LULEÅ
TEKNISKA
UNIVERSITET

2000:081

MASTER'S THESIS

Developement of a INS/GPS navigation loop for an UAV

Sven Rönnbäck

Civilingenjörsprogrammet

Institutionen för Systemteknik
Avdelningen för Robotik och automation

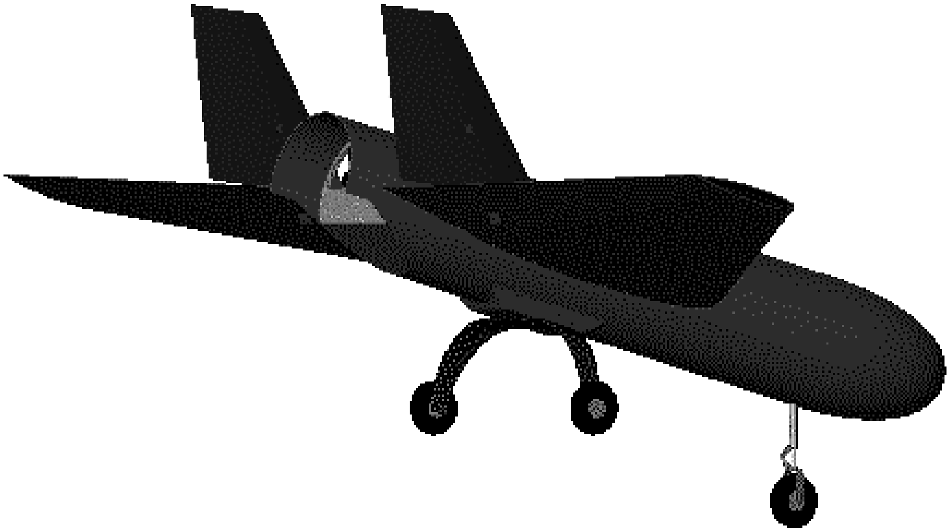
2000:081 • ISSN: 1402-1617 • ISRN: LTU-EX--00/081--SE

MASTER'S THESIS

Developement of a INS/GPS navigation loop

Sven Rönnbäck

sven_ronnback@hotmail.com



February 2000

Luleå University of Technology

Department of Computer Science and Electrical Engineering

Abstract

This Master Thesis report presents an INS[1]/GPS[2] navigation filter written in C++ using a standard matrix library called MPP. The filter have been tested on an airvehicle called Brumby. Here data have been logged from the Inertial Measurement Unit (IMU) and the Global Positioning System (GPS) receiver. This data have then been postprocessed and run through the navigation filter to estimate the position, attitude and velocity of the vehicle during flights.

The filter are feed with position, velocity and attitude observations. The filter loop estimates the attitude within 2^o and with 95% confidence. The velocity $+-1m/s$ 95%, and the position $+-2m$ with 95% confidence.

Some of the work have been done on a redundant (four axes) IMU[3] called the Tetrad.

Keywords: Unmanned Aerial Vehicle, Inertial Measuring Unit, Inertial Navigation System, Information Filter,Navigation,Realtime, Distributed, C++,Autonomous

[1] Inertial Navigation System
[2] Global Positioning System
[3] Inertial Measuring Unit

0.1 Acknowledgements

I am very grateful to Professor Hugh Durrant-Whyte who gave me the opporturnity to do my thesis work at the Department of Mechanical and Mechatronic Engineering at Sydney University.

Thanks to Professor Åke Wernersson who have inspired me to work in the field of Robotics.

Thanks to Ph.D. student Ben Grosholsky at the Department of Aeronautical Engineering for good co-operation and support.

Thanks to my supervisor Ph.D. Salah Sukkarei for supporting me in my work.

Thanks to all my friends I meet in Sydney that made my visit very pleasant.

Thanks to Saint Andrews College for an interesting stay.

And at last I want to thank the mighty God for the existens of the Universe.

Contents

Abstract **1**

0.1 Acknowledgements 1

0.2 List Of Symbols 7

0.3 Symbols 8

1 Introduction **11**

1.1 Scope of this thesis 12

1.1.1 Picture of the thesis work 12

1.2 Background 12

1.2.1 The ANSER project 14

1.2.2 The Brumby UAV airframe 15

1.3 The Future of UAVs 16

1.4 Frames 17

1.4.1 Body Frame 17

1.4.2 Earth Surface North-East-Down (NED) Frame 17

1.4.3 WGS-84 , World Geodetic System 1984 18

1.4.4 ECI-Earth Centered Interial Frame 19

1.4.5 ECEF - Earth Centered Earth Fixed Frame 19

2 Attitude Representation **20**

2.1 Rotations 21

2.1.1 Roll rotation,ϕ 22

2.1.2 Pitch,θ 23

2.1.3 Yaw,Heading ψ 24

2.1.4 True attack and attack angle 25

2.1.5 Sideslip 25

2.2 Body Rotation Rate 26

2.3 C_{bn}, Direction Cosine Representation 27

2.3.1 The C_{bn} matrix and small angles 28

2.3.2 Conversion to Eulerangles from C_{bn} represention 28

2.3.3 Calculation of Eulerangles with large pitch angle 29

2.3.4 The changerate of C_{bn} matrix 29

2.3.5 C_{bn} matrix integration in discrete time 30

2.3.6 Very accurate C_{bn} matrix integration in discrete time . . 30

2.3.7 Normalization of C_{bn} matrix 30

2.4 Eulerangle Representation 30

2.4.1 Calculation of $[\dot{\phi}, \dot{\theta}, \dot{\psi}]$ using [p,q,r] 31

2.4.2 Eulerangle integration in a discrete time system 32

2.5 Quaternion Angle Representation 33
2.5.1 Calculation of Eulerangles using Quaterion angles 34
2.5.2 Quaternionangle calculation using Euler representation . 34
2.5.3 Calculate Quaternion using the C_{bn} matrix representation 35
2.5.4 Quaternion Norm 35
2.5.5 Quaternationangle change rate 35
2.5.6 Quaternionangle update in discrete time 36

3 Inertial Sensors, IMU and INS 37
3.1 Newtonian/Inertial Sensors 38
3.1.1 Sensor Termal Compensation 38
3.1.2 Newtonian Sensor Nonlinearity 38
3.2 Multi sensor unit 39
3.2.1 The Accelerometer 39
3.2.2 The Rate Gyros 40
3.2.3 Sensor Bias 40
3.2.4 Redundant Sensor Unit 41
3.3 IMU-Inertial Measuring Unit 42
3.3.1 The Tetrad 43
3.3.2 Tetrad Commands 44
3.3.3 Tetrad IMU Fault Detection 45
3.4 INS-Inertial Navigation System 45
3.4.1 Leveling 45
3.5 Navigation Equations 47

4 Kalman and Information Filters 49
4.1 Linear System in continous time 50
4.2 Linear System in discrete time 50
4.2.1 Calculation of discrete time matrixes 50
4.2.2 Discrete Input Covariance Matrix 51
4.3 Nonlinear System 51
4.3.1 Linearized Nonlinear System in continous time 53
4.3.2 Linearized Nonlinear System in discrete time 53
4.4 The Complement Filter 53
4.4.1 Complement filter with feedback 54
4.4.2 Complement filter without feedback 54
4.5 The Kalman Filter 54
4.6 The Information Filter 56
4.6.1 Prediction 57
4.6.2 Observation 57
4.6.3 Estimation 57
4.6.4 The Innovation 58
4.6.5 Fusion 58
4.7 A Linear Error Model 59
4.8 Nonlinear Model using Eulerangles 60
4.8.1 State transition matrix 61
4.8.2 Control matrix 61
4.8.3 Observation Models 62
4.9 Nonlinear Model with Quaternions 63
4.9.1 State transition matrix 63

4.9.2 Control matrix 64
4.9.3 Observation Models 64

5 GPS-Global Positioning System 66
5.1 GPS-Global Positioning System 67
5.2 GPS-Signal errors 69
5.2.1 Multipath 70
5.2.2 Signal attenuation 70
5.2.3 Satellite motion 70
5.2.4 Ionosphere propagation delay 70
5.2.5 Satellite clock error 71
5.2.6 Troposhpere propagation delay 71
5.2.7 GPS Receiver Clock Model 71
5.3 WGS-84 transformations 72
5.3.1 ECEF to geodetic transformation 73
5.3.2 Geodetic to ECEF transformation 73
5.3.3 ECEF vector to NED transformation 73
5.4 Position Determination 74
5.4.1 Dilution Of Precition 76
5.5 Velocity determination 77
5.6 Attitude determination 78
5.7 GPS interface software 79
5.8 Coupled INS/GPS 80
5.8.1 Uncoupled INS/GPS system 80
5.8.2 Loosley coupled INS/GPS system 80
5.8.3 Tightly coupled INS/GPS system 80

6 The Implementation 82
6.1 Hungarian Notation 83
6.1.1 C++ example using hungarian notation 83
6.2 The Software 84
6.2.1 The flow of the software 84
6.3 Classes 84
6.4 For The IMU 84
6.4.1 The Sensor Class 84
6.4.2 The Sensor Cluster Class 85
6.5 The Readdata Class 85
6.6 The INS Class 85
6.7 The Angle Class 85
6.7.1 The Eulerangle Class 86
6.7.2 The Cbn Class 86
6.7.3 The Quaternians Class 86
6.8 The Linear System Class 86
6.8.1 The Kalman Filter Class 86
6.8.2 The Information Filter Class 86
6.8.3 The Innovation Class 86
6.9 The Vector Integration Class 87
6.9.1 The Trapetz Vector Integration Class 87
6.9.2 The Simpson Vector Integration Class 87
6.9.3 The Runge-Kutta Vector Integration Class 87

6.10 The Model Class . . . 87
6.10.1 The Quaternions Model Class . . . 88
6.10.2 The Eulerangle Model Class . . . 88
6.10.3 The Linear 9 State Model Class . . . 88
6.11 Other Classes . . . 88
6.11.1 The Vector Statistics Class . . . 88
6.11.2 The Fast Trigonometric Class . . . 88
6.11.3 The Earth Class . . . 88
6.11.4 The Serial Communication Class . . . 88
6.11.5 The Firmware Class . . . 88
6.11.6 The Logging Class . . . 88
6.12 The target computer . . . 89

7 Results 90
7.1 Testflight . . . 91
7.2 The Results . . . 94
7.3 Tetrad and Mpac IMU solutions . . . 94
7.3.1 IMU accelerations . . . 94
7.4 Navigation filter results . . . 95
7.5 IMU-Inertial Measurement Unit . . . 96
7.5.1 PSD of Inertial Sensors . . . 96
7.5.2 IMU Acceleration Solution . . . 98
7.5.3 Body frame velocity . . . 99
7.5.4 Body frame velocity angles, slideslip and angle of attack . 100
7.5.5 IMU Rotationrate Solution . . . 101
7.5.6 INS-Inertial Navigation System . . . 101
7.5.7 INS acceleration . . . 102
7.5.8 INS velocity . . . 103
7.5.9 INS omega, eulerangle velocities . . . 104
7.5.10 INS attitude . . . 105
7.5.11 The Position Innovation . . . 106
7.5.12 The Velocity Innovation . . . 107
7.5.13 The Attitude Innovation . . . 108
7.6 The Flight Playback . . . 109
7.7 The conclusions . . . 114
7.7.1 Bias and nonlinearity states in filter . . . 114
7.7.2 More efficient algoritms . . . 114
7.7.3 Vibration damping . . . 114
7.7.4 Tilt sensors. . . . 114
7.7.5 Placement of GPS antennas . . . 115
7.7.6 The angle classes . . . 115
7.7.7 The eulerangle model . . . 115
7.7.8 The quaternion model . . . 115
7.7.9 Use of GPS pseudoranges . . . 116
7.8 Future Hardware Model . . . 116

8 Appendix **1**
8.1 Data Specifications 2
8.1.1 Desk Computer Platform 2
8.1.2 Target System Computer Platform 2
8.1.3 Motion Pack IMU 2
8.1.4 Data specification of the Watson IMU 3
8.1.5 Tetrad IMU 3
8.1.6 GPS Receiver 3
8.2 Numerical Integrators 4
8.3 Gaussian Distrubutions 4
8.4 Confidence Ellipses 6
8.4.1 Two Dimensional Confidence Ellipse 7
8.5 Rayleigh Distribution 8
8.6 Wienerprocess-Randomwalk 8

Nomenclature

0.2 List Of Symbols

0.3 Symbols

Symbol	Explanation
$\dot{x}$	Time derivative of position x. In this case velocity $[m/s]$
$\ddot{x}$	Second time derivative of position x. In this case acceleration $[m/s^2]$
∇F	Linearization of system equation given by F.
a_N	North acceleration in Navigation Frame. [m/s^2]
a_E	East acceleration in Navigation Frame. [m/s^2]
a_D	Down acceleration in Navigation Frame. [m/s^2]
x_k	Is the k'th element from a sequence, $x = [x_1, x_2, ..., x_n]^T$
b_i	Represents bias
c_0	Velocity of light 299792458 [m/s]
c_{ij}	Components of matrix C. $j, i = 1..3$
C_{bn}	Rotation from Navigation Frame to Body Frame
C_{nb}	Rotation from Body Frame to Navigation Frame
D_i	Dopplershift to satellite i
$E = [e_0, e_1, e_2, e_3]$	Quaternion angle with elements.
f	System matrix in a continous time system, $\dot{x} = f\,x$
$\mathbf{g_e}$	Earth gravity vector $[m/s^2]$
I^{ixi}	Eye matrix with size i x i.
K	Kalman gain matrix
$\mathbf{l}$	Inline sight vector from position **a** to **b**. $\mathbf{l} = \frac{\mathbf{b}-\mathbf{a}}{\lvert\mathbf{b}-\mathbf{a}\rvert}$
L_1, L_2	GPS frequencies $L_1 = 1575.42$ [MHz], $L_2 = 1227.6$ [MHz]
0^{ixj}	Zero matrix with i rows and j columns.
p	Measured rotationsspeed of x_{BODY}-axis [rad/s]
q	Measured rotationspeed of y_{BODY}-axis [rad/s]
r	Measured rotationspeed of z_{BODY}-axis [rad/s]
P	State Covariance Matrix
Q	Process Covariance Matrix
R	Measurement Covariance Matrix
R_e	Equatorial radius of the Earth. 6379137 [m]
S_b, S_f	Classical Allan variance parameters.
t	Time [s]
v	Velocity [m/s]
v_D	Vertical velocity in Navigation Frame[m/s]
v_E	East velocity in Navigation Frame [m/s]
v_N	North velocity in Navigation Frame [m/s]
z	Measurement vector

Greek Symbols

Symbol	Explanation
α	Angular acceleration. $\alpha = \dot{\omega} = \ddot{\theta}$. [rad/$s^2$]
$\delta(u-v)$	Discrete Delta Function $\delta(0) = 1$ $\delta(n) = 0, n \neq 0$
δ_{ion}	Ionosphere time delay [s]
δ_{trop}	Troposhere time delay [s]
δ_u	Receiver clock bias error [s]
$\dot{\delta}_u$	Receiver clock drift [s/s]
$\Delta\rho_i$	Error in pseudorange to satellite i [m]
$\Delta\dot{\rho}_i$	Error in pseudorangerate to satelite i [m/s]
Δt	Time between samples [s]
λ	Longitude [rad]
λ	Wavelength [m]
ω	Angular speed. Here $\omega = [\dot{\phi}, \dot{\theta}, \dot{\psi}]$[rad/s]
$\dot{\Omega}_e$	The rotationrate of the Earth, 7.2921151467E-5 [rad/s]
ϕ	Latitude [rad]
ϕ	Roll with angle ϕ. [rad]
$\dot{\phi}$	Roll rate, in Eulerangle representation [rad/s]
Φ	Discrete time state transition matrix
Φ_c	Discrete time state transition matrix of GPS clock state
π	Mathematical constant pi≈3.1415926535898
ψ	Yaw angle. North is zero. ψ [rad]
$\dot{\psi}$	Yaw rotation rate . [rad/s]
ρ	Preudorange. Inline sight distance to the satellite. [m]
$\dot{\rho}$	Preudorange rate [m/s]
σ	Standard deviation
θ	Pitch angle. Elevation angle of vehicle. [rad]
$\dot{\theta}$	Pitch angle rate.[rad/s]

Abbreviations

Abbreviation	Explanation
AAV	Autonomous Aerial Vehicle
ADC	Analog to Digital Converter
ASCII	American Standard Code for Information Interchange
C/A	Course/Acquisition code
DAC	Digital to Analog Converter
DGPS	Differential GPS
DOP	Dilution of precision
ECEF	Earth-Centered Earth-Fixed Frame
ECI	Earth Centered Inerial Frame
EIF	Extended Information Filter
EKF	Extended Kalman Filter
FFT	Fast Fourier Transform
GDOP	Geometric DOP
GPS	Global Positioning System
IF	Information Filter
IMU	Inertial Measuring Unit
INS	Inertial Navigation System
KF	Kalman Filter
NAVSTAR	Navigation System with Timing and Ranging
OS	Operating System
PDOP	Position DOP
PPS	Precise Positioning Service
PSD	Power Spectrum Density
RMS	Root Mean Square
RPM	Rotations Per Minute
SA	Selective Availability
SPS	Standard Positioning Service
TAS	True Air Speed
TDOP	Time DOP
UAV	Unmanned Aerial Vehicle
UTC	Universal Coordinated Time
VDOP	Vertical DOP
WGS-84	World Geodetic System 1984
ZOH	Zero and Hold

Chapter 1

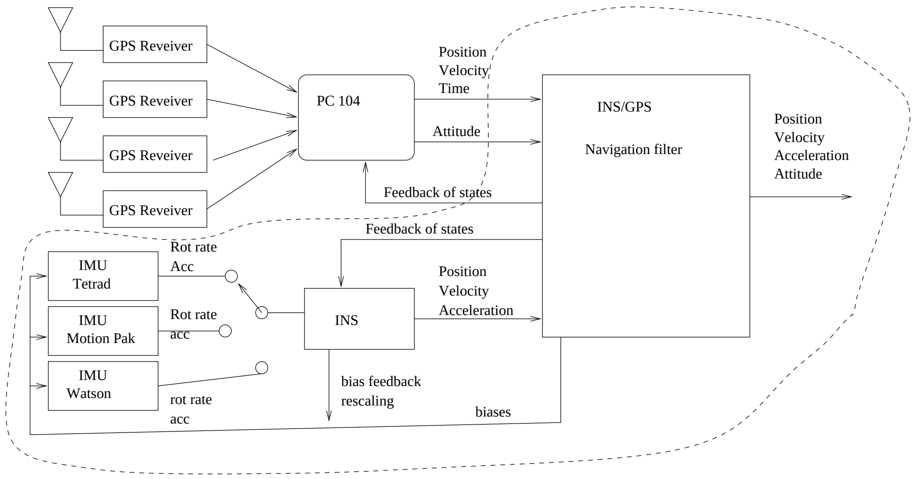
Introduction

1.1 Scope of this thesis

The main project was to implement a INS[1]/GPS[2] navigation filter that can be used on different vehicles, just by changing the model of the vehicle the filter will be optimized for a specific purpose. The error filter must accept all possible external observations. The INS/GPS navigation filter is written in C++ using a matrix library called MPP[3]. This thesis presents a navigation filter for an UAV[4].The filter is used to estimate the position, velocity and attittude of the vehicle. The navigation filter uses high frequency information given by an IMU[5] and uses low frequency external observations from GPS receivers to give a better estimation of the states.

1.1.1 Picture of the thesis work

The dashed area represents the work done in the thesis. The core of the work is the big block named "INS/GPS Navigation Filter". Here you can see feedback signal loops to other blocks.



In the figure we can see that the switch is on to use the Terad IMU. Three different IMUs have been used. Therefore a IMU software have been derived. The IMUs works all different so subclasses have been derived to handle what is unique with each one of the IMUs.

1.2 Background

The Department of Aeronautical Engineering at University of Sydney have a long history of building low-cost operating Unmanned Aerial Vehicles (UAVs).The UAV Research Group consists of about ten academics and research students. The Department of Aeronautical Engineering is now working together with the

[1] Inertial Navigation System
[2] Global Positioning System
[3] MPP-Matrix Plus Plus
[4] Unmanned Aerial Vehicle
[5] Inertial Measurement Unit

Deparment of Meachanical and Mechatronic Engineering to develop a new generation of UAV's. The airframe used is called *Brumby*, see 1.2.2 An previous UAV named Ariel has earlier been designed, manufactured and flown as a demostration platfrom for different aeronautical research activitis.

Brumby airframe with some of the researchers

Current UAV research activities at Sydney University

- Design studies on misson-specific UAVs.
- Micro UAVs.
- Design and fabrication of airframe components using advanced composite materials.
- Wind-tunnel and flight based experimental research in aerodynamics.
- Modelling of engine/propeller performance and aircraft stability characteristics.
- Aircraft model developement for simulation bases control system.
- Trajectory optimisation and autonomous guidance for self-piloted aircrafts.
- Sensor fusion strategies for state estimation using redundant sensors, including GPS.

- System identification using neural networks for fault detection.
- Robustness analysis of control laws in the presence of uncertain dynamics and wind gusts.
- Robust nonlinear high-performance tracking for autonomous aircrafts.
- Safe recovery of UAVs
- Flight control real-time software.

1.2.1 The ANSER project

It is a huge project to build an fully operating autonomous aircraft. Such projects must be broken down in small subprojects.

The ANSER sub projects

- Sensors
  - Functional specification
  - Radar sensors
  - IMU sensors
  - GPS
  - Vision
  - Laser
- Theory
  - Mission planing
  - Autonomy
  - Map building
- Control
- Mechanical
- Hardware
- Software
  - Flight software
  - Ground station software
  - Simulator software

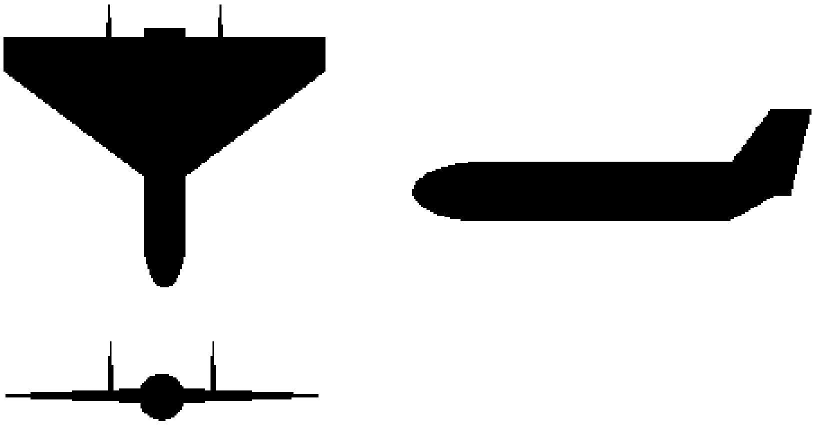
1.2.2 The Brumby UAV airframe

Brumby is an airframe designed at the deparment of Aeronautical Engineering at the University of Sydney. Brumby looks like a delta wing. This makes it a high performance aircraft but difficult to control. Especially landings can be tricky because the airframe will first drop in altitude when the pitch angle increases.The aircraft can fly approximately 45 minutes on full tank with standard two stroke petrol/oil mixture. The aircraft is remote controlled by a pilot on the ground. Brumby is collecting data during the flights used later for postprocessing. There is an onboard computer used for logging IMU, Video and GPS data to onboard mounted harddisks. Flights with logging to RAM disks have been done. The current operating system is Linux. Later it will be possible to switch over the control to a onboard computer that will fly Brumby autonomously and perform different tasks. If the radio connection is broken, Brumby will be able to land by itself.

The states of Brumby like position, velocity and attitude and other parameters will be transmitted to a Ground Control Station for visualization. A new and bigger version of the Brumby airframe is under manufacturing. The new airframe will be able to take more payload.

The design of Brumby is well planned. It is very easy to unload and load different payload. A special format for payload boxes exists and research groups all over the world can design their own payloads to be flown in Brumby.

The airframe seen from three directions



Data of Brumby airframe

The values inside the brackets are the data for the new airframe, Brumby MkII.

Airframe	**Value**
Wingspan	2.36 m (2.82 m)
Wing Area	1.61 m^2 (1.95 m^2)
Fuselage Length	1.97 m
Empty Weight (flying capable)	17 kg (19 kg)
Dry / Operational Empty Weight (OEW)	28 kg (33 kg)
Fuel Weight (2.4 liters)	1.9 kg
Useful Payload (sensors, flight control)	11 kg (14 kg)
Max Take Off Weight	30 kg (35 kg)
Engine	Zenoah 74cc Twin
Engine Power	Approx 5.2 kW (7 hp)
Max Speed	185 km/h
Landing Speed	65 km/h

1.3 The Future of UAVs

An idea is to let a number of unmanned airvehicles take off and perform tasks together, like surveillance. They can create maps of the surrounding terrain and find targets. If one aircraft finds a target, this information can be distributed to other UAV's in a near distance. The transmission can be done using laser or radio links. The UAV's can together be viewed as a flying decentralized system with flying nodes. If one UAV crashes only one node disappear from the flying network and almost all the information is kept within the other UAV's. The UAV's can in the future be equipped with different sensors, and all sorts of payload.

Interesting UAV applications

- Surveillance of important things and areas.
- Search for cars, lost people ...
- Emergency transports. Example: organs between hospitals.
- Rescue operations.
- Flying Shepherd Dog.
- Military tasks.

The military is very interested in these sort of vehicles, UAV's can easily in the future be used for different combat purposes.

Airbases now need people to patrol the fences, this can be done with UAV's. With good algorithms they can operate within big cities to prevent traffic jams and report traffic accidents and help ambulances through the traffic.

They can be used to track highspeed cars and criminals. Unmanned UAV can be helpful by monitoring the water status of lakes, protecting the rainforrests from illegal tree cutting, etc

With magnetic and gravity sensors as payload it is possible to indicate ore spots from the air. Airpollution can be tested near chimenys and oil spots can be found on the ocean.

1.4 Frames

A frame is the same as a coordinate system. When navigation is done we need atleast two frames. One for the body/intertial representation (vehicle) and one for the navigation frame representation (map).

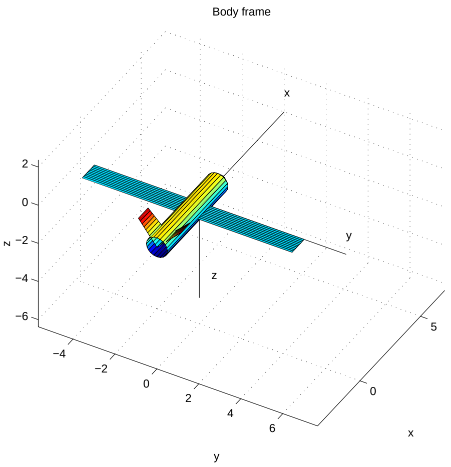
List of different frames

- Body frame
- Earth Surface NED Frame
- WGS-84 , World Geodetic System 1984
- ECI - Earth Centered Intertial Frame
- ECEF-Earth Centered Earth Fixed Frame.

1.4.1 Body Frame

This is the basic frame for the inertial sensors. The axis of this frame is in this case the same as the vehicle frame, and you can say that this is a reference frame carried by the vehicle. The x-axis is pointing forward, y-axis is pointing to the right and the z-axis completes a right-hand ortogonal system by pointing down and are ortogonal to both x and y-axis. The origin is at the center of gravity. The velocity of the vehicle in body frame is expressed using $[u, v, w]$.

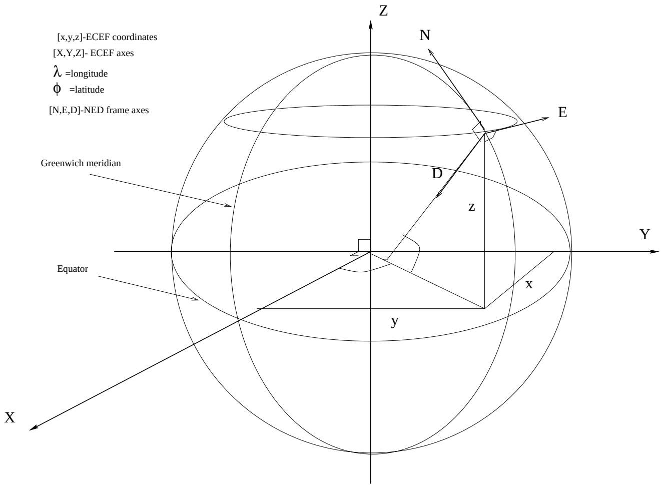
Body Frame



1.4.2 Earth Surface North-East-Down (NED) Frame

The NED-Frame is for navigation. We have three vectors that form a right hand ortogonal system. The N vector is pointing North, E vector is pointing East and D is pointing Down along the local gravity vector. The N and E vectors span a plane that tangent the surface of the Earth. Observe that vector D is mapped from the origin of this plane. This frame works fine when the operating distances are limited near the origin.

Earth NED frame



1.4.3 WGS-84 , World Geodetic System 1984

The geodetic frame reminds very much about spherical mapping and uses two angles and the height. It is good for navigation over longer distances. The sea level of the Earth is mapped if the height is put equal to zero.

Longitude,λ: At the Greenwich meridian the longitude is equal zero ($\lambda = 0$) and completes 360^o in the east direction. The longitude says how far to the east or west we are from the Greenwich meridian.

Latitude,ϕ: At the equator $\phi = 0$ and reaches $\phi = 90^o$ at the North Pole and $\phi = -90^o$ at the South Pole. The latitude says how far we are from the equator.

height, h: Height in meter above the Sealevel.

WGS-84 ellipsoid four defining parameters

These WGS-84 parameters defines the shape of the earth.

Parameter	**Notation**	**Magnitude**
Semimajor axis	a	6378137 m
Normalized Second Degree Zonal Harmonics Coefficient of the Gravitational Potential	$C_{2,0}$	$-484.16685E{-6}$
Angular Velocity of the Earth	$\dot{\Omega}_e$	$7292115.1467E - 10$ rad/s
Earth's Gravity Constant	μ	$3986005E8\ m^3/s^2$

From this parameters the geometric flattering of the earth and the semiminor axis can be derived.

1.4.4 ECI-Earth Centered Interial Frame

This frame is for inertial reference. The inertial frame is fixed with some of the axes pointing at distant stars. This frame does not follow the rotation of the earth, fixed directions of the axes in space.

Origin: The mass center of the Earth.

X-axis: The X-axis points to a distant star called The Vernal Equinox, this vector lies in the equatorial plane.

Y-axis: This axis spans the equatorial plane with the X-axis. The Y-axis forms a right handed ortogonal system together with the X and Z axes.

Z-Axis: Points from the origin out through the north pole. This axis is parallel to the rotation vector of the Earth.

1.4.5 ECEF - Earth Centered Earth Fixed Frame

This one is fixed with respect to the Earth and follows the rotation of the Earth. The ECEF frame is used in the GPS system called WGS-84. GPS observations are represented in this frame. This frame can also be called the Earth cartesian frame because of the cartesian coordianates $[x, y, z]_{ECEF}$. For more information about this frame read in [22] and [6].

Origin: Earth's center of mass.

X-axis: This axis pointing through the Greenwich meridian (longitude=0) and the equator. Intersection of the WGS-84 reference meridian plane and the plane of the mean astronomic equator, the reference meridian being parallel to the zero meridian.

Y-axis: Completes the right-handed ECEF ortogonal system.

Z-axis: This axis is pointing from origin out through the North Pole. Parallel to the direction of the COVENTIONAL INTERNATIONAL ORIGIN (CIO) for polar motion. Same as the Z_{ECI} axis.

Chapter 2

Attitude Representation

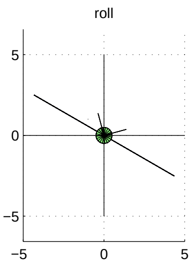
2.1 Rotations

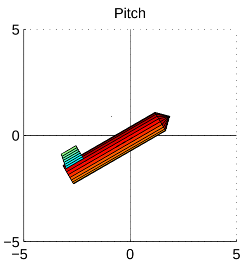
This chapter will explain the three different rotations of an vehicle. These rotations are roll, pitch and yaw. This chapter will also explain the different angle representations used. The euler angle, C_{bn} matrix and the quaternion angle representation.

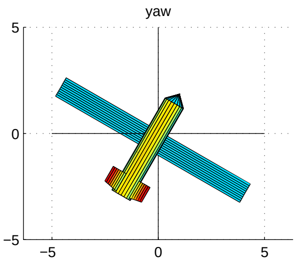
We can be an observer in the navigation frame (standing on the ground) and observe the airvehicle. What is the acceleration and the rotation rate of the object that are under observation ?. This is the information needed to do some navigation.

In the airframe we have a mounted strapdown[1] The IMU measures the acceleration $[a_x, a_y, a_z]$ and rotation rates $[p, q, r]$. We need to transform these vectors down to the navigation frame. The transformation matrix used is the C_{bn} matrix. $_{bn}$ indicates rotation of a vector from body frame to navigation frame. If we use this matrix in body frame we will change the coordinate system from body frame to navigation frame. The rotation is done with three angles, called the eulerangles $[\phi, \theta, \psi]$. In aeronautics (reference [10]) these angles are called roll, pitch and yaw. The rotations are done with respect to the body frame.

Roll Pitch and Yaw







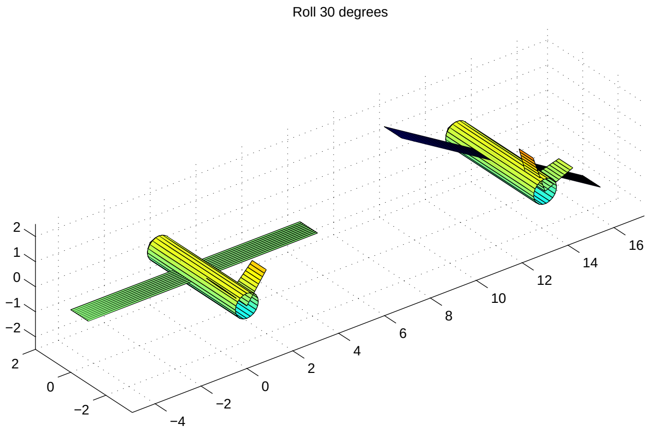
[1] This means that the inertial sensors are rigid mounted in the vehicle frame and there exists no moving parts. The calibration is done with pure mathematic operations

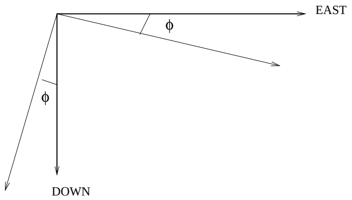
2.1.1 Roll rotation,ϕ

This rotation represents wingtips up/down. This is identical with rotation about the center-line of the airframe. A positive angle will make a clockwise rotation. When $\phi = 0$ the wings are in horizontal position.

3D visualization of 30^o roll

The aircraft tips the wings to 30^o.





Rotation with angle ϕ about x-axis.

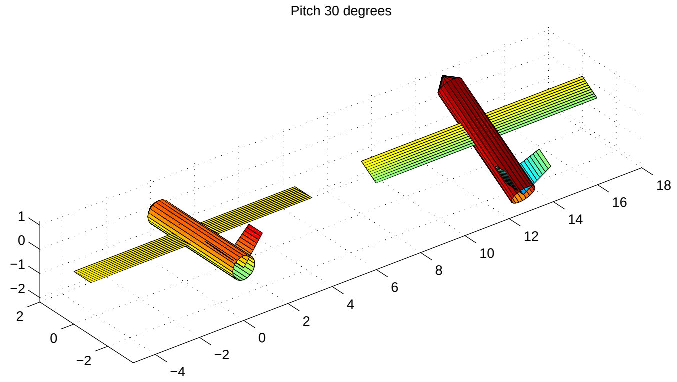
$$R(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{2.1}$$

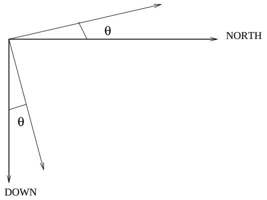
2.1.2 Pitch,θ

In aeronautics this is airframe nose up/down. When the vehicle is in horizontal position the pitch angle is zero, $\theta = 0$.

3D visualization of 30^o pitch

The airframe to the left in the figure below changes the pitch to 30^o.





A more mathematical definition is the rotation with angle θ about y-axis.

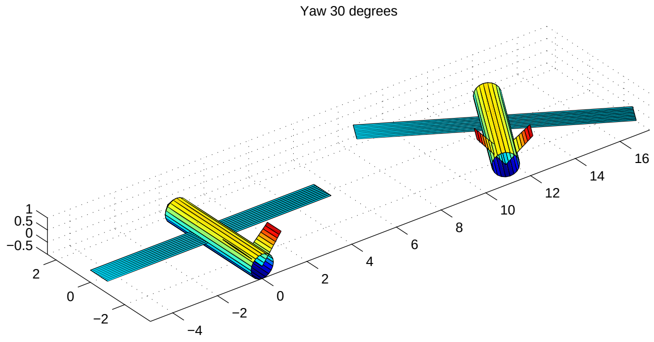
$$R(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.2}$$

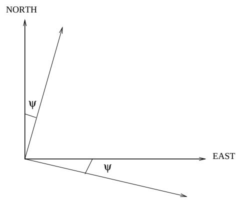
2.1.3 Yaw,Heading ψ

Yaw rotation of the airframe, nose left/right. The yaw angle represents the rotation of the airframe about the gravity vector. The heading is the direction of travel in navigation frame. The heading and yaw angle are almost the same and differ if the vehicle has a sideslip component. This sideslip can be caused by sidewinds or by the dynamics of the airframe. The airframe starts poiting north with the yaw angle equal zero, $\psi = 0$. Rotation is done clockwise around the gravity vector.

3D visualization of 30^o yaw

Yaw rotation of an airvehicle, the airvehicle changes the heading with 30^o.





Another definition of the yaw angle is rotation with angle ψ about z-axis.

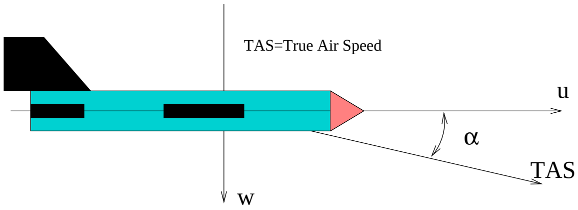
$$R(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

2.1.4 True attack and attack angle

If the direction of the flight doesn't follow the pitch angle we have a small difference between the true air speed and the pitch angle. The sideslip angle is called α. We can calculate the sideslip with equation (2.4).

$$\alpha = \theta - \arctan\left(\frac{w}{\sqrt{u^2 + v^2}}\right) \tag{2.4}$$

True attack

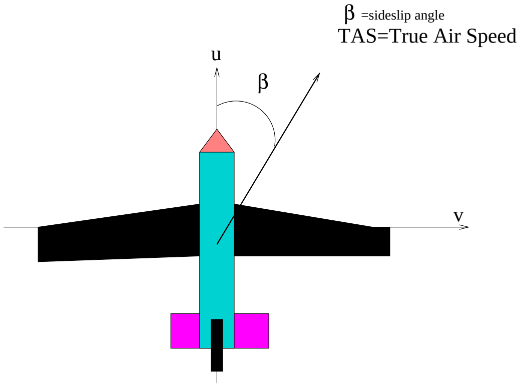


2.1.5 Sideslip

If the heading of the aircraft in navigationframe differ from the yaw angle, we have a sideslip. This sideslip is often caused by winds or by the dynamics of the airframe. The β angle is called the sideslip angle and are the difference between the heading and the yaw angle.

$$\beta = \psi - heading = \psi - \arctan\left(\frac{v}{u}\right) \tag{2.5}$$

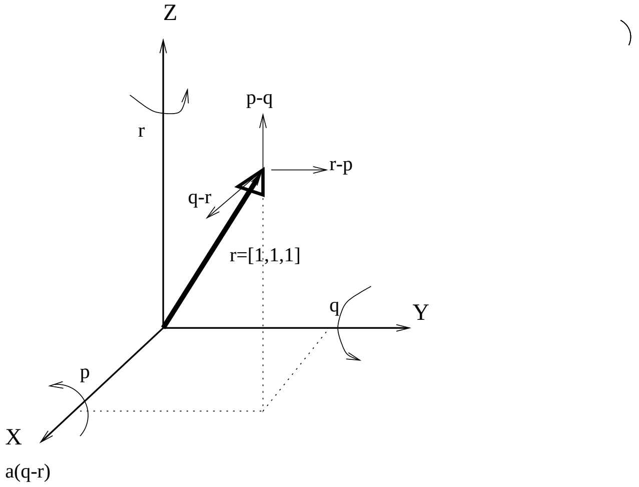
Sideslip



2.2 Body Rotation Rate

The rotation rate measured by the IMU in bodyframe is $[p, q, r]^T$. By taking the crossproduct with the eigenvectors in the bodyframe gives us the rotationrate in matrix representation. If we have a vector $[1, 1, 1]^T$ the resulting crossproduct with $[p, q, r]^T$ will be $[q - r, r - p, p - q]^T$. The vector $[1, 1, 1]^T$ is a linear combination of the base vectors $[1, 0, 0]^T$,$[0, 1, 0]^T$ and $[0, 0, 1]^T$ these can be put as a I^{3x3} matrix and the crossproduct with the rotationrate will result in a skewsymetric matrix, equation (2.6). In the figure below we have the vector $[1, 1, 1]^T$,the rotation rates $[p, q, r]^T$ about X,Y and Z-axis and the resulting changerate vector.

Frame vectors crossproducted with $[p, q, r]^T$



From the figure we see that we have a resulting velocity vector. This can be put into a skewsymmetric matrix.

$$PQR(p,q,r) = I^{3x3} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{2.6}$$

An interesting thing to do is to write the changes as a differential equation

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -p \\ 0 & p & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.7}$$

The solution to this equation is

$$\begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \exp\left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -p \\ 0 & p & 0 \end{bmatrix} t \right) + constant \tag{2.8}$$

If we take a short time Δt, this gives us the angle $\phi = \Delta tp$.

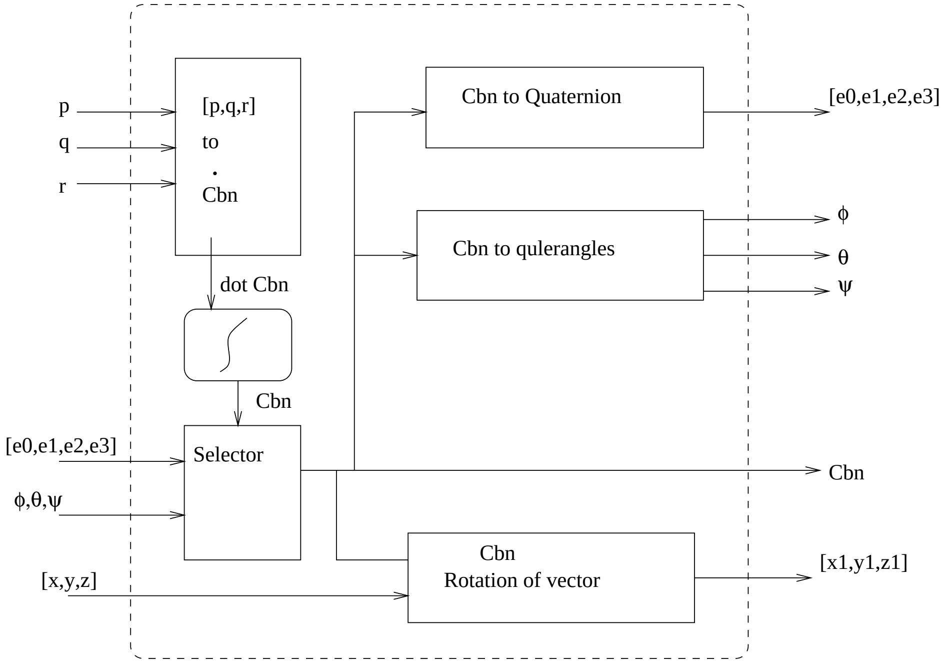
$$\exp\left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\phi \\ 0 & \phi & 0 \end{bmatrix}\right) = I^{3x3} + \sum_{k=1}^{\infty} \frac{1}{k!} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\phi \\ 0 & \phi & 0 \end{bmatrix}^k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.9}$$

Which inded is a very interesting result. The same procedure can be repeated for both the q and r rotationrates and we end up with the rotation matrixes for all three directions.

2.3 C_{bn}, Direction Cosine Representation

In this representation the angle are kept within the C_{bn} matrix. The rotationrates are used to update the matrix. If we want the attitude the angles can be calculated from the elements of the C_{bn} matrix.

Block schematics of C_{bn} matrix representation



The roll, pitch and yaw rotations can be put in a sequence, the result will be a rotationmatrix that rotates a vector from navigationframe to bodyframe with the angles $[\phi, \theta, \psi]$. This rotation matrix is called the C_{bn} matrix and is defined in equation (2.10).

$$C_{bn} = R(\phi, \theta, \psi) = R_\phi R_\theta R_\psi \tag{2.10}$$

$$C_{bn} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{2.11}$$

The C_{bn} matrix is a 3x3 ortogonal matrix with a total of nine elements. It have the property that $detC_{bn} = 1$ which means that the volume is preserved and no rescaling is done. The rows and columns of the C_{bn} matrix are ortogonal to each other.

$$C_{bn}C_{bn}^T = I^{3x3} \tag{2.12}$$

This matrix is possible to invert for all angles. If we write out all terms using trigonometric functions we get (2.13).

$$C_{bn}(\phi, \theta, \psi) = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \cos\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{2.13}$$

2.3.1 The C_{bn} matrix and small angles

When small angles are used we can use first order Taylor expansion of the trigonometric functions in the C_{bn} matrix, $\cos\alpha \approx 1$ and $\sin\alpha \approx \alpha$. We then get a skewsymetric rotationmatrix, equation (2.14).

$$C_{bn}(\phi, \theta, \psi) = I^{3x3} + I^{3x3} \times \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 1 & -\phi & \theta \\ \phi & 1 & -\psi \\ -\theta & \psi & 1 \end{bmatrix} \tag{2.14}$$

2.3.2 Conversion to Eulerangles from C_{bn} represention

If the C_{bn} matrix is given, the eulerangles can easily be calculated with equations (2.15),(2.16) and (2.17). These equations is derived from (2.13). The c_{ij} variables are found in the C_{bn} matrix.

$$\phi = \arctan\left(\frac{c_{32}}{c_{33}}\right) = \arctan\left(\frac{\sin\phi\cos\theta}{\cos\phi\cos\theta}\right) \tag{2.15}$$

$$\theta = \arcsin\left(-c_{31}\right) = \arcsin\left(\sin\theta\right) \tag{2.16}$$

$$\psi = \arctan\left(\frac{c_{21}}{c_{11}}\right) = \arctan\left(\frac{\cos\theta\sin\psi}{\cos\theta\cos\psi}\right) \tag{2.17}$$

2.3.3 Calculation of Eulerangles with large pitch angle

If we have a C_{bn} matrix with a pitch angle such that $\cos\theta \approx 0$ we can not use equations (2.15),(2.16) and (2.17) because the equations are not numerical stable. There exits other equations for this case. In [37] we find the equations (2.18) to (2.21). The c_{ij} variables are elements in the C_{bn} matrix. i=row, j=column.

$$c_{23} - c_{12} = (\sin\theta + 1)\sin(\psi - \phi) \tag{2.18}$$

$$c_{13} + c_{22} = (\sin\theta + 1)\cos(\psi - \phi) \tag{2.19}$$

$$c_{23} + c_{12} = (\sin\theta - 1)\sin(\psi + \phi) \tag{2.20}$$

$$c_{13} - c_{22} = (\sin\theta - 1)\cos(\psi + \phi) \tag{2.21}$$

Equations (2.20), (2.21), (2.18) and (2.19) have three unknown variables, this equation system can be solved. If we divide equation (2.18) with (2.19) we get (2.23), (2.18) and (2.19) gives (2.22).

$$\psi = \arctan\left(\frac{c_{23} + c_{12}}{c_{13} - c_{22}}\right) \tag{2.22}$$

$$\psi - \phi = \arctan\left(\frac{c_{23} - c_{12}}{c_{13} + c_{22}}\right) \tag{2.23}$$

By combining (2.22) and (2.23) we get:

$$\phi = \arctan\left(\frac{c_{23} - c_{12}}{c_{13} + c_{22}}\right) - \arctan\left(\frac{c_{23} - c_{12}}{c_{13} + c_{22}}\right) \tag{2.24}$$

The θ angle is calculated with the previous equation (2.16).

$$\theta = \arcsin(-c_{31}) = \arcsin(\sin\theta) \tag{2.25}$$

2.3.4 The changerate of C_{bn} matrix

We know how to calculate the rotationrate matrix, lets calculate the changerate of the C_{bn} matrix . The C_{bn} matrix have bodyframe as reference frame because the rotation is always done relative to the airframe. The $[p, q, r]$ vector represents the rotationrate about each axis. It is shown in 2.2 that the crossproduct can be written as a skewsymmetric matrix. The C_{bn} matrix is multiplied with (2.6) and this gives us an expression to calculate the changerate of the C_{bn} matrix.

$$\dot{C}_{bn} = C_{bn}\begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{2.26}$$

We get the C_{bn} matrix by integration

$$C_{bn} = \int \dot{C}_{bn}\mathrm{dt} = \int \mathrm{C_{bn}}\begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}\mathrm{dt} \tag{2.27}$$

2.3.5 C_{bn} matrix integration in discrete time

Equation (2.27) is a continous time integrator that we need to discretisize for software implementation. The change the C_{bn} matrix will do under a short time $\Delta t = t_{k+1} - t_k$ is $\dot{C}_{bn}\Delta T$. This change must be added to the actual $C_{bn}(k)$ matrix. A simple ZOH integrator is

$$C_{bn}(k+1) = C_{bn}(k) + \dot{C}_{bn}(k)\Delta T = C_{bn}(k) \begin{bmatrix} 1 & -r & q \\ r & 1 & -p \\ -q & p & 1 \end{bmatrix} \Delta T \qquad (2.28)$$

2.3.6 Very accurate C_{bn} matrix integration in discrete time

To make the update of the C_{bn} matrix more accurate we can use the Taylor expansion presented in [37]. The idea is to expand the rototationrate matrix, e^A using $\sigma = \sqrt{p^2 + q^2 + r^2}$ which lead to equation (2.29)

$$e^A = I^{3x3} + \frac{\sin\sigma}{\sigma}A + \frac{1-\cos\sigma}{\sigma^2}A^2 \qquad (2.29)$$

$$A = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \Delta T \qquad (2.30)$$

$$A^2 = \begin{bmatrix} -(q^2+r^2) & pq & pr \\ pq & -(p^2+r^2) & qr \\ pr & qr & -(q^2+r^2) \end{bmatrix} \Delta T^2 \qquad (2.31)$$

The C_{bn} matrix update will be

$$C_{bn}(k+1) = C_{bn}(k) \left(I^{3x3} + \frac{\sin\sigma}{\sigma}A + \frac{1-\cos\sigma}{\sigma^2}A^2 \right) \qquad (2.32)$$

2.3.7 Normalization of C_{bn} matrix

If the C_{bn} matrix is updated (rotated) alot this introduces rescaling of the rows vectors. The row vector will after a while not be ortogonal to each other. This can be caused by the floating point aritmetic in a modern computer. Lets say we have a $\tilde{C}_{bn}$ matrix who $\tilde{C}_{bn}\tilde{C}_{bn}^T <> I^{3x3}$.

The matrix can be normalized using (2.33).

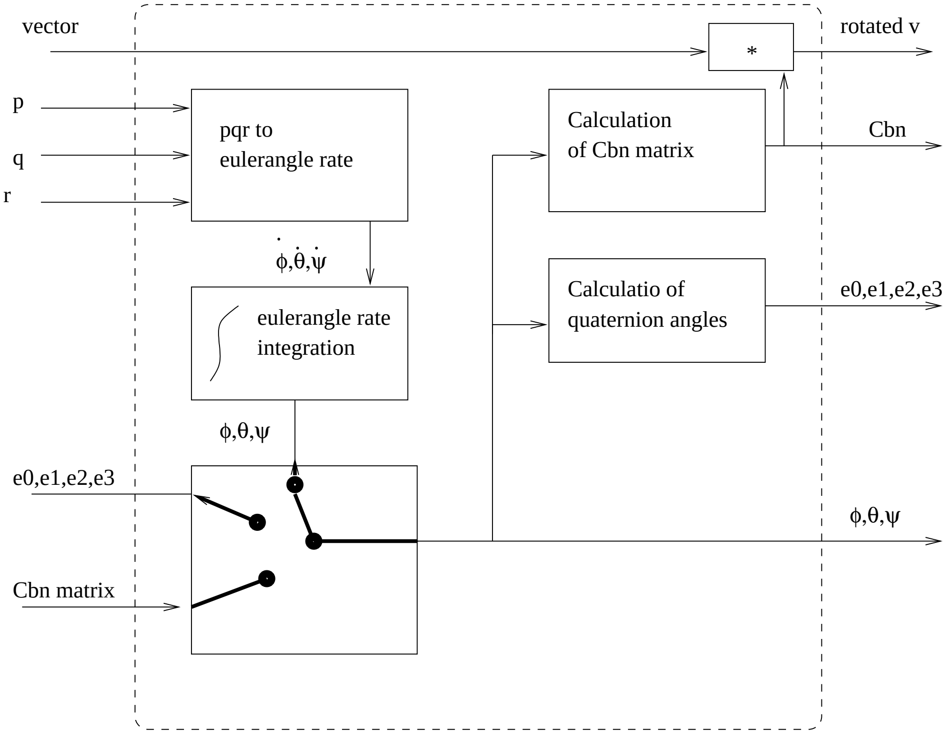
$$C_{bn} = \tilde{C}_{bn} - 0.5 \left(\tilde{C}_{bn}\tilde{C}_{bn}^T \right) \tilde{C}_{bn} \qquad (2.33)$$

For more information read [37]

2.4 Eulerangle Representation

In eulerangle representation the attitude are kept in a vector $[\phi, \theta\psi]$. The rotationrates $[p, q, r]$ must update the $[\phi, \theta, \psi]$ vector. This is done with an integrator.

Block schematics of Eulerangle representation



2.4.1 Calculation of $[\dot{\phi}, \dot{\theta}, \dot{\psi}]$ using [p,q,r]

The elements in the C_{bn} matrix can be calculated if we know the eulerangles. By knowing this we can calculate the changerate of the eulerangles with the rotationrates as arguments. We can transform rotationrates $[p, q, r]_{BODYFRAME}$ to navigationframe and then we will get the anglerates of the eulerangles $[\dot{\phi}, \dot{\theta}, \dot{\psi}]$. Because the rotation using eulerangles are done in sequence $R(\phi)R(\theta)R(\psi)$ we must have this in mind when we want to calculate $[\dot{\phi}, \dot{\theta}, \dot{\psi}]$. We put this into a equation system and get (2.34).

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R(\phi) \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R(\phi)R(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \tag{2.34}$$

This equation can be written as

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & -\sin\theta & 0 \\ 0 & \cos\theta & \cos\theta\sin\phi \\ -\sin\phi & 0 & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.35}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = C_{pqr|\dot\phi\dot\theta\dot\psi} \begin{bmatrix} \dot\phi \\ \dot\theta \\ \dot\psi \end{bmatrix} \tag{2.36}$$

To solve $[\dot\phi, \dot\theta, \dot\psi]^T$ we take the inverse of $C_{pqr|\dot\phi\dot\theta\dot\psi} = C^{-1}_{\dot\phi\dot\theta\dot\psi|pqr}$. The $C_{\dot\phi\dot\theta\dot\psi|pqr}$ transformation matrix will be :

$$C_{\dot\phi\dot\theta\dot\psi|pqr} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \tag{2.37}$$

So the euleranglerates can be calculated using equation (2.38).

$$\begin{bmatrix} \dot\phi \\ \dot\theta \\ \dot\psi \end{bmatrix} = C_{\dot\phi\dot\theta\dot\psi|pqr} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.38}$$

$$\begin{bmatrix} \dot\phi \\ \dot\theta \\ \dot\psi \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.39}$$

There is a problem with the $C_{\dot\phi\dot\theta\dot\psi|pqr}$ matrix. We have a singularity in the matrix, element c_{33}, when $\cos\theta = 0$ leads to $\sec\theta = 1/\cos\theta = undefined$. The singularity in a strapdown system is equivalent of gimbal lock. This occurs when the aircraft is flying with the nose straight up or straight down. Because the euleranglerate has singularities it is bad angle representation in high performance aircrafts like fighter aeroplanes. This three parameter euler algoritm are normally used in system with bounded pitch angle, between $+-30^o$. We have other models to bypass this, both the C_{bn} matrix and the Quaternionangle representation supports this.

Equation (2.39) is not defined when $\sec\theta$ becomes singular, and this can be bypassed by these equations found in [23]. The equations have not been implemented in the software at all, but is an interesting thing to investigate.

$$\begin{aligned} &\dot\phi = p + \dot\psi\sin\theta \\ &\dot\theta = q\cos\phi - r\sin\phi \\ &\dot\psi = (\dot\phi - p)\sin\theta + (q\sin\phi + r\cos\phi)\cos\theta \end{aligned} \tag{2.40}$$

2.4.2 Eulerangle integration in a discrete time system

In a navigation system we need to keep track of the eulerangles. The rate gyros in the IMU[2] measures the rotationrates of the airframe which are transformed to euleranglerates. To update eulerangles we run the euleranglerates trough an ZOH integrator.

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}_{k+1} \approx \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}_{k} + \begin{bmatrix} \dot\phi \\ \dot\theta \\ \dot\psi \end{bmatrix} \Delta T \tag{2.41}$$

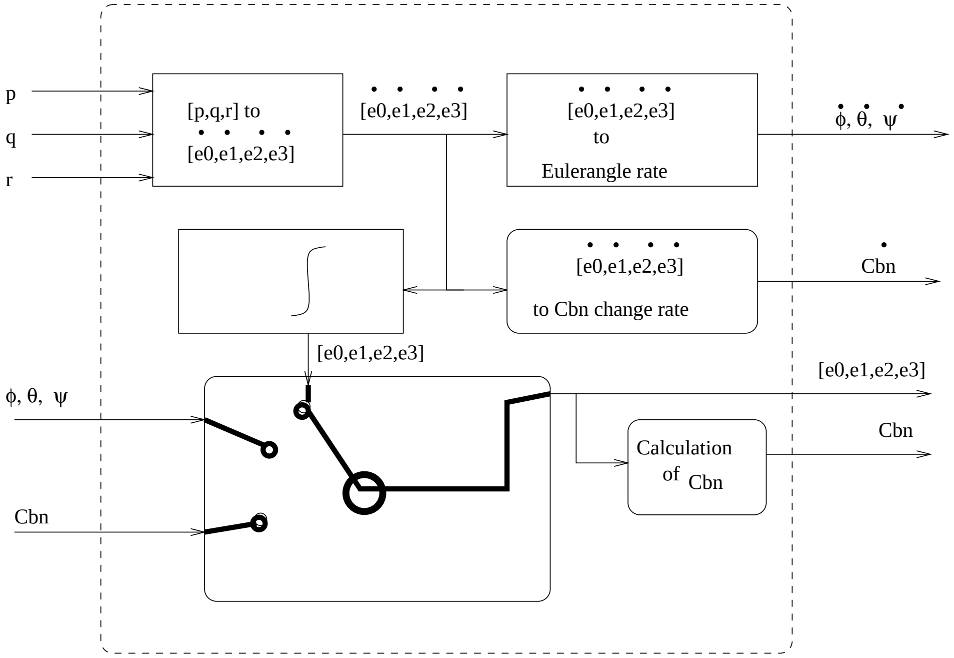
[2] Inertial Measuring Unit

This model is very fast but it's weakness is that the error is of order $O(h^2)$, with an IMU samplerate of 300Hz and a 2π makes this a very bad model. However let's not cry, there exists more fancy integrators. In the software several different discrete integration methods were tested, the best one implemented is a four order Rungekutta integrator with an error of $O(h^5)$. Simpson, and Trapetz integration methods have been tested and implemented and are good methods if highspeed integration is needed.

2.5 Quaternion Angle Representation

The idea with this method is to use four variables for rotation in R^3 instead of three. It can also be shown that this system with quaternions is valid for all attitudes. The $[p, q, r]$ rotationrates are used to update the quaternion angle.

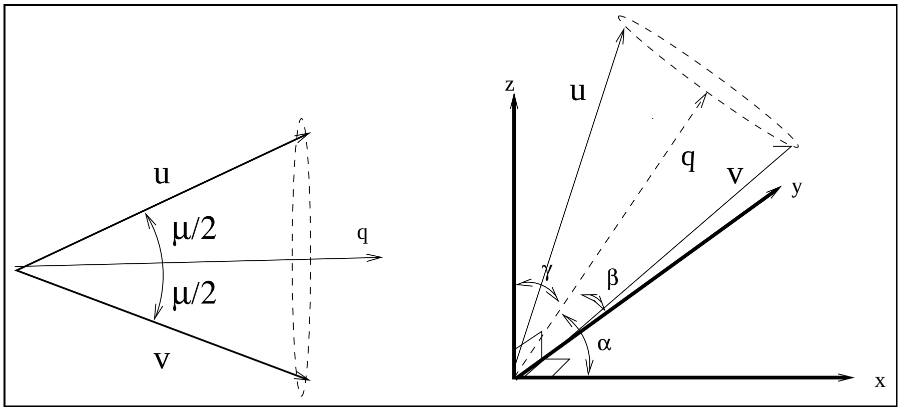
Block schematics of quaternion angle representation



From this stage quaternion angles are viewed as:

$$E = \left[\begin{array}{cccc} e_0 & e_1 & e_2 & e_3 \end{array} \right] \quad (2.42)$$

Let $\mathbf{u}$ be the vector before a single R^3 rotation about vector $\mathbf{q}$ and $\mathbf{v}$ the resulting vector. Then μ is defined as the angle between the two vectors. So $\mu = \arccos\left(\frac{\mathbf{u}\times\mathbf{v}}{|\mathbf{u}||\mathbf{v}|}\right)$.



The **q** axis makes the angles $\cos^{-1}\alpha$, $\cos^{-1}\beta$ and $\cos^{-1}\gamma$ with the inertial axes x,y and z, see figure above. The equations to calculate the the E vector elements are:

$$\begin{aligned} e_0 &= \cos(\mu/2) \\ e_1 &= \alpha\sin(\mu/2) \\ e_2 &= \beta\sin(\mu/2) \\ e_3 &= \gamma\sin(\mu/2) \end{aligned} \tag{2.43}$$

It can be shown that the quaternionangle vector elements can be used to calculate the C_{bn} matrix, the equation (2.44) is found in [37].

$$C_{bn} = \begin{bmatrix} (e_0^2+e_1^2-e_2^2-e_3^2) & 2(e_1e_2-e_0e_3) & 2(e_1e_3+e_0e_2) \\ 2(e_1e_2+e_0e_3) & (e_0^2-e_1^2+e_2^2-e_3^2) & 2(e_2e_3-e_0e_1) \\ 2(e_1e_3-e_0e_2) & 2(e_2e_3+e_0e_1) & (e_0^2-e_1^2-e_2^2+e_3^2) \end{bmatrix} \tag{2.44}$$

2.5.1 Calculation of Eulerangles using Quaterion angles

The Eulerangles can be calculated from equation (2.44) using the known C_{bn} relation, see equations (2.13) .

$$\begin{aligned} \phi &= \arctan\left(\tfrac{c_{32}}{c_{33}}\right) = \arctan\left(\tfrac{2(e_0e_1+e_2e_3)}{e_0^2-e_1^2-e_2^2+e_3^2}\right) \\ \theta &= \arcsin\left(-c_{31}\right) = \arcsin\left(2(e_0e_2-e_3e_1)\right) \\ \psi &= \arctan\left(\tfrac{c_{21}}{c_{11}}\right) = \arctan\left(\tfrac{2(e_0e_3+e_1e_2)}{e_0^2+e_1^2-e_2^2-e_3^2}\right) \end{aligned} \tag{2.45}$$

2.5.2 Quaternionangle calculation using Euler representation

If we do not have the C_{bn} matrix we can use the eulerangles $[\phi, \theta, \psi]$ to calculate the quaternion vector elements using equations (2.46),(2.47), (2.48) and (2.49).

$$e_0 = \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \tag{2.46}$$

$$e_1 = \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \tag{2.47}$$

$$e_2 = \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \tag{2.48}$$

$$e_3 = \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \tag{2.49}$$

After that the vector elements have been calculated the quaterionangle vector need to be normalized, $e_0^2 + e_1^2 + e_2^2 + e_3^2 + e_4^2 = 1$ should always be true.

2.5.3 Calculate Quaternion using the C_{bn} matrix representation

Using equation (2.44) we can calculate the quaternionangle vector elements.

$$\begin{array}{l} e_0 = \frac{1}{2}\sqrt{1 + c_{11} + c_{22} + c_{33}} = \frac{1}{2}\sqrt{4e_0^2} \\ e_1 = \frac{1}{4e_0}(c_{32} - c_{23}) = \frac{1}{4e_0}(2(e_2e_3 + e_0e_1) - 2(e_2e_3 - e_0e_1)) \\ e_2 = \frac{1}{4e_0}(c_{13} - c_{31}) = \frac{1}{4e_0}(2(e_1e_3 + e_0e_2) - 2(e_1e_3 - e_0e_2)) \\ e_3 = \frac{1}{4e_0}(c_{21} - c_{12}) = \frac{1}{4e_0}(2(e_1e_2 + e_0e_3) - 2(e_1e_2 - e_0e_3)) \end{array} \tag{2.50}$$

2.5.4 Quaternion Norm

The L_2 norm of the quaternion angle vector should always be one and is used to correct for accumulated computational errors.

$$1 = |E| = \sqrt{e_0^2 + e_1^2 + e_2^2 + e_3^2} \tag{2.51}$$

If a quaternionangle vector $\tilde{E}$ doesn't have the right L_2 norm it can be normalized using equation (2.52). If we look at equation (2.45) we see that the eulerangles are calculated using all elements of the quaternion angle vector. This can introduce some unwanted effects. If one of the quaternion elements have a small error this will affect all the other elements when the normalization is done. And say if we are using eulerangles and have an error in pitch, this error will contribute to all quaternion angle elements. When the vector is normalized the error is spread to both roll and yaw.

Quaternionangle vector normalization

$$E = \frac{\tilde{E}}{|\tilde{E}|} = \frac{\tilde{E}}{\sqrt{\tilde{e}_0^2 + \tilde{e}_1^2 + \tilde{e}_2^2 + \tilde{e}_3^2}} \tag{2.52}$$

2.5.5 Quaternationangle change rate

The components denoted $[p, q, r]$ are the rotationrates in body frame. The relation between quaternion angle rate and $[p, q, r]$ can be found in [37] and is:

$$\begin{bmatrix} \dot{e}_0 \\ \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} -e_1 & -e_2 & -e_3 \\ e_0 & -e_3 & e_2 \\ e_3 & e_0 & -e_1 \\ e_2 & e_1 & e_0 \end{bmatrix}\begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.53}$$

The equation is continous in time and is a system of differential equations. This should be implemented in a computer system and we prefer to keep the quaternion angle in vector form. So we do the following rewriting of equation (2.53).

$$\begin{bmatrix} \dot{e}_0 \\ \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} = \dot{E} = A(p,q,r)E \qquad (2.54)$$

We know the angle at all time if we integrate (2.54).

$$E(t) = \int \dot{E}(p(t), q(t), r(t))dt \qquad (2.55)$$

2.5.6 Quaternionangle update in discrete time

The previos section lead us to equation (2.55). This equation must however be discrete to fit in a computer system, hence

$$E_{n+1} = E_n + \int_{t_n}^{t_{n+1}} \dot{E}_n dt = E_n + \int_{t_n}^{t_{n+1}} A(p,q,r)E_n dt, n \in \{0..N\} \qquad (2.56)$$

With $\Delta t = t_{n+1} - t_n$, and assuming that p,q and r are almost constant between t_{n+1} and t_n. Equation (2.56) can be implemented using a ZOH integrator.

$$E_{n+1} = E_n + \dot{E}_n \Delta t = (I + A\Delta t)E_n \qquad (2.57)$$

Equation (2.57) is identical to :

$$\begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix}_{n+1} = \frac{1}{2} \begin{bmatrix} 1 & -p\Delta t & -q\Delta t & -r\Delta t \\ p\Delta t & 1 & r\Delta t & -q\Delta t \\ q\Delta t & -r\Delta t & 1 & p\Delta t \\ r\Delta t & q\Delta t & -p\Delta t & 1 \end{bmatrix} \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix}_n \qquad (2.58)$$

This is a first order integrator. If a more accurate integration is needed we can use the same method as in section 2.3.6. A model of this can be derived for quaternions, read more about this in [37].

Chapter 3

Inertial Sensors, IMU and INS

3.1 Newtonian/Inertial Sensors

They can measure newtonian/inertial values like forces, and are called inertial sensors. Different sensors exists to especially measure acceleration and rotation-rates. It is prefered to have very accurate and sensitvte sensors when we want to measure acceleration. The error in position will grow with a square factor in time with respect to the error in acceleration, shown in equation (3.1).

$$\Delta\mathbf{p} = \int\int \Delta\mathbf{a}\,\mathrm{dtdt} = \mathrm{O}(\mathrm{t}^2) \tag{3.1}$$

3.1.1 Sensor Termal Compensation

A sensor, especially a gyro, is strongly affected by the surrounding temperature. An IMU that is not temperature stabilized or temperature compensated is almost useless. Even if the temperature is monitored and we try to keep it at a desired temperature will there be fluctation. In order to get a high performing IMU/INS system we must have a mathematical model of how the temperature affects the measurements from the inertial sensors. Equation 3.2 represents a good model.

Variable	**Explanation**
t	Time [s]
λ	Self-Heat time constant. $[s]$
T	Actual temperature [K]
T_{ref}	Reference temperature. [K]
B_0, B_1, B_2	Calibration constants. $[1, K^{-1}, K^{-2}]$

$$S(t,T) = B_0 \exp\left(\frac{-t}{\lambda}\right) + B_1(T - T_{ref}) + B_2(T - T_{ref})^2 \tag{3.2}$$

When a sensor is switched on it takes a while before the temperature reaches equlibrium or working point. The time it takes for the sensor to reach this working point is unique for every sensor and the enviroment it is placed in. The variable $\lambda, T_{ref}, B_0, B_1 and B_2$ must be calculated from tests.

3.1.2 Newtonian Sensor Nonlinearity

All sensors are unique. So therefore they will give different outputs even if they are of the same type and placed in the same enviroment. A lot of sensors are delivered with calibration data and different terms to put into calibration equations. This is normally the case for very expensive and accurate sensors. If a sensor is bought and no calibration paper is submitted with the sensor it is normally a low cost sensor that only have standard parameters given. The output from a newtonian sensor is often an analog signal. This signal contains a bias 'b' we need to know or calculate. A sensor work in a special range, an ideal sensor is linear in this region. Some sensors are almost linear and no effort are done in calculating the nonlinear terms. A very common and simple model to improve a sensor is to use equation (3.3).

$$m = SF^{-1}\,\mathbf{n}\cdot\mathbf{v} + \mathrm{b} \tag{3.3}$$

This is rearranged and we get :

$$\mathbf{n}\cdot\mathbf{v} = \mathrm{sf}(1+\mathrm{sf_e})(\mathrm{m}-\mathrm{b}) \tag{3.4}$$

If we have a strong nonlinearity then sf_e can be made to a function, with the measured value m as argument, $sf_e(m)$.

Variable	Explanation
m	Measured value
b	Measured sensor bias
sf_e	Scalefactor error
sf	Scalefactor
SF^{-1}	1/(sf(1+sf_e))
$\mathbf{n}$	Sensor normalvector, $\lvert\mathbf{n}\rvert = 1$. Direction of sensitivity
$\mathbf{v}$	Real newtonian vector. We want to measure this one

3.2 Multi sensor unit

We have a sensor unit with lot of sensors of the same type. The sensors axes are pointing in different directions. To measure the vector $\mathbf{v}$ in R^3 we need atleast three sensors mounted such that the sensoraxes spans R^3, optimal is to place them ortogonal to each other.

The total scalefactor of sensor k is written as s_k, this includes both the re-scalefactor and the scalefactor error.

$$s_k = sf_k(1+sf_{e_k}) \tag{3.5}$$

We have a unit with three sensors with normalvectors (n_1, n_2 and n_3) that are linear independent.

$$\begin{bmatrix} \mathbf{n_1} \\ \mathbf{n_2} \\ \mathbf{n_3} \end{bmatrix} \mathbf{v} = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix} \begin{bmatrix} m_1 - b_1 \\ m_2 - b_2 \\ m_3 - b_3 \end{bmatrix} \tag{3.6}$$

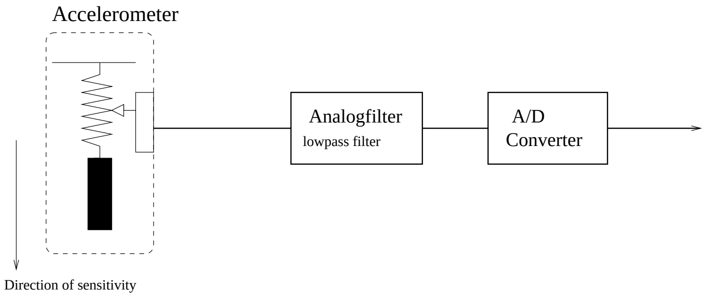
$$bfv = \begin{bmatrix} \mathbf{n_1} \\ \mathbf{n_2} \\ \mathbf{n_3} \end{bmatrix}^{-1} \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix} \begin{bmatrix} m_1 - b_1 \\ m_2 - b_2 \\ m_3 - b_3 \end{bmatrix} \tag{3.7}$$

3.2.1 The Accelerometer

This sensor can be modelled as a weight hanging in spring. A tension sensor is connected to the spring. The spring must be very strong to prevent oscillations. The force is linear to the acceleration acting on the mass, equation (3.8). The accelerometer have a direction of sensitivity, this direction unit vector is called $\mathbf{n}$.

$$\mathbf{F} = \mathrm{m}(\mathbf{n} \cdot \mathbf{a}) \tag{3.8}$$

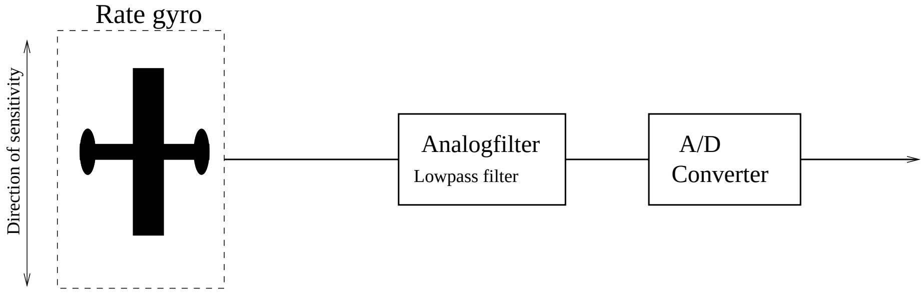
When the force is acting on the "spring" this will create a tension that can be measured. The analog signal is converted to digital signals by using an Analog to Digital Converter (ADC). Before this is done the signal must pass a lowpass analog filter. The cutfrequency should be at maximum the Nyquist frequency, half the sample rate.



3.2.2 The Rate Gyros

The standard old gyro is a rotating disc with a very high rotating speed. The gyro will try keep the direction of the rotating vector. If the direction will change there will be a momentum applied to the mount axis. With some sensors at the mounting points it is possible to measure the force and or the stress of the material. With this measurement you calculate the change rate of the rotating vector. With this it is possible to keep track of angles.

The angluar rate gyros of today use vibrating piezoelectric ceramic transducers.



3.2.3 Sensor Bias

Some of the sensors will drift in time, the bias or even the scalefactor. This is mainly caused by temperature changes. But in a gyro the bias will change over time caused by the rotation of the earth.

Gyro Bias Drift Example

Lets have a bias b_r in the rotation rate r measured by the IMU, we assume the bias to be small. The unit of b_r is $[rad/s]$. We get the increse in angle by integration.

$$\delta\theta = \int b_r r \, dt = b_r r \, t \tag{3.9}$$

This small angle will cause an misalignment of the IMU. The acceleration vector is projected wrong. We get an acceleration in the north direction, $a_n = sin(\theta) g_e \approx b_r g_e t$. This is integrated twice to get the error in position.

$$\delta p = \int \int b_r g_e t \, dtdt = b_r g_e (t^3/6 + C_1 t + C_2) = O(t^3) \tag{3.10}$$

We have an error in position that increases cubic in time when an bias error exists on the gyros.

To get a high performance IMU the biases of the gyros should be added to the navigation filter. This can be modelled with this simple two state model, where $\epsilon_{b_{GYRO}} \in N(0, \sigma_{b_{GYRO}})$

$$\begin{bmatrix} \dot{b}_1 \\ \dot{b}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \epsilon_{b_{GYRO}} \tag{3.11}$$

The b_1 parameter is the bias and b_2 is the bias change rate. The gyro bias is:

$$b_{GYRO} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{3.12}$$

3.2.4 Redundant Sensor Unit

If the we have more sensors than unknown dimensions we have a redundant sensor system. The sensor normalvectors are placed such that they together spans R^3. But they are not linear independent, and there will not exist a inverse to matrix $[\mathbf{n_1}, \mathbf{n_2}, \ldots, \mathbf{n_k}]^{\mathrm{T}}$, k>3. We have the following sensor system

$$\begin{bmatrix} \mathbf{n_1} \\ \mathbf{n_2} \\ \vdots \\ \mathbf{n_k} \end{bmatrix} \mathbf{v} = \begin{bmatrix} s_1 & 0 & \ldots & 0 \\ 0 & s_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & s_k \end{bmatrix} \begin{bmatrix} m_1 - b_1 \\ m_2 - b_2 \\ \vdots \\ m_k - b_k \end{bmatrix} \tag{3.13}$$

Equation (3.13) can be solved in least square norm. Assuming all sensor have the same measurement noise the best solution of $\mathbf{v}$ in L_2 norm is :

$$\tilde{\mathbf{v}} = \left(\begin{bmatrix} \mathbf{n_1} \\ \mathbf{n_2} \\ \vdots \\ \mathbf{n_k} \end{bmatrix}^T \begin{bmatrix} \mathbf{n_1} \\ \mathbf{n_2} \\ \vdots \\ \mathbf{n_k} \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{n_1} \\ \mathbf{n_2} \\ \vdots \\ \mathbf{n_k} \end{bmatrix}^T \begin{bmatrix} s_1 & 0 & \ldots & 0 \\ 0 & s_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & s_k \end{bmatrix} \begin{bmatrix} m_1 - b_1 \\ m_2 - b_2 \\ \vdots \\ m_k - b_k \end{bmatrix} \tag{3.14}$$

If we have different noise characteristics on the sensors this can be put in the measurement noise matrix $\mathbf{R}$. The optimal solution $\tilde{\mathbf{v}}$ will in this case become.

$$\tilde{\mathbf{v}} = \left(\begin{bmatrix} \mathbf{n_1} \\ \mathbf{n_2} \\ \vdots \\ \mathbf{n_k} \end{bmatrix}^T \mathbf{R^{-1}} \begin{bmatrix} \mathbf{n_1} \\ \mathbf{n_2} \\ \vdots \\ \mathbf{n_k} \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{n_1} \\ \mathbf{n_2} \\ \vdots \\ \mathbf{n_k} \end{bmatrix}^T \mathbf{R^{-1}} \begin{bmatrix} s_1 & 0 & \dots & 0 \\ 0 & s_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & s_k \end{bmatrix} \begin{bmatrix} m_1 - b_1 \\ m_2 - b_2 \\ \vdots \\ m_k - b_k \end{bmatrix} \tag{3.15}$$

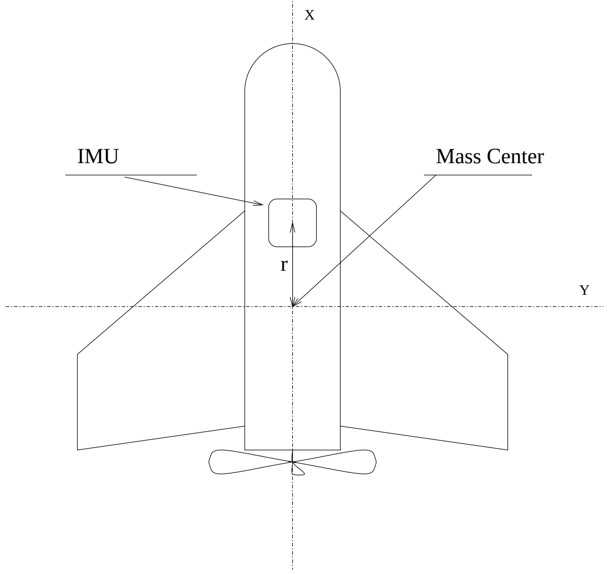
3.3 IMU-Inertial Measuring Unit

In an autonomous vehicle it is prefered to know its position at a specific time. This can be achieved with external sensors. They are usually low rate sensors that need a lot of dataprocessing. A rather simple way to give information at a high frequency, 100-500Hz is to use inertial sensors mounted on or inside the vehicle frame. Normally these sensors are placed in one intertial measuring unit (IMU). The IMU provide readouts about acceleration and rotationrates using accelerometers and rate gyros as sensors.

A strapdown[1] mounted IMUs were used in this thesis and the transformation between diferent frame is done mathematically. The c_{bn} matrix is very useful to to this, look equation 2.13.

In the picture below we see a typical mounting of an IMU. In this case the IMU is placed at a distance of r from the mass center. When the vehicle is under rotation this will cause an acceleration that is measured by the IMU. A high performance navigation system must compensate for this acceleration.

IMU mountpoint in airframe

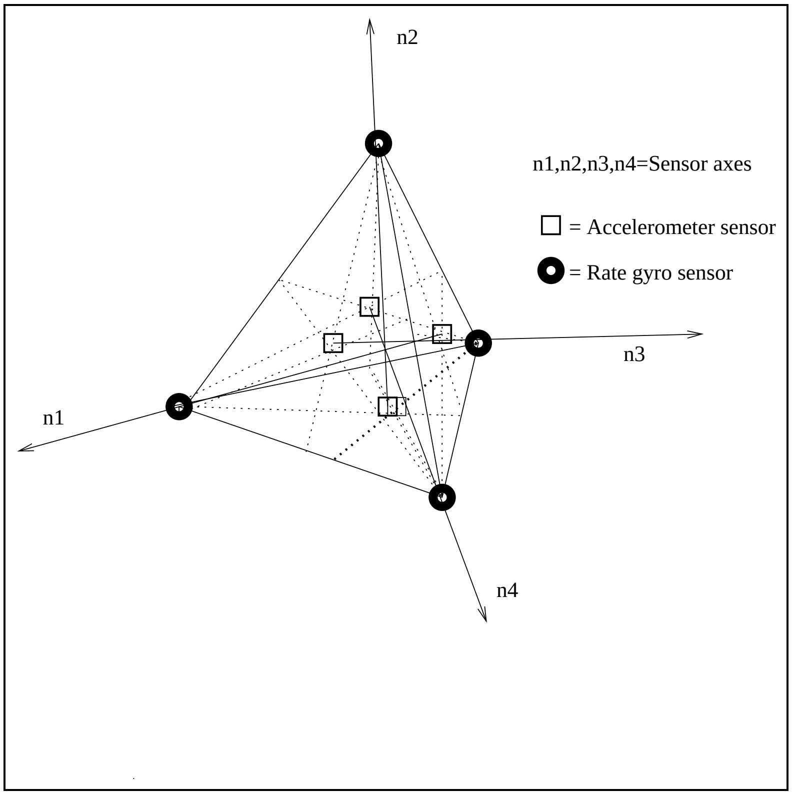


[1] The relation between the sensor axes and the body axes remains the same. The sensors are rigid mounted inside the IMU unit.

3.3.1 The Tetrad

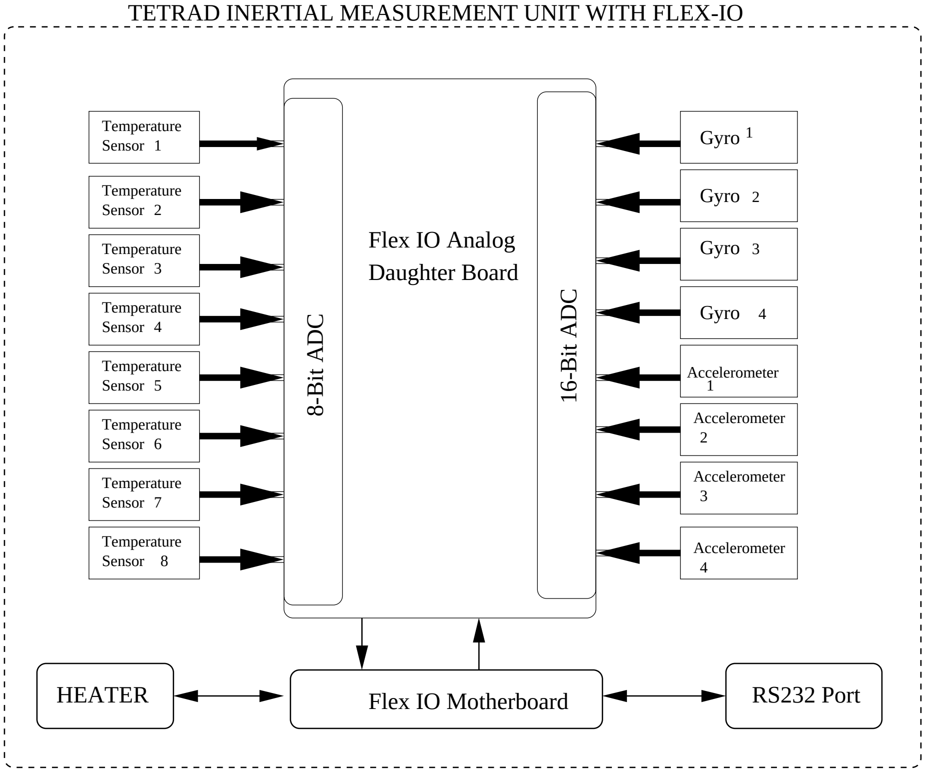
The tetrad have a set-up of newtonian sensors, four rate gyros and four accelerometers. These are mounted on each surface of a tetrad such that $accelerometer_i$ and $gyro_i$ have parallel normalvectors except of opposite directions.

The tetrad



The sensors values are measured with an 16-bit ADC. The temperature sensors are measured with an 8-bit ADC. The desired working temperature approximate 40^oC is reached with heating elements, low ohmic resistors. The current through the resistors are controlled using a DAC (U=RI, U=Voltage, R=Resistance, I=Current).The ADC's and DAC's are controlled with a PIC microcontroller. The communication to the hardware and mircocontroller is done through a FlexIO system developed Ph.D. student Keith Willis at the Department och Mechanical and Mechatronic Engineering at Sydney University. The FlexIO board have both CAN and serial bus interface. The software I wrote uses the serial interface to read and control the Tetrad.

The Tetrad IMU hardware



3.3.2 Tetrad Commands

The Tetrad IMU can be controlled with serveral commands sent through the serial channel. The software inside the Tetrad IMU is called the Firmware. The most useful commands are listed in the table below.

COMMAND	ASCII	DESCRIPTION
RESET	T	Reset the Tetrad
VERSION	V	Read Tetrad Firmware version
STATUS	U	Read Firmware status flag
TIME	T	Return Timestamp. Units 10us
AUTOZERO	Z	Average Channel Values. Calculates the biases.
TEMPERATURE	P	Read Temperatures.
SET	S	Set temperature and sample frequencies.
READ	R	Read specific channel.
READ_ALL	A	Return ADC channel values with time stamps.
START	C	Continous sampling of ADC channels.
STOP	F	Halts continous output

3.3.3 Tetrad IMU Fault Detection

3.4 INS-Inertial Navigation System

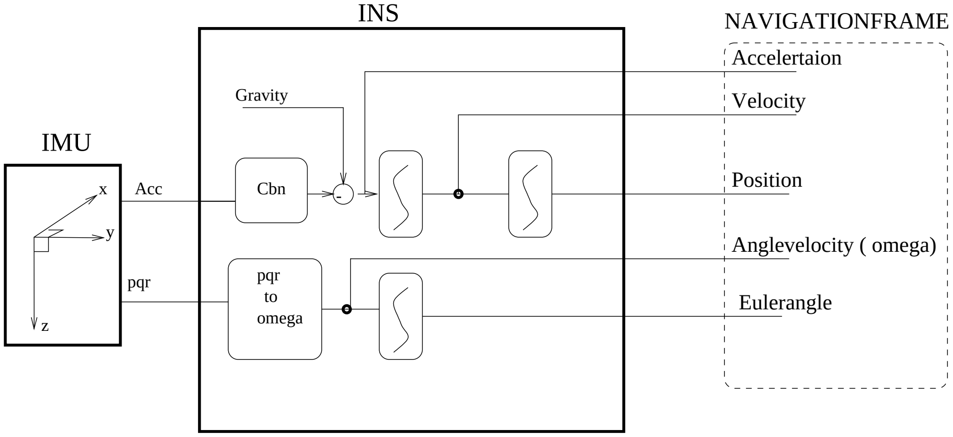
The information provided by the IMU such as the body accelerations and rotationrates $[p, q, r]$ are processed in this unit to provide other systems information about the vehicle position, velocity and attitude. The acceleration in bodyframe is transformed to navigation frame and the gravity vector is subtracted. The resulting acceleration vector is integrated with respect to time and we get the velocity of the vehicle. Another way is to stay in the bodyframe and integrate the body accelearation after the gravity have been cancelled.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \int \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} dt \tag{3.16}$$

The velocity vector is then integrated and we can read the position of the vechicle. If the the acceleration transformed down to navigationframe and integrated we get the position $[x, y, z]^T$ vector.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \int C_{bn}(\phi, \theta, \psi) \begin{bmatrix} u \\ v \\ w \end{bmatrix} dt \tag{3.17}$$

The position vector is given in navigation frame coordinates.



3.4.1 Leveling

Alignment of the IMU is done by leveling. In a strapdown inertial system this is a mathematical operation. The idea is to calculate the biases of the sensors. The scalefactors are assumed to remain the same. The IMU is placed in horizontal position and during the leveling the IMU should remain stationary. If the IMU have tilt sensors the leveling equation can be coorected with the angles.

Accelerometers: We know the normalvectors of the accelerometers. The local gravity vector is projected onto the sensor axes. We can calculate the teoretical readout from the sensor. The sensors are sampled over a time period. The delta bias is the differece between the teoretical value and the calculated mean. The samples m are calculated using (3.4).

$$\delta b_{acc_j} = SF^{-1}_{acc_j}\left(\mathbf{n_{acc_j}}\cdot\mathbf{g_e} - \frac{1}{\mathrm{N}}\sum_{\mathrm{i=0}}^{\mathrm{N}}\mathrm{m}_{\mathrm{acc_j}}[\mathrm{i}]\right) \tag{3.18}$$

Rate Gyros: When the IMU is placed stationary we know that the attitude remains the same. Therefor the meanvalue of the sampled rotationrates should be zero.

$$\delta b_{gyro_j} = SF^{-1}_{gyro_j}\left(0 - \frac{1}{N}\sum_{i=0}^{N} m_{gyro_j}[i]\right) \tag{3.19}$$

The known sensor biases are updated with the biases errors calculated in equations (3.18) and (3.19).

3.5 Navigation Equations

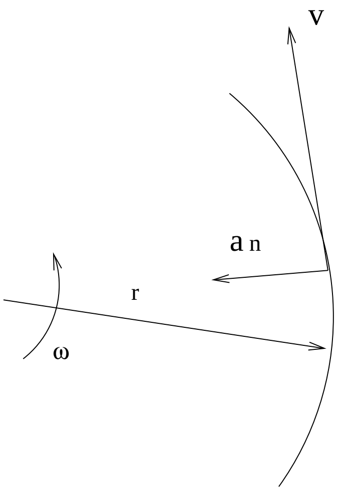
We have $[\dot{u}, \dot{v}, \dot{w}]$ as the true vehicle acceleration in the body frame. This acceleration is not the acceleration measured by the IMU. The IMU accelerometer sensors will measure the gravity vector, the vehicle acceleration the centripetal acceleration and some noise. The radius of curvature is $\mathbf{r}$. The acceleration due to angularvelocity is

$$a_n = v^2/r = \frac{\omega r \omega r}{r} = \omega v \tag{3.20}$$

In vector form this will be

$$\mathbf{a_n} = \omega \times \mathbf{v} = \omega \times (\omega \times \mathbf{r}) \tag{3.21}$$

Centripetal acceleration



The IMU will also measure the gravity acceleration and can not know the difference between this acceleration and true vehicle acceleration. To solve the vehicle acceleration we must substract the known gravity components from the measured accelerations. This gives us equation (3.24). $[a_x, a_y, a_z]$ are the measured acceleration in bodyframe and $[g_x, g_y, g_z]$ are the gravity component expressed in body frame.The gravity vector in the bodyframe is not dependent of the yaw or heading of the airvehicle.

$$\begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} = C_{bn}[0, 0, g_e]^T = \begin{bmatrix} -g_e \sin\theta \\ g_e \cos\theta \sin\phi \\ g_e \cos\theta \cos\phi \end{bmatrix} \tag{3.22}$$

We get the θ and ϕ angles from the INS and can then calculate the gravity components in body frame.

We have another acceleration term if the IMU is not placed in the center of gravity. Suppose the IMU is placed at coordinate $[r_x, r_y, r_z]_{BODY}$ relative to the center of gravity, then there will be an acceleration when the vehicle turns. We call this acceleration $\mathbf{a_{IMU}}$ and the magnitude is dependent of the rotationrates and the placement of the IMU. This extra acceleration can be calculated with equation (3.23)

$$\begin{bmatrix} a_{IMU\,x} \\ a_{IMU\,y} \\ a_{IMU\,z} \end{bmatrix} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \left[\begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \right] \tag{3.23}$$

We use equation (2.6) and (3.22) and get:

$$\begin{aligned} \dot{u} - vr + wq + g_x + a_{IMU\,x} = a_x \\ \dot{v} + ur - wp + g_y + a_{IMU\,y} = a_y \\ \dot{w} - uq + vp + g_z + a_{IMU\,z} = a_z \end{aligned} \tag{3.24}$$

We can solve the true vehicle acceleration from equation (3.24) . Have assumed that the IMU is placed at the center of gravity hence $\mathbf{a_{IMU}} = \mathbf{0}$.

$$.\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} - \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} \tag{3.25}$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} 0 & -w & v \\ w & 0 & -u \\ -v & u & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} - \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} \tag{3.26}$$

Other acceleration terms

If the navigation system is ment to work over longer distances we need to take in calculation the Corriolis acceleration caused by the rotation of the Earth. The Corriolis acceleration will be nonzero if the vehicle is travelling such that the latitude is changing.

Chapter 4

Kalman and Information Filters

This chapter will explain the nonlinear filter and the kalman filter used in the navigation filter. There different models are presented, one almost linear, one using eulerangle representation and one using quaternion angle representation. The complement filter are also presented here.

4.1 Linear System in continous time

In control theory and filter theory we often use continous time models. We can set up two very common equations for continous time systems, equations (4.1) and (4.2).

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{4.1}$$

$$y(t) = Cx(t) + Du(t) \tag{4.2}$$

4.2 Linear System in discrete time

Equations (4.1) and (4.2) can not be use in a computer controlled enviroment. It is prefered to convert/transform the system matrixes into disctete time. We want in some way convert equations (4.1) and (4.2) to (4.3) and (4.4)

$$x(k+1) = \Phi x(k) + \Gamma u(k) \tag{4.3}$$

$$y(k) = Cx(k) + Du(k) \tag{4.4}$$

4.2.1 Calculation of discrete time matrixes

Let h be the discrete time steps, $h = t_{k+1} - t_k$. The time discrete system matrix and control matrix can be calculated with equations (4.5) and (4.6).

$$\Phi = e^{Ah} \tag{4.5}$$

$$\Gamma = \int_0^h e^{As} ds\, B \tag{4.6}$$

If the samplingtime h is small then Φ and Γ is calculated with the formulas found in [33]

Calculation of Φ

$$\Psi = \int_o^h e^{As} ds = Ih + \frac{Ah^2}{2!} + \frac{A^2h^3}{3!} + \ldots + \frac{A^{n-1}h^n}{n!} + \ldots \tag{4.7}$$

$$\Phi = I + A\Psi \tag{4.8}$$

Calculation of Γ

$$\Gamma = \Psi B \tag{4.9}$$

Matrixes C and D remains the same.

4.2.2 Discrete Input Covariance Matrix

We have the ideal control signal $\tilde{u}(t)$ with noise $\epsilon_{u(t)}$.

$$u(t) = \tilde{u}(t) + \epsilon_{u(t)} \tag{4.10}$$

The covariance matrix of this noise is Q. When the continoustime system is discretisized the Q matrix must be recalculated. There are different methods to calculate the Q_k matrix. In [33] and [5] we find two equations, (4.11) and (4.12).

$$Q_k = BQB^T h \tag{4.11}$$

$$Q_k = \frac{1}{2}\left(ABQB^T A^T + BQB^T\right) h \tag{4.12}$$

Triple Integrator Example

The Triple Integrator is described by the system

$$\dot{x}(t) = f\,x(t) + g\,u(t) \tag{4.13}$$

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(t) \tag{4.14}$$

Hence the discrete model for the system is

$$x(k+1) = \Phi\,x(k) + \Gamma\,u(k) \tag{4.15}$$

$$X(k+1) = \begin{bmatrix} 1 & h & h^2/2 \\ 0 & 1 & h \\ 0 & 0 & 1 \end{bmatrix} X(k) + \begin{bmatrix} h^3/6 \\ h^2/2 \\ h \end{bmatrix} U(k) \tag{4.16}$$

4.3 Nonlinear System

In a nonlinear system the state transitionmatrix is built of nonlinear functions such as trigonometric functions, higher order polynomials and square root terms. The nonlinear system is mainly built up by equation (4.17). The vehicle or the system is described by the f model function.

$$\dot{x}(t) = f(x(t), u(t), t) \tag{4.17}$$

$$z_n(t) = h_n(x(t), t) + \epsilon_{z_n(t)} \tag{4.18}$$

We may want to use different sensor models. The sensor models are described by the functions h_n. If we only have one sensor we have only one sensor model, h_0. Usually we have a lot of different types of sensors in the system and therefor different sensor models. In this thesis I used sensor models for velocity, position and attitude updates.

$$u(t) = \tilde{u}(t) + \epsilon_{u(t)} \tag{4.19}$$

The expectated meanvalue for the noiseterms is equal zero.

$$E[\epsilon_{u(t)}] = E[\epsilon_{z_n(t)}] = 0 \tag{4.20}$$

The control noise signal covariance matrix is Q.

$$E[\epsilon_{u(i)}\epsilon_{u(j)}^T] = \delta_{ij}Q(i) \tag{4.21}$$

The observation noise signal covariance matrix is R.

$$E[\epsilon_{z_n(i)}\epsilon_{z_n(j)}^T] = \delta_{ij}R(i) \tag{4.22}$$

And there is no correlation between the observation noise and the control noise.

$$E[\epsilon_{u(t)}\epsilon_{z_n(t)}^T] = 0 \tag{4.23}$$

In equation (4.17) we have a nonlinear function f. We need to have a linearsystem like $\dot{x} = Fx$. This can not be done by linearisation of equation (4.17) at the working point. We derives the

Linearisation of the state transition function f

$$\mathbf{F} = \nabla \mathbf{f} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{4.24}$$

Linearisation statetransition function with respect to the control signal

$$\mathbf{G} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} & \cdots & \frac{\partial f_1}{\partial u_n} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} & \cdots & \frac{\partial f_2}{\partial u_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \frac{\partial f_n}{\partial u_2} & \cdots & \frac{\partial f_n}{\partial u_n} \end{bmatrix} \tag{4.25}$$

Linearisation of the observation function

$$\mathbf{H} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \frac{\partial h_n}{\partial x_2} & \cdots & \frac{\partial h_n}{\partial x_n} \end{bmatrix} \tag{4.26}$$

4.3.1 Linearized Nonlinear System in continous time

The nonlinear system equation (4.17) and (4.60) have been linearized taking the first order derivatives with respect to the states x and the control vector u. The linearisation is done at the workingpoint $x(t)$ and $u(t)$ The result is equation (4.27) and (4.28)

$$\delta\dot{\mathbf{x}}(\mathbf{t}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\delta\mathbf{x}(\mathbf{t}) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\delta\mathbf{u}(\mathbf{t}) = \mathbf{F}\delta\mathbf{x}(\mathbf{t}) + \mathbf{G}\delta\mathbf{u}(\mathbf{t}) \tag{4.27}$$

$$\delta\mathbf{z} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\delta\mathbf{x} + \epsilon = \mathbf{H}\delta\mathbf{x} \tag{4.28}$$

4.3.2 Linearized Nonlinear System in discrete time

We calculate the discrete transition matrix and control matrix using equations (4.8) and (4.9).

$$\Delta\mathbf{x}(\mathbf{k}+\mathbf{1}) = \Phi\Delta\mathbf{x}(\mathbf{k}) + \Gamma\Delta\mathbf{u}(\mathbf{k}) \tag{4.29}$$

$$\Delta\mathbf{z}(\mathbf{k}) = \mathbf{H}\Delta\mathbf{x}(\mathbf{k}) \tag{4.30}$$

4.4 The Complement Filter

In the previous chapters we have discussed the diffent frames, bodyframe and navigationframe, that are used during the navigation. We are interested in the position, velocity and attitude of the airvehicle. A complement filter have the property to estimate the error by cancelling the 'real' values. Suppose we want to calculate the error in position. One position is given by the INS and one is measured with a GPS receiver. Both positions contains errors.

$$p_{ins} = p_{real} + \epsilon_{ins} \tag{4.31}$$

$$p_{gps} = p_{real} + \epsilon_{gps} \tag{4.32}$$

The error ϵ_{ins} and ϵ_{gps} modelled as Gaussian noise with zero as meanvalue, $p \in N(0, \sigma_p)$. To estimate the position error we subtract equations (4.31) and (4.32) and end up with equation (4.33).

$$\Delta p = p_{ins} - p_{gps} = p_{real} + \epsilon_{ins} - p_{real} + \epsilon_{gps} = \epsilon_{ins} - \epsilon_{gps}. \tag{4.33}$$

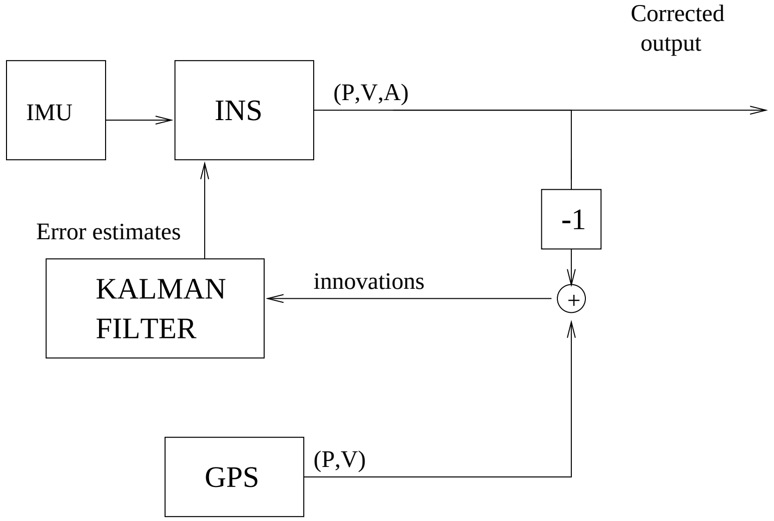
The standard deviation of Δp is $\sigma^2_{\Delta p} = \sigma^2_{ins} + \sigma^2_{gps}$ if σ_{ins} and σ_{gps} are uncorrolated.

There exists two different types of complement filters.

- Complement filter with feedback.
- Complement filter without feedback

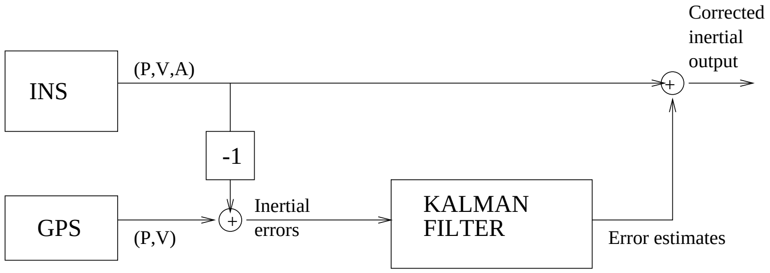
4.4.1 Complement filter with feedback

The INS keeps track of the states of the aircraft. The error filter matrixes are calculated using the states in the INS. The filter is linearized at the working point given by the INS states. The error filter uses Kalmanfilter theory to estimate the error. We have external observations of the states. After each step the error is estimated and fed back to the INS. The error state is cleared. The new filter matrixes is calculated. And a prediction of the error states is done.



4.4.2 Complement filter without feedback

Here there is no feedback to the INS. Instead the error is cancelled at the output. The problem with this is that the INS states diverges if no error feedback is given. The acceleration error will increase with $O(t^2)$ and after a while the position will diverge. Therefor I haven't implemented this version.



4.5 The Kalman Filter

This famous filter technique was invented my Kalman and is very useful especially in navigation and enviroments with gaussian noise. The filter contains four basic blocks in the flowchart. Imagine that you are sleeping in your room and wake up in the middle of the night and the electricity is gone. The whole house is dark and the only way for you to go to the kitchen is to use the information you have about the enviroment , the initial position you have and as sensors you only can use your feets and hands.

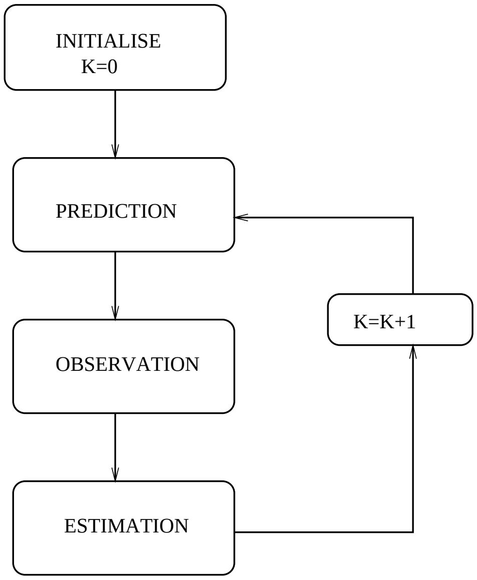
You almost know your current position and know that if you walk in a desired direction you will be able to reach a wall or maybe a carpet that you can feel with your feets.

As you reach something know by you like a carpet, a door, a starcase you have an observation. You can directly use this information to estimate where you are and then predict what will happen if you walk in a special direction. This is repeated and in the end you will reach your goal.

In the story above we can isolate four different states.

- Initialisation
- Prediction
- Observation
- Estimation

Flowchart of Kalmanfilter



We work in discrete time and after the estimation step we increase k by one and go back to the prediction step. The different equation involved in these steps are (4.34) to (4.39).

Initialisation

We set x_0 to a desired value and P_0 to nonzeros in the diagonal and makes the matrix symetic along the diagonal, it is important that this matrix is possible to invert.

Prediction

$$\hat{x}_{k+1} = \Phi_k x_k \tag{4.34}$$

$$\hat{P}_{k+1} = \Phi_k P_k \Phi_k^T + Q_k \tag{4.35}$$

Observation

The observation matrix z_k is measured by a sensor. The difference between the predicted sensor value $H_k \hat{x}_{k+1}$ and the observation is called the innovation. A perfect innovation vector is true gaussian distributed with zero mean.

$$inno = z_k - H_k \hat{x}_{k+1} \tag{4.36}$$

Estimation

After the innovation have been calculated and all the sensor information have been added we can estimate the states of the sensor. The kalman gain is calculated for every time step. The equation below is valid only for one observation. If more observation are used, another equation must be used. Read in [8].

$$K_k = \hat{P}_{k+1} H_{k+1}^T \left(H_k \hat{P}_{k+1} H_k^T + Q_k \right)^{-1} \tag{4.37}$$

The kalman gain K_k have been calculated the predicted states are fused with the observations. And we get a new optimal estimation of the states.

$$x_{k+1} = \hat{x}_{k+1} + K_k \left(z_k - H_k \hat{x}_{k+1} \right) \tag{4.38}$$

The covariance matrix of the new states states is calculated using equation (4.39).

$$P_{k+1} = \left(I + K_k H_k \right) \hat{P}_{k+1} \left(I + K_k H_k \right)^T \tag{4.39}$$

4.6 The Information Filter

The Kalmanfilter used in this thesis is a kalmanfilter in information form. Here the statevector and the uncertaiity matrix is not the same as in a traditional kalmanfilter. We here use the information state vector y with the corresponding information matrix Y, and there exists relations to convert between information-form and standard Kalman form. The relations are equation (4.40) and (4.41).

$$Y(k) = P(k)^{-1} \tag{4.40}$$

The and the P matrix can not be singular, because we need to be able to calculate the inverse. The problem is very big if this matrix become bad conditioned and, the matrix is symetric and this is used in the numerical calculation to make the matrix calculations more stable.

$$y(k) = P(k)^{-1}(k)x(k) = Y(k)\,x(k) \tag{4.41}$$

4.6.1 Prediction

At time k we have the statevector y and matrix Y, we can with this information predict what the states will be at time k+1 when the system have an input with noisematrix Q. The predicted values of y and Y can be calculated with equation(4.42) and (4.44)

$$Y(k+1|k) = [F(k)Y^{-1}(k|k)F^T(k) + Q(k)]^{-1} \tag{4.42}$$

$$L(k) = Y(k+1|k)^{-1}F(k)Y(k|k) = P(k+1|k)F(k)Y(k|k) \tag{4.43}$$

$$y(k+1|k) = L(k)y(k) \tag{4.44}$$

4.6.2 Observation

After the prediction step we fuse the filter state with external observations, an observation can for example be GPS position. The observation is the z vector and with the uncertainty matrix R. With this the z vector and R matrix we calculate the information observation vecor i and the I matrix using (4.45) and (4.46).

$$i(k) = H^T(k)R^{-1}(k)z(k) \tag{4.45}$$

$$I(k) = H^T(k)R^{-1}(k)H(k) \tag{4.46}$$

If more then one observation exists the information form have a very nice property that makes it possible to just all the observations, and this will make it very easy to decentralize the filter calculations.

$$i(k)_{SUM} = \sum_{n=0}^{N=\#OBS} = H^T(k)R_n^{-1}(k)z_n(k) = \sum_{n=0}^{N=\#OBS} i_n(k) \tag{4.47}$$

$$I(k)_{SUM} = \sum_{n=0}^{N=\#OBS} H^T(k)R_n^{-1}(k)H(k) = \sum_{n=0}^{N=\#OBS} I_n(k) \tag{4.48}$$

4.6.3 Estimation

With the predicted values and the observations we can estimate the best informationstate vector and the information matrix. When we are working with a informationfilter whis is done very easily with only the plus operator. With only one observation we use equations (4.49) and (4.50). Again it can be noted that this makes a this filter a very good candidate for decentralized systems with multiple observations, equations (4.51) and (4.52).

$$y(k+1|k+1) = y(k+1|k) + i(k) \tag{4.49}$$

$$Y(k+1|k+1) = Y(k+1|k) + I(k) \tag{4.50}$$

$$y(k) = y(k+1|k+1) = y(k+1|k) + \sum_{n=0}^{N=\#OBS} i_n(k) \tag{4.51}$$

$$Y(k) = Y(k+1|k+1) = Y(k+1|k) + \sum_{n=0}^{N=\#OBS} I_n(k) \tag{4.52}$$

4.6.4 The Innovation

The innovation is the difference between the observation and the predicted value. The innovation can be tested with the χ-square distribution. If the innovation is outside the limits the observation are thrown away. Let Δz_k be the innovation with S_k as the innovation covariance matrix.

χ-square distribution test

$$\Delta z_k^T S_k^{-1} \Delta z_k \leq \varsigma \tag{4.53}$$

The value ς is set to reject observation beyond 95% threshold. This is an excellent to cancel multipath effects in the GPS solution. The filter will remain in predicted state, or the next estimate will be the current prediction. If the observations are rejected the navigation filter runs only on the IMU and that can be a risk. It is here very important with a filter that have been tuned correctly.

4.6.5 Fusion

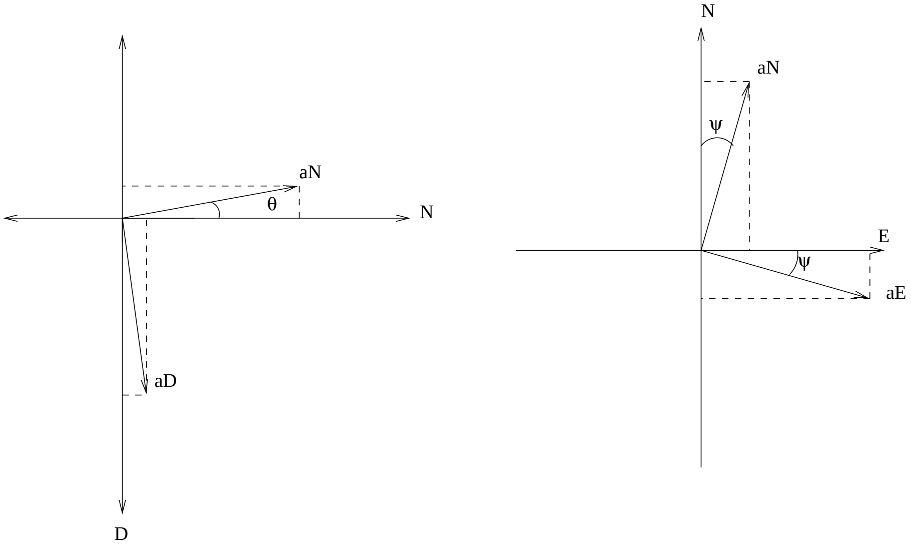
There exists different sensors that can be mounted on an UAV like GPS, air velocity probes, altitude sensors (pressure), ccd cameras, radar and horizon detector. Every sensor can described by a sensor model. The observation matrix H_i are calculated using that model. We can therefor have lot of different sensors with separate observation matrixes. The sensor observations are added to the filter states when present.

4.7 A Linear Error Model

A very common linear error model is a model found in [37].

$$\begin{bmatrix} \delta\dot{P} \\ \delta\dot{V} \\ \delta\dot{\Psi} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I^{3x3}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A_n}\times \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta P \\ \delta V \\ \delta\Psi \end{bmatrix} \tag{4.54}$$

Lets explain it with a figure, we have the accelerations $[a_N, a_E, a_D]$ in navigation frame (NED-frame). The IMU is misaligned with small angles $[\phi, \theta, \psi]$. We want to know the resulting acceleartion on the frame axes caused by the misalignment.



From the figure we calculate the resulting acceleration a_N and we see that the misalignment angles Ψ and θ will project the accelerations a_D and a_E to the x-axis (North). From this we put up the equations

$$\dot{v}_N = a_D \sin\theta - a_E \sin\psi \tag{4.55}$$

If ψ and θ are small we can approximate the equation with

$$\dot{v}_N = a_D\theta - a_E\psi \tag{4.56}$$

We get $\delta\dot{V} = A_n \times \delta\Psi$. Were A_n is a skewsymetric matrix.

$$A_n\times = [a_N, a_E, a_D]^T \times I^{3x3} = \begin{bmatrix} 0 & a_D & -a_E \\ -a_D & 0 & a_N \\ a_E & -a_N & 0 \end{bmatrix} \tag{4.57}$$

The equation (4.54) is written out with all its terms. This model works fine if the trigonometric functions can be approximated with first order Taylor expansion. We can use this model in the leveling process. The misalignment angles can be found if the vehicle is stationary and if the readouts from the INS say that the vehicle is moving. The model is also a good error model for trucks and cars.

We can notice that the misalignment on a stationary vehicle will project the acceleration caused by the gravity forward if we have an small elevation angle and to the left if we have an bank angle. More about this can be read in [37] and [1]. The model was tested on the airvehicle but didn't work so fine.

The model

$$\begin{bmatrix} \delta\dot{N} \\ \delta\dot{E} \\ \delta\dot{D} \\ \delta\dot{v}_N \\ \delta\dot{v}_E \\ \delta\dot{v}_D \\ \delta\dot{\phi} \\ \delta\dot{\theta} \\ \delta\dot{\psi} \end{bmatrix} = \begin{bmatrix} 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&a_D&-a_E \\ 0&0&0&0&0&0&-a_D&0&a_N \\ 0&0&0&0&0&0&a_E&-a_N&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \end{bmatrix} \begin{bmatrix} \delta N \\ \delta E \\ \delta D \\ \delta v_N \\ \delta v_E \\ \delta v_D \\ \delta\phi \\ \delta\theta \\ \delta\psi \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \delta a_N \\ \delta a_E \\ \delta a_D \\ \delta\phi \\ \delta\theta \\ \delta\psi \end{bmatrix} \tag{4.58}$$

The model can be expanded to hold the Coriolis accelerations caused by the rotation of the Earth. In this thesis these acceleration have been neglected.

4.8 Nonlinear Model using Eulerangles

At this stage we are ready to present a nonlinear model that describes an airvehicle. We have a nonlinear system, equations (4.59) and (4.60).

$$\mathbf{\dot{x}(t) = f(x(t), u(t), t)} \tag{4.59}$$

$$\mathbf{y(t) = h(x(t), u(t), t)} \tag{4.60}$$

We take the equations (3.17),(3.25) and (2.39) and put these equatations into a matrix and gets (4.61) This gives a state transition matrix with the size 9x9.

$$\mathbf{f(x,u)} = \begin{bmatrix} \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \cos\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ \begin{bmatrix} u \\ v \\ w \end{bmatrix} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{bmatrix} g_e + \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \\ \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \end{bmatrix} \tag{4.61}$$

The state vector of this nonlinear model have the elements position,velocity in bodyframe and attitude in eulerangles.

$$\mathbf{x} = \begin{bmatrix} X & Y & Z & u & v & w & \psi & \theta & \psi \end{bmatrix}^{\mathrm{T}} \tag{4.62}$$

The timederivative of $\mathbf{x}$ is :

$$\dot{\mathbf{x}} = \frac{\partial \mathbf{x}}{\partial \mathrm{t}} = \begin{bmatrix} \dot{X} & \dot{Y} & \dot{Z} & \dot{u} & \dot{v} & \dot{w} & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^{\mathrm{T}} \tag{4.63}$$

4.8.1 State transition matrix

The nonlinear state transition function is linearized. The matrix contains the partial derivatives of $\mathbf{f}$ with respect to $\mathbf{x}$.

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \left[\frac{\partial \mathbf{f}}{\partial X}, \frac{\partial \mathbf{f}}{\partial Y}, \frac{\partial \mathbf{f}}{\partial Z}, \frac{\partial \mathbf{f}}{\partial u}, \frac{\partial \mathbf{f}}{\partial v}, \frac{\partial \mathbf{f}}{\partial w}, \frac{\partial \mathbf{f}}{\partial \phi}, \frac{\partial \mathbf{f}}{\partial \theta}, \frac{\partial \mathbf{f}}{\partial \psi}\right] \tag{4.64}$$

$$\left[\frac{\partial \mathbf{f}}{\partial X}, \frac{\partial \mathbf{f}}{\partial Y}, \frac{\partial \mathbf{f}}{\partial Z}\right] = \mathbf{0^{9x3}} \tag{4.65}$$

$$\left[\frac{\partial \mathbf{f}}{\partial u}, \frac{\partial \mathbf{f}}{\partial v}, \frac{\partial \mathbf{f}}{\partial w}\right] = \begin{bmatrix} \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \cos\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \\ \mathbf{I^{3x3}} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{bmatrix} \\ \mathbf{0^{3x3}} \end{bmatrix} \tag{4.66}$$

$$\frac{\partial \mathbf{f}}{\partial \phi} = \begin{bmatrix} \begin{bmatrix} 0 & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi & \cos\phi\sin\psi - \sin\phi\sin\theta\cos\psi \\ 0 & -\sin\phi\cos\psi - \sin\phi\sin\theta\sin\psi & -\cos\phi\cos\psi - \sin\phi\sin\theta\sin\psi \\ 0 & \cos\phi\cos\theta & -\sin\phi\cos\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ \begin{bmatrix} 0 \\ \cos\theta\cos\phi \\ -\cos\theta\sin\phi \end{bmatrix} g_e \\ \begin{bmatrix} 0 & \cos\phi\tan\theta & \sin\phi\tan\theta \\ 0 & -\sin\phi & -\cos\phi \\ 0 & \cos\phi\sec\theta & -\sin\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \end{bmatrix} \tag{4.67}$$

$$\frac{\partial \mathbf{f}}{\partial \theta} = \begin{bmatrix} \begin{bmatrix} -\sin\theta\cos\psi & \sin\phi\cos\theta\cos\psi & \cos\phi\cos\theta\cos\psi \\ -\sin\theta\sin\psi & \cos\phi\cos\theta\sin\psi & \cos\phi\cos\theta\sin\psi \\ -\cos\theta & -\sin\phi\sin\theta & -\cos\phi\sin\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ -\begin{bmatrix} \cos\theta \\ \sin\theta\sin\phi \\ \sin\theta\cos\phi \end{bmatrix} g_e \\ \begin{bmatrix} 0 & \sin\phi(1 + \tan^2\theta) & \cos\phi(1 + \tan^2\theta) \\ 0 & 0 & 0 \\ 0 & \sin\phi\sec\theta\tan\theta & \cos\phi\sec\theta\tan\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \end{bmatrix} \tag{4.68}$$

$$\frac{\partial \mathbf{f}}{\partial \psi} = \begin{bmatrix} \begin{bmatrix} -\cos\theta\sin\phi & -\cos\psi\cos\phi - \sin\psi\sin\theta\sin\phi & \sin\psi\cos\phi - \cos\psi\sin\theta\sin\phi \\ \cos\theta\cos\phi & -\cos\psi\sin\phi + \cos\psi\sin\theta\cos\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ \mathbf{0^{6x1}} \end{bmatrix} \tag{4.69}$$

4.8.2 Control matrix

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \left[\frac{\partial \mathbf{f}}{\partial a_x}, \frac{\partial \mathbf{f}}{\partial a_y}, \frac{\partial \mathbf{f}}{\partial a_z}, \frac{\partial \mathbf{f}}{\partial p}, \frac{\partial \mathbf{f}}{\partial q}, \frac{\partial \mathbf{f}}{\partial r}\right] \tag{4.70}$$

$$\left[\frac{\partial \mathbf{f}}{\partial a_x}, \frac{\partial \mathbf{f}}{\partial a_y}, \frac{\partial \mathbf{f}}{\partial a_z}\right] = \begin{bmatrix} \mathbf{0^{3x3}} \\ \mathbf{I^{3x3}} \\ \mathbf{0^{3x3}} \end{bmatrix} \tag{4.71}$$

$$\left[\frac{\partial \mathbf{f}}{\partial p}, \frac{\partial \mathbf{f}}{\partial q}, \frac{\partial \mathbf{f}}{\partial r}\right] = \begin{bmatrix} \mathbf{0^{3x3}} \\ \begin{bmatrix} u \\ v \\ w \end{bmatrix} \times \mathbf{I^{3x3}} = \begin{bmatrix} 0 & -w & v \\ w & 0 & -u \\ -v & u & 0 \end{bmatrix} \\ \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \end{bmatrix} \tag{4.72}$$

$$G = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{I}^{3\text{x}3} & \begin{bmatrix} u \\ v \\ w \end{bmatrix} \times \mathbf{I}^{3\text{x}3} \\ \mathbf{0} & \mathbf{C}_{\dot\phi\dot\theta\dot\psi|\mathbf{pqr}} \end{bmatrix} \tag{4.73}$$

$$\mathbf{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -w & v \\ 0 & 1 & 0 & w & 0 & -u \\ 0 & 0 & 1 & -v & u & 0 \\ 0 & 0 & 0 & 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & 0 & 0 & 0 & \cos\phi & -\sin\phi \\ 0 & 0 & 0 & 0 & -\sin\phi\sec\theta & -\cos\phi\sec\theta \end{bmatrix} \tag{4.74}$$

The process noise is the noise of the input to the filter. The inputs are the accelerations and rotationrates in bodyframe of the vehicle.

$$\epsilon_{\mathbf{u(t)}} = \begin{bmatrix} \epsilon_{a_x} & \epsilon_{a_y} & \epsilon_{a_z} & \epsilon_p & \epsilon_q & \epsilon_r \end{bmatrix}^{\mathrm{T}} \tag{4.75}$$

4.8.3 Observation Models

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \\ \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \cos\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \end{bmatrix} \tag{4.76}$$

Position observation model

The position observation equation is:

$$\delta\mathbf{z}_{\mathrm{XYZ}} = \mathbf{H}_{\mathrm{XYZ}}\delta\mathbf{x} + \epsilon_{\mathrm{XYZ}} \tag{4.77}$$

With the observation matrix H_{XYZ}, derived from (4.76).

$$\mathbf{H}_{\mathrm{XYZ}} = \begin{bmatrix} \mathbf{I}^{3\text{x}3} & \mathbf{0}^{6\text{x}3} \end{bmatrix} \tag{4.78}$$

Velocity obsevation model

Velocity observation equation and the corresponding observation matrix $H_{\dot{X}\dot{Y}\dot{Z}}$.

$$\delta \mathbf{z}_{\dot{X}\dot{Y}\dot{Z}} = \mathbf{H}_{\dot{X}\dot{Y}\dot{Z}} \delta \mathbf{x} + \epsilon_{\dot{X}\dot{Y}\dot{Z}} \tag{4.79}$$

$$\mathbf{H}_{\dot{X}\dot{Y}\dot{Z}} = \begin{bmatrix} \mathbf{0}^{3\text{x}3} & C_{bn} & \mathbf{0}^{6\text{x}3} \end{bmatrix} \tag{4.80}$$

Attitude observation model

$$\delta \mathbf{z}_{\phi\theta\psi} = \mathbf{H}_{\phi\theta\psi} \delta \mathbf{x} + \epsilon_{\phi\theta\psi} \tag{4.81}$$

$$\mathbf{H}_{\phi\theta\psi} = \begin{bmatrix} \mathbf{0}^{6\text{x}3} & \mathbf{I}^{3\text{x}3} \end{bmatrix} \tag{4.82}$$

4.9 Nonlinear Model with Quaternions

Another model tested is this model that uses quaternions. The quaternions have the advantage that all we use polynomials of order two in the continous time transision state matrix, equation (4.83). This matrix is built of three equations namely (3.17),(3.25) and (**??**).

$$\mathbf{f}(\mathbf{x},\mathbf{u}) = \begin{bmatrix} \begin{bmatrix} (e_0^2+e_1^2-e_2^2-e_3^2) & 2(e_1e_2-e_0e_3) & 2(e_1e_3+e_0e_2) \\ 2(e_1e_2+e_0e_3) & (e_0^2-e_1^2+e_2^2-e_3^2) & 2(e_2e_3-e_0e_1) \\ 2(e_1e_3-e_0e_2) & 2(e_2e_3+e_0e_1) & (e_0^2-e_1^2-e_2^2+e_3^2) \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ \begin{bmatrix} u \\ v \\ w \end{bmatrix} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} 2(e_1e_3-e_0e_2) \\ 2(e_2e_3+e_0e_1) \\ (e_0^2-e_1^2-e_2^2+e_3^2) \end{bmatrix} g_e + \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \\ \frac{1}{2} \begin{bmatrix} -e_1 & -e_2 & -e_3 \\ e_0 & -e_3 & e_2 \\ e_3 & e_0 & -e_1 \\ -e_2 & e_1 & e_0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \end{bmatrix} \tag{4.83}$$

4.9.1 State transition matrix

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \left[\frac{\partial \mathbf{f}}{\partial X}, \frac{\partial \mathbf{f}}{\partial Y}, \frac{\partial \mathbf{f}}{\partial Z}, \frac{\partial \mathbf{f}}{\partial u}, \frac{\partial \mathbf{f}}{\partial v}, \frac{\partial \mathbf{f}}{\partial w}, \frac{\partial \mathbf{f}}{\partial e_0}, \frac{\partial \mathbf{f}}{\partial e_1}, \frac{\partial \mathbf{f}}{\partial e_2}, \frac{\partial \mathbf{f}}{\partial e_3} \right] \tag{4.84}$$

$$\left[\frac{\partial \mathbf{f}}{\partial X}, \frac{\partial \mathbf{f}}{\partial Y}, \frac{\partial \mathbf{f}}{\partial Z} \right] = \mathbf{0^{9x3}} \tag{4.85}$$

$$\left[\frac{\partial \mathbf{f}}{\partial u}, \frac{\partial \mathbf{f}}{\partial v}, \frac{\partial \mathbf{f}}{\partial w} \right] = \begin{bmatrix} \begin{bmatrix} (e_0^2+e_1^2-e_2^2-e_3^2) & 2(e_1e_2-e_0e_3) & 2(e_1e_3+e_0e_2) \\ 2(e_1e_2+e_0e_3) & (e_0^2-e_1^2+e_2^2-e_3^2) & 2(e_2e_3-e_0e_1) \\ 2(e_1e_3-e_0e_2) & 2(e_2e_3+e_0e_1) & (e_0^2-e_1^2-e_2^2+e_3^2) \end{bmatrix} \\ \mathbf{I^{3x3}} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{bmatrix} \\ \mathbf{0^{4x3}} \end{bmatrix} \tag{4.86}$$

$$
\frac{\partial \mathbf{f}}{\partial \mathbf{e_0}} = \begin{bmatrix} 2e_0 \begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ 2e_0 \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} g_e \\ \frac{1}{2} \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \end{bmatrix} \quad \frac{\partial \mathbf{f}}{\partial \mathbf{e_1}} = \begin{bmatrix} 2e_1 \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ 2e_1 \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} g_e \\ \frac{1}{2} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \end{bmatrix}
$$

$$
\frac{\partial \mathbf{f}}{\partial \mathbf{e_2}} = \begin{bmatrix} 2e_2 \begin{bmatrix} -1 & 1 & 1 \\ 1 & 1 & 1 \\ -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ 2e_2 \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} g_e \\ \frac{1}{2} \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \end{bmatrix} \quad \frac{\partial \mathbf{f}}{\partial \mathbf{e_3}} = \begin{bmatrix} 2e_3 \begin{bmatrix} -1 & -1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ 2e_3 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} g_e \\ \frac{1}{2} \begin{bmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \end{bmatrix} \tag{4.87}
$$

4.9.2 Control matrix

$$
\frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \left[\frac{\partial \mathbf{f}}{\partial a_x}, \frac{\partial \mathbf{f}}{\partial a_y}, \frac{\partial \mathbf{f}}{\partial a_z}, \frac{\partial \mathbf{f}}{\partial p}, \frac{\partial \mathbf{f}}{\partial q}, \frac{\partial \mathbf{f}}{\partial r} \right] \tag{4.88}
$$

$$
\left[\frac{\partial \mathbf{f}}{\partial a_x}, \frac{\partial \mathbf{f}}{\partial a_y}, \frac{\partial \mathbf{f}}{\partial a_z} \right] = \begin{bmatrix} \mathbf{I^{3x3}} \\ \mathbf{0^{7x3}} \end{bmatrix} \tag{4.89}
$$

$$
\left[\frac{\partial \mathbf{f}}{\partial p}, \frac{\partial \mathbf{f}}{\partial q}, \frac{\partial \mathbf{f}}{\partial r} \right] = \begin{bmatrix} \mathbf{0^{6x3}} \\ \begin{bmatrix} u \\ v \\ w \end{bmatrix} \times \mathbf{I^{3x3}} = \begin{bmatrix} 0 & -w & v \\ w & 0 & -u \\ -v & u & 0 \end{bmatrix} \\ \frac{1}{2} \begin{bmatrix} -e_1 & -e_2 & -e_3 \\ e_0 & -e_3 & e_2 \\ e_3 & e_0 & -e_1 \\ -e_2 & e_1 & e_0 \end{bmatrix} \end{bmatrix} \tag{4.90}
$$

4.9.3 Observation Models

Position observation model

$$
\delta \mathbf{z}_{\mathrm{XYZ}} = \mathbf{H}_{\mathrm{XYZ}} \delta \mathbf{x} + \epsilon_{\mathrm{XYZ}} \tag{4.91}
$$

$$
\mathbf{H}_{\mathrm{XYZ}} = \begin{bmatrix} \mathbf{I}^{3\mathrm{x}3} & \mathbf{0}^{7\mathrm{x}3} \end{bmatrix} \tag{4.92}
$$

Velocity observation model

The velocity in the statevector **x** is represented in bodyaxis. The velocities (u,v,w) need to be transformed to navigation frame and is done with equation (4.93).

$$
\mathbf{H}_{\dot{\mathrm{X}}\dot{\mathrm{Y}}\dot{\mathrm{Z}}} = \begin{bmatrix} \mathbf{0}^{3\mathrm{x}3} & C_{bn} & \mathbf{0}^{7\mathrm{x}3} \end{bmatrix} \tag{4.93}
$$

Attitude observation model

Because we in use quaternions and the observations are in eulerangle representation we need to do a convertion. The formula (2.45) is written in matrix from and we get equation (4.94).

$$\mathbf{h}_{\phi\theta\psi}(\mathbf{x}) = \begin{bmatrix} \arctan\left(\frac{2(e_0e_1+e_2e_3)}{e_0^2-e_1^2-e_2^2+e_3^2}\right) \\ \arcsin\left(2(e_0e_2-e_3e_1)\right) \\ \arctan\left(\frac{2(e_0e_3+e_1e_2)}{e_0^2+e_1^2-e_2^2-e_3^2}\right) \end{bmatrix} \tag{4.94}$$

This model must be linearized, this gives us (4.95)

$$\mathbf{H}_{\phi\theta\psi} = \begin{bmatrix} \mathbf{O^{3x6}} & \frac{\partial h_{\phi\theta\psi}(x)}{\partial e_0} & \frac{\partial h_{\phi\theta\psi}(x)}{\partial e_1} & \frac{\partial h_{\phi\theta\psi}(x)}{\partial e_2} & \frac{\partial h_{\phi\theta\psi}(x)}{\partial e_3} \end{bmatrix} \tag{4.95}$$

$$\frac{\partial \mathbf{h}_{\phi\theta\psi}(\mathbf{x})}{\partial \mathbf{e_0}} = \begin{bmatrix} \frac{2e_1(e_0^2-e_1^2-e_2^2+e_3^2)-4e_0(e_0e_1+e_2e_3)}{(e_0^2-e_1^2-e_2^2+e_3^2)^2+4(e_0e_1+e_2e_3)^2} \\ \frac{2e_2}{\sqrt{1+4(e_0e_2-e_3e_1)^2}} \\ \frac{2e_3(e_0^2+e_1^2-e_2^2-e_3^2)-4e_0(e_0e_3+e_1e_2)}{(e_0^2+e_1^2-e_2^2-e_3^2)^2+4(e_0e_3+e_1e_2)^2} \end{bmatrix} \tag{4.96}$$

$$\frac{\partial \mathbf{h}_{\phi\theta\psi}(\mathbf{x})}{\partial \mathbf{e_1}} = \begin{bmatrix} \frac{2e_0(e_0^2-e_1^2-e_2^2+e_3^2)+4e_1(e_0e_1+e_2e_3)}{(e_0^2-e_1^2-e_2^2+e_3^2)^2+4(e_0e_1+e_2e_3)^2} \\ \frac{-2e_3}{\sqrt{1+4(e_0e_2-e_3e_1)^2}} \\ \frac{2e_2(e_0^2+e_1^2-e_2^2-e_3^2)-4e_1(e_0e_3+e_1e_2)}{(e_0^2+e_1^2-e_2^2-e_3^2)^2+4(e_0e_3+e_1e_2)^2} \end{bmatrix} \tag{4.97}$$

$$\frac{\partial \mathbf{h}_{\phi\theta\psi}(\mathbf{x})}{\partial \mathbf{e_2}} = \begin{bmatrix} \frac{2e_3(e_0^2-e_1^2-e_2^2+e_3^2)+4e_2(e_0e_1+e_2e_3)}{(e_0^2-e_1^2-e_2^2+e_3^2)^2+4(e_0e_1+e_2e_3)^2} \\ \frac{2e_0}{\sqrt{1+4(e_0e_2-e_3e_1)^2}} \\ \frac{2e_1(e_0^2+e_1^2-e_2^2-e_3^2)+4e_2(e_0e_3+e_1e_2)}{(e_0^2+e_1^2-e_2^2-e_3^2)^2+4(e_0e_3+e_1e_2)^2} \end{bmatrix} \tag{4.98}$$

$$\frac{\partial \mathbf{h}(\mathbf{x})_{\phi\theta\psi}}{\partial \mathbf{e_3}} = \begin{bmatrix} \frac{2e_2(e_0^2-e_1^2-e_2^2+e_3^2)-4e_3(e_0e_1+e_2e_3)}{(e_0^2-e_1^2-e_2^2+e_3^2)^2+4(e_0e_1+e_2e_3)^2} \\ \frac{-2e_1}{\sqrt{1+4(e_0e_2-e_3e_1)^2}} \\ \frac{2e_0(e_0^2+e_1^2-e_2^2-e_3^2)+4e_3(e_0e_3+e_1e_2)}{(e_0^2+e_1^2-e_2^2-e_3^2)^2+4(e_0e_3+e_1e_2)^2} \end{bmatrix} \tag{4.99}$$

Chapter 5

GPS-Global Positioning System

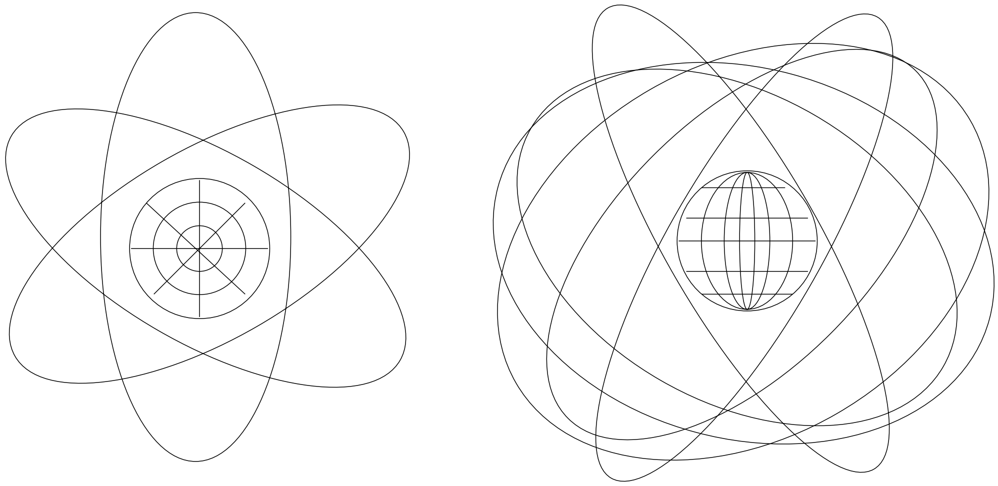
5.1 GPS-Global Positioning System

The information about the GPS system is read in [6],[22], [17] and [20]. The GPS system is a worldwide navigation system based on radio transmitters placed in 26 satellites. The GPS system was developed by Department of Defence (DOD), USA. The satellites are placed in circular orbits around the Earth with an altitude of 20 200 km. There are six different orbital planes. Each plane make an inclination of 55^o relative to Earth's equator. The satellites are placed so that a receiver can track atleast four satellites with a 5^o elevation mask above the horizon.

GPS working principle by Professor Bradford Parkinsson

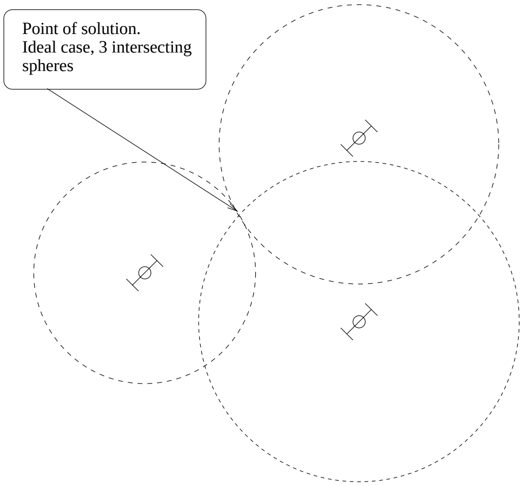
The fundamental navigation technique for the GPS is to use one-way ranging from GPS satellites that are also broadcasting their estimated positions. Pseudoranges are measured to four satellites simultaneously in view by matching (correlating) the incoming signals with a user-generated replica signal and measuring the received phase against the user's (relative crude) crystal clock. With four satellites and appropriate geometry, four unknowns can be determined; typically they are latitude, longitude, altitude and a correction to the user's clock.

Orbits of GPS satellites



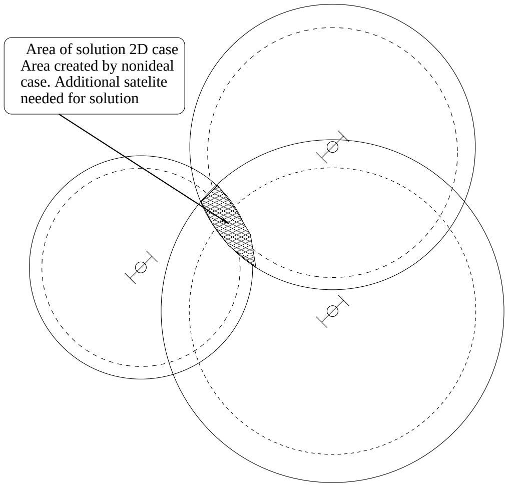
Pseudorange

The pseudorange is measured by correlating the incoming signal with a identical signal re-produced by the clock in the GPS receiver. With the pseudorange to atleast four satellites it is possible to calculate the receivers position in three dimension. One element in the solution vector gives an estimate of the receiver clock offset.



If we had a perfect clock in both the satellite and the receiver and if the signal wasn't affected by noise then there would be really easy to calculate the receivers position.

Because of bad crystals in the receivers, propagationdelays, gravity changes and lot more we have a system with four unknown variables. Three for position and one for time offset of the GPS receiver clock.



GPS-SPS and PPS

There are two different GPS users, those with Y-code access which give PPS-Precise Positioning Service and those without access and those have SPS-Standrad Positioning Service. The GPS signals are transmitted on two carrier frequencies, 1575.42 MHz and 1227.60Mhz GHz. The marjor frequency is L_1 at 1575.42Mhz. The signals are modulated by two different pseudorandom noise codes. One with a frequency of 1.023Mhz called the C/A-code and one at 10.23Mhz called the Y-code. The Y-code have a wavelength of 30m and is restricted for military purposes and can be used to generate very accurate position and velocity solutions. The coarse/acquisition code (C/A-code) have a wavelenght of 300m is less accurate but free to use.

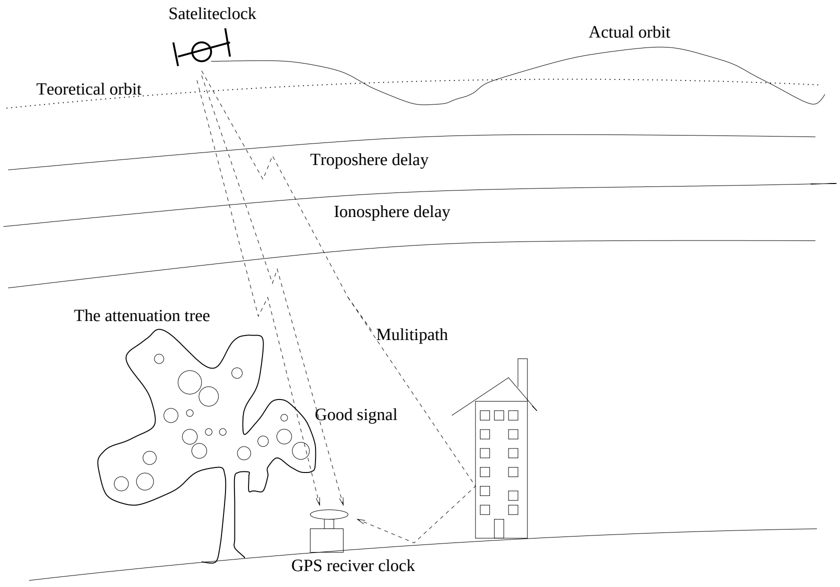
The L_2 signal contains the Y-code modulation and are used to compensate the transmitted signals for Ionoshpere delays. This can be done with two frequencies because the delay is frequency dependent. But Y-code is restricted only for authorized users. Additional to this 50bit/s satellite messages are transmitted on L_1 and is repeated every 30min. The message blocks are 1500bit long and contains different informations that are unique for every satellite. Information about satellite position, estimated trajectoria, ionosphere delay and status are received in these messages.

5.2 GPS-Signal errors

The GPS signal errors are of different types, the main errors is

- Multipath
- Ionosphere delays
- Troposphere delays
- Signal attenuation
- Ephemeris error
- Satellite clock error
- Receiver clock error

GPS errors



5.2.1 Multipath

This occurs when the satellite signal bounces against like a wall and then received by the GPS receiver. The distance the signal have traveled is not the shortest geometric distance. Use a antenna with cut-off angle and placement of antenna such that it minimizes the problem. An reference station can also be used to cancel these errors. With an filter it is possible to reject pseudoranges that are beyond the filter mask.

Fermats Principle

Fermat's Principle says that:
If a wave travel between two points A and B the path will always give the minimum time.

$$(t_s - t_r)_P = \min \frac{1}{c_0} \int_P n(s) ds, P \in \{All\, possible\, paths\} \tag{5.1}$$

5.2.2 Signal attenuation

Here the GPS signal is damped by different materials, like forests........

5.2.3 Satellite motion

The perfect satellite orbits should be ellipses. The satellites are affected by the Earth gravity vector and this vector does not have the same magnitude all over the planet. The satellites are also under influence of forces from other planets, the moon and the sunwinds. These forces can not be neglected. The paths of the satellites are predicted and estimated by the ground stations. The harmonics coefficients for the angle inclination and orbit radius are update in each satellite continously.

5.2.4 Ionosphere propagation delay

When the GPS signal propagates through the Ionosphere the signal will collide with free ions. The delay is dependent of the density of ions and the elevation of the signal (distance of travel). The number of ions is dependent on several factors like time, magnetic latitude and the sunspot cycle.

A method to calculate Ionosphere propagation delay was presented by Saastamoinen in 1973. This model does a serial expansion and says that the refraction index n is frequency dependent.

$$n(s) = 1 + \frac{c_1}{f^2} + \frac{c_2}{f^4} + \dots \tag{5.2}$$

We integrate (5.2) over the signal path and get the error in pseudorange caused by the Ionosphere, see equation (5.3). The parameters b_1 and b_2 are transmitted from the satellite and this is a mode that is restricted only to the military. It is only those with Y-code access that get PPS.

$$\delta\rho^I(f) = \int_P (n(s) - 1) ds \approx \frac{b_1}{f^2} + \frac{b_2}{f^4} \tag{5.3}$$

With this dual-frequency techinque gives a 1-2 ranging accuracy for well calibrated receivers, more information can be found in [6] page 479.

5.2.5 Satellite clock error

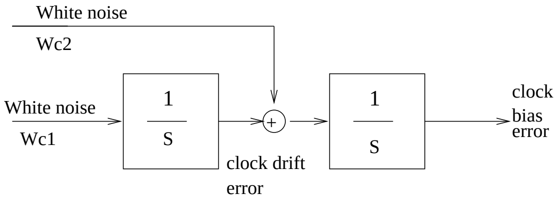
The GPS uses atomic clock, cesium and rubidium oscillators. The drift in one day of these clocks result in 10^{-8} s or about 3.5m error in range.

5.2.6 Troposhpere propagation delay

The Troposhpere closer to the Earth surface. The delay here is sensitive to user altitude and signal elevation angle and can be modelled with a function that takes temperature, water vapour and temperature as arguments.

5.2.7 GPS Receiver Clock Model

The GPS satellites have onboard atomic clocks but the normal receivers do not have atomic clocks. Instead they have different sorts of oscillators usually based on differennt sort of crystals. This will cause the clocks to drift in time. This can be modelled with a model introduced by Allan and found in [20]. Is is a double integrator of whitenoise. Allan have in this modell two different white noise sources with spectral densities S_b and S_f. S_b, S_f are called the Classical Allan variance parameters.



$$w_c = \begin{bmatrix} w_{c1} \\ w_{c2} \end{bmatrix} = N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} S_b & 0 \\ 0 & S_f \end{bmatrix}\right) \tag{5.4}$$

Typical values for the Classical Allan Variance parameters.

Variable	Value
h_0	$2 \cdot 10^{-19}$
h_{-2}	$2 \cdot 10^{-20}$
S_f	$2h_0$
S_g	$8\pi^2 h_{-2}$

The model estimates the frequency drift error and the frequency bias error of the receiver clock. We have a model where the frequency rate are integrated from jerks, (White noise) with a spectral density of S_f. The frequency offset (bias) are the sum of integrated white noise and the frequency rate. It can be written in a continious time system like

$$\dot{x}_c(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x_c(t) + w_c(t) \tag{5.5}$$

$$y_c(t) = [10]x_c(t) \tag{5.6}$$

We make this to a discrete time system with a discrete time step of of Δt.

$$x_c(k+1) = \Phi_c x_c(k) + w_c(k) \tag{5.7}$$

$$x_c = [\Delta b_c, \Delta f_c]^T \tag{5.8}$$

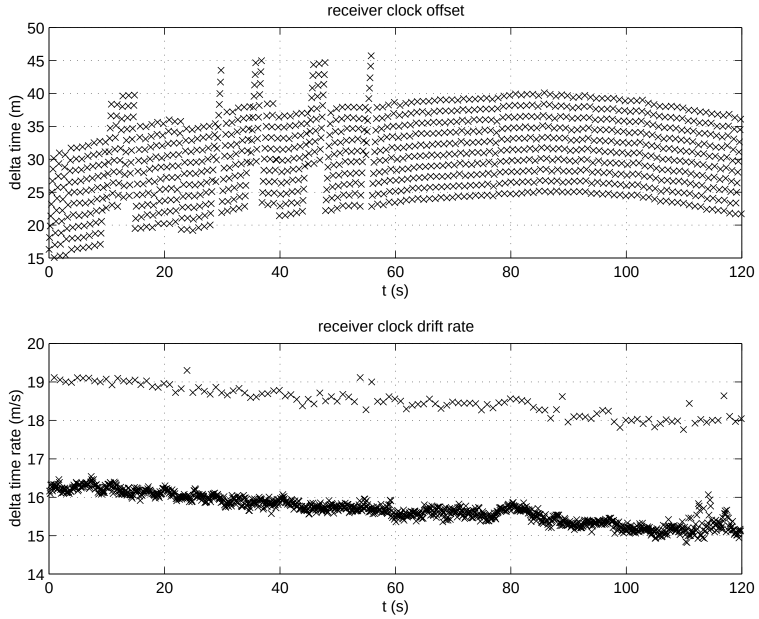
$$\Phi_c = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \tag{5.9}$$

The process noise covariance matrix is

$$Q_c = \begin{bmatrix} S_b \Delta t + S_f \frac{\Delta t^3}{3} & S_f \frac{\Delta t^2}{2} \\ S_f \frac{\Delta t^2}{2} & S_f \Delta t \end{bmatrix} \tag{5.10}$$

GPS-clock drift expressed in pseudorange

This figure was calculated with a Matlab script using data from a flight done in August 1999. The GPS position solution prepared by Ph.D. student Ben Grosholsky.



5.3 WGS-84 transformations

It is important to be able to do coordinate transformation from and to the navigation frame from the ECEF frame. The variables used in this transformation is listed below.

Variable	
λ	Longitude
ϕ	Latitude
R_N	Local Earth radius
h	Height above sea level
a	Earth semi major axis, Earth radius at equator.a=6378137.0
b	Earth semi minor axis, radius at poles.b=a(1-f)
f	Earth flattering.f=0.003352810664747

5.3.1 ECEF to geodetic transformation

Equations to transform ECEF coordinates $[x, y, z]$ to geodetic coordinates $[\lambda, \phi, h]$.

$$p = \sqrt{x^2 + y^2} \tag{5.11}$$

$$\theta = \tan^{-1}\left(\frac{z\,a}{p\,b}\right) \tag{5.12}$$

$$\dot{e}^2 = \frac{a^2 - b^2}{b^2} = \frac{2f - f^2}{(1-f)^2} \tag{5.13}$$

$$e^2 = f(2 - f) \tag{5.14}$$

$$\phi = \tan^{-1}\left(\frac{z + \dot{e}^2 b \sin(\theta)^3}{p - e^2 a \cos(\theta)^3}\right) \tag{5.15}$$

$$R_N = \frac{a}{\sqrt{1 - e^2 \sin^2 \phi}} \tag{5.16}$$

$$h = \frac{p}{\cos\phi} - R_N \tag{5.17}$$

$$\lambda = \tan^{-1}\left(\frac{y}{x}\right) \tag{5.18}$$

$$\tag{5.19}$$

5.3.2 Geodetic to ECEF transformation

The local Earth radius is dependent of latitude because the flattering of the Earth. The Earth looks almost like an ellipse and the radius can be calculated using (5.20).

Local Earth radius

$$R_N(\phi) = \frac{a^2}{\sqrt{a^2 \cos^2\phi + b^2 \sin^2\phi}} \tag{5.20}$$

The cartesian coordinates becomes

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{ECEF} = \begin{bmatrix} (R_N + h)\cos\phi\cos\lambda \\ (R_N + h)\cos\phi\sin\lambda \\ (\frac{a^2}{b^2}R_N + h)\sin\phi \end{bmatrix} \tag{5.21}$$

5.3.3 ECEF vector to NED transformation

The calculated satellite positions and velocities need to be transformed to the navigationframe. To do this we need the approximate longitude and latitude.

The transformation is done using equation (5.22), λ is longitude and ϕ is latitude.

$$C(\lambda,\phi) = \begin{bmatrix} -\sin\phi\cos\lambda & -\sin\phi\sin\lambda & \cos\phi \\ -\sin\lambda & \cos\lambda & 0 \\ -\cos\phi\cos\lambda & -\cos\phi\sin\lambda & -\sin\phi \end{bmatrix} \tag{5.22}$$

Example

ECEF velocity to NED velocity frame

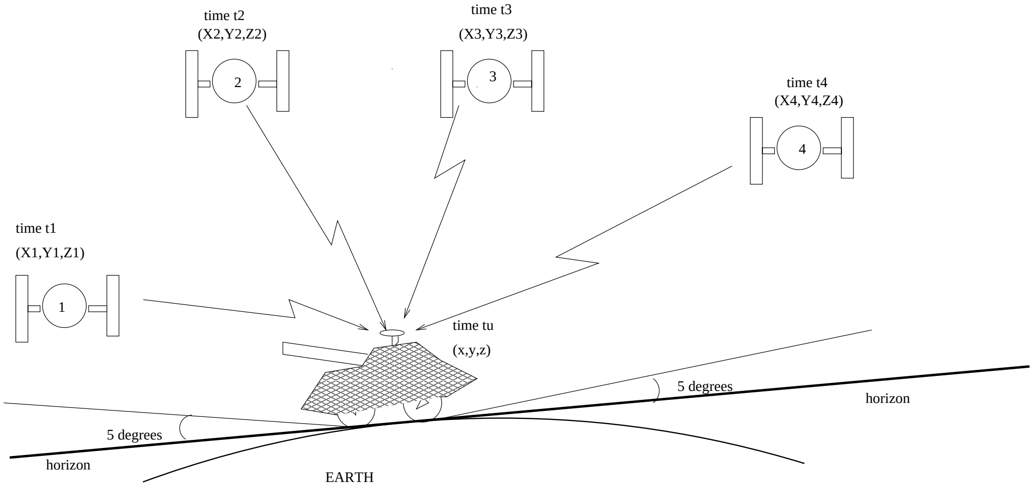
$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_{NED} = C(\lambda,\phi) \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_{ECEF} \tag{5.23}$$

5.4 Position Determination

The position of the receiver is solved by knowing the distance to at minimum four satellites. The distance between receiver and satellites is just to measurethe time if takes for a radiosignal to be transmitted from a satellite to the receiver. Here it is important that the clock in the receiver and the satellite are syncronized. The pseudorange are the geometric distance between the receiver and the transmitter. This geometric distance are calculated in ECEF frame, WGS-84 standard. Transformation from ECEF to NED frame must be done later, in order to use the results in the navigation filter.

If the position solution is determined with the same setup of satellites the solution will be rather smooth. But if different setup of satellites are used this can cause abrupt jumps in the solutions. The accuracy is highly dependent of the satellite positions.

GPS reveicer that is tracking four satellite signals.



Variable	
ρ_i	Pseudorange to satellite i.
c_0	Speed of light
(X_i, Y_i, Z_i)	ECEF position of satellite i at transmission
(x_r, y_r, z_r)	ECEF receiver position at time of reception
b	Receiver clock bias
ϵ_{pi}	Error in pseudorange.
$\mathbf{l_i}$	Direction unit vector between receiver and satellite.
$\hat{\mathbf{l}}_\mathbf{i}$	Predicted direction unit vector between receiver and satellite.
$\hat{\rho}_i(\mathbf{x_k})$	Predicted pseudorange.

From the receiver we get the pseudorange to satellite i. The clock in the receiver are not perfectly syncronized with the satellite clocks so we add a time bias. The distance the wave transmitts over a time b is $c_0 b$.

$$\rho_i = \sqrt{(X_i - x_r)^2 + (Y_i - y_r)^2 + (Z_i + z_r)^2} + c_0 \Delta b + \epsilon_{\rho_i} \tag{5.24}$$

Linearization of pseudorange ρ_i to satellite i gives us equation (5.25) gives us the unit direction vector to the satellite.

$$\mathbf{l_i} = \left[\frac{\partial \rho_\mathrm{i}}{\partial \mathrm{x}}, \frac{\partial \rho_\mathrm{i}}{\partial \mathrm{y}}, \frac{\partial \rho_\mathrm{i}}{\partial \mathrm{z}}\right] \tag{5.25}$$

A very nice property of $\mathbf{l_i}$ is that $|\mathbf{l_i}| = 1$. The partial derivatives are listed below, equation (5.26) to (5.28)

$$\frac{\partial \rho_i}{\partial x_r} = -\frac{X_i - x_r}{\sqrt{(X_i - x_r)^2 + (Y_i - y_r)^2 + (Z_i + z_r)^2}} \tag{5.26}$$

$$\frac{\partial \rho_i}{\partial y_r} = -\frac{Y_i - y_r}{\sqrt{(X_i - x_r)^2 + (Y_i - y_r)^2 + (Z_i + z_r)^2}} \tag{5.27}$$

$$\frac{\partial \rho_i}{\partial z_r} = -\frac{Z_i - z_r}{\sqrt{(X_i - x_r)^2 + (Y_i - y_r)^2 + (Z_i + z_r)^2}} \tag{5.28}$$

We have predicted pseudoranges to the satellites $\hat{\rho}_i(\mathbf{x_k})$ and $\mathbf{x_r}$ is the predicted position of the receiver at time k. The difference between the predicted pseudorange and the measured pseudorange can be put down into formulas.

$$\begin{bmatrix} \rho_1 \\ \rho_2 \\ \vdots \\ \rho_n \end{bmatrix} - \begin{bmatrix} \hat{\rho}_1 \\ \hat{\rho}_2 \\ \vdots \\ \hat{\rho}_n \end{bmatrix} = \begin{bmatrix} \frac{\partial \rho_1}{\partial x_r} & \frac{\partial \rho_1}{\partial y_r} & \frac{\partial \rho_1}{\partial z_r} & 1 \\ \frac{\partial \rho_2}{\partial x_r} & \frac{\partial \rho_2}{\partial y_r} & \frac{\partial \rho_2}{\partial z_r} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \rho_n}{\partial x_r} & \frac{\partial \rho_n}{\partial y_r} & \frac{\partial \rho_n}{\partial z_r} & 1 \end{bmatrix} \begin{bmatrix} \Delta x_r \\ \Delta y_r \\ \Delta z_r \\ c_0 \Delta b \end{bmatrix} + \begin{bmatrix} \Delta \epsilon_{\rho_1} \\ \Delta \epsilon_{\rho_2} \\ \vdots \\ \Delta \epsilon_{\rho_n} \end{bmatrix} \tag{5.29}$$

Equation (5.29) is written in a shorter form with matrixes.

$$\Delta \rho = \mathrm{G} \Delta \mathbf{x} + \Delta \epsilon_\mathrm{p} \tag{5.30}$$

$$\Delta\rho = \begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \vdots \\ \Delta\rho_n \end{bmatrix} \quad G = \begin{bmatrix} \mathbf{\hat{l}_1} & 1 \\ \mathbf{\hat{l}_2} & 1 \\ \vdots & \vdots \\ \mathbf{\hat{l}_n} & 1 \end{bmatrix}$$

$$\Delta\mathbf{x} = \begin{bmatrix} \Delta x_r \\ \Delta y_r \\ \Delta z_r \\ c_0\Delta b \end{bmatrix} \quad \Delta\epsilon_\rho = \begin{bmatrix} \Delta\epsilon_{\rho_1} \\ \Delta\epsilon_{\rho_2} \\ \vdots \\ \Delta\epsilon_{\rho_n} \end{bmatrix} \tag{5.31}$$

Least square solution of position

The position of the receiver can be solved if four valid satellite signals are being tracked by the GPS receiver. The $\Delta\mathbf{x}$ vector have four unknown and we need atleast four pseudoranges to solve the system. This is none with the normal equation (5.32)

$$\Delta\hat{\mathbf{x}} = (\mathrm{G^TG})^{-1}\mathrm{G^T}\Delta\rho \tag{5.32}$$

If the signals have different noise, we use (5.33). Where R is the measurement noise of the signal.

$$\Delta\hat{\mathbf{x}} = (\mathrm{G^TR^{-1}G})^{-1}\mathrm{G^TR^{-1}}\Delta\rho \tag{5.33}$$

5.4.1 Dilution Of Precition

It is clear that the satellite geometry will affect the calculated position. The covariance of the solution is :

$$E[\Delta\hat{\mathbf{x}}\Delta\hat{\mathbf{x}}^\mathbf{T}] = (\mathrm{G^TG})^{-1}\mathrm{G^TRG}(\mathrm{G^TG})^{-\mathrm{T}} \tag{5.34}$$

If the pseudoranges have the same noise and if the noise of the pseudoranges are uncorrolate $E[\Delta\rho_i\Delta\rho_j] = 0, i <> j$ we can replace $\mathbf{R}$ with a scalar $\sigma_R = E[\Delta\rho_i\Delta\rho_i$. With this the resulting position covariance will be

$$E[\Delta\hat{\mathbf{x}}\Delta\hat{\mathbf{x}}^\mathbf{T}] = (\mathrm{G^TG})^{-1}\mathrm{G^T}\sigma_\mathrm{R}^2\mathrm{G}(\mathrm{G^TG})^{-\mathrm{T}} = \sigma_\mathrm{R}^2(\mathbf{G^TG})^{\mathbf{-1}} \tag{5.35}$$

If we list all elements of the matrix (5.35) we get:

$$\sigma_R^2(\mathbf{G^TG})^{\mathbf{-1}} = \begin{bmatrix} E[\Delta x\Delta x] & E[\Delta x\Delta y] & E[\Delta x\Delta z] & E[\Delta x c_0\Delta b] \\ E[\Delta y\Delta x] & E[\Delta y^2] & E[\Delta y\Delta z] & E[\Delta y c_0\Delta b] \\ E[\Delta z\Delta x] & E[\Delta z\Delta y] & E[\Delta z^2] & E[\Delta z c_0\Delta b] \\ E[c_0\Delta b\Delta x] & E[c_0\Delta b\Delta y] & E[c_0\Delta b\Delta z] & E[c_0\Delta b^2] \end{bmatrix} \tag{5.36}$$

The $(\mathbf{G^TG})^{\mathbf{-1}}$ matrix is called the dilution matrix. If we want the dilution in another frame than ECEF we need to do a frame change. The frame change is done with a matrix C. Is can be appropirate to use the dilution in navigation frame. In this case the C matrix will be the equation (5.22). Then the new dilute matrix will be.

$$A = C\,(\mathbf{G^TG})^{\mathbf{-1}}\,\mathbf{C^T} \tag{5.37}$$

Geometric Dilute Of Position Solution

$$GDOP = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2 + \sigma_b^2} = \sqrt{trace(A)} \tag{5.38}$$

Position Dilute Of Position Solution

$$PDOP\sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2} = \sqrt{A_{11} + A_{22} + A_{33}} \tag{5.39}$$

Time Dilute Of Position Solution

$$TDOP = \sigma_b = \sqrt{A_{44}} \tag{5.40}$$

Horizontal Dilute Of Position Solution

$$HDOP = \sqrt{\sigma_x^2 + \sigma_y^2} = \sqrt{A_{11} + A_{22}} \tag{5.41}$$

Vertical Dilute Of Position Solution

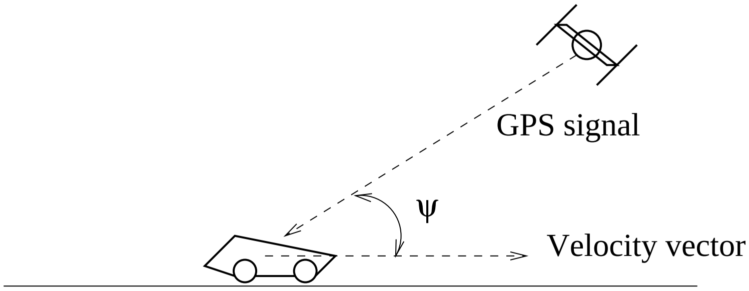
$$VDOP = \sigma_z = \sqrt{A_{33}} \tag{5.42}$$

5.5 Velocity determination

The measured frequency by the receiver will be dependent of the vehicle velocity. The doppler will affect the frequency in the way that if the distance between the satellite and the receiver is decreasing the locked frequency will be higher than the transmitted frequency. We can use this frequency shift to estimate the velocity of the receiver. The velocity of the satellite can be calculated with very high precision. The frequency shift follows equation (5.43)

$$f_r = f_t\left(1 + \frac{\dot{\rho}}{c_0}\right) = f_t\left(1 + \frac{v_r \cos\psi}{c_0}\right) \tag{5.43}$$

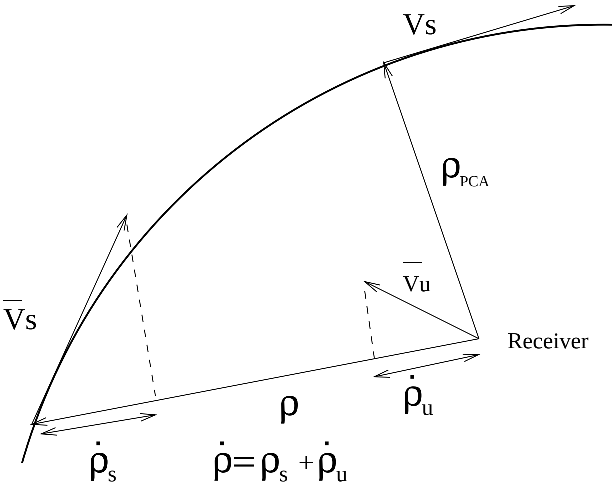
Doppler effect



As we see the dopplershift of the frequency is highly dependent of the angle between the velocity vector and the inline sight vector to the satellite. The model is inded more complex than this. We must take in calculation that the satellite also have a velocity and a insted of $v_r \cos\psi$ we use $(\mathbf{v_r} - \mathbf{v_i}) \cdot \mathbf{l_i}$, where $\mathbf{v_i}$ is the satellite velocity and $\mathbf{l_i}$ is the inline sight vector. Equation (5.43) can be rearranged and we solve $\dot{\rho}$

$$\dot{\rho} = c_0 \left(1 - \frac{f_r}{f_t}\right) \tag{5.44}$$

Pseudoranges and pseudorange rates



If the GPS satellite signal is tracked we can use integration of the doppler shifted frequency to keep track of the actual position, see equation (5.45). If the position is well known and the satellite signals are locked this is a very good way of position estimation.

$$\Delta\rho = \int_t^{t+\Delta t} \dot{\rho} dt = \int_t^{t+\Delta t} (\mathbf{v_r} - \mathbf{v_i}) \cdot \mathbf{l_i} \mathrm{dt} = \frac{\mathrm{c_0}}{\mathrm{f_t}} \int_{\mathrm{t}}^{\mathrm{t+\Delta t}} (\mathrm{f_t} - \mathrm{f_r})\mathrm{dt} \tag{5.45}$$

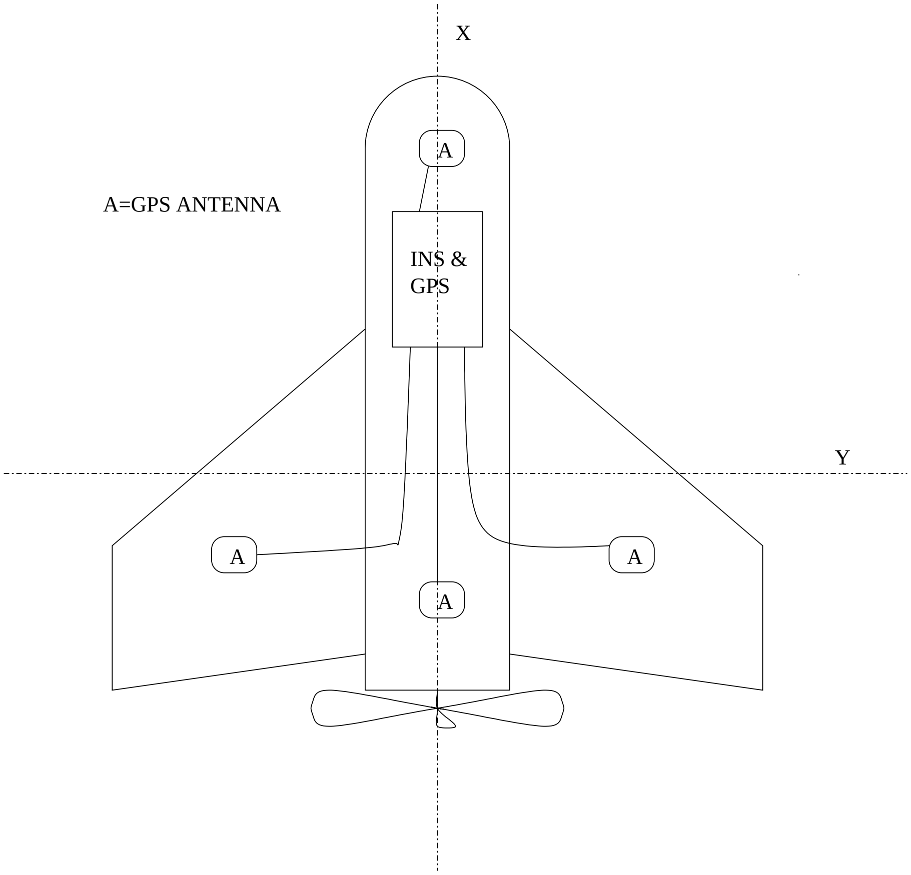
$$\begin{bmatrix} \dot{\rho}_1 \\ \dot{\rho}_2 \\ \vdots \\ \dot{\rho}_n \end{bmatrix} - \begin{bmatrix} \hat{\dot{\rho}}_1 \\ \hat{\dot{\rho}}_2 \\ \vdots \\ \hat{\dot{\rho}}_n \end{bmatrix} = \begin{bmatrix} \frac{\partial\rho_1}{\partial x_r} & \frac{\partial\rho_1}{\partial y_r} & \frac{\partial\rho_1}{\partial z_r} & 1 \\ \frac{\partial\rho_2}{\partial x_r} & \frac{\partial\rho_2}{\partial y_r} & \frac{\partial\rho_2}{\partial z_r} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial\rho_n}{\partial x_r} & \frac{\partial\rho_n}{\partial y_r} & \frac{\partial\rho_n}{\partial z_r} & 1 \end{bmatrix} \begin{bmatrix} \Delta\dot{x}_r \\ \Delta\dot{y}_r \\ \Delta\dot{z}_r \\ c_0\Delta b \end{bmatrix} + \begin{bmatrix} \Delta\epsilon_{\rho_1} \\ \Delta\epsilon_{\rho_2} \\ \vdots \\ \Delta\epsilon_{\rho_n} \end{bmatrix} \tag{5.46}$$

$$\Delta\dot{\rho}_i = \hat{\dot{\rho}}_i - \dot{\rho}_i = \begin{bmatrix} \mathbf{l_i} & 1 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{v} \\ \Delta f \end{bmatrix} + \Delta\epsilon_{\dot{\rho}_i} \tag{5.47}$$

5.6 Attitude determination

The GPS receivers can be used to give periodic position an attitude updates to the INS, Inertial Navigation System. With three receivers and antennas mounted on the frame of the vehicle it is possible to estimate the attitude. This is presented in [36]. By measuring the phaseshift difference between the received signals it is possible to calculate the attitude with a resolution down to 0.5 degrees.

Four GPS antennas placed on the airframe



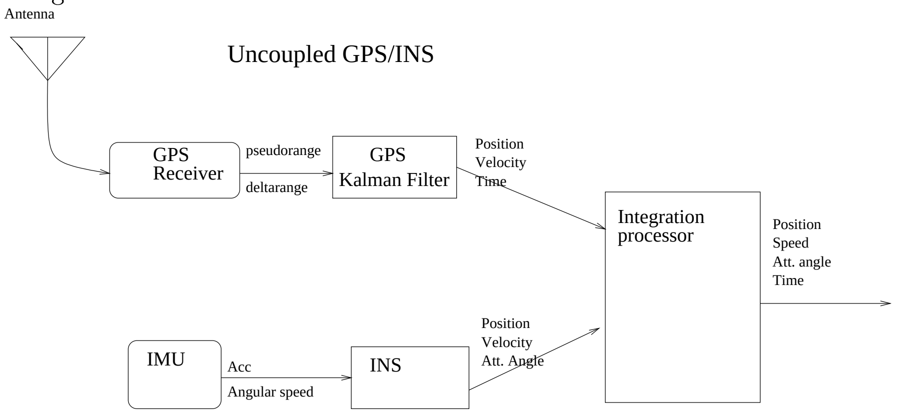
5.7 GPS interface software

The receiver software consists of three modules, the GPS binary, Data I/O and the User Interface. The software can control multiple receivers and output navigation data. With a ground receiver the onboard software can compensate the outputs using Differential GPS technique.

5.8 Coupled INS/GPS

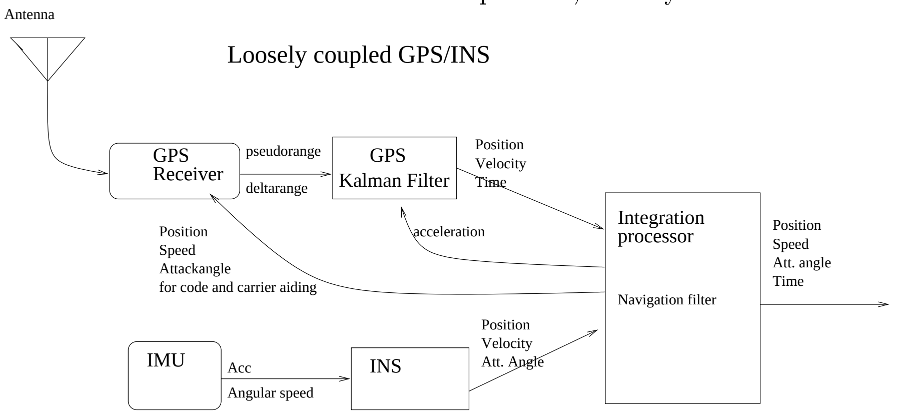
5.8.1 Uncoupled INS/GPS system

Here the information given by the GPS is estimated position and speed of the receiver. This if feed into the integration processor for system update. From the INS we also get acceleration, velocity and position of the system. The information from the GPS and INS is fused together for the optimal estimate using a Kalman Filter.



5.8.2 Loosley coupled INS/GPS system

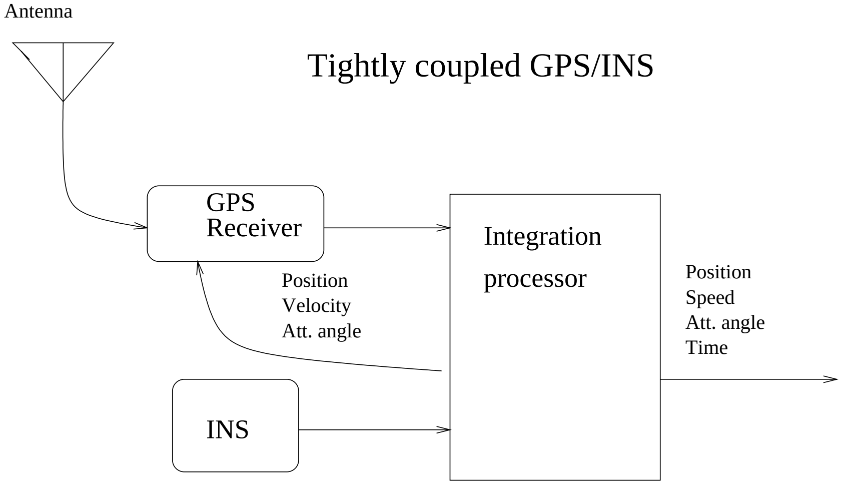
Here we have a feedback loop to the GPS receiver. The external navigation filter have an estimation of position, velocity and attitude. This information is fed to the receiver to aid the internal filter of the GPS. Acceleration is fed to the GPS kalmanfilter that estimates the position, velocity and time.



5.8.3 Tightly coupled INS/GPS system

In this system solution we use pseudorange and pseudorate information from the satellites to aid the navigation filter. This system will be more integrated than the other solutions and the information from less than four satellites will contribute to the accuracy of the position and velocity. The information sent from the receiver is pseudorange and pseudorange rate to different satellites.

Position, Velocity and attitude are fed back to the GPS receiver from the navigation filter strictly to be used for code and carrier aiding.



Chapter 6

The Implementation

6.1 Hungarian Notation

I have used something called hungarian notation. This way of writing the software is very powerful when you need a simple and effective way of identify the types of the variables, classes and the type returned by functions.

Hungarian notation used in the software

Variabletype	Hungarian Notation	Example
array	a	char caName[10]
boolean	b	bool bStatus
char	c	char cTmp
class	C	class Ctmp alt. class Ttmp
double	r	double rTmp
int	i	int iTmp
long	l	long lTmp
float	sr	float srTmp
obejct	o	class Ctmp oTmp
pointer	p	char *pcName
short	s	short sTmp
struct	S	struct Stmp
unsigned	u	unsigned int uiTmp
void	v	void vTmp(int i)

The letters i,j,k are used for indexing in nested loops. If more index variables are needed they should be defined as the rules above. Variables of the same type begin with the letter decribing the type and then a number. Like if you need two varibles of type double, r1 and r2. Constants are defined with capital letters like the math constant π should be represented by MATH_PI.

6.1.1 C++ example using hungarian notation

```
class Ccomplex
{
    double rImag, rReal;
  public:
    Ccomplex(double rAbs, double rArg);
}

Ccomplex::Ccomplex(double rIm, double rRe)
{
    rImag = rAbs*rSin(rArg);
    rReal = rAbs*rCos(rArg);
}

void main( void)
{
   Ccomplex *poNumber= new Ccomplex( 0.1,0.1);
   delete poNumber;
}
```

6.2 The Software

6.2.1 The flow of the software

The main program of the navigation loop is a Information Filter written generic in C++. The filter estimates the errors in position, velocity and attitude. The prediction step uses the states given by the INS to calculate the system matrixes. The filter matrixes are calculated inside an object that represents the vehicle, the observation models are updated if needed.

The software flow

- Read measurements from the IMU hardware.
- Convert to true accelearations and rotationrates.
- Transform body acceleration and rorationrate vectors to navigation frame.
- Velocity and position integration. Update attitude.
- Exists external GPS observations ? If yes:
  - Add observations to error filter.
  - Estimate position, velocity and attitude.
  - Feedback errors to the INS.
  - Clear errors in error filter.
  - Calculate the new state transition matrix, calls the vehicle model.
  - Predict 0.1s ahead, external observations with 10Hz rate.

6.3 Classes

6.4 For The IMU

6.4.1 The Sensor Class

- Set reference temperature
- Set sensor temperature
- Set bias
- Set scale factors
- Set standard deviation
- Set normal vector
- Get normal vector
- Get scale factor
- Rescale

6.4.2 The Sensor Cluster Class

This class contains several instances of the Sensor Class. The number of sensors in each class can be choosed arbitary. An IMU normally have a cluster of accelerometers and one gyro-cluster. This class contains the important functions to setup the sensors in a sensor frame.

6.5 The Readdata Class

- Read Watson IMU data file
- Read Tetrad IMU
- Read Tetrad IMU data file
- Read Motion-Pack IMU data file
- Read GPS

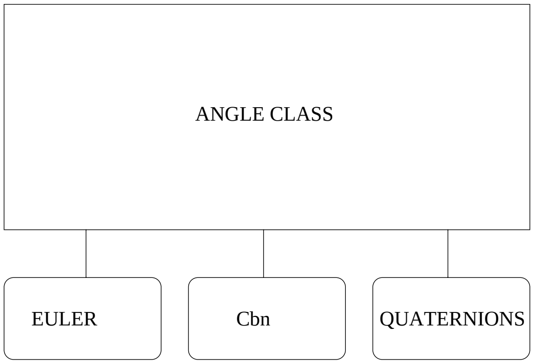
6.6 The INS Class

Class that have the implemeted functions to keep track of a vehicle, navigation. The input to this class is from the IMU and error feedback from the error filter. The filter contains a lot.

- Set
  - Set acceleration
  - Set velocity
  - Set position
  - Set attitude
- Get
  - Get acceleration
  - Get velocity
  - Get position
  - Get attitude
- Error feedback

6.7 The Angle Class

This Class contains a lot of virtual functions that are used for dataconversion between different angle representations. The angle class are mainly used in the code but the subclasses Eulerangle, Cbn and Quaternion classes are those who are created as objects in the program. All three classes have been implemented and gives almost the same result, but the Eulerangle class is the most accurate because it uses a fourth order Runge-Kutta integrator to keep track of the angles. The picture below gives an idea how the implementation is done.



6.7.1 The Eulerangle Class

There are several methods to keep track of the angle, one of them is simply to transform the body turn rates (pqr) to navigation frame which gives us omega, see equation (2.39), this is integrated with discrete integrators and the Runge-Kutta fourh order integrator is used as default. However integrators with Trapetz and Simpsons Rule can also be used with an additional switch if lover accuracy is prefered.

6.7.2 The Cbn Class

This class keeps the rotationrates in bodyframe and uses equations (2.32) or (??) to update the Cbn matrix.

6.7.3 The Quaternians Class

In this class equations (2.58) or (??) are used to update the Quaternion angles.

6.8 The Linear System Class

The basic class for the filters is the linear equations presented in section 4.2. This class implements all equations needed to use a linear equation system.

6.8.1 The Kalman Filter Class

This class is derived from the linear system class and implements the Kalman Filter in section 4.5.

6.8.2 The Information Filter Class

Derived from the linear system class. The class implements the information filter presented in section 4.6

6.8.3 The Innovation Class

Functions implemented to operate on the innovations. Functions to calculate eigenvalue, eigenvectors, standard deviatons. χ-square distribution tests.

6.9 The Vector Integration Class

6.9.1 The Trapetz Vector Integration Class

Implementation of the trapetz integration rule.

6.9.2 The Simpson Vector Integration Class

Implementates the Simpson rule of numerical integration.

6.9.3 The Runge-Kutta Vector Integration Class

Fourth order numerical integration with an error of order $O(h^5)$.

6.10 The Model Class

The idea with this class is to make it very simple to change the vehicle model without doing a lot of changes in the existing sourcecode. By putting isolating the model within a class we just have to derive a new subclass of the Cmodel class to test a new model in the software. The importand functions are declared as virutal and must be defined in the new class, because they are very model dependent. Three different models were implemented and tested in this thesis.

- Linear Class
- Eulerangle Class
- Quaternionangle Class

The header file of the Cmodel class looks like this.

```
class Cmodel
{
public:
  DoubleMatrix oF;
  DoubleMatrix oH[16];
  DoubleMatrix oG;

public:
  Cmodel::Cmodel(double rSampleTime);
  Cmodel::~Cmodel();
  virtual void vCalcF( void);
  virtual void vCalcG( void);
  virtual void vCalcHn( int n=0);
  virtual void vUpdate( DoubleVec oX, DoubleVec oU);
  virtual bool bNeedUpdate( void);
};
```

- Calculate F matrix
- Calculate G matrix
- Calculate H matrixes

6.10.1 The Quaternions Model Class

This class is a subclass of Cmodel, and here the Quaternion model in section 4.9.

6.10.2 The Eulerangle Model Class

This class is a subclass of Cmodel, and here the Quaternion model in section 4.8.

6.10.3 The Linear 9 State Model Class

This class is a subclass of Cmodel, and here the Quaternion model in section **??**.

6.11 Other Classes

6.11.1 The Vector Statistics Class

This class contains simple statistical functions to operate on the DoubleVector class. Functions that calculates the mean, standard deviation and variance is implemented.

6.11.2 The Fast Trigonometric Class

6.11.3 The Earth Class

6.11.4 The Serial Communication Class

Initially I worked with the Tetrad IMU. The communication with the IMU is done through a serial interface at 115200 Baud and I developed this class to be able to have different instances for different serial communication ports. With this class you create instances for the different serial communication ports and chooses the desired baudrate, parity, port number and bit length.

6.11.5 The Firmware Class

To commuicate with the Tetrad IMU you need a very special communication protocol. This most basic commands for this protocol is found in table **??**. With the Tetrad you can communicate over a CAN-bus interface and a serial interface. At the moment if a instance of this class is created the class communicates over a with a serial port instance that is created.

6.11.6 The Logging Class

A class for logging have been made, this class

The $Matlab^{TM}$ Logging Class

This class is derived from The Logging Class, and is contains functions to print vectors and matrixes to logging files.

6.12 The target computer

The hardware in the airframe so far is a single computer, see section 8.1.2. To this computer all the sensors and receivers are connected, total four GPS receivers and at maximum two IMU's. At the moment the solutions is postprocessed from datafiles that have been logged during the testflight.

Chapter 7

Results

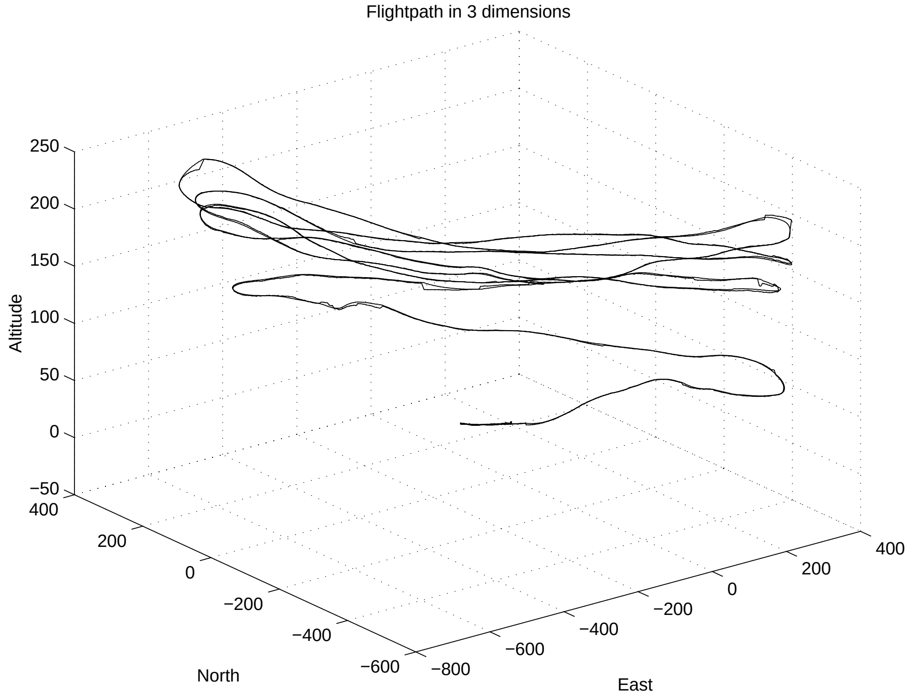
7.1 Testflight

The testflight was taken place on a farm north of Goulburn in NSW. The farm is owned by the University of Sydney, the farm area is very large and will be perfect for testflights as the project proceed.

Brumby takeoff

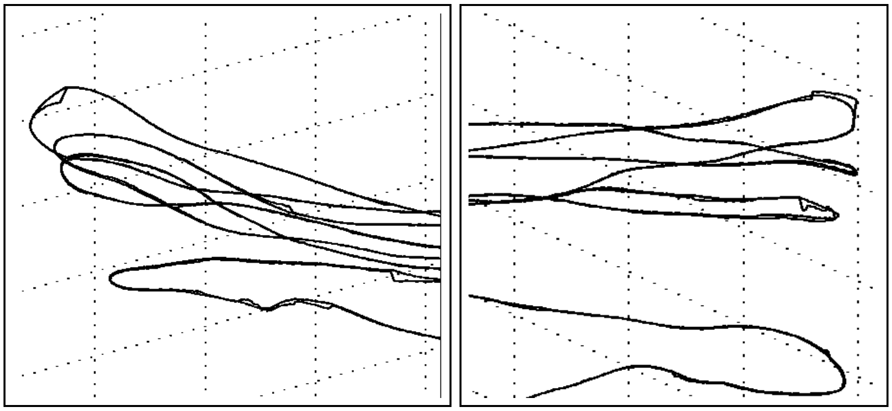
There was a flight which the inertial system and the GPS in November 1999 and the flight was named the CCD FLIGHT. During this flight measurements was collected and stored and in some ways processed by the onboard computer system based on the PC104 standard. Under previous flights there have been some problems to store data on a physical harddisc. In this flight the information was stored in RAM discs. The flight was successful and data from the flight could be downloaded through a TCP/IP connection to a external computer and stored on harddisc. The GPS/INS data was later processed and I wrote a matlab script to show the trajectoria flown during the testflight. The aircraft position and velocity were calculated using the data from the GPS receiver. The figures show the aircrafts position in navigation frame, using NED frame (North, East and Down).

Flight path plotted in three dimensions

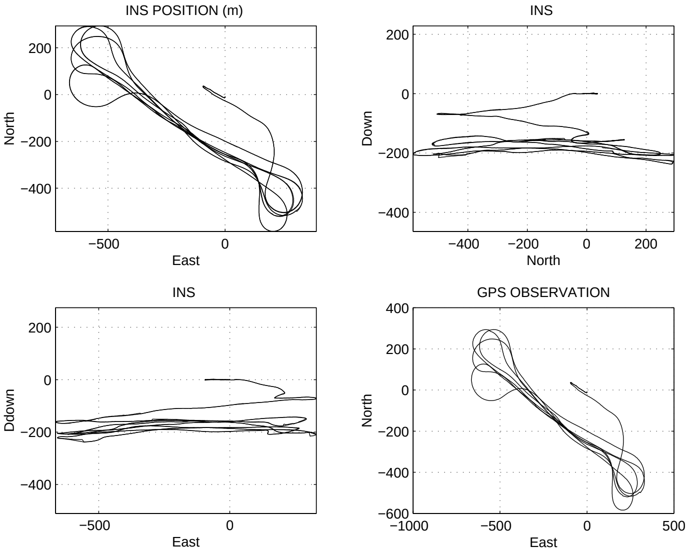


3D flights zoomed at turns

The 3D visualization of the flight contains both the GPS observations and the esitmated position of the UAV. In these figures zoomed from the one above we can clearly see differences in the curves.



Flight path plotted in two dimensions



7.2 The Results

The logged data from the testflight were processed by the navigation filter software. This software creates several outputs that are later processed by $Matlab^{TM}$ scripts to create the figures presented in the following sections. The implemeted filter estimates the position of the UAV and a special $Matlab^{TM}$ script was written to simulate the flight using this data.

7.3 Tetrad and Mpac IMU solutions

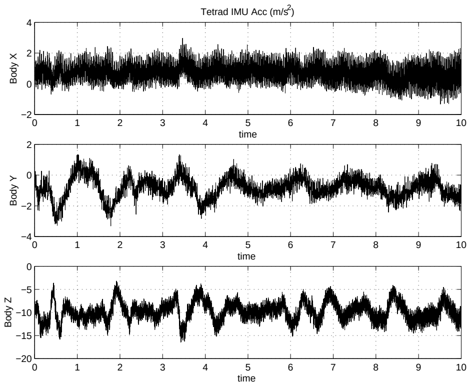
In one flight two IMU's where mounted in the airframe. The Motion-Pack IMU is the standard IMU in the airvehicle. The Tetrad is a IMU under developement and the researchers needed some real data. The Tetrad IMU was mounted in the airframe together with the Motion-Pack IMU. The idea with this was to calculate how accurate the Tetrad are in a real application compared with a reference IMU. The logged data are used to calculate the acceleration and rotationrate in body frame.

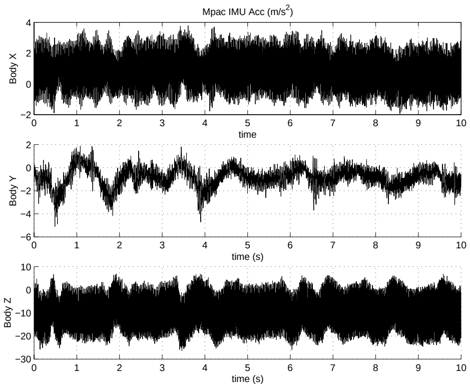
The figures shows a ten secound slice of the calculated acceleration and rotation rates.

7.3.1 IMU accelerations

As we see here the Tetrad IMU is very sensitive. The measurements have a less of noise in the signal compared to the Motion-Pac IMU. The Tetrad are mounted on vibration dampers and is more heavy than the Motion-Pac IMU, this is a mechanical low-pass filter. The Motion-Pac is lighter and follows the oscillation of the airframe and it can also be that the airframe resonates at the point where this IMU was placed.

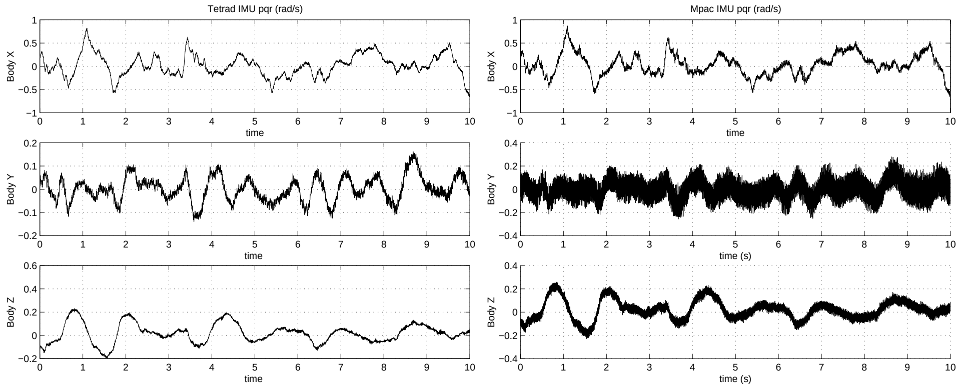
But overall, we can clearly see that the signals follows the same average trajectoria.





Rotationrates compared from two IMUs

The Tetrad-IMU was mounted on vibrationdampers. The angle oscillations will be damped by the dampers and that the Tetrad-IMU infact is very heavy. The Motion-Pac IMU is lighter and follows the motion more and is therefor it can be more sensitive. The noise may be caused by the ADC card and do not need to be mechanical, electrical effects like disturbance from the electrical ignition of the engine.



7.4 Navigation filter results

The errorfilter written in this thesis is in informationform and uses a nonlinear state space model. This model is linearized after every estimation step of the Kalmanfilter. The external observations feed to the filter is GPS position/velocity and attitude of the aircraft. One way to measure the error between the filter and the observation is the innovation. Here I present the the innovations of the observations. The solid line in the plots are the $2\sigma_{x_n}$ boundarys calculated from state matrix P. The outer dashed line represents the 2σ boundary calculated from the innovation covariance matrix.

The following standard deviations were used

SIGNAL/OBSERVATION	STANDARD DEVIATION
IMU PQR	$3.2/\sqrt{40.0}$ Degrees/s
IMU ACCELERATION	0.4 m/s^2
GPS POSITION	3.0 m
GPS SPEED	2.0 m/s
GPS ATTITUDE	2.0 Degrees

Standard deviations used for the low pass filtered data

SIGNAL/OBSERVATION	STANDARD DEVIATION
IMU PQR	$1.2/\sqrt{40.0}$ Degrees/s
IMU ACCELERATION	0.1 m/s^2
GPS POSITION	3.0 m
GPS SPEED	2.5 m/s
GPS ATTITUDE	2.5 Degrees

7.5 IMU-Inertial Measurement Unit

7.5.1 PSD of Inertial Sensors

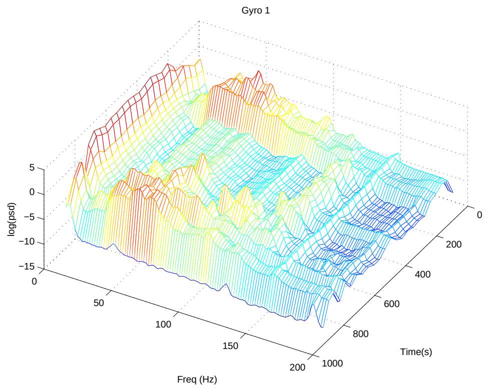
An interesting aspect of the sensors is to know how they are affected by systematic error such as vibrations and misalignment. The IMU with the inertial sensors is rigid mounted inside the airframe and even if the IMU is very well isolated from vibrations with vibration dampers, the sensors will measure lot of the engine vibrations and the vibrations caused by the eigenfrequencies of the airframe. My idea here is to use FFT[1] with a time axis and calculate the PSD "surface" for each sensor. In this flight the MotionPac-IMU, datasheet 8.1.3 was used. The PSD results can be seen in the section 7.5.1. The figures reveals a trace of the engine rpm clearly in atleast three figures: gyro 2,acceleromter 1 and accelerometer 3. With some imagination we can see that the engine rotation per minute is ≈ 60*50 to 60*130 rpm. Between zero and ≈ 150 seconds we can see heavy signals in the low frequency band, 50Hz and below. This is repeted at 600 seconds during landing and this is probably both the engine at low rpm and the aircraft taxing on the runway. The runway at the farm in Goulburn far from perfect. It is not prepared in anyway and is only a flat grassfield and can not be compared with a standard runway. The airframe is rater light and the wheels of the aircraft is small so even small differences in the grassfield will be measured by the IMU.

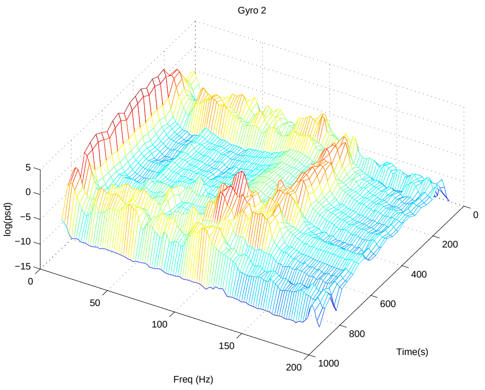
From this I think that the engine pistons are placed in vertial direction, because pitch angle have heavy signals and and the same for the vertical accelerometer.

PSD of the rate gyros

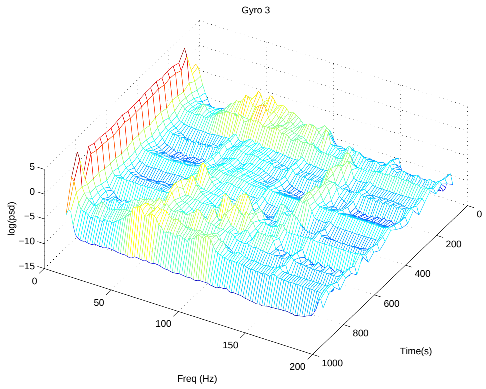
- Gyro 1 - x-axis
- Gyro 2 - y-axis
- Gyro 3 - z-axis

The PSD of gyro 2 we can see the oscillations, strong signals in the low frequency, when the aircraft is taxing on the ground. Flips with the wings because of a rough grassfield.



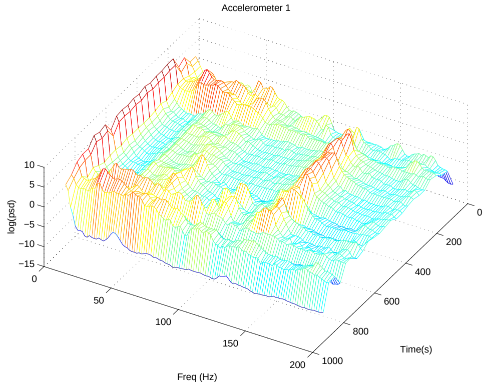


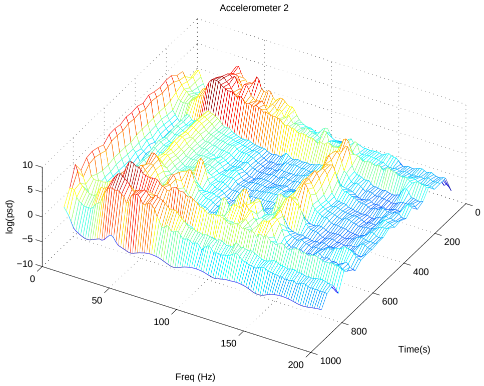
[1] Fast Fourier Transform

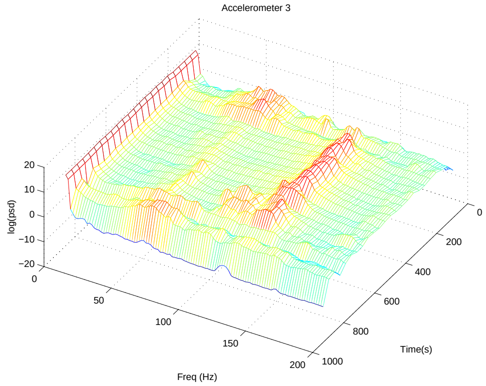


PSD of the accelerometers

- Accelerometer 1 - x-axis
- Accelerometer 2 - y-axis
- Accelerometer 3 - z-axis

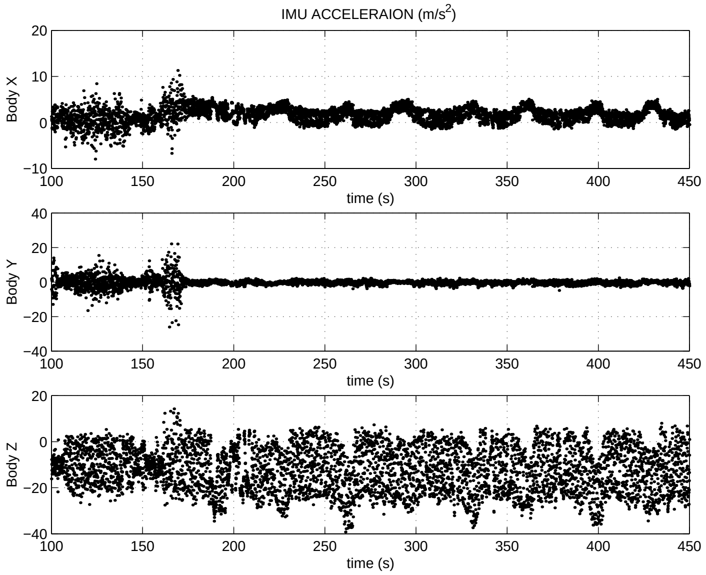






7.5.2 IMU Acceleration Solution

The largest acceleration is in the z direction. This is airframe up/down. At approximate 170 s we see accelerations in x from acceleration during takeoff. The z-axis acceleration is more hidden in noise.

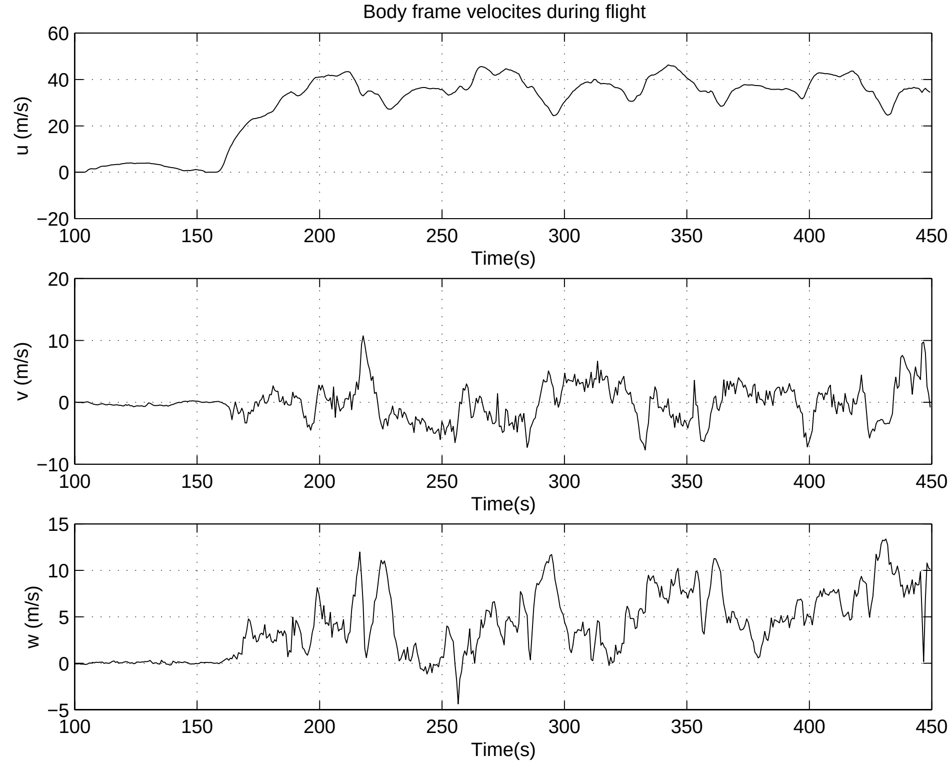


7.5.3 Body frame velocity

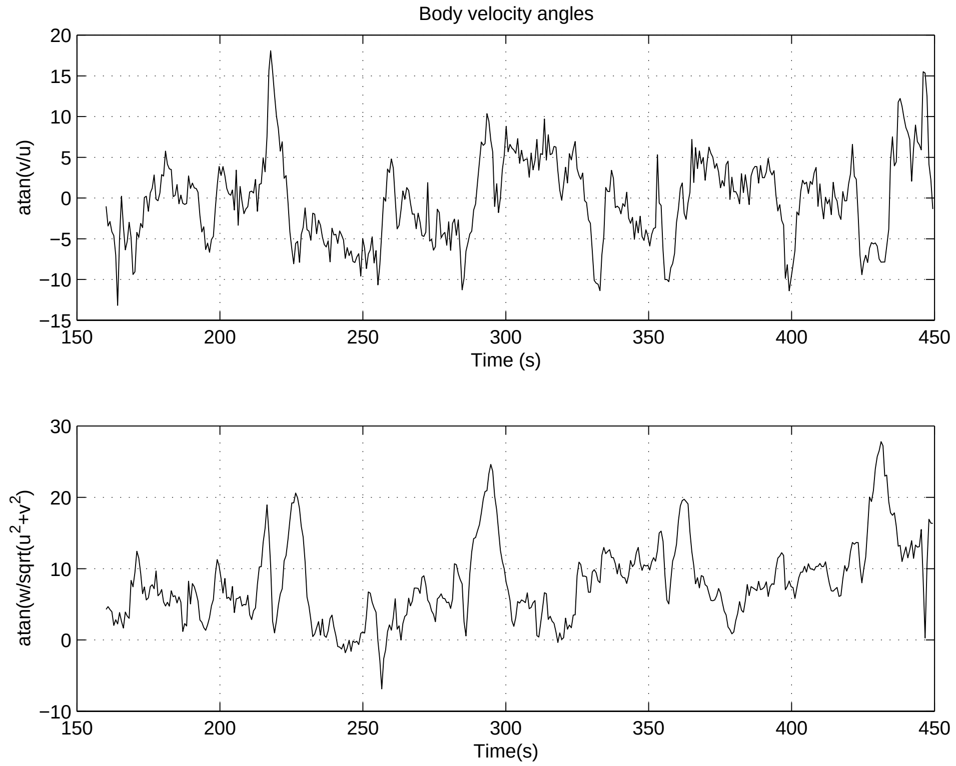
The velocity of the airvehicle can be given in bodyframe. The forward velocity $[u, 0, 0]$ have the highest magnitude. I thought that the sideway velocity $[0, v, 0]$ should be zero, but as the aircraft does sharp turns this will affect the aircraft to slip sideways. The aircraft has a up/down velocity $[0, 0, w]$ because the aircraft is flying with an angle of attack. If the aircraft is flying to slow the aircraft will stall and the down velocity will increase rapidly.

The maximum speed of the airframe is 185 km/h. From the figure we can see that the velocity u reaches a magnitude of about $45 * 3.6 \approx 160$ km/h

Body frame velocity

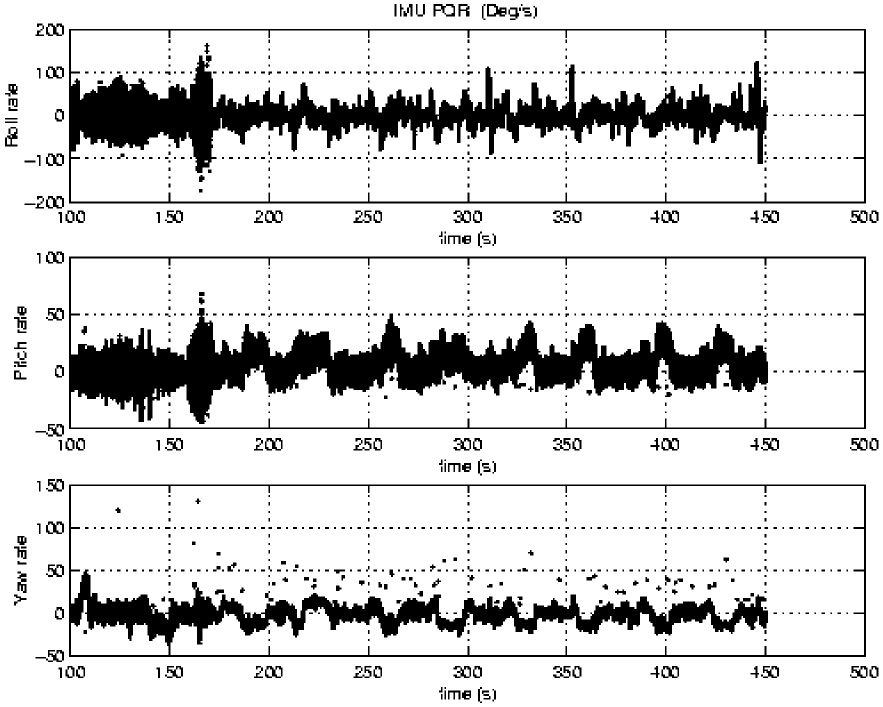


7.5.4 Body frame velocity angles, slideslip and angle of attack



7.5.5 IMU Rotationrate Solution

The rotationrate p and q looks quite ok except for hugh noise in the measured signals. The rotationrate r, nose left/right, have some nasty spikes, and they are all distributed as clockwise rotation. The spikes can be a big problem if the gyros are modelled with low noise. The spikes may be caused[2] by the ADC card used by the PC104 onboard computer. Different methods can be applied to eliminate the spikes. The filter as it is will compensate this “noise” and will be eliminated through the error feedback to the INS.



7.5.6 INS-Inertial Navigation System

The Inertial Navigation System provides the user and navigation filter with

- acceleration
- velocity
- position
- angle velocity
- attitude

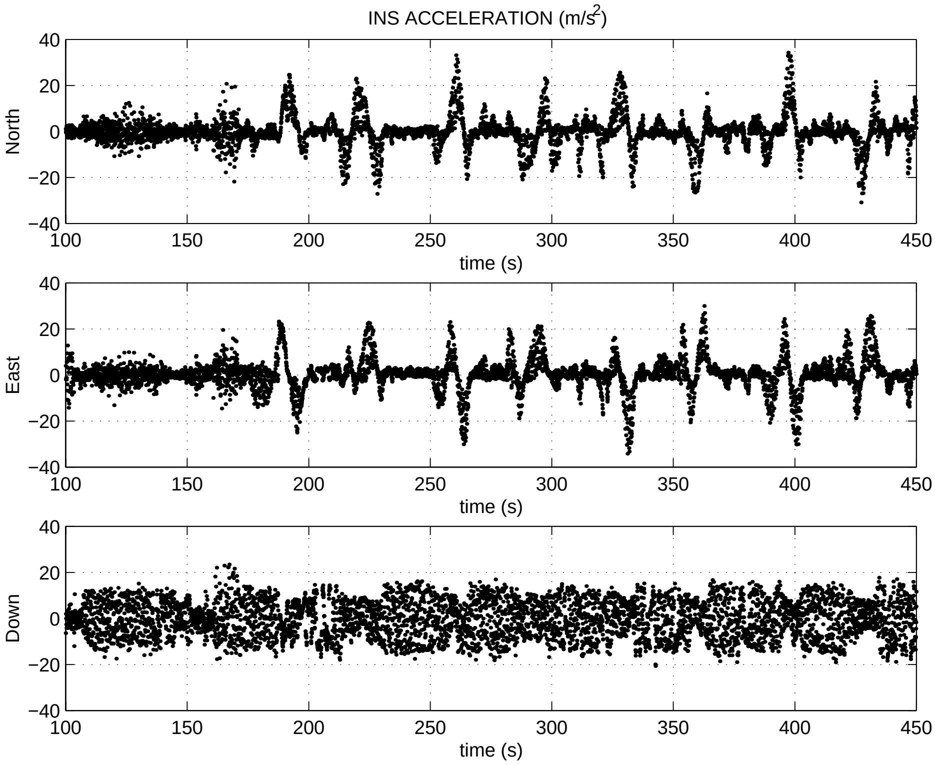
[2] Ben Grosolsky at the Aeronautical Deparment said he had some trouble with reading the status flag from the ADC card

7.5.7 INS acceleration

This is the acceleration solution provided by the INS. The acceleration is calculated using the IMU acceleration and subtracting the gravity vector and centripetal acceleration.

In the figures we see the acceleration in north, east and down direction. Here the vehicle is view as a rigid body under these accelerations in an inertial frame. The dynamics of the airframe have been cancelled at this stage.

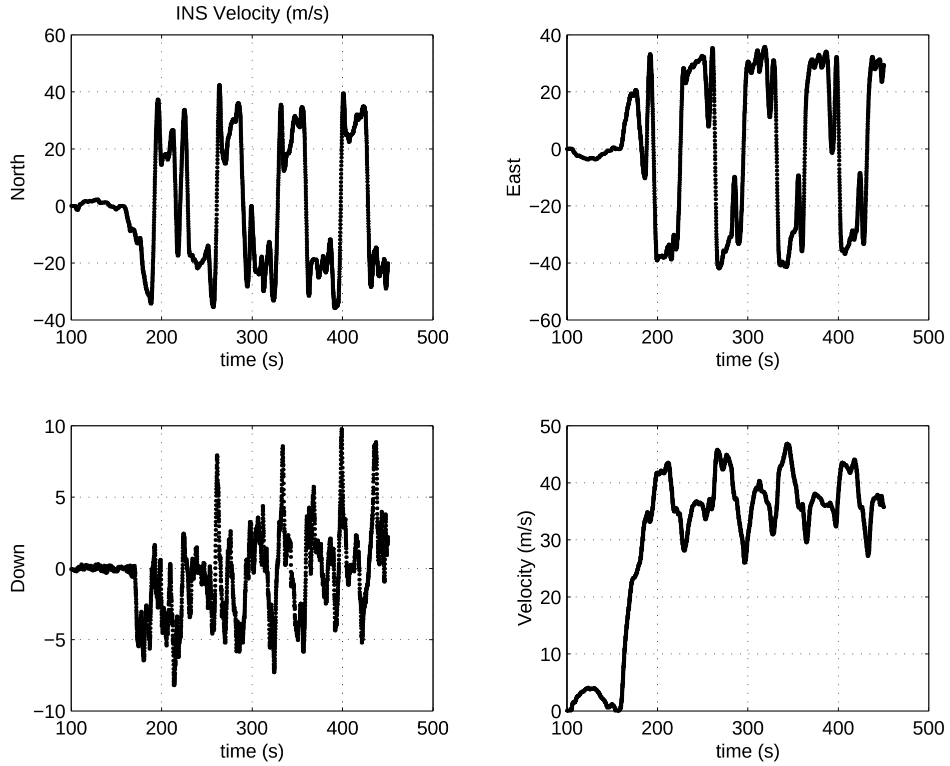
Navigation frame acceleration



7.5.8 INS velocity

From these figures we can see that the UAV flies in a symetric path. It is very hard to visualize how the flight path look like in tree dimensions. The last figure is the absolut value of $[u, v, w]$, as we see the velocity reaches the same magnitude as the velocity in body frame.

Navigation frame velocity



7.5.9 INS omega, eulerangle velocities

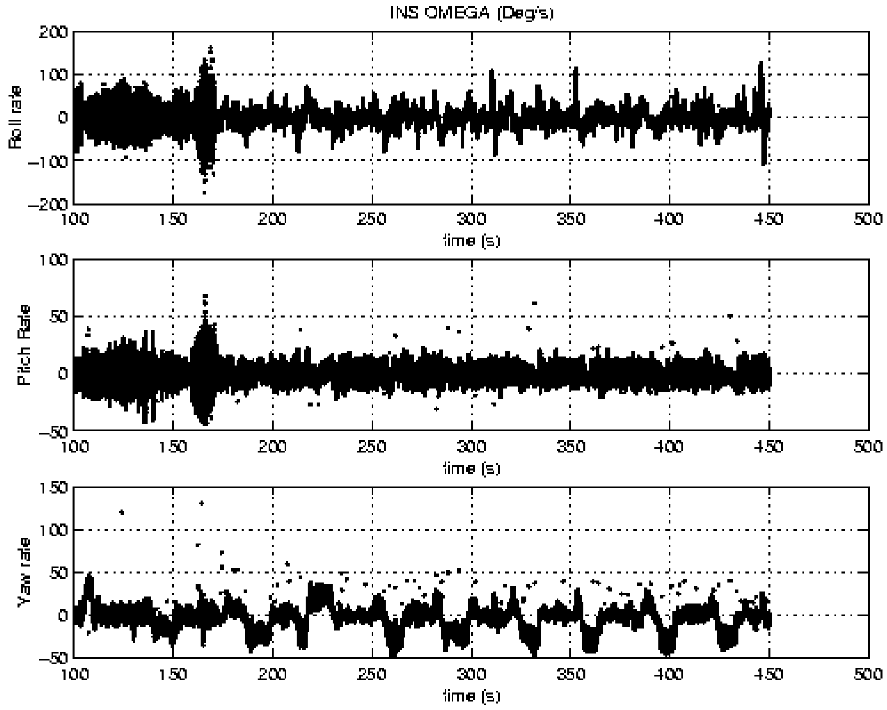
This is the angle velocities of the vehicle in navigation frame. We see that the spurioses from the body frame rotationrate $[0, 0, r]^T$ have been transformed down and are visible in the pitch and yaw rates. From the yaw rate we see that the vehicle almost always turns to the left.

Left turn in navigation frame

1. Starts with wings in horizontal position.
2. Tips the wing to the left (negative yaw velocity).
3. Does a left turn and then back
4. Rotates the wings back to horizontal position (positive yaw velocity).

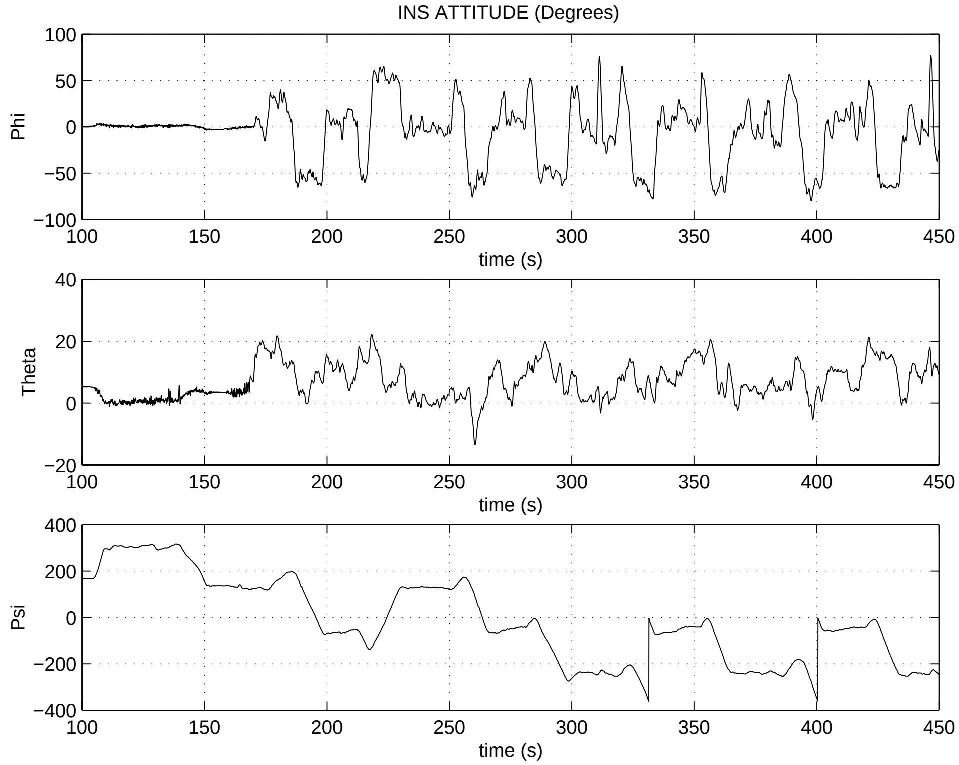
Tips the wing to the left (negative yaw velocity). to horizontal position.

INS frame angle velocities



7.5.10 INS attitude

When ϕ is plotted over time we see that sometimes the angle drops to -50^o. This is when the aircraft does a sharp turn. At some points the roll is near -90^o and then the wings are almost in vertical position. At 360 and 400 s we can see very sharp turns and that the z-axis accceleration $[0, 0, a_z]_{BODY}$ have a very high magnitude here.



7.5.11 The Position Innovation

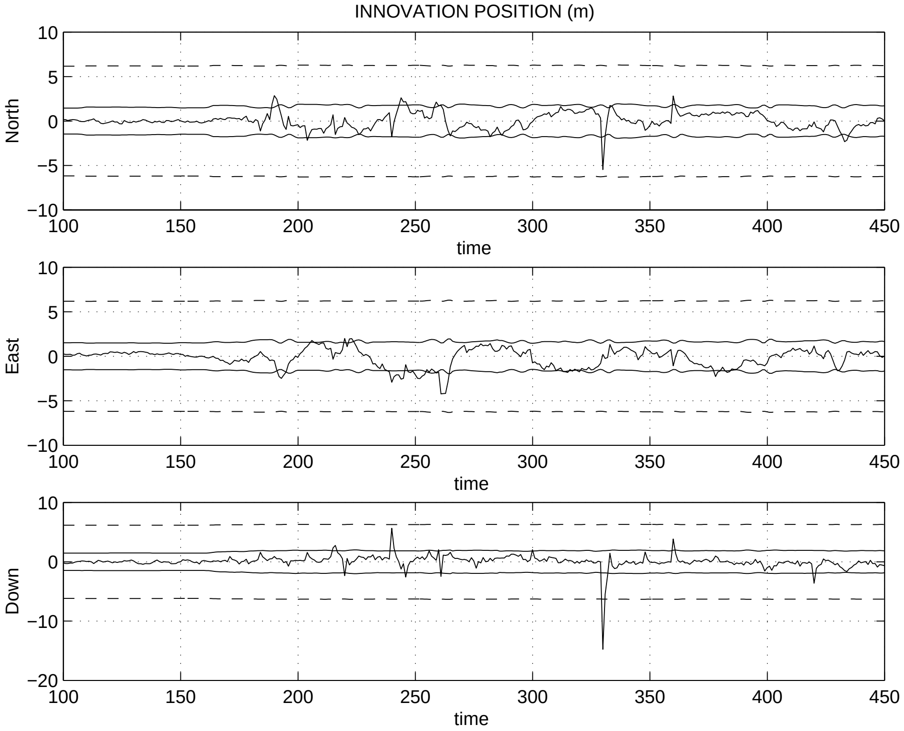
A good innovations signal is white noise. No bias and all frequencies representeed. The innovation looks very good but it seems like the GPS receivers looses track of one or more satellites during the flight, especially in the sharp turns. This can explain the "jumps" in the GPS solution. This observation noise is not normaldistributed in some parts and this is a problem. Perhaps this noise can be modelled with Markow Chains.

Reasons for bad GPS solutions

- Bad GPS signals, to much noise.
- Lost track of one or more satellites.
- Begin to track more satellites.
- The measurement noise of the pseudoranges are not gaussian.

If you look at the figure below and say at 330 s, you can see a heavy spike that is visualized in section 7.6. This innovation spuriose can be masked away if a tighter mask is used. Now the limit is at 3σ. If 2σ was used this would probably be masked away.

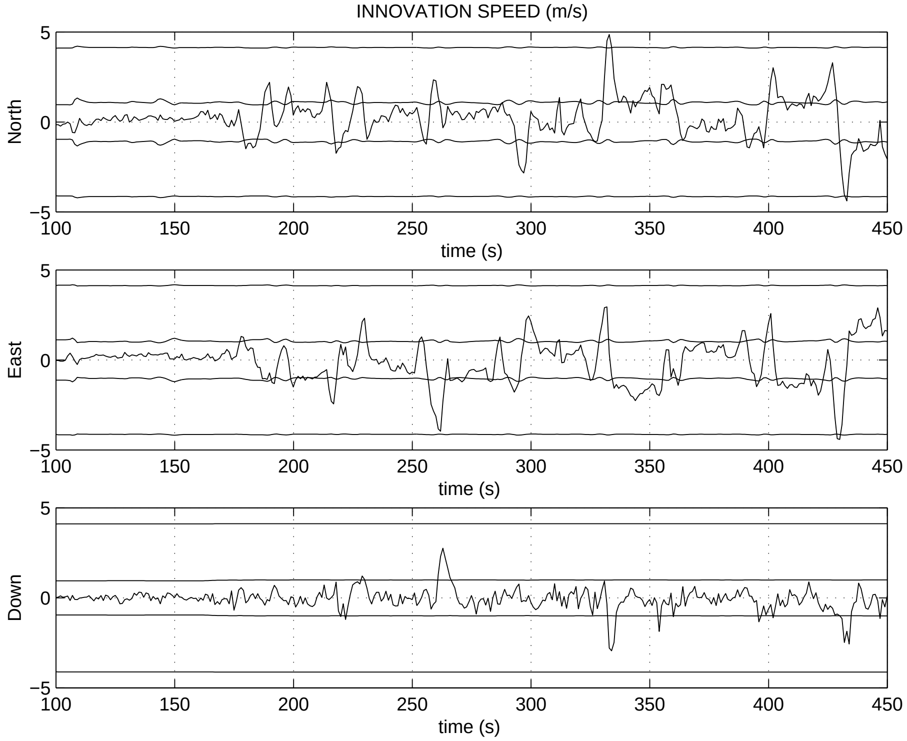
Position Innovation



7.5.12 The Velocity Innovation

The vehicle velocity are measured by the GPS receivers and are a very accurate, down to a resolution of 0.02-0.03 m/s. It seems that there were some kind of lag in this information and an higher standard deviation was put on these observations.

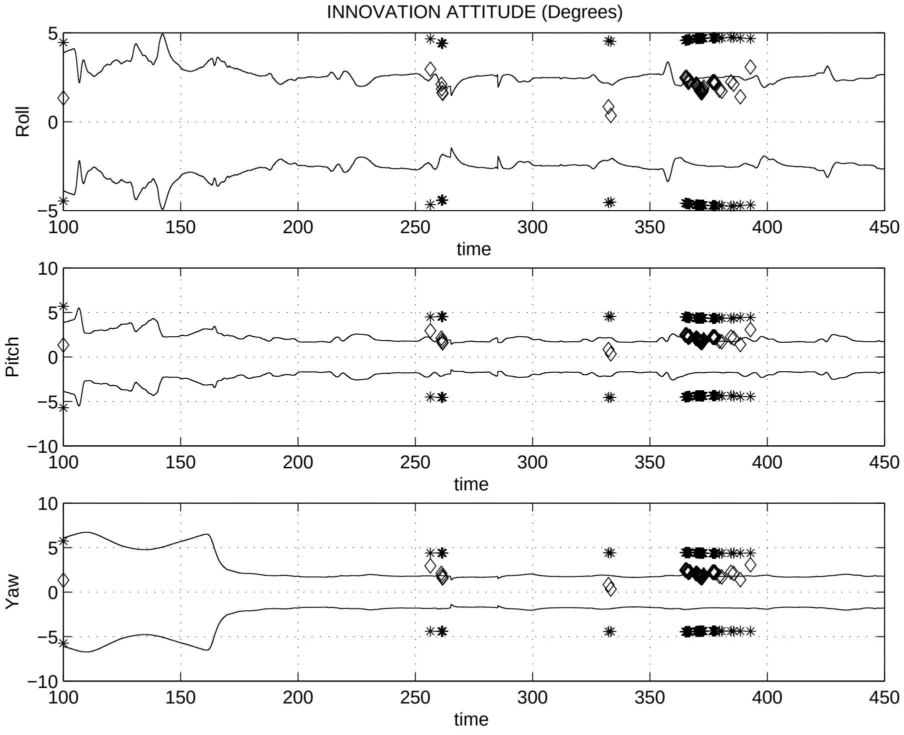
Airvehicle Velocity Innovation



7.5.13 The Attitude Innovation

On Brumby airframe,see 1.2.2, four gps antennas are placed all over the airframe. One antenna on each wing, one near the nose and one at the back. With these four antennas and the GPS receivers and some software it is possible to calculate a GPS attitude solution, look in **??** for more information. The attitude observations are added to the INS error filter after takeoff. The reason for this is that the airframe bounces a lot during taxing and takeoff. Only a few attitude observations are feed to the filter. The attitude algorithm rejects a lot of solutions that are obvious wrong. In the figure below we have some cluster of observations at 260s and 360s. We can see from the figure that the attitude angles of the filter are not affected that much by the GPS observations. The GPS attitude observations seems to be two degrees off in roll, pitch and yaw. My explanaition is that the IMU or the GPS antennas are not calibrated against the airframe.

Airvehicle Attitude Innovation

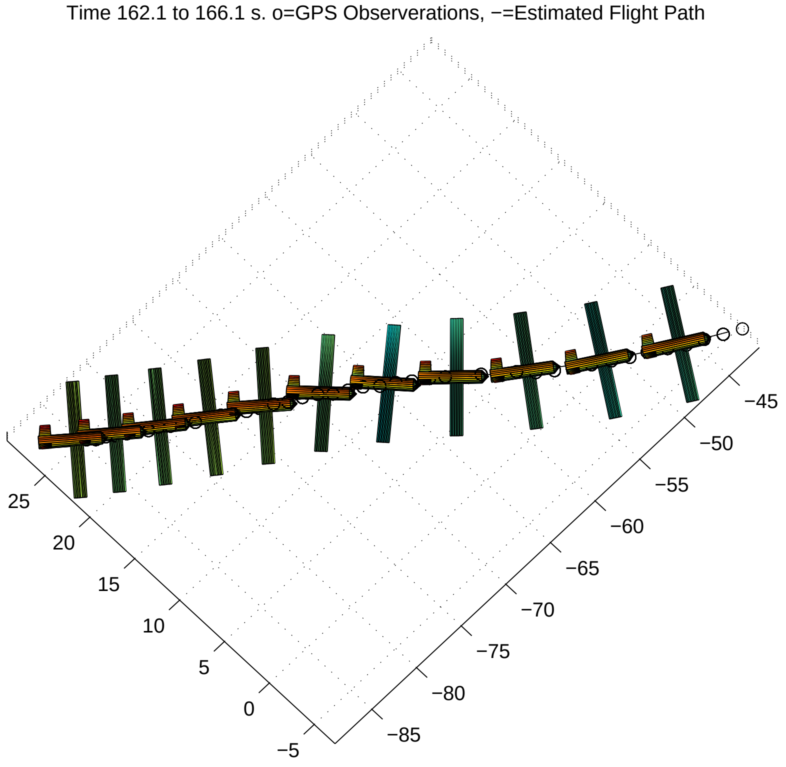


7.6 The Flight Playback

A $Matlab^{TM}$ script was written to do playback simulation of the flight. The inertial sensors and the GPS information are logged to harddisk during the flights. This data have been preprocessed and then run through the navigation filter. The output from this filter is used for the flight playback from takeoff until landing. This gives an idea how accurate the produced filterdata are. It can be very hard to visualize data just in figures. The following pictures shows interesting parts of the flight.

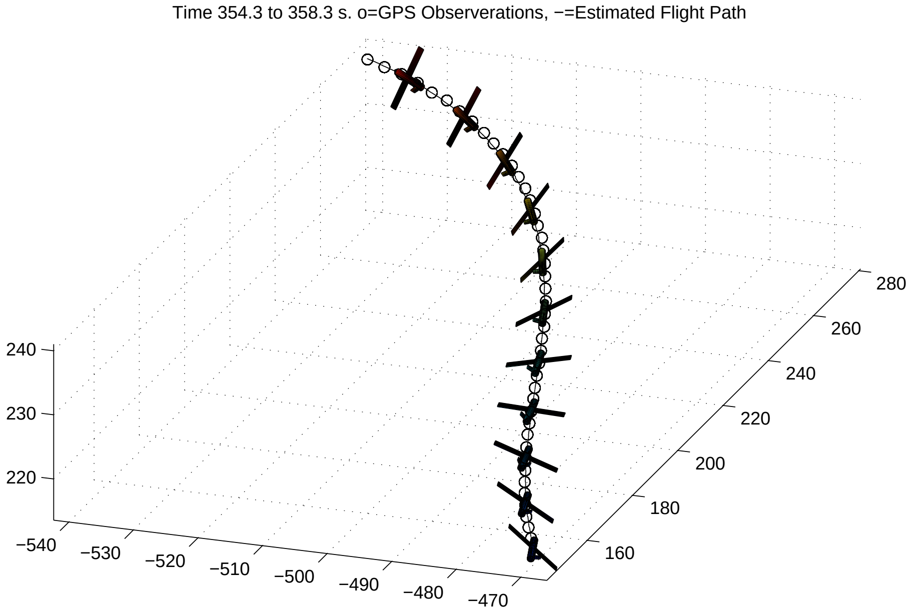
Flight playback at 162s when Brumby is accelerating for takeoff

Here we see the aircraft accelerating on the ground for takeoff. There is a small right turn and then correction, and my idea here is that this is from the spikes that are in the IMU data, in z-direction.



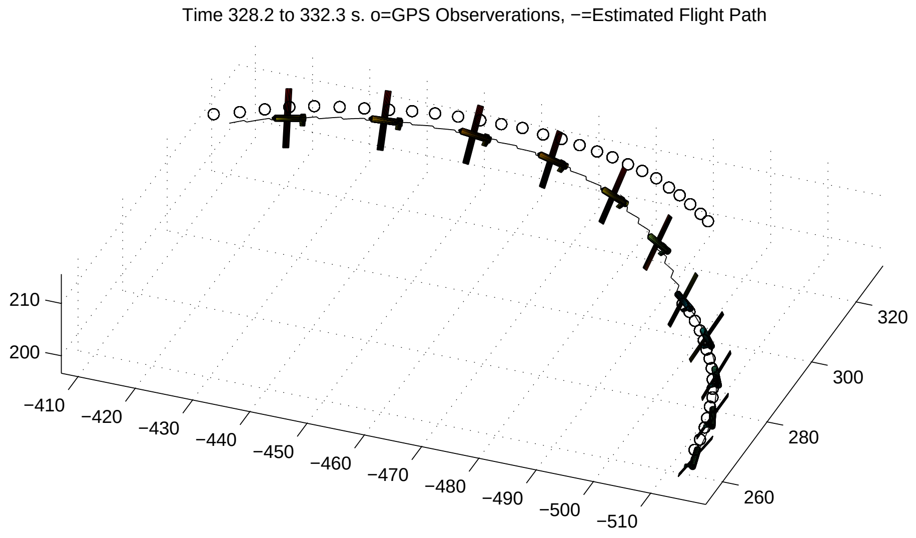
Nice left and right turn with climbing

Here we see the aircraft first in a right turn and rolls over to a left turn and climbs at the same time. This section contains good GPS observations and therefor the estimated trajectoria is very good.



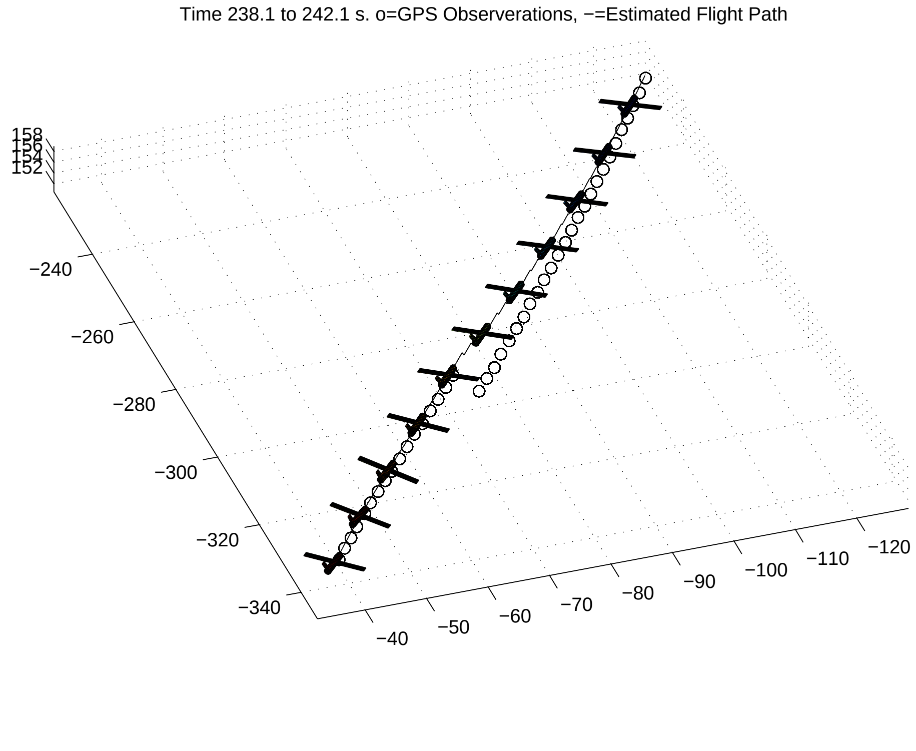
Flight playback at 328s with a bad GPS solution

Here we see how the GPS solution suddenly jumps sideways approximate 15 meter. The filter adjust fast to the new "GPS trajectoria" because the inertial accelerometers is modelled with a relative high noise. We can lower this noise, and reject observations beyond 2σ calculated from S.



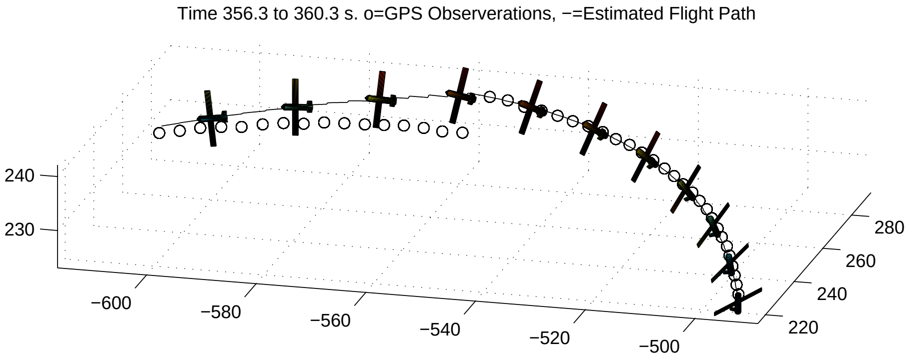
Flight playback at 238s with a bad GPS solution

The same here, suddenly the GPS solution changes very dramatically in height.

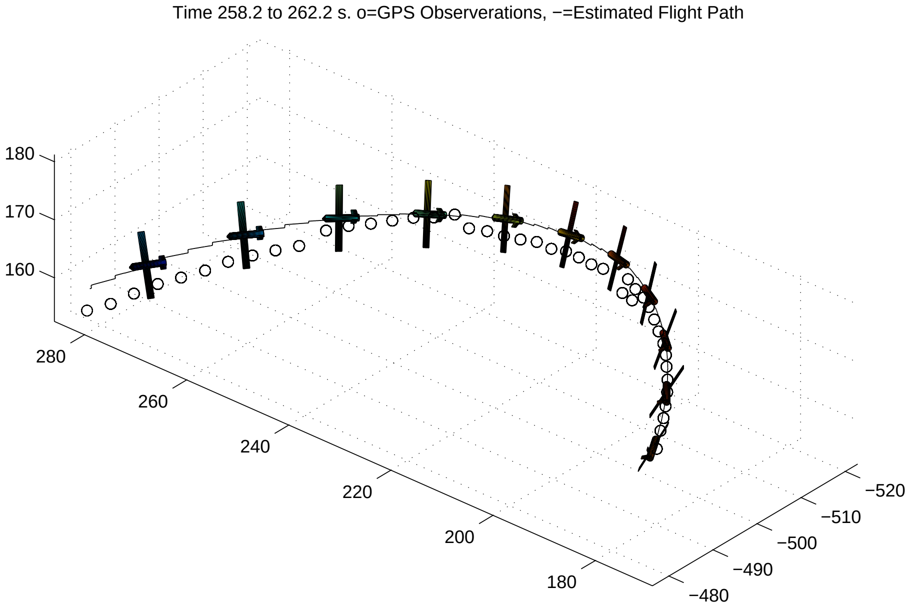


Playback at 356s

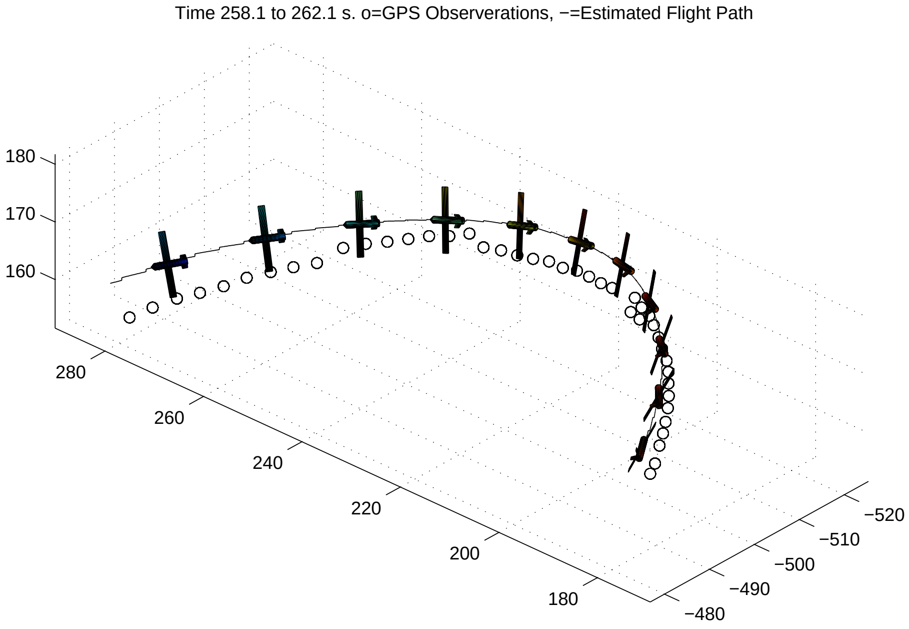
At 356s the aircraft are in a sharp left turn and the GPS solution changes in height.



Flight playback at 258s with a bad GPS solution



Flight playback at 258s with higher accuracy on the inertial sensors



Hungarian notation

If the code is to be rewritten these hungarian notations should be added.

Variabletype	Hungarian Notation	Example
matrix	m	DoubleMatrix mF(9,9)
vector	v	DoubleVector vX(9)
void		void Tmp(int i)

7.7 The conclusions

7.7.1 Bias and nonlinearity states in filter

To make the navigation filter even more accurate the navigation filter should be expanded to hold states to estimate the biases and nonlinearity terms of the inertial sensors. Optimal would be to have the bias and the driftrate for the rate gyros. This is 2*3 more states. With this the states would increse from 9/10 to 15/16 states, this would leed to an approximate increase of computations operations with $16^3/10^3$ but is nessesary for a high performance INS system. States to estimate the scalefactor errors in the accelerometers can also be added. There are no problem with this at this stage, because the biggest problem is the hugh vibrations in the airframe.

7.7.2 More efficient algoritms

The navigation filter is implemented in C++ using MPP as matrix library . The passing of information is done by passing matrixes. A lot of computer power are lost just in shuffling this matrixes into functions as arguments and then return new matrixes as return values. Instead this could be more efficient by sending pointers to the software functions.

7.7.3 Vibration damping

Heavy oscillations in the airframe is caused by the engine, this vibrations must be damped in some way. This can be done in different ways

- Vibration damping of the engine.
- Change the engine to a jetengine.
- Isolate the IMU from vibrations.

Personally I think that small jetengine should be used and that the IMU should be vibrationisolated from the airframe.

7.7.4 Tilt sensors.

There must be a way to know if the airframe is horizontal or not. The test flights starts with calibrating the IMU against the local gravity vector. It is asumed that the aircraft is placed in horizontal position, but tilt sensors should be installed in the airframe to make this for sure.

7.7.5 Placement of GPS antennas

There are four GPS antennas placed all over the airframe, one near the nose, one at the back and one on each wind. This makes a cross and are used to calculate the attitude of the aircraft. This GPS attitude system now only gives a few GPS attitude observations. This system must use INS for aiding and this will give good observations. The attitude solution calculated using the GPS must be calibrated, now it looks like there is an angle bias in the angle solution.

7.7.6 The angle classes

The attitude of the vehicle can be monitored by three different angle classes

- Eulerangle representation.
- Direction cosine representation, C_{bn} matrix.
- Quaternion representation.

7.7.7 The eulerangle model

This model works very fine, a lot of tuning was done and different process noise was tested. An interesting thing to do with this model is to expand it to hold the GPS clock model, and to implement the pseudoranging.

7.7.8 The quaternion model

This model is very pleasant to work with and was used with success. All the partial derivatives of the nonlinear system model turns out to be linear. The system matrix is free from high order function such sinus and cosine. Because of this, the linearized system matrix can be calculated very fast. However the linearized system matrix have a sizeof 10x10 and it is not sure that the simple structure of the matrix will lead to less calculation. Matrix multiplication is of order N^3 and the filters contains a lot of matrix calculations. If the state transition matrix is expanded to hold the clock model we suddenly have a 12x12 matrix.

In the external quaternionangle class that was used for keeping track of the angle there seemed to be some problem. Let's explain it, say that we have a small error in angle yaw. When we are using quaternion angle representation all four angles will be affected, the error is spread out on all quaternion elements. This is not a problem but the error is added to the real quaternion angle and so far is seems to be ok. One property of the quaternionsangle representation is that the norm should be one. If the quaternions are normalized after the error update, it seems likte the error in yaw will spread to both roll and pitch. Personally I think that the Eulerangle model is better to use.

7.7.9 Use of GPS pseudoranges

The GPS gives position solution if atleast four satellites is tracked. If more satellites are tracked the solution becomes more accurate. The problem occur however when less than four satellites are tracked. If we instead use the pseudorange to each satelite, the geometric distance, as observations to a navigation filter we can use information given with just one satellite present. This is the Tightly Coupled INS/GPS. navigation loop. And for this loop the system matrixes must be expanded with the GPS clock model, see section 5.2.7. Only two more states must be added, one is the GPS clock frequency offset and the other state is the GPS clock frequency change rate.

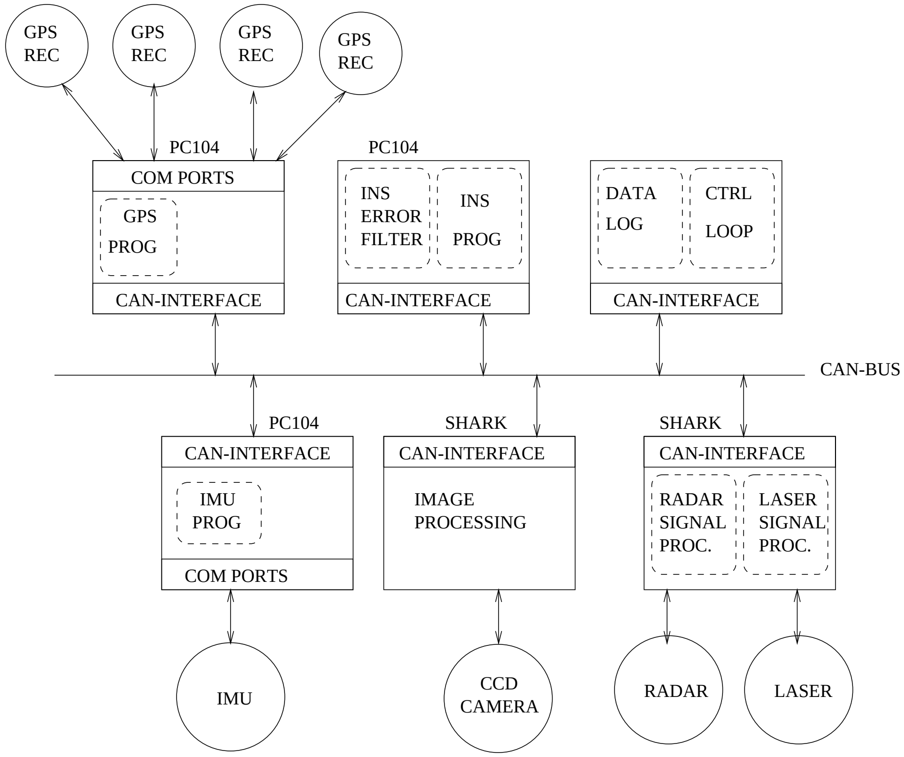
$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{x}}_{\mathbf{c}} \end{bmatrix} = \begin{bmatrix} \mathbf{F} & 0^{9x2} & \\ 0^{2x9} & 0 & 1 \\ & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{x}_{\mathbf{c}} \end{bmatrix} + \ldots \tag{7.1}$$

7.8 Future Hardware Model

The future hardware in the airframe is connected through a CAN bus. The basic computer node will be based on the shark-DSP board developed at the department. At the moment the software that communicates with the GPS receivers are written under Linux and even if the OS[3] in this project will be Embedded NT this software will probably stay in the Linux enviroment and will be ported very late in this project. There are four GPS receivers connected to a PC104 through serial ports.

[3] Operating System

Decentralized hardware



Bibliography

[1] Salah Sukkarei, Eduardo M. Nebot and Hugh F. Durrant Whyte. *A High Integrity INS/GPS Navigation Loop for Autonomous Land Vehicle Application*, Australian Centre for Field Robotics. Department of Mechanical and Mechatronic Engineering. University of Sydney

[2] Bernard Etkin and Lloyd Duff Reid, *Dynamics of Flight Stability an d Control*, Wiley, ISBN 0-471-03418-5

[3] Robert Grover Brown and Patrick Y.C Hwang, *Introduction to Random Signals and applied Kalman Filtering*, Wiley, ISBN 0-471-12839-2

[4] R.P.G. Collinson, *Introduction to Avionics*, Chapman and Hall, ISBN 0-7803-3433-7

[5] Peter S.Maybeck, *Stochastic Models, Estimation and Control Volume 1*,Navateck Books and Software.

[6] Parkinson and Spilker, *Global Position System Theory and Application I and II*, American Institute of Aeronautics and Astronautics, ISBN 1-56347-107-8,ISBN 1-56347-106-X

[7] C.K. Chui and G.Chen, *Kalman Filtering with Real-Time Application* , Springer-Verlag, ISBN 3-540-18395-7

[8] Harold W.Sorenson, *Kalman Filtering Theory And Application,* IEEE PRESS, ISBN 0-87942-191-6

[9] Curtis F. Gerald and Patrick O. Wheatley, *Applied Numerical Analysis*, Addison-Wesley, ISBN 0-201-59290-8

[10] E L Houghton and N B Carruthers, *Aerodynamics for Engineering Students*, Edward Arnold, ISBN 0-7131-433-X

[11] Alan V.Oppenheim and Roland W. Schafer, *Discrete-Time Signal Processing* , Prentice Hall, ISBN 0-13-216711-9

[12] Alan V.Oppenheim and Alan S. *Signals and System I*, Willsky, Prentice Hall, ISBN 0-13-651175-9

[13] Arthur G. O. Mutambara, *Decentralized Estimation and Control with Application to a Modular Robot*, Merton College Oxford

[14] Mohinder S. Grewal and Angus P. Andrews, *Kalman Filtering Theory and Practice*, Prentice and Hall, ISBN 0-13-211335-X

[15] Lennart Ljung, *System Identification Theory for the User*, Prentice and Hall, ISBN 0-13-881640-9

[16] Eykhoff, *System Identification* , Wiley, ISBN 0-471-24980-7

[17] Eric Nettleton, *GPS Satellite and Receiver Position Estimation*

[18] F.A. Evans and James C. Wilcox, *Experimental Strapdown Redundant Sensor Inertial Navigation System*

[19] Min H. Kao and Donald H. Eller, *Multiconfiguration Kalman Filter Design for High-Performance GPS Navigation*

[20] Report written by Peter Gibbens and other researchers. *Final Report*, University of Sydney. Department of Aeronautical Engineering.

[21] *Decentralized Filtering and Redundancy Management for Multisensor Navigation*

[22] Navstar *GPS Navstar, Global Positioning System Standard Positioning Service Signal Specification* United States Air Force, 1995 http://www.navcen.uscg.mil/gps/geninfo/gpsdocuments/sigspec/default.htm

[23] James L. Farrell, *Integrated Aircraft Navigation*,ISBN 0-12-249750-3

[24] Arnold and Maunder, *Gyrodynamics and its Engineering Applications*

[25] Krakiwsky, *Geodesy The Concept*, ISBN 0-444-87775-4

[26] Cochin, *Analysis and design of the gyroscope for inertial guidance*

[27] Meriam, *Dynamics*

[28] Beta

[29] Navigation and Course Estimation

[30] Don J. Torrieri, U.S. Army *Statistical Theory of Passice Location Systems*

[31] Alberto Isidori, *Nonlinear Control Systems*,ISBN 3-540-19916-0

[32] Stuart Bennet, *Real-Time Computer Control*, ISBN 0-13-764176-1

[33] Karl. J Astrom, *Computer Controlled System Theory and Design*, ISBN 0-13-314899-8

[34] S. Grime and H.F. Durrant-Whyte, *Data Fusion In Decentralized Sensor Networks*, University of Oxford

[35] John. J Leonard and Hugh Durrant-Whyte, *Directed Sonar Sensing for Mobile Robot Navigation*,ISBN 0-7923-9242-6

[36] Oskar Sander *Real-Time GPS Attitude Determination for Control of an Unmanned Aircraft*

[37] D.H. Titterton and J.L. Weston *Strapdown inertial navigation technology*, Peter Peregrinus Ltd

Chapter 8

Appendix

8.1 Data Specifications

8.1.1 Desk Computer Platform

The desk computer used for simulation and processing of data.

Variable	**Value**
Type	PC
Processor	Pentium III 450Mhz
Memory	160MB
Harddisk	4.2 + 8.4 GB
Graphic Card	Dimond Viper V550
Operating System	Redhat 6.0 and Windows NT 4.0
NT Program Used	Matlab, Visual C++, Borland C++ Builder
Linux Program Used	Matlab, Maple, Latex, gcc, xfig

8.1.2 Target System Computer Platform

The Target System uses a computer in PC104 format with Linux as the operating system. This System is mainly used for logging data and are connected to other parts of the system through an CAN-bus. The ground station comunicates with this platform using a serial or ethernet cable.

Variable	**Value**
Type	PC-104
Processor	Pentium 200Mhz
Memory	128MB
Harddisk	$>$ 512MB
Operating System	Slackware Linux
Extra	2 A/D Boards and one extra serial board

8.1.3 Motion Pack IMU

The rigid mouted IMU in the airframe is a IMU from Systron-Donner. The IMU have three accelerometers and three gyros mounted ortogonal.

Systron-Donner Motion Pack Data	**Rate Sensor**	**Acceleration Sensor**
Ranges	$+-200, 500^o/sec$	3,5,10g
Bias Variation over Temperature	$3^o/sec$ from 22^o	100 $g/^oC$
Long Term Stability (1 year)	$< 2.0^o/s$	$<$ 1mg
Bandwidth	$>$ 60Hz	>300Hz
Output Noise (0 to 100Hz)	$0.01^o/src$	7mV

Startup Time	$<$ 1.0 sec
Operating Temperature	-40^oC to 80^oC
Shock	200 g
Input Voltage	+-15VDC +-10%

8.1.4 Data specification of the Watson IMU

This IMU have two tilt sensors that gives elevation angle and bank angle. These sensors are used in the leveling process when the IMU is calibrated against the local gravity vector.

Watson IMU data	**Value**
Rate Range	$+-50^o/sec$
Rate Bias	2% F.S
Rate Resolution	$< 0.05^o/s$
Acceleration Range	+- 2g's
Acceleration Bias	$<$0.5% F.S
Acceleration Accuracy	$<$0.5% F.S
Acceleration Resolution	Better than 2mg
Sensor Alignment	$< 0.25^o$- all sensors
Frequency Responce	DC-20 Hz, $>$70 Hz for rates
Power Requirement	$<$ 600mA, 10-16V DC
Output Format	Digital, RS-232, Analog

8.1.5 Tetrad IMU

8.1.6 GPS Receiver

The GPS Recivers used in the airframe is from the Canadian Marconi Company, they communicates with the onboard computer through the serial ports.

Receiver Data	**Value**
Manufacturer	Canadian Marconi Company (CMC)
Model	ALLSTAR OEM (CMT-1200) REV.G
Number of Channels	Maximum 12 satellites tracked simutaneously
Max Velocity	1800 km/h
Max Acceleration	4g
Power	5.0V, 1.7W
Dimensions	67x102x19 mm
Weight	93g
Serial Communications	2xRS232 asynchronous ports. 300 to 38,400 BAUD
Serial Protocol	NMEA 183v2.0 and CMC binary

8.2 Numerical Integrators

Block Itegrator

$$I = \sum_{k=0}^{N} a_k \Delta T \tag{8.1}$$

Trapetz integrator

$$I = \sum_{k=0}^{N} \frac{a_{k-1} + a_k}{2} \Delta T \tag{8.2}$$

Simpson integrator

$$I = \sum_{k=0}^{N} \frac{a_{k-2} + 3a_{k-1} + a_k}{5} \Delta T \tag{8.3}$$

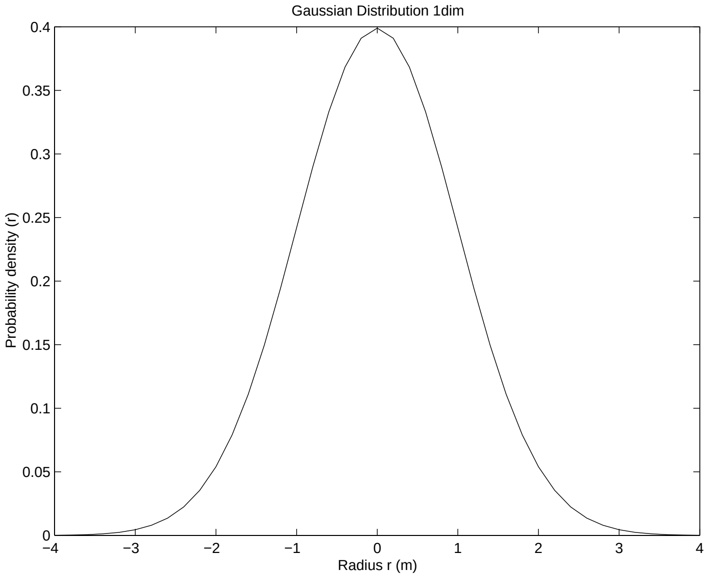
Runge-Kutta integrator

$$I = \sum_{k=0}^{N} \frac{a_{k-3} + 2a_{k-2} + 2a_{k-1} + ak}{6} \Delta T \tag{8.4}$$

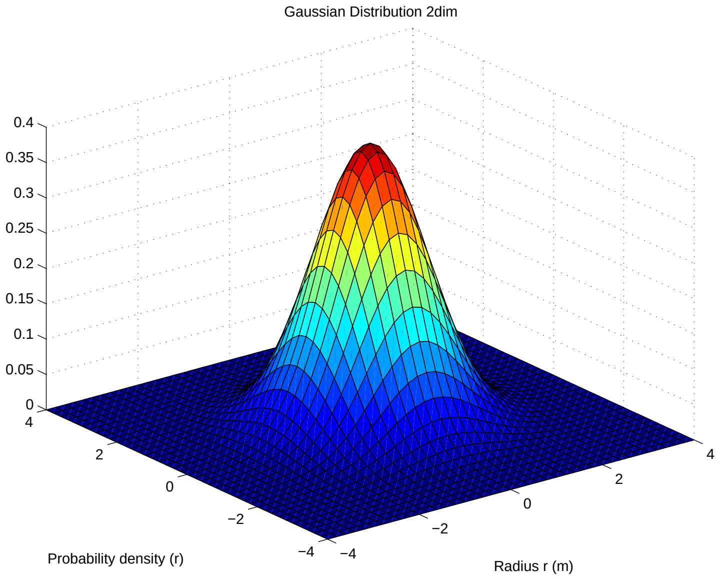
8.3 Gaussian Distrubutions

Stocastic variables in one dimension is modelled using a standard gauss distribution. When you add more dimensions it become more complicated with representations, in two dimensions the probability density function looks like a hill. In tree dimensions the probability density function can be visualized as a density cloud

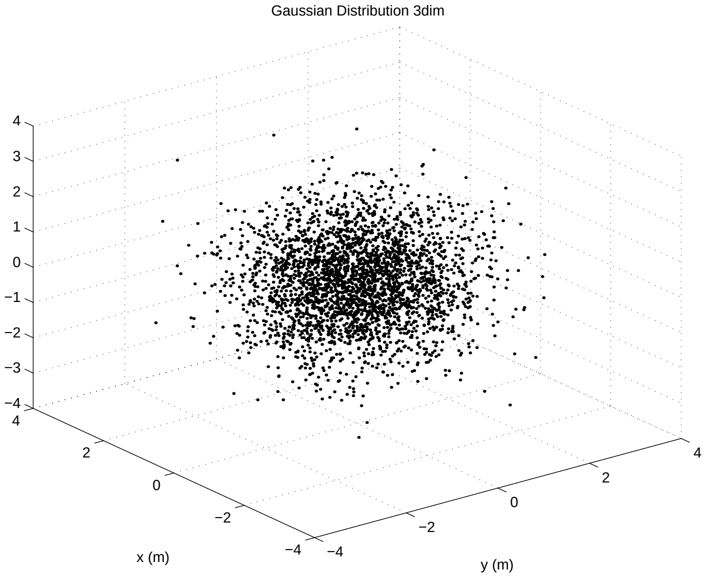
One Dimensional Probability Density Function



Two Dimensional ProbabilityDensity Function



Three Dimensional Probability Density Function



8.4 Confidence Ellipses

We have a multi dimensional stocastic variable x that is gaussian distributed. In this example we bound the dimension of x to R^3. The probability function of x is

$$P(x) = \frac{exp[-\frac{1}{2}\,(x - E[x])^T\, C^{-1}\,(x - E[x])]}{\sqrt{(2\pi)^m |C(x)|}} \tag{8.5}$$

We search the eigenvalues σ_i and eigenvectors z_i of C_α, $i = 1..3$. This can then be put into this quadratic formula.

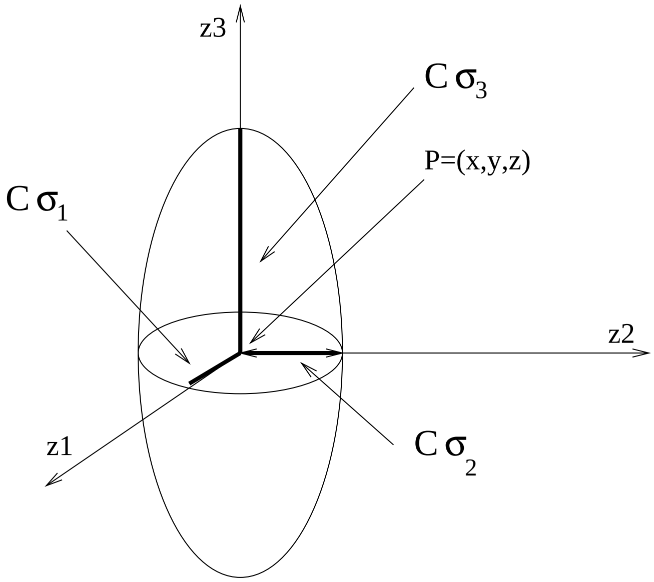
$$\begin{bmatrix} z_1 & z_2 & z_3 \end{bmatrix} \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{bmatrix}^{-1} \begin{bmatrix} z_1 & z_2 & z_3 \end{bmatrix}^T = C_\alpha^2 \tag{8.6}$$

And can be rearranged into

$$\frac{z_1^2}{C_\alpha^2 \sigma_1^2} + \frac{z_2^2}{C_\alpha^2 \sigma_2^2} + \frac{z_3^2}{C_\alpha^2 \sigma_3^2} = 1 \tag{8.7}$$

Which makes a clear picture of what we got here. It is no doubt that this can be represented as a ellipsiod in R^3. Volyme of the ellipsoid is equal to probability $1 - \alpha$ from C_α, normally $C_{0.05}$ is used. This way is just a way to view the uncertanity of a stocatic variable. It says that a stocastic variable with a probability of 0.95 lies within or on the boundary of the ellipsiod. The eigenvalues, z_i, of C spans the ellipsiod in R^3. The form of the ellipsiod is controlled by $\sigma_1, \sigma_2, \sigma_3$. If we have $\sigma_1 = \sigma_2 = \sigma_3 = \sigma$ then we will have a sphere.

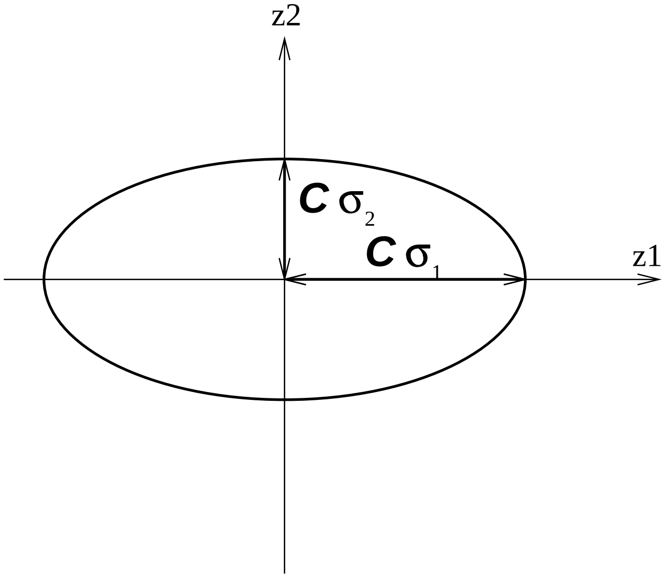
3 Dimensional Confidence Oval



8.4.1 Two Dimensional Confidence Ellipse

In two dimensions the confidence section will be an ellipse in on a two dimensional sheet. In this case we say that a stocastic variable lies within the area bounded by the ellipse with a probability of $1 - \alpha$.

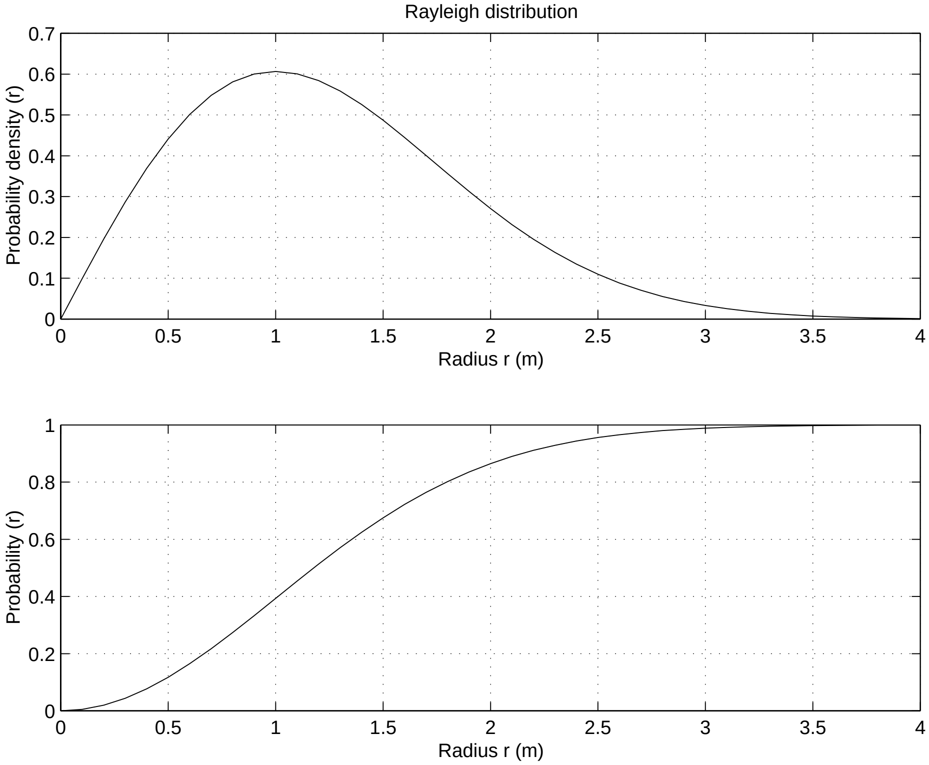
2 Dimensional Confidence Ellipse



8.5 Rayleigh Distribution

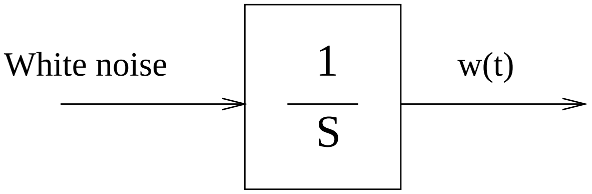
We look at the probability density function in respect to the radius from zero, this is called the Rayleigh distribution. From this we get that the highest probability to find the stocastic variable is not in the center, but instead at the distance σ from the origin.

Rayleigh probability and density functions



8.6 Wienerprocess-Randomwalk

A Wienerprocess w(t) is integrated whitenoise and can be used to modell a lot of different physical phenomeias. It was developed to modell Brownsk motion and can be used to modell the drift in gyros, accelerometers and crystal oscillators.



$\nu \in N(0,1)$, $E[w(t)] = 0$

and $E[w(t)^2] = t$

$$\dot{x}(t) = b\nu(t) = b\dot{w}(t) \tag{8.8}$$

$$x(t) = b\int \nu(t)dt \tag{8.9}$$

$$x(t_{i+1}) = x(t_i) + b\int_{t_i}^{t_{i+1}} \nu(s)ds \tag{8.10}$$

With the assumption that $\nu(t)$ is constant in between t_i and t_{i+1} the continous time modell will become.

$$x(t_{i+1}) = x(t_i) + b\sqrt{t_{i+1} - t_i}\nu(t_i) \tag{8.11}$$