most cases, the **velocity uncertainty** is initialized to be close to zero.
- If the attitude measurement is implemented for data fusion: the **attitude uncertainty** is in all cases dependant on the inertial sensors biases identified by calibration (see chapter 4.4.5) or during initial alignment (see chapter 4.4.3). If the biases are known, attitude uncertainty can be small. If the biases are not known, analytical approach should be taken as described in [81 pp. 70-71]. However, attitude aiding was not implemented in the SMC-INS navigation algorithm.

5.4 SMC MECHANIZATION DESIGN

As described in chapter 5.1, the integrated SMC-INS system consists of the *INS mechanization algorithm* to provide inertial navigation variables (see chapter 4.4.4), the *SMC mechanization algorithm* to provide the aiding in velocity and position, and the *SMC-INS error model* for the KF to estimate the errors and perform data fusion (see chapter 5.3.5). The velocity and position aiding is based on the developed legged odometer that computes stride length and change in heading from the information obtained by the SMC sensors, i.e. from the SMC signals. All SMC signals have periodic character since the motor signals, which control the motion, are also periodic. Therefore, with the current knowledge of the gait type and its frequency the position and velocity increments can be computed for each period independently. This computation is based on a set of indicators constructed from measured SMC data by exploiting a novel data-driven approach that is described in detail in chapter 5.4.2.

5.4.1 GAIT CHARACTERISTICS

For both ALAN as well as its simulated counterpart SIMALAN created in Webots, the gait signal is determined by the controller to specify the gait type or in other words to define the robot's behaviour regarding motion dynamics. Statistical analysis was performed for each gait type in its optimal form to obtain characteristic speed of motion and radius of turning for turning gaits. Concerning SIMALAN these gait characteristics are together with the list of selected gait types concluded in **Table 5-2**. For each desired type of motion optimally tuned gait was defined by a set of the following parameters: *frequency*, *amplitude*, *offset*, and *phase lag* of the motor signal. Using self-evaluation algorithms[3] these parameters were defined in such a way to ensure natural motion capability. First, optimal gaits were defined for the simulated SIMALAN, and then they were transferred and accommodated to the real robot. Whole spectrum of motion dynamics was available for ALAN just by specifying the frequency and the gait type.

GAIT TYPE	Description	Frequency (Hz)	Speed (m/s)	Radius of turning (m)
0	Sprint	4.00	0.53	0
1	Sit-still	0	0	0
2	Turn pacing sharp left	2.3382	0.37	1.00
3	Turn pacing sharp right	2.3382	0.37	1.00
4	Turn on spot left	2.15251	0.32	0.4
5	Turn on spot right	2.15251	0.32	0.4

Table 5-1 SIMALAN optimal gait characteristics

[3] Developed by M. Hoffmann, University of Zurich, AI Laboratory

5.4.2 LEGGED ODOMETER DEVELOPMENT

To develop a legged odometer for a quadruped robot, the relationship among the variables of interest - stride length and delta heading - and the available sensory and motor signals was investigated. For every leg following signals were available: motor frequency signal, amplitude of hip motor signal, hip and knee angular positions, and feet pressure signal.

Every legged robot such as ALAN is a complex nonlinear dynamic system whose interaction with the environment is very complicated to model. During motion the contacts with the ground introduce notorious discontinuities into any model. Moreover, modelling of the self-motion cues sensors would even increase the complexity. Thus, it was not realistic to develop an analytical model and a *data-driven approach* was adopted instead.

Large number of sensory data was collected together with referential trajectory data both in Webots as well as in real. The data were then *analysed for correlations*, *fitted with linear functions* and *the fitting error was evaluated statistically*. The succesfully identified correlations were marked as *indicators* for motion. These indicators were in fact the parameters of a linear function (its slope and its offset) that mapped the SMC measurements onto stride length and delta heading values. The quality of each indicator was given by the statistical fitting error. A methodology[4] was proposed taking the advantage of the Webots to simulate a large number of surfaces of variable friction. The methodology was as follows:

1. **Measurement phase**: For different gaits tested on different surfaces defined by its friction large sets of data were collected for selected frequencies of the motor signals. Each measurement set contained: the actual *gait* signal, the motor *frequency* signals, and for every leg the active *hip joint amplitudes*, the passive *knee joint amplitudes* and *feet pressure signals*.
2. **Indicator proposal phase:** Since the gait motion is periodical with the period given by the motor frequency signal, all of the indicators were computed with respect to the particular period. The two desired outputs were the *stride length* and *delta heading.* The stride length defines the increment in distance travelled over the gait period. The delta heading defines the angular increment in the heading over the gait period (taken clockwise with respect to the north). A set of indicators was chosen by intuition using the measured data or any of their combination. When choosing these indicators, symmetry along the longitudinal axis was assumed for all turning gaits, i.e. an indicator positively correlated for turning right had to be negatively correlated for turning left.
3. **Correlation investigation phase:** The stride length, delta heading and all of the indicators selected in the previous phase were computed from all of the data sets measured for all possible combinations of gaits and surfaces (the training data set). A correlation matrix was generated such that the correlation for a given indicator with all of the other indicators could be investigated. A Hinton graph was used to represent the correlations by a matrix of colored squares, where the size of each square was proportional to the magnitude of the correlation coefficient and the color determined the sign (red for negative correlation, green for positive); for examples see **Figure 5-3** and **Figure 5-4** (following indicator legend is used: *freq* – motor frequency, *friction* – surface friction value, *stride* – stride length, *dHeading* – delta heading, *SUM hip*

[4] Developed in cooperation with M. Hoffmann, University of Zurich, AI Laboratory, as part of the project *From locomotion to cognition*, grant Nr. 200020-122279/1, supported by Swiss National Science Foundation

amp – sum of all hip signal amplitudes, *SUM knee amp* – sum of all knee signal amplitudes, *SUM touch* – numerical integral of all touch pressure signals, *L2R hip ratio* – ratio of the sum of the left-side hip amplitudes to the right-side hip amplitudes, *L2R knee ratio* – ratio of the sum of the left-side knee amplitudes to the right-side knee amplitudes, *L2R touch* – ratio of the numerical integral of the left-side touch pressure signals to the right-side ones).

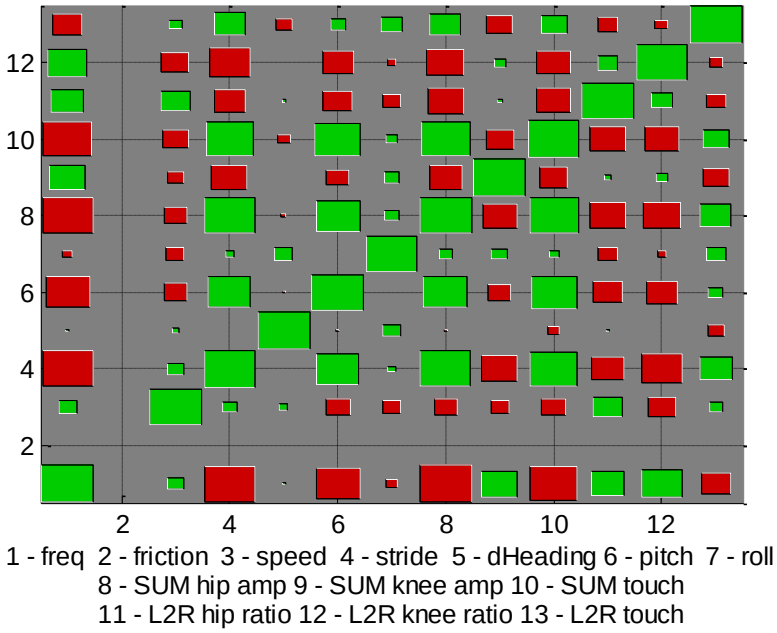


Figure 5-3 Example of a Hinton graph to represent correlation of indicators based on the data obtained for the sprint gait (gait type 0) and standard friction (friction value of 1)

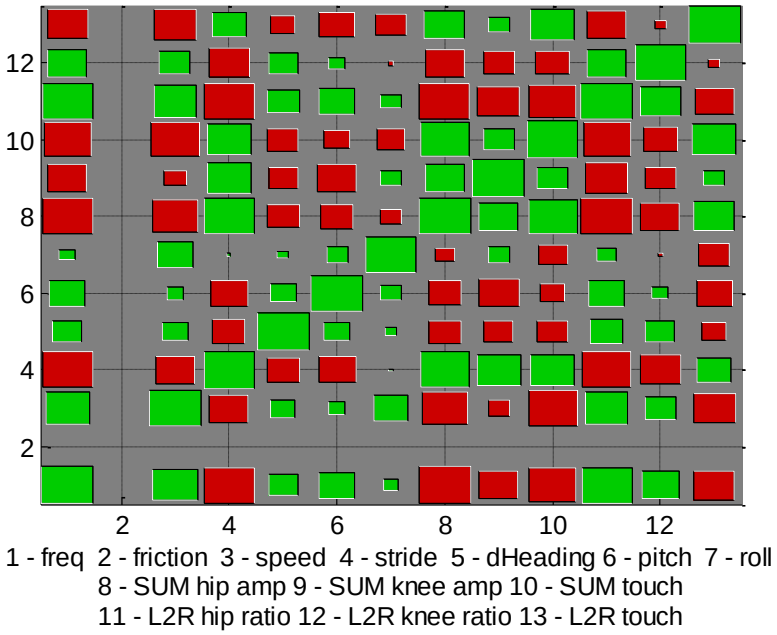


Figure 5-4 Example of a Hinton graph to represent correlation of indicators based on the data obtained for turning right using sharp pacing gait (gait type 3) and standard friction (friction value of 1)

These Hinton graphs were used to identify all the possible linear correlations among the constructed indicators. The indicators were then grouped according to the relevance to

individual gaits with respect to the stride length and delta heading. Strong and weak correlations were distinguished. The *Statistics Toolbox* for *MATLAB* was used to obtain the required correlation matrices and Hinton graphs.

4. **Iteration phase:** phases 1 to 3 were repeated and the most suitable indicators, those with strongest position on the Hinton graphs, were selected from all iterations. A data set for each indicator was formed to capture all possible relations among indicators; for example the data in **Figure 5-5** correspond to the Hinton graph in **Figure 5-3** (delta heading was not evaluated for the sprint gait since it is assumed to be zero), and the data in **Figure 5-6** and **Figure 5-7** correspond to the Hinton graph in **Figure 5-4**. Such a data set can be characterized by plot with the desired odometry output on the y-axis (the stride length or the delta heading) and the selected indicator on the x-axis. Strong positive linear correlation then corresponds to positive-slope linear plot and strong negative linear correlation to negative-slope linear plot. Weak correlation is characterized by a randomly shaped cloud of data.

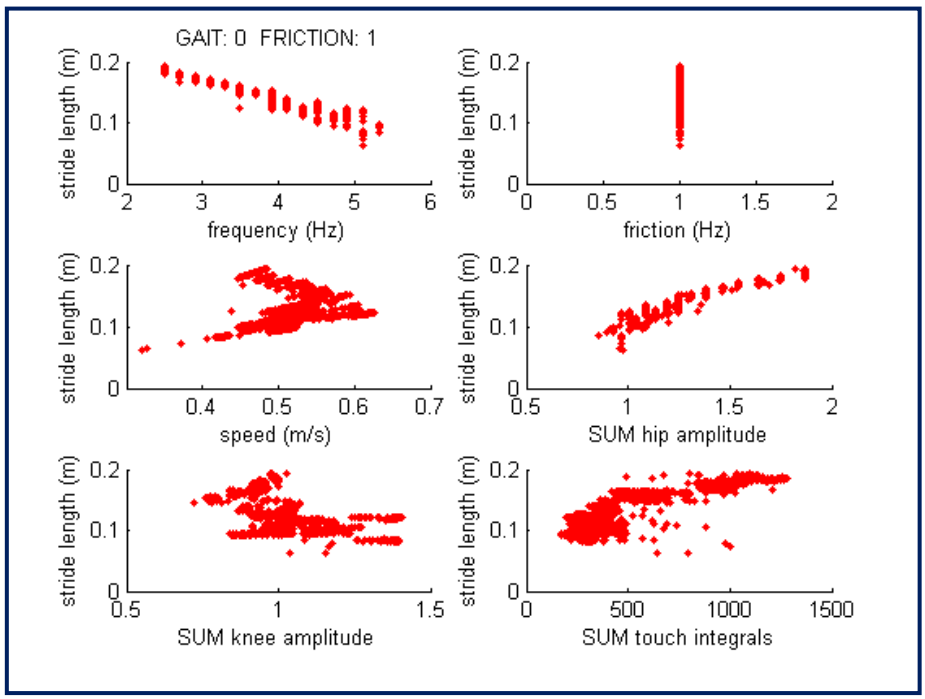


Figure 5-5 Example of indicator data for stride length based on the measurements obtained for the sprint gait (gait type 0) and standard friction (friction value of 1)

For example, in **Figure 5-5** the plot of stride length vs. sum of the amplitudes of the hip signals (*SUM hip amp*) shows strong positive correlation which corresponds to a large green square at position 4, 8 on the Hinton matrix in **Figure 5-3**, i.e. the *SUM hip amp* can be considered as a good indicator. Another example: in **Figure 5-7** the plot of delta heading vs. ratio of the integral of the left-side touch pressure signals to the right-side ones (*L2R touch*) shows weak negative correlation which corresponds to a small red square at position 5, 13 on the Hinton matrix in **Figure 5-4**, i.e. the *L2R touch* can be considered as a bad indicator.

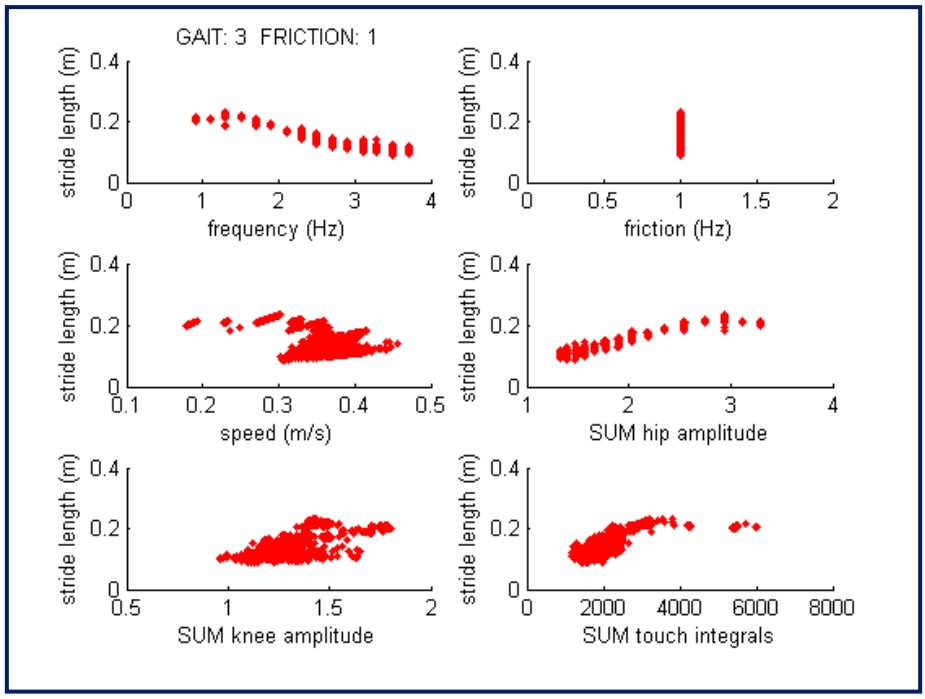


Figure 5-6 Example of indicator data for stride length based on the measurements obtained for turning right using sharp pacing gait (gait type 3) and standard friction (friction value of 1)

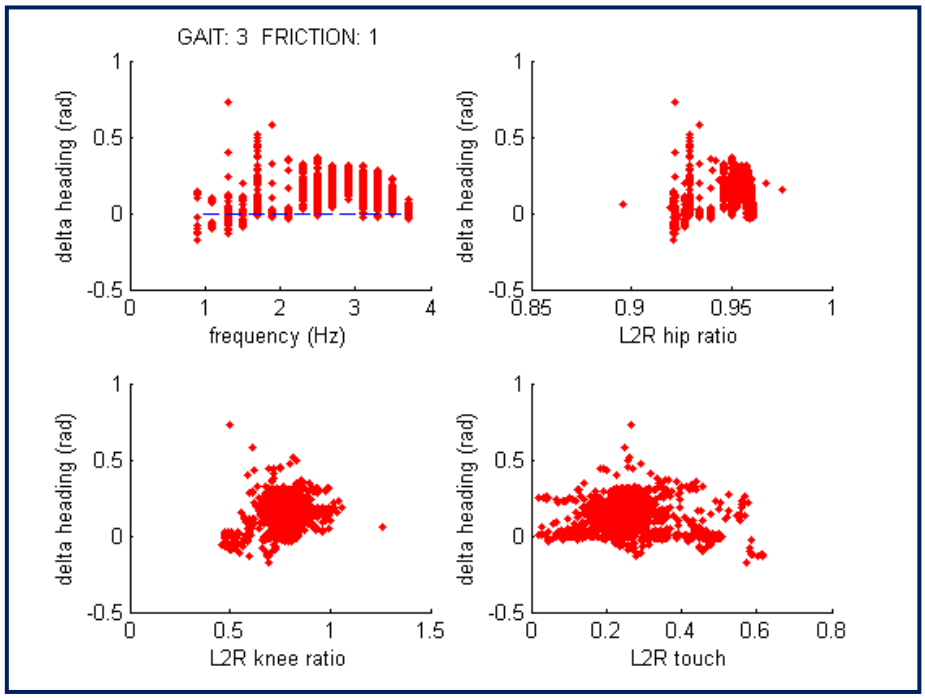


Figure 5-7 Example of indicator data for delta heading based on the measurements obtained for turning right using sharp pacing gait (gait type 3) and standard friction (friction value of 1)

5. **Least squares fitting phase:** In the previous phase, data sets relating the selected good indicators to the odometry output were formed. To define this relation empirically the set of SMC measurements obtained at 50 Hz for ALAN (at 500 Hz for SIMALAN) was fitted in least

square sense to obtain the best line fit. Each fit was described by two parameters: the *slope* and the *offset* of the fitted line. The quality of each indicator was therefore given by the RMS fitting error. For the actual data fitting and for determination of the fitting error, the *Spline Toolbox* for *MATLAB* was used; as an example of indicator data fit see **Figure 5-8** .

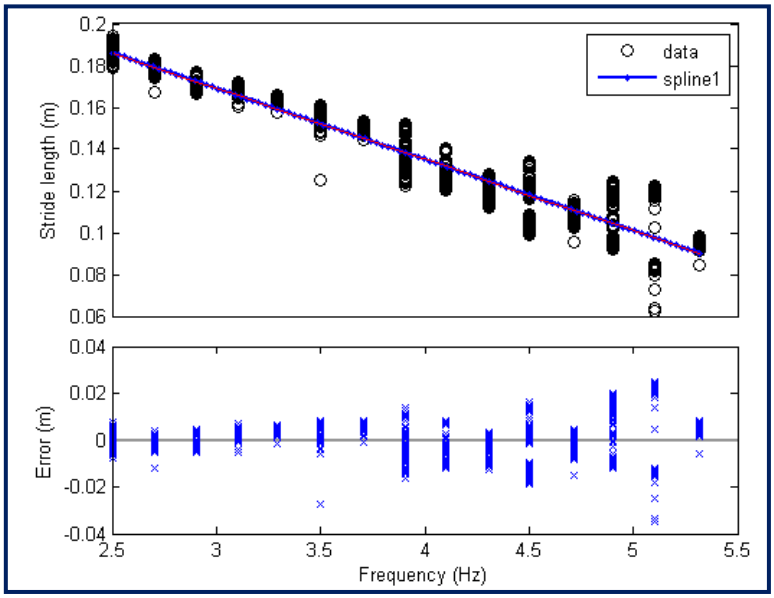


Figure 5-8 Example of the *frequency* indicator data fitting in least squares sense for sprint gait (gait type 0)

6. **Self-motion cues mechanization:** the SMC mechanization was designed as a case-based algorithm that at each time step, given by the output data rate of the sensors, distinguishes change in gait and/or frequency. Gait change determines the choice of indicators; frequency change indicates the change in dynamics. All of the indicators are period-based, i.e. computed from SMC measurements accumulated over the last gait period, and can either be stored in a buffer or used for aiding signal computation. SMC measurements enter the linear equations of the indicators to produce number of stride length and/or delta heading values. To get the optimal stride length or delta heading estimate these values are recombined using the weight computed from the inverse value of the best line fit RMS errors, i.e. the indicator with the lowest fitting RMS error becomes the most trusted one.
7. **Aiding signal update:** The aiding signal that enters the SMC-INS error model through the equation (5-55) is updated at the end of every gait period in both the position and velocity. The estimated position and velocity at the beginning of the last period is superposed appropriately with the newly computed increments given by the stride length, delta heading, and the gait period length. Simple trigonometry is used to decompose the incremental vector given by its magnitude (stride length) and bearing (delta heading) into the northern and the eastern position components in the NED frame. The velocity components are derived from the position increments divided by the period length. The vertical velocity component in the body frame is assumed zero mean over any gait period and is resolved via the nonholonomic constraints proposed in chapter 5.5. Optimal indicators identified for legged odometer development are concluded in the **Table 5-2**.

INDICATOR	**STRIDE LENGTH Gait 0** (Sprint)	**STRIDE LENGTH Gait 2 & 3** (Turn pacing sharp right)	**STRIDE LENGTH Gait 4 & 5** (Turn on spot right)	**DELTA HEADING Gait 0** (Sprint)	**DELTA HEADING Gait 2 & 3** (Turn pacing sharp right)	**DELTA HEADING Gait 4 & 5** (Turn on spot right)
Frequency of the motor signals (**freq**)	−	−	−			−
Sum of all hip amplitudes (**sum hip amp**)	+	+	+			
Integral of all the touch sensor signals (**sum touch**)	+	+				
Sum of all knee amplitudes (**sum knee amp**)		+	+			
Ratio of the sum of left hip amplitudes to the sum of the right hip amplitudes (**l2r hip ratio**)			+		+	+
Ratio of the sum of left knee amplitudes to the sum of the right knee amplitudes (**l2r knee ratio**)					+	+
Ratio of the integral of the left touch pressure sensors to the integral of the right touch pressure sensors (**l2r touch**)					−	

Table 5-2 Overview of the indicators for SIMALAN legged odometer; green = positive linear correlation, red = negative linear correlation, grey = no significant correlation

The methodology described in this chapter can be implemented in the same manner for almost any legged robotic platform that utilizes any self-motion cues sensors. It was followed for both the SIMALAN and its real counterpart ALAN. In case of SIMALAN the concept of indicators was tested and verified using large data sets and various environment settings. In case of ALAN the training data sets were limited by the real world aspects; such that ALAN's range of motion was limited by cables, and the "true" trajectory reference was limited by the precision of tracking using external camera.

5.5 SMC-INS ENHANCEMENT BY NONHOLONOMIC CONSTRAINTS

When a vehicle or navigated object is bounded to the surface, it is said to be governed by nonholonomic constraints. These constraints can be exploited at high frequency to aid the estimation of alignment of an IMU [20]. They can be combined with the observation measurements and directly incorporated into the measurement model.

In ideal case nonholonomic constraints for an object moving bounded to the surface, provided no jumping or lateral sliding takes place, is zero lateral and vertical body frame velocity [20 p. 733]. However, these strong constraints are usually violated due to presence of side slip during cornering, vibration, or aerial phases of motion of legged robots. Therefore, approximated nonholonomic constraints were implemented to the measurement model; the extent of constraints violation with respect to the nature of ALAN's motion was modelled as follows:

$$v_y^b - \nu_y = 0 \tag{5-64}$$

$$v_z^b - \nu_z = 0 \tag{5-65}$$

where v_y^b is the lateral BF velocity, v_z^b is the vertical BF velocity, ν_y and ν_z are Gaussian white noise sources with zero mean and variance σ_{vby}^2 and σ_{vbz}^2 respectively - these variances are transformed to the NED frame and implemented into matrix $\boldsymbol{R}_k$ as defined in equation (5-63).

The nonholonomic constraints computations are performed in the *Measurement vector* block that is part of the implementation scheme in **Figure 5-9**. The update rate is typically 10% of the gait frequency, usually at 10 – 25 Hz, i.e. at least ten times higher than update from the SMC mechanization. After the nonholonomic constraints were incorporated into the model, the performance improved dramatically since it allowed correct capturing of the motion dynamics but still it limited the error growth as desired.

5.6 SMC-INS OVERVIEW

The following scheme describes the principle of operation of the SMC-INS system:

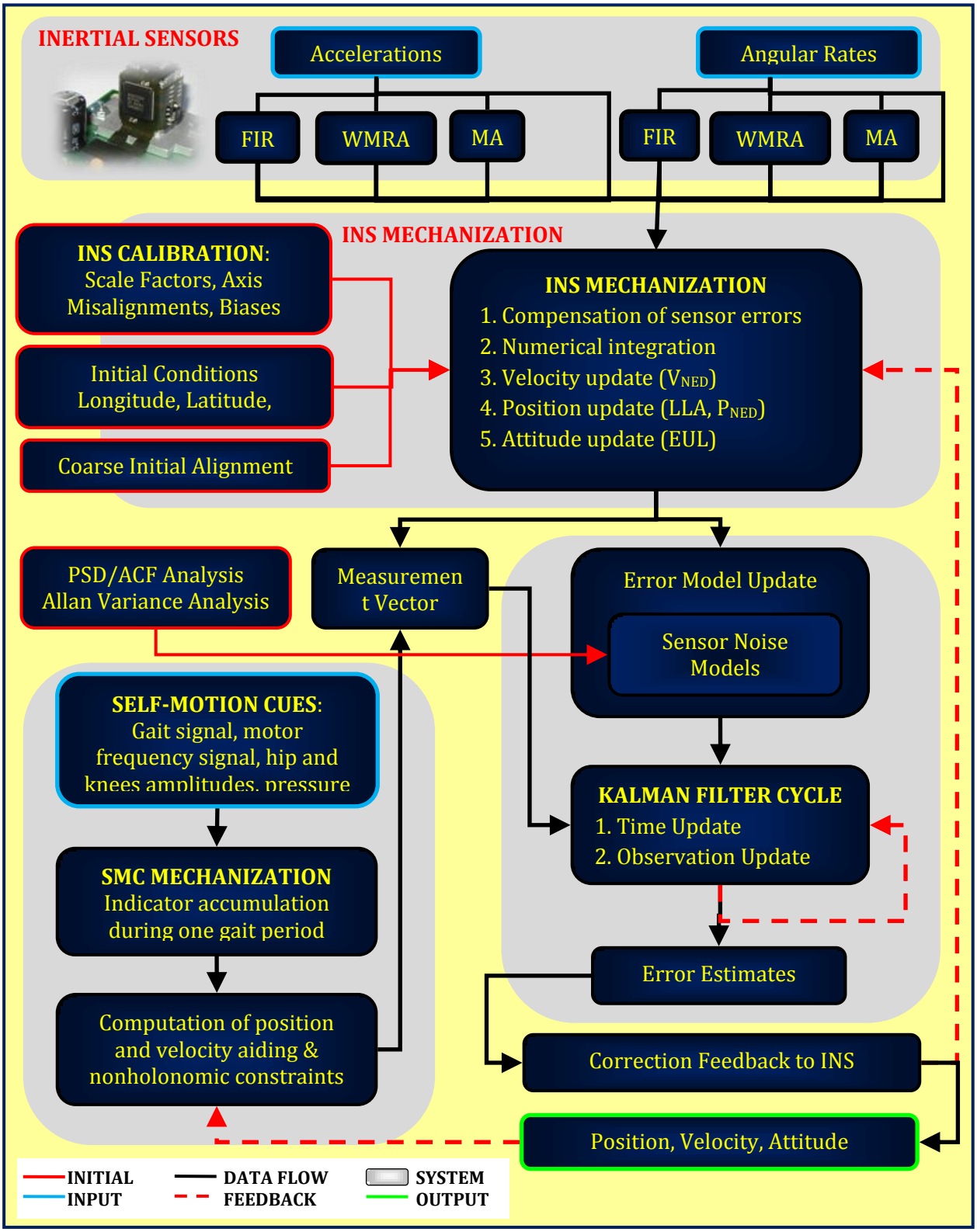


Figure 5-9 SMC-INS implementation scheme as realized in MATLAB

The overall operation of the developed SMC-INS system can be described in 5 major steps:

1. **Pre-processing: R**aw inertial data, three accelerations and three angular rates, enter the SMC-INS system at a rate of 100 Hz for real ALAN and 500 Hz for SIMALAN in Webots. These data are pre-filtered at each data step by a filter of choice: digital low-pass FIR filter of chosen cut-off frequency, moving average filter of chosen span, or the WMRA de-noising procedure suggested in chapter 4.3 (see results in chapter 6.2 for optimal WMRA parameters). The SMC-INS starts with an initialization procedure where:
    a. Scale factors, axis misalignments and biases determined by offline calibration for each of the six inertial sensors are loaded and used as parameters for the observation equation for error compensation; see equation (4-72).
    b. Initial latitude, longitude, and altitude are entered for correct computation of the local gravity value.
    c. Coarse alignment algorithm estimates the initial attitude values (see chapter 4.4.3). At this point external heading information can be supplied to improve the precision of the alignment.
2. **INS mechanization:** The pre-filtered data and the initial conditions enter the INS mechanization algorithm, which runs in an infinite cycle at the body rate frequency. Five mechanization phases are repeated:
    a. Compensation of sensor errors identified during the offline calibration is performed (see chapter 4.4.5). This step can be omitted if the inertial sensors already provide raw calibrated data as in case of the MT9 and the MTi-OEM units from Xsens.
    b. Numerical integration is performed such that velocity and angle increments are computed (see chapter 5.2.1).
    c. Velocity update is performed; compensation for rotational motion and sculling effect of accelerometers is computed (see chapter 5.2.2).
    d. Position update is performed (see chapter 5.2.3).
    e. Attitude update is performed; compensation for gyroscope coning effect is computed (see chapter 5.2.4).

    The output of the INS mechanization algorithm is the updated velocity vector, position vector (in both relative distance travelled and longitude, latitude and altitude terms) and attitude vector composed of Euler angles. Quaternion approach is used extensively for attitude representation and update.
3. **SMC mechanization:** After the INS mechanization cycle has proceeded, the SMC mechanization algorithm is initiated using the SMC data provided by SIMALAN/ALAN's internal sensors (see chapter 2.3.2). The SMC mechanization runs in an infinite cycle at the body frame rate as well and it consists of two steps:
    a. *The legged odometry computation*: the gait and its frequency are monitored and according to the predefined indicators, stride length and delta heading increments are computed for each gait period. Since the legged odometer output rate is given by the length of the gait period, no synchronization of the SIMALAN/ALAN's internal sensors is required. The output of the legged odometer over one gait period can directly be converted into the aiding signal (see next step) or stored in a buffer.
    b. *The aiding signal computation*: the SMC-INS aiding signal is updated using the legged odometer output as it comes, i.e. the legged odometer is asynchronous to the INS

system. The INS and SMC outputs are fused to form a measurement vector that enters the KF. At this stage, nonholonomic constraints are applied to improve the aiding signal, increase the observability of some states, and to bind the errors of the unaided states. If the output of the legged odometer was stored in a buffer previously, it can be sampled and the actual aiding frequency can be increased. However, this buffering introduces a fixed delay of one gait period length.

4. **SMC-INS error model update:** The output of the INS mechanization algorithm together with the parameters of the inertial sensor noise models are used at each time step to update the SMC-INS error model according to the trajectory travelled (see chapter 5.3). The SMC-INS error model is composed of two parts:
    a. Full-state INS error model which captures the position, velocity and attitude error dynamics described using the perturbation equations in chapter 5.3.
    b. Sensor noise models of variable parameters identified during offline sensor analysis (using PSD analysis or Allan variance analysis, see chapters 4.2.3 and 4.2.4 respectively). There are following noise models implemented, which can be chosen according to the specific sensors used: white noise model, random walk model, exponentially correlated noise model and autoregressive noise model of selectable order (no higher than order 10 should be chosen due to a strong increase in computation load).
5. **Kalman filter cycle:** There were two Kalman filter types implemented into the SMC-INS: the conventional extended Kalman filter (see chapter 4.5.2) and the sampling-based unscented Kalman filter (see chapter 4.5.4). According to the filter type chosen prior to initialization of navigation, the time update and the observation update proceeds. As described in chapter 5.3.6, the KF runs only as a predictor with the Kalman gain set to zero, if the measurement vector is not provided. If at least one KF update has occurred and if sufficient time has passed to build up the errors (this is controlled by specifying the filter feedback frequency), feedback to the INS mechanization algorithm triggers and correction of errors proceeds. Each time this feedback occurs, the errors are corrected and the state vector is reset to zero to prevent unbounded error growth. The actual SMC-INS output corresponds to the INS mechanization output but corrected for the estimated errors. In case there is no aiding signal available at all, navigation becomes based on the INS mechanization only, i.e. the SINS.

5.6.1 COVARIANCE ANALYSIS

The KF is a state estimator and its performance can be evaluated by the state estimate error covariance matrix [24 p. 217]. The main objective of the covariance analysis is to evaluate the performance using the *design model (*the suboptimal reduced-order model that is intended to be implemented), when using the data obtained from the *truth model* (the complex high-fidelity model of the real system). The details regarding the covariance analysis are described thoroughly by Farrell in [24 pp. 217-223] and can be exploited even further to perform the error budgeting [24 pp. 224-229].

The covariance analysis of the SMC-INS was done using the conventional KF equations for both the time and measurement covariance updates (chapter 4.5). These equations are dependent on the characteristics of the measured data (update rate, $\boldsymbol{Q}$, and $\boldsymbol{R}$), but independent of the actual measured data [24 p. 219]. There are two major reasons for doing so [24 p. 219]:

1. The covariance propagation accounts for the majority of filter computation. Such pre-computation is hence essential in case the computational capabilities are limited.
2. Covariance equations give the measure of the state estimation accuracy. This enables to run the covariance analysis in simulation prior to the hardware construction in order to judge the implementation tradeoffs.

As suggested above, the error model developed in chapter 5.3 was evaluated using the covariance analysis with respect to the selected 9 error states (3 position errors in NED frame, 3 velocity errors in NED frame, and 3 attitude errors in roll, pitch, yaw). The **Figure 5-10** shows the convergence of the 3 velocity errors and the **Figure 5-11** shows the convergence of the 3 position errors. Regarding the attitude errors, as shown in the **Figure 5-11**, pitch and roll errors are observable and converge as well; however, the yaw error diverges. This divergence is not caused by a mistake in the error model but by the unobservability of the yaw error state since no measurements are supplied to excite it. For detailed observability analysis of the proposed error model see next chapter 5.6.2.

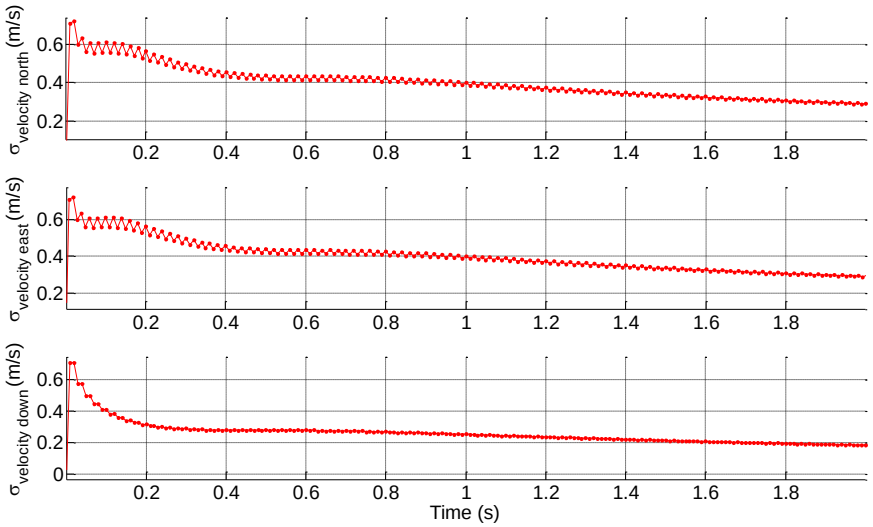


Figure 5-10 Covariance analysis of the developed error model – the velocity error states

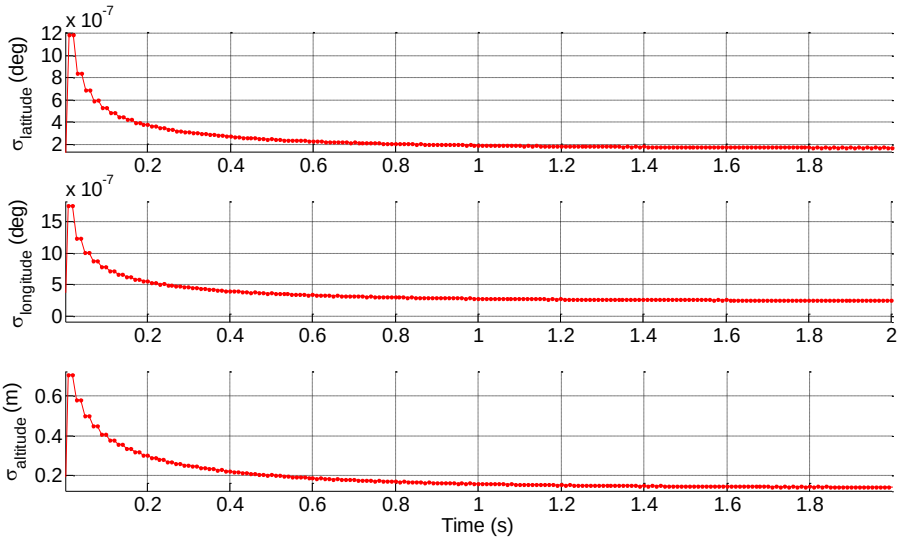


Figure 5-11 Covariance analysis of the developed error model – the position error states

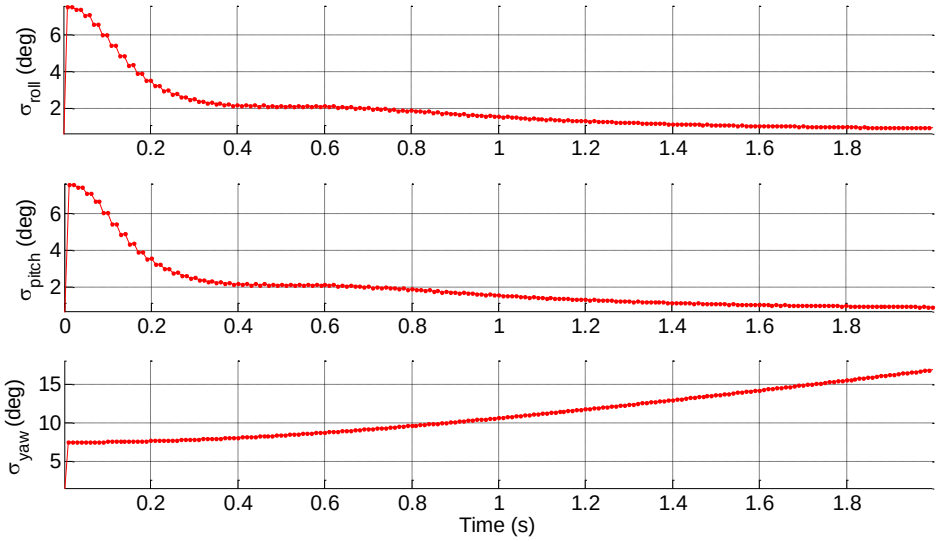


Figure 5-12 Covariance analysis of the developed error model – the attitude error states

Concluded, the covariance analysis is a method for analyzing performance of a filter design relative to a truth model. If the model represents the physical system inaccurately, the performance of the filter will not be optimal for the model [24 p. 230]. Although the covariance analysis may predict that the system will achieve the desired level of performance, the truly achieved level may differ. Assuming flawless implementation, the difference in the actual performance from the predicted one indicates that some important aspect of the system model has been neglected; typically it is the [24 p. 230]:

- neglecting unstable or marginally stable state,
- the process noise is too small hence the error variance and Kalman gain are too small,
- or a particular state is not sufficiently excited causing conditional unobservability.

5.6.2 OBSERVABILITY ANALYSIS

The observability analysis is a technique for examining any state estimator to determine, whether it's states can be estimated from the available outputs [24 p. 85]. The observability was continuously tested during the SMC-INS design. The testing was defined as a problem of estimating state $\boldsymbol{x}_k$ from a sequence of outputs $\{\boldsymbol{y}_i\}_{i=k}^{N+k}$ of a system described by the following state-space equations [24 p. 86]:

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Phi}_k \boldsymbol{x}_k + \boldsymbol{\Gamma}_k \boldsymbol{u}_k \quad \text{and} \quad \boldsymbol{y}_k = \boldsymbol{H}_k \boldsymbol{x}_k \tag{5-66}$$

For this discrete system, following set of equations was defined [24 p. 86]:

$$\boldsymbol{y}_k = \boldsymbol{H}_k \boldsymbol{x}_k \tag{5-67}$$

$$\boldsymbol{y}_{k+1} - \boldsymbol{H}_{k+1}\boldsymbol{\Gamma}_k \boldsymbol{u}_k = \boldsymbol{H}_{k+1}\boldsymbol{\Phi}_k \boldsymbol{x}_k \tag{5-68}$$

$$\vdots$$

$$\boldsymbol{y}_{k+N} - \boldsymbol{H}_{k+N} \sum_{j=0}^{n-1} \prod_{i=j+1}^{n-1} \boldsymbol{\Phi}_{k+i}\boldsymbol{\Gamma}_{k+j}\boldsymbol{u}_{k+j} = \boldsymbol{H}_{k+N}\boldsymbol{\Phi}_{k+N-1} \dots \boldsymbol{\Phi}_k \boldsymbol{x}_k \tag{5-69}$$

where all the quantities on the left-hand side were known and denoted as $\boldsymbol{Z}$ [24 p. 86]:

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{H}_k \\ \boldsymbol{H}_{k+1}\boldsymbol{\Phi}_k \\ \vdots \\ \boldsymbol{H}_{k+N}\boldsymbol{\Phi}_{k+N-1} \dots \boldsymbol{\Phi}_k \end{bmatrix} \boldsymbol{x}_k \tag{5-70}$$

The state vector was fully observable and the states $\boldsymbol{x}_k$ could therefore be estimated from the outputs $\{\boldsymbol{y}_i\}_{i=k}^{N+k}$ if and only if the matrix on the right-hand side had a full column rank [24 p. 86]. The above equation (5-70) was implemented for observability check of the proposed time-variant SMC-INS error model (see chapter 5.3). The results regarding observability of the 9 error states corresponding to the 9 navigation variables can be concluded as follows:

Aiding type	Aided variables	Rank	Comments
SMC with nonhol. constraints	position, velocity	9	Full convergence except for cases when the system is stationary – the yaw angle becomes unobservable
SMC only	position	8	lat., lon., attitude, v_e, and v_n covariance divergence
nonhol. constraints only	velocity	8	lon. covariance divergence, alt. unobservable
SMC, lateral nonhol. constraint	lat., lon., v_e	7	v_d and alt. covariance divergence
SMC, vertical nonhol.constraint	lat., lon., alt.	7-8	v_n and v_e covariance divergence

Table 5-3 Observability analysis of the proposed SMC-INS error model

Based on the proof in [20] it has to be noted that forward velocity is unobservable when certain degrees of freedom are not excited, i.e. when travelling along a straight path, without any pitching or yawing motion. Therefore the forward velocity was computed from the SMC measurements (see chapter 5.4.2) and provided as aiding besides the aiding in position.

6 EXPERIMENTS RESULTS

In this chapter, results regarding the inertial sensors modelling, inertial sensor signals de-noising, and evaluation of the adaptive filtering methods are presented.

6.1 INERTIAL SENSOR MODELLING AND ANALYSIS

The most crucial sensor errors were identified according to the methodology proposed in chapter 4.2; the Allan variance approach was chosen to do so since it proved superior to the PSD approach. The AVAR is a method of characterizing both the long-term stability and the broadband noise characteristics of a measured noise signal. The parameters of random processes chosen to model the noise signal can be ascertained from a graph of Allan deviation versus τ. For inertial sensors, the random walks and bias instability are considered as the principal errors. Hence the AVAR method was used to obtain parameters. The AVAR analysis was implemented in MATLAB and was employed on the AHRS M3 unit (Innalabs), 3DM-GX2 unit (MicroStrain), and MT9 and MTi-OEM units (both Xsens). At least 2 hours long noise measurements (7 hours for Xsens units) were analyzed. Sample plots for angular rate sensors and accelerometers of the MTi-OEM unit (Xsens) are presented in **Figure 6-1** and **Figure 6-2**:

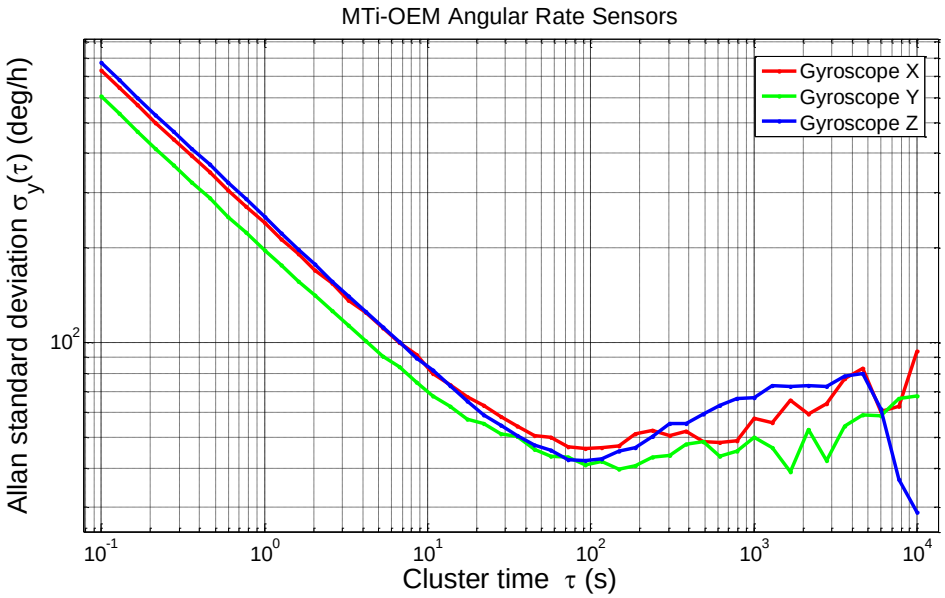


Figure 6-1 Allan deviation of the MTi-OEM Xsens angular rate sensors

The complete overview of all identified parameters is presented in the following tables: **Table 6-1** for 3DM-GX2 unit (MicroStrain), **Table 6-2** for AHRS M3 unit (Innalabs), **Table 6-3** for MT9 unit (Xsens), and **Table 6-4** for MTi-OEM units (Xsens). Parameters regarding the MTi-OEM unit were implemented into the developed SMC-INS (see chapter 5.3) because the MTi-OEM unit was the one used with ALAN. For each sensor noise parameter the one sigma error boundary was computed and is given in the tables. This precision in determining the sensor noise parameters was directly proportional to the time-length of the measured noise data – the longer the measurement the more

errors were identified more precisely. Simplified AVAR algorithm implementation for MATLAB is presented in **Matlab 8-11** in Appendix 8.5.

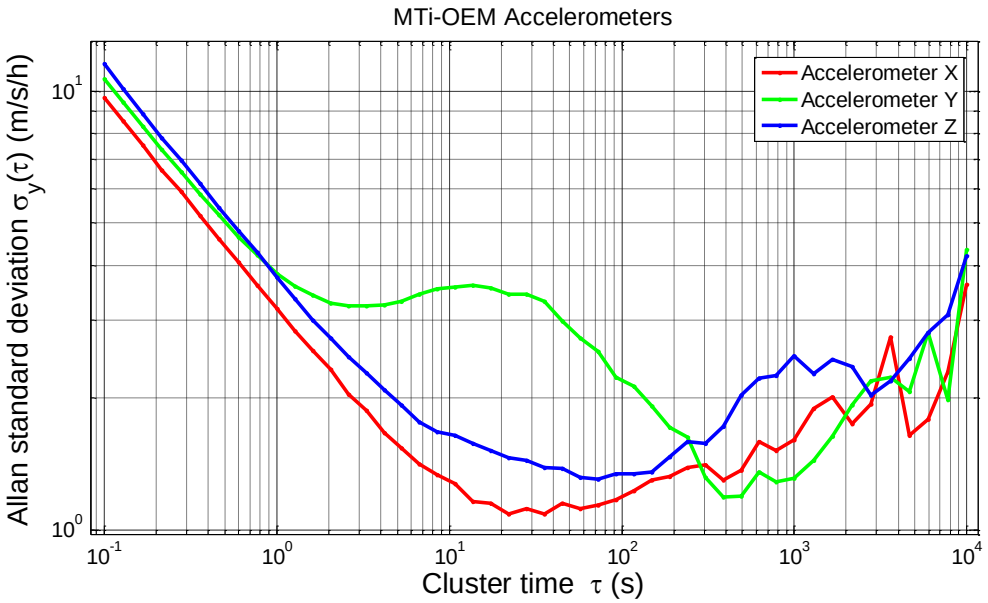


Figure 6-2 Allan deviation of the MTi-OEM Xsens accelerometers

Allan Variance Analysis of the 3DM-GX2 Navigation Unit				
SENSOR TYPE	**RANDOM WALK** [m/s/√h] resp. [°/√h]	VRW/ARW 1σ	**BIAS INSTABILITY** [m/s/h] resp. [°/h]	BI 1σ
Accelerometer (x-axis)	4.93E-02	5.19E-04	1.04E+00	1.07E-01
Accelerometer (y-axis)	5.23E-02	5.55E-04	1.30E+00	5.75E-02
Accelerometer (z-axis)	5.08E-02	5.40E-04	1.01E+00	7.22E-02
Angular Rate Sensor (x-axis)	1.95E+00	2.07E-02	2.32E+01	2.68E+00
Angular Rate Sensor (y-axis)	1.71E+00	1.82E-02	2.47E+01	2.85E+00
Angular Rate Sensor (z-axis)	1.74E+00	1.85E-02	2.28E+01	2.34E+00

Table 6-1 Allan Variance analysis of the 3DM-GX2 unit

Allan Variance Analysis of the AHRS M3 Navigation Unit				
SENSOR TYPE	**RANDOM WALK** [m/s/√h] resp. [°/√h]	VRW/ARW 1σ	**BIAS INSTABILITY** [m/s/h] resp. [°/h]	BI 1σ
Accelerometer (x-axis)	5.34E-02	3.63E-04	1.55E+00	3.92E-02
Accelerometer (y-axis)	5.31E-02	3.61E-04	8.62E-01	9.08E-02
Accelerometer (z-axis)	2.41E+00	1.64E-02	6.96E+01	1.23E+00
Angular Rate Sensor (x-axis)	2.30E+00	1.56E-02	3.72E+01	2.16E+00
Angular Rate Sensor (y-axis)	2.67E+00	1.82E-02	4.44E+01	4.16E+00
Angular Rate Sensor (z-axis)	2.40E+00	1.64E-02	3.40E+01	3.18E+00

Table 6-2 Allan Variance analysis of the AHRS M3 unit

Allan Variance Analysis of the MT9 Xsens Navigation Unit				
SENSOR TYPE	**RANDOM WALK** [m/s/√h] resp. [°/√h]	VRW/ARW 1σ	**BIAS INSTABILITY** [m/s/h] resp. [°/h]	BI 1σ
Accelerometer (x-axis)	8.65E-02	1.26E-03	2.85E+00	1.54E-01
Accelerometer (y-axis)	8.82E-02	1.28E-03	2.65E+00	1.82E-01
Accelerometer (z-axis)	8.76E-02	1.27E-03	1.92E+00	1.89E-01
Angular Rate Sensor (x-axis)	3.41E+00	4.96E-02	NA	NA
Angular Rate Sensor (y-axis)	3.39E+00	4.93E-02	7.85E+01	6.83E+00
Angular Rate Sensor (z-axis)	3.21E+00	4.66E-02	6.25E+01	7.75E+00

Table 6-3 Allan Variance analysis of the MT9 Xsens unit

Allan Variance Analysis of the MTi-OEM Xsens Navigation Unit				
SENSOR TYPE	**RANDOM WALK** [m/s/√h] resp. [°/√h]	VRW/ARW 1σ	**BIAS INSTABILITY** [m/s/h] resp. [°/h]	BI 1σ
Accelerometer (x-axis)	5.39E-02	4.59E-04	1.17E+00	5.89E-02
Accelerometer (y-axis)	6.35E-02	5.40E-04	3.26E+00	5.02E-02
Accelerometer (z-axis)	6.08E-02	5.17E-04	1.27E+00	9.12E-02
Angular Rate Sensor (x-axis)	3.90E+00	3.32E-02	4.82E+01	4.94E+00
Angular Rate Sensor (y-axis)	3.31E+00	2.81E-02	4.76E+01	4.89E+00
Angular Rate Sensor (z-axis)	4.19E+00	3.56E-02	3.38E+01	4.98E+00

Table 6-4 Allan Variance analysis of the MTi-OEM Xsens unit

6.2 DE-NOISING OF THE INERTIAL SENSOR OUTPUT DATA

The WMRA procedure introduced in chapter 4.3 was implemented as one of the possible de-noising methods for the SMC-INS. It was tested using position testing (chapter 6.2.1) to identify its optimal parameters. Then it was applied on real field-test data and evaluated (chapter 6.2.2).

6.2.1 RESULTS: WMRA POSITIONING TEST

The measuring setup for testing of inertial sensors (see Appendix 8.4) was used to evaluate the quality of de-noised attitude measurements for different parameters of the WMRA algorithm. The WMRA procedure (chapter 4.3) was tested and evaluated on the measured data. The pitch and roll angle measurements were recorded for different angular rates in such a way to cover the whole motion spectrum. Then, the WMRA algorithm was applied on all the data with various parameters and the RMSE was computed for the de-noised data compared to the reference signal from the precise incremental optical sensors. Using iterative approach, the optimal parameter setup for the WMRA algorithm was identified. RMSE of the noisy and the de-noised data with respect to the measured data were compared and served as the de-noising quality criterion. For example of optimal WMRA application on the roll angle data measured by AMHRS M3 unit at angular rate of 1 °/s see **Figure 6-3** and **Figure 6-4**.

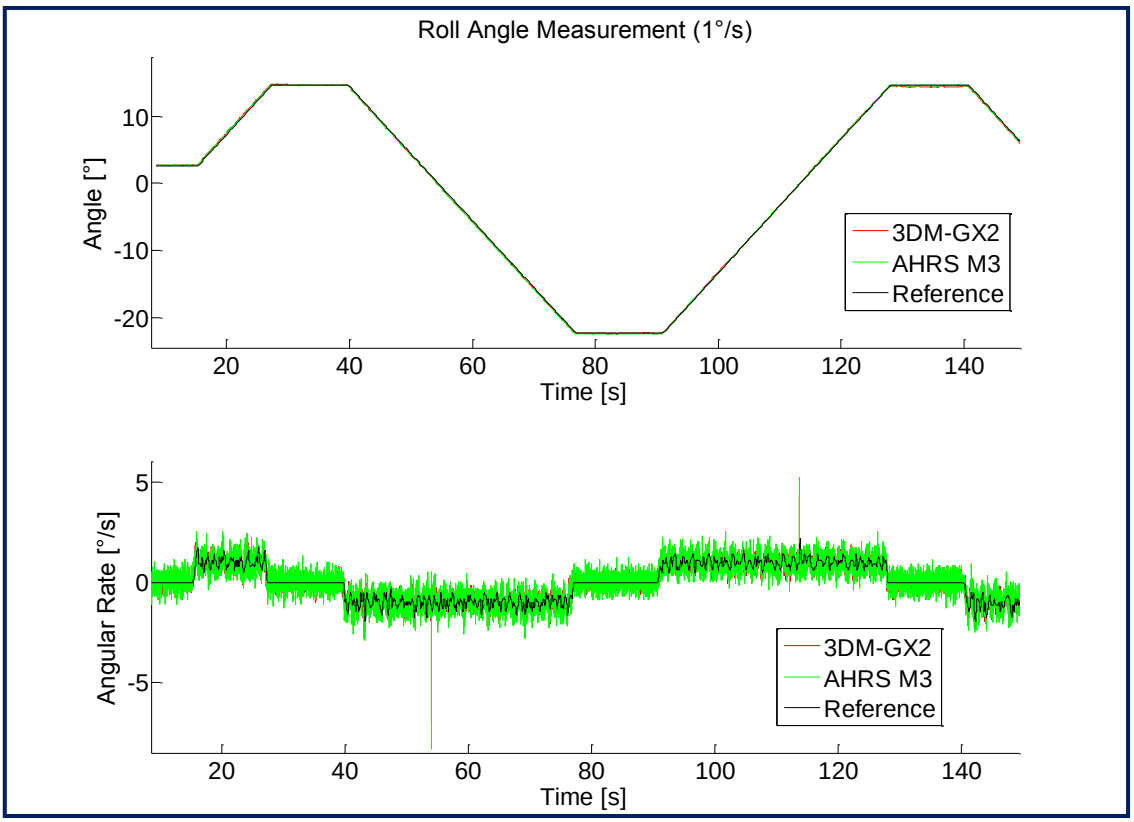


Figure 6-3 Example of roll angle data measured at angular rate of 1 °/s by AHRS M3 unit and the IRC sensors serving as reference

To carry out this analysis two of the MATLAB Toolboxes were used extensively: the *Wavelet Toolbox* for MATLAB and the *Signal Processing Toolbox* for MATLAB.

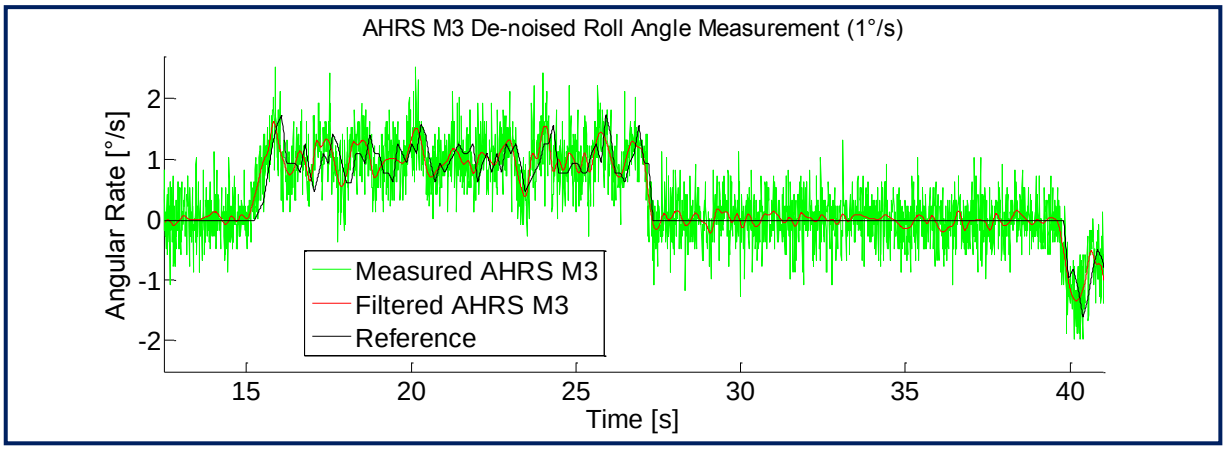


Figure 6-4 Example of de-noised angular rate data using optimal WMRA for AHRS M3 unit

Synchronization of the measured data was ensured using computation of the cross-correlation function to obtain the delay interval between data sequences. The WMRA algorithm was tested for different parameters available in MATLAB and compared to classical digital low-pass FIR filters of various order and cut-off frequencies using the same RMSE approach. In the final, the WMRA with optimal parameter setup outperformed all of the FIR filters but with a serious drawback. The WMRA proved to be much more hardware demanding than a simple digital FIR filtering. Conclusion can be drawn, that by the means of post-processing WMRA with optimal parameter setup should be applied. For real-time signal processing the selection of appropriated data de-noising procedure is strongly dependant on the hardware and computational load required by the rest of the navigation algorithm, given for example by the update frequency, model complexity, data rate etc. The optimal parameter setup is presented in **Table 6-5**; abbreviations and symbols in **Table 6-5** for the particular names of WMRA parameters are equivalent to the notation used in the *Wavelet Toolbox* for MATLAB.

	AHRS M3 (Innalabs)	**3DM-GX2 (MicroStrain)**	**MTi-OEM, MT9 (Xsens)**
Level of decomposition	**5**	**4**	**4**
Threshold selection rule	'**heursure**' - heuristic variant of the stein's unbiased risk selection threshold rule	'**minimaxi**' – for minimax thresholding	'**minimaxi**' – for minimax thresholding
Threshold type	'**s**' – soft	'**h**' - hard	'**h**' - hard
Threshold rescaling	'**sln**' - rescaling using a single estimation of level noise based on first level coefficients	'**one**' – no rescaling	'**one**' – no rescaling
Wavelet type	'**sym8**' – Symmetric wavelet	'**sym8**' – Symmetric wavelet	'**sym8**' – Symmetric wavelet

Table 6-5 Proposed optimal parameter setup for the WMRA de-noising algorithm

6.2.2 RESULTS: WMRA DYNAMIC FIELD-TEST

The WMRA procedure (chapter 4.3) was tested and evaluated on raw inertial data acquired during field-test using car navigation and the 3DM-GX2 unit (MicroStrain). Collected data were analysed for different parameter setups, given by the type of thresholding and the wavelet type, in the same manner as in case of WMRA positioning test (chapter 6.2.1).

The aim of this evaluation was to investigate the influence of different WMRA parameter combinations on the quality of data de-noising under real dynamic conditions that include car engine vibration. The optimal parameter combination for the WMRA procedure, as identified for both the AHRS M3 unit and the 3DM-GX2 unit (see **Table 6-5**), was investigated on the measured data and compared in the frequency domain. When the optimal set of parameters was applied, the frequency spectrum became clearly discriminated with respect to the applied level of decomposition. The motion dynamics became highlighted due to appropriate noise suppression. For any other parameter setup the motion dynamics became partially suppressed, hence the noise discrimination was not so clear.

For example: WMRA de-noising applied on the data acquired at 100 Hz using the accelerometer and the angular rate sensor aligned in the lateral axis of motion is presented in **Figure 6-5** (accelerometer) and **Figure 6-6** (angular rate sensor):

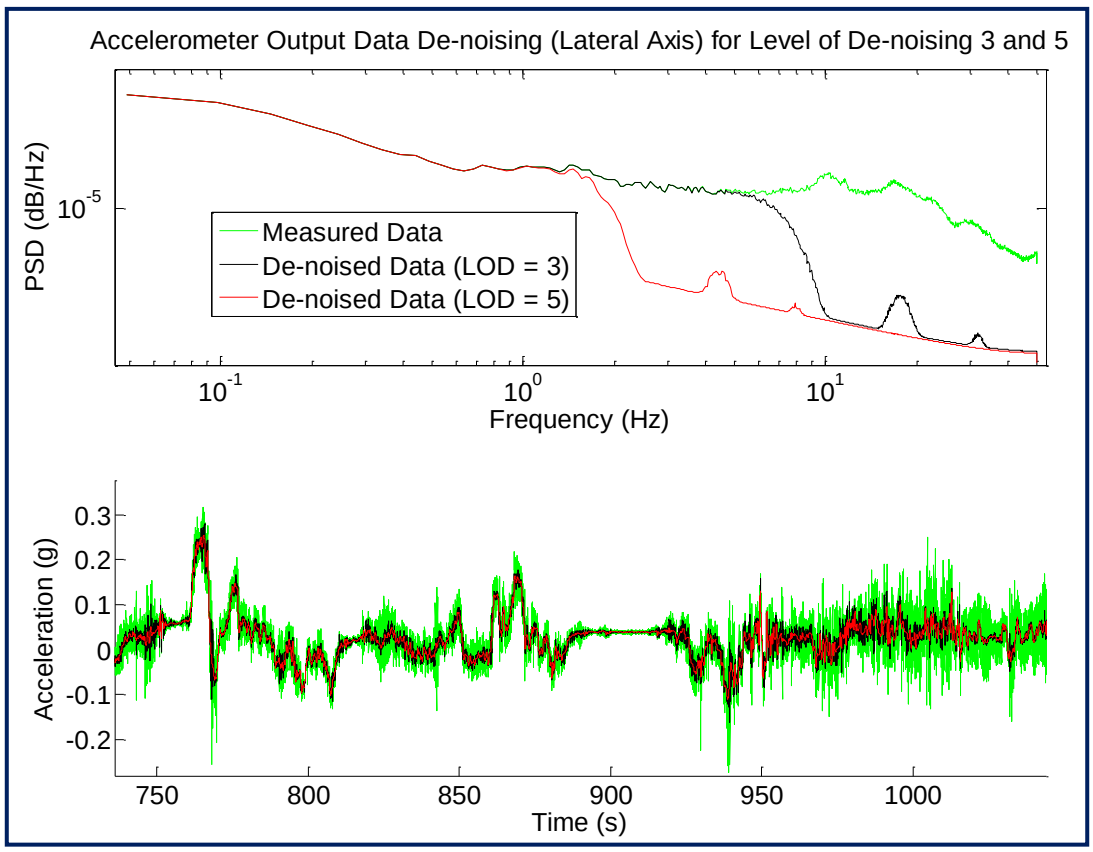


Figure 6-5 Example of accelerometer data de-noised using the WMRA procedure for level of signal decomposition 3 and 5 (parameters: *heursure, soft, sln, sym8,* sampling at 100 Hz)

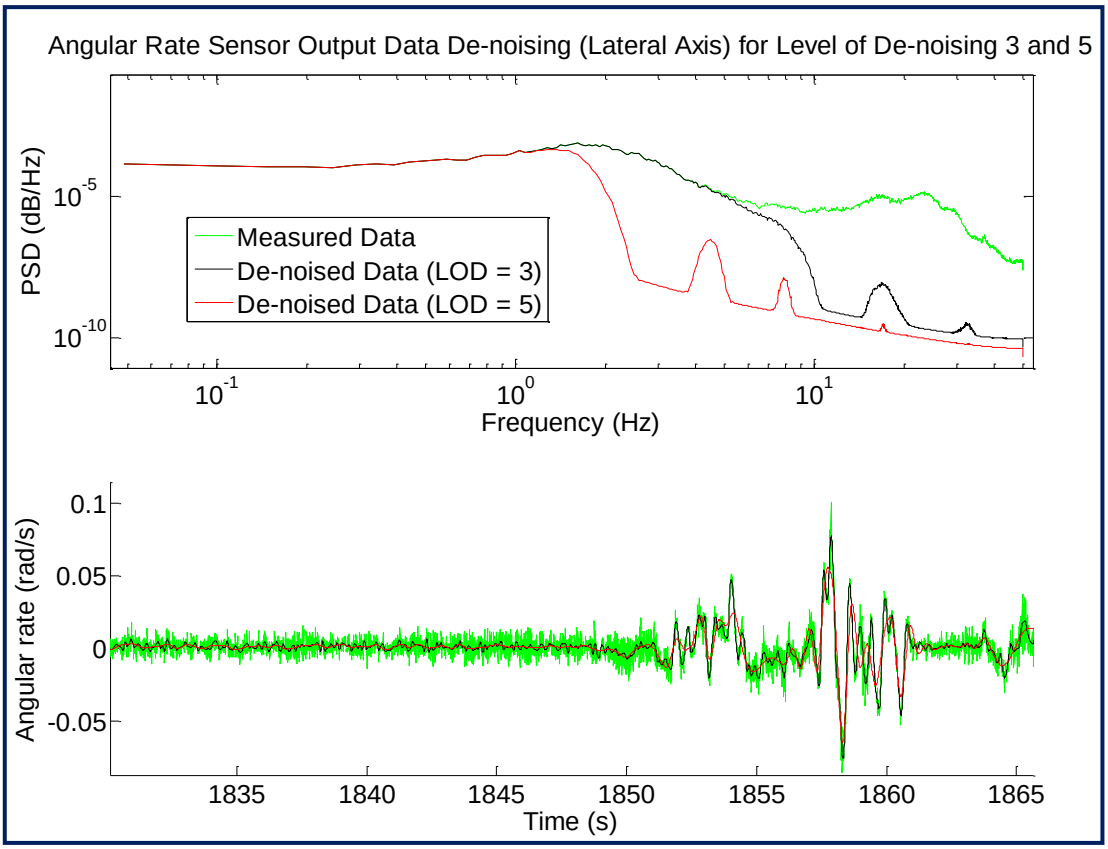


Figure 6-6 Angular rate sensor data de-noised using the WMRA procedure for level of signal decomposition 3 and 5 (parameters: *minimaxi, hard, one, sym8*, sampling at 100 Hz)

Therefore, a conclusion can be drawn, that the results obtained during the positioning test were confirmed by inspection for the dynamic test data and the WMRA proved to be a reliable de-noising procedure for navigation field-testing; of course by means of post-processing. This conclusion agrees with results presented by Hamid [34], who claims an overall position accuracy improvement of up to 60% for an integrated INS/GPS system [34 p. 103].

6.3 EVALUATION OF THE ADAPTIVE FILTERING METHODS FOR INS

When evaluating any navigation algorithm, two main aspects are usually considered:

1. The **stability of attitude angles** under dynamic conditions,
2. The **precision in estimation of trajectory of position** by means of RMS error between the true trajectory and the estimated one.

From the theory derived in chapter 4 it is obvious, that the attitude stability influences the position precision, but the position precision can be deteriorated not only because of attitude instabilities. This means that inertial systems weak in position precision can still be fully applicable for attitude determination purposes; for example to operate as an artificial horizon. However, inertial systems unstable in attitude determination are barely usable.

In chapter 6.3.1 the attitude stability is investigated by the means of experimental measurement using the real ALAN platform. The conclusions are then drawn for unaided INS and for both the EKF and the UKF aided SMC-INS.

In chapter 6.3.2 the SMC-INS is then analysed by simulation in Webots to evaluate the long-term tracking in environment of controllable friction and with precise referential trajectory. This long-term tracking is unfortunately unavailable for real ALAN due to the cable limitations.

In chapter 6.3.3, the SMC-INS is analysed by means of trajectory estimation using real ALAN platform with referential trajectory obtained from video tracking. Results are presented on two selected navigation examples: slow, steady turning gait, and combination of two completely different gaits. Comparison between the EKF based SMC-INS and the UKF based SMC-INS enhanced by WMRA, is presented on the same example. Conclusions are then made on the basis of over one hundred navigation experiments carried out with ALAN.

Regarding the real experiments using ALAN the MTi-OEM unit (Xsens) was used. The MT Manager software (**Figure 6-7**) provided by the Xsens manufacturer was used for logging of calibrated inertial data at 100 Hz. To align the orientation of the Xsens navigation unit with respect to the standard axis orientation the unit was mounted upside down.

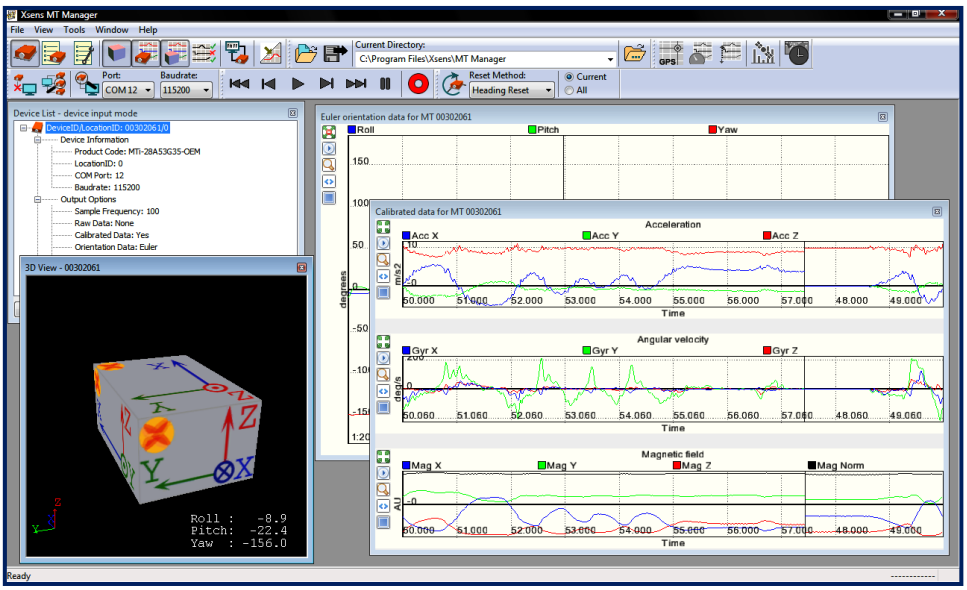


Figure 6-7 MT Manager (Xsens) software for calibrated inertial data logging and online visualisation

6.3.1 RESULTS: ATTITUDE STABILITY EVALUATION

First of all, the static attitude stability was evaluated using the rotational tilt platform (see Appendix 8.4). Results obtained using the stand-alone INS, i.e. using unaided INS mechanization algorithm, were largely influenced by the drifts of individual inertial sensors as expected according to the values specified in datasheet. Evaluation of the attitude stability using either the EKF or the UKF for aiding proved to be of no relevance since the attitude stability was strongly determined by the quality of the static aiding signal, which for static conditions was assumed almost ideal – sensor drifts were compensated completely and the observed attitude error was negligible.

Since such a static test is of very low relevance to the dynamic conditions of the real world during navigation, a special experimental setup was created to enable measurements under dynamic conditions. Robot ALAN was equipped with the MTi-OEM unit (Xsens) and placed on a constrained treadmill (see **Figure 6-8**).

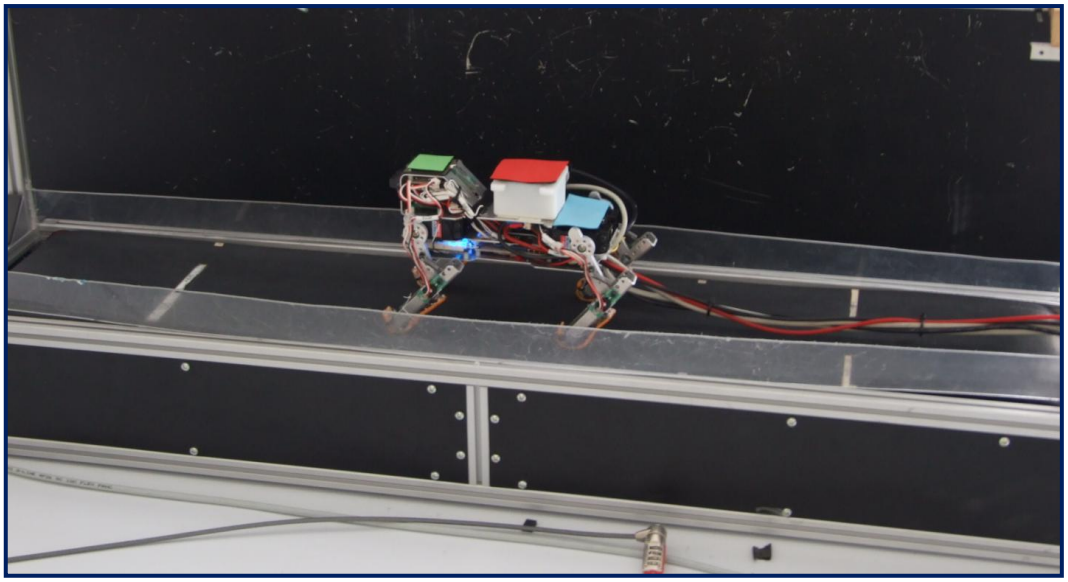


Figure 6-8 Experimental setup for attitude stability evaluation under high dynamics: constrained treadmill with speed regulation and ALAN with MTi-OEM inertial unit (Xsens)

The speed of the treadmill was controlled by a DC motor and ALAN's running gait was initiated. The speed regulation allowed to maintain ALAN's position fixed relative to the ground. Due to this feedback setup ALAN regulated its speed approximately to the same value as the speed of the treadmill. Although the yaw signal was deteriorated by ALAN's collisions with the constraints, the pitch and roll angles could easily be evaluated for the unaided INS mechanization only, and for both the EKF and the UKF aided SMC-INS algorithm (regarding the EKF/UKF algorithms, the performance was evaluated for various sensor noise models).

After performing the experimental run, the initial and the final pitch and roll values were expected to be approximately equal, because ALAN's static stand-still gait was applied in the beginning and at the end of the experiment. Of course, a difference of $\pm$ 5 deg was fully tolerable, because the conveyor belt was not ideally horizontal. **Table 6-6** shows sample results of one such dynamic attitude test. The data acquired were processed using different KF algorithms tuned for different sensor noise models. For the unaided INS mechanization approach the difference between

the initial and the final pitch and roll values, i.e. the error in attitude stability, was remarkably high; approximately 36.8 deg in pitch and 12.7 deg in roll after just 3 minutes of run. This was caused by uncompensated sensor errors integrated under high dynamics. Using the same inertial data the SMC-INS algorithm proved that such divergence in attitude can easily be compensated, reducing the difference to less than 1 deg in pitch and to approximately less than 4 deg in roll. Comparable results were obtained for both the EKF based SMC-INS and the UKF based SMC-INS.

ALGORITHM	NOISE MODEL	Pitch stability error (deg)	Roll stability error (deg)
SINS (MECH)		36.84	12.66
SMC-INS (EKF)	White Noise	0.20	3.54
SMC-INS (UKF)	White Noise	0.19	3.55
SMC-INS (EKF)	Random walk	0.23	3.92
SMC-INS (UKF)	Random Walk	0.22	3.90
SMC-INS (EKF)	Exp. Correlated	1.56	5.70
SMC-INS (UKF)	Exp. Correlated	1.58	5.64
SMC-INS (EKF)	AR order 2	0.15	3.57
SMC-INS (EKF)	AR order 3	0.20	3.64
SMC-INS (EKF)	AR order 5	0.17	3.71
SMC-INS (EKF)	AR order 10	0.14	3.77

Table 6-6 Evaluation of attitude stability under dynamic conditions with respect to processing algorithm

These results confirmed that the SMC-INS algorithm provides the desired attitude stability under dynamic conditions for both the EKF and the UKF regardless the sensor noise models. This attitude stability can more clearly be demonstrated on the comparison plots of the data processed by the INS mechanization only, i.e. unaided, and by the EKF based SMC-INS, see **Figure 6-9**.

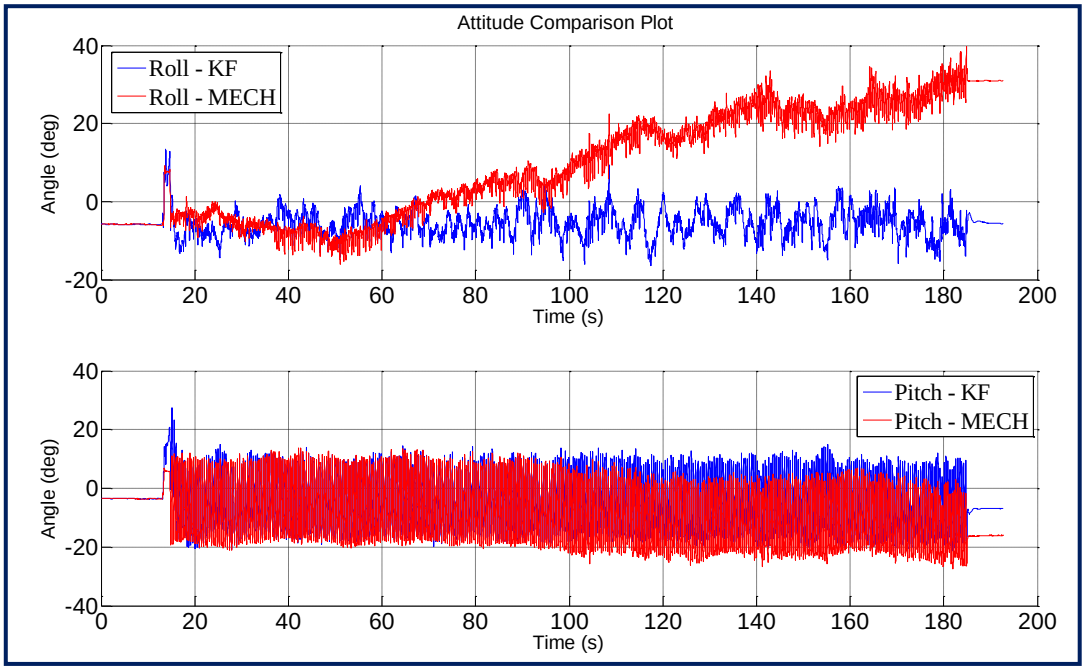


Figure 6-9 Attitude comparison plots: pitch and roll signals processed using INS mechanization only (red) and with legged odometer aiding using the EKF based SMC_INS (blue)

6.3.2 RESULTS: TRAJECTORY EVALUATION BY SIMULATION

The ability of the SMC-INS navigation to provide accurate attitude estimates was proven in the previous chapter. The compensation of attitude drift was successfully evaluated for different SMC-INS configurations, regarding inertial sensors noise models and the KF type. However, the main intention was to evaluate the whole concept, including real application, with respect to the estimation of position, i.e. the motion trajectory. This was done in the first place using the Webots simulation environment (see chapter 2.3.3).

Although the Webots simulation provided only limited clues about the real world application the opportunity to parameterise the simulation for different conditions as well as the opportunity to have a true precise reference trajectory for data of almost any length was priceless. The mentioned parameterization mainly concerned the trajectory specifications, change in the strength of gravity, accommodation of the surface friction and shape of the terrain, and definition of the sensors noise type and level.

The Webots environment (see **Figure 6-10**) was used to prove the concept of SMC-INS by experiments concerning trajectory estimation. All of the experiments were based on the physical model of ALAN's body, the SIMALAN[5], equipped with a virtual inertial measurement unit mounted with the exactly same orientation as on real ALAN. The gravity field strength in Webots was set correspondingly to the local gravity value. A large number of simulations was performed for different tracks, surface friction, and sensor noise settings.

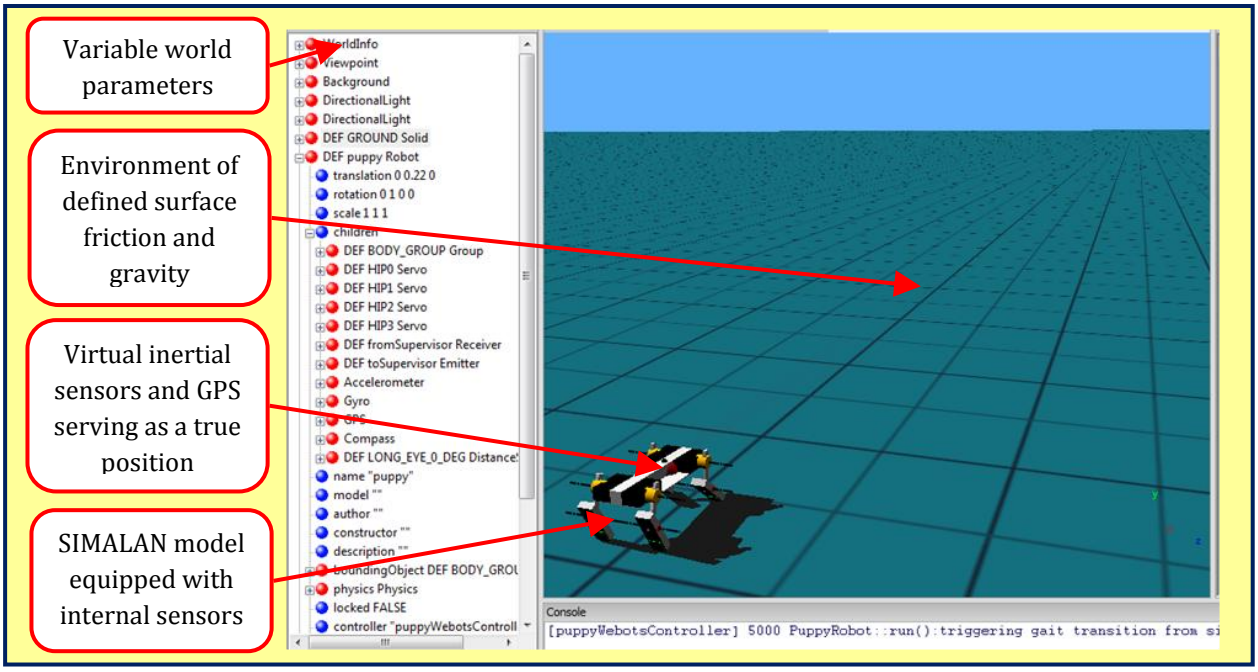


Figure 6-10 SMC-INS concept verification using SIMALAN simulation in Webots

To demonstrate the SMC-INS verification on data collected in Webots an experiment consisting of the following sequence of gaits was performed: *sit-still (gait 1), sprint gait (gait 0), turn pacing right*

[5] Model developed by M. Hoffmann, University of Zurich, AI Laboratory, as part of the project *From locomotion to cognition*, grant Nr. 200020-122279/1, supported by Swiss National Science Foundation

(gait 3), sprint gait (gait 0), turn on spot right (gait 5), sprint gait (gait 0), turn pacing right (gait 3), and sprint gait (gait 0), i.e. 3 different gaits and 7 transitions (see **Table 5-1** for the gait details). Before executing the simulation, it was necessary to specify the gait evolution through time such that a track could be plotted. Every simulation started from the resting *sit-still* gait to provide sufficient amount of stationary data for the initial alignment. This was the same for the experiments with real ALAN. The inertial data and the sensors providing the self-motion cues were logged in the simulation at given rate, usually at 500 Hz. After the simulation in Webots was finished the SMC-INS algorithm was executed in MATLAB. Measured accelerations and angular rates collected using the virtual inertial measurement unit in Webots are shown in **Figure 6-11**.

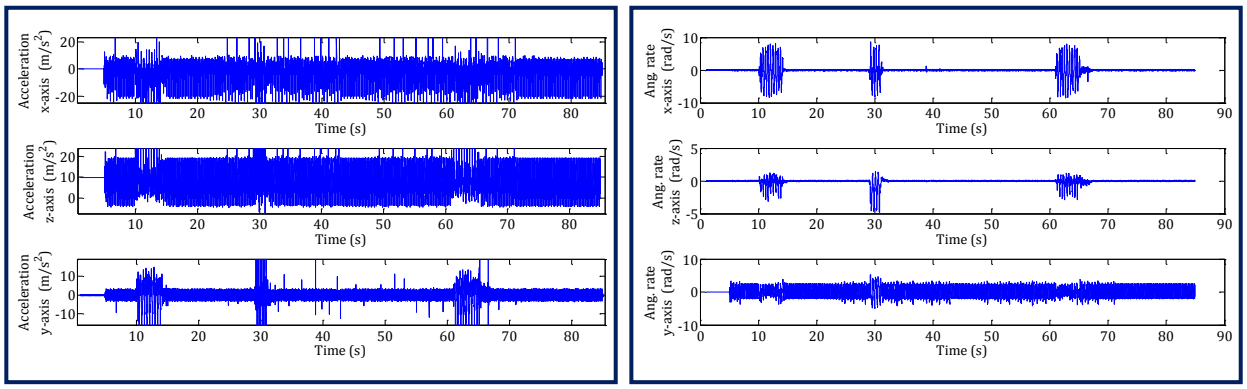


Figure 6-11 Raw accelerations and angular rates data collected using the Webots inertial measurement unit

In the SMC-INS the raw inertial data in the **Figure 6-11** were processed by the INS mechanization and fused with the output of the legged odometer in the **Figure 6-12** that was computed from the self-motion cues data. Using for example the EKF the sensor errors were estimated and the corrected navigation variables were obtained as presented in the **Figure 6-13**; estimated attitude angles compared to the true reference are shown in the **Figure 6-14**:

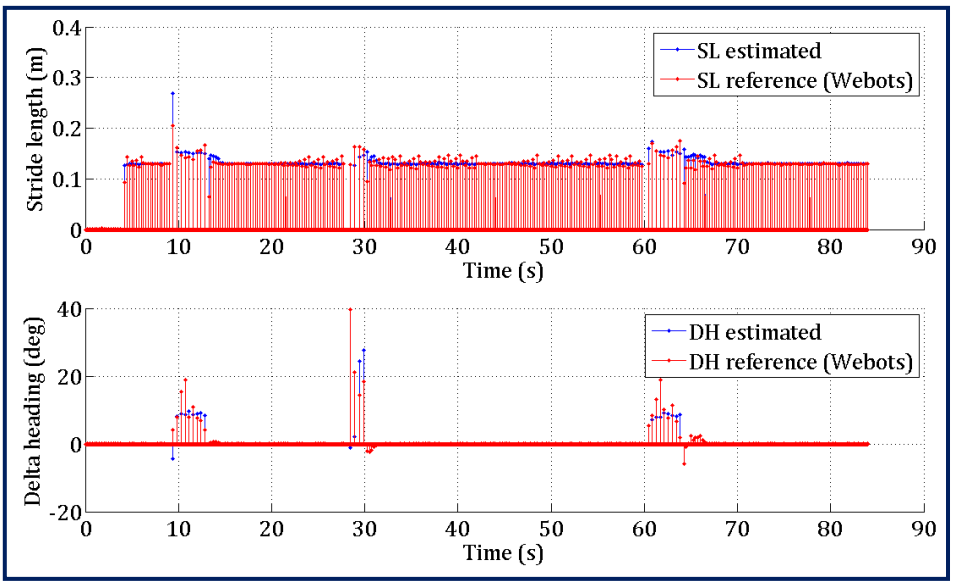


Figure 6-12 Legged odometer data obtained from SMC data measured in Webots; estimated stride length and delta heading (blue), true stride length and delta heading reference (red)

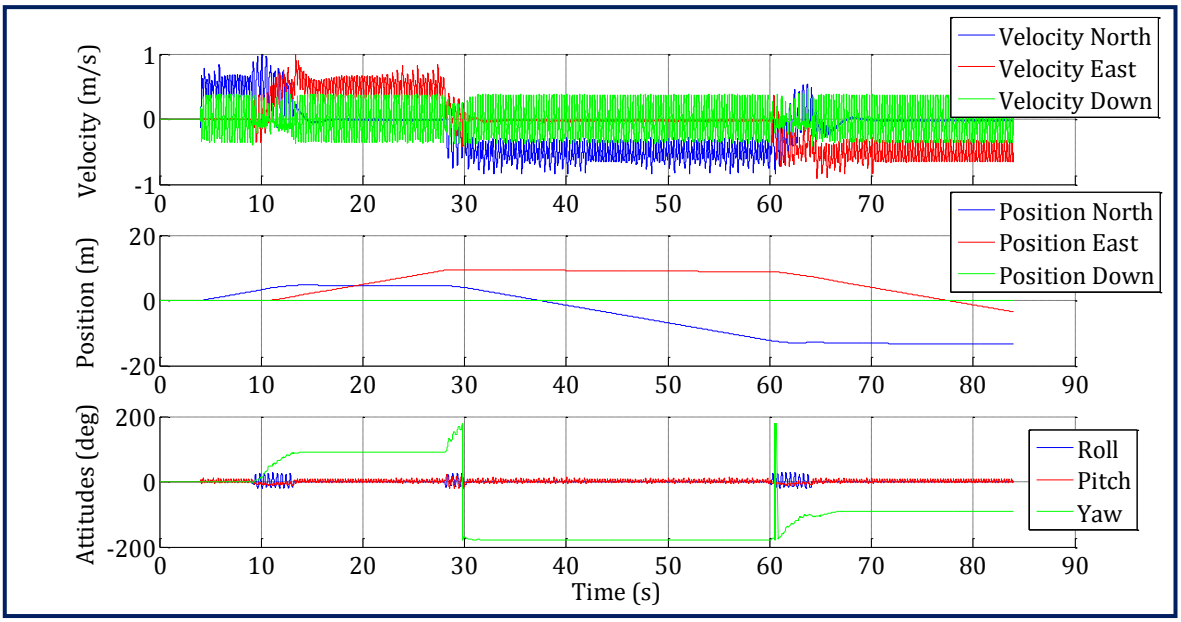


Figure 6-13 Estimated velocity, position and attitude results regarding the SMC-INS applied on Webots data

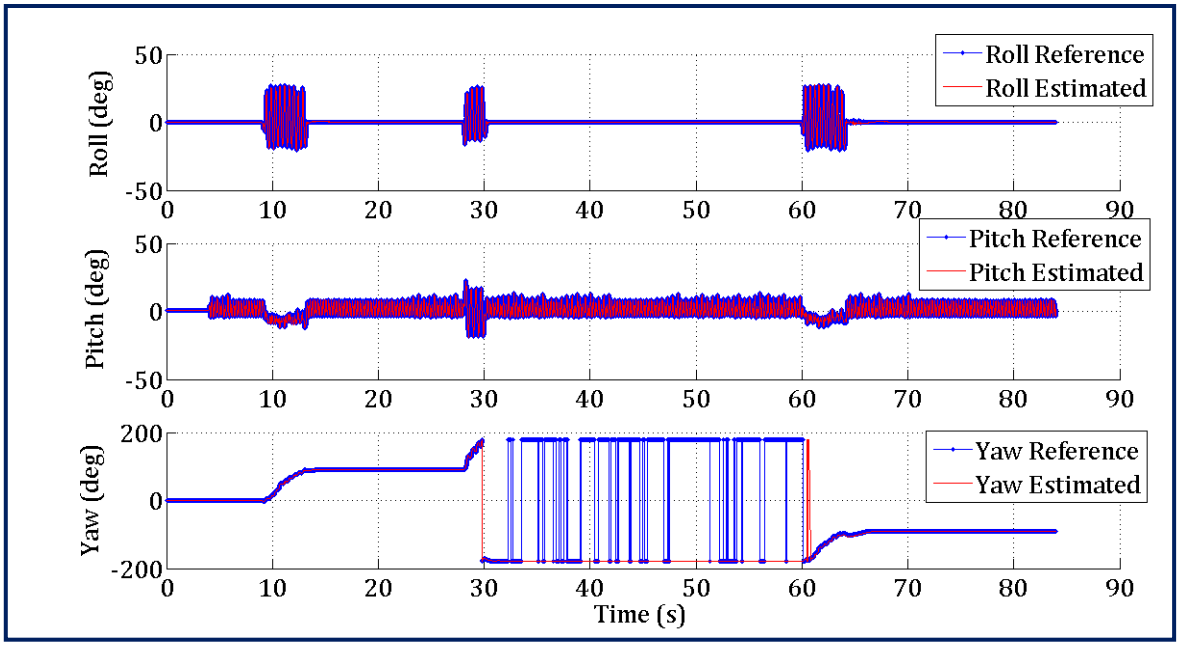


Figure 6-14 Attitude comparison plots regarding the SMC-INS algorithm applied on Webots data; estimated SMC-INS attitude signals (blue), true attitude reference (red); RMSE less than 0.3 deg for all angles

The Webots environment gave a priceless merit for the actual SMC-INS development since it provided absolutely precise reference for both the relative position in the navigation frame and the attitude angles. The position and attitude angles could easily be compared in the RMSE sense and the performance of the SMC-INS for different configurations was evaluated. The attitude comparison to the precise reference, as presented in **Figure 6-14**, proved that the actual RMSE values for all three attitude angles were less than 0.3 degree.

The position determination precision was evaluated in a similar way for both the absolute position, based on the initial latitude, longitude, and altitude, and for the relative position in the navigation frame. Continuing with the example, corresponding position comparison plot is presented in **Figure 6-15**. The track computed using the standalone INS, i.e. the INS mechanization algorithm only (the blue line), was compared to the aided SMC-INS output (the red line), and to the precise reference (the green line). For these plots, the overall RMSE errors between the SMC-INS estimates and the reference were as follows: 0.72 m in northern direction, 0.53 m in the eastern direction and 0.08 m in the down direction.

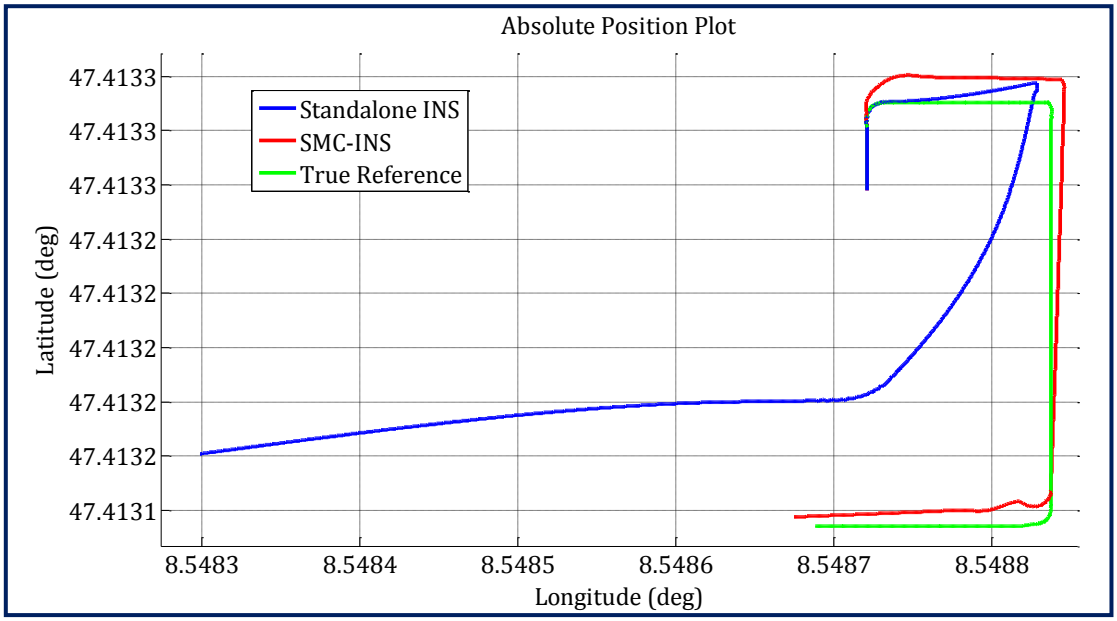


Figure 6-15 Absolute position plot regarding the SMC-INS applied on Webots data

Concerning all of the Webots navigation experiments, the actual precision of navigation was not as important. Far more important it was the conclusion that the concept of the SMC-INS was proven by these simulations and the SMC-INS worked as expected with satisfying precision.

However, not only the concept of the SMC-INS was proven, but also the navigation experiments helped to identify the main possible sources of error. The most crucial weakness of the SMC-INS navigation was proven to be the delta heading indicator for the legged odometry, i.e. the aiding during turning gaits, especially the ones carried out at high dynamics (the gaits 4 and 5). This error was caused by the gait transitions, especially by their timing, since discontinuous jumps in the self-motion cues sensor signals were observed. Although this behaviour of sudden gait changes was correct from the real-world point of view, the implementation using sudden changes in motor signals caused serious errors. The effect of this error was usually observed in the heading estimates right after the turning gait had ended. If a long straight gait followed such turning gait, the error in heading obviously accumulated with the distance travelled. The example of the navigation data presented in this chapter shows the manifestation of this error during the transition of the *turning on spot right* gait (the second turning gait) into the *sprint* gait. Clearly, the heading during the transition was badly estimated and the path slowly diverged from the reference trajectory; as shown in **Figure 6-15** and **Figure 6-16**. **Figure 6-13** and **Figure 6-14** also show that this error was not caused by a drift in the angular rate sensors since the estimated yaw angle did not diverge and

stayed constant during all of the *sprint* gaits. However, this constant yaw value was still biased by this transition ill-estimate. A possible solution to cope with this error is supplying additional sensory information; for example aiding using occasional visual landmarks as proposed in [12].

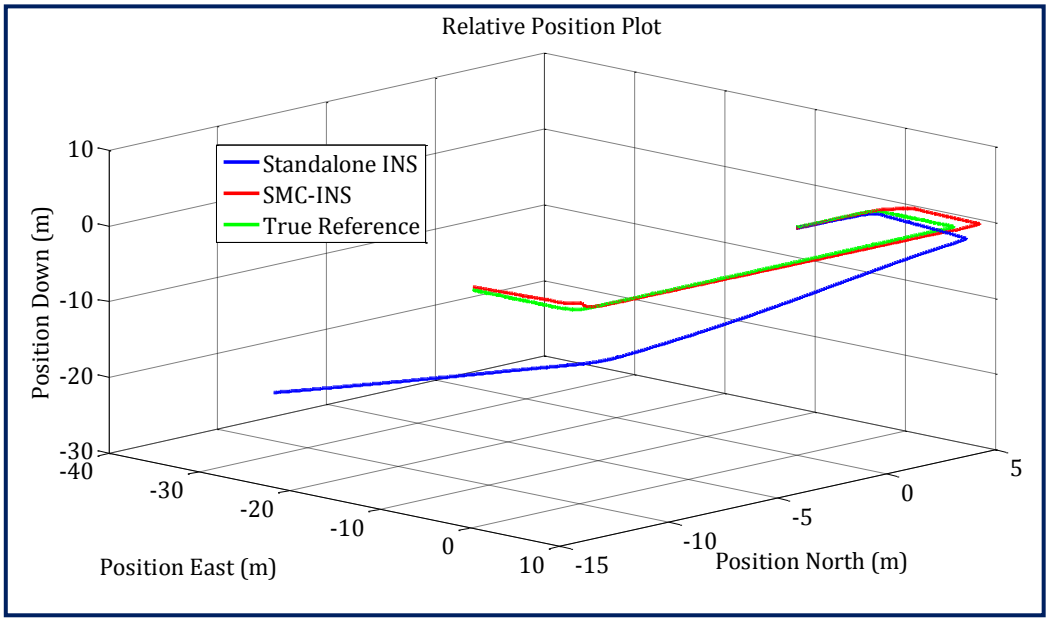


Figure 6-16 Relative 3D position plot regarding the SMC-INS applied on Webots tracking data

Another possible source of error was discovered in the simulation directly in the raw inertial data generated by the ODE engine in Webots. Occasionally, random high peaks in the acceleration values were observed, especially during the simulations of feet collision with the ground. Although these peaks were removed as extraneous data since they could be easily distinguished, it was a flaw that did not correspond to the real world behaviour. However, for the verification of the concept and for the SMC-INS algorithm evaluation Webots were otherwise well suitable and more than useful.

A second example of SMC-INS algorithm verification is presented in the same manner as the previous example and is shown in **Figure 6-17** and **Figure 6-18**. A long-term navigation experiment with one single high dynamics turning gait (gait 3) was simulated. **Figure 6-17** shows in detail the estimated velocity, position and attitude results regarding SMC-INS applied on Webots data. **Figure 6-18** then presents the comparison of the same estimated SMC-INS position plot in relative position (blue line) and the true reference provided by Webots (red line). For these data plots, the overall RMSE errors between the SMC-INS estimated trajectory and the reference were as follows: 0.33 m in northern direction, 0.35 m in the eastern direction and 0.08 m in the down direction, 0.32 deg in roll and 0.28 deg in pitch angles. Assuming the body length of the SIMALANA/ALAN was approximately 0.25 m, the position error close to one body length was fully acceptable. Hence, the Webots simulation again provided proof for the SMC-INS concept even for a long-term high dynamics turning gait, which was expected to be the greatest challenge.

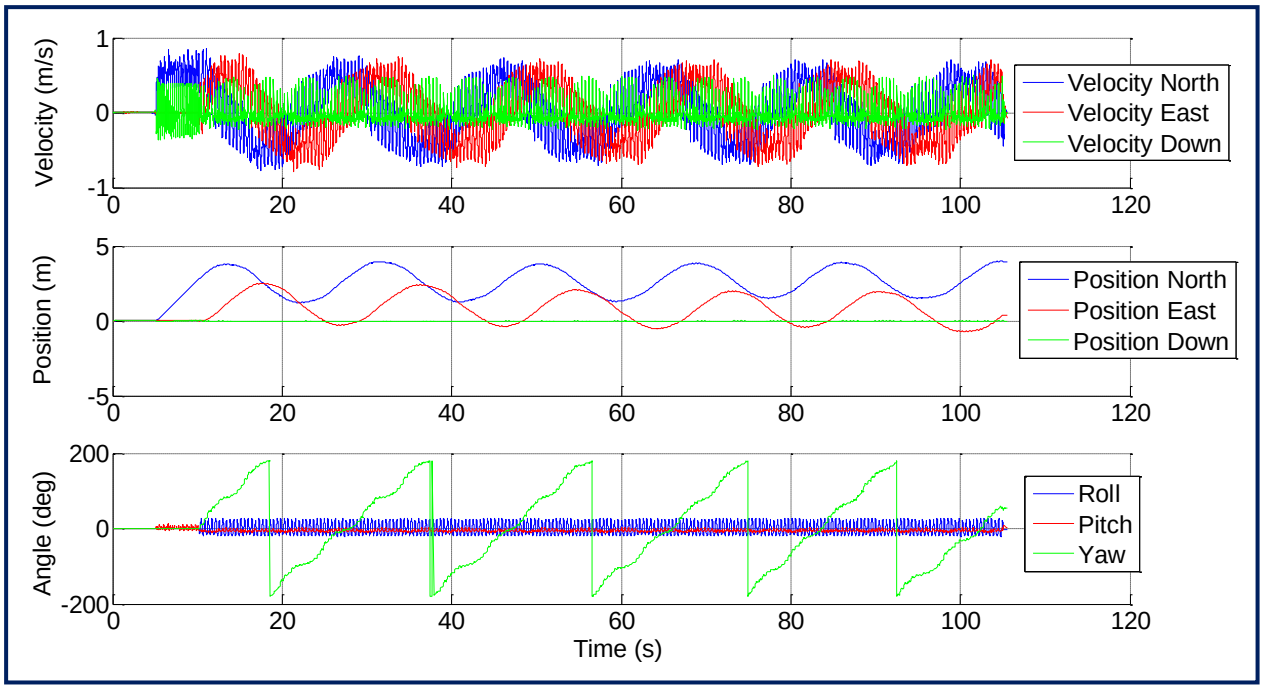


Figure 6-17 Estimated velocity, position and attitude results regarding the SMC-INS applied on long round-track Webots data

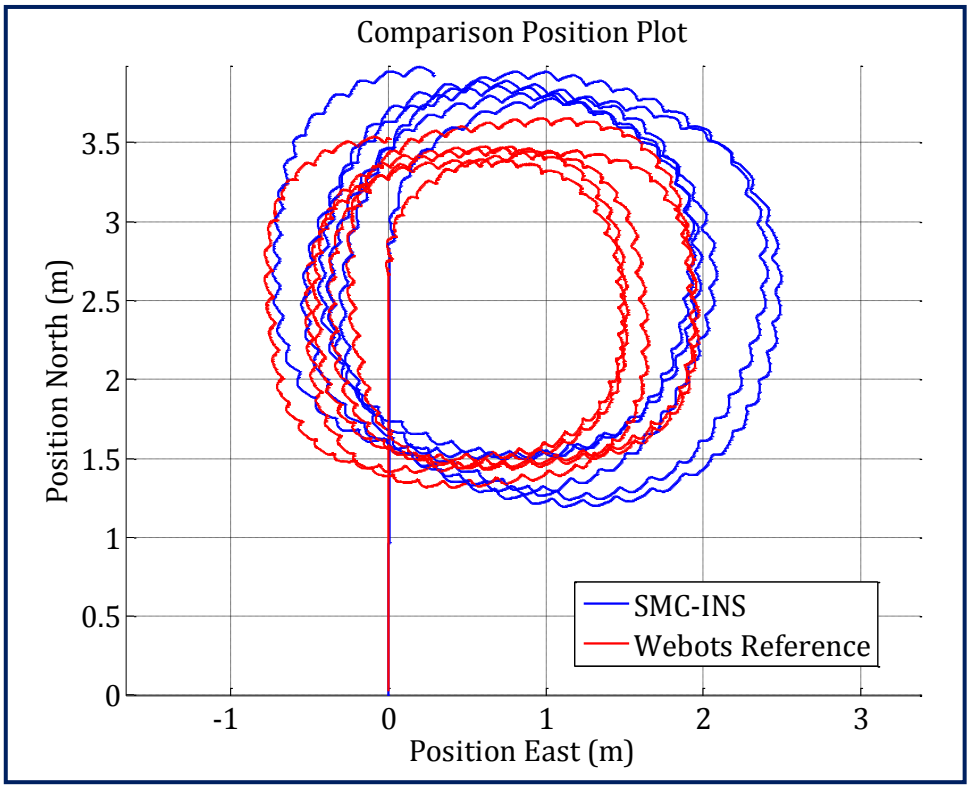


Figure 6-18 Relative position plot regarding the SMC-INS applied on Webots turning gait data; estimated SMC-INS track (blue), Webots true reference (red)

6.3.3 RESULTS: TRAJECTORY EVALUATION BY FIELD-TEST

In this chapter, the goal is to present some typical sample results regarding SMC-INS trajectory estimation for both the EKF and the UKF approaches applied on real data. More than 100 navigation experiments were carried out in an arena monitored by a ceiling camera for motion tracking to provide referential trajectory. The real navigation data were acquired using the ALAN robotic platform equipped with the MTi-OEM inertial measurement unit (Xsens); see **Figure 6-19**. This IMU unit provided the raw but calibrated inertial data. ALAN and its sensors provided the SMC data necessary for computation of the output of the legged odometer, i.e. the stride length and the delta heading information, necessary for aiding. There are three examples of navigation experiments to be presented in this chapter:

1. First example covers the slow, steady gait for continuous turning left that is similar to the Webots simulation presented in **Figure 6-18**.
2. Second example covers a combination of two different gaits: highly dynamical gait for turning right followed by a slow gait for turning left with large turning radius. This experiment was one of those aiming at providing analysis of gait transitions as well as the EKF and the UKF performance comparison. Gait frequencies used in this experiment were not included in the data set used for training the indicators; hence generalization of the legged odometer to the gait frequency is presented.
3. Third example describes the robustness of the SMC-INS to slippage for the same combination of the two turning gaits as in previous example. This was a typical experiment used to show that SMC-INS is robust to slippage even on a terrain with significantly low friction. Generalization to gait frequencies outside the original indicator training set is also presented.

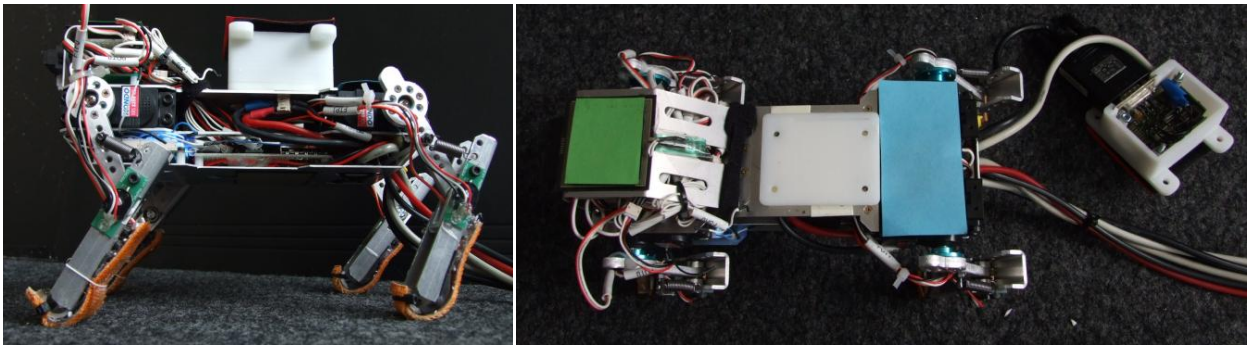


Figure 6-19 Quadruped robotic platform ALAN equipped with MTi-OEM unit (Xsens)

FIELD-TEST: TURNING LEFT GAIT

In this chapter the overall results regarding field-testing of the UKF based SMC-INS algorithm on slow, steady turning left gait are presented. The duration of the gait was approximately 2 minutes and the gait frequency was 0.75 Hz. The referential trajectory was obtained using free video tracking software Tracker 3.1[6]; see **Figure 6-20**. The raw but calibrated inertial data are shown in **Figure 6-21**. Velocity, position, and attitude results are presented in **Figure 6-22**. The estimated trajectory, the aiding signal and the referential trajectory are plotted in **Figure 6-23**. The precision of the estimated trajectory by means of RMSE development in time is in **Figure 6-24**.

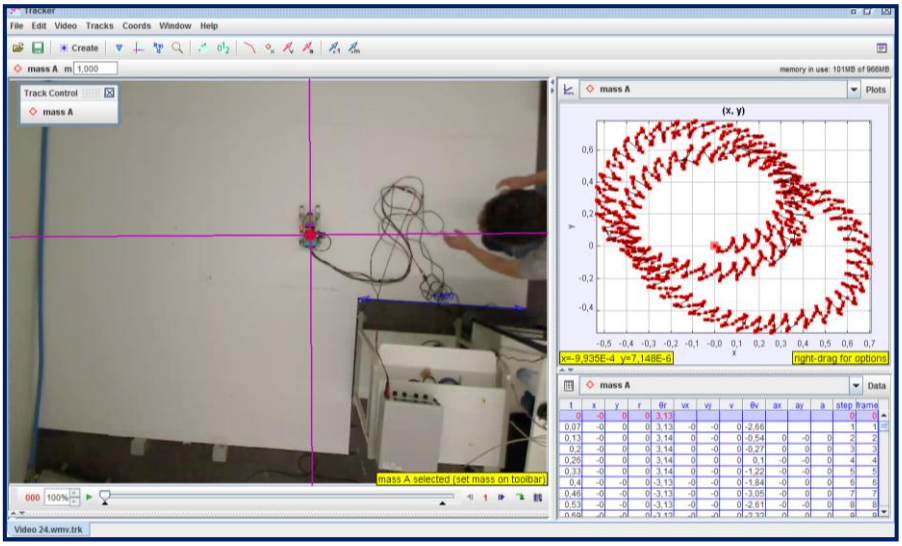


Figure 6-20 Tracker 3.1: free video tracking software for computing precise reference trajectory

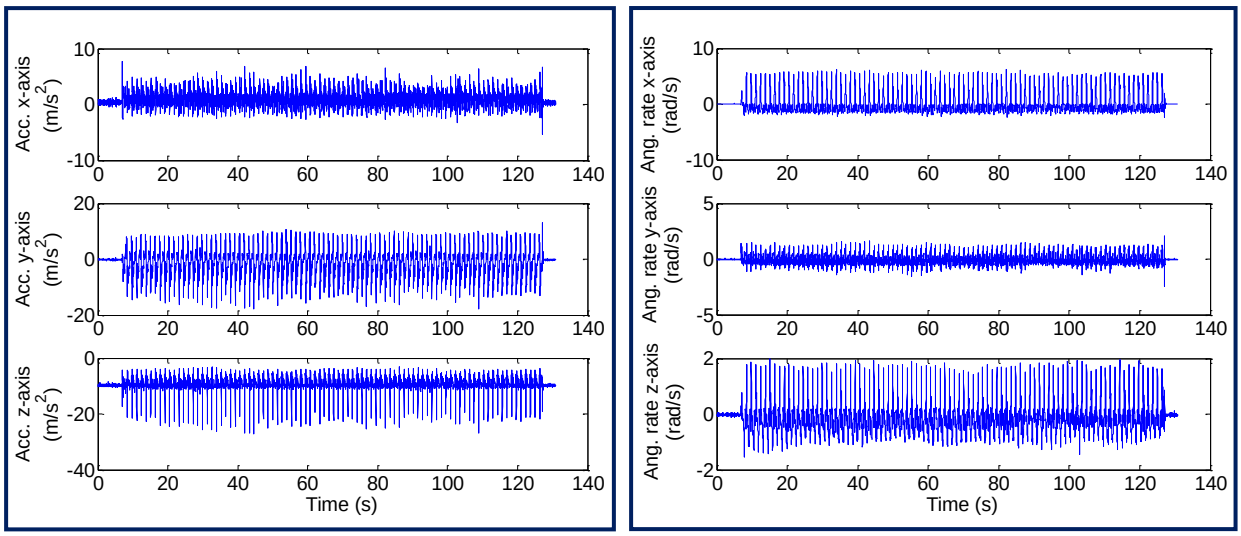


Figure 6-21 Accelerations and angular rates for slow turning left gait measured by the MTi-OEM unit (Xsens)

[6] Developed by Douglas Brown, downloadable at http://cabrillo.edu/~dbrown/tracker/ (published in this dissertation with author's permission).

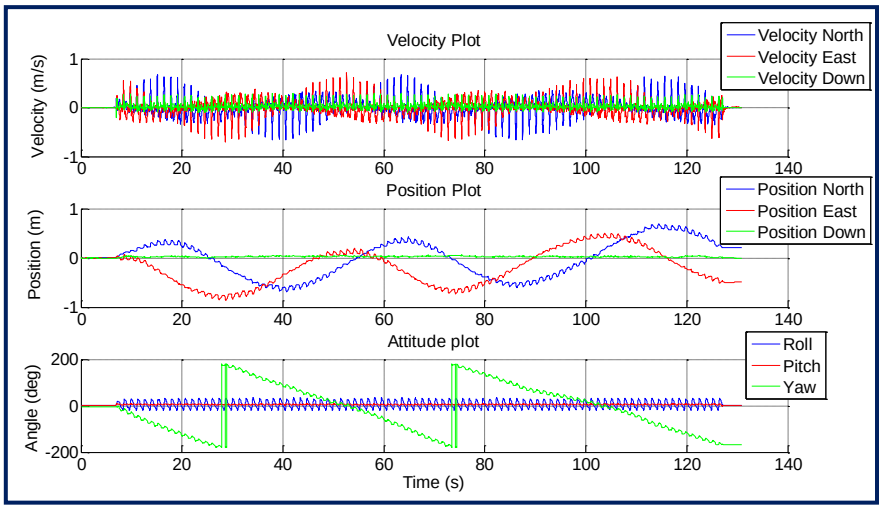


Figure 6-22 Velocity, position, and attitude results regarding SMC-INS application on real inertial and SMC data acquired with ALAN during turning left gait

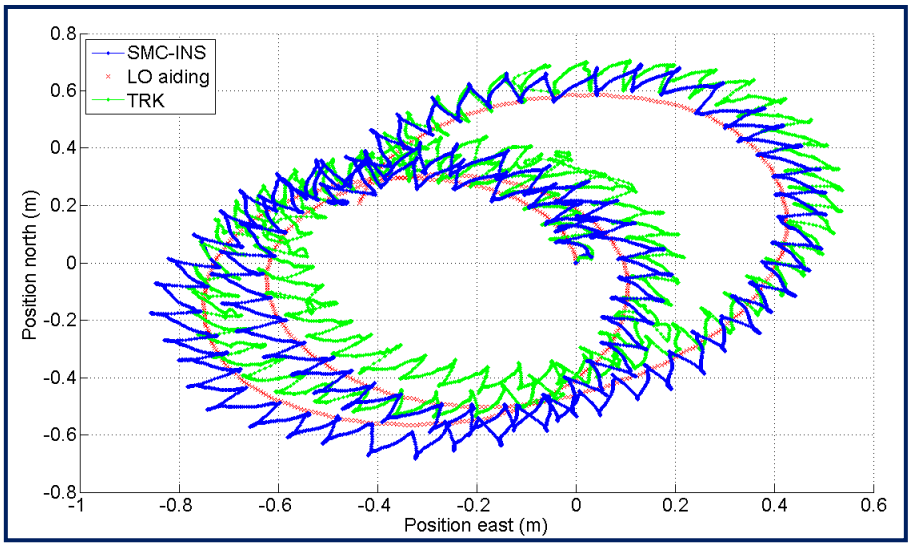


Figure 6-23 Trajectory results regarding SMC-INS applied on real inertial and SMC data measured during turning left gait (includes SMC-INS trajectory, LO legged odometer aiding signal and TRK video reference)

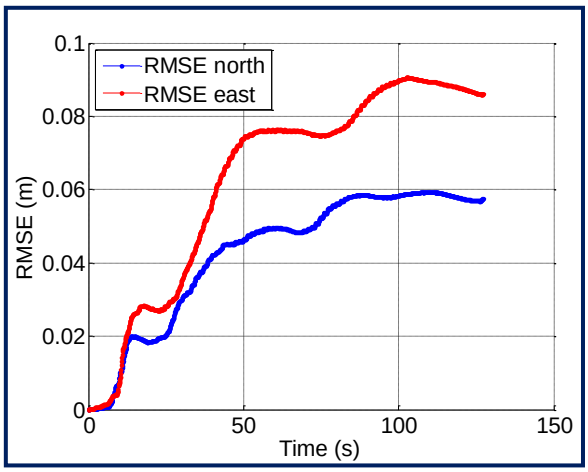


Figure 6-24 Precision of the estimated trajectory in time by means of RMSE for turning left gait

FIELD-TEST: COMBINED GAIT

In this chapter the overall results regarding field-testing of the SMC-INS algorithm for combination of two different gaits (dynamical turning right gait followed by slow turning left gait with large turning radius) are presented. The duration of the combined gait was approximately 15 seconds and the gait frequency was 1 Hz for both gaits. The limited duration was given by the size of the arena and by the reach of ALAN's cables. The measured raw but calibrated inertial data are shown in **Figure 6-25**. Velocity, position, and attitude results regarding the UKF based SMC-INS are presented in **Figure 6-26**. In this case, the performance comparison of the EKF and the UKF enhanced with WMRA de-noising procedure was evaluated. The effect of data de-noising is shown in **Figure 6-27** and **Figure 6-28**. The trajectory estimated by the EKF and the UKF, the aiding signal, and the reference are compared in **Figure 6-29**. The precision analysis by means of RMSE plots is presented in **Figure 6-30**.

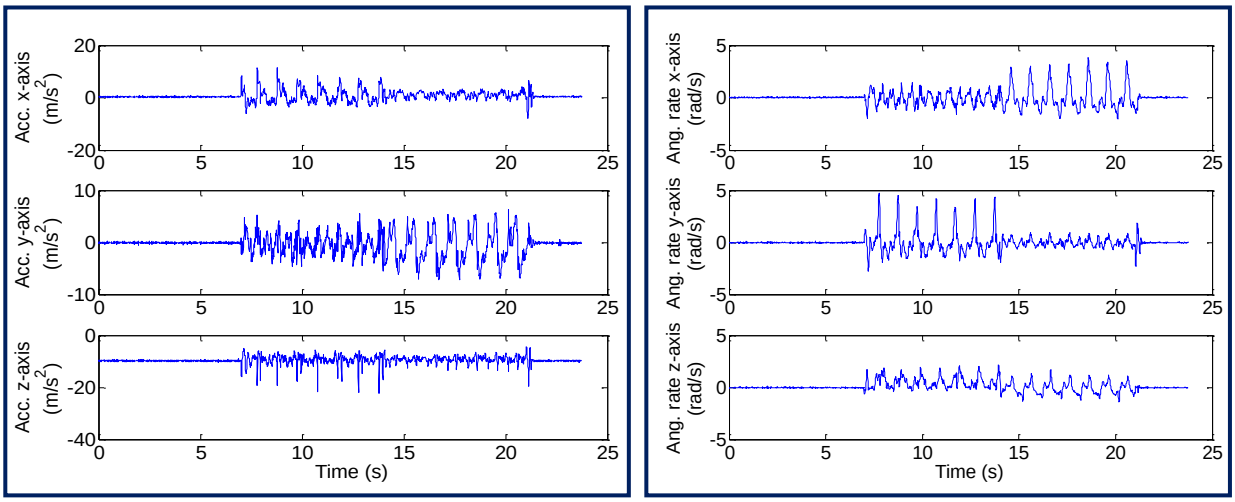


Figure 6-25 Accelerations and angular rates for combined gait measured by the MTi-OEM unit (Xsens)

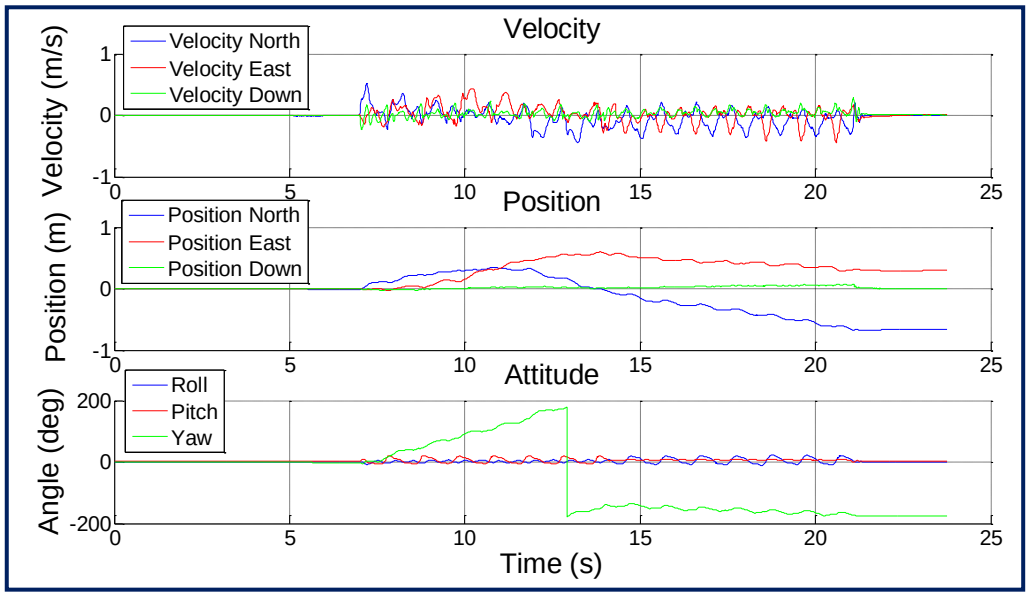


Figure 6-26 Velocity, position and attitude results regarding the SMC-INS applied on real data inertial data (MTi-OEN, Xsens) and SMC data measured for combined gait

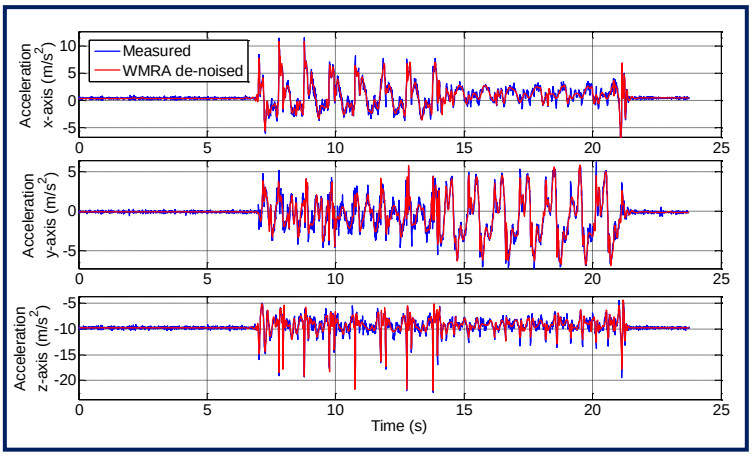


Figure 6-27 Measured and de-noised acceleration data (de-noising using WMRA LOD 4 with sym8 wavelet)

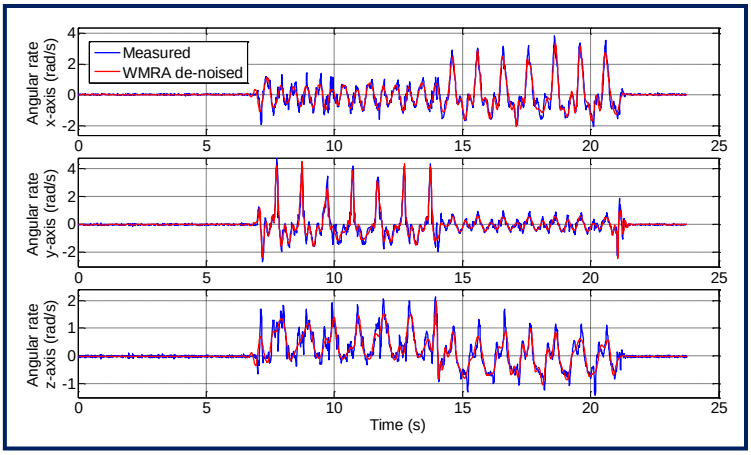


Figure 6-28 Measured and de-noised angular rates (de-noising using WMRA LOD 4 with sym8 wavelet)

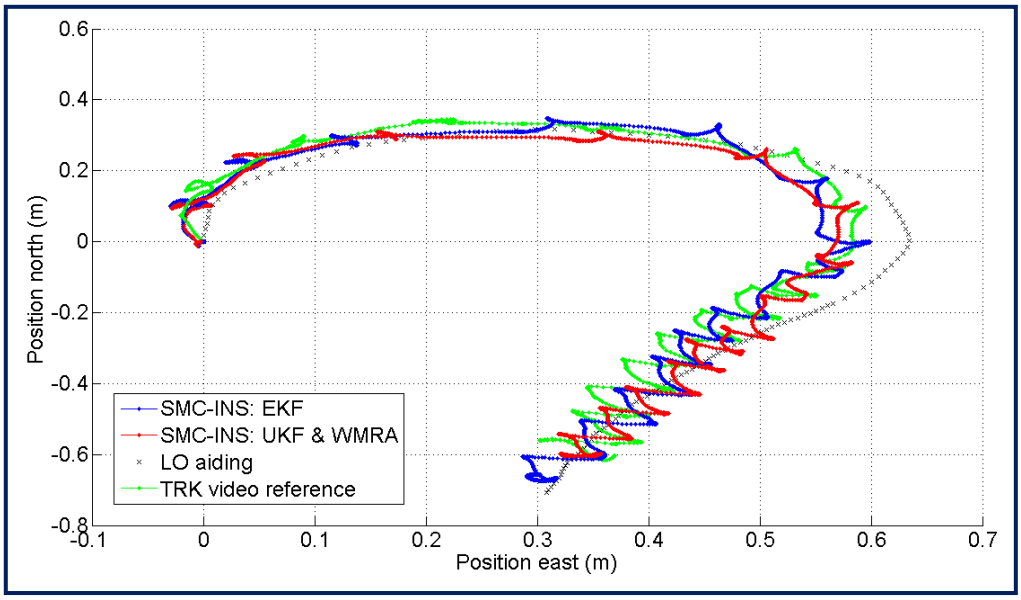


Figure 6-29 Trajectory comparison of the SMC-INS based on the EKF and the UKF with optimal WMRA data de-noising (includes SMC-INS trajectory estimated using EKF, SMC-INS trajectory estimated using UKF with WMRA data de-noising, LO legged odometer aiding signal, and TRK video reference)

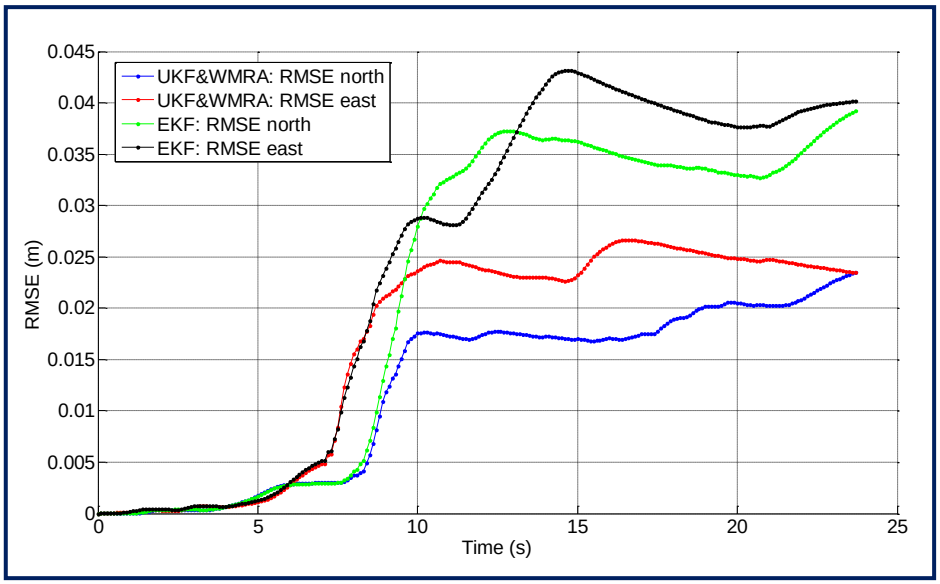


Figure 6-30 Comparison of precision development in time by means of RMSE between the trajectory estimated using the EKF (green and black) and the UKF with optimal WMRA de-noising (blue and red)

The results of the RMSE analysis in **Figure 6-30** clearly show and hence confirm the assumed superior performance of the UKF when enhanced by the proposed WMRA de-noising. The overall RMSE achieved on this combined gait was less than 0.03 m in both northern and eastern directions.

FIELD-TEST: ROBUSTNESS TO SLIPPAGE

In this chapter results regarding the robustness to slippage of the SMC-INS are presented. For the purpose of this field-test the testing ground was covered with a blue-foil in order to lower the friction. Since the methodology for legged odometer design included the friction as one of the indicators and training data sets on this blue-foil were provided for the legged odometer development, the SMC-INS was expected to cope with the low friction accordingly. Since slippage is a crucial issue that concerns all of the legged robots, decrease in precision was expected.

The same combination of two different gaits was used and analysed as in the previous chapter. The selected gait frequencies were not the ones included in the indicator training set during legged odometer development to prove that the SMC-INS can generalise for any gait frequency. The duration of the turning right gait was 30 seconds and the gait frequency was approximately 0.746 Hz. The duration of the turning left gait was 60 seconds with the gait frequency approximately 0.746 Hz as well. **Figure 6-31** shows the results regarding the UKF based SMC-INS performing on the blue-foil terrain with very low friction that caused serious slippage. **Figure 6-32** (top) presents the stride length data as estimated by the legged odometer and the stride length data computed from the reference. Attitude results are shown in **Figure 6-32** (bottom) to demonstrate the captured dynamics. **Figure 6-33** concludes the overall RMSE achieved. These results clearly indicate that the influence of the slippage caused the radius of the turning right gait to decrease; the turning left gait fails to provide turning effect at all. Although the precision in trajectory estimation deteriorated (the RMSE was approximately 0.12 m in both northern and eastern directions, which is still less than one body-length), the SMC-INS did correctly capture the dynamics regarding attitude and, what is more important, it did compensate for the slippage as desired.

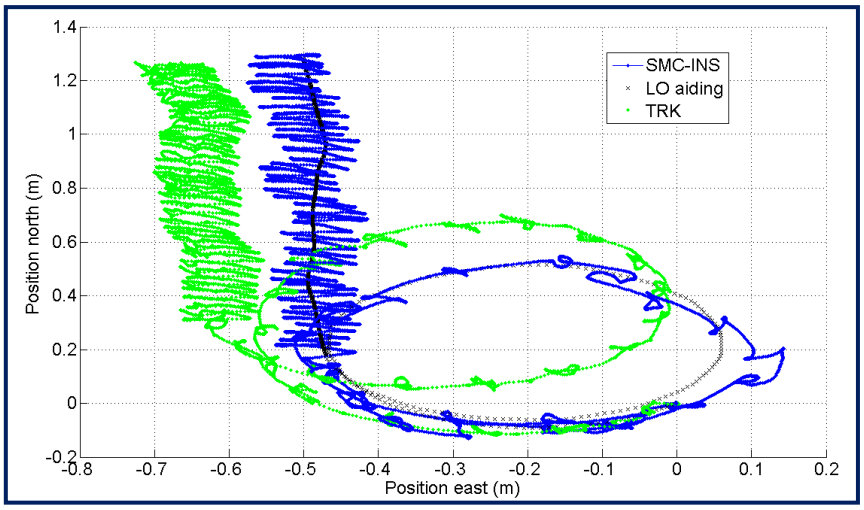


Figure 6-31 Trajectory results regarding the UKF based SMC-INS performing on terrain with very low friction that causes slippage (SMC-INS estimated trajectory, LO legged odometer aiding, and TRK reference)

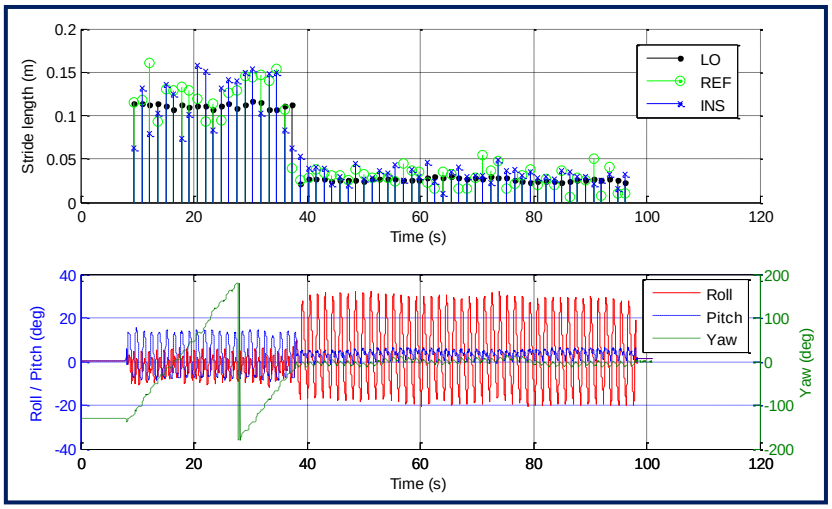


Figure 6-32 Stride length data (LO stride length estimated by legged odometer and provided as aiding, REF stride length reference, INS stride length computed after data fusion using the UKF based SMC-INS) (top). Attitude results regarding the UKF based SMC-INS performing on terrain with very low friction (bottom).

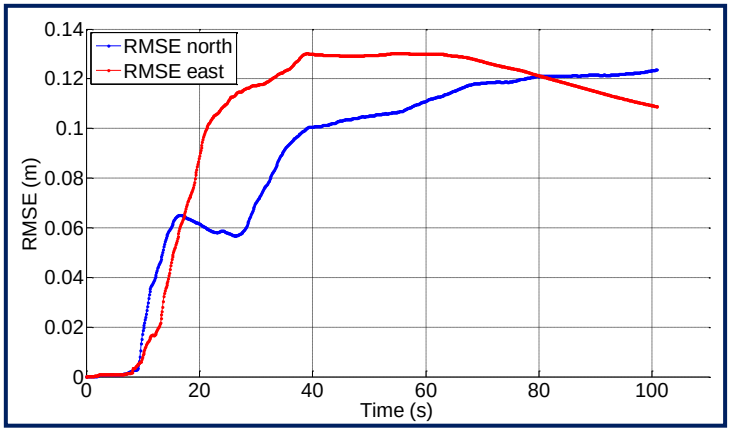


Figure 6-33 Precision of the estimated trajectory in time for the combined gait on terrain with low friction

7 CONCLUSIONS AND RECOMMENDATION

The main objective of this dissertation was to investigate the use of adaptive filtering methods, especially the EKF and the UKF algorithms, for inertial navigation. The final application concerned a highly dynamic quadruped robot. This investigation required a development of navigation algorithm which covered three main aspects: *the inertial sensors error modelling* (chapter 4.2), *inertial sensor output data de-noising* (chapter 4.3), and *Kalman filtering based data fusion* (chapters 4.4 and 4.5). This chapter gives a brief summary of the research achievements of this dissertation. Conclusions drawn from the achievements are presented together with recommendations for future research.

7.1 SUMMARY OF THE ACHIEVEMENTS

The aims of the dissertation (chapter 3) were fulfilled and the major contributions can be concluded in the following points:

- A fully autonomous self-motion cues aided inertial navigation system for a quadruped robot was designed and realized. The SMC-INS combines a complex INS error model (chapter 5.3) and data-driven SMC approach (chapter 5.4) to provide information regarding 3D orientation and position of a quadruped robot. The SMC-INS is independent of any external signal and its operation was verified by more than 100 navigation experiments performed on various surfaces and tested for gaits of various motion dynamics (chapter 6.3).
- A complete quaternion based real-time INS mechanization algorithm for raw inertial data processing was implemented; including sculling and coning effect compensation, and optimal initial alignment algorithm (chapter 4.4.4).
- A methodology for legged odometry for a quadruped robot based on its self-motion cues was proposed. A novel SMC mechanization algorithm based on correlation indicators was designed and implemented (chapter 5.4).
- An appropriate error model for the INS and the SMC mechanization data fusion was implemented (chapter 5.3) and these two algorithms were integrated using both the EKF and the UKF (5.6). The performance of both the EKF and the UKF versions of the SMC-INS were tested with respect to different noise models and different de-noising procedures for the attitude stability (chapter 6.3.1) as well as for the position precision (chapter 6.3.2). The EKF and the UKF were compared and evaluated by means of RMS error development in time for a combination of dynamic turning gaits (chapter 6.3.3); video tracking was used to obtain referential trajectory.
- Regarding the actual SMC-INS implementation, suboptimal modelling techniques (chapter 4.7) based on the covariance analysis (chapter 5.6.1) and observability analysis (chapter 5.6.2) were used to tune the error model. This suboptimal error model was further enhanced by implementing nonholonomic constraints to assure the desired observability as well as to prevent filter divergence.

7.2 CONCLUSIONS

Regarding the **inertial sensors modelling and analysis** it was proven that the precise determination of the noise model plays a crucial role and affects the stability of the whole navigation algorithm. The white noise model proved to be the most robust. For the short term navigation experiments it sufficiently substituted the random walk or exponentially correlated noise models. The AR noise models proved to be of comparable quality and in some cases more robust. However, the computational load for AR noise models of order more than 10 was unbearable for the SMC-INS implementation. The UKF for higher order AR noise models became repeatedly unstable.

Regarding the **de-noising procedures** the best results were obtained for the optimal WMRA inertial data de-noising. However, due to the high computational load requirements of the SMC-INS, the WMRA application was limited to post-processing only. If the WMRA procedure is combined together with RTS smoother, the best estimated trajectories can be obtained.

Regarding the **evaluation of adaptive filtering methods for inertial navigation** a number of conclusions can be made in accordance with the current state of the art (chapter 2.2):

- The main difference between the EKF and the UKF is in the way they deal with non-linearities especially for large attitude errors. The UKF in this case cannot deal with complete uncertainties.
- Concerning the EKF, derivation of Jacobian matrices for both system and observation equations can be complicated for complex models and limits the actual implementation. However, it was proven that the analytical complexity can be partially resolved by suboptimal modelling techniques.
- Solution based on the SEKF, i.e. computation of both Jacobian and Hessian matrices, was excluded from the final comparisons since the overall analytical complexity did not bring any significant improvement.
- For land vehicles and robots the UKF is preferable than the EKF since heading error can grow large without causing the filter to diverge. However, for the proposed SMC-INS both the EKF and the UKF proved to be of comparable performance.
- The EKF performance is strongly given by the error model used, which depends on the actual sensors type, accuracy, noise characteristics, navigation task, and platform type.
- The UKF performance is invariant to such sensitivity to the error model choice - only general system process and measurement models need to be defined. Hence, system model development stage is simplified. Unfortunately, this requires every sigma point of the UKF to pass through the INS mechanization algorithm, causing great computation load, which is in case of the SMC-INS unbearable for real time processing. However, UKF can still be applied by using the same nonlinear suboptimal error model as for the EKF, ensuring similar performance.

Finally, it can be concluded that the developed **self-motion cues aided inertial navigation system** is a vital tool for evaluation of various adaptive filtering methods. The chosen implementation proved to be robust to slippage, stable in attitude, and sufficiently precise with RMSE less than one body-length for trajectory estimation. The independence of the SMC-INS of any external signal is invaluable property for indoor navigation. Proposed methodology for SMC-INS design is easily adaptable to different SMC sensor combinations and in general it can be exploited to

almost any legged robot. The modular structure of the SMC-INS also enables further enhancement by additional aiding sensors such as GPS, magnetometer or vision sensor. When compared to the current state of the art, especially the [52], [53], the proposed SMC-INS extends the above-mentioned concept in many important aspects:

1. Velocity aiding rather than attitude is utilized. Velocity can be obtained as a product of frequency of locomotion (given by the motor signal) and stride length (which can be obtained from sensors on legs).
2. The algorithm for the legged odometer is data-driven, or empirical, rather than analytical.
3. The legged odometer is based on a fusion of multiple sensors – hip and knee angular sensors, and feet pressure sensors.
4. The proposed architecture of the SMC-INS can deal with different velocities and different gaits, including their transitions, and with substantial slippage as well.
5. The SMC-INS concept enables full pose estimation (velocity, position and attitude) in three dimensions.

7.3 ISSUES FOR FURTHER RESEARCH

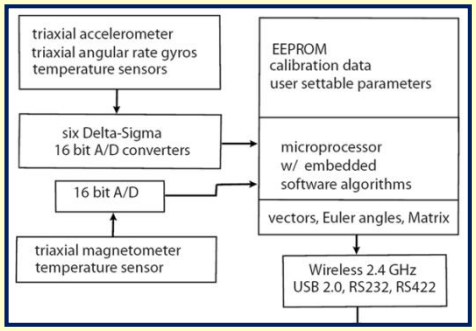
Finally, following steps can be taken to extend the work presented in this dissertation:

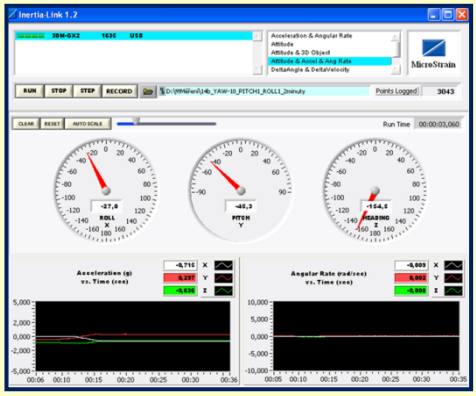
1. The sensor suit can further be enhanced by additional self-motion cues sensors to improve the precision in heading determination, especially for long-term experiments. This can be done exploiting the same methodology as proposed.
2. To evaluate the SMC-INS for outdoor navigation, GPS should be included to provide absolute position reference to ensure best performance. Furthermore, for large scale outdoor experiments the power source should be embedded on the platform; this however could negatively influence the dynamics.
3. Only linear relationships (correlations) between SMC sensory data, the stride length and the delta heading were investigated so far. The method can be extended to nonlinear relationships, using for example the mutual information.
4. Although the SMC-INS was evaluated by means of post-processing the algorithm was proposed and designed to run in real time. Implementation into the embedded hardware would be an interesting challenge.

8 APPENDICES

8.1 3DM-GX2 ATTITUDE AND HEADING REFERENCE SYSTEM

One of the multi-sensor units used for raw inertial data collection was the Attitude and Heading Reference System developed and manufactured by MicroStrain, Inc.; for details see **Figure 8-1**:





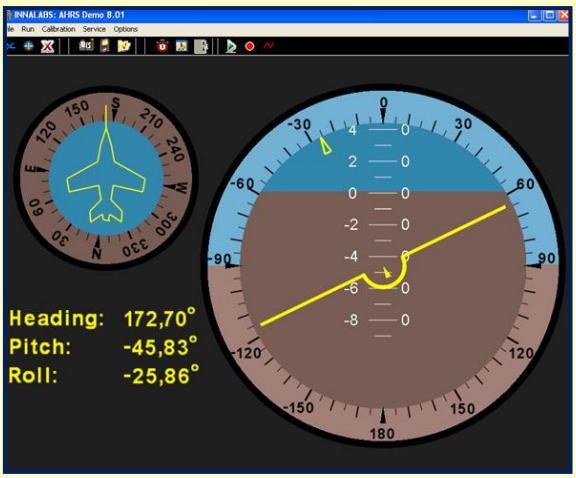
Orientation range (pitch, roll, yaw)	360° about all axes
Accelerometer range	accelerometers: ± 5 *g* standard ± 10 *g* and ± 2 *g* also available
Accelerometer bias stability	± 0.010 *g* for ± 10 *g* range ± 0.005 *g* for ± 5 *g* range ± 0.003 *g* for ± 2 *g* range
Accelerometer nonlinearity	0.2%
Gyro range	gyros: ± 300°/sec standard, ± 1200°/sec, ± 600°/sec, ± 150°/sec, ± 75°/sec also available
Gyro bias stability	± 0.2°/sec for ± 300°/sec
Gyro nonlinearity	0.2%
Magnetometer range	± 1.2 Gauss
Magnetometer nonlinearity	0.4%
Magnetometer bias stability	0.01 Gauss
A/D resolution	16 bits
Orientation Accuracy	± 0.5° typical for static test conditions ± 2.0° typical for dynamic (cyclic) test conditions & for arbitrary orientation angles
Orientation resolution	<0.1° minimum
Repeatability	0.20°
Output modes	acceleration and angular rate, deltaAngle and deltaVelocity, Euler angles, rotation matrix
Interface options	RS232, RS422, USB 2.0 and wireless - 2.45 GHz IEEE 802.15.4 direct sequence spread spectrum, license free worldwide (2.450 to 2.490 GHz) - 16 channels
Wireless communication range	70 m
Digital output rates	1 to 250 Hz with USB interface 1 to 100 Hz with wireless interface
Serial data rate	115200 bps
Supply voltage	5.2 to 9.0 volts
Power consumption	90 mA
Connectors	micro DB9
Operating temp.	-40 to +70°C with enclosure -40 to +85°C without enclosure
Dimensions	41 mm x 63 mm x 32 mm with enclosure 32 mm x 36 mm x 24 mm without enclosure
Weight	39 grams with enclosure, 16 grams without enclosure
Shock limit	1000 *g* (unpowered), 500*g* (powered)

Figure 8-1 3DM-GX2 Attitude and Heading Reference System (MicroStrain)

8.2 AHRS M3 ATTITUDE AND HEADING REFERENCE SYSTEM

One of the multi-sensor units used for raw inertial data collection was the Attitude and Heading Reference System developed and manufactured by Innalabs, Inc.; for details see **Figure 8-2**:

Parameter	Unit	Value
Update Rate	Hz	1.....100 (user settable)
Start-up Time	sec	< 1
Full Accuracy Data (Warm-up Time)	sec	< 60
Heading		
Range	deg	0 to 360
Static Accuracy at Normal Conditions (Tc=25°C)	deg RMS	0.5 (1)
Static Accuracy in Temperature Range (Tc=-40°C to +70°C)	deg RMS	0.8 (1)
Dynamic Accuracy	deg RMS	1.0 (2)
Noise (at 100Hz output)	deg RMS	0.03
Resolution	deg	0.01
Attitude		
Range: Pitch, Roll	deg	±90, ±180
Static Accuracy at Normal Conditions (Tc=25°C)	deg RMS	0.04
Static Accuracy in Temperature Range (Tc=-40°C to +70°C)	deg RMS	0.1
Dynamic Accuracy	deg RMS	0.4 (2)
Noise (at 100Hz output)	deg RMS	0.02
Resolution	deg	0.01
Angular Rate		
Input Range: Yaw, Pitch, Roll	deg/sec	±300
Bias	deg/sec RMS	0.02
Scale Factor Accuracy	%	0.2
Non-Linearity	% FS	0.1
Random Walk	deg/sqrt(hr)	6
Resolution	deg/sec	0.01
Bandwidth	Hz	40
Linear acceleration		
Input Range: X/Y/Z	g	±2
Bias	mg RMS	0.3
Scale Factor Accuracy	%	< 0.1
Non-Linearity	% FS	0.2
Random Walk	m/s/sqrt(hour)	0.06
Resolution	mg	< 10
Bandwidth	Hz	40
Environment		
Operating Temperature	deg C	–40 to +70
Non-Operating Temperature	deg C	–55 to +85
Electrical		
Input Voltage	VDC	+5.5 to +6.5
Input Current	mA	< 110
Power Consumption	W	0.66 (at 6V powering)
Digital Output Format		RS-232
Physical		
Size	mm	109 x 31 x 29 (case) 127 x 31 x 29 (with mounting lugs and connector)
Weight	kg	0.19 / 0.16 (3)
Connector		5-Pin Binder 719 (Male)



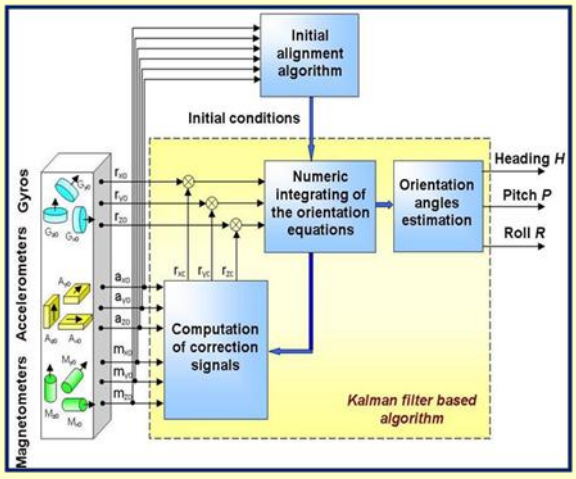
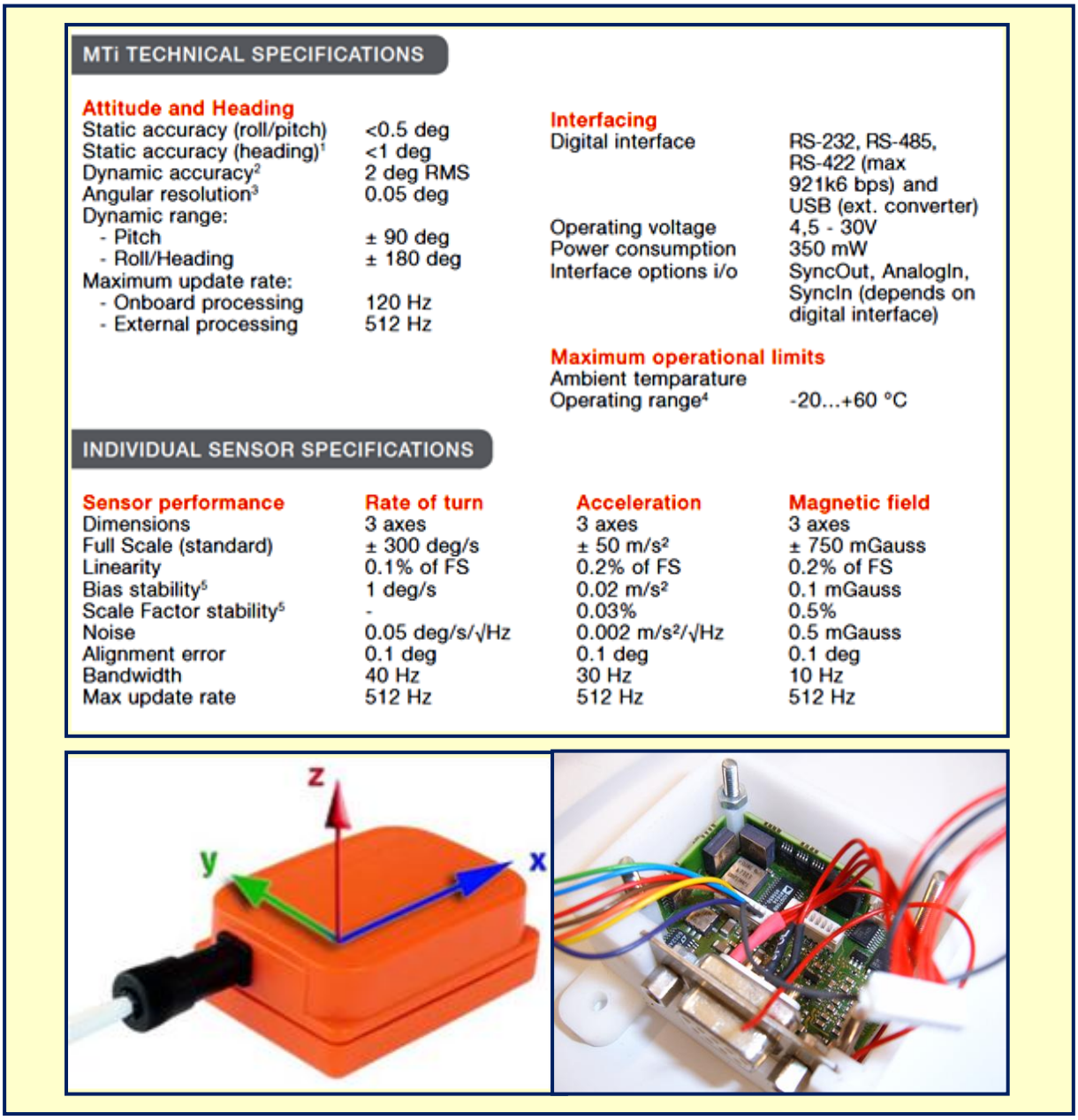


Figure 8-2 AHRS M3 Attitude and Heading Reference System (Innalabs)

8.3 MTI-OEM AND MT9 XSENS ATTITUDE AND HEADING REFERENCE SYSTEMS

Two Attitude and Heading Reference Systems developed and manufactured by Xsens: the MT9 and its newer version the MTi-OEM. Both the MT9 and the MTi-OEM were used for field-tests and evaluation of the proposed navigation algorithms; for details see **Figure 8-3**:



MTi TECHNICAL SPECIFICATIONS

Attitude and Heading

Static accuracy (roll/pitch)	<0.5 deg
Static accuracy (heading)[1]	<1 deg
Dynamic accuracy[2]	2 deg RMS
Angular resolution[3]	0.05 deg
Dynamic range:	
- Pitch	± 90 deg
- Roll/Heading	± 180 deg
Maximum update rate:	
- Onboard processing	120 Hz
- External processing	512 Hz

Interfacing

Digital interface	RS-232, RS-485, RS-422 (max 921k6 bps) and USB (ext. converter)
Operating voltage	4,5 - 30V
Power consumption	350 mW
Interface options i/o	SyncOut, AnalogIn, SyncIn (depends on digital interface)

Maximum operational limits

Ambient temparature	
Operating range[4]	-20...+60 °C

INDIVIDUAL SENSOR SPECIFICATIONS

Sensor performance	Rate of turn	Acceleration	Magnetic field
Dimensions	3 axes	3 axes	3 axes
Full Scale (standard)	± 300 deg/s	± 50 m/s²	± 750 mGauss
Linearity	0.1% of FS	0.2% of FS	0.2% of FS
Bias stability[5]	1 deg/s	0.02 m/s²	0.1 mGauss
Scale Factor stability[5]	-	0.03%	0.5%
Noise	0.05 deg/s/√Hz	0.002 m/s²/√Hz	0.5 mGauss
Alignment error	0.1 deg	0.1 deg	0.1 deg
Bandwidth	40 Hz	30 Hz	10 Hz
Max update rate	512 Hz	512 Hz	512 Hz

Figure 8-3 MTi-OEM and MT9 Xsens Attitude and Heading Reference Systems (Xsens)

8.4 SYSTEM FOR POSITIONING TESTS: ROTATIONAL TILT PLATFORM

To measure attitude angles, gravitational accelerations and angular rates with respect to a precise reference a measurement setup that provided such reference was required. This setup consisted of a rotational tilt platform, communication unit and applications for PC ensuring automatic control and data acquisition. The hearth of this measurement system was the rotational tilt platform as shown in **Figure 8-4** and the complete measurement setup was realized according to the block scheme shown in **Figure 8-5**.This platform enabled both the mechanical and the electrical control realized by either specifying final angular positions or in differential way by specifying of incremental angular position i.e. speeds of rotation. Attitude control given by the number of rotations was supported as well. This platform required voltage supply of 8V (0.8 A) for the electronics and 24V (5A) for the motors in each axes of rotation. Communication with the platform was realised using RS232 serial interface based on a communication protocol specified by the manufacturer.

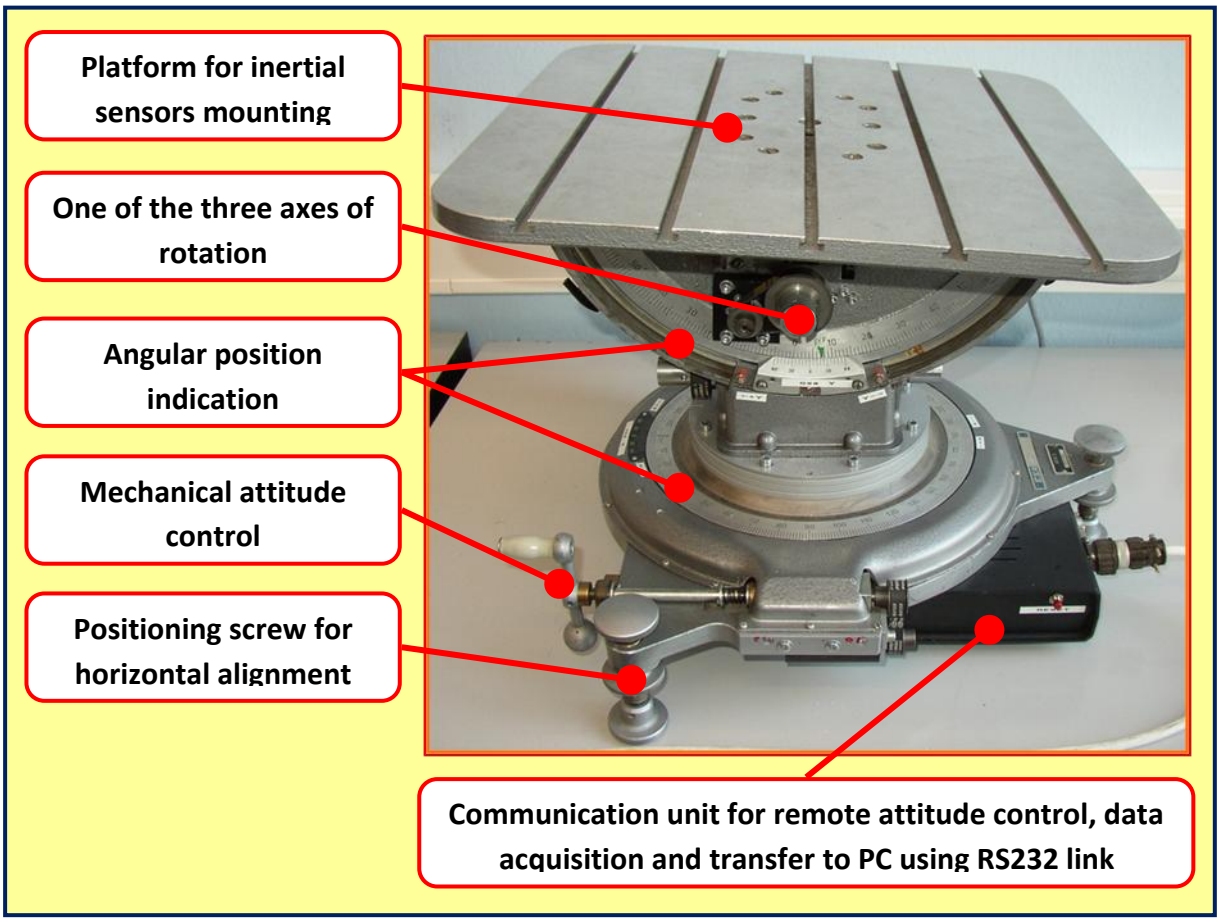


Figure 8-4 Rotational tilt platform for inertial sensors testing and calibration

For the purpose of performing complex measurement procedures a control application for PC was developed in LabWindows/CVI by P. Ficek as part of his diploma thesis [99] at CTU, FEL under the author's supervision. The application enabled various combinations of attitude control to be defined. Data acquisition from the incremental optical sensors embedded in each axis of rotation

was performed with variable sampling period from 0.04s to 10s. Acquired data were plotted and stored for further analysis.

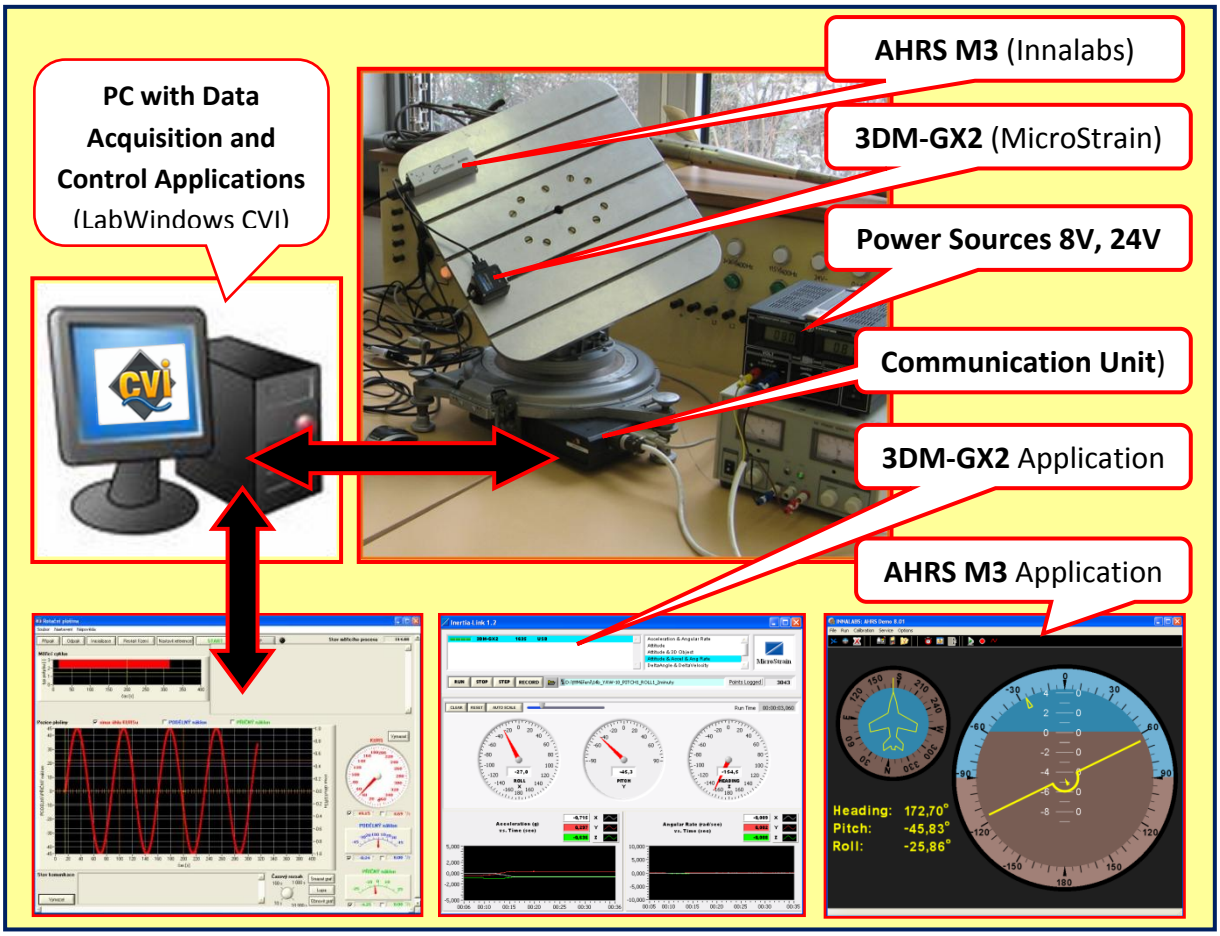


Figure 8-5 Measuring setup for inertial sensors testing

The technical parameters of the rotational tilt platform, such as the range of measurable attitude angles, the speed of rotation and the angle resolution by the optical incremental sensors are given in **Table 8-1**:

Turning axis	Range	Speed of Rotation	Resolution in Angle
X	0° - 360°	±310°/s	2.6“
Y	±50°	±42°/s	1.15“
Z	±30°	±60°/s	2.3“

Table 8-1 Parameters of the rotational tilt platform [99 p. 8]

8.5 MATLAB CODES

This chapter presents some of the MATLAB codes used. The complete SMC-INS code is not included.

Matlab 8-1 Van Loan discretization method 123
Matlab 8-2 Algorithm for noise generation according to different random processes 123
Matlab 8-3 Euler - quaternion conversions 124
Matlab 8-4 Static initial alignment algorithm 124
Matlab 8-5 Discrete Kalman filter algorithm 124
Matlab 8-6 Example of the structure of the Extended Kalman filter algorithm 125
Matlab 8-7 Example of the structure the Unscented Kalman filter algorithm 126
Matlab 8-8 RTS Two-pass smoother algorithm for Kalman filter 127
Matlab 8-9 Computation of the normal gravity value according to WGS84 127
Matlab 8-10 Implementation of the Kalman filter feedback to strapdown mechanization 128
Matlab 8-11 Simplified approach to computation of the Allan deviation 128

% Van Loan Discretization Method

```
function [A,Qk] = discretize(F,L,Qt,dt)
% Original system:      dx/dt = F x + L w,  w ~ N(0,Qt)
% Discrete system:      x[k] = A x[k-1] + q, q ~ N(0,Q)
% where  dt = Time step
A = expm(F*dt);
n   = size(F,1);
Phi = [F L*Qt*L'; zeros(n,n) -F'];
AB  = expm(Phi*dt)*[zeros(n,n);eye(n)];
Qk   = AB(1:n,:)/AB((n+1):(2*n),:);
```

Matlab 8-1 Van Loan discretization method

% Noise generation according to different random processes

```
noise0(1) = randn;        % unit white gaussian noise
noise1(1) = 128;          % random walk (Wiener process)
noise2(1) = randn;        % exponentially correlated
damping2  = exp(-1/100); % damping factor for exponentially correlated noise
sigma2    = sqrt(1 - damping2^2);
noise3(1) = randn;        % harmonic noise
damping3  = exp(-1/100); % damping factor for harmonic noise
sigma3    = sqrt(1 - damping3^2);
n4        = randn;        % quadrature of noise
delta     = 2*pi/32;      % harmonic noise angular increments
c = cos(delta);
s = sin(delta);
counter = 4098;
fs = 64;
for k=2: counter
   noise0(k) = randn;
   noise1(k) = noise1(k-1) + randn;
   noise2(k) = noise2(k-1)*damping2 + sigma2*randn;
   noise3k   = (c*noise3(k-1) + s*n4)*damping3 + sigma3*randn;
   n4        = (-s*noise3(k-1) + c*n4)*damping3 + sigma3*randn;
   noise3(k) = noise3k;
end;
```

Matlab 8-2 Algorithm for noise generation according to different random processes

% Euler - Quaternion Conversion Implementations

```
function q = euler2quat( angles )
% EULER2QUAT conversion of Euler angles to quaternion.
cang = cos( angles/2 );
sang = sin( angles/2 );
q = [ cang(:,1).*cang(:,2).*cang(:,3) + sang(:,1).*sang(:,2).*sang(:,3),
      sang(:,1).*cang(:,2).*cang(:,3) - cang(:,1).*sang(:,2).*sang(:,3),
      cang(:,1).*sang(:,2).*cang(:,3) + sang(:,1).*cang(:,2).*sang(:,3),
      cang(:,1).*cang(:,2).*sang(:,3) - sang(:,1).*sang(:,2).*cang(:,3)];
function angles = quat2euler( q )
% QUAT2EULER conversion of quaternion to Euler angles.
qin   = quatnormalize( q );
roll  = atan2(2.*(qin(:,3).*qin(:,4) + qin(:,1).*qin(:,2)), ...
                qin(:,1).^2 - qin(:,2).^2 - qin(:,3).^2 + qin(:,4).^2);
pitch= asin(-2.*(qin(:,2).*qin(:,4) - qin(:,1).*qin(:,3)));
yaw   = atan2(2.*(qin(:,2).*qin(:,3) + qin(:,1).*qin(:,4)), ...
                qin(:,1).^2 + qin(:,2).^2 - qin(:,3).^2 - qin(:,4).^2);
angles = [roll pitch yaw];
```

Matlab 8-3 Euler - quaternion conversions

% Initial Alignment Algorithm

```
function [ roll pitch g] = alignment_static( ACCx, ACCy, ACCz)
% Computation of the initial alignment angles (roll & pitch) from the
% static acceleration measurements. MATLAB symbolic solver is used,
% reference gravity value is computed.
% INPUT:    acceleration vectors in x, y, z axes (m/s^2)
% OUTPUT:   roll & pitch angles (deg), local gravity value (m/s^2)
syms r p;
S = solve(  sprintf('g*sin(p)= %d', ACCx),...
            sprintf('-g*cos(p)*sin(r)= %d', ACCy),...
            sprintf('-g*cos(p)*cos(r)= %d', ACCz));
roll = rad2deg(double(S.r)); pitch = rad2deg(double(S.p)); g = double(S.g);
end
```

Matlab 8-4 Static initial alignment algorithm

% Discrete Kalman Filter Algorithm

```
I = eye(length(P))
% TIME UPDATE   (k-1)-->(k):
Pminus = Phi*Pplus*Phi'+Q;
Xminus = Phi*Xplus;
% OBSERVATION UPDATE    (k):
PHt = Pminus*H';
K = PHt/(H*PHt+R);
Pplus = (I - K*H)*Pminus;
Xplus = Xminus+K*(z-H*Xminus);
```

Matlab 8-5 Discrete Kalman filter algorithm

% Extended Kalman Filter Algorithm:

```
%% EKF INITIALIZATION:
% x is given by a nonlinear system function f -> F
% z is given by a nonlinear measurement function h -> H
x = ;       % Initial values of the state vector
x_ = ;      % Initial state vector estimate
sigmav = 0.1; Q = sigmav*sigmav;  % Process error covariance
sigmaw = 0.5; R = sigmaw*sigmaw;  % Measurement error covariance
P = ;       % Initial error covariance matrix values
I = eye(length(P)); % Unit matrix of appropriate dimension
F = ;       % F is the Jacobian of the system transfer functions due to each state
(involved variables)
G = ;       % G is the Jacobian of the plant transfer functions due to the error.
H = ;       % H is the Jacobian of the sensor transfer functions due to the variables
involved
W = ;       % W is the Jacobian of the sensor transfer functions due to the error.
% !NOTE: For discrete EKF use for eg Van Loan discretization method on F, H, G and W
N = ;       % Number of EKF filter steps
%% EKF ALGORITHM
for i = 2: N         % start at time=2
  % REAL SYSTEM:
  % Use measured data in vector z or generate using real system simulation
  x() = ;
  z() = ;
  % PREDICTION:
  % Predict x_ and z_ using nonlinear system function f and nonlinear measurement
function h
  % 1. Extrapolate first to obtain system state prediction x_
  % 2. Use x_ estimate and h to obtain estimated measurement vector z_
  x_() = ;
  z_() = ;
  % JACOBIAN UPDATE:
  % Update system matrix F Jacobian with newely predicted alues of x_
  F = ;
  G = ;
  % COVARIANCE TIME UPDATE:
  Pminus = F*P*F' + G*Q*G';
  % OBSERVATION UPDATE:
  PHt = Pminus*H';              % Implementation boost
  K = PHt*inv(H*PHt+R);         % Kalman gain
  P = (I-K*H)*Pminus;           % A posteriori covariance matrix estimation
  P = 0.5*(P+P');               % Implementation boost (symmetry of P)
  x_() = x_() + K*(z()-z_());   % System state observation update
  % DATA STORE: Here store desired system states and covariances to be plot
  sigmaP() = sqrt(diag(P));     % Store std for one sigma boundary plot
end
```

Matlab 8-6 Example of the structure of the Extended Kalman filter algorithm (requires definitions of the used variables for execution)

% Unscented Kalman Filter Algorithm:

```
alpha = 1e-3; % UKF parameter - spread of sigma points around mean state
beta = 2; % UKF parameter - prior knowledge of distribution (if x gaussian, beta = 2)
ki = 0;   % UKF parameter - scaling parameter (if x gaussian, ki = 3 - N)
L1 = alpha^2*(N+ki)-N;                      % Scaling factor L1
L2 = N+L1;                                  % Scaling factor L2
Wm = [L1/L2 0.5/L2+zeros(1,2*N)];           % Weights for means
Wc = Wm;  Wc(1) = L1/L2+(1-alpha^2+beta);   % Weights for covariances
% FILTER INITIALIZATION:
x = xo_a;          % Augmented state vector reshaping
P = Po_a;           % Covariance matrix
x_ = 0;             % Predicted state vector initialization
P_ = 0;             % Predicted covariance matrix initialization
z_ = 0;             % Predicted measurement vector initialization
Pxz = 0;            % State vector - measurements covariance matrix initialization
Pzz = 0;            % Innovations covariance matrix initialization
%% UKF CYCLE
for i = 1:steps
    % SIGMA POINT GENERATIION:
    base = x(:,ones(1,numel(x)));    % Sigma point base
    mod = sqrt(L2)*cholesky(P)';     % Sigma point modifier (Cholesky decomp. to obtain
"sqrt(P)")
    Xsp = [x base+mod base-mod];     % Vector of sigma points
    Nsp = size(Xsp,2);               % Number of sigma points
    % TIME UPDATE:
    % 1. Transformation of the SPs through nonlinear system function
    % 2. Computation of the mean of the transformed SPs
    % 3. Computation of the covariance of the transformed SPs
    for k = 1:Nsp
        XUT(:,k) = f(Xsp(:,k));      % 1.
        x_ = x_+Wm(k)*XUT(:,k);      % 2.
    end
    P1 = XUT-x_(:,ones(1,Nsp));
    P_ = P1*diag(Wc)*P1';            % 3.
    % MEASUREMENT UPDATE:
    % 1. Sigma points are transformed through measurement model function
    % 2. Comutation of the predicted measurements from the transformed SPs
    % 3. Computation of the covarinace of the innovation sequence
    % 4. Computation of the covariance betweenn the states and the measurements
    for k = 1:Nsp
        ZUT(:,k) = h(XUT(:,k));      % 1.
        z_ = z_+Wm(k)*ZUT(:,k);      % 2.
    end
    Z2 = ZUT-z_(:,ones(1,Nsp));
    P2 = Z2*diag(Wc)*Z2';            % 3.
    P12 = P1*diag(Wc)*Z2';           % 4.
    % 5. Kalman Filter UPDATE EQUATIONS:
    K = P12*inv(P2 + R);         % Kalman gain computation
    x = x_ + K*(ZDAT(i) - z_);   % Innovation (Measured data input)
    P = (P_+ Q) - K*P2*K';           % DATA SAVING:
    XDAT(:,i)= x;                % System state estimates
    PDAT(:,i)= sqrt(diag(P));    % Standard deviation
end
```

Matlab 8-7 Example of the structure the Unscented Kalman filter algorithm
(requires definitions of the used variables for execution)

% RTS Two-pass Smoother Algorithm for Kalman Filter

```
% PERFORMANCE EVALUATION of the Rauch-Tung-Striebel smoother using random
% walk estimation (RTS algorithm based on Grewal & Andrews 2001, p. 162)
N = 500;  % Number of samples of process used in simulations
Q = .01;                            % Variance of random walk increments
R = .5;                             % Variance of sampling noise
sigw  = sqrt(Q); sigv  = sqrt(R);   % Standard deviations
Pp      = 80;                       % Covariance value of the initial uncertainty
x(1)    = sqrt(Pp)*randn;           % Initial value of true process
xp(1)   = 0;                        % Initial (predicted) value of true process
RMS   = sqrt(Pp);                   % for RMS estimation uncertainty plot
ts         = 0;                     % Time vector
%% KALMAN FILTER: FORWARD PASS
for k = 1:N;
   t(k)     = k-1;
   if k~=1
      x(k)  = x(k-1) + sigw*randn;  % Random walk process
      Pp(k) = Pc(k-1) + Q;
      RMS = [RMS,sqrt(Pp(k))];
      ts       = [ts,t(k)];
      xp(k) = xc(k-1);
   end;
   z(k)     = x(k) + sigv*randn;    % Noisy measurement sample generation
   K        = Pp(k)/(Pp(k)+R);      % Kalman gain computation
   xc(k) = xp(k) + K*(z(k) - xp(k));% Correction of the Kalman filter estimate
   Pc(k) = Pp(k) - K*Pp(k);
   RMS = [RMS,sqrt(Pc(k))]; ts = [ts,t(k)];
end;
%% SMOOTHER FOR KALMAN FILTER: BACKWARD PASS
xs = xc; % Initialization of the smoothed value using corrected value at time N
for k = N-1:-1:1,
   A          = Pc(k)/Pp(k+1);     % Smoothed Kalman gain computation
   xs(k) = xs(k) + A*(xs(k+1) - xp(k+1));
end;
```

Matlab 8-8 RTS Two-pass smoother algorithm for Kalman filter

% Computation of the Normal Gravity Value According to WGS84

```
function gravity = comp_gravity(LLA)
% Computation of gravitaty in Earth's frame based on WGS84 Gravity model
% Precision of the computation is 10^-6 up to the 20000m of height above surface
% INPUT:          LLA = [latitude (deg); longitude (deg); height (m)];
% OUTPUT:         gravity vector (m/s^2)
lat = deg2rad(LLA(1)); alt = LLA(3);
% WGS84 Gravity model constants:
a1 = 9.7803267715; a2 = 0.0052790414; a3 = 0.0000232718; a4 = -0.0000030876910891;
a5 = 0.0000000043977311; a6 = 0.0000000000007211;
% Gravity computation
gnz = a1*(1+a2*sin(lat)^2+a3*sin(lat)^4)+(a4+a5*sin(lat)^2)*alt+a6*alt;
gravity = [0 0 gnz]';
```

Matlab 8-9 Computation of the normal gravity value according to WGS84

% Kalman Filter with Feedback to Strapdown Mechanization

```
% NOTE: "i" is the time step given by the sampling period of the inertial signals
% TIME UPDATE:  from (k-1) to (k):
count = count + 1;
Pminus = Phi*Pplus*Phi'+Qk;
Mminus = Phi*Mplus;
% OBSERVATION UPDATE: at (k):
PHt = Pminus*H';
K = PHt/(H*PHt+Rk);
% If aiding data unavailable set K = 0 for the current KF step
if LLA_rdat(:,i)== zeros(3,1);
    K = zeros(states,6);
end
Pplus = (I - K*H)*Pminus;  % Standard covariance matrix update
Pplus = 0.5*(Pplus+Pplus');% Symmetrization of P
Mplus = Mminus+K*(ZZ(:,i)-H*Mminus);
MM(:,i) = Mplus;           % Storing of mean values
PP(:,:,i) = Pplus;         % Storing of covariance values
% MECHANIZATION FEEDBACK:
if count == kf2mech_update % Condition for feedback to mechanization
    count = 0;             % Counter reset
    LLA(:,i) = LLA(:,i) - MM(1:3,i);            % Position correction
    VN(:,i) = VN(:,i) - MM(4:6,i);              % Velocity correction
    C_b2n(:,:,i) = (eye(3) + comp_skew(MM(7:9,i)))*C_b2n(:,:,i);
    Q_b2n(:,i) = dcm2quat(C_b2n(:,:,i)')';% Attitude correction - quaternion update
    Mplus = zeros(states,1);                    % Error state vector is reset
end
```

Matlab 8-10 Implementation of the Kalman filter feedback to strapdown mechanization

% Allan Deviation Computation Algorithm

```
function RESULTS = allanvar(signal,Ts)
% INPUTS:  signal = input data signal, Ts = sampling period (s)
% OUTPUTS: tau = measurement time (s), sig =  N samples, adev = Normal Allan
n = length(signal);
count = floor( log((n-1)/3)/log(2) );           % Max. tau computation
tau = zeros(1,count+1); sig = tau; adev =tau;   % Memory allocation
    for j=0:count
        m=2^j;
        tau(j+1)=m*Ts;
        D=zeros(1,n-m+1);
        for i=1:n-m+1
            D(i)=sum(signal(i:i+m-1))/m;
        end
        sig(j+1)=std(D(1:m:n-m+1)); fprintf('.');
        adev(j+1)=sqrt(0.5*mean((diff(D(1:m:n-m+1)).^2)));fprintf('.');
    end
RESULTS = [tau', sig', adev'];
end
```

Matlab 8-11 Simplified approach to computation of the Allan deviation

List of Tables

Table 2-1 Comparison of AI-based approach (ANFIS) to the conventional KF [34 p. 190] 6
Table 4-1 Shaping filter equations with σ being standard deviation, δ Kronecker delta, $1/\alpha$ correlation time, $\boldsymbol{P}$ covariance matrix and k discrete time step; definitions according to [26 p. 80] and [39 p. 20] 17
Table 4-2 Summary of the root AVAR definitions for different errors sources 24
Table 4-3 Different initial alignment procedures for INS 39
Table 4-4 Centralized and decentralized INS/GPS integration schemes [34 p. 71] 45
Table 4-5 Characteristics of INS, GPS and integrated INS/GPS systems [35 pp. 27-2] 46
Table 4-6 Alternative indexing schemes for memory saving purposes [26 p. 323] 60
Table 5-1 SIMALAN optimal gait characteristics 78
Table 5-2 Overview of the indicators for SIMALAN legged odometer; green = positive linear correlation, red = negative linear correlation, grey = no significant correlation 84
Table 5-3 Observability analysis of the proposed SMC-INS error model 91
Table 6-1 Allan Variance analysis of the 3DM-GX2 unit 93
Table 6-2 Allan Variance analysis of the AHRS M3 unit 93
Table 6-3 Allan Variance analysis of the MT9 Xsens unit 94
Table 6-4 Allan Variance analysis of the MTi-OEM Xsens unit 94
Table 6-5 Proposed optimal parameter setup for the WMRA de-noising algorithm 96
Table 6-6 Evaluation of attitude stability under dynamic conditions with respect to processing algorithm 101
Table 8-1 Parameters of the rotational tilt platform [99 p. 8] 122

List of Figures

Figure 2-1 Platform for dynamic tests: quadruped robot (platform developed by F. Iida and M. Hoffmann, University of Zurich, AI Laboratory) 8
Figure 2-2 Webots 6.1.5 modelling environment: simulated model of robot ALAN, the SIMALAN (model developed by M. Hoffmann, University of Zurich, AI Laboratory) 9
Figure 4-1 Components of gyro drift rate according to IEEE Std 528-2001 [63 p. 5] 14
Figure 4-2 Power spectral density characteristics of various noise types 16
Figure 4-3 Autocorrelation function of the 1st order Gauss-Markov process 17
Figure 4-4 White noise simulation; its power spectral density and autocorrelation function 18
Figure 4-5 Random walk simulation; its power spectral density and autocorrelation function 19
Figure 4-6 Harmonic noise simulation; its power spectral density and autocorrelation function ... 19
Figure 4-7 Exponentially correlated noise simulation; its power spectral density and autocorrelation function 19
Figure 4-8 Allan Variance sample plots with typical correlation times [74 p. 115] 20
Figure 4-9 Wavelet shifting and scaling in the discrete wavelet transformation 26
Figure 4-10 Signal decomposition principle and the level of decomposition tree [39 pp. 69, 71] 28
Figure 4-11 Accelerometer output data de-noising by the WMRA procedure 29
Figure 4-12 Conventional scheme for INS/GPS integration using a Kalman filter 30
Figure 4-13 Frames of reference used for the aircraft navigation (ECI, ECEF, NED) 35
Figure 4-14 Classical conceptual scheme of INS mechanization defined for local navigation frame [81 p. 25] 41
Figure 4-15 Definition of the misalignment angles between the non-orthogonal and the orthogonal sensor frame, i.e. the platform frame (right); definition of accelerometer and gyros sensing axes (left) [85 p. 2] 42
Figure 4-16 Centralized INS/GPS integration – open loop [34 p. 69] 46
Figure 4-17 Centralized INS/GPS integration – closed loop [34 p. 70] 47
Figure 4-18 Decentralized INS/GPS integration – open loop [34 p. 70] 47
Figure 4-19 Decentralized INS/GPS integration – closed loop [34 p. 71] 47
Figure 4-20 Time diagram showing prediction, filtering and smoothing using the Kalman filter 48
Figure 4-21 Block scheme of the discrete linearized Kalman filter operation [26 p. 122] 49
Figure 5-1 Principle of the SMC-INS integration using a KF (red arrow for feedback) [97 p. 244] .. 63
Figure 5-2 Simplified conceptual INS mechanization scheme as implemented 64
Figure 5-3 Example of a Hinton graph to represent correlation of indicators based on the data obtained for the sprint gait (gait type 0) and standard friction (friction value of 1) 80
Figure 5-4 Example of a Hinton graph to represent correlation of indicators based on the data obtained for turning right using sharp pacing gait (gait type 3) and standard friction (friction value of 1) 80
Figure 5-5 Example of indicator data for stride length based on the measurements obtained for the sprint gait (gait type 0) and standard friction (friction value of 1) 81
Figure 5-6 Example of indicator data for stride length based on the measurements obtained for turning right using sharp pacing gait (gait type 3) and standard friction (friction value of 1) 82
Figure 5-7 Example of indicator data for delta heading based on the measurements obtained for turning right using sharp pacing gait (gait type 3) and standard friction (friction value of 1) 82

Figure 5-8 Example of the *frequency* indicator data fitting in least squares sense for sprint gait (gait type 0) ... 83
Figure 5-9 SMC-INS implementation scheme as realized in MATLAB ... 86
Figure 5-10 Covariance analysis of the developed error model – the velocity error states ... 89
Figure 5-11 Covariance analysis of the developed error model – the position error states ... 90
Figure 5-12 Covariance analysis of the developed error model – the attitude error states ... 90
Figure 6-1 Allan deviation of the MTi-OEM Xsens angular rate sensors ... 92
Figure 6-2 Allan deviation of the MTi-OEM Xsens accelerometers ... 93
Figure 6-3 Example of roll angle data measured at angular rate of 1 °/s by AHRS M3 unit and the IRC sensors serving as reference ... 95
Figure 6-4 Example of de-noised angular rate data using optimal WMRA for AHRS M3 unit ... 96
Figure 6-5 Example of accelerometer data de-noised using the WMRA procedure for level of signal decomposition 3 and 5 (parameters: *heursure, soft, sln, sym8,* sampling at 100 Hz) ... 97
Figure 6-6 Angular rate sensor data de-noised using the WMRA procedure for level of signal decomposition 3 and 5 (parameters: *minimaxi, hard, one, sym8*, sampling at 100 Hz) ... 98
Figure 6-7 MT Manager (Xsens) software for calibrated inertial data logging and online visualisation ... 99
Figure 6-8 Experimental setup for attitude stability evaluation under high dynamics: constrained treadmill with speed regulation and ALAN with MTi-OEM inertial unit (Xsens) ... 100
Figure 6-9 Attitude comparison plots: pitch and roll signals processed using INS mechanization only (red) and with legged odometer aiding using the EKF based SMC_INS (blue) ... 101
Figure 6-10 SMC-INS concept verification using SIMALAN simulation in Webots ... 102
Figure 6-11 Raw accelerations and angular rates data collected using the Webots inertial measurement unit ... 103
Figure 6-12 Legged odometer data obtained from SMC data measured in Webots; estimated stride length and delta heading (blue), true stride length and delta heading reference (red) ... 103
Figure 6-13 Estimated velocity, position and attitude results regarding the SMC-INS applied on Webots data ... 104
Figure 6-14 Attitude comparison plots regarding the SMC-INS algorithm applied on Webots data; estimated SMC-INS attitude signals (blue), true attitude reference (red); RMSE less than 0.3 deg for all angles ... 104
Figure 6-15 Absolute position plot regarding the SMC-INS applied on Webots data ... 105
Figure 6-16 Relative 3D position plot regarding the SMC-INS applied on Webots tracking data ... 106
Figure 6-17 Estimated velocity, position and attitude results regarding the SMC-INS applied on long round-track Webots data ... 107
Figure 6-18 Relative position plot regarding the SMC-INS applied on Webots turning gait data; estimated SMC-INS track (blue), Webots true reference (red) ... 107
Figure 6-19 Quadruped robotic platform ALAN equipped with MTi-OEM unit (Xsens) ... 108
Figure 6-20 Tracker 3.1: free video tracking software for computing precise reference trajectory ... 109
Figure 6-21 Accelerations and angular rates for slow turning left gait measured by the MTi-OEM unit (Xsens) ... 109
Figure 6-22 Velocity, position, and attitude results regarding SMC-INS application on real inertial and SMC data acquired with ALAN during turning left gait ... 110

Figure 6-23 Trajectory results regarding SMC-INS applied on real inertial and SMC data measured during turning left gait (includes SMC-INS trajectory, LO legged odometer aiding signal and TRK video reference) 110
Figure 6-24 Precision of the estimated trajectory in time by means of RMSE for turning left gait 110
Figure 6-25 Accelerations and angular rates for combined gait measured by the MTi-OEM unit (Xsens) 111
Figure 6-26 Velocity, position and attitude results regarding the SMC-INS applied on real data inertial data (MTi-OEN, Xsens) and SMC data measured for combined gait 111
Figure 6-27 Measured and de-noised acceleration data (de-noising using WMRA LOD 4 with sym8 wavelet) 112
Figure 6-28 Measured and de-noised angular rates (de-noising using WMRA LOD 4 with sym8 wavelet) 112
Figure 6-29 Trajectory comparison of the SMC-INS based on the EKF and the UKF with optimal WMRA data de-noising (includes SMC-INS trajectory estimated using EKF, SMC-INS trajectory estimated using UKF with WMRA data de-noising, LO legged odometer aiding signal, and TRK video reference) 112
Figure 6-30 Comparison of precision development in time by means of RMSE between the trajectory estimated using the EKF (green and black) and the UKF with optimal WMRA de-noising (blue and red) 113
Figure 6-31 Trajectory results regarding the UKF based SMC-INS performing on terrain with very low friction that causes slippage (SMC-INS estimated trajectory, LO legged odometer aiding, and TRK reference) 114
Figure 6-32 Stride length data (LO stride length estimated by legged odometer and provided as aiding, REF stride length reference, INS stride length computed after data fusion using the UKF based SMC-INS) (top). Attitude results regarding the UKF based SMC-INS performing on terrain with very low friction (bottom). 114
Figure 6-33 Precision of the estimated trajectory in time for the combined gait on terrain with low friction 114
Figure 8-1 3DM-GX2 Attitude and Heading Reference System (MicroStrain) 118
Figure 8-2 AHRS M3 Attitude and Heading Reference System (Innalabs) 119
Figure 8-3 MTi-OEM and MT9 Xsens Attitude and Heading Reference Systems (Xsens) 120
Figure 8-4 Rotational tilt platform for inertial sensors testing and calibration 121
Figure 8-5 Measuring setup for inertial sensors testing 122

REFERENCES

[1] Luo, R. C., Yih, Chih-Chen and Su, Kuo Lan., "Multisensor fusion and integration: approaches, applications, and future research directions." *Sensors Journal, IEEE .* DOI: 10.1109/JSEN.2002.1000251 , 2002, Vol. 2, Issue 2, pp.107-119.
[2] Luczak, S., Oleksiuk, W. and Bodnicki, M., "Sensing Tilt With MEMS Accelerometers." *Sensors Journal, IEEE.* DOI: 10.1109/JSEN.2006.881433, 2006, Vol. 6, Issue 6, pp.1669-1675.
[3] Tan, U-Xuan, et al., "Estimating Displacement of Periodic Motion With Inertial Sensors." *Sensors Journal, IEEE.* DOI: 10.1109/JSEN.2008.917488, 2008, Vol. 8, Issue 8, pp.1385-1388.
[4] Syed, Z. F., et al., "Civilian Vehicle Navigation: Required Alignment of the Inertial Sensors for Acceptable Navigation Accuracies." *Vehicular Technology, IEEE Transactions on.* DOI 10.1109/TVT.2008.921616, 2008, Vol. 57, Issue 6, pp. 3402-3412.
[5] Alban, S., "Design and Performance of a Robust GPS/INS Attitude System for Automobile Applications." *PhD Thesis.* Stanford : s.n., 2004.
[6] Gebre-Egziabher, D., Hayward, R. C. and Powell, J. D., "A low-cost GPS/inertial attitude heading reference system (AHRS) for general aviation applications." *Position Location and Navigation Symposium, IEEE 1998.* DOI 10.1109/PLANS.1998.670207, 1998, pp. 518-525.
[7] Zhu, Rong and Zhou, Zhaoying., "A Small Low-Cost Hybrid Orientation System and Its Error Analysis." *Sensors Journal, IEEE.* DOI: 10.1109/JSEN.2008.2012196, 2009, Vol. 9, Issue 3, pp. 223-230.
[8] Vadlamani, A. K. and de Haag, M. U., "Synthesis of Airborne Laser Measurements for Navigation Algorithms." *Sensors Journal, IEEE.* DOI: 10.1109/JSEN.2008.920717 , 2008, Vol. 8, Issue 8, pp. 1411-1412.
[9] Vasconcelos, J. F., et al., "Embedded UAV model and LASER aiding techniques for inertial navigation systems." *Control EngineeringPractice.* Elsevier, 2010, Vol. 18, Issue 3, pp. 262–278.
[10] Thrun, S., et al., "Stanley: The Robot that Won the DARPA Grand Challenge." *Journal of Field Robotics 23(9), 661–692 (2006) © 2006.* Wiley Periodicals, Inc., 2006, Vol. 9, Issue 23.
[11] Lazarus, S. B., et al., "Vehicle Localization Using Sensors Data Fusion Via Integration of Covariance Intersection and Interval Analysis." *Sensors Journal, IEEE.* DOI: 10.1109/JSEN.2007.901556, 2007, Vol. 2, Issue 2, pp.107-119.
[12] Sturm, J and Visser, A., "An appearance-based visual compass for mobile robots." *Elsevier.* Robotics and Autonomous Systems, 31 May 2009, Vol. 57, Issue 5, pp. 536-545.
[13] Klaassen, B., et al., "Biomimetic walking robot SCORPION: Control and modeling." *Robotics and Autonomous Systems.* Elsevier, 2002, Vol. 41, pp. 69-76.
[14] Cobano, J. A., Estremera, J. and Gonzalez de Santos, P., "Location of legged robots in outdoor environments." *Robotics and Autonomous Systems.* Elsevier, 2008, Vol. 56, pp. 751–761.
[15] Etienne, A. S. and Jeffery, K. J., "Path Integration in Mammals." *HIPPOCAMPUS.* Wiley-Liss, Inc, 2004, Vol. 14, pp. 180-192.
[16] Wang, Chieh-Chih, et al., "Simultaneous Localization, Mapping and Moving Object Tracking." *The International Journal of Robotics Research.* 2007, Vol. 26, pp. 889-916.
[17] Gustafsson, F., "Automotive safety systems." *Signal Processing Magazine, IEEE.* DOI 10.1109/MSP.2009.932618, 2009, Vol. 26, Issue 4, pp. 32-47.
[18] Fleming, W. J., "New Automotive Sensors—A Review." *Sensors Journal, IEEE .* DOI: 10.1109/JSEN.2008.2006452 , 2008, Vol. 8, Issue 11, pp. 1900-1921.

[19] Neul, R., et al., "Micromachined Angular Rate Sensors for Automotive Applications." *Sensors Journal, IEEE .* DOI: 10.1109/JSEN.2006.888610 , 2007, Vol. 7, Issue 2, pp.302-309.
[20] Dissanayake, G., et al., "The Aiding of a Low-Cost Strapdown Inertial Measurement Unit Using Vehicle Model Constraints for Land Vehicle Applications." *IEEE Transactions on Robotics and Automation.* 2001, Vol. 17, Issue 5, pp. 731 - 747.
[21] Liu, Tao, Inoue, Yoshio and Shibata, Kyoko., "Development of a wearable sensor system for quantitative gait analysis." *Measurement.* Elsevier, 2009, Vol. 42, Issue 7, pp. 978-988, ISSN 0263-2241, DOI: 10.1016/j.measurement.2009.02.002.
[22] Lightner, M. R. and Erdogmus, D., "Signal processing challenges in cognitive assistive technology." *Signal Processing Magazaine, IEEE.* DOI 10.1109/MSP.2008.928315, 2008, Vol. 25, Issue 5, pp. 103-108.
[23] Titterton, D. H. and Weston, J. L., *Strapdown Inertial Navigation Technology.* Lavenham, UK : The Lavenham Press ltd, 1997. ISBN 0 86341 260 2.
[24] Farrell, J. A., *Aided Navigation: GPS with High Rate Sensors.* s.l. : McGraw-Hill, 2008. DOI: 10.1036/0071493298.
[25] Salychev, O., *Applied Inertial Navigation:Problems and Solutions.* ISBN 5-7038-2395-1 : Bauman MSTU Press, 2004.
[26] Grewal, M. S. and Andrews, A. P., *Kalman Filtering - Theory and Practice using Matlab.* New York : Wiley-Interscience, 2001.
[27] Grewal, M. S., Weill, L. R. and Andrews, A. P., *Global Positioning Systems, Inertial Navigation, and Integration.* s.l. : John Wiley & Sons, Inc., 2001. ISBN: 0-471-20071-9.
[28] Savage, P. G., *What Do Accelerometers Measure?* [pdf] s.l. : Strapdown Associates, Inc., May 8, 2005. Available on the Internet at www.strapdownassociates.com.
[29] Maybeck, P. S., *Stochastic Models, Estimation and Control.* New York : Academic Press, 1997. ISBN 0-12-480701-1(v.1).
[30] Rogers, R. M., *Applied Mathematics in Integrated Navigation Systems.* Reston, USA : s.n., 2003.
[31] Kalman, R. E., "A new approach to linear filtering and prediction problems." *ASME Journal of Basic Engineering .* 1960, Vol. 82, pp. 34-45.
[32] Shin, E-H., "Estimation Techniques for Low-Cost Inertial Navigation." *Ph.D. dissertation.* Calgary, Canada : Department of Geomatics Engineering, University of Calgary, 2005. Vol. UCGE Reports Number 20219.
[33] Bergman, N., "Recursive Bayesian Estimation: Navigation and Tracking Applications." *PhD Thesis.* Linkoping, Sweden : Department of Electrical Engineering, Linkoping University, 1999. Vols. SE-581 83.
[34] Abdel-Hamid, W., "Accuracy Enhancement of Integrated MEMS-IMU/GPS Systems for Land Vehicular Navigation Applications." *Ph.D. dissertation.* Calgary, Canada : Department of Geomatics Engineering, University of Calgary, 2005. Vol. UCEG Reports Number 20207.
[35] Sotak, M., Sopata, M. and Berezny, S., "Integrated navigation system using sigma-point Kalman filter and particle filter." *Military Capabilities Enabled by Advances in Navigation Sensors.* s.l. : RTO Sensors and Electronics Technology Panel (SET) Symposium : Antalya, 1-2 October 2007. RTO/NATO, 2007. p. 27-1-27-12.
[36] Hayes, M. and Treichler, J., "Adaptive Filtering [Best of the Web]." *Signal Processing Magazine, IEEE.* DOI 10.1109/MSP.2008.929817, 2008, Vol. 25, Issue 6, pp. 169-172.

[37] Julier, S. J. and Uhlmann, J. K., "A General Method for Approximating Nonlinear Transformations of Probability Distributions." *tech.rep.* s.l. : Dept. Engineering Science, University of Oxford, 1996. Vol. OX1 3PJ.
[38] —. "A New Extension of the Kalman Filter to Nonlinear Systems." *tech.rep.* s.l. : Dept. of Engineering Science, University of Oxford, UK, 1997. Vol. OX1 3PJ.
[39] Nassar, S., "Improving the Inertial Navigation System (INS) Error Model for INS and INS/DGPS Applications." *PhD Thesis.* Calgary, Canada : Department of Geomatics Engineering, University of Calgary, 2003. Vol. UCEG Reports Number 20183.
[40] Doucet, A. and Wang, Xiaodong., "Monte Carlo methods for signal processing: a review in the statistical signal processing context." *Signal Processing Magazine, IEEE.* DOI 10.1109/MSP.2005.1550195, 2005, Vol. 22, Issue 6, pp.152-170.
[41] Kandepu, R., Foss, B. and Imsland, L., "Applying the unscented Kalman filter for nonlinear state estimation." *Journal of Process Control.* Elsevier, 4 November 2007, Vol. 18 (2008), pp. 753–768.
[42] Kolas, S. and Foss, B. A., "Constrained Nonlinear State Estimation based on the UKF approach." *Computers and Chemical Engineering.* 5 January 2009, Vols. PII: S0098-1354(09)00020-9, DOI: doi:10.1016/j.compchemeng.2009.01.012.
[43] Skog, I. and Händel, P., "In-Car Positioning and Navigation Technologies—A Survey." *IEEE Transactions on Intelligent Transportation Systems.* March 2009, Vol. 10, Issue 1, pp. 4-21.
[44] Pfeifer, R., Lungarella, M. and Iida, F., "Self-Organization, Embodiment, and Biologically Inspired Robotics." *Science.* November 2007, Vol. 318, ISSN 0036-8075; online ISSN 1095-9203.
[45] Pfeifer, R. and Scheier, Ch., *Understanding Intelligence.* 700 pp. : MIT Press, September 1999. ISBN-10: 0-262-16181-8, ISBN-13: 978-0-262-16118-7.
[46] Pfeifer, R. and Bongard, J. C., *How the Body Shapes the Way We Think - a New View of Intelligence.* 394 pp. : MIT Press, November 2006. ISBN-10: 0-262-16239-3, ISBN-13:978-0-262-16239-5.
[47] Delcomyn, F. and Nelson, M. E., "Architectures for a biomimetic hexapod robot." *Robotics and Autonomous Systems.* Elsevier, 2000, Vol. 30, pp. 5-15.
[48] Durgin, F. H., et al., "The precision of locometry in humans." *Exp Brain Res.* 2009, Vol. 193, pp.429-436, pp. 429-436.
[49] Wittlinger, M, Wehner, R. and Wolf, H., "The ant odometer: stepping on stilts and stumps." *Science.* 2006, Vol. 312.
[50] Liu, S.-H., Huang, T.-S. and Yen, J.-Y., "Comparison of sensor fusion methods for an SMA-based hexapod." *Robotics and Autonomous Systems.* Elsevier, 2009, Vols. ISSN 0921-8890, doi:10.1016/j.robot.2009.10.006.
[51] Thielin-Bescond, M. and Beuhnon, G., "Vision-independent odometry in the ant Cataglyphis cursor." *Naturwissenschaften.* 2005, Vol. 92, pp. 193-197.
[52] Lin, P-C., Komsuoglu, H. and Koditschek, D. E., "Sensor data fusion for body state estimation in a hexapod robot with dynamical gaits." *IEEE Transactions on Robotics.* 2006, Vol. 22, Issue 5, pp. 932-943.
[53] —. "A Leg Configuration Measurement System for Full Body Posture Estimates in a Hexapod Robot." *IEEE TRansactions on RObotics.* 2005, Vol. 21, Issue 3, pp. 411-422.
[54] —. "Legged Odometry from Body Pose in a Hexapod Robot." *Springer Tracts in Advanced Robotics .* 2004, Vol. 21: Experimental Robotics IX, pp. 439-448.

[55] Haykin, S., "Adaptive Systems for Signal Process." [book auth.] S. Stergiopoulos. *Advanced Signal Processing Handbook.* Boca Raton : CRC Press LLC, 2001.
[56] Gorce, P., "Dynamic postural control method for biped in unknown environment." *IEEE Trans. Robot. Automat.* 1999, Vol. 29, 6, pp. 616-626.
[57] Rehbinder, H. and Xiaoming, H., "Drift-free attitude estimation for accelerated rigid bodies." *Automatica.* 2004, Vol. 40.
[58] Iida, F. and Pfeifer, R., "Sensing through body dynamics." *Elsevier.* Robotics and Autonomous Systems, August 2006, Vol. 54, Issue 8, pp. 631-640.
[59] Webots., *Webots Reference Manual.* [Online] October 1, 2009. http://www.cyberbotics.com/cdrom/common/doc/webots/reference/reference.html.
[60] Vikas Kumar, N., "Integration of Inertial Navigation System and Global Positioning System Using Kalman Filtering." *M.S. dissertation.* Bombay : Indian Institute of Technology, Department of Aerospace Engineering, July 2004.
[61] Savage, P. G., *What Do Gyros Measure?* [pdf] s.l. : Strapdown Associates, Inc., January 4, 2006. Available on the Internet at www.strapdownassociates.com.
[62] —. *What Do Inertial Sensors Measure?* [pdf] s.l. : Strapdown Associates, Inc., September 18, 2005. Available on the Internet at www.strapdownassociates.com.
[63] IEEE., *IEEE Standard for Inertial Sensor Terminology.* [PDF] New York : IEEE Std 528-2001, 2001. ISBN 0-7381-3022-2 SS94961.
[64] Poularikas, A. D., *The Hanbook of Formulas and Tables for Signal Processing.* USA : A CRC Handbook Published in Cooperation with IEEE Press, 1999. ISBN 0-8493-8579-2.
[65] Madisetti, V. K. and Williams, D. B., *Digital Signal Processing Handbook.* [pdf] Atlanta, USA : s.n., 1999.
[66] IEEE., *IEEE Standard Specification Format Guide and Test Procedure for Single-axis Laser Gyros.* [PDF] s.l. : IEEE Std 647-2006 (Revision of IEEE Std 647-1995), 2006. DOI 10.1109/IEEESTD.2006.246241, pp.1-83.
[67] Allan, D. W., "Statistics of Atomic Frequency Standards." *Procedings of the IEEE.* 1966, Vol. 54, Issue 2, pp. 221–230.
[68] Lawrence, C. Ng., "On The Application of Allan Variance Method for Ring Laser Gyro Performance Characterization." *tech.rep.* USA : Lawrence Livermore National Laboratory, 1993. UCRL-ID-115695.
[69] Sotak, M., "Estimation of Stochastic Coefficients of Inertial Sensors." *Science & Military.* ISSN 1336-8885, 2008, Vol. 3, No. 2, pp. 13-16.
[70] Woodman, O. J., "An introduction to inertial navigation." *Technical Report.* s.l. : University of Cambridge, August 2007. UCAM-CL-TR-696. ISSN 1476-2986.
[71] El-Sheimy, N., Haiying, H. and Xiaoji, N., "Analysis and modeling of inertial sensors using Allan variance." *IEEE Transactions on Instrumentation and Measurement .* 2008, Vol. 57, pp. 140-149.
[72] Trusov, A. A., Schofield, A. R. and Shkel, A. M., "Performance characterization of a new temperature-robust gain-bandwidth improved MEMS gyroscope operated in air." *Sensors and Actuators A: Physical.* November 2008.
[73] Gambis, D., "Allan Variance in earth rotation time series analysis." *Advances in Space Research.* July 2002, Vol. 30, Issue 2, pp. 207 - 212.
[74] Reinštein, M., Šipoš, M. and Roháč, J., "Error Analyses of Attitude and Heading Reference Systems." *Przeglad Elektrotechniczny.* 2009, Vol. 85, Issue 8, 114-118.

[75] Savage, P. G., "Strapdown Inertial Navigation Integration Algorithm Design Part 1: Attitude Algorithms." *Journal of Guidance, Control, and Dynamics.* 1998, Vol. 21, No.1, pp. 19 - 28.
[76] —. "Strapdown Inertial Navigation Integration Algorithm Design Part 2: Velocity and Position Algorithms." *Journal of Guidance, Control, an Dynamics.* 1998, Vol. 21, No. 2, pp. 208 - 221.
[77] MathWorks, The., "Matlab Product Help." s.l. : The MathWorks. September 17, 2008. 7.7.0.471 (R2008b).
[78] Sotak, M., et al., *Navigation Systems Integration.* Kosice, Slovakia : Published by Robert Breda, 2006. ISBN 80-969619-9-3 (in Slovak).
[79] Philips, W. F., *Mechanics of Flight.* February 2004. ISBN 978-0-471-33458-3.
[80] Britting, K. R., *Inertial Navigation Systems Analysis .* s.l. : John Wiley & Sons Inc, 1971. ISBN-10: 047110485X, ISBN-13: 978-0471104858.
[81] Shin, E-H., "Accuracy Improvement of Low Cost INS/GPS for Land Applications." *M.S. thesis.* Calgary, Canada : Department of Geomatics Engineering, University of Calgary, December 2001. Vol. UCGE Reports Number 20156.
[82] Ali, J. and Ushaq, M., "A consistent and robust Kalman filter design for in-motion alignment of inertial navigation system." *Measurement.* Elsevier, 10 October 2008, DOI doi:10.1016/j.measurement.2008.10.002.
[83] Sotak, M., "Application of Wavelet Analysis to Inertial Measurements." *Science & Military.* ISSN 1336-8885, 2008, Vol. 3, No. 2, pp. 17-20.
[84] Chatfield, A. B., *Fundamentals of High Accuracy Inertial Navigation.* USA : American Institute of Aeronautics and Astronautics, Inc., 1997. ISBN 1-56347-243-0.
[85] Skog, I. and Händel, P., *Calibration of a MEMS Inertial Measurement Unit.* [PDF] Rio de Janeiro, Brazil : XVII IMEKO WORLD CONGRESS, Metrology for a Sustainable Development, 2006. pp. 17-22.
[86] Madsen, K., Nielsen, H. B. and Tingleff, O., *Methods for Non-linear Least Squares Problems.* s.l. : Informatics and Mathematical Modelling, Technical University of Denmark, April 2004.
[87] Jurman, D., et al., "Calibration and data fusion solution for the miniature attitude and heading reference system." *Sensors and Actuators A: Physical.* Elsevier, 2007, Vol. 138, Issue 2, pp. 411-420.
[88] Caron, F., et al., "GPS/IMU data fusion using multisensor Kalman filtering: Introduction of contextual aspects." *Information Fusion.* 2006, Vol. 7, pp. 221-230.
[89] Sotak, M. and Breda, R., "New Approach in the Systems Integration." *MTA Review.* ISSN 1843-3391, 2008, Vol. XVIII, No. 1, pp. 63-68.
[90] Liggins, M. E., Hall, D. L. and Llinas, J., *Handbook of Multisensor Data Fusion: Theory and Practice Electrical Engineering & Applied Signal Processing.* s.l. : Crc Press, 2008. ISBN 9781420053081 (1420053086).
[91] Hall, D. L. and McMullen, S. A. H., *Mathematical Techniques in Multisensor Data Fusion.* s.l. : Artech House Publishers, 2004. ISBN: 9781580533355 (1580533353).
[92] Obolensky, N., "Kalman Filtering Methods for Moving Vehicle Tracking." *MSc Thesis.* s.l. : University of Florida, 2002.
[93] Simon, D., *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches.* s.l. : John Wiley & Sons, Inc., July 2006. ISBN: 978-0-471-70858-2.
[94] Zhang, Haitao, et al., "The Application and Design of Second Order EKF Based on GPS/DR Integration for Land Vehicle Navigation." s.l. : MultiMedia and Information Technology, 2008. MMIT '08. International Conference on, 30-31 December 2008. ISBN: 978-0-7695-3556.

[95] Zhang, Haitao, Rong, Jian and Zhong, Xiaochun., "The performance comparison and algorithm analysis of first order EKF, second order EKF and smoother for GPS/DR navigation." s.l. : Communication Technology, 2008. ICCT 2008. 11th IEEE International Conference on, 10-12 November 2008. ISBN: 978-1-4244-2250-0.
[96] Abdel-Hafez, M. F., "On the development of an inertial navigation error-budget system." *Journal of the Franklin Institute.* Elsevier, 2009, Vols. In Press, Corrected Proof, doi:10.1016/j.jfranklin.2009.02.008.
[97] Reinštein, M., Roháč, J. and Šipoš, M., "Algorithms for Heading Determination using Inertial Sensors." *Przeglad Elektrotechniczny.* 2010, Vol. 85, Issue 8, pp. 243-246.
[98] Thong, Y. K., et al., "Numerical double integration of acceleration measurements in noise." *Measurement.* Elsevier, July 2004, Vol. 36, Issue 1, pp. 73-92.
[99] Ficek, P., "Systém pro řízení náklonné rotační plošiny." *Diploma Thesis.* Prague : CTU, Faculty of Electrical Engineering, 2009.

RELATED WORK AND PUBLICATIONS

All of the related and unrelated author's publications are given in the list below sorted in chronological order; journal articles and international conference proceedings are differentiated.

2007: International Conference Proceedings

[1] Reinštein, M. - Roháč, J.: *Measuring System for Angular Rate Sensors*. In POSTER 2007 [CD-ROM]. Prague: CTU, Faculty of Electrical Engineering, 2007,

[2] Reinštein, M. - Roháč, J.: *Modelling and Evaluation of Inertial Sensors*. In MDS - Měření, diagnostika, spolehlivost palubních soustav letadel - 7. mezinárodní vědecká konference. Brno: Univerzita obrany, Fakulta vojenských technologií, 2007, p. 97-105. ISBN 978-80-7231-281-8.

[3] Reinštein, M. - Roháč, J.: *Signal Processing and Data Transfer Concerning Angular Rate Sensors*. In The third AIAA-Pegasus Student Conference [CD-ROM]. Napoli: Universita degli Studi di Napoli "Federico II", 2007,

[4] Reinštein, M. - Roháč, J.: *System for Verification of Adaptive Algorithms Enhancing the Precision of Low Cost Inertial Sensors*. In MOSATT 2007, Proceedings of the International Scientific Conference "Modern Safety Technologies in Transportation". Košice: Slovak Transport Society at the Slovak Academy of Sciences, 2007, p. 224-230. ISBN 978-80-969760-2-7.

2008: Journals

[5] Reinštein, M. - Roháč, J. - Šipoš, M.: *Improving Performance of a Low-cost AHRS*. Acta Avionica. 2008, vol. X, no. 16, p. 132-137. ISSN 1335-9479.

2008: International Conference Proceedings

[6] Reinštein, M. - Pačes, P. - Roháč, J. - Šipoš, M.: *Advanced Implementations Techniques in Kalman Filtering*. In 2008 PEGASUS-AIAA Student Conference [CD-ROM]. Prague: Czech Technical University, 2008,

[7] Reinštein, M. - Šipoš, M.: *Improving Performance of a Low-cost AHRS*. In New Development Trends In Aeronautics. Košice: Technical University of Košice, 2008, p. 109-110. ISBN 978-80-553-0067-2.

[8] Reinštein, M. - Roháč, J. - Šipoš, M.: *Turbulence Modelling for Attitude Evaluation Purposes*. In Sborník z 8. mezinárodní vědecké konference "Měření, diagnostika a spolehlivost palubních soustav letadel". Brno: Univerzita obrany, Fakulta vojenských technologií, 2008, p. 46-54. ISBN 978-80-7231-555-0.

[9] Šipoš, M. - Reinštein, M. - Roháč, J.: *Levenberg-Marquardt Algorithm for Accelerometers Calibration*. In 8. mezinárodní konference "Měření, diagnostika a spolehlivost palubních soustav letadel". Brno: Univerzita obrany, Fakulta vojenských technologií, 2008, p. 39-45. ISBN 978-80-7231-555-0.

[10] Šipoš, M. - Reinštein, M. - Roháč, J.: *System for Measuring Tilt-Angle*. In New Development Trends In Aeronautics. Košice: Technical University of Košice, 2008, p. 120-121. ISBN 978-80-553-0067-2.

[11] Šipoš, M. - Roháč, J. - Reinštein, M.: *Measurement with Electrolytic Tilt Sensor*. In 2008 PEGASUS-AIAA Student Conference [CD-ROM]. Prague: Czech Technical University, 2008,

[12] Roháč, J. - Reinštein, M.: *New Trends in Angular Rate Sensing and Attitude Evaluation*. In Nové trendy v civilním letectví. Praha: ČVUT, Fakulta dopravní, 2008, vol. 1, p. 155-161. ISBN 978-80-7204-604-1.

[13] Pačes, P. - Reinštein, M. - Draxler, K.: *Fusion of Smart Sensor Standards and Sensors with Self-validating Abilities*. In 27th DASC Digital Avionics Systems Conference [CD-ROM]. St. Paul, Minnesota: IEEE, 2008, ISBN 978-1-4244-2208-1.

2009: Journals

[14] Reinštein, M. - Šipoš, M. - Roháč, J.: *Error Analyses of Attitude and Heading Reference Systems*. Przeglad Elektrotechniczny. 2009, vol. 85, no. 8, p. 114-118. ISSN 0033-2097

2009: International Conference Proceedings

[15] Reinštein, M.: *Development of a Low-cost Inertial Navigation Unit and a Testing of Filters for Inertial Navigation*. In Workshop 09 [CD-ROM]. Prague: CTU, 2009, ISBN 978-80-01-04286-1.
[16] Šipoš, M. - Pačes, P. - Reinštein, M. - Roháč, J.: Flight Attitude Track Reconstruction Using Two AHRS Units under Laboratory Conditions In: IEEE SENSORS 2009 - The Eighth IEEE Conference on Sensors [CD-ROM]. Christchurch: IEEE Sensors Council, 2009, p. 675-678. ISBN 978-1-4244-5335-1.

2010: Journals

[17] Pačes, P. - Reinštein, M. - Draxler, K.: *Fusion of Smart Sensor Standards and Sensors with Self-Validating Abilities. Journal of Aircraft*. 2010, vol. 47, no. 3, p. 1041-1046. ISSN 0021-8669
[18] Reinštein, M. - Roháč, J. - Šipoš, M.: *Algorithms for Heading Determination using Inertial Sensors In: Przeglad Elektrotechniczny*. 2010, vol. 86, no. 9, p. 243-246. ISSN 0033-2097

2010: International Conference Proceedings

[19] Ripka, P. - Nováček, P. - Reinštein, M. - Roháč, J.: *Position sensing system for eddy-current mine imager* In: EUROSENSORS XXIV - Proceedings [CD-ROM]. Linz: Elsevier BV, 2010, p. 1-4. ISSN 1877-7058

Publications unrelated to the dissertation:

[20] Kubínyi, M. - Reinštein, M. - Kreibich, O.: *GNS Ambiguity Resolution Techniques*. In CUAS 2008 - Conference Proceedings [CD-ROM]. Praha: Česká odborná společnost letecká, 2008, ISBN 978-80-902522-2-6
[21] Pačes, P. - Šipoš, M. - Reinštein, M. - Roháč, J.: *Sensors of Air Data Computers - Usability and Environmental Effects* In: ICMT'09 - Proceedings of the International Conference on Military Technologies. Brno: Univerzita obrany, 2009, p. 401-409. ISBN 978-80-7231-649-6.
[22] Reinštein, M. - Petrucha, V.: *Setup for Speed Measurement Using Correlation*. In Poster 2009 [CD-ROM]. Praha: České vysoké učení technické v Praze, 2009.