



Department of Engineering Cybernetics
Norwegian University of Science and Technology (NTNU)

A real-time DVL and pressure sensor AINS comparison study between EKF, ESKF and NLO for Manta-2020

Øyvind Denvik

Master of Science in Cybernetics and Robotics

Main supervisor: Edmund Førland Brekke

Associative supervisor: Rudolf Mester

Co-supervisor: Andreas Vaage

Date: July 2020

Preface

This master thesis is the culmination of my work in the 5 year integrated MCs cybernetics and robotics study program at the Norwegian University of Science and Technology (NTNU) in Trondheim. The work was performed in the period of 1th of January to 8th of July. My main supervisor was Edmund Førland Brekke with Rudolf Mester and Andreas Vaage as the associative supervisor and co-supervisor respectively, for which I will especially thank for insights and discussion applied to my work. In addition, I will also thank Torleiv Håland Bryne for useful insights and implementation details of the nonlinear observer. Lastly, I would thank my fellow Vortex-NTNU student, Ambjørn Waldum for all the hard work in the Marine Cybernetics Lab during the experimental testing period under the COVID-19 restrictions.

The main work of this thesis, is directly linked to Manta-2020. This is an autonomous underwater vehicle that is used to compete in the Robosub competition in San Diego, USA. The need for a robust navigation is critical in order for Manta-2020 to perform well in this competition. Therefore two state estimators, working real-time, one with acceleration and gyro bias estimation and the other with only gyro bias estimation, were implemented to see how they compares to the already implemented state estimator.

Much of the written material and work done in this thesis is based on my previous work, which is given in my project thesis[1] during the period of 20th of august to 17th of december 2019. This thesis contributes to the real-time implementation of the ESKF, which were implemented previously in MATLAB, a IMU wild point filter, sensor-synchronization, sensor-buffering, and a attitude nonlinear observer which is feedback interconnected to a translational motion observer. This observer seems to have many steps like the EKF. So a background in these estimators are assumed in this thesis. Also knowledge about linear algebra and statistics are assumed to be known. However, full prior knowledge is not required. This thesis outlines concepts and theory before applying them.

All of the work done on the C++ implementations on the state estimators and the IMU wild-point filter are open source and lies on my github profile <https://github.com/oyvind1501>. Whomever who wants to contribute, are pleased to make changes to my open-source code.

As a last note, the COVID-19 restrictions, delayed much of the experimental testing work. The students working at Vortex-NTNU were not allowed to work on Manta-2020 from 12th of Mars. Two weeks before the experimental testing, a fellow hardware Vortex-NTNU student gave me, MCs student Ambjørn Waldum and three other Vortex students to work in his bed-sit room. This meant that extensive work on Manta-2020, like installing a new Sonar, tether cable, new battery module had to be implemented on Manta-2020 before the experimental testing could start.

Abstract

An error-state Kalman filter (ESKF) based on Joan Solàs version [2] and a nonlinear observer(NLO) with a feedback interconnected translational motion observer(TMO) based on the paper [3] were implemented to work real-time on Manta-2020. Both of the these state estimators were written in C++ with a corresponding robot operating system (ROS) node. These filters, together with the already implemented extended Kalman filter, were then tested with interoceptive sensor measurements coming from an inertial measurement unit(IMU), Doppler velocity log(DVL) and a pressure sensor, on a set of real test scenarios. These scenarios included both "above water" and underwater scenarios which were then compared to an land-based and underwater-based Qualisys motion capture camera system. This capture system were used to give ground truth. In addition IMU-buffering, IMU-wild point filtering and sensor-synchronization were then added to further enhance the state-estimation. Plots with the trajectories of all the state estimators and the Qualisys motion capture system, were then used to compare the filters side by side, with DVL and pressure sensor NIS tests of all filters to evaluate the filter consistency.

Summary

This thesis presents a real-time doppler velocity logger and pressure aided inertial navigation system comparison between an error-state Kalman filter with acceleration and gyro bias estimation, a nonlinear observer with gyro bias estimation and an extended Kalman filter on Manta-2020 autonomous underwater vehicle using the Robot operating system platform. The two implemented state estimators, ESKF and NLO, are two very recent filters in their respective field. A comparison between four real world testing scenarios were performed at the MC-lab at Tyholt in Trondheim. These included two "above water" tests and two underwater tests. The "above water" tests included a one round eight-shaped test and a 30 minutes square-shaped test. For the underwater tests this included an square-shaped test and a sinusoidal-shaped test. The comparison were then compared to MATLAB produced figures with a 3D east-north-altitude plot, position, velocity, attitude, bias estimates, position error, velocity error, altitude error and lastly the NIS values.

IMU-sensor buffering, IMU-wild point filtering and sensor-synchronization were added to get better filter estimates.

Sammendrag

Denne rapporten presenterer en sanntids doppler velocity logger og trykksensor hjulpet treghets navigasjons system. Den sammenligner en feil-tilstand Kalman tilstandestimator basert på Joan Solà versjon [2] med gyro og akkselerometer bias-estimering, en ulinær tilstandestimator basert på [3] med gyro bias-estimering og et forlenget Kalman filter på Manta-2020 autonome-undervannsfarkosten med bruk av robot operativ system plattformen (ROS). De to implementerte tilstandestimatorene ESKF og EKF er veldig nylige filtere i deres felt. En sammenligning av fire eksperimentelle test senarior var gjort på Tyholt i Trondheim. Dette inkluderte to "overvanns" tester og to undervanns tester. Første overvanns test var en åttetalls-aktig test, mens andre overvanns test var en 30 minutters firkant-aktig test. For undervannstestene, var første test en firkant-aktig test og andre test var en sinsus-funksjons-aktig test. Rosbag resultatene av filterene var da lagt inn i MATLAB for å produsere figurer av 3D øst,nord og høyde, posisjon, hastighet, attityde, bias estimasjonene, posisjonsfeilene, hastighetsfeilene og tilslutt DVL og trykksensor NIS verdiene.

IMU-sensor buffering, IMU-viltpunkt filtrering og sensor-synkronisering var lagt til for å få bedre tilstand-estimar på tilstandsestimatorene.

Table of Contents

Preface	i
Abstract	ii
Summary	iii
Sammendrag	iv
List of Figures	v
List of Tables	ix
Abbreviations	x
1 Introduction	1
1.1 Background	1
1.2 Vortex-NTNU	1
1.3 Robosub 2020	2
1.4 Motivation	3
1.5 Problem description	3
1.6 Contributions	4
1.7 Previous work	4
1.8 Related work	5
1.9 Thesis outline	6
2 Autonomous Underwater Vehicle Modeling	7
2.1 Kinematics and reference frames	7
2.1.1 Kinematic model	8
2.1.2 Unit Quaternions	11
2.1.3 Converting between quaternions and Euler angles	12
2.2 Kinetics	12
2.2.1 The inertia matrix \mathbf{M}	13
2.2.2 The Coriolis matrix $\mathbf{C}(\mathbf{v})$	15
2.2.3 The damping matrix $\mathbf{D}(\mathbf{v})$	16
2.2.4 The restoring forces $\mathbf{g}(\boldsymbol{\eta})$	17
3 Inertial Navigation Systems	19

3.1	The IMU	19
3.1.1	Accelerometers	19
3.1.2	Gyroscopes	20
3.1.3	Error characteristics	20
3.1.4	Allan variance	22
3.1.5	Wild point filtering	23
3.1.6	Prolonged prediction	26
3.1.7	Inclination estimate	26
3.2	Aided inertial navigation	27
3.2.1	Doppler velocity log	28
4	The AUV of interest - Manta 2020	29
4.1	Concept	29
4.2	Sensor and thruster stack	29
4.2.1	Interoceptive sensors	30
4.2.2	Exteroceptive sensors	30
4.3	The IMU - STIM300	31
4.3.1	Axis	31
4.4	Doppler Velocity Log (DVL) - DVL1000	33
4.4.1	Axes and mechanical specifications	34
4.4.2	DVL Measurement equation	34
4.4.3	Pressure sensor measurement equation	35
4.4.4	Sensor alignments	35
4.5	Exteroceptive sensors	37
4.6	Electronics system	37
4.7	ROS - Robot operating system	39
4.8	Software system overview	40
5	State estimators	42
5.1	The Bayes filter	42
5.2	The discrete Kalman filter	43
5.3	Tuning of the process noise covariance \mathbf{Q} and filter consistency	49
5.4	The extended Kalman filter	50
5.4.1	Linearization scheme	50
5.4.2	Algorithm differences compared to the Kalman filter	50
5.5	Manta-2020 Robot_localization - The open source EKF	52
5.5.1	The Kinematic prior model	52
5.5.2	Frames	52
5.5.3	Static transformations	53
5.6	Limitations of the extended Kalman filter	54
5.7	The error-state Kalman filter	55
5.7.1	The state values	55
5.7.2	The true states kinematics	55
5.7.3	The nominal state kinematics	56
5.7.4	The error state kinematics	57
5.7.5	The discrete nominal state kinematics	57
5.7.6	The discrete error state kinematics	57
5.7.7	The prediction step	59
5.7.8	The update step	59
5.7.9	ESKF injection and reset step	61
5.8	Nonlinear Observer	62
5.8.1	Assumptions and sensor configuration	62

5.8.2	Attitude estimation	62
5.8.3	Position, velocity and NED acceleration estimation	63
5.8.4	The prediction step	64
5.8.5	The update step	65
5.9	Real-time aspects	66
5.9.1	Time-synchronization	66
5.9.2	Sensor-synchronization and sensor-buffering	68
5.9.3	Execution time	69
6	Experimental testing	70
6.1	Preparation and COVID-19	70
6.2	Experimental testing at the Marine Cybernetics laboratory	70
6.3	Calibration and set-up of the Qualisys motion capture systems	71
6.3.1	"Above water" Qualisys camera setup	72
6.3.2	Fixed world frame setup	72
6.3.3	Calibration volume setup	72
6.3.4	Underwater Qualisys camera system	73
6.4	Qualisys calibration results	75
6.5	Defining the 6 DOF rigid body frame	76
6.6	Testing scenarios	77
6.6.1	Underwater testing scenarios	79
6.6.2	"Above water" testing scenarios	81
6.7	Recording and data gathering	82
6.8	Tuning of filter parameters	83
6.8.1	The EKF and NLO continuous time process noise covariance \mathbf{D}	83
6.8.2	Tuning the gyros and accelerometers	84
6.8.3	Tuning the DVL	85
6.8.4	Tuning the pressure sensor	86
6.8.5	Tuning the Gauss-Markov bias model for ESKF	86
6.8.6	Tuning of the k_1 , k_2 , k_i and M_{gyro} parameters for the NLO	86
6.8.7	The resulting tuning parameters	86
6.9	Initialization	87
7	Results and discussion	88
7.1	Experimental results	88
7.1.1	"Above water" eight number	89
7.1.2	"Above water" 30 min square	110
7.1.3	Underwater sinus	128
7.1.4	Underwater square	149
8	Conclusion	170
9	Further work	171
	Appendix A	172
	Bibliography	173

List of Figures

1.1	The ROV's built in 2016 (Mealstorm), 2017 (Terrapin), 2018 (Manta 2018), 2019 (Manta 2019)	2
1.2	The TRANSDEC Anechoic competition pool used for Robosub (a) and Cornell University in the Robosub 2014 finals (b)	3
1.3	HUGIN AINS structure(a) and HUGIN Superior AUV(b)	5
2.1	The ECI, ECEF, NED and Body frames. Figure from [4]	8
2.2	6 DOF BODY and NED coordinate frames representation	10
2.3	Visualzation of the quaternion. Courtesy([5])	11
3.1	IMU error-sources. Figure from [6]	22
3.2	Incoming sensor signal errors. Courtesy([7])	24
3.3	Wild point filtering method Courtesy([7])	24
3.4	Wild point filter demonstration	26
3.5	IMU mounting method	27
3.6	DVL work principle on the Manta-2020 AUV. Courtesy([5])	28
4.1	The AUV of interest - Manta 2020	29
4.2	Manta-2020 sensor stack	30
4.3	The STIM300 axes and mechanical dimensions. Figure from [8]	32
4.4	IMU acceleration error-characteristics table. Figures from [8]	32
4.5	IMU angular rate sensor error-characteristics table. Figure from [8]	33
4.6	DVL sensor configuration. Figure from [9]	34
4.7	The DVL1000 coordinate system and INS configuration origin. Figures from [9]	34
4.8	Manta-2020 interoceptive sensor alignment	36
4.9	Drawing of Manta-2020 main electronic motherboard. Courtesy: ([5], Rakstad)	37
4.10	Schematics of the electronic overview of Manta-2020	39
4.11	Software overview of Manta-2020. Courtesy([5], Rakstad)	40
5.1	The discrete Kalman filter algorithm	44
5.2	The initial step of the kalman filter	45
5.3	The prediction step of the Kalman filter	46
5.4	The innovation step of the Kalman filter	47
5.5	The current posterior state estimate step of the Kalman filter	48
5.6	The differences in the algorithm between the Kalman filter and the extended Kalman filter	51
5.7	EKF ROS - Robot localization coordinate frames	53
5.8	Linearization error that occurs by linearizing a nonlinear function. Figure from [10]	54

5.9	Common buffer sensor synchronization	68
6.1	The Marine Cybernetics laboratory.	71
6.2	The Qualisys motion capture system cameras	71
6.3	Qualisys "above water" world coordinate setup	72
6.4	Qualisys "above water" calibration volume setup	73
6.5	Qualisys "above water" calibration volume	73
6.6	Qualisys underwater cameras locations	74
6.7	Qualisys underwater camera system physical calibration setup	74
6.8	Qualisys underwater camera system calibration volume	75
6.9	Manta-2020 tracking balls setup	75
6.10	Qualisys "above" water calibration results	76
6.11	The Qualisys motion capture 6DOF rigid body setup	77
6.12	Manual testing procedure	78
6.13	Manta-2020 stick installment	79
6.14	Underwater planned square-testing scenario	80
6.15	Underwater performed square-testing scenario	80
6.16	Underwater "sinus-wave" testing scenario	80
6.17	"Above water" square testing scenario	81
6.18	"Above water" square 30 min lap testing scenario	81
6.19	"Above water" eight number testing scenario	82
6.20	NLO, EKF and ESKF Sensor data gathering method	83
6.21	The allan variance of the gyros	84
6.22	The allan variance of the accelerometers	85
7.1	East-north-altitude 3D trajectory comparison between the EKF and Qualisys - eight number	89
7.2	East-north-altitude 3D trajectory comparison between the ESKF and Qualisys - eight number	90
7.3	East-north-altitude 3D trajectory comparison between the NLO and Qualisys - eight number	91
7.4	EKF/ESKF/NLO east trajectory comparison with Qualisys - eight number	92
7.5	EKF/ESKF/NLO north trajectory comparison with Qualisys - eight number	93
7.6	EKF/ESKF/NLO altitude trajectory comparison with Qualisys - eight number	94
7.7	Velocity x comparison between the EKF, ESKF and NLO compared to Qualisys - eight number	95
7.8	Velocity y comparison between the EKF, ESKF and NLO compared to Qualisys - eight number	96
7.9	Velocity z comparison between the EKF, ESKF and NLO compared to Qualisys - eight number	97
7.10	Roll comparison of EKF,ESKF and NLO compared to Qualisys - eight number	98
7.11	Pitch comparison of EKF,ESKF and NLO compared to Qualisys - eight number	99
7.12	Yaw comparison of EKF,ESKF and NLO compared to Qualisys - eight number	100
7.13	ESKF bias estimates - eight number	101
7.14	NLO bias estimates - eight number	102
7.15	Position error of EKF,ESKF and NLO compared to Qualisys - eight number	103
7.16	Velocity error of EKF,ESKF and NLO compared to Qualisys - eight number	104
7.17	Velocity error of EKF,ESKF and NLO compared to Qualisys - eight number	105
7.18	NIS DVL - EKF/ESKF/NLO comparison - eight number	106
7.19	Gauss compare DVL - EKF/ESKF/NLO comparison - eight number	107
7.20	NIS Pressure - EKF/ESKF/NLO comparison - eight number	108
7.21	Gauss compare Pressure - EKF/ESKF/NLO comparison - eight number	109
7.22	EKF/ESKF/NLO east trajectory comparison with Qualisys - 30 min square	110
7.23	EKF/ESKF/NLO north trajectory comparison with Qualisys - 30 min square	111
7.24	EKF/ESKF/NLO altitude trajectory comparison with Qualisys - 30 min square	112
7.25	Velocity x comparison between the EKF, ESKF and NLO compared to Qualisys - 30 min square	113
7.26	Velocity y comparison between the EKF, ESKF and NLO compared to Qualisys - 30 min square	114
7.27	Velocity z comparison between the EKF, ESKF and NLO compared to Qualisys - 30 min square	115

7.28	Roll comparison between the EKF, ESKF and NLO compared to Qualisys - 30 min square	116
7.29	Pitch comparison between the EKF, ESKF and NLO compared to Qualisys - 30 min square	117
7.30	Yaw comparison between the EKF, ESKF and NLO compared to Qualisys - 30 min square	118
7.31	ESKF bias estimates - 30 min square	119
7.32	ESKF bias estimates - 30 min square	120
7.33	Position error of EKF,ESKF and NLO compared to Qualisys - 30 min square	121
7.34	Velocity error of EKF,ESKF and NLO compared to Qualisys - 30 min square	122
7.35	Attitude error of EKF,ESKF and NLO compared to Qualisys - 30 min square	123
7.36	NIS DVL - EKF/ESKF/NLO comparison - 30 min square	124
7.37	Gauss compare DVL - EKF/ESKF/NLO comparison - eight number	125
7.38	NIS pressure - EKF/ESKF/NLO comparison - 30 min square	126
7.39	Gauss compare pressure - EKF/ESKF/NLO comparison - square	127
7.40	East-north-altitude 3D trajectory comparison between the EKF and Qualisys - underwater sinus	128
7.41	East-north-altitude 3D trajectory comparison between the ESKF and Qualisys - underwater sinus	129
7.42	East-north-altitude 3D trajectory comparison between the NLO and Qualisys - underwater sinus	130
7.43	EKF/ESKF/NLO east trajectory comparison with Qualisys - underwater sinus	131
7.44	EKF/ESKF/NLO north trajectory comparison with Qualisys - underwater sinus	132
7.45	EKF/ESKF/NLO altitude trajectory comparison with Qualisys - underwater sinus	133
7.46	Velocity x comparison between the EKF, ESKF and NLO compared to Qualisys - underwater sinus	134
7.47	Velocity y comparison between the EKF, ESKF and NLO compared to Qualisys - underwater sinus	135
7.48	Velocity z comparison between the EKF, ESKF and NLO compared to Qualisys - underwater sinus	136
7.49	Roll comparison between the EKF, ESKF and NLO compared to Qualisys - underwater sinus	137
7.50	Pitch comparison between the EKF, ESKF and NLO compared to Qualisys - underwater sinus	138
7.51	Yaw comparison between the EKF, ESKF and NLO compared to Qualisys - underwater sinus	139
7.52	ESKF bias estimates - underwater sinus	140
7.53	NLO bias estimates - underwater sinus	141
7.54	Position error of EKF,ESKF and NLO compared to Qualisys - underwater sinus	142
7.55	Velocity error of EKF,ESKF and NLO compared to Qualisys - underwater sinus	143
7.56	Attitude error of EKF,ESKF and NLO compared to Qualisys - underwater sinus	144
7.57	NIS DVL - EKF/ESKF/NLO comparison - underwater - sinus	145
7.58	Gauss compare DVL - EKF/ESKF/NLO comparison - underwater sinus	146
7.59	NIS pressure - EKF/ESKF/NLO comparison - underwater square	147
7.60	Gauss compare pressure - EKF/ESKF/NLO comparison - underwater sinus	148
7.61	East-north-altitude 3D trajectory comparison between the EKF and Qualisys - underwater square	149
7.62	East-north-altitude 3D trajectory comparison between the ESKF and Qualisys - underwater square	150
7.63	East-north-altitude 3D trajectory comparison between the NLO and Qualisys - underwater square	151
7.64	EKF/ESKF/NLO east trajectory comparison with Qualisys - underwater square	152
7.65	EKF/ESKF/NLO north trajectory comparison with Qualisys - underwater square	153
7.66	EKF/ESKF/NLO altitude trajectory comparison with Qualisys - underwater square	154
7.67	Velocity x comparison between the EKF, ESKF and NLO compared to Qualisys - underwater square	155

7.68	Velocity y comparison between the EKF, ESKF and NLO compared to Qualisys - underwater square	156
7.69	Velocity z comparison between the EKF, ESKF and NLO compared to Qualisys - underwater square	157
7.70	Roll comparison between the EKF, ESKF and NLO compared to Qualisys - underwater square	158
7.71	Pitch comparison between the EKF, ESKF and NLO compared to Qualisys - underwater square	159
7.72	Yaw comparison between the EKF, ESKF and NLO compared to Qualisys - underwater square	160
7.73	ESKF bias estimates - underwater square	161
7.74	NLO bias estimates - underwater square	162
7.75	Position error of EKF,ESKF and NLO compared to Qualisys - underwater square	163
7.76	Velocity error of EKF,ESKF and NLO compared to Qualisys - underwater square	164
7.77	Attitude error of EKF,ESKF and NLO compared to Qualisys - underwater square	165
7.78	NIS DVL - EKF/ESKF/NLO comparison - underwater square	166
7.79	Gauss compare DVL - EKF/ESKF/NLO comparison - underwater square	167
7.80	NIS pressure - EKF/ESKF/NLO comparison - underwater square	168
7.81	Gauss compare pressure - EKF/ESKF/NLO comparison - underwater square	169

List of Tables

4.1	Parameters set in the STIM300 IMU	31
4.2	Quality of the accelerometer on STIM 300	33
4.3	Quality of the gyros on STIM 300	33
5.1	ESKF states list	55
5.2	Execution time of NLO and ESKF	69
6.1	Qualisys tracking manager setup	82

Abbreviations

AINS Aided Inertial Navigation Solution. v, 3–5

AUV Autonomous Underwater Vehicle. v, 1–7, 10, 12–14, 18, 26–29, 40, 42, 60, 70, 72, 73, 75, 77–79, 94, 97, 100, 112, 130, 141

AUVSI Association for Unmanned Vehicle Systems International. 2

BODY body-fixed. 7, 8

CAD Computer-Aided design. 4

CO Center of origin. 7

CPU Central processing unit. 66

CRC Cyclic Redundancy Check. 31

DOF Degrees of freedom. 19, 30, 77, 78

DVL Doppler Velocity Log. ii, iv, 3–5, 7, 8, 27, 28, 30, 33–35, 38, 41, 42, 52, 60, 62, 63, 65, 69, 82, 83, 86, 97, 102, 109, 115, 133, 136, 139, 170

ECEF Earth-centered Earth-fixed. 7

ECI Earth-centered inertial. 7

EKF Extended Kalman filter. iv, vi–viii, 1, 3–6, 52, 53, 62, 64, 65, 77, 81, 83, 85, 87–89, 94–100, 103–105, 109, 112–118, 121–123, 127, 128, 130, 133–139, 142–144, 148, 149, 155–160, 163–165, 170

ENU East-North-Up. 53

ESKF Error-state Kalman filter. ii–iv, vi–ix, 1, 3–6, 24, 26, 35, 52, 53, 61, 62, 69, 74, 77, 81–83, 85, 87, 88, 90, 94–100, 102–105, 109, 112–118, 120–123, 129, 130, 133–144, 148, 150, 151, 154–160, 162–165, 167, 169, 170

GNC Guidance, Navigation and Control. 1, 29, 38, 40, 70, 77

GNSS Global Navigation Satellite System. 1, 3, 7, 19, 26, 27, 102, 112

GPU Graphical Processing Unit. 38

IMU Inertial Measurement Unit. ii–iv, 3–5, 7, 8, 19, 21, 23, 24, 26, 27, 29–31, 35, 41, 42, 52, 62, 67, 82, 83, 94, 112, 133, 170

INS Inertial Navigation System. v, 6, 27, 34

IR Infrared. 71, 72

IRQ Interrupt request. 67

MATE Marine Advanced Technology Education. 4

MCM Mine CounterMeasures. 5

NED North-East-Down. 7, 8, 74, 82, 112

NLO Nonlinear Observer. ii, iii, vi–ix, 3, 4, 6, 24, 26, 52, 63, 69, 74, 77, 81–83, 85–88, 91, 94–100, 102–105, 109, 112–118, 120–123, 127, 130, 133–139, 141–144, 148, 151, 155–160, 162–165, 169, 170

NTNU Norwegian University Of Science and Technology. 1, 70

OBC On-Board Computer. 37, 38, 58, 67, 69

PCB Printed Circuit Board. 4

PVA Position velocity and attitude. 19, 66, 76, 88, 123

QTM Qualisys Track Manager. 72, 82

REA Rapid Enviromental Assessment. 5

RMSE Root mean square error. 105, 123, 144, 165

ROS Robot Operating System. ii, 27, 39, 40, 52

ROV Remotely Operated Vehicle. v, 1, 2, 4

SBC Single Board Computer. 38

SLAM Simultaneous localization and mapping. 5, 29, 31, 37, 41

TMO Translational motion observer. ii, 62, 63

UUV Unmanned Underwater Vehicle. 3

Introduction

1.1 Background

Until recently, autonomous systems have seen a drastic increase in a number of applications. The immense research and development, together with advances in computational power and numerical optimization, have led autonomous systems to be able to solve complex tasks in more complex environments. Especially this cover AUV's. Previously, they were mostly used in geoscience [11] and as research objects to study diffusion, acoustic transmission and submarine wakes. Today, they contribute in several applications, such as deep sea seafloor mapping with the help of multi-beam sonars, payload delivery, inspections and measurements of various underwater compounds.

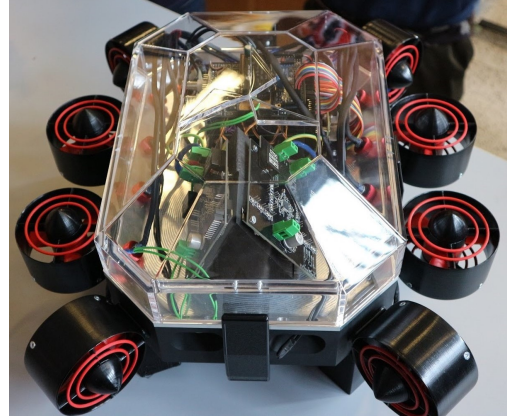
One key limitation with AUV's, is the inability to use GNSS sensors underwater. Because of this, the AUV is forced to perform dead reckoning if no other global sensor system, such as acoustics is used. Thus in order for the AUV to perform several complex underwater tasks accurately, its GNC has to be robust. Especially, this includes the navigation system, which requires high accuracy sensors with little noise and a robust state-estimation.

1.2 Vortex-NTNU

Vortex-NTNU is a student-organization that is located at NTNU at Gløshaugen in Trondheim. The organization has ranged from 18-24 students from different engineering disciplines at NTNU, hiring new students each year. The main purpose of the organization is to build underwater vehicles that is capable to compete in underwater competitions, such as Robosub and MATE international ROV competitions. Vortex-NTNU began in 2016, building a ROV named Mealstrom. This ROV is seen in figure 1.1 (a). In 2017, a new ROV was built named Terrapin, which is seen in the figure 1.1 (b). In 2018, yet a again, a new ROV was built named Manta, which is seen in figure 1.1 (c) . All of these ROV's competed in the MATE international ROV competition in USA. Going forward to 2019, the team decided to build an AUV from the old ROV model, Manta. From 2019 up until now, the same AUV model, Manta, has been used, with the main idea of further improve the software and add additional sensors. The team consist of four groups. The software-control, software-perception, hardware-mechanical and hardware-electronic. The control team is mostly responsible for the guidance and control system of Manta, while the perception team is responsible for most of the navigation system and sensor-interfacing. The hardware teams are responsible for the hardware-design, electronic communication with the sensors, embedded micro-controller and thrusters, waterproofing and computer aided graphics design.



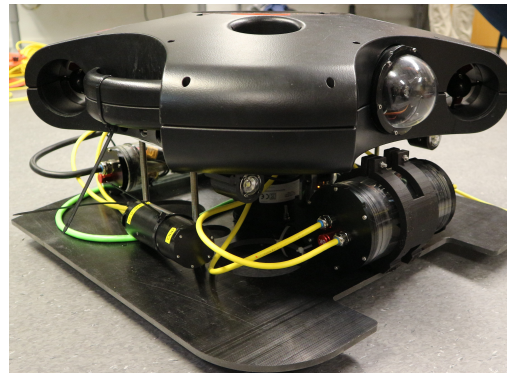
(a) Mealstorm



(b) Terrapin



(c) Manta 2018



(d) Manta 2019

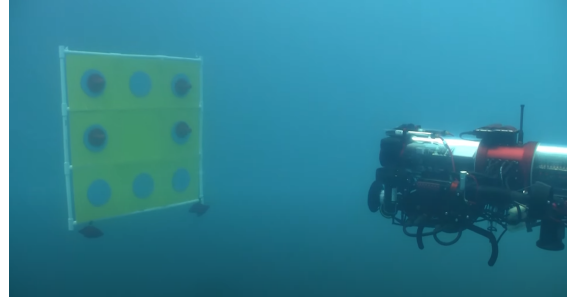
Figure 1.1: The ROV's built in 2016 (Mealstorm), 2017 (Terrapin), 2018 (Manta 2018), 2019 (Manta 2019)

1.3 Robosub 2020

Robosub is an international student-AUV competition that takes place every year in the summer at NIWC Pacific TRANSDEC, San Diego, California. The competition is operated by Robonation, inc (previously AUVSI foundation) with the first competition in 1998 [12]. The competition is done in the TRANSDEC Anchoic pool, seen in figure 1.2a. The main goal is to challenge engineering students to perform several realistic underwater missions with new themes each year. For the Robosub 2020 competition the theme is Skidoo. The tasks include driving through and under a gate with two pictures at each side, bumping into underwater "bouys", collecting "items" and place them in respective bins and fire small and safe torpedoes through different shaped holes. The reader is referred to [13] for more detail about the tasks, missions, rules and scoring.



(a) The TRANSDEC Anechoic pool in San Diego, California. Courtesy: ([14], RoboNation, Inc)



(b) Cornell University running the Robosub 2014 finals. Courtesy: ([15])

Figure 1.2: The TRANSDEC Anechoic competition pool used for Robosub (a) and Cornell University in the Robosub 2014 finals (b)

1.4 Motivation

The main motivation for this thesis is to gain insight, document and compare different real-time aided inertial navigation solutions for AUVs and UUVs in general. This is especially relevant because of the inability to use GNSS in underwater environments as described in the background section, meaning a robust navigation solution must be in place for the guidance and control to perform satisfactory, which is crucial in order for the AUV to perform well in the upcoming Robosub-2020 competition. Also one bi-motivation is to facilitate and give access to the open-source implementations of NLO and ESKF implementations in this thesis. This is to give students at Vortex-NTNU and other interested contributors the opportunity to improve upon the solutions, which will further enrich the development of underwater navigation.

1.5 Problem description

The main problem of this thesis is to develop and hopefully improve the existing navigation system on the Vortex-AUV Manta-2020, which is based upon an EKF state estimator. This will be in the form of an experimental testing comparison between the existing state estimator, EKF, NLO and ESKF state estimators based upon [3] and [2] respectively. Furthermore the problem description can be summarized with the following bullet points list:

- A literature study of IMU's, DVL's and pressure sensors and their error analysis.
- A literature study Kalman filters and nonlinear observers in general and a theoretical overview of the NLO, ESKF and EKF used in the experimental testing comparisons.
- A literature study of real-time aspects for AINS
- Design and implementation of real-time aspects of the NLO and ESKF. This includes execution time, sensor-synchronization and sensor-buffering.
- Design and implementation of an IMU wild-point filter.
- Design and implementation of a NLO and ESKF for real-time sensor fusion of IMU, DVL and pressure sensor.
- Simulation testing comparison of the EKF, ESKF and NLO.

- Experimental testing comparison of the EKF, ESKF and NLO with ground truth and consistency analysis.
- Design and implementation of a user-friendly parameter - interface on the ESKF and NLO.
- Design and implementation an user-friendly general sensor-interface, such that other sensors may be added without implementing code.

1.6 Contributions

With the list above, the following contributions were done throughout the thesis.

- A literature study of IMU's, and DVL's and their error analysis. item A literature study Kalman filters and nonlinear observers in general and a theoretical overview of the NLO, ESKF and EKF used in the experimental testing comparisons.
- A literature study of real-time aspects for AINS
- Designed and implemented an ESKF and NLO in C++.
- Designed and implemented an IMU wild point filter.
- Designed and implemented real-time aspects of the NLO and ESKF, which included execution time, sensor-synchronization and sensor-buffering.
- Validated that the NLO and ESKF estimated real-time on Manta-2020.
- Validated and compared the EKF, ESKF and NLO trough experimental testing with both underwater and "above water" camera-based system giving ground truth.
- NIS consistency analysis of EKF, ESKF and EKF.
- Designed and implemented a user-friendly parameter-interface on the ESKF and NLO.

1.7 Previous work

Manta has been developed since autumn 2017. The main focus then was the hardware-specific design and construction. This included designing and installing relevant mechanical and electronic components and accessories. Most of the design was done using CAD based methods and PCB software tools such as Altium. There was also some focus on the software system where the Vortex-2017/2018 team implemented software such that it was fully capable of manual operation by the use of a remote control. At the end of may 2018, the team had made a fully working ROV which competed in the MATE international competition.

During autumn 2018 the Vortex team decided to develop the ROV to an AUV. The focus then shifted to do research of finding relevant exteroceptive and interoceptive sensors to use for autonomous operation. After finding most of the relevant sensors described in chapter 3, except the sonar, custom software drivers for these sensors were then implemented. The author contributed together with his co-supervisor Andreas Vaage, the implementation of the IMU and most of the DVL custom drivers. In the meantime, the control-software team began developing a simulation model of the AUV with a corresponding dynamical model and a 1:1 scale design of the Marine Cybernetics laboratory at Tyholt, Trondheim and the Robosub TRANSDEC Anechoic pool in San Diego, California, depicted in figure 1.2a(a), in the simulation software tool Gazebo [5]. During the second semester the software team had developed a non-working, but complete motion control, guidance, mission control system and a navigation solution based upon a 3 DOF PD controller, line of sight guidance, state machine and an extended Kalman filter respectively.

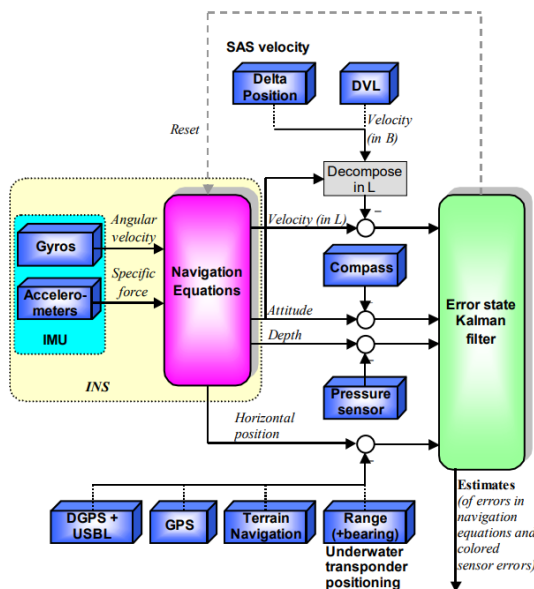
Reaching august 2019 the main objective was to further develop, do experimental testing, enhance the software architecture and find alternative or better solutions on the AUV with the goal of reaching the finals

at the Robosub competition. Also this year a new exteroceptive sensor was added. The sonar, which added the feature of long-distance 2D object detection. The development was then mainly on adding robust path planning, object detection, camera and sonar- sensor fusion SLAM, AINS and neural network based object detection for the front camera.

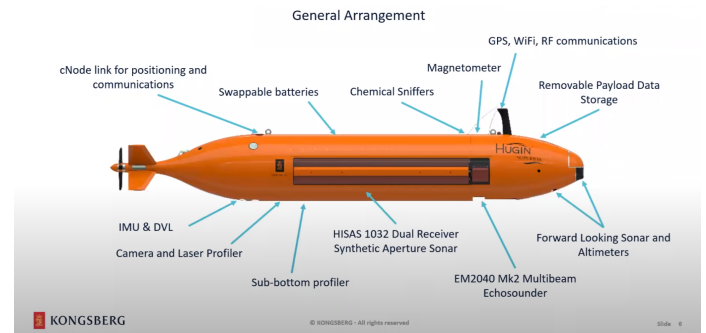
The author of this project contributed in the development of the EKF and an offline implementation of the ESKF in his project thesis[1]. This thesis also had real-world test scenarios based on the Qualisys camera system. Therefore much of the theory, static transforms between the sensor frames, tuning of the ESKF and EKF have already been done in this thesis. Therefore much of the written material in the authors project thesis will also be in this thesis, but some theory have been modified and further adjusted.

1.8 Related work

There are many related research and commercial AUV's. One of them is the HUGIN family from Kongsberg Maritime and Norwegian Defence Research Establishment. These AUV's are used both for military and commercial applications such as deep-sea seabed mapping, offshore surveying for the oil and gas industry, Naval MCM and REA operations and hydrography [16]. Depicted in figure 1.3b is one their newest AUV's combined with updated on-board data processing and an overall improved sensor stack. Depicted in figure 1.3a, is the on-board integrated inertial navigation system used for the HUGIN 1000 and 3000 AUV's. It is based on an DVL, pressure sensor and compass AINS with different forms of position measurement updates [17] which are seen in the lower part in 1.3a. Here DGPS + USBL, GPS surface fix, bathymetric terrain navigation and Underwater transponder positioning are the methods used for position updates[17]. The state estimator is an ESKF with a rotation-matrix based attitude estimation. Compared to the various integrated navigation solutions contributed from this thesis, all use a quaternion based attitude estimation with DVL, IMU and pressure sensor without any form of position measurement updates.



(a) HUGIN integrated inertial navigation system structure. Courtesy: ([18])



(b) The HUGIN Superior AUV, general sensor stack and arrangement. Courtesy: ([19])

Figure 1.3: HUGIN AINS structure(a) and HUGIN Superior AUV(b)

1.9 Thesis outline

This thesis organizes its content in chapters and sections. The second chapters outlines the autonomous underwater vehicle modeling. This chapter focus mostly on the mathematical notation, reference frames, kinematics and kinetics for describing the marine AUV in motion.

Chapter 3 details theory of how INS systems generally work and their error analysis. In addition it presents methods of finding the measurement noises using the Allan variance method. Also a wild point filtering method will be introduced.

Chapter 4 outlines the sensor stack, electronic system and the details regarding the interoceptive sensors on Manta-2020.

Chapter 5 goes trough the theory of the EKF,ESKF and the NLO. Together with this is it also shows realtime aspects, such as time-synchronization, sensor-synchronization and the execution time.

Chapter 6 outlines the experimental testing scenarios, and how the preparation of the tests were done with Qualisys motion capture system.

Chapter 7 discuss the results of the experimental testing scenarios.

Chapter 8 outlines conclusions based on the the results.

Chapter 9 discuss further work, and what can be contributed in order to make the state estimators perform better.

Autonomous Underwater Vehicle Modeling

Most of following mathematical notation, reference frames and kinematics are based on theory found in [4].

2.1 Kinematics and reference frames

In order to analyze the kinetics and kinematics of the AUV, different reference frames are then needed. Especially is this important to take into account, because the measurements from different sensors are given in a specific reference frame. It is therefore critical to transform these measurements to a common reference frame, before using them in for example a state observer. For this project, for instance, the DVL, IMU and pressure sensor will have reference frames to their corresponding sensor frame, which is dependent on their alignment.

There are different types of reference frames that can be used, such as ECI, ECEF, NED, BODY and the respective reference frames for each sensor alignment, which will in general in this thesis be denoted as `SENSOR_FRAME`. For convenience and easier notation, these frames will be denoted as $\{i\}$, $\{e\}$, $\{n\}$ and the `SENSOR_FRAME` to be the abbreviation name of the respective sensor.

The ECEF frame is an inertial frame that is mostly used for terrestrial navigation. It is a non-accelerating frame, where the newtons second laws of motion can be applied. This reference frame is where for instance the IMU has its measurements with respect to. Here the origin lies in the center of earth. The second reference frame is the ECI. This frame has the same properties as the ECEF frame, but with the difference that its axis rotate relative to the ECEF frame with an angular rotation of $w = 7.2921 * 10^{-5}$ [4]. Mostly this angular rotation is often neglected for vessels or marine crafts that are moving at low speed[4]. This frame is mostly used for marine crafts that move long distances between continents. The NED frame is a local reference frame that has its origin in defined relative to the earth reference ellipsoid (World Geodetic System, 1984) [4]. This frame is where for instance the measurements coming from a GNSS sensors are given in. Last but not least the BODY frame is the frame that is fixed to the marine vessel. It has its origin as defined by the user, denoted CO, which is often designed to be midships in the waterline[4].

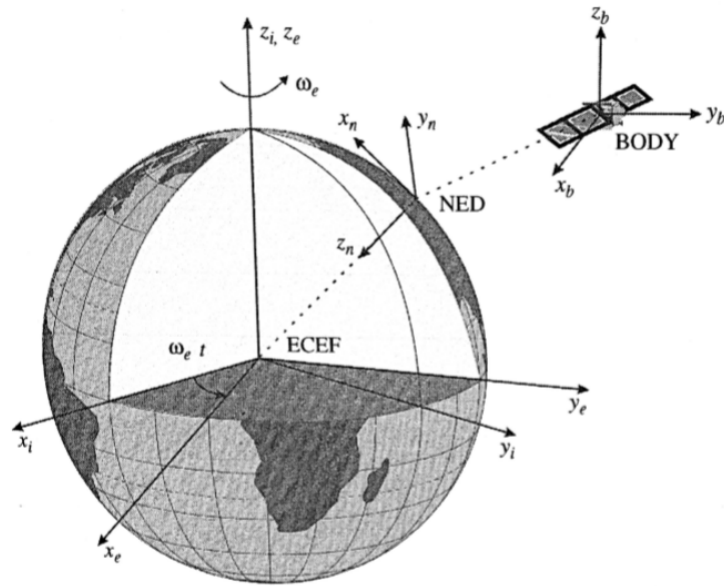


Figure 2.1: The ECI, ECEF, NED and Body frames. Figure from [4]

For Manta-2020, the use of long distance terrestrial navigation is unnecessary. Therefore the most convenient choice of reference frames, will then be NED, BODY and the SENSOR_FRAMES. In coordinate from, these will be denoted as follows

$$\{n\} = [x_w, y_w, z_w]^T \quad (2.1)$$

$$\{b\} = [x_b, y_b, z_b]^T \quad (2.2)$$

and the reference frames for the IMU, DVL and pressure sensor as follows.

$$\{imu\} = [x_{imu}, y_{imu}, z_{imu}]^T \quad (2.3)$$

$$\{dvl\} = [x_{dvl}, y_{dvl}, z_{dvl}]^T \quad (2.4)$$

$$\{pressure\} = [x_{pressure}, y_{pressure}, z_{pressure}]^T \quad (2.5)$$

2.1.1 Kinematic model

In order to define the kinematic model of any marine vessel, following a convenient and often used notation of position, velocities, forces and moments, will be easier for the reader to comprehend and understand the theory presented. Here the SNAME (1950) notation will be used shown in table 1.1.

DOF	Forces & Moments	Velocities	Positions & Euler Angles
Surge	X	u	x
Sway	Y	v	y
Heave	Z	w	z
Roll	K	p	ϕ
Pitch	M	q	θ
Yaw	N	r	ψ

Table 1.1: SNAME-notation for marine vessels. Table from [5], Courtesy(SNAME (1950))

Based on the SNAME notation any marine vessel that uses $\{n\}$ frame as its "world" coordinate reference frame and the coordinate origin of $\{b\}$, can be described as in table 1.2.

Description	Linear and Angular	Linear	Angular
Position	$\eta = [\mathbf{p}_{b/n}^n, \mathbf{v}_{b/w}^b]^T$	$\mathbf{p}_{b/n}^b = [x_{b/n}^n, y_{b/n}^n, z_{b/n}^n]^T$	$\Theta_{nb} = [\phi_{nb}, \theta_{nb}, \psi_{nb}]^T$
Velocity	$\nu = [\mathbf{v}_{b/n}^b, \omega_{b/n}^b]^T$	$\mathbf{v}_{b/n}^b = [u_{b/n}^n, v_{b/n}^n, w_{b/n}^n]^T$	$\omega_{b/n}^b = [p_{b/n}^n, q_{b/n}^n, r_{b/n}^n]^T$
Force/moment	$\tau = [\mathbf{f}_b^b, \mathbf{m}_b^b]$	$\mathbf{f}_b^b = [X, Y, Z]$	$\mathbf{m}_b^b = [K, M, N]^T$

Table 1.2: 6 DOF kinematics on marine vessels using SNAME notation

These may be described as follows

$\mathbf{p}_{b/n}^n$ = Position of the origin of $\{b\}$ (\mathbf{o}_b) with respect to $\{n\}$ expressed in $\{n\}$

$\mathbf{v}_{b/n}^v$ = Linear velocity of the origin of $\{b\}$ (\mathbf{o}_b) with respect to $\{n\}$ expressed in $\{b\}$

Θ = Euler angles between $\{n\}$ and $\{b\}$

$\omega_{b/n}^b$ = Angular velocity of $\{b\}$ with respect to $\{n\}$ expressed in $\{b\}$

\mathbf{f}_b^b = Force with line of action trough the origin of $\{b\}$ (\mathbf{o}_b) expressed in $\{n\}$

\mathbf{m}_b^b = Moment about the origin of $\{b\}$ (\mathbf{o}_b) expressed in $\{n\}$

Figure 2.2 shows how table 1.2 can be represented visually on Manta-2020 or any other kind of marine vessel.

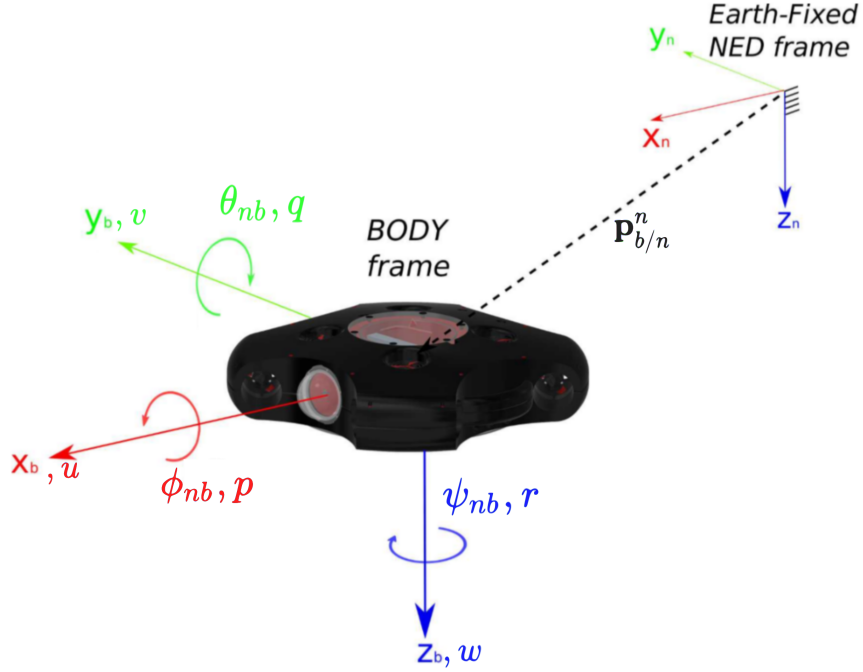


Figure 2.2: 6 DOF BODY and NED coordinate frames representation

With table 1.2 in mind the 6 DOF kinematic equation of a marine vessel can be expressed as follows [4].

$$\dot{\eta} = \mathbf{J}_{\Theta}(\eta)\nu \iff \begin{bmatrix} \dot{\mathbf{p}}_{b/n}^n \\ \dot{\Theta}_{nb} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_b^n(\Theta_{nb}) & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{T}_{\Theta}(\Theta_{nb}) \end{bmatrix} \begin{bmatrix} \mathbf{v}_{b/n}^b \\ \boldsymbol{\omega}_{b/n}^b \end{bmatrix} \quad (2.6)$$

where $\mathbf{R}_b^n(\Theta_{nb}) : S^3 \rightarrow SO(3)$ is the Euler angle rotation matrix using the zyx convention from {b} to {n} and $\mathbf{T}_{\Theta}(\Theta_{nb})$ is the Euler angle transformation matrix. These are defined as follows

$$\mathbf{R}_b^n(\Theta_{nb}) = \mathbf{R}_{\psi} \mathbf{R}_{\theta} \mathbf{R}_{\phi} \iff \mathbf{R}_b^w(\Theta_{nb}) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\psi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\psi & c\theta c\psi \end{bmatrix} \quad (2.7)$$

$$\mathbf{T}_{\Theta}(\Theta_{nb}) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix} \quad (2.8)$$

Where s, c, t denote \sin, \cos, \tan respectively.

For underwater vessels like Manta-2020, a full 6 DOF navigation solution is required for the AUV to be fully functional for underwater maneuvers and actions. One problem with the use of Euler angles, is that the $\mathbf{T}_{\Theta}(\Theta_{nb})$ is undefined when the pitch angle θ is $\pm 90^\circ$, which is known as the Gimbal lock. If Manta-2020 where to do a maneuver such that the pitch angle where to get close to θ is $\pm 90^\circ$, it would potentially break a navigation solution and give garbage data. Thus a another attitude representation has to be used. A good candidate here is quaternions. They have the advantages of not having singularities/discontinuities

and is mathematically simple. In this thesis, all of the implemented state estimators use quaternions as their attitude representation. Other alternatives like rotation matrices could also have been used, like the navigation solution of the HUGIN family, but they usually are over-parameterized, requires more storage and is more susceptible to round-off errors [20].

2.1.2 Unit Quaternions

A unit quaternion is a four dimensional complex number representation with one real η and three imaginary parts $\epsilon \in [4]$, $\mathbf{q} = [\eta, \epsilon_1, \epsilon_2, \epsilon_3]^T$.

There exist several different conventions of the unit quaternions. For instance the JPL and Hamilton conventions. In this thesis, the Hamilton convention is used in the design of the state estimators. A visualization of this quaternion representation is shown in figure 2.3

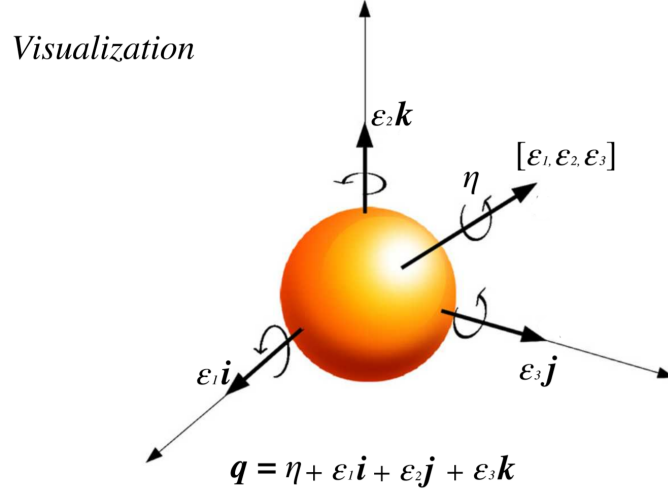


Figure 2.3: Visualization of the quaternion. Courtesy([5])

With this the 6 DOF kinematics equations can now be represented with quaternions as follows

$$\dot{\eta} = \mathbf{J}_{\mathbf{q}}(\eta)\nu \iff \begin{bmatrix} \dot{\mathbf{p}}_{b/n}^n \\ \dot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_b^n(\mathbf{q}) & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \mathbf{T}_{\mathbf{q}}(\mathbf{q}) \end{bmatrix} \begin{bmatrix} \mathbf{v}_{b/n}^b \\ \omega_{b/n}^b \end{bmatrix} \quad (2.9)$$

where the $\mathbf{R}_b^n(\mathbf{q}_{wb})$ and $\mathbf{T}_{\mathbf{q}}(\mathbf{q})$ are defined as:

$$\mathbf{R}_b^n(\mathbf{q}) = \begin{bmatrix} 1 - 2(\epsilon_2^2 + \epsilon_3^2) & 2(\epsilon_1\epsilon_2 - \epsilon_3\eta) & 2(\epsilon_1\epsilon_3 + \epsilon_2\eta) \\ 2(\epsilon_1\epsilon_2 - \epsilon_3\eta) & 1 - 2(\epsilon_1^2 + \epsilon_3^2) & 2(\epsilon_2\epsilon_3 + \epsilon_1\eta) \\ 2(\epsilon_1\epsilon_3 - \epsilon_2\eta) & 2(\epsilon_2\epsilon_3 + \epsilon_1\eta) & 1 - 2(\epsilon_1^2 + \epsilon_2^2) \end{bmatrix} \quad (2.10)$$

$$\mathbf{T}_{\mathbf{q}}(\mathbf{q}) = \frac{1}{2} \begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ \eta & -\epsilon_3 & \epsilon_2 \\ \epsilon_3 & \eta & -\epsilon_1 \\ -\epsilon_2 & \epsilon_1 & \eta \end{bmatrix} \quad (2.11)$$

There are also several mathematical properties of these quaternions that are used extensively in the design and implementation of the state estimators. These properties are the sum of two quaternions, tensor product, norm, conjugate and normalization, which are represented below respectively.

$$\mathbf{q}_a + \mathbf{q}_b = \begin{bmatrix} \eta_a + \eta_b \\ \epsilon_a + \epsilon_b \end{bmatrix} \quad (2.12)$$

$$\mathbf{q}_a \otimes \mathbf{q}_b = \begin{bmatrix} \eta_a \eta_b - \epsilon^T \epsilon_b \\ \eta_b \epsilon_a + \eta_a \epsilon_b + \epsilon_a x \epsilon_b \end{bmatrix} \quad (2.13)$$

$$\|\mathbf{q}\| = \sqrt{\eta^2 + \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2} \quad (2.14)$$

$$\mathbf{q}^* = \begin{bmatrix} \eta \\ -\epsilon \end{bmatrix} \quad (2.15)$$

Especially is the normalization procedure 1.14 important to satisfy the constraint:

$$\mathbf{q}^T \mathbf{q} = \epsilon^2 + \eta_1^2 + \eta_2^2 + \eta_3^2 \quad (2.16)$$

when integrating $\dot{\mathbf{q}} = \mathbf{T}(\mathbf{q})\omega_{b/n}^b$.

2.1.3 Converting between quaternions and Euler angles

In order to get a visual representation of the AUV, Euler angles is the best suited attitude representation. Thus conversion from and to quaterions and Euler angles are needed.

The conversion from quaternions to Euler angles is represented as follows [21]:

$$\phi = atan2(2(\epsilon_2 \epsilon_3 + \eta \epsilon_1), \eta^2 - \epsilon_1^2 - \epsilon_2^2 + \epsilon_3^2) \quad (2.17)$$

$$\theta = asin(2(\eta \epsilon_2 - \epsilon_1 \epsilon_3)) \quad (2.18)$$

$$\psi = atan2(2(\epsilon_1 \epsilon_2 + \eta \epsilon_3), \eta^2 + \epsilon_1^2 - \epsilon_2^2 - \epsilon_3^2) \quad (2.19)$$

and from Euler angles to quaternions as follows:

$$\mathbf{q} = \begin{bmatrix} C_{\frac{\phi}{2}} C_{\frac{\theta}{2}} C_{\frac{\psi}{2}} + s_{\frac{\phi}{2}} s_{\frac{\theta}{2}} s_{\frac{\psi}{2}} \\ s_{\frac{\phi}{2}} C_{\frac{\theta}{2}} C_{\frac{\psi}{2}} - C_{\frac{\phi}{2}} s_{\frac{\theta}{2}} s_{\frac{\psi}{2}} \\ C_{\frac{\phi}{2}} s_{\frac{\theta}{2}} C_{\frac{\psi}{2}} + s_{\frac{\phi}{2}} C_{\frac{\theta}{2}} s_{\frac{\psi}{2}} \\ C_{\frac{\phi}{2}} C_{\frac{\theta}{2}} s_{\frac{\psi}{2}} - s_{\frac{\phi}{2}} s_{\frac{\theta}{2}} C_{\frac{\psi}{2}} \end{bmatrix} \quad (2.20)$$

2.2 Kinetics

In order to simulate Manta-2020, its 6 DOF rigid body kinetics must be described. All of the numbered values and methods described in this section are found by the Vortex-NTNU control team and is thoroughly described in Kristoffer Rakstad Solberg master thesis [5].

The general 6 DOF rigid-body kinetics for a vessel is described in [4] as follows:

$$\mathbf{M}\dot{\mathbf{v}} + \mathbf{C}(\mathbf{v})\mathbf{v} + \mathbf{D}(\mathbf{v})\mathbf{v} + \mathbf{g}(\eta) + \mathbf{g}_0 = \tau_{propolusion} + \tau_{wind} + \tau_{wave} \quad (2.21)$$

Which is derived from the Newton-euler equation of a rigid body. Here the system inertia matrix $\mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_a$ consist the rigid-body mass matrix and the added mass matrix. The $\mathbf{C}(\mathbf{v}) = \mathbf{C}(\mathbf{v})_{RB} + \mathbf{C}(\mathbf{v})_A$ matrices, represent the Coriolis matrices for the rigid-body and added mass, respectively. The matrix $\mathbf{D}(\mathbf{v}) = \mathbf{D}_P + \mathbf{D}_V + \mathbf{D}_n(\mathbf{v}_r)$ represent the damping matrix. For manta-2020 the static forces due to ballast systems \mathbf{g}_0 will be neglected. The propolusion forces consist of the thruster forces from the AUV. By these assumptions the 6 DOF rigid body kinetics breaks down to the following for an AUV:

$$\mathbf{M}\dot{\mathbf{v}} + \mathbf{C}(\mathbf{v})\mathbf{v} + \mathbf{D}(\mathbf{v})\mathbf{v} + \mathbf{g}(\eta) = \tau_{propolusion} + \tau_{wind} + \tau_{wave} \quad (2.22)$$

Where \mathbf{M} , $\mathbf{C}(\mathbf{v})$, $\mathbf{D}(\mathbf{v})$ and $\mathbf{g}(\eta)$ are described as:

$$\begin{aligned} \mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_A &= \begin{bmatrix} m\mathbf{I}_{3 \times 3} & -m\mathbf{S}(\mathbf{r}_g^b) \\ m\mathbf{S}(\mathbf{r}_g^b) & \mathbf{I}_b \end{bmatrix} - \text{diag}\left(\frac{\partial \mathbf{X}}{\partial \dot{u}}, \frac{\partial \mathbf{Y}}{\partial \dot{v}}, \frac{\partial \mathbf{Z}}{\partial \dot{w}}, \frac{\partial \mathbf{K}}{\partial \dot{p}}, \frac{\partial \mathbf{M}}{\partial \dot{q}}, \frac{\partial \mathbf{N}}{\partial \dot{r}}\right) \\ &= \begin{bmatrix} m - \frac{\partial \mathbf{X}}{\partial \dot{u}} & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m - \frac{\partial \mathbf{Y}}{\partial \dot{v}} & 0 & -mz_g & 0 & mx_g \\ 0 & 0 & m - \frac{\partial \mathbf{Z}}{\partial \dot{w}} & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_x - \frac{\partial \mathbf{K}}{\partial \dot{p}} & -I_{xy} & -I_{xz} \\ mz_g & 0 & -mx_g & -I_{yx} & I_y - \frac{\partial \mathbf{M}}{\partial \dot{q}} & -I_{yz} \\ -my_g & mx_g & 0 & -I_{zx} & -I_{zy} & I_z - \frac{\partial \mathbf{N}}{\partial \dot{r}} \end{bmatrix} \end{aligned} \quad (2.23)$$

$$\begin{aligned} \mathbf{C}(\mathbf{v}) &= \mathbf{C}_{RB}(\nu) + \mathbf{C}_A(\nu_r) \\ &= \begin{bmatrix} \mathbf{0}_{3 \times 3} & -m\mathbf{S}(\mathbf{v}_{b/w}^b) - m\mathbf{S}(\omega_{b/w}^b)\mathbf{S}(\mathbf{r}_g^b) \\ -m\mathbf{S}(\mathbf{v}_{b/w}^b) - m\mathbf{S}(\mathbf{r}_g^b)\mathbf{S}(\omega_{b/w}^b) & -\mathbf{S}(\mathbf{I}_b\omega_{b/w}^b) \end{bmatrix} \\ &+ \begin{bmatrix} \mathbf{0}_{3 \times 3} & -\mathbf{S}(\mathbf{A}_{11}\nu_1 + \mathbf{A}_{12}\nu_2) \\ -\mathbf{S}(\mathbf{A}_{11}\nu_1 + \mathbf{A}_{12}\nu_2) & -\mathbf{S}(\mathbf{A}_{21}\nu_1 + \mathbf{A}_{22}\nu_2) \end{bmatrix} \end{aligned} \quad (2.24)$$

$$\mathbf{D}(\mathbf{v}) = \mathbf{D}_L + \mathbf{D}_{NL}(\mathbf{v}_r)$$

$$= - \begin{bmatrix} X_{|u|u}|u_r| + \frac{\partial \mathbf{X}}{\partial u} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{|v|v}|v_r| + \frac{\partial \mathbf{Y}}{\partial v} & 0 & \frac{\partial \mathbf{Y}}{\partial p} & 0 & Y_{|v|r}|v_r| + \frac{\partial \mathbf{Y}}{\partial r} \\ 0 & 0 & Z_{|w|w}|w_r| + \frac{\partial \mathbf{Z}}{\partial w} & 0 & \frac{\partial \mathbf{Z}}{\partial q} & 0 \\ 0 & K_v + \frac{\partial \mathbf{K}}{\partial v} & 0 & K_{|p|p}|p| + \frac{\partial \mathbf{K}}{\partial p} & 0 & K_r + \frac{\partial \mathbf{K}}{\partial r} \\ 0 & 0 & M_w + \frac{\partial \mathbf{M}}{\partial q} & 0 & M_{|q|q}|q| + \frac{\partial \mathbf{M}}{\partial q} & 0 \\ 0 & N_{|v|v}|v_r| + \frac{\partial \mathbf{N}}{\partial v} & 0 & \frac{\partial \mathbf{N}}{\partial p} & 0 & N_{|v|r}|v_r| + \frac{\partial \mathbf{N}}{\partial r} \end{bmatrix}$$

$$\mathbf{g}(\eta) = - \begin{bmatrix} \mathbf{R}_b^n(q)^{-1}(\mathbf{f}_g^n + \mathbf{f}_b^n) \\ \mathbf{r}_g^b \times \mathbf{R}_b^n(q)^{-1}\mathbf{f}_g^n + \mathbf{r}_b^b \times \mathbf{R}_b^n(q)^{-1}\mathbf{f}_b^n \end{bmatrix} \quad (2.25)$$

The matrices \mathbf{M} , $\mathbf{C}(\mathbf{v})$, $\mathbf{v}(\mathbf{v})$ and $\mathbf{g}(\eta)$ are stated here for reference. Further analysis and justification is stated below.

2.2.1 The inertia matrix M

The inertia matrix \mathbf{M} is, as previously written, is composed of two parts \mathbf{M}_{RB} and \mathbf{M}_A , which is the rigid-body mass matrix and added mass, respectively.

The \mathbf{M}_{RB} matrix is defined according to [4] as follows

$$\mathbf{M}_{RB} = \begin{bmatrix} m\mathbf{I}_{3 \times 3} & -m\mathbf{S}(\mathbf{r}_g^b) \\ m\mathbf{S}(\mathbf{r}_g^b) & \mathbf{I}_b \end{bmatrix} \quad (2.26)$$

Here $\mathbf{I}_{3 \times 3}$ denotes the 3×3 identity matrix, m is the total mass of AUV in kg . $\mathbf{S}(\mathbf{r}_g^b)$ is defined as the skew symmetric matrix. \mathbf{r}_g^b is defined as the location of center of gravity (CG) with respect to center of origin (CO). \mathbf{I}_b is the inertia matrix about the {b} frame's origin. This can be found in equation (3.34) in [4] on p.50 by the use of the parallel axis theorem.

$$\mathbf{I}_b = \mathbf{I}_g - m\mathbf{S}^2(\mathbf{r}_g^b) = \mathbf{I}_g - m(\mathbf{r}_g^b(\mathbf{r}_g^b)^T - (\mathbf{r}_g^b)^T\mathbf{r}_g^b\mathbf{I}_{3 \times 3}) \quad (2.27)$$

The inertia matrix \mathbf{I}_g about center of gravity (CG) is defined according to the equations found on p. 49 in [4].

and \mathbf{r}_g^b is defined as the vector from the center of origin of {b} frame to the center of gravity. This is found according to equation (3.11) in [4]

$$\mathbf{r}_{g/w} = \mathbf{r}_{b/w} + \mathbf{r}_g \quad (2.28)$$

Expanded \mathbf{M}_{RB} can be written as follows.

$$\mathbf{M}_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m & 0 & -mz_g & 0 & mx_g \\ 0 & 0 & m & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_x & -I_{xy} & -I_{xz} \\ mz_g & 0 & -mx_g & -I_{yx} & I_y & -I_{yz} \\ -my_g & mx_g & 0 & -I_{zx} & -I_{zy} & I_z \end{bmatrix} \quad (2.29)$$

The total added mass matrix \mathbf{M}_A is defined as follows:

$$\mathbf{M}_A = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} = - \begin{bmatrix} \frac{\partial \mathbf{X}}{\partial \dot{u}} & \frac{\partial \mathbf{X}}{\partial \dot{v}} & \frac{\partial \mathbf{X}}{\partial \dot{w}} & \frac{\partial \mathbf{X}}{\partial \dot{p}} & \frac{\partial \mathbf{X}}{\partial \dot{q}} & \frac{\partial \mathbf{X}}{\partial \dot{r}} \\ \frac{\partial \mathbf{Y}}{\partial \dot{u}} & \frac{\partial \mathbf{Y}}{\partial \dot{v}} & \frac{\partial \mathbf{Y}}{\partial \dot{w}} & \frac{\partial \mathbf{Y}}{\partial \dot{p}} & \frac{\partial \mathbf{Y}}{\partial \dot{q}} & \frac{\partial \mathbf{Y}}{\partial \dot{r}} \\ \frac{\partial \mathbf{Z}}{\partial \dot{u}} & \frac{\partial \mathbf{Z}}{\partial \dot{v}} & \frac{\partial \mathbf{Z}}{\partial \dot{w}} & \frac{\partial \mathbf{Z}}{\partial \dot{p}} & \frac{\partial \mathbf{Z}}{\partial \dot{q}} & \frac{\partial \mathbf{Z}}{\partial \dot{r}} \\ \frac{\partial \mathbf{K}}{\partial \dot{u}} & \frac{\partial \mathbf{K}}{\partial \dot{v}} & \frac{\partial \mathbf{K}}{\partial \dot{w}} & \frac{\partial \mathbf{K}}{\partial \dot{p}} & \frac{\partial \mathbf{K}}{\partial \dot{q}} & \frac{\partial \mathbf{K}}{\partial \dot{r}} \\ \frac{\partial \mathbf{M}}{\partial \dot{u}} & \frac{\partial \mathbf{M}}{\partial \dot{v}} & \frac{\partial \mathbf{M}}{\partial \dot{w}} & \frac{\partial \mathbf{M}}{\partial \dot{p}} & \frac{\partial \mathbf{M}}{\partial \dot{q}} & \frac{\partial \mathbf{M}}{\partial \dot{r}} \\ \frac{\partial \mathbf{N}}{\partial \dot{u}} & \frac{\partial \mathbf{N}}{\partial \dot{v}} & \frac{\partial \mathbf{N}}{\partial \dot{w}} & \frac{\partial \mathbf{N}}{\partial \dot{p}} & \frac{\partial \mathbf{N}}{\partial \dot{q}} & \frac{\partial \mathbf{N}}{\partial \dot{r}} \end{bmatrix} \quad (2.30)$$

Since this AUV operates under the wave affected zone, the added mass matrix M_A will be constant for any frequency of the waves. Also this AUV will also only be allowed to move at low-speed. Thus the following added mass matrix M_A can then be found.

$$\mathbf{M}_A = -diag\left(\frac{\partial \mathbf{X}}{\partial \dot{u}}, \frac{\partial \mathbf{Y}}{\partial \dot{v}}, \frac{\partial \mathbf{Z}}{\partial \dot{w}}, \frac{\partial \mathbf{K}}{\partial \dot{p}}, \frac{\partial \mathbf{M}}{\partial \dot{q}}, \frac{\partial \mathbf{N}}{\partial \dot{r}}\right) = - \begin{bmatrix} \frac{\partial \mathbf{X}}{\partial \dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\partial \mathbf{Y}}{\partial \dot{v}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\partial \mathbf{Z}}{\partial \dot{w}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial \mathbf{K}}{\partial \dot{p}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\partial \mathbf{M}}{\partial \dot{q}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\partial \mathbf{N}}{\partial \dot{r}} \end{bmatrix} \quad (2.31)$$

Where $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ are the added mass forces, and $(\dot{u}, \dot{v}, \dot{w})$ are the accelerations in the (x, y, z) directions in the frame $\{\mathbf{b}\}$.

Thus the total inertia matrix \mathbf{M} is found to be:

$$\begin{aligned} \mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_A &= \begin{bmatrix} m\mathbf{I}_{3 \times 3} & -m\mathbf{S}(\mathbf{r}_g^b) \\ m\mathbf{S}(\mathbf{r}_g^b) & \mathbf{I}_b \end{bmatrix} - \text{diag}\left(\frac{\partial \mathbf{X}}{\partial \dot{u}}, \frac{\partial \mathbf{Y}}{\partial \dot{v}}, \frac{\partial \mathbf{Z}}{\partial \dot{w}}, \frac{\partial \mathbf{K}}{\partial \dot{p}}, \frac{\partial \mathbf{M}}{\partial \dot{q}}, \frac{\partial \mathbf{N}}{\partial \dot{r}}\right) \\ &= \begin{bmatrix} m - \frac{\partial \mathbf{X}}{\partial \dot{u}} & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m - \frac{\partial \mathbf{Y}}{\partial \dot{v}} & 0 & -mz_g & 0 & mx_g \\ 0 & 0 & m - \frac{\partial \mathbf{Z}}{\partial \dot{w}} & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_x - \frac{\partial \mathbf{K}}{\partial \dot{p}} & -I_{xy} & -I_{xz} \\ mz_g & 0 & -mx_g & -I_{yx} & I_y - \frac{\partial \mathbf{M}}{\partial \dot{q}} & -I_{yz} \\ -my_g & mx_g & 0 & -I_{zx} & -I_{zy} & I_z - \frac{\partial \mathbf{N}}{\partial \dot{r}} \end{bmatrix} \end{aligned} \quad (2.32)$$

Here the computer-aided design components make a total of 16.7 kg and the electronic parts plus the sensors with a total of roughly 14.0 kg. This gives Manta-2020 a total weight of $m = 30.7$ kg.

With methods described in Kristoffer Solberg Rakstad master thesis [5], the following numbers were used on Manta-2020 \mathbf{M}_{RB} and \mathbf{M}_A respectively ([5], equations 4.37, 4.38, p.54) and where $\mathbf{r}_g^b = [0, 0, 0.1]^T$

$$\mathbf{M}_{RB} = \begin{bmatrix} 30.70 & 0 & 0 & 0 & 3.0700 & 0 \\ 0 & 30.70 & 0 & -3.0700 & 0 & 0 \\ 0 & 0 & 30.700 & 0 & 0 & 0 \\ 0 & -3.070 & 0 & 0.5032 & -0.0002 & 0.0005 \\ 3.0700 & 0 & 0 & -0.0002 & 0.4934 & 0 \\ 0 & 0 & 0 & 0.0005 & 0 & 0.9198 \end{bmatrix} \quad (2.33)$$

$$\mathbf{M}_A = \begin{bmatrix} 10.7727 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10.7727 & 0 & 0 & 0 & 0 \\ 0 & 0 & 49.7679 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0092 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0092 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.34)$$

2.2.2 The Coriolis matrix $\mathbf{C}(\mathbf{v})$

The Coriolis matrix $\mathbf{C}(\mathbf{v})$ consist of two parts $\mathbf{C}_{RB}(\mathbf{v})$ and $\mathbf{C}_A(\mathbf{v})$, which correspond to the rigid-body and added mass Coriolis matrices respectively. These terms are due to a rotation between the $\{\mathbf{b}\}$ frame and local NED frame $\{\mathbf{n}\}$ [5].

The $\mathbf{C}_{RB}(\mathbf{v})$ can be found according to equation (3.55) on p.55 in [4]. This equation is stated as follows

$$\mathbf{C}_{RB}(\mathbf{v}) = \begin{bmatrix} \mathbf{0}_{3 \times 3} & -m\mathbf{S}(\mathbf{v}_{b/w}^b) - m\mathbf{S}(\omega_{b/w}^b)\mathbf{S}(\mathbf{r}_g^b) \\ -m\mathbf{S}(\mathbf{v}_{b/w}^b) - m\mathbf{S}(\mathbf{r}_g^b)\mathbf{S}(\omega_{b/w}^b) & -\mathbf{S}(\mathbf{I}_b\omega_{b/w}^b) \end{bmatrix} \quad (2.35)$$

With the centripital matrix $\mathbf{C}_A(\nu)$ defined as: ([4], p.120, equation (6.43))

$$\mathbf{C}_A(\nu) = \begin{bmatrix} \mathbf{0}_{3 \times 3} & -\mathbf{S}(\mathbf{A}_{11}\nu_1 + \mathbf{A}_{12}\nu_2) \\ -\mathbf{S}(\mathbf{A}_{11}\nu_1 + \mathbf{A}_{12}\nu_2) & -\mathbf{S}(\mathbf{A}_{21}\nu_1 + \mathbf{A}_{22}\nu_2) \end{bmatrix} \quad (2.36)$$

2.2.3 The damping matrix $\mathbf{D}(\mathbf{v})$

The damping terms consist of five different effects. Potential damping, skin friction, wave drift damping, damping due to vortex shedding and drag/lifting forces. Potential damping occurs when a body is forced to oscillate with the wave excitation frequency. Since AUVs are not operating at the surface in the wave affected zone. This damping will be assumed zero. This damping will be denoted \mathbf{D}_P . Skin friction is a resistant force acting on an object in either laminar or turbulent viscous fluid. Wave drift damping will also be considered to be zero, because the AUV operates below the wave affected zone. Damping due to vortex shedding occurs because of frictional forces due to vortex sheets arises [4] p.122. The drag/lifting forces arise when there is a linear circulation of water around the hull of the AUV and by cross-flow drag. These damping terms consist of a linear part and nonlinear part. The linear part will be denoted \mathbf{D}_L and the nonlinear term $\mathbf{D}_{NL}(\mathbf{v}_r)$. Where \mathbf{v}_r denotes the relative velocity vector.

Thus the damping matrix $\mathbf{D}(\mathbf{v})$ on Manata-202 can be written as

$$\mathbf{D}(\mathbf{v}) = \mathbf{D}_L + \mathbf{D}_{NL}(\mathbf{v}_r) \quad (2.37)$$

Where the linear viscous damping \mathbf{D}_L and \mathbf{D}_{NL} are calculated as follows [4].

$$\mathbf{D}_L = - \begin{bmatrix} \frac{\partial \mathbf{X}}{\partial u} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\partial \mathbf{Y}}{\partial v} & 0 & \frac{\partial \mathbf{Y}}{\partial p} & 0 & \frac{\partial \mathbf{Y}}{\partial r} \\ 0 & 0 & \frac{\partial \mathbf{Z}}{\partial w} & 0 & \frac{\partial \mathbf{Z}}{\partial q} & 0 \\ 0 & \frac{\partial \mathbf{K}}{\partial v} & 0 & \frac{\partial \mathbf{K}}{\partial p} & 0 & \frac{\partial \mathbf{K}}{\partial r} \\ 0 & 0 & \frac{\partial \mathbf{M}}{\partial w} & 0 & \frac{\partial \mathbf{M}}{\partial q} & 0 \\ 0 & \frac{\partial \mathbf{N}}{\partial v} & 0 & \frac{\partial \mathbf{N}}{\partial p} & 0 & \frac{\partial \mathbf{N}}{\partial r} \end{bmatrix} \quad (2.38)$$

$$\mathbf{D}_{NL}(\nu) = - \begin{bmatrix} X_{|u|u}|u_r| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{|v|v}|v_r| & 0 & 0 & 0 & Y_{|v|r}|v_r| \\ 0 & 0 & Z_{|w|w}|w_r| & 0 & 0 & 0 \\ 0 & K_v & 0 & K_{|p|p}|p| & 0 & K_r \\ 0 & 0 & M_w & 0 & M_{|q|q}|q| & 0 \\ 0 & N_{|v|v}|v_r| & 0 & 0 & 0 & N_{|v|r}|v_r| \end{bmatrix} \quad (2.39)$$

Thus the total \mathbf{D}_ν then becomes

$$\mathbf{D}(\mathbf{v}) = \mathbf{D}_L + \mathbf{D}_{NL}(\mathbf{v}_r)$$

$$= - \begin{bmatrix} X_{|u|u}|u_r| + \frac{\partial \mathbf{X}}{\partial u} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{|v|v}|v_r| + \frac{\partial \mathbf{Y}}{\partial v} & 0 & \frac{\partial \mathbf{Y}}{\partial p} & 0 & Y_{|v|r}|v_r| + \frac{\partial \mathbf{Y}}{\partial r} \\ 0 & 0 & Z_{|w|w}|w_r| + \frac{\partial \mathbf{Z}}{\partial w} & 0 & \frac{\partial \mathbf{Z}}{\partial q} & 0 \\ 0 & K_v + \frac{\partial \mathbf{K}}{\partial v} & 0 & K_{|p|p}|p| + \frac{\partial \mathbf{K}}{\partial p} & 0 & K_r + \frac{\partial \mathbf{K}}{\partial r} \\ 0 & 0 & M_w + \frac{\partial \mathbf{M}}{\partial q} & 0 & M_{|q|q}|q| + \frac{\partial \mathbf{M}}{\partial q} & 0 \\ 0 & N_{|v|v}|v_r| + \frac{\partial \mathbf{N}}{\partial v} & 0 & \frac{\partial \mathbf{N}}{\partial p} & 0 & N_{|v|r}|v_r| + \frac{\partial \mathbf{N}}{\partial r} \end{bmatrix} \quad (2.40)$$

For manta-2020 the following values were found [5].

$$\mathbf{D}_L = \begin{bmatrix} 19.5912 & 0 & 0 & 0 & 0 & 0 \\ 0 & 19.5912 & 0 & 0 & 0 & 0 \\ 0 & 0 & 50.5595 & 0 & 0 & 0 \\ 0 & 0 & 0 & 13.3040 & 0 & 0 \\ 0 & 0 & 0 & 0 & 13.218 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.1559 \end{bmatrix} \quad (2.41)$$

and $\mathbf{D}_{NL}(\mathbf{v}_r)$

$$\mathbf{D}_{NL}(\mathbf{v}_r) = \begin{bmatrix} 7.3486|u_r| & 0 & 0 & 0 & 0 & 0 \\ 0 & 7.3486|v_r| & 0 & 0 & 0 & 0 \\ 0 & 0 & 26.1105|w_r| & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.42)$$

2.2.4 The restoring forces $\mathbf{g}(\eta)$

The restoring forces that acts on the AUV consist of gravitational and buoyancy forces. This can be represented as follows [4]

$$\mathbf{g}(\eta) = - \begin{bmatrix} \mathbf{R}_b^n(\mathbf{q})^{-1}(\mathbf{f}_g^n + \mathbf{f}_b^n) \\ \mathbf{r}_g^b \times \mathbf{R}_b^n(\mathbf{q})^{-1}\mathbf{f}_g^n + \mathbf{r}_b^b \times \mathbf{R}_b^n(\mathbf{q})^{-1}\mathbf{f}_b^n \end{bmatrix} \quad (2.43)$$

Where \mathbf{f}_g^n and \mathbf{f}_b^n are defined as follows [4]

$$\mathbf{f}_g^n = \begin{bmatrix} 0 \\ 0 \\ W \end{bmatrix} \quad (2.44)$$

$$\mathbf{f}_b^n = - \begin{bmatrix} 0 \\ 0 \\ B \end{bmatrix} \quad (2.45)$$

The W is denoted as the weight of the AUV. This is defined as

$$W = mg \quad (2.46)$$

The weight has its origin in its center of gravity. On manta-2020 this is easily calculated to be with mass $m = 30.7kg$ and with gravity as the regional gravity $g = 9.825$ defined in equation 3.2, as follows

$$W = 30.7 * 9.825 = 301.6275 \text{ N} \quad (2.47)$$

The definition of B is defined as the buoyancy force and is defined as

$$B = \rho g \nabla \quad (2.48)$$

Here ρ is defined as the density of fresh water, g as the gravitational acceleration and ∇ as the displaced fluid by the AUV. On manta-2020 it assumed that the AUV is neutrally bouyant and that $W = B$.

Inertial Navigation Systems

An inertial navigation system is a system that uses inertial sensor measurements and performs mathematical equations and data processing, to estimate position, linear velocity and attitude (e.g PVA). This navigation system will then have the important role of providing accurate and reliable estimates to the control system. Some of the written material in this section will be directly from the author's project thesis [1].

3.1 The IMU

One inertial sensor that used in inertial navigation is the IMU. This sensor usually consist of 3-axis accelerometers and 3-axis gyros, which provide measurements of acceleration and angular velocity. With this it is possible to integrate the acceleration and angular velocity measurements to provide PVA estimation. However this system is a type of a dead reckoning system which will drift over time. This is because of the inaccuracies and errors that the accelerometers and gyros in IMU have. It is therefore very important, especially in underwater environments where all 6 DOF is important and no GNSS system to provide absolute position, to have a reliable IMU as possible. In this section the MEMS accelerometer, MEMS gyros and their error characteristics will be discussed.

3.1.1 Accelerometers

Broadly, there are according to [22] two main types of accelerometers, mechanical and solid state. In this thesis only the mechanical accelerometer will be presented. The reader is referred to [22] for more detail about the solid state device approach. Also an introduction to micro electrical mechanical system (MEMS) accelerometers will be discussed.

The mechanical accelerometer is based upon a mass that is attached to a spring. By measuring its deflection, the specific force is measured from newton's law $F = ma$ in its input direction. Since newton laws applies only in an inertial frame the spesific force will be denoted as $f_{b/i}^b$ for strapdown INS. However, if local navigation is used where $\{n\}$ is assumed inertial, the force can be stated as $f_{b/n}^b$. By combining three accelerometers together, with orthogonal input directions that forms a three dimensional Cartesian coordinate system, a 3D specific force is measured $\mathbf{f}_{b/w}^b$. If the the accelerometer is defined with a 3D axis Cartesian right handed

coordinate system with a strapdown approach, it should measure

$$\mathbf{f}_{b/n}^b = \begin{bmatrix} 0 \\ 0 \\ +9.8xxx \end{bmatrix} \quad (3.1)$$

Where $9.8xxx$ is a local gravity. This local gravity can be calculated with the WGS84 ellipsoidal gravity formula by the use of the latitude in the specific region as follows[23].

$$\mathbf{g} = 9.7803253359 \left[\frac{1+0.00193185265265241\sin^2(\phi)}{\sqrt{(1-0.00669437999013\sin^2(\phi))}} \right] \frac{m}{s^2} \quad (3.2)$$

The MEMS based accelerometers is according to [22] based on the same principles as the mechanical and solid state methods. But comes with some advantages over traditional techniques, such as low power consumption, short start-up times, light weighted and are small in size.

For the Tyholt region in Trondheim, where the experimental testing took place, this value was calculated to

$$\mathbf{g} = 9.7963744538548 \quad (3.3)$$

with a latitude of 63.420164 converted in to decimal degrees.

3.1.2 Gyroscopes

According to [22], there are three main types of gyroscopes, the mechanical, optical and MEMS gyroscopes. Here the MEMS gyroscope will be presented.

The MEMS gyroscope use the Coriolis effect, which says that within a rotating reference frame with angular velocity ω , a mass m and velocity v , it experiences a force \mathbf{F} [22]. This reference frame is inertial and is relative to the $\{i\}$ frame.

$$\mathbf{F} = -2m(\omega_{b/i}^{gyro} \times \mathbf{v}_{b/i}^{gyro}) \quad (3.4)$$

Where \times denotes the cross product. MEMS gyros contains vibrating elements to measure the Coriolis effect[22]. This is done with a single mass driven to vibrate along a drive axis. If the gyroscope is rotated a secondary vibration is induced along the perpendicular sense axis. Thus here the angular velocity $w_{b/i}^b$ is measured by measuring this secondary rotation. These types of gyros are typically cheaper and less accurate than mechanical, or optical gyroscopes [24]. Also in this case, since $\{w\}$ is assumed inertial, the equation becomes $w_{b/w}^b$. Gyroscopes also measures the earth angular rotation rate, which was stated in chapter 2.1 as being $w = 7.2921 * 10^{-5}$ which should be compensated for. Since this is a small value, this error source can be neglected. By the use of 3-axis gyroscopes, a 3D angular velocity $\omega_{b/w}^b$ can be measured.

3.1.3 Error characteristics

Inertial sensors have many different types of errors that have to be accounted for in an inertial navigation system. Some of these errors are biases, scale-factor errors, cross-coupling, random noise and nonlinearities [6]. If these are not accounted for or evaluated when choosing an IMU, the integration of noise, scale-factor and bias corrupted measurements will produce large errors in position, velocity and attitude estimation [25]. This is especially true when the IMU only is used to track 6DOF estimations. There are also other error-characteristics like G-dependencies, but this will not be discussed in this thesis.

The bias error is composed of two components, as shown in the equation 3.5 for the accelerometer and angular rate sensor respectively. The static part is a run-run bias that is constant for each time the IMU is powered [6]. This constant is both dependent on the bias repeatability and changes its value each time the IMU powers on and off. This bias contributes to around 90% of the total bias [6]. The second term is the dynamic bias, which is an in-run bias A run-to-run static part $\mathbf{b}_{acc,static}^b$ that is constant for each time the IMU is powered[6]. This constant is both dependent on the bias repeatability and changes each time the IMU powers on and off[6]. This bias contributes to around 90% of the total bias. The second term is a dynamic bias which is an in-run bias that depends on the bias stability [6]. Together these biases often contribute to the most dominant error source.

$$\mathbf{b}_{acc}^b = \mathbf{b}_{acc,static}^b + \mathbf{b}_{acc,dynamic}^b \quad \mathbf{b}_{ars}^b = \mathbf{b}_{ars,static}^b + \mathbf{b}_{ars,dynamic}^b \quad (3.5)$$

The scale factor error is based on the discrepancy of the gradient between the true motion and the measurement [6]. This error is a propotional constant to the true motion input. Fortunately this error can be corrected for by factory calibration. These scale factors can be described as shown in equation 3.6.

$$\mathbf{S}_{acc}^b = \begin{bmatrix} s_{acc,x}^b \\ s_{acc,y}^b \\ s_{acc,z}^b \end{bmatrix} \quad \mathbf{S}_{ars}^b = \begin{bmatrix} s_{ars,x}^b \\ s_{ars,y}^b \\ s_{ars,z}^b \end{bmatrix} \quad (3.6)$$

Cross-coupling is one of the errors that is IMU-specific only. It describes the orthogonality error between the sensitivity axes of the angular rate and accelerometer sensors and the body frame. These errors are possible to factory calibrate. These errors can be written as in equation 3.7 [6].

$$\mathbf{M}_{acc}^b = \begin{bmatrix} s_{acc,x}^b & m_{acc,xy}^b & m_{acc,xz}^b \\ m_{acc,yx}^b & s_{acc,y}^b & m_{acc,yz}^b \\ m_{acc,zx}^b & m_{acc,zy}^b & s_{acc,z}^b \end{bmatrix} \quad \mathbf{M}_{ars}^b = \begin{bmatrix} s_{ars,x}^b & m_{ars,xy}^b & m_{ars,xz}^b \\ m_{ars,yx}^b & s_{ars,y}^b & m_{ars,yz}^b \\ m_{ars,zx}^b & m_{ars,zy}^b & s_{ars,z}^b \end{bmatrix} \quad (3.7)$$

Where the diagonal terms are the scale factors from equation 3.6

IMUs are also affected by random noise. These noise terms can be modeled as white Gaussian noise. They are exhibited by all inertial sensors, and are not specific for IMU's only. Especially for MEMS IMU's , electrical noise limits the resolution of the sensors. These errors are also dependent on frequency and range. These noise terms will be denoted as \mathbf{w}_{acc}^b for the accelerometers and \mathbf{w}_{ars}^b for the angular rate sensors. For 3-axis measurements these are described as follows.

$$\mathbf{w}_{acc}^b = \begin{bmatrix} w_{acc,x} \\ w_{acc,y} \\ w_{acc,z} \end{bmatrix} \quad \mathbf{w}_{ars}^b = \begin{bmatrix} w_{ars,x} \\ w_{ars,y} \\ w_{ars,z} \end{bmatrix} \quad (3.8)$$

IMU's are also affected by mounting errors that can be described by a static rotation matrix. This occurs because it is very hard to perfectly mount the IMU such that it is aligned with the {b} coordinate frame. This can however be fixed by simply defining the {b} coordinate axis to be aligned with the IMU coordiante axis, but then this could for example be a problem later if the estimates were to control heading. These misalignment static rotation matrices will be denoted for the acceleormeter and gyros as follows.

$$\mathbf{R}_b^{acc}(\mathbf{q}) \quad \mathbf{R}_b^{gyro}(\mathbf{q}) \quad (3.9)$$

Together the 3-axis IMU measurements together with bias, scale factor, cross-coupling, static misalignment error can be expressed as follows[6].

$$\mathbf{a}_{imu}^{acc} = \mathbf{R}_b^{acc}(\mathbf{q})(\mathbf{I}_{3x3} + \mathbf{M}_{acc})\mathbf{R}_w^b(\mathbf{q})(\dot{\mathbf{v}}_{b/w}^w - \mathbf{g}^w) + \mathbf{b}_{acc}^b + \mathbf{w}_{acc}^b \quad (3.10)$$

and

$$\omega_{imu}^{gyro} = \mathbf{R}_b^{gyro}(\mathbf{q})(\mathbf{I}_{3x3} + \mathbf{M}_{acc})\mathbf{R}_w^b(\mathbf{q})(\omega_{b/w}^b) + \mathbf{b}_{ars}^b + \mathbf{w}_{ars}^b \quad (3.11)$$

All of the discussed error-sources will behave differently on the measurement outputs. This can be further seen in figure 3.1. Here it is seen that with no errors the input u and output y has the same output as input. Looking at the scale-factor plot it can be seen that it can potentially give greater errors than the biases. Combining them results in a scaled, biased and noisy measurement as seen in the lower right figure.

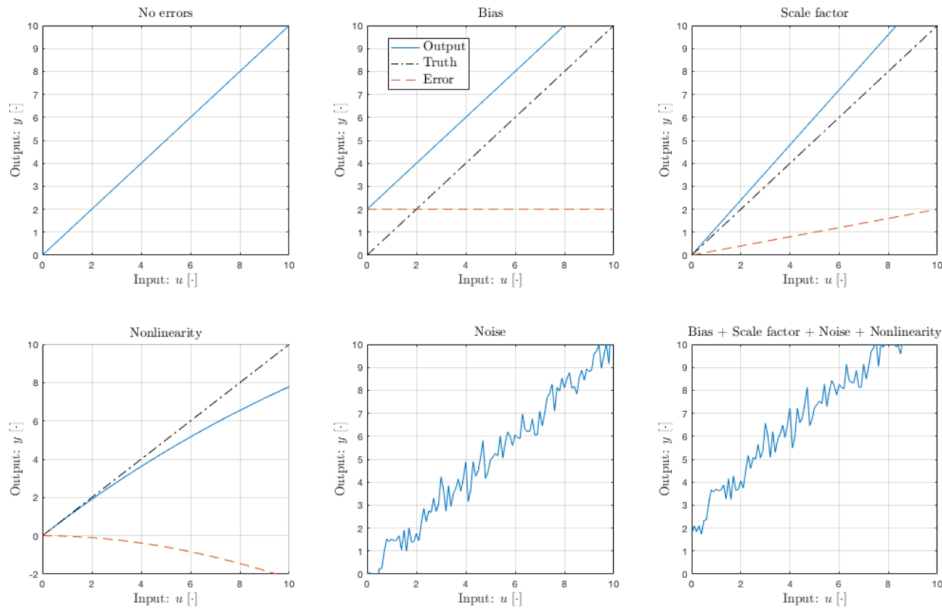


Figure 3.1: IMU error-sources. Figure from [6]

Looking at the error sources it is therefore critical to find the correct local gravity, factory calibrate the IMU to remove most of the scale factor and cross-coupling errors and estimation of the bias and noise terms.

3.1.4 Allan variance

As described in the previous sections, multiple sources of errors arise for accelerometers and gyros, with one of them being the noise terms \mathbf{w}_{acc}^b and \mathbf{w}_{gyro}^b . These are modeled as Gaussian distributions with zero mean and a variance. With the Allan variance method, it is possible to determine these variances. In MEMS accelerometers and gyros the biases \mathbf{b}_{acc}^b and \mathbf{b}_{ars}^b may wander over time, due to flicker noise. This noise can be seen as an bias driving noise. With the Allan variance method it is possible to determine the variance of this noise also.

According to [22] the Allan variance is a time domain analysis for characterizing noise and stability of a signal. The Allan variance method can be described as follows [22].

1. Take a long sequence of data and divide it into bins of length t .
2. Average the data in each bin to obtain a list of averages $(a(t_1), a(t_2), \dots, a(t_n))$, where n is the number of bins.
3. The Allan variance can then be calculated as follows

$$AVAR(t) = \frac{1}{2(n-1)} \sum_i (a(t_{i+1}) - a(t_i))^2 \quad (3.12)$$

with the Allan deviation as

$$AD(t) = \sqrt{AVAR(t)} \quad (3.13)$$

Normally the Allan variance for the accelerometers and gyros are specified in the data-sheet as their respective random walk and bias instability found in their functional specifications.

3.1.5 Wild point filtering

Incoming measurements signals coming from any sensor can have many erroneous sources as shown in figure 3.2. For the errors shown in the figure, there are wild points, high variance, frozen signals and high derivatives. Such measurements may give potentially large errors in the state estimation and should be checked and captured. For both the accelerometer and gyro measurements coming from the IMU, it is therefore important to perform some form of signal processing. This could for example be in the form of a range-check, variance check or a wild point detection and removal. The range check could be done by defining a specific range on the output and discarding its output if its below or above the range. The variance check can be used to indicate if the sensor has high variance output or if the measurement is a frozen signal if it has low variance. The wild point detection and removal can be used to sense if there are large spikes in the sensor output that corresponds to a measurements which is far off from the neighborhood measurements.

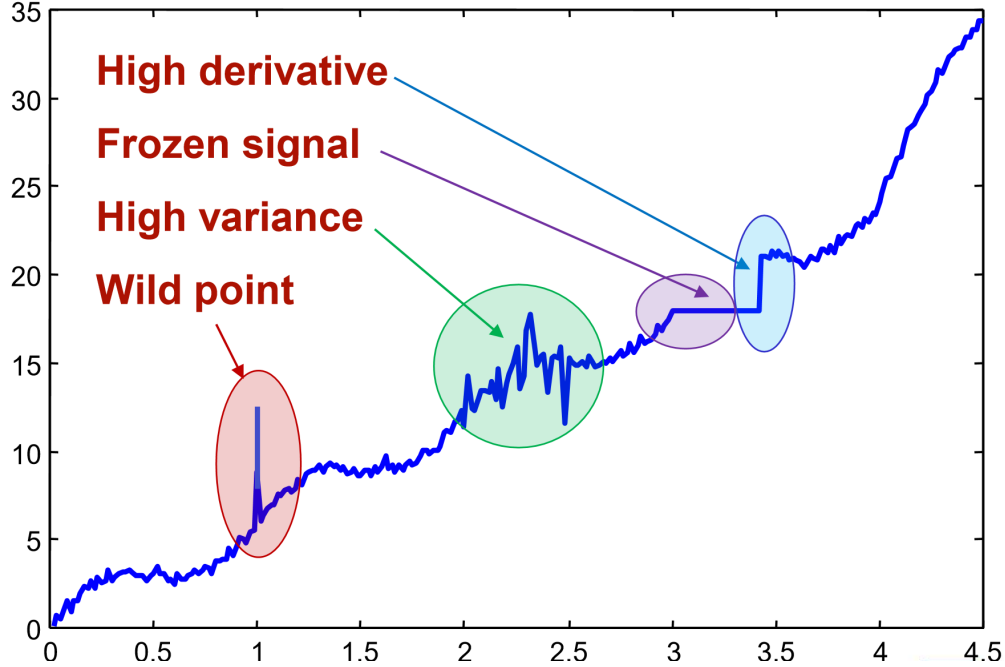
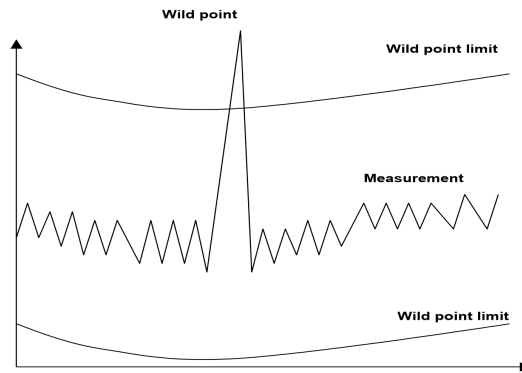


Figure 3.2: Incoming sensor signal errors. Courtesy([7])

Maybe the most critical error to consider for the IMU is the wild point detection and removal. In the state estimators such large spikes can especially lead to poor estimation if the IMU measurements are directly used in their prediction step, like the ESKF and the NLO presented in this thesis. For measurements in acceleration, such spikes occur if the vessel bumps into an object.

There are several approaches to wild point filtering. The one presented in [7] is to discard estimates that is outside a range around the estimated signal mean σ . This measurement signal value will then be rejected for one sample. Figure 3.3 shows a visual representation of this.



$$x[k] \in [\bar{x}_k - a\sigma, \bar{x}_k + a\sigma]$$

Figure 3.3: Wild point filtering method Courtesy([7])

The acceleration and gyro wild point filter implemented by the author for the IMU on Manta-2020 is based

on Brages master thesis [26] wild point filter implementation stated below.

```
if  $|a_i - a_{i_{prev}}| < a_{tolerance}$  or dropped measurements  $>$  max dropped allowed
   $a_i \rightarrow$  approved
  dropped measurement = 0
else
   $a_i \rightarrow$  rejected
  dropped measurement + = 1
```

Where the subscript $i = x, y, z$ denotes the corresponding acceleration axis.

Here the wild point filter takes the absolute value of the current and previous measurement and sees if this is less than a tolerance. This tolerance is a tuning parameter that can be specified by the user. If this is true the current measurement becomes approved and sent. Otherwise it becomes rejected. In addition to this, this wild point filter also sets a limit on how many of the measurements that can be dropped in the "max dropped allowed" variable. This is recommended because if not implemented it can potentially "wild point filter" too many measurements consecutively.

For the gyro a slightly different tolerance methodology was used. This is defined as follows [26].

$$|\omega_i - \omega_{i_{prev}}| > \max(2 \cdot |\omega_{i_{prev}}|, \omega \text{ noise threshold})$$

Where ω is the noise threshold set slightly larger than the largest observed peak-to-peak noise.

This method is based on the assumption that an object which is hitting another unmovable object, will be fully elastic in a system without friction. The system will then have the same velocity after the collision, but then only with a angle of direction that is mirrored[26].

The figure 3.4 shows how the wild point filter drops measurements coming from the a_x accelerometer axis.

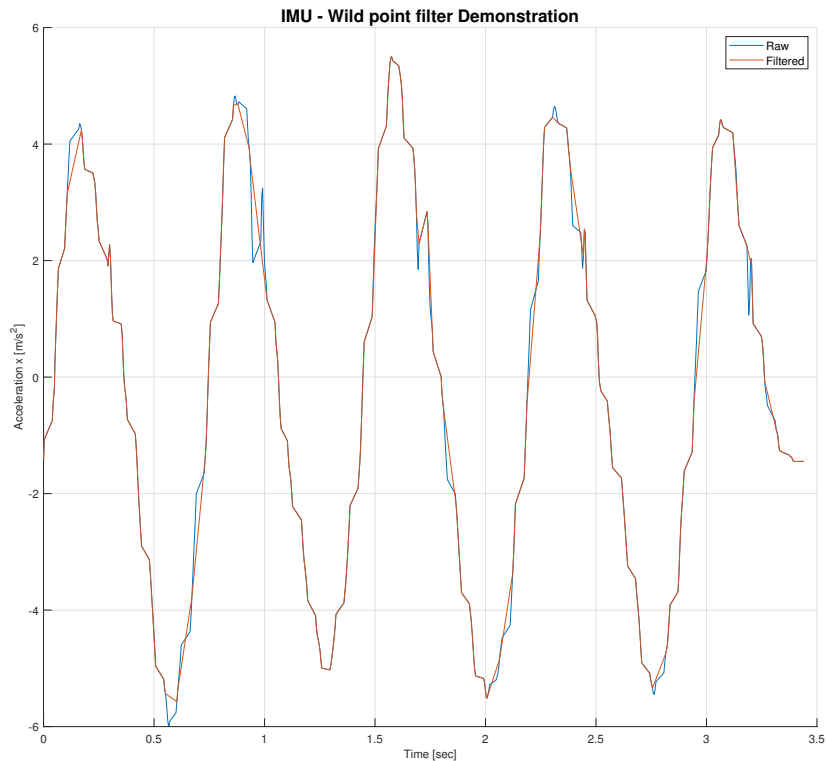


Figure 3.4: Wild point filter demonstration

3.1.6 Prolonged prediction

For the ESKF and NLO the gyro and acceleration measurements are directly used in their prediction. If the aiding sensor measurements drops out, these filters runs a prolonged 6 DOF prediction of the states where the acceleration is integrated twice to get position and once to get velocity. The gyro is integrated once to give attitude. Since the IMU have several error sources discussed above, these errors will propagate during prediction. For a relatively slowly moving system, like Manta-2020 these errors will have the largest contributions to the propagation errors than the actual dynamics of the system. Drifting will then occur rapidly. One solution to this is to implement gyro and accelerometer bias estimation. The 3-DOF acceleration biases will then be observable for aiding sensors that measures absolute 3 DOF position, like a GNSS. For Manta-2020 the pressure sensor measures absolute vertical position, which will render the a_z bias observable. For gyro bias estimation will the ω_x and ω_y biases become observable by knowing the absolute direction of the gravity vector. Fortunately this is given by the accelerometer as stated in equation 3.1. Then with bias estimation the prolonged prediction will hopefully give less inaccurate estimates.

3.1.7 Inclination estimate

The ideal measurement coming from the accelerometers when the AUV is at rest, is shown in 3.1. Since the gravity vector is measured, it is then possible to calculate the initial roll and pitch estimate by the following

equations.

$$\phi = \text{atan2}(a_{imu_y}^{acc}, a_{imu_z}^{acc}) \quad (3.14)$$

$$\theta = \text{atan2}(-a_{imu_x}^{acc}, \sqrt{(a_{imu_y}^{acc})^2 + (a_{imu_z}^{acc})^2}) \quad (3.15)$$

Using these inclination equations as roll ϕ and pitch θ measurements directly into state estimators is not recommended. This is because it is hard to distinguish between the biases and the specific force exerted on the AUV in relationship to the gravity vector.

On Manta-2020 these inclinations estimates are used to find the initial mounting error $\mathbf{R}_b^{acc}(\mathbf{q})$ and $\mathbf{R}_b^{gyro}(\mathbf{q})$. This was done by taking an average of M samples, finding their respective roll and pitch inclination value and then taking the mean. This was done to minimize the effect of biases and errors. The author this time implemented an ROS-service in the device driver of the IMU. By calling the message "calibrate" these mean values were directly given to the user. It was then possible to put these values in the "alignment error" parameter in the tuning parameter list given as a yaml file. This can be clearer seen in the figure 3.5.

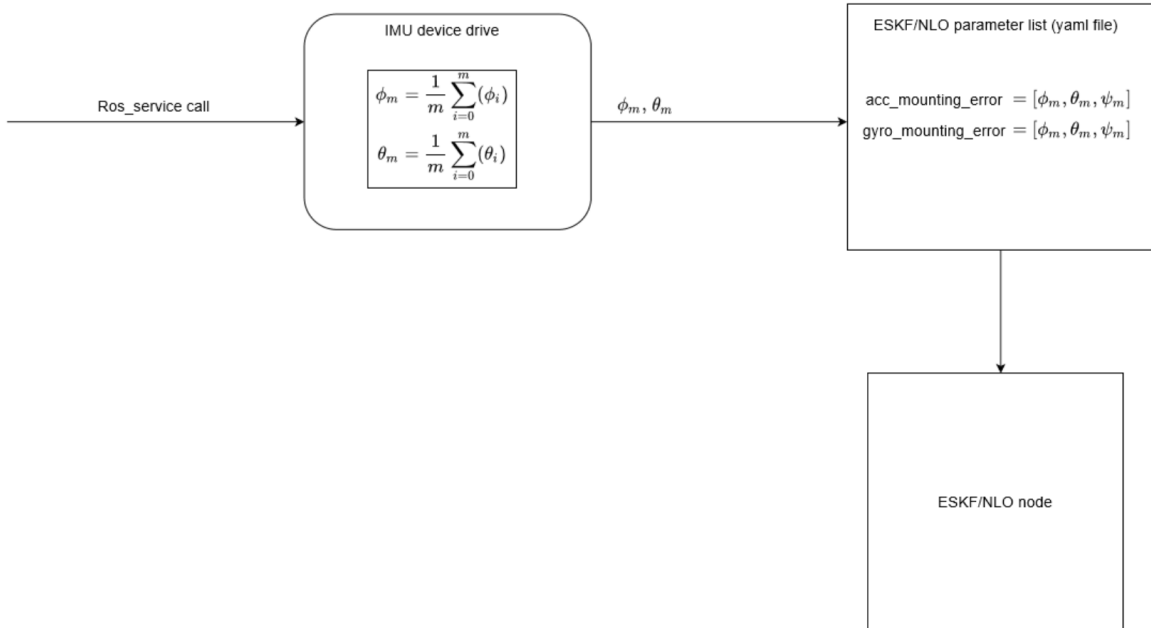


Figure 3.5: IMU mounting method

3.2 Aided inertial navigation

As described above an INS alone is not enough to get accurate estimates of the integrated quantities. To limit the drift occurred, the INS must be aided by other sensors that provide direct measurements of the integrated quantities. These may be GNSS, pressure meter, magnetic compasses, Doppler velocity log (DVL) or underwater transponders. In this section, the DVL and the pressure sensor will be discussed.

3.2.1 Doppler velocity log

A Doppler velocity log is an acoustic sensor that provides linear velocity measurements with respect to the sea bottom or water. The measurement principle in the DVL is the Doppler effect. This effect describes the frequency shift that occurs when moving towards or away from a relative object or target. The DVL uses transducer beams together with a long pulse to achieve this. This is done by sending the long pulse vertically to the bottom and at least three transducer beams in different directions. This can be seen in the figure below.

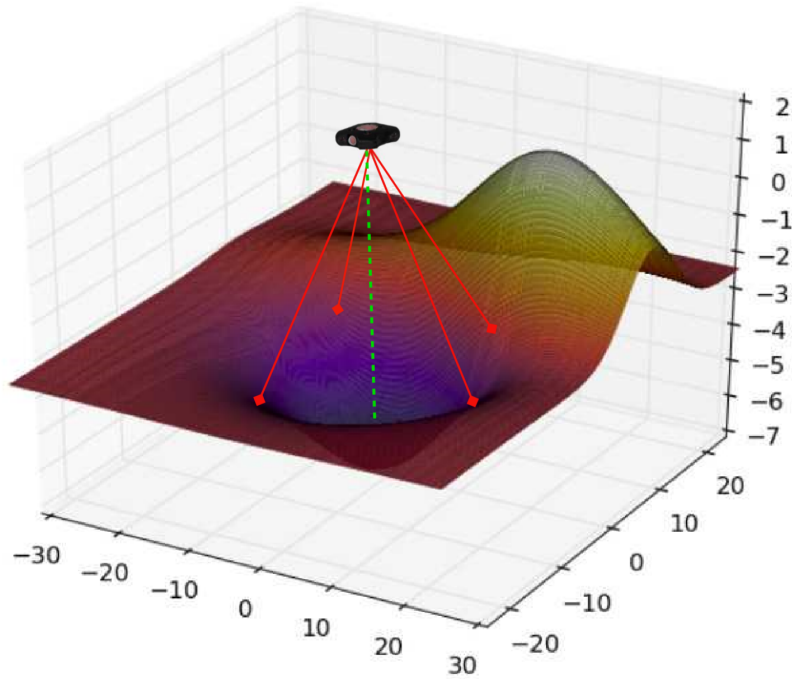
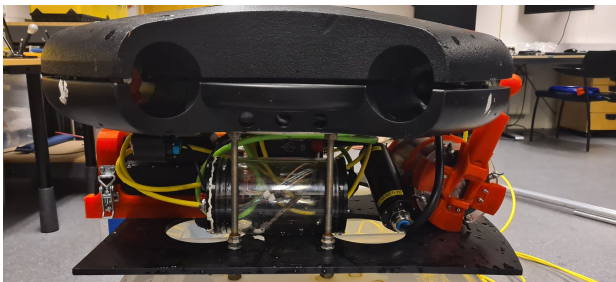


Figure 3.6: DVL work principle on the Manta-2020 AUV. Courtesy([5])

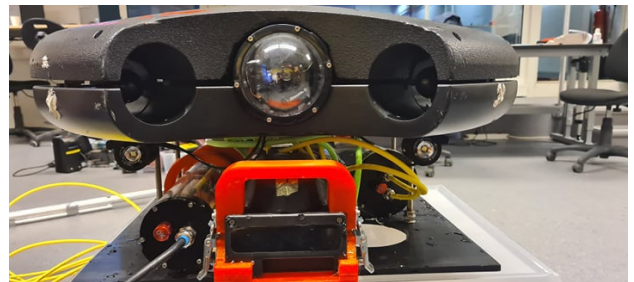
The AUV of interest - Manta 2020

4.1 Concept

Manta is an inspection-class hovering AUV designed and developed by students of the student-organization Vortex-NTNU. Manta both serves as an experimental underwater platform for students writing their master thesis and as an AUV that competes in the Robosub competition. The main idea is to give students practical experience with underwater robotics and foster ties to organizations and companies developing AUV technologies.



(a) Side view



(b) Front view

Figure 4.1: The AUV of interest - Manta 2020

4.2 Sensor and thruster stack

In order for the AUV to perform accurate maneuvers from the guidance and control system, it is critical that the navigation system is robust. An important part of this robustness comes from having accurate and precise sensors. The choice of sensors therefore plays an important part for the autonomy. What type of sensors also plays an critical role. By choosing sensors that provides measurements of the navigational states, will increase the observability of the GNC system. By having too few observable states will limit the control and guidance system, meaning less robust maneuvering in the AUV's mission plan. For example in trajectory and path following. Manta-2020 is here equipped with an IMU from Sensoror, STIM300, Blue Robotics Low-light HD Camera used for object detection, FLIR Blackfly S camera to use for object detection and SLAM, Gemini 720i-imaging sonar and a Nortek DVL1000. All sensors used on the AUV is seen in

figure 4.2.

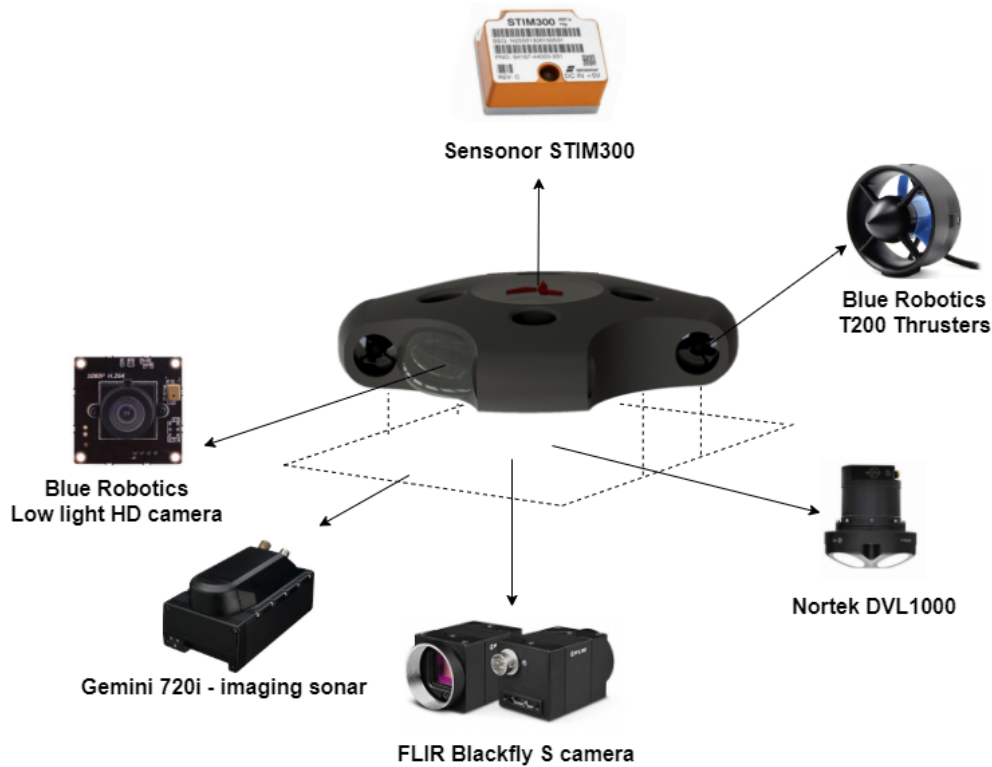


Figure 4.2: Manta-2020 sensor stack

The design and choice of actuators also plays a critical role. Their respective location and number determines the controllability of the system, which in the end determines how many degrees of freedom the controller can control. Here Manta-2020 is equipped with 8 T200 Blue Robotics thrusters, which is a three-phase out-runner motor, which run with a nominal voltage between 12-16 volt, with a maximum of 20 volt [27].

4.2.1 Interoceptive sensors

As seen in figure 4.2, Manta currently relies on three main interoceptive sensors to provide 6 DOF estimates of the navigational states. The first main sensor is a MEMS based IMU from Sensoron [28], STIM300. This sensor contains 3-axis accelerometers, 3-axis angular rate sensors and 3-axis inclinometers [8]. This sensor provides measurements of the AUV's acceleration and angular velocity. On Manta-2020 this sensor is mounted in the center of gravity in a strap-down approach. The second main interoceptive sensor is an DVL. This sensor measures velocity w.r.t the seafloor or water. Also equipped with the DVL is an inbuilt pressure gauge to measure depth. These sensors will be the main sensors to be used in the state estimation.

4.2.2 Exteroceptive sensors

Also seen in figure 4.2, Manta is also equipped with three exteriorceptive sensors. The Low light HD camera is used as the primary source for close-up front object detection. This camera has a resolution of 2.24 MP and a supply voltage of 5V. This camera is ideally suited to be used in underwater environments, because of its low-light performance, color handling and on-board video compression [29]. The second exteriorceptive

sensor is the FLIR Blackfly S camera. This is used for bottom object detection and SLAM. These cameras are ideal for integration with computer vision, because of its features such as manual control over image capture and on-camera pre-processing [30]. Some of these features include precise control of acquisition, gain, black level, white balance, color transformation and transfer [31]. The newly installed Gemini 720i - imaging sonar is used as an long range "2D scanner" which is mainly used for geographical mapping of the underwater environment and SLAM.

4.3 The IMU - STIM300

The first main sensor used on Manta-2020 is an tactical-graded IMU (STIM300) from Sensoror. This sensor contains 3-axis MEMS-based angular rate sensors, 3-axis MEMS-based accelerometers and 3-axis inclinometers. It has an operating temperature between 40.0°C and $+85.0^{\circ}\text{C}$. The sensor uses the RS422 protocol for communication with 1 start bit, 8 data length bits and 1 stop bit. It also contains a -3dB low pass filter on each output. The startup time plus time to get valid data is a maximum of 6 seconds. It is also possible to customize sample rate, output units, low pass filter dB frequency and RS422 bit-rate. For Manta-2020 the STIM300 have the following parameters set

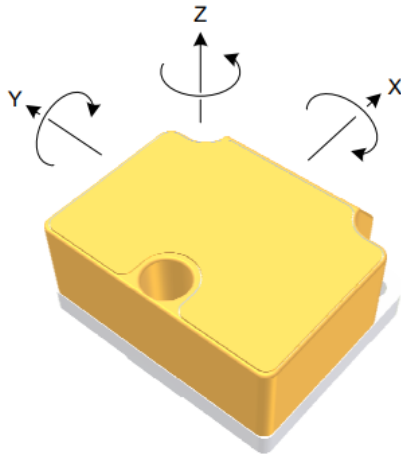
Parameters	Units
Sample rate	125 samples/s
Low-pass filter -3dB frequency	66 Hz
Measurement output units ARS	Angular rate [$^{\circ}/s$]
Measurement output units Accelerometer	Acceleration [g]
RS422 BIT-RATE	921600 bits/s

Table 4.1: Parameters set in the STIM300 IMU

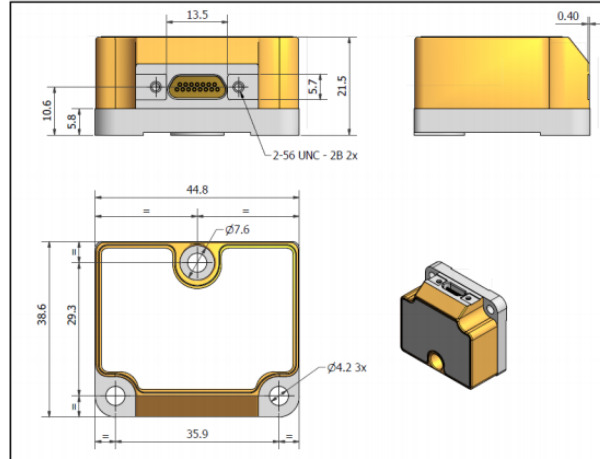
Also this IMU has an inbuilt synchronization signal named TOV that will synchronize the sensor channels. The IMU also performs CRC for fault detection. On Manta-2020, the input voltage is set to 5V and where the AUX+ and AUX- are not used. The output channels of the IMU is connected with a transmission over USB 2.0 which has an theoretical maximum transfer speed of 480 *Mbits/s*[32].

4.3.1 Axis

The STIM300 axis is defined as shown in figure 4.3a. The definition of axes are needed in order to define the correct static correction matrix between the $\{imu\}$ and $\{body\}$ frame. In figure 4.3a, the mechanical dimensions of the IMU is displayed. On Manta-2020 the STIM300 is placed as an strap-down approach, which means it is rigidly attached to the unit. As seen in figure 4.8 the location of the IMU lies in the center of origin $\{CO\}$. As described in Chapter 2, the center of origin $\{CO\}$ is chosen to be equal to the center of gravity $\{CG\}$.



(a) The STIM300 definition of axes



(b) The mechanical dimensions of the STIM300

Figure 4.3: The STIM300 axes and mechanical dimensions. Figure from [8]

From the STIM300 datasheet, table 4.4 and 4.5 specifies the functional specification parameters for the angular rate sensors and the accelerometers.

Parameter	Conditions	Min	Nom	Max	Unit	Note
ACCELEROMETER						
Full Scale (FS)			±10		g	1
Resolution			24		bits	
Scale Factor Accuracy			1.9		µg	
Non-Linearity	±10g		±300		ppm	2
Bandwidth (-3dB)	LP-filter -3dB = 262Hz	90	214		Hz	3
Sample Rate				2000	samples/s	4
Group Delay	LP-filter -3dB = 262Hz		6.5		ms	5
	LP-filter -3dB = 131Hz		8.0		ms	5
	LP-filter -3dB = 66Hz		11		ms	5
	LP-filter -3dB = 33Hz		17		ms	5
	LP-filter -3dB = 16Hz		29		ms	5
Bias switch on/off repeatability		-0.75	0	+0.75	mg	6
Bias error over temperature	$\Delta T \leq \pm 1^\circ\text{C}/\text{min}$		±2		mg rms	6
Bias Instability	Allan Variance @25°C		0.05		mg	6
Velocity Random Walk	Allan Variance @25°C		0.06		m/s/√hr	6
Vibration Rectification Coefficient			Ref.Figure 6-3		mg/g ² _{rms}	
Misalignment			1		mrad	7

Figure 4.4: IMU acceleration error-characteristics table. Figures from [8]

Parameter	Conditions	Min	Nom	Max	Unit	Note
GYRO						
Full Scale (FS)			±400		°/s	1
Resolution			24		bits	
			0.22		°/h	
Scale Factor Accuracy			±500		ppm	
Non-Linearity	±200°/s		25		ppm	2
	±400°/s		50		ppm	2
Bandwidth (-3dB)			262		Hz	3
Sample Rate				2000	samples/s	4
Group Delay	LP-filter -3dB = 262Hz		1.5		ms	5
	LP-filter -3dB = 131Hz		3.0		ms	5
	LP-filter -3dB = 66Hz		6.0		ms	5
	LP-filter -3dB = 33Hz		12		ms	5
	LP-filter -3dB = 16Hz		24		ms	5
Bias Range		-250	0	+250	°/h	
Bias error over temperature	Static temperatures		5		°/h	6
Bias error over temperature gradients	$\Delta T \leq \pm 1^\circ\text{C}/\text{min}$		10		°/h	7
Bias Instability	Allan Variance @25°C		0.5		°/h	
Angular Random Walk	Allan Variance @25°C		0.15		°/√hr	
Linear Acceleration Effect						
Bias	With g-compensation		1		°/h /g	8
	No g-compensation			15	°/h /g	8
Scale Factor	With g-compensation		30		ppm/g	8
	No g-compensation		400		ppm/g	8
Vibration Rectification Coefficient			Ref. Figure 6-1		°/h /g _{rms}	
Misalignment			1		mrad	9

Figure 4.5: IMU angular rate sensor error-characteristics table. Figure from [8]

As seen in table 4.1, the sample rate was set to 125 hz to minimize the contribution of the random noise error. Also Manta-2020 is not designed for high-speed applications, so a greater sample rater was not needed.

Error	Stanarddeviation
Misalignment	1 mrad
Scalar	300 ppm
Bias	0.05 mg

Table 4.2: Quality of the accelerometer on STIM 300

Error	Stanarddeviation
Misalignment	1 mrad
Scalar	500 ppm
Bias	$0.5^\circ/\sqrt{h}$

Table 4.3: Quality of the gyros on STIM 300

4.4 Doppler Velocity Log (DVL) - DVL1000

The second main interoceptive sensor used on Manta-2020 is a DVL1000 from Nortek. This sensor provides velocity measurements relative to the sea-bottom or water. This is done by sending a pulse with three or more acoustic beams in different directions. This DVL is designed to provide bottom tracking between 0.2 – 75m with a maximum of 300 meters operational depth [33]. This DVL can operate between -4 to 40 °C and is configurable to the RS-422 interface. It can operate between a DC-voltage input of 12 - 48 volts.

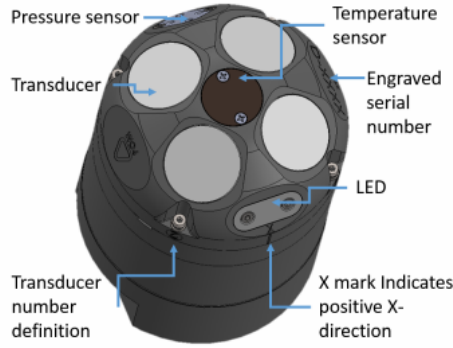


Figure 4.6: DVL sensor configuration. Figure from [9]

4.4.1 Axes and mechanical specifications

In order to get the correct lever-arm \mathbf{r}_{dvl}^b between the $\{imu\}$ and $\{dvl\}$ frame, the mechanical dimensions, axes and center of origin must be known. On Manta-2020, the DVL is located as shown in figure 4.8 and is rigidly attached to the AUV. The full sensor overview is seen in figure

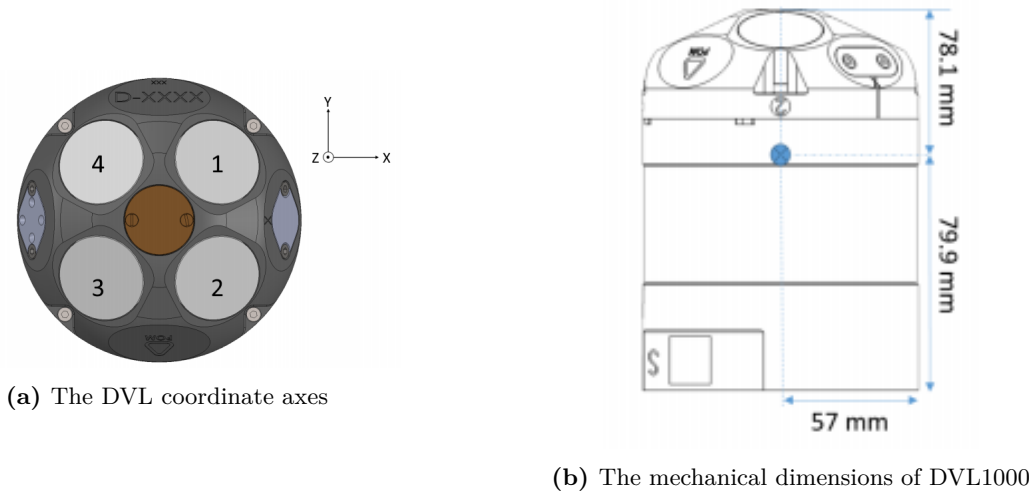


Figure 4.7: The DVL1000 coordinate system and INS configuration origin. Figures from [9]

4.4.2 DVL Measurement equation

On Manta-2020 the DVL is configured to use bottom tracking and not water tracking. This is because the environments Manta-2020 will operate in will have a static sea bottom that has no motion. The velocity measurement will then be written as follows [34], where $\{\text{bottom}\}$ is a fixed reference frame at the bottom of the sea.

$$\mathbf{v}_{dvl/bottom}^{dvl} = \mathbf{R}_b^{dvl}(\mathbf{q})\mathbf{R}_n^b(\mathbf{q})(\mathbf{v}_{b/n}^n - \mathbf{R}_b^n(\mathbf{q})S(\omega_{b/n}^b)\mathbf{r}_{dvl}^b) \quad (4.1)$$

Where $S(\omega_{b/n}^b)$ is the skew symmetric matrix and \mathbf{r}_{dvl}^b is the lever arm from the $\{dvl\}$ frame to the $\{b\}$ frame.

$\mathbf{R}_b^n(q)S(\omega_{b/n}^b)\mathbf{r}_{dvl}^b$ is the extra velocity contributed from the rotation of the body frame. This however is considered small if the lever arm \mathbf{r}_{dvl}^b is small, and assuming the DVL is rigidly attached to the vessel. This is the case for Manta-2020.

4.4.3 Pressure sensor measurement equation

As seen in the figure 4.6 the DVL-1000 is equipped with a pressure sensor. This is an absolute pressure sensor that is configured to output depth measurements according to the following equation.

$$z_{b/n}^{pressure} = \frac{(P_n^{pressure} + b_n^{pressure} + w_n^{pressure})P_0}{\rho_{water}g_n} \quad (4.2)$$

Where $P_n^{pressure}$ is the measured pressure, P_0 is the atmospheric pressure, ρ is the water density and g^n is the local gravity.

This depth measurement gives an absolute position measurement in the local world frame that is used on Manta-2020. With this it is possible to get an observable z - acceleration bias output from the bias estimation used in the ESKF.

The pressure sensor is located as seen in figure 4.6. The {pressure} frame has its origin in the {dvl} frame, and since it outputs absolute measurements relative to the world frame, there is no static rotation matrix from the pressure frame to the body frame. However, the world frame does not have its origin in the {dvl} frame. Therefore to get measurements in $z_{b/n}^n$, an vertical arm $r_{pressure}^n$ from the {pressure} frame to the {n} must be added to $z_{b/n}^{pressure}$. This is stated as follows.

$$z_{b/n}^n = r_{pressure}^n + z_{b/n}^{pressure} \quad (4.3)$$

This vertical arm was found to be by measuring with a precise ruler.

$$r_{pressure}^b = -0.211 \quad (4.4)$$

4.4.4 Sensor alignments

In figure 4.8 shows how the sensor- alignments for IMU and DVL are located on Manta-2020 and their respective frames. Here \mathbf{r}_{dvl}^b is the lever arm between {dvl} frame and {imu} frame.

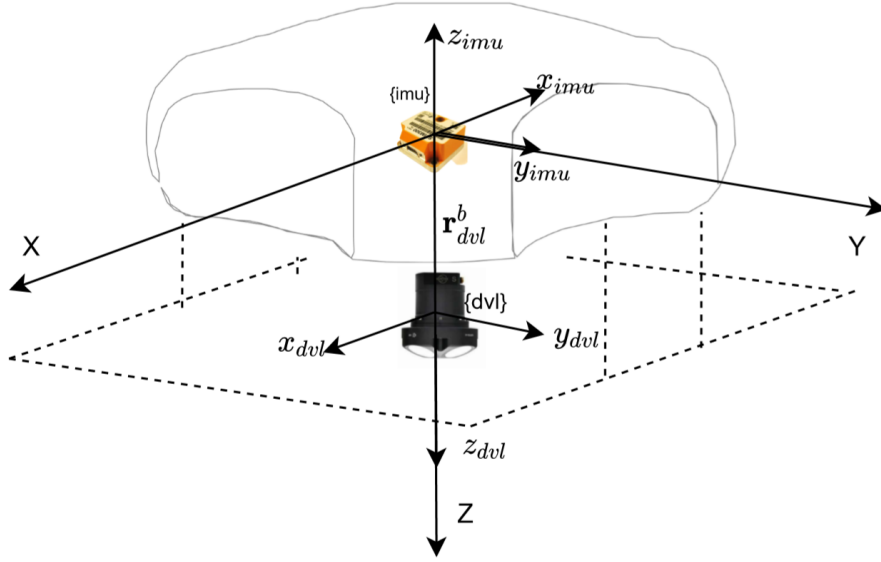


Figure 4.8: Manta-2020 interoceptive sensor alignment

Also seen in figure 4.8 the mounting reference frame $\{dvl\}$ has the same orientation as the body frame. Thus there are no rotation from the DVL to the body frame, and the the rotation matrix $\mathbf{R}_{dvl}^b(\mathbf{q})$ can be expressed as follows.

$$\mathbf{R}_{dvl}^b(\mathbf{q}) = \mathbf{I}_{3 \times 3} \quad (4.5)$$

As also seen in the figure 4.8, there is a distance from the origin of the IMU to the origin of the $\{dvl\}$ frame (See figure 4.7b) By measuring the distance with a precise ruler, the lever arm \mathbf{r}_{dvl}^b was found to be.

$$\mathbf{r}_{dvl}^b = \begin{bmatrix} -0.035 \\ -0.017 \\ -0.211 \end{bmatrix} \quad (4.6)$$

Also seen in figure 4.8, a rotation of π in pitch ϕ in euler angles will rotate the $\{acc\}$ and $\{gyro\}$ mounting frames to the body frame $\{b\}$. Thus the static euler angles for the $\{acc\}$ and $\{gyro\}$ frames are found to be as follows.

$$\Theta_{bacc} = \Theta_{bgyro} = \begin{bmatrix} 0 \\ \pi \\ 0 \end{bmatrix} \quad (4.7)$$

and by converting to quaternions by the formula 2.20, \mathbf{q}_{bacc} and \mathbf{q}_{bgyro} were found to be

$$\mathbf{q}_{bacc} = \mathbf{q}_{bgyro} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad (4.8)$$

Now \mathbf{q}_{bacc} and \mathbf{q}_{bgyro} can now be inserted in the rotation matrices $\mathbf{R}_{acc}^b(\mathbf{q})$ and $\mathbf{R}_{gyro}^b(\mathbf{q})$.

$$\mathbf{R}_{acc}^b(\mathbf{q}) = \mathbf{R}_{gyro}^b(\mathbf{q}) = \begin{bmatrix} -1.0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1.0 \end{bmatrix} \quad (4.9)$$

The pressure sensor had to be converted to NED and thus the vertical arm $r_{pressure}^b$ was set to the following.

$$r_{pressure}^b = -0.211 \quad (4.10)$$

4.5 Exteroceptive sensors

Also seen in figure 4.2, Manta is also equipped with three exteriorceptive sensors. The Low light HD camera is used as the primary source for close-up front object detection. This camera has a resolution of 2.24 MP and a supply voltage of 5V. This camera is ideally suited to be used in underwater environments, because of its low-light performance, color handling and on-board video compression [29]. The second exteroceptive sensor is the FLIR Blackfly S camera. This is used for bottom object detection and SLAM. These cameras are ideal for integration with computer vision, because of its features such as manual control over image capture and on-camera pre-processing [30]. Some of these features include precise control of acquisition, gain, black level, white balance, color transformation and transfer [31]. The newly installed Gemini 720i - imaging sonar is used as an long range "2D scanner" which is mainly used for geographical mapping of the underwater environment and SLAM.

4.6 Electronics system

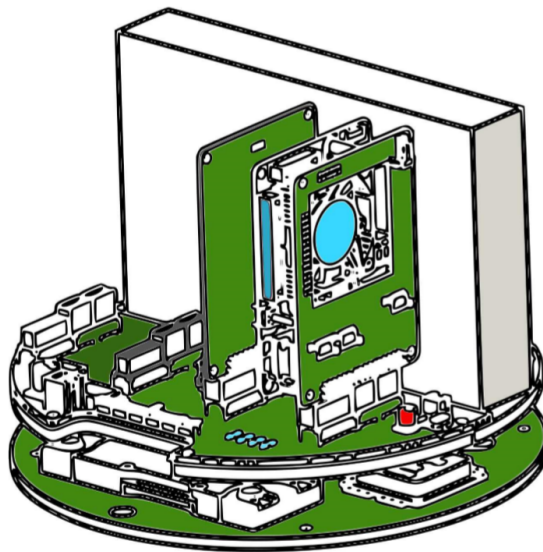


Figure 4.9: Drawing of Manta-2020 main electronic motherboard. Courtesy: ([5], Rakstad)

Figure 4.9 shows the electronic motherboard on Manta-2020. On the lower right side, is an Odroid XU4 microcontroller. This computer acts as Mantas On-Board main computer (OBC), which is responsible to

run the GNC system. The odroid has an Exynos 5422 application processor. This processor has a ARM quad-core Cortex A15 (2.0 Ghz) plus a ARM quad-core Cortex A7 (1.3 Ghz) CPUs. It is also packed with 2 GB LPDDR3 RAM at 933 Mhz [35]. The OBC is installed with Ubuntu minimal 16.04 to make less use for graphical user interface and other background programs that are not needed. These programs may higher the computational burden on the microcontroller and may potentially lower the computing power needed for the GNC system. The Odroid is also interfaced with a number of peripheral modules which are connected with a dedicated NanoPi Neo plus 2 (SBC) for standalone communication with the OBC[5]. The use of a dedicated SBC makes Manta-2020 modular and makes it possible to do changes without opening the main electronic enclosure [5]. On the upper right corner is the location of the switch. This switch has 6 Ethernet input ports which are connected to the sonar,DVL, front camera and carrier board of the Nvidia Jetson TX2.

Manta is also equipped with a NVIDIA Jetson TX2 4GB module. Since this module is packed with a 256-core NVIDIA pascal GPU[36], it is responsible for pre- and post processing of the camera feedback coming from the FLIR Blackfly S and Blue Robotics HD camera. Looking at figure 4.10 the NVIDIA Jetson TX2 is connected to its TX2 carrier board j120. This unit is packed with 1 USB 2.0, 1 USB 3.0 port and ethernet jack. The USB 3.0 is used for communication with the FLIR Blackfly S and the ethernet jack is connected to the switch.

Also seen in figure 4.10 the STIM300 is connected with a USB 2.0 directly to the Odroid XU4 USB 2.0 hubs. This standard was used because of the easy interface with the device driver, that was written in-house in C++11 (See Appendix A). The DVL is connected with a Ethernet connection to the switch. Its device driver is also written in-house in Python 2.7 (See Appendix A). Manta-2020 is also equipped with a kill switch, which turns off the battery voltage with the use of an magnetic sensor packed on the relè.

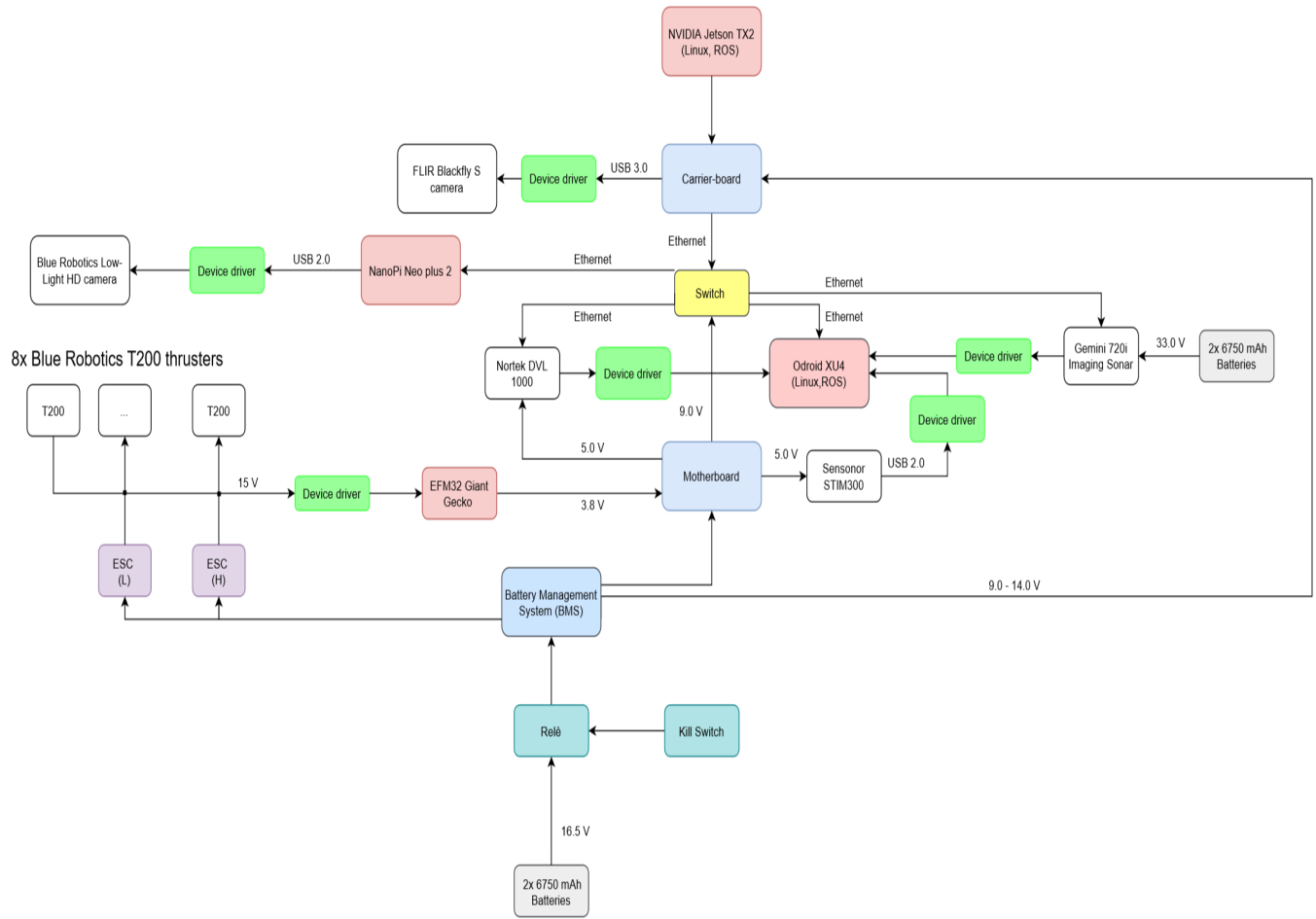


Figure 4.10: Schematics of the electronic overview of Manta-2020

4.7 ROS - Robot operating system

The robot operating system is a set of software frameworks and tools for developing robotic software [37]. It is open-source and provides services like hardware abstraction, low-level device control and message-passing between processes [37]. By having it open-source, the possibility of sharing code is open to everyone, such that high-quality code can be shared between contributors. ROS is based on modulation of software code with programming languages such as Matlab, Python, C++ and more. These "modules" are named as nodes in the ROS environment, consisting of software source-code of a particular application. ROS utilizes different type of communication between the these nodes with both synchronous and asynchronous communication. Synchronous communication is based upon "services" and continuously asynchronous communication over "topics". The nodes that continuously publishes information to another node, are called "publisher" and those who listens are called "subscribers". As with the ROS services is based on a server which provides a callback to the client request.

ROS also comes packed with a software package called "tf"[38] and "rviz". The "tf" package offers the user to keep track of multiple coordinate frames over time. Included in this package is also a homogeneous static transform publisher that can be used to find the transformation between different coordinate frames. This

is very useful in order to find the static rotations and lever arms between the sensor frames and body frame such as $\mathbf{R}_{bdvl}^b(\mathbf{q})$, $\mathbf{R}_{gyro}^b(\mathbf{q})$, $\mathbf{R}_{acc}^b(\mathbf{q})$, $\mathbf{R}_{pressure}^b(\mathbf{q})$ and \mathbf{r}_{dvl}^b [1]. The "rviz" package is a 3D-visualization tool that can display the sensor frames, body frame, world frame and shows their respective static rotations and lever arms. This tool is very useful when debugging and get a "feel" of how the localization behaves.

4.8 Software system overview

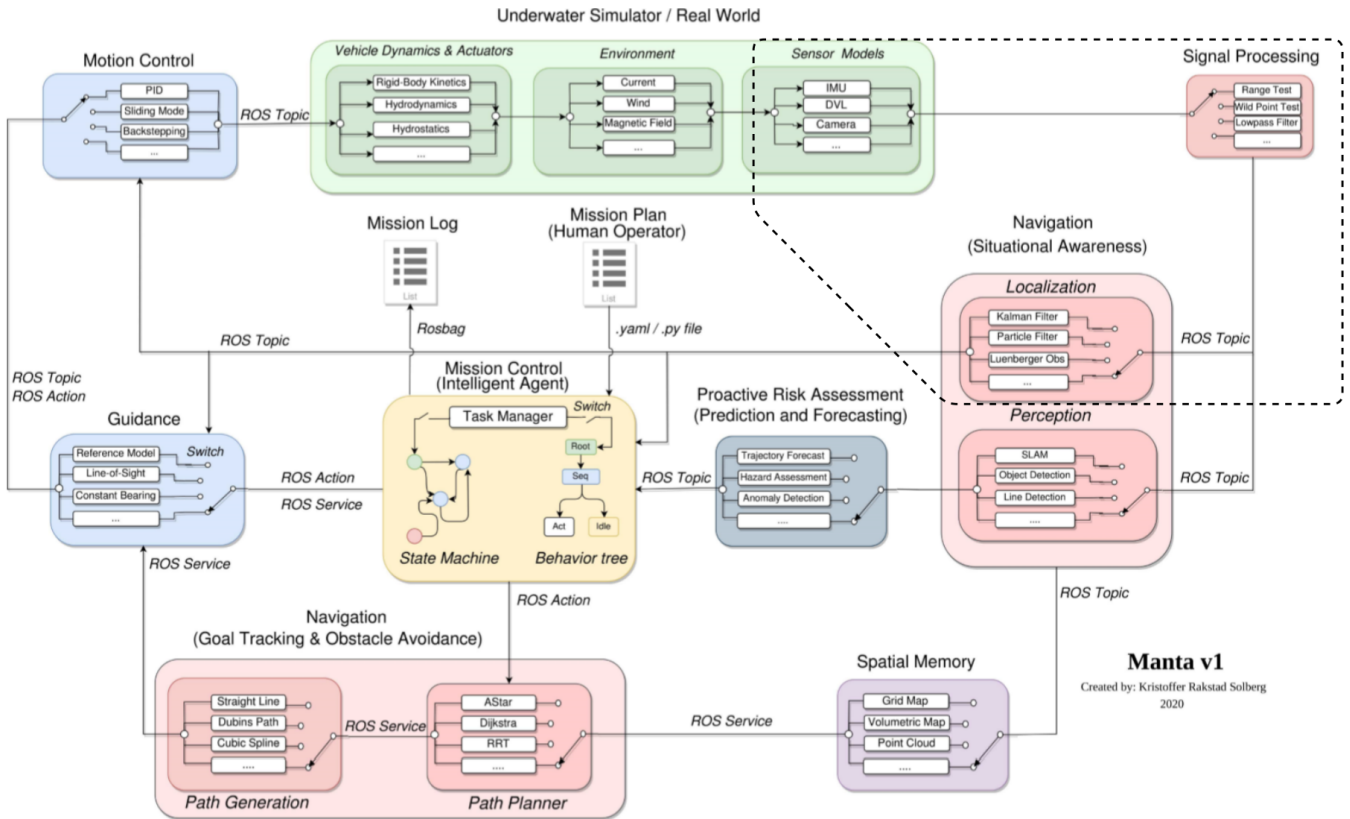


Figure 4.11: Software overview of Manta-2020. Courtesy([5], Rakstad)

Manta-2020 software architecture is based on a distributed network topology and a client/server network communication between the components [5]. This communication is provided by ROS kinetic level as described in section 1.4.3.1. The complete general schematic overview of the software of Manta-2020 is seen in figure 4.11. The flow starts at the mission plan and travels clockwise as shown by the arrows [5]. Here a human operator or a communicator sets apriori information. This is may be of the form of targeted way-points or a global or local map of the environment the AUV will operate in. This information is needed in order to get the reference information, such that the GNC can calculate its errors with the reference. These waypoints or maps are then added into a *.yaml* or *.py* file [5]. These will then be interpreted by a task manager such as a finite state machine or a behavior tree [5]. Once the mission controller is finished compiling, it will arm the thrusters and launch the exteriorceptive and interoceptive sensors modules by roslaunch commands. The raw_data coming from each sensor is being published through a ROS topic coming from their implemented device drivers as shown in the figure 4.10. The measurements are then going through a signal processing block, such as a range test, wild point test or a lowpass filter. The filtered measurements are then also published by a ros-topic to a localization node, such as a Kalman filter, nonlinear observer, particle filter

or a Luenberger observer. Exteriorceptive sensory information such as camera feedback is going through either SLAM, object detection or line detection. All of this "situational awareness" information is then going through a Proactive Risk Assessment and spatial memory which then go further to the path planner and path generator. In this report the main focus will be the blocks inside the dotted lines shown in the higher right corner in figure 4.11. Specifically the sensor models for the interoceptive sensors like the IMU, pressure gauge sensor and DVL, signal processing such as a wild point filter for the accelerometer and angular rate sensor for the IMU, a nonlinear state observer based on [3], Joan Solàs version of the Error-state Kalman-filter [2] and an extended Kalman-filter from Charles River Analytics, Inc [39]. For more information of the software overview, see [5].

State estimators

Measurements coming directly from the DVL, IMU and the pressure sensor, do not alone give a full 6 DOF state estimate of the AUV. Because of this state observers, together with a mathematical model and the measurements coming from the sensors, can be used to estimate the 6 DOF states from 2.9 and 2.6. In this section some theory of the Bayes filter will first be presented, then the discrete Kalman filter, extended Kalman filter, the theory of Joan Solàs error-state Kalman filter based on [2] and finally theory about the nonlinear observer based on [3].

5.1 The Bayes filter

In order to understand the Kalman filter, the Bayes filter must first be understood. First of all, the Bayes filter is based upon a Markov process. (page 47 in [21]), where the filtering problem is described of two models. The first is the kinematic prior or Markov model. This is usually denoted as $p(\mathbf{x}_k|\mathbf{x}_{k-1})$. The second model describes the measurement model or likelihood. This model is specified as $p(\mathbf{z}_k|\mathbf{x}_k)$. Here the x vector describes the states, while the z vector describes the incoming measurements. This can be the output of different sensors.

In Bayesian filtering the problem is to estimate the posterior distributions $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ and $p(\mathbf{x}_k|\mathbf{z}_{1:k})$. In order to calculate these, the total probability theorem and Bayes' theorem are used respectively.

$$p(\mathbf{x}|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})d\mathbf{x}_{k-1} \quad (5.1)$$

and

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} \quad (5.2)$$

One of the key assumptions regarding the Bayes filter, is that the Markov property must hold. The Markov property means that \mathbf{x}_k is independent of $\mathbf{x}_0, \dots, \mathbf{x}_{k-1}$ when \mathbf{x}_k is given [21].

In order to find closed-form solutions of 5.1 and 5.2, the kinematic prior and measurement model must be Gaussian and linear. By using a Gaussian distribution the linearity, the fundamental product identity and independence rules can be used, and thus the solutions of 5.1 and 5.2 can be a closed-form solution. The product identity states that a product of two Gaussian's will produce another Gaussian distribution. The

linearity rule states that if a Gaussian random variable goes through a linear transform the new random variable is also Gaussian.

5.2 The discrete Kalman filter

The Kalman filter is in simple words an stochastic, optimal and recursive data processing algorithm based upon a Bayes filter where the underlying distributions are assumed to be Gaussian[21]. The algorithm uses a series of measurements, normally coming from the output of sensors, which contain statistical noise, compare these measurements with a stochastic physical model, and produces optimal estimates of the true states. These estimates tend to be more accurate than just the measurement readings alone.

The kinematic prior and the measurement model of the discrete Kalman filter is written as

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{v}_k \quad (5.3)$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k \quad (5.4)$$

Here the \mathbf{F} matrix denotes the transition matrix, which describes how the states will be predicted for each cycle k . The \mathbf{v}_k describes the kinematic prior noise. The \mathbf{H} matrix is defined as the measurement matrix, and describes for example which of the states that are measured from a sensor. All output from these sensors will contain noise in their measurements. This is captured in the \mathbf{w}_k vector, and is known as the measurement noise. Both the measurement noise and the kinematic prior noise are assumed to be mutually independent and have a Gaussian probability density function with zero mean[21]. The Gaussian distribution with zero mean is to ensure that there exist a closed-form solution, and that the estimate of the true states are unbiased. Thus these noise vectors can be written as follows

$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \quad (5.5)$$

$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \quad (5.6)$$

Here \mathbf{Q} is the process noise matrix representing the covariance of the kinematic prior, and accounts for the uncertainty in the process model. This matrix is constant and does not get updated by the filter. The tuning of this matrix is thus important in order to get accurate posterior estimates. The \mathbf{R} matrix is known as the measurement noise covariance and account for the uncertainty in the measurement model, and defines the confidence in each of the measurements \mathbf{z}_k .

In order to get a probability density function of the kinematic prior and the measurement model, the expected value and the variance can be found as follows. The steps are the same the measurement model.

$$\mathbb{E}[\mathbf{x}_k] = \mathbb{E}[\mathbf{F}\mathbf{x}_{k-1} + \mathbf{v}_k] \quad (5.7)$$

$$\mathbb{E}[\mathbf{x}_k] = \mathbf{F}\mathbf{x}_{k-1} + \mathbb{E}[\mathbf{v}_k] \quad (5.8)$$

$$\mathbb{E}[\mathbf{x}_k] = \mathbf{F}\mathbf{x}_{k-1} \quad (5.9)$$

$$\text{Var}[\mathbf{x}_k] = \text{Var}(\mathbf{F}\mathbf{x}_{k-1} + \mathbf{v}_k) \quad (5.10)$$

$$\text{Var}[\mathbf{x}_k] = \mathbf{0} + \text{Var}[\mathbf{v}_k] \quad (5.11)$$

$$\text{Var}[\mathbf{x}_k] = \mathbf{Q} \quad (5.12)$$

Thus the probability density function of the kinematic prior and the measurement model, can be written as

$$p(\mathbf{x}|\mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k; \mathbf{F}\mathbf{x}_{k-1}, \mathbf{Q}) \quad (5.13)$$

$$p(\mathbf{z}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k; \mathbf{H}\mathbf{x}_k, \mathbf{R}) \quad (5.14)$$

The algorithm step by step

The full Kalman filter algorithm is seen in figure 5.1. The probability density figures take only the first cycle into account. This is from $k = 0$ to $k = 1$. This is done in order to get an intuitive understanding of how the filter predicts and updates and how the comparison between the prediction and measurement is done. Also the figure clearly shows that all probability density functions are assumed to be Gaussian. One thing to note in the figure is that the outputs are in univariate Gaussians. This is done for illustration purposes only. Normally higher dimensions are used and thus the multivariate Gaussian will be the underlying probability density function.

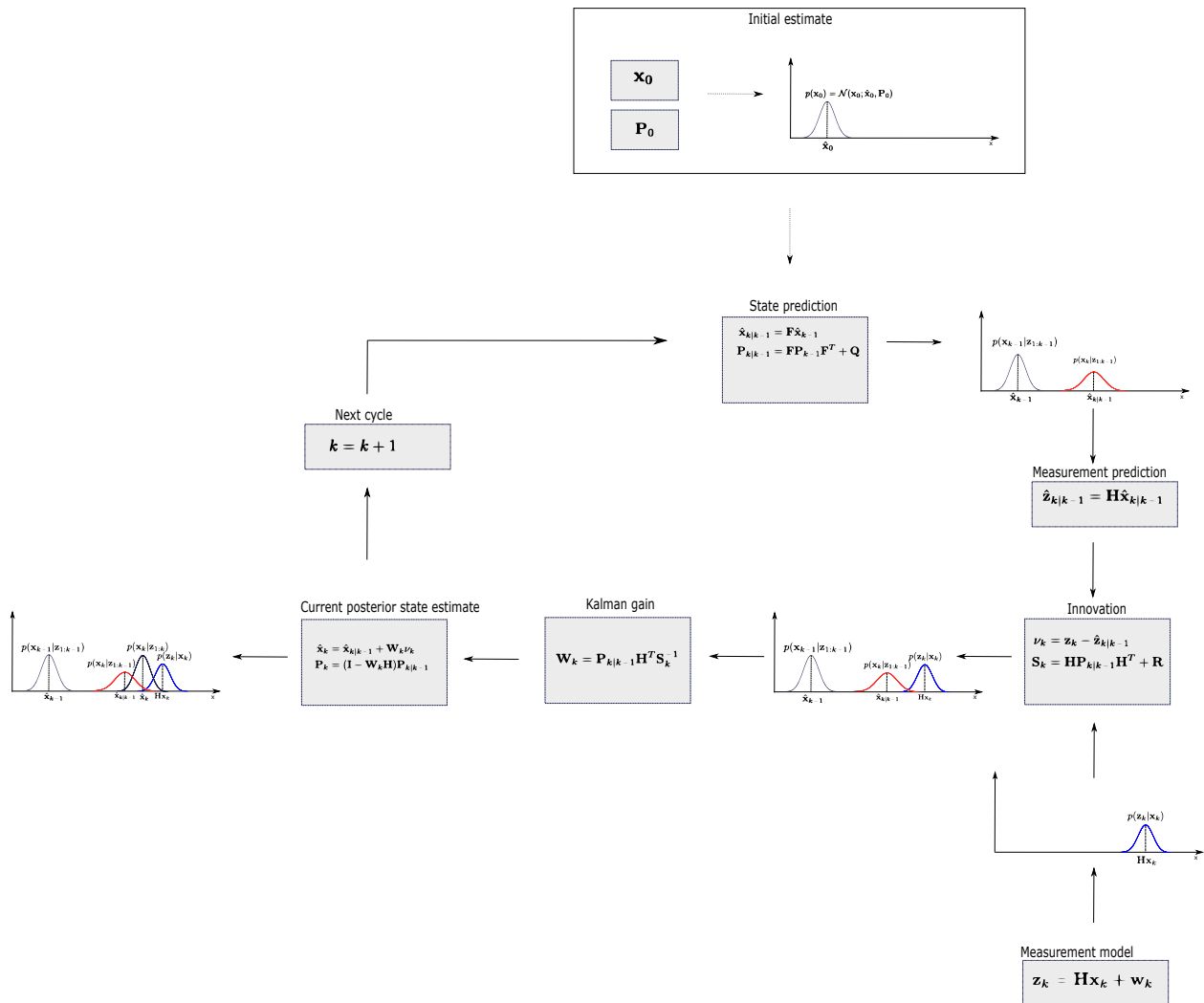


Figure 5.1: The discrete Kalman filter algorithm

The initial estimate

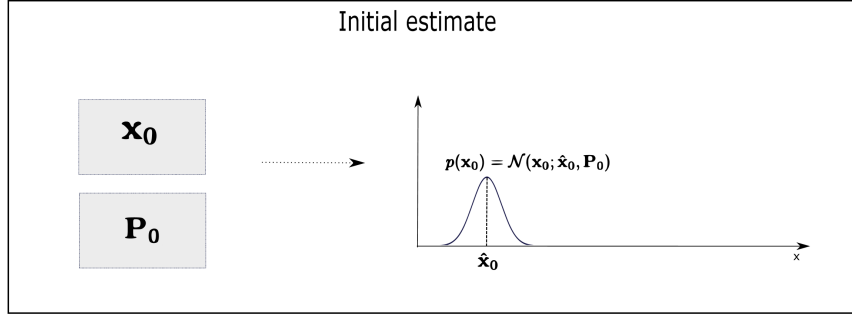


Figure 5.2: The initial step of the kalman filter

The first step in the discrete Kalman filter algorithm is the initial estimate. Looking at figure 5.2, there are parameters that have to be initialized, the initial state estimate \mathbf{x}_0 and the initial error covariance \mathbf{P}_0 . Together these form a Gaussian distribution with a mean of $\hat{\mathbf{x}}_0$ and a variance of \mathbf{P}_0 . In an compact form, this can be written as follows:

$$p(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0; \hat{\mathbf{x}}_0, \mathbf{P}_0) \quad (5.15)$$

These parameters are critical in order for the Kalman filter to converge quickly to an optimal solution that is more accurate than the measurement alone, the initial guess of the states \mathbf{x}_0 should have values close to the true states \mathbf{x}_t . This may be the initial state measurements \mathbf{z}_0 . If no sensors are to provide a state measurement, for example position, then these states may be initialized to zero. For example if an accelerometer and a GPS where to be used, the position estimate would be the last GPS reading, the velocity as the last velocity GPS reading or the last accelerometer reading. The \mathbf{P}_0 could be initialized with the measurement noise covariance matrix \mathbf{R} for the given sensors that provide the states. For the example at hand, \mathbf{P}_0 can be stated as follows:

$$\mathbf{P}_0 = blkdiag(\mathbf{R}_{gps}, \mathbf{R}_{gps} \text{ or } \mathbf{R}_{acc}, \mathbf{0}, \mathbf{0}, \mathbf{R}_{acc}) \quad (5.16)$$

The prediction step

Since the posterior probability density function of the previous time-step $p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1})$ and the kinematic prior model with probability density function given as in 5.13, are known, it is possible to calculate the prediction step as given in 5.1. This is possible by the use of the fundamental product identity of Gaussian probability density distributions and with the use of the total probability theorem. This can be seen mathematically as [21]

$$p(\mathbf{x}|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})d\mathbf{x}_{k-1} \quad (5.17)$$

$$= \int \mathcal{N}(\mathbf{x}_k; \mathbf{F}\mathbf{x}_{k-1}, \mathbf{Q})\mathcal{N}(\mathbf{x}_{k-1}; \hat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1})d\mathbf{x}_{k-1} \quad (5.18)$$

$$= \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}) \quad (5.19)$$

Thus this step combines the previous posterior estimate with the kinematic prior and makes a new Gaussian distribution based on the product rule for Gaussian distributions as seen in figure 5.3.

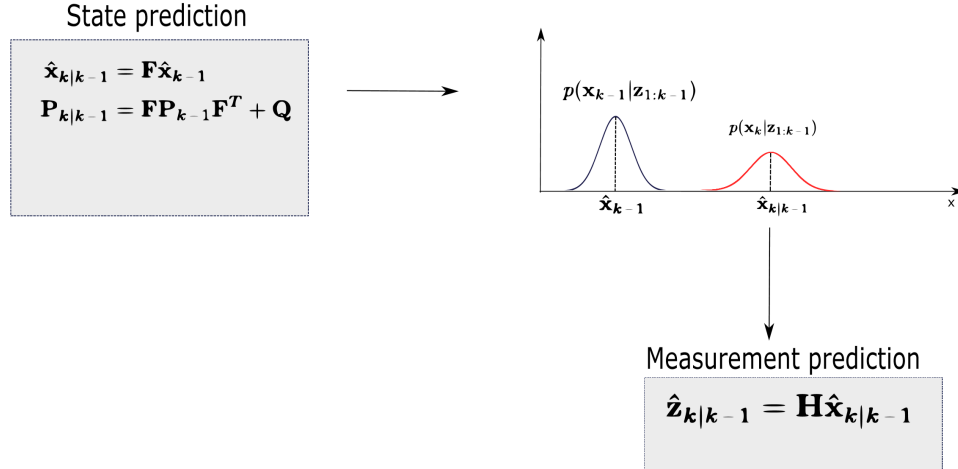


Figure 5.3: The prediction step of the Kalman filter

After the state prediction step, the next step is the measurement prediction. This can be stated as follows

$$\hat{\mathbf{z}}_{k|k-1} = \mathbf{H}\hat{\mathbf{x}}_{k|k-1} \quad (5.20)$$

Here the \mathbf{H} matrix is known. This prediction step is important in order to compare the correct states with the incoming measurements \mathbf{z} , which leads to the innovation.

The innovation

After the prediction step, the Kalman algorithm takes in measurements coming from the sensors in the measurement model $\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k$. Looking at figure 5.4 this forms a Gaussian distribution with a mean the measured states $\mathbf{H}\mathbf{x}_k$ and a covariance of \mathbf{R} coming from the noise term in \mathbf{w}_k . \mathbf{z}_k then becomes subtracted from the predicted estimate $\hat{\mathbf{z}}_{k|k-1}$ together with its covariance calculated, known as the innovation and innovation covariance respectively. This is a combination of the predicted covariance $\mathbf{P}_{k|k-1}$ and the measurement covariance \mathbf{R} .

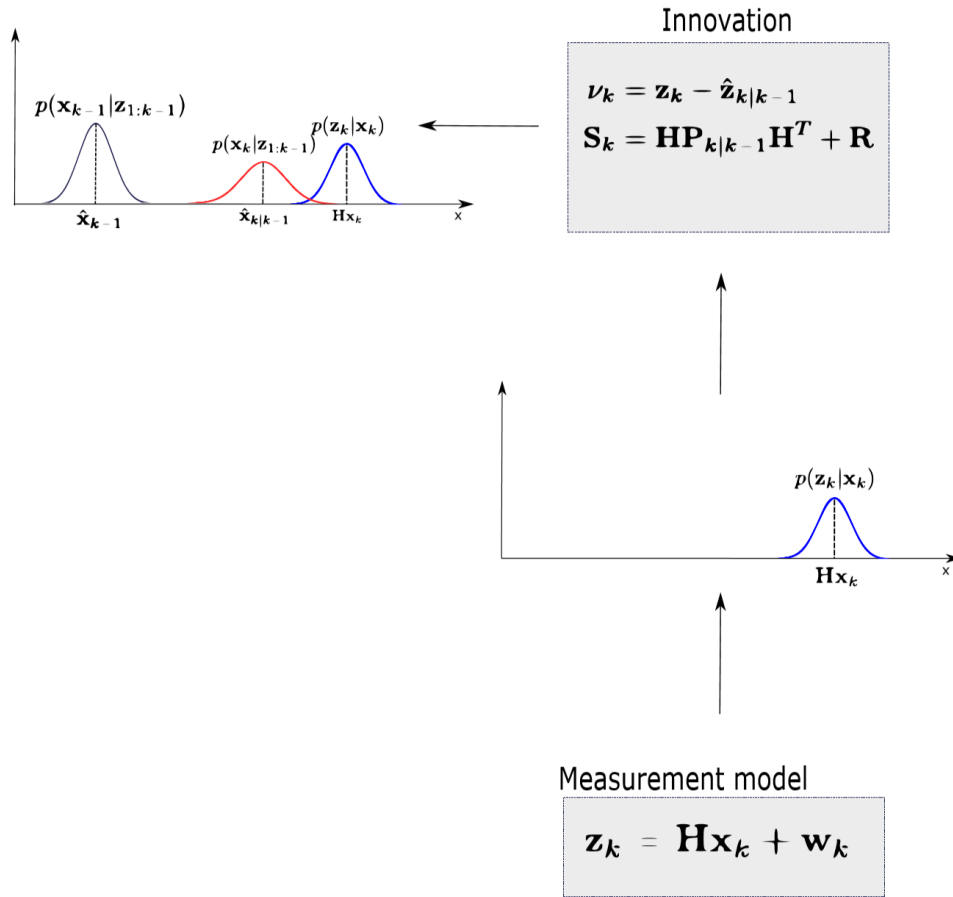


Figure 5.4: The innovation step of the Kalman filter

The Kalman gain

With the innovation calculated, it is now possible to calculate the Kalman gain. This gain is found by the following equation.

$$\mathbf{W}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T\mathbf{S}_k^{-1} \quad (5.21)$$

This gain measure how much the algorithm should trust the prediction compared to the measurements.

The current posterior state estimate

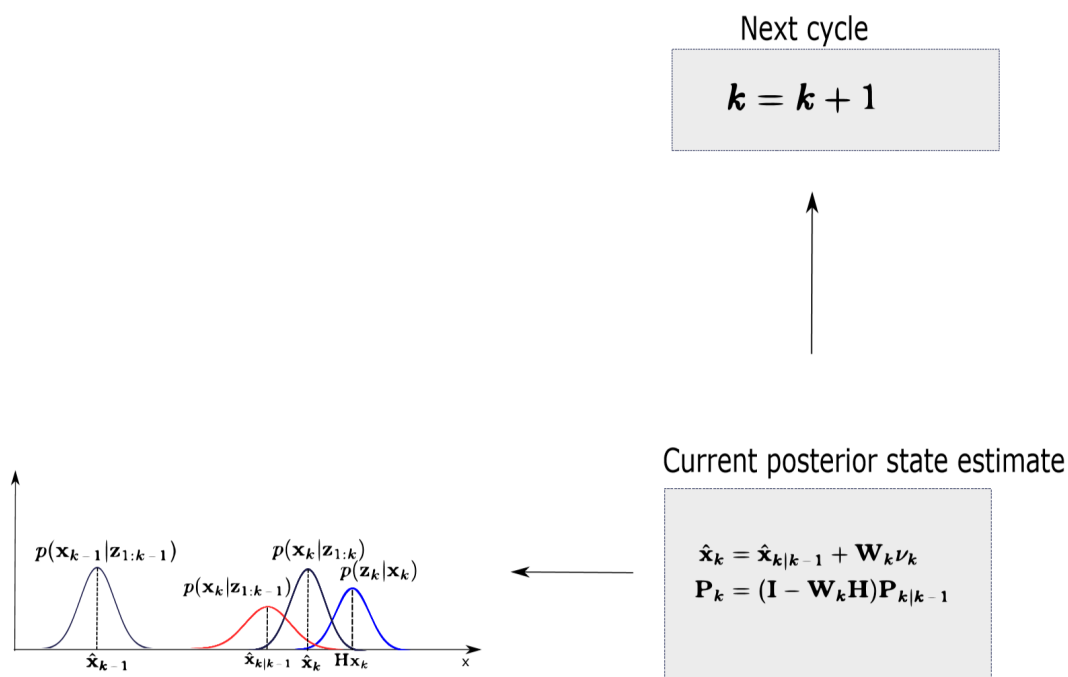


Figure 5.5: The current posterior state estimate step of the Kalman filter

The last step in the discrete Kalman filter algorithm is the current posterior state estimate. As seen in figure 5.5, this is where the Kalman filter does its update based on the prediction, Kalman gain and innovation. Looking at the resulting Gaussian distributions, the posterior estimate is intuitively an optimal Gaussian distribution based on the product rule between the prediction and measurement Gaussians. Mathematically this is calculated as follows.

$$p(\mathbf{x} | \mathbf{z}_{1:k-1}) \propto p(\mathbf{z}_k | \mathbf{x}_k) p(\mathbf{x}_k | \mathbf{z}_{1:k-1}) \quad (5.22)$$

$$= \mathcal{N}(\mathbf{z}_k; \mathbf{H}\mathbf{x}_k, \mathbf{R}) \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}) \quad (5.23)$$

$$\propto \mathcal{N}(\mathbf{x}_k; \hat{\mathbf{x}}_k, \mathbf{P}_k) \quad (5.24)$$

\mathbf{P}_k is known as the posterior error covariance. This is the covariance of the posterior update probability density function of 5.2. In intuitive language, the inverse of this matrix shows how "confident" the Kalman filter is in its posterior state estimate [21].

5.3 Tuning of the process noise covariance \mathbf{Q} and filter consistency

Tuning of the process noise covariance is important for the filter to get accurate estimates of the states close to the true states. The process noise covariance is, as already written above, used in the kinematic prior as a noise term. Since the Kalman gain is dependent upon the predicted error covariance, the process noise covariance will say something about how much the prediction is trusted.

There are different methods of tuning the process noise \mathbf{Q} . One method is based upon physical considerations [21]. This takes into account how the true system will be maneuvered. If the maneuvers accurately represents the state prediction, a low process noise covariance will then be set. In the opposite case, where the kinematic prior model does not accurately predict the given maneuvers, a high process noise covariance should be set [21].

A second method is based on filter consistency. With the use of the normalized estimation error squared (NEES), stated as follows: [21]

$$\epsilon_k = (\hat{\mathbf{x}}_k - \mathbf{x}_k)^T \mathbf{P}_k^{-1} (\hat{\mathbf{x}}_k - \mathbf{x}_k) \quad (5.25)$$

it is possible to find a reasonable process noise covariance. This will be the case if the kinematic prior model gives reasonable predictions of the true maneuvering, the measurement noise covariance \mathbf{R} is correctly given with a Gaussian probability density function, and that the state errors $\hat{\mathbf{x}}_k - \mathbf{x}_k$ have magnitude commensurate with the posterior error covariance \mathbf{P}_k . If these requirements are met, it can be shown that the pdf of ϵ_k is χ^2 distributed. By combining all realizations of k together, a χ^2 distribution form, which has a stochastic variable as the average NEES (ANEES). By constructing a reasonable confidence interval of the mean between α and $1 - \alpha$, it is possible to find out if the filter is overconfident or underconfident. If the value of ANEES is below the confidence interval lower value, means that the \mathbf{Q} matrix is set too large, and the filter is underconfident. While if the value of ANEES is above the confidence upper value, the \mathbf{Q} matrix is set too small, and the filter is overconfident.

Thus the most reasonable method to tune the process noise covariance \mathbf{Q} would then be to combine the first and second method described above. Then it is possible to have both physical considerations and mathematical inspection to tune the process noise.

A more real life scenario of tuning the \mathbf{Q} matrix is with the use of the NIS test, stated below as

$$\epsilon_k^v = \nu_k^T \mathbf{S}_k^{-1} \nu_k \quad (5.26)$$

where ν_k^T is the innovation and \mathbf{S}_k^{-1} is the inverse of the innovation covariance.

The results of both of these tests should mimic a zero-mean univariate Gaussian distribution. If so, NIS or NEES are modeled as white noise, which indicates that the filter is assumed to have an optimal estimation. When doing simulations or in real world scenarios where ground truth are given, both the NIS and NEES can be calculated and is a critical step in order to tune the filter to for instance not diverge under long periods of estimation.

5.4 The extended Kalman filter

The discrete Kalman filter has now been thoroughly studied and has given a fundamental building block to the theory of the extended Kalman filter. The algorithm has many of the same steps, and these steps will not be studied in this section. This section, however, will account for the added steps and differences in the algorithm.

5.4.1 Linearization scheme

The extended Kalman filter is in general based upon a nonlinear kinematic prior model and measurement model. These are stated as follows [21].

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{v}_k \quad (5.27)$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{w}_k \quad (5.28)$$

where \mathbf{v}_k and \mathbf{w}_k corresponds to 5.5 and 5.6 respectively. The terms \mathbf{f} and \mathbf{h} denotes the nonlinear functions. In order for the extended Kalman filter to have close-formed solutions of 5.1 and 5.2, \mathbf{f} and \mathbf{h} must be linearized.

The linearization method is based upon a first order Taylor expansion approximation of $\mathbf{f}(\mathbf{x}_{k-1})$ and $\mathbf{h}(\mathbf{x}_k)$. This can be stated as follows[21].

$$\mathbf{f}(\mathbf{x}_{k-1}) \approx \mathbf{f}(\hat{\mathbf{x}}_{k-1}) + \mathbf{F}(\hat{\mathbf{x}}_{k-1})\Delta\mathbf{x}_{k-1} \quad (5.29)$$

$$\mathbf{h}(\mathbf{x}_k) \approx \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) + \mathbf{H}(\hat{\mathbf{x}}_{k|k-1})\Delta\mathbf{x}_k \quad (5.30)$$

Here $\Delta\mathbf{x}$ corresponds to the following [21].

$$\Delta\mathbf{x}_{k-1} = \mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1} \quad (5.31)$$

$$\Delta\mathbf{x}_k = \mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1} \quad (5.32)$$

5.4.2 Algorithm differences compared to the Kalman filter

Since there is now a nonlinear kinematic prior model and measurement model, the algorithm used for the extended Kalman filter has some differences with the Kalman filter described in the previous section. The differences are seen in figure 5.6.

The Kalman filter	The extended Kalman filter
<p data-bbox="532 289 690 317">State prediction</p> <div data-bbox="495 331 751 472" style="border: 1px solid black; padding: 5px;"> $\hat{\mathbf{x}}_{k k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1}$ $\mathbf{P}_{k k-1} = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}$ </div>	<p data-bbox="906 289 1063 317">State prediction</p> <div data-bbox="868 331 1125 472" style="border: 1px solid black; padding: 5px;"> $\hat{\mathbf{x}}_{k k-1} = \mathbf{f}(\hat{\mathbf{x}}_{k-1})$ $\mathbf{F} = \left. \frac{\partial}{\partial \mathbf{x}_k} \mathbf{f}(\mathbf{x}_{k-1}) \right _{\mathbf{x}_{k-1} = \hat{\mathbf{x}}_{k-1}}$ $\mathbf{P}_{k k-1} = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}$ </div>
<p data-bbox="495 552 717 579">Measurement prediction</p> <div data-bbox="495 583 751 657" style="border: 1px solid black; padding: 5px;"> $\hat{\mathbf{z}}_{k k-1} = \mathbf{H}\hat{\mathbf{x}}_{k k-1}$ </div>	<p data-bbox="873 510 1096 537">Measurement prediction</p> <div data-bbox="868 541 1125 699" style="border: 1px solid black; padding: 5px;"> $\hat{\mathbf{z}}_{k k-1} = \mathbf{h}(\hat{\mathbf{x}}_{k k-1})$ $\mathbf{H} = \left. \frac{\partial}{\partial \mathbf{x}_k} \mathbf{h}(\mathbf{x}_k) \right _{\mathbf{x}_k = \hat{\mathbf{x}}_{k k-1}}$ </div>
<p data-bbox="511 762 701 789">Measurement model</p> <div data-bbox="495 793 751 867" style="border: 1px solid black; padding: 5px;"> $\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k$ </div>	<p data-bbox="889 762 1079 789">Measurement model</p> <div data-bbox="868 793 1125 867" style="border: 1px solid black; padding: 5px;"> $\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{w}_k$ </div>

Figure 5.6: The differences in the algorithm between the Kalman filter and the extended Kalman filter

In the state prediction block of the extended Kalman filter, the linearization of \mathbf{f} is done. Since the true states are not known, the linearization operating point of \mathbf{F} is the most recent posterior state estimate. For the linearization of \mathbf{h} , the operating point is the most recent state prediction.

5.5 Manta-2020 Robot_localization - The open source EKF

Having discussed the theoretical aspect of the Bayes filter, the Kalman filter and the extended Kalman filter, the problem is then to put this into place for Manta-2020. The use of just a discrete Kalman filter to estimate, is insufficient. This is because the dynamics of Manta-2020 is nonlinear. This is because rotation are required between the different frames described in section 1.1. With this an extended Kalman filter had to be used.

The extended Kalman filter on Manta-2020 was implemented using the open-source implementation robot_localization. This is a ROS package that contains a collection of nonlinear state estimation nodes. In this package the EKF is named ekf_localization_node. This is a C++ written ROS node which subscribes on IMU and aiding sensors measurements. With this the drivers for the IMU and DVL had to be made as ROS publisher nodes. The IMU measurements were published in a standard ROS message of type sensor_IMU, while the DVL and pressure measurements were published with an nav_/Odometry message. In the driver the IMU measurements was set to publish with a frequency of 125 Hz. This is the standard frequency of the STIM 300. The DVL and pressure sensor measurements were set to publish with 8 Hz, which was the maximum frequency that could be used. This was done to get as many measurements of the DVL and pressure sensor as possible, such that the EKF minimized dead reckoning.

5.5.1 The Kinematic prior model

The EKF robot localization nonlinear kinematic prior model is based upon a standard 3D kinematic model derived from Newtonian mechanics [40] with a 15 dimensional state vector \mathbf{x}_k . The state vector can be written as follows [40].

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{p}_{(b/w)_k}^w \\ \Theta_{(wb)_k} \\ \mathbf{v}_{(b/w)_k}^b \\ \omega_{(b/w)_k}^b \\ \mathbf{a}_{(b/w)_k}^b \end{bmatrix} \quad (5.33)$$

Here the coordinate system $\{w\}$ corresponds to the world frame of robot_localization, which is discussed in section 5.5.2 And the nonlinear kinematic transition model $\mathbf{f}(\mathbf{x}_{k-1})$ as follow.

$$\mathbf{f}(\mathbf{x}_{k-1}) = \begin{bmatrix} \mathbf{p}_{(b/w)_{k-1}}^w + \mathbf{R}_b^w(\Theta_{(wb)_{k-1}})\mathbf{v}_{(b/w)_{k-1}}^b \Delta t + \frac{1}{2}\mathbf{R}_b^w(\Theta_{(wb)_{k-1}})\mathbf{a}_{(b/w)_{k-1}}^b \Delta t^2 \\ \Theta_{(wb)_{k-1}} + \mathbf{T}_\Theta(\Theta_{(wb)_{k-1}})\omega_{(b/w)_{k-1}}^b \Delta t \\ \mathbf{v}_{(b/w)_{k-1}}^b + \mathbf{a}_{(b/w)_{k-1}}^b \Delta t \\ \omega_{(b/w)_{k-1}}^b \\ \mathbf{a}_{(b/w)_{k-1}}^b + \mathbf{R}_w^b(\Theta_{(wb)_{k-1}})\mathbf{g}^w \end{bmatrix} \quad (5.34)$$

Here $\mathbf{T}_\Theta(\Theta_{(wb)_{k-1}})\omega_{(b/w)_{k-1}}^b$ is given in 2.8.

The sampling frequency $f = \frac{1}{\Delta t}$ was set to match the frequencies of the NLO and ESKF.

5.5.2 Frames

The frames for Robot localization are defined as *odom,map* and *base_link*. The *odom* frame is an local world fixed coordinate system that that is defined according to the initial state estimate in the EKF. The *base_link* is the coordinate frame that is affixed to the robot. The *map* frame is a frame used for fusion with GNSS systems. In the implementation and design of the ROS Param Yaml file, the initial attitude state estimates were set to the default values, which were 0,0,0 for the given test scenarios. *Odom* and *base_link*

were then both defined as ENU (East-North-Up) coordinate systems. These coordinate systems are better seen in figure 5.7

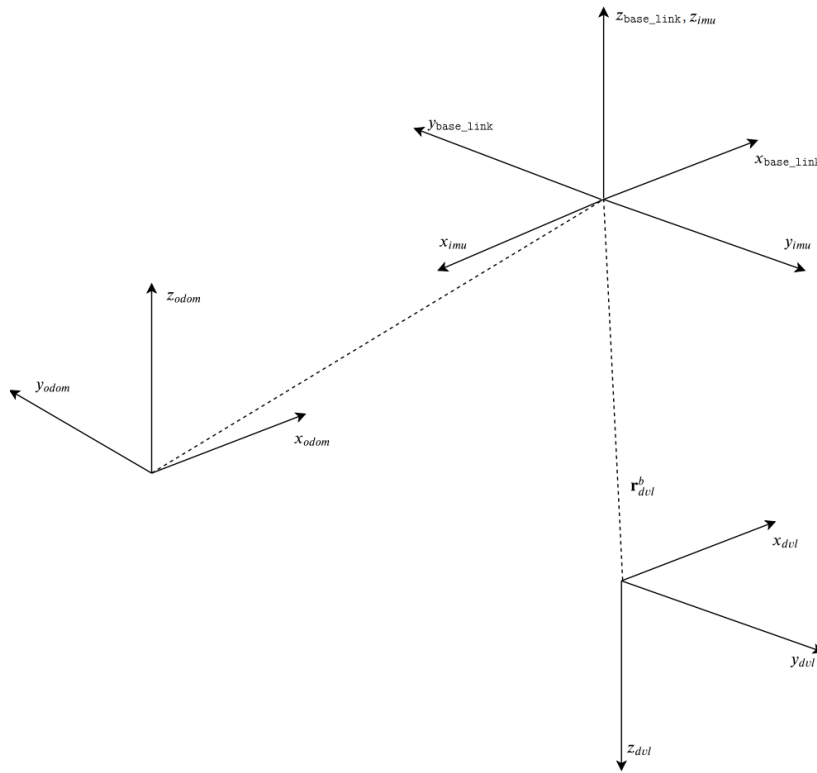


Figure 5.7: EKF ROS - Robot localization coordinate frames

The EKF node assumes that the IMU measurements are defined according to ENU (East-North-Up). The students at Vortex-NTNU then found that the best solution is to have the body frame as a ENU coordinate system. Therefore the static transformations are not similar to the ESKF and EKF.

5.5.3 Static transformations

The ROS framework uses `Tf_static_transform_publisher` as static rotations and leverarm between different frames. The static transform publisher is defined as follows [41]

- `static_transform_publisher x y z yaw pitch roll frame_id child_frame_id period_in __ms`

Where x,y,z is the leverarm and $yaw,pitch,roll$ are the euler angles that must be specified in order to make the static rotation matrix between `frame_id` `child_` and `child_frame_id`.

For the DVL the static transform is found to be $roll = \pi$ and $x, y, z = -0.035, -0.017, -0.211$ and for the IMU the static transform was set to $yaw = \pi$ yielding the following `static_transform_publishers`

- `static_transform_publisher -0.035 -0.017 -0.211 0 0 π base_link dvl 125`
- `static_transform_publisher 0 0 0 π 0 0 base_link imu 8`

With the pressure sensor the output of the driver was set to output positive values when the AUV was moving towards the water surface and negative when moving towards the bottom surface. This was done in order

to have the $z_{b/odom}^{pressure}$ axis in the same orientation as the z_{odom} . With this the vertical leverarm from the $odom$ to the pressure frame had to be set. Following equation 4.3, an verticalarm with $z = r_{pressure}^{odom} = 0.211$ gives the static rotation. The static transform for the pressure sensor then became as follows

- static_transform_publisher 0 0 0.211 0 0 0 odom pressure 125

5.6 Limitations of the extended Kalman filter

A first order Taylor expansion of a nonlinear function will make a linear approximation. This linear approximation will give raise to a linearization error. This is visualized in figure 5.8. This error depends on how nonlinear the function is. If the nonlinear function is highly nonlinear, the linearization error will be large. If the nonlinear function is mostly linear and varies slowly, the error will be low. Also, as seen in the figure, the further away from the operating point \mathbf{a} the linear approximation is, the larger the linearization error becomes.

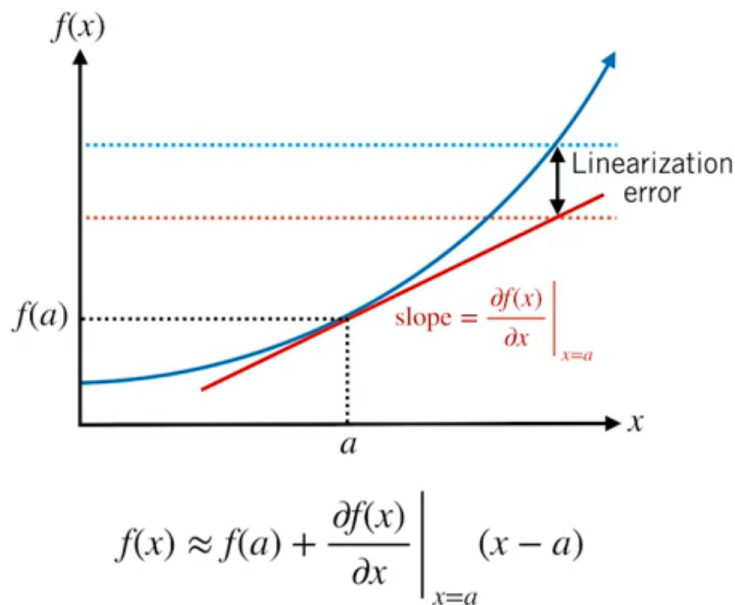


Figure 5.8: Linearization error that occurs by linearizing a nonlinear function. Figure from [10]

For the extended Kalman filter the linearization error will be a problem, if the kinematic prior and the measurement model are highly nonlinear. The true behavior of the system will then not be captured very well. Also if the sensors have slow sampling time relative to how fast the system is evolving, the linearization error will raise. This has some important consequences for the estimation of the EFK. First, the posterior state estimate will have a large deviation from the true states. Second, the posterior error covariance will not capture the true uncertainty. In the worst scenario the linearization error may cause the extended Kalman filter to become overconfident in its posterior estimate, even when the estimate is far off from the true states and uncertainty in the states. This may cause the filter to diverge.

Another limitation of the extended Kalman filter is found if the kinematic or measurement model are not first order differentiable. Also computing the Jacobian matrices may be computationally expensive for small embedded systems.

5.7 The error-state Kalman filter

Having discussed the limitations of the extended-KF, an alternative is the error-state Kalman filter (ESKF), which has the important advantage of not having singularities both in state representation and in the covariance representation. This filter is generally more suitable for inertial navigation systems (INS) than the discrete KF and the EKF. The filter integrates the high frequency measurements from a gyro and an accelerometer and estimates a 6 DOF state vector. By direct measurement coming from for example GNSS, pressure sensor or magnetic compasses, the accelerometer and gyro biases become observable and thus the drifting in INS will get smaller with time. The results are then with full pose estimates with INS only will become more accurate.

5.7.1 The state values

The ESKF consist of a nominal state \mathbf{x} , error-state $\delta\mathbf{x}$ and true states \mathbf{x}_t . The nominal states \mathbf{x} account for a perfect state model without biases and noise terms. The error state on the other hand accounts only for the biases and noise terms. The true states will then be the combination of the nominal and the error states. A further inspection of these states are seen in the table

States list					
Description	True	Nominal	Error	Composition	Measured Noise
Position	$\mathbf{p}_{(b/n)t}^n$	$\mathbf{p}_{b/n}^n$	$\delta\mathbf{p}_{b/n}^n$	$\mathbf{p}_{(b/n)t}^n = \mathbf{p}_{b/n}^n + \delta\mathbf{p}_{b/n}^n$	
Velocity	$\mathbf{v}_{(b/n)t}^n$	$\mathbf{v}_{b/n}^n$	$\delta\mathbf{v}_{b/n}^n$	$\mathbf{v}_{(b/n)t}^n = \mathbf{v}_{b/n}^n + \delta\mathbf{v}_{b/n}^n$	
Attitude	\mathbf{q}_t	\mathbf{q}	$\delta\theta$	$\mathbf{q}_t = \mathbf{q} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta \end{bmatrix}$	
Accelerometer bias	$\mathbf{b}_{(acc)t}^b$	\mathbf{b}_{acc}^b	$\delta\mathbf{b}_{acc}^b$	$\mathbf{b}_{(acc)t}^b = \mathbf{b}_{acc}^b + \delta\mathbf{b}_{acc}^b$	\mathbf{b}_{accn}^b
Gyro bias	$\mathbf{b}_{(gyro)t}^b$	\mathbf{b}_{gyro}^b	$\delta\mathbf{b}_{gyro}^b$	$\mathbf{b}_{(gyro)t}^b = \mathbf{b}_{gyro}^b + \delta\mathbf{b}_{gyro}^b$	\mathbf{b}_{gyron}^b
Acceleration	$\mathbf{a}_{(imu)t}^b$				\mathbf{a}_{imu}^b
Angular velocity	$\omega_{(imu)t}^b$				ω_{imu}^b

Table 5.1: ESKF states list

The idea is that the nominal states are assumed as large-signal and the error-state as a small signal. By this the nominal states becomes non-linear integrable, and the error-states linearly integrable. By this the error-states are suitable for linear-Gaussian filtering.

5.7.2 The true states kinematics

By combining the gyroscope measurements given in 3.11 and the accelerometer measurements 3.10 the formulation of the true states kinematic system is formulated as a nonlinear state vector $\dot{\mathbf{x}} = f(\mathbf{x}_t, \mathbf{u}, \mathbf{b})$. Where \mathbf{x}_t , \mathbf{u} and \mathbf{w} are given as

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{p}_{(b/n)t}^n \\ \mathbf{v}_{(b/n)t}^n \\ \mathbf{q}_t \\ \mathbf{b}_{(acc)t}^b \\ \mathbf{b}_{(gyro)t}^b \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} \mathbf{a}_{imu}^b - \mathbf{w}_{acc}^b \\ \omega_{imu}^b - \mathbf{w}_{gyro}^b \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_{accn}^b \\ \mathbf{b}_{gyron}^b \end{bmatrix} \quad (5.35)$$

Where \mathbf{a}_{imu}^b is the equation given in 3.10 and ω_{imu}^b is the equation given in 3.11. The \mathbf{w}_{acc}^b and \mathbf{w}_{gyro}^b denotes the acceleration and gyro measurements noise respectively. \mathbf{b}_{gyro}^b and \mathbf{b}_{acc}^b denotes the acceleration bias and gyro bias noise respectively.

With these the true-state kinematics system can be formulated as follows [21]

$$\dot{\mathbf{p}}_{(b/n)_t}^n = \mathbf{v}_{(b/n)_t}^n \quad (5.36)$$

$$\dot{\mathbf{v}}_{(b/n)_t}^n = \mathbf{R}_b^n(\mathbf{q})(\mathbf{a}_{imu}^b - \mathbf{b}_{(acc)_t}^b - \mathbf{w}_{acc}^b) + \mathbf{g}^n \quad (5.37)$$

$$\dot{\mathbf{q}}_t = \frac{1}{2} \mathbf{q}_t \otimes (\omega_{imu}^b - \mathbf{b}_{(gyro)_t}^b - \mathbf{w}_{gyro}^b) \quad (5.38)$$

$$\dot{\mathbf{b}}_{(acc)_t}^b = -p_{b_{acc}^b} \mathbf{I}_{3 \times 3} \mathbf{b}_{(acc)_t}^b + \mathbf{b}_{accn}^b \quad (5.39)$$

$$\dot{\mathbf{b}}_{(gyro)_t}^b = -p_{b_{gyro}^b} \mathbf{I}_{3 \times 3} \mathbf{b}_{(gyro)_t}^b + \mathbf{b}_{gyron}^b \quad (5.40)$$

The bias $\dot{\mathbf{b}}_{(acc)_t}^b$ and $\dot{\mathbf{b}}_{(gyro)_t}^b$ are modeled as a Gauss-Markov process, where $p = 1/T$ and T is a time constant [21]. $p_{b_{gyro}^b}$ is known as the gyro bias reciprocal time constant and $p_{b_{acc}^b}$ is known as the acceleration bias reciprocal time constant.

The bias noises \mathbf{b}_{accn}^b and \mathbf{b}_{gyron}^b and measurement noises \mathbf{a}_n^{acc} and \mathbf{a}_n^{gyro} are modeled as a white noise process. These bias noises are modeled as a white noise process, and are assumed Gaussian. These can be stated as follows. [2]

$$\mathbf{w}_{acc}^b \sim \mathcal{N}(\mathbf{0}, \mathbf{V}) \quad (5.41)$$

$$\mathbf{w}_{gyro}^b \sim \mathcal{N}(\mathbf{0}, \mathbf{\Theta}) \quad (5.42)$$

$$\mathbf{b}_{accn}^b \sim \mathcal{N}(\mathbf{0}, \mathbf{A}) \quad (5.43)$$

$$\mathbf{b}_{gyron}^b \sim \mathcal{N}(\mathbf{0}, \mathbf{\Omega}) \quad (5.44)$$

Where \mathbf{V} , $\mathbf{\Theta}$, \mathbf{A} and $\mathbf{\Omega}$ are given as [2]

$$\mathbf{V} = \sigma_{w_{acc}^b}^2 \Delta t^2 \mathbf{I}_{3 \times 3} \quad (5.45)$$

$$\mathbf{\Theta} = \sigma_{w_{gyro}^b}^2 \Delta t^2 \mathbf{I}_{3 \times 3} \quad (5.46)$$

$$\mathbf{A} = 2p_{b_{acc}^b} \sigma_{b_{accn}^b}^2 \mathbf{I}_{3 \times 3} \quad (5.47)$$

$$\mathbf{\Omega} = 2p_{b_{gyro}^b} \sigma_{b_{gyron}^b}^2 \mathbf{I}_{3 \times 3} \quad (5.48)$$

The $\sigma_{w_{acc}^b}^2$, $\sigma_{w_{gyro}^b}^2$, $\sigma_{b_{accn}^b}^2$ and $\sigma_{b_{gyron}^b}^2$ can be found by the Allan variance method described in section 3.1.4.

5.7.3 The nominal state kinematics

The nominal state kinematics are defined as the true states without any noise. Thus the nominal states become

$$\dot{\mathbf{p}}_{b/n}^n = \mathbf{v}_{b/n}^n \quad (5.49)$$

$$\dot{\mathbf{v}}_{b/n}^n = \mathbf{R}_b^n(\mathbf{q})(\mathbf{a}_{imu}^b - \mathbf{b}_{acc}^b) + \mathbf{g}^n \quad (5.50)$$

$$\dot{\mathbf{q}} = \frac{1}{2} \mathbf{q} \otimes (\omega_{imu}^b - \mathbf{b}_{gyro}^b) \quad (5.51)$$

$$\dot{\mathbf{b}}_{acc}^b = -p_{b_{acc}^b} \mathbf{I}_{3 \times 3} \mathbf{b}_{acc}^b \quad (5.52)$$

$$\dot{\mathbf{b}}_{gyro}^b = -p_{b_{gyro}^b} \mathbf{I}_{3 \times 3} \mathbf{b}_{gyro}^b \quad (5.53)$$

5.7.4 The error state kinematics

The error state kinematics takes as stated all the noise and perturbations in account. The error state is denoted as $\delta \mathbf{x}$ and are described by the following equations.

$$\delta \dot{\mathbf{p}}_{b/n}^n = \delta \mathbf{v}_{b/n}^n \quad (5.54)$$

$$\delta \dot{\mathbf{v}}_{b/n}^n = -\mathbf{R}_b^n(\mathbf{q})S(\mathbf{a}_{imu}^b - \mathbf{b}_{acc}^b)\delta\theta - \mathbf{R}_b^n(\mathbf{q})\delta\mathbf{b}_{acc}^b - \mathbf{R}_b^n(\mathbf{q})\mathbf{w}_{acc}^b \quad (5.55)$$

$$\delta \dot{\theta} = -S(\omega_{imu}^b - \mathbf{b}_{gyro}^b)\delta\theta - \delta\mathbf{b}_{gyro}^b - \mathbf{w}_{gyro}^b \quad (5.56)$$

$$\delta \dot{\mathbf{b}}_{acc}^b = -p_{b_{acc}^b} \mathbf{I}_{3 \times 3} \mathbf{b}_{acc}^b + \mathbf{b}_{acc}^b \quad (5.57)$$

$$\delta \dot{\mathbf{b}}_{gyro}^b = -p_{b_{gyro}^b} \mathbf{I}_{3 \times 3} \mathbf{b}_{gyro}^b + \mathbf{b}_{gyro}^b \quad (5.58)$$

5.7.5 The discrete nominal state kinematics

In order to use the error-state Kalman filter, the nonlinear dynamics of the nominal states must be discretized from continuous time. Since there does not exist closed-form solution of the continuous nominal state kinematics [2]. The discretization must be done by numerical integration. This can be done by numerical integration of the with the implicit Runge-kutta of order 4 (RK4). The discrete nominal state kinematics can be stated as follows.

$$\mathbf{p}_{b/n}^n = \mathbf{p}_{b/n}^n + \mathbf{v}_{b/n}^n \Delta t + \frac{1}{2} (\mathbf{R}_b^n(\mathbf{q})(\mathbf{a}_{imu}^b - \mathbf{b}_{acc}^b) + \mathbf{g}^n) \Delta t^2 \quad (5.59)$$

$$\mathbf{v}_{b/n}^n = \mathbf{v}_{b/n}^n + (\mathbf{R}_b^n(\mathbf{q})(\mathbf{a}_{imu}^b - \mathbf{b}_{acc}^b) + \mathbf{g}^n) \Delta t \quad (5.60)$$

$$\mathbf{q} = \frac{1}{2} \mathbf{q} \otimes \mathbf{e} \left(\begin{bmatrix} 0 \\ \omega_{imu}^b \end{bmatrix} - \begin{bmatrix} 0 \\ \mathbf{b}_{gyro}^b \end{bmatrix} \right) \Delta t \quad (5.61)$$

$$\mathbf{b}_{acc}^b = -p_{b_{acc}^b} \mathbf{I}_{3 \times 3} \mathbf{b}_{acc}^b \quad (5.62)$$

$$\mathbf{b}_{gyro}^b = -p_{b_{gyro}^b} \mathbf{I}_{3 \times 3} \mathbf{b}_{gyro}^b \quad (5.63)$$

5.7.6 The discrete error state kinematics

Since the continuous error-state kinematics is a linear time-varying (LTV) system, it can be formulated as follows. Here $\mathbf{0}$ and \mathbf{I} are defined as a 3×3 matrices

$$\delta \dot{\mathbf{x}} = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_b^n(\mathbf{q})S(\mathbf{a}_{imu}^b - \mathbf{b}_{acc}^b) & -\mathbf{R}_b^n(\mathbf{q}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -S(\omega_{imu}^b - \mathbf{b}_{gyro}^b) & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -p_{b_{acc}^b} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -p_{b_{gyro}^b} \mathbf{I} \end{bmatrix} \delta \mathbf{x} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{R}_b^n(\mathbf{q}) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{n} \quad (5.64)$$

Where $\mathbf{n} = [\mathbf{w}_{acc}^b, \mathbf{w}_{gyro}^b, \mathbf{b}_{accn}^b, \mathbf{b}_{gyron}^b]^T$ represents the noises in the IMU input. By looking at equations 5.41 this can be modeled as a Gaussian distribution as follows.

$$\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{D}) \quad (5.65)$$

Where \mathbf{D} is the combination of \mathbf{V} , $\mathbf{\Theta}$, \mathbf{A} and $\mathbf{\Omega}$ and is represented as a diagonal matrix $\mathbf{D} = blkdiag(\mathbf{V}, \mathbf{\Theta}, \mathbf{A}, \mathbf{\Omega})$.

The system is now on the form

$$\delta \dot{\mathbf{x}} = \mathbf{A} \delta \mathbf{x} + \mathbf{G} \mathbf{n} \quad (5.66)$$

The discretized version will be on the form

$$\delta \mathbf{x} = \mathbf{F} \delta \mathbf{x} + \mathbf{v} \quad (5.67)$$

where $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$. Here \mathbf{Q} denotes the discrete time version covariance matrix of \mathbf{D} . This can be seen as the kinematic prior of the error-state Kalman filter.

If the quantities \mathbf{q} , $\mathbf{a}_{imu}^b - \mathbf{b}_{acc}^b$ and $\omega_{imu}^b - \mathbf{b}_{gyro}^b$ must be assumed constant over a time interval for discretization to be possible [21]. This discretization can then be done by Van Loan's formula [21] (eq. 4.63 p.61).

$$e^{\begin{bmatrix} -\mathbf{A} & \mathbf{G} \mathbf{D} \mathbf{G}^T \\ \mathbf{0} & \mathbf{A}^T \end{bmatrix} \mathbf{T}} = \begin{bmatrix} e^{-\mathbf{A} \mathbf{T}} & e^{-\mathbf{A} \mathbf{T}} \mathbf{Q} \\ \mathbf{0} & e^{(\mathbf{A} \mathbf{T})^T} \end{bmatrix} \quad (5.68)$$

where \mathbf{T} denotes the discretization time difference $t_k - t_{k-1}$. This time difference can be set to be equal to $\frac{1}{f_{imu}}$. Where f_{imu} is the frequency of which IMU measurements arrive. With this \mathbf{F} and \mathbf{Q} can be found as follows

$$\mathbf{F} = e^{\mathbf{A} \mathbf{T}} \quad (5.69)$$

$$\mathbf{Q} = (e^{\mathbf{A} \mathbf{T}^T})^T e^{-\mathbf{A} \mathbf{T}} \quad (5.70)$$

Since these equations requires a matrix exponential, it may be slow on embedded hardware. To handle this, the transition matrix \mathbf{F} can be Taylor approximated. This approximation is stated as follows

$$\mathbf{F} = e^{\mathbf{A} \mathbf{T}} = \mathbf{I} + \mathbf{A} \mathbf{T} + \frac{1}{2} \mathbf{A}^2 \mathbf{T}^2 + \frac{1}{6} \mathbf{A}^3 \mathbf{T}^3 + \dots = \sum_{k=0}^{\infty} \frac{1}{k!} \mathbf{A}^k \mathbf{T}^k \quad (5.71)$$

where the \mathbf{Q} matrix can be defined as

$$\mathbf{Q} = \mathbf{G} \mathbf{D} \mathbf{G}^T \quad (5.72)$$

On Manta-2020 a third order approximation of the transition matrix \mathbf{F} was used instead of Van Loan when estimating real time on Manta-2020. Van Loan increased the computation time dramatically, which were not feasible to be calculated in the OBC in Manta-2020. This was due to the matrix exponential calculation.

5.7.7 The prediction step

Having now found the discrete error state kinematics, it is now possible to form the prediction step in the filter. This can be written as follows [2].

$$\delta\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\delta\hat{\mathbf{x}}_{k-1} \quad (5.73)$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{Q} \quad (5.74)$$

which is the same prediction step as in the KF, but now with the error-state kinematics instead of the previous posterior state estimate.

5.7.8 The update step

In order for the filter to not just use dead reckoning, complementary sensor data such as DVL and pressure sensor have direct measurements of $\mathbf{v}_{b/n}^b$ and $\mathbf{z}_{b/n}^n$ respectively. This is done in the update step of the ESKF.

The incoming measurement can be stated as follows.

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_{kt}) + \mathbf{w}_k \quad (5.75)$$

with \mathbf{w}_k as in 5.6.

Since this is a function of the true kinematics \mathbf{x}_t it is formulated as follows.

$$\mathbf{x}_t = \mathbf{x} \oplus \delta\mathbf{x} = \begin{bmatrix} \mathbf{p}_{b/n}^n + \delta\mathbf{p}_{b/n}^n \\ \mathbf{v}_{b/n}^n + \delta\mathbf{v}_{b/n}^n \\ \mathbf{q} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\theta \end{bmatrix} \\ \mathbf{b}_{acc}^b + \delta\mathbf{b}_{acc}^b \\ \mathbf{b}_{gyro}^b + \delta\mathbf{b}_{gyro}^b \end{bmatrix} \quad (5.76)$$

where \oplus denotes a conventional sum for all the states except for the attitude.

Thus 5.75 can be rewritten as follows [21].

$$\mathbf{z}_k = h(\mathbf{x} \oplus \delta\mathbf{x}) + \mathbf{w}_k \quad (5.77)$$

Using the results found in [2], the Jacobian of \mathbf{h} can now be found as follows.

$$\mathbf{H} = \left. \frac{\partial h}{\partial \delta\mathbf{x}} \right|_{\mathbf{x}_t=\mathbf{x}} = \left. \frac{\partial h}{\partial \mathbf{x}_t} \right|_{\mathbf{x}_t=\mathbf{x}} \left. \frac{\partial \mathbf{x}_t}{\partial \delta\mathbf{x}_t} \right|_{\mathbf{x}_t=\mathbf{x}} = \mathbf{H}_x \mathbf{X}_{\delta\mathbf{x}} \quad (5.78)$$

Where $\mathbf{X}_{\delta\mathbf{x}}$ is found to be [2]

$$\mathbf{X}_{\delta\mathbf{x}} = \begin{bmatrix} \mathbf{I}_{6 \times 6} & 0 & 0 \\ 0 & \mathbf{Q}_{\delta\mathbf{x}} & 0 \\ 0 & 0 & \mathbf{I}_{6 \times 6} \end{bmatrix} \quad (5.79)$$

with $\mathbf{Q}_{\delta\mathbf{x}}$ as follows.

$$\mathbf{Q}_{\delta\mathbf{x}} = \frac{1}{2} \begin{bmatrix} -\epsilon_1 & -\epsilon_2 & -\epsilon_3 \\ \eta & -\epsilon_3 & \epsilon_2 \\ \epsilon_3 & \eta & -\epsilon_1 \\ -\epsilon_2 & \epsilon_1 & \eta \end{bmatrix} \quad (5.80)$$

Where $\mathbf{q} = [\eta, \epsilon_1, \epsilon_2, \epsilon_3]^T$

With this the update step of the ESKF can be formed

$$\mathbf{W}_k = \mathbf{P}_{k|k-1} \mathbf{H}^T (\mathbf{H} \mathbf{P}_{k|k-1} \mathbf{H}^T + \mathbf{R})^{-1} \quad (5.81)$$

$$\delta \hat{\mathbf{x}}_k = \mathbf{W}_k (\mathbf{z}_k - h(\mathbf{x}_k)) \quad (5.82)$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{W}_k \mathbf{H}) \mathbf{P}_{k|k-1} \quad (5.83)$$

For Manta-2020 the aiding sensors are the DVL and pressure sensor, which are measuring $\mathbf{v}_{b/n}^{dvl}$ and $\mathbf{z}^{pressure}$ according to the equations 4.1 and 4.2. To implement the DVL the following method can be used [1].

$$\mathbf{z}_k = h(\mathbf{v}_{(b/n)tk}^n) + \mathbf{w}_k \quad (5.84)$$

$$= \mathbf{v}_{(b/n)k}^b + \mathbf{w}_k \quad (5.85)$$

$$= \mathbf{R}_w^b(\mathbf{q}_{tk}) (\mathbf{v}_{(b/w)tk}^n + \mathbf{R}_b^n(\mathbf{q}_{tk}) S(\omega_{b/nk}^b) \mathbf{r}_{sensor}^b) + \mathbf{w}_k \quad (5.86)$$

Where $\mathbf{v}_{(b/sensor)k}^b = \mathbf{R}_{sensor}^b(\mathbf{q}) \mathbf{v}_{sensor/b}^{sensor}$.

For vessels and vehicles where the moment arm \mathbf{r}_{sensor}^b is small, it can be neglected. Also for slowly moving AUV's, with relatively small motion in roll and pitch, this moment arm will contribute very little to the velocity. This was the case for Manta-2020.

$$\mathbf{z}_k = h(\mathbf{v}_{(b/n)tk}^n) + \mathbf{w}_k \quad (5.87)$$

$$= \mathbf{v}_{(b/bottom)k}^b + \mathbf{w}_k \quad (5.88)$$

$$= \mathbf{R}_n^b(\mathbf{q}_{tk}) \mathbf{v}_{(b/n)tk}^n + \mathbf{w}_k \quad (5.89)$$

Here it is seen that the measurement model are dependent on the states \mathbf{q} and \mathbf{v} . With this the measurement Jacobin can be found as follows.

$$\mathbf{H}_x = [\mathbf{0}_{3 \times 3} \quad \mathbf{H}_v \mathbf{3 \times 3} \quad \mathbf{H}_q \mathbf{3 \times 4} \quad \mathbf{0}_{3 \times 6}] \quad (5.90)$$

where \mathbf{H}_v and \mathbf{H}_q are stated as follows.

$$\mathbf{H}_v = \left. \frac{\partial \mathbf{R}_n^b(\mathbf{q}_{tk}) (\mathbf{v}_{(b/n)tk}^n)}{\partial \mathbf{v}_{(b/n)tk}^n} \right|_{\mathbf{v}_{(b/n)tk}^n = \mathbf{v}_{(b/n)k}^n} \quad (5.91)$$

$$\mathbf{H}_q = \left. \frac{\partial \mathbf{R}_n^b(\mathbf{q}_{tk}) (\mathbf{v}_{(b/n)tk}^n)}{\partial \mathbf{q}_{tk}} \right|_{\mathbf{q}_{tk} = \mathbf{q}_{tk}} \quad (5.92)$$

The solutions are then found to be [2]

$$\mathbf{H}_v = \mathbf{R}_n^b(\mathbf{q}_{tk}) \quad (5.93)$$

$$\mathbf{H}_q = 2[\eta \mathbf{v}_{(b/n)_{tk}}^n + \epsilon \times \mathbf{v}_{(b/n)_{tk}}^n \mid \epsilon^T \mathbf{v}_{(b/n)_{tk}}^n \mathbf{I}_{3 \times 3} + \epsilon (\mathbf{v}_{(b/n)_{tk}}^n)^T - \mathbf{v}_{(b/n)_{tk}}^n \epsilon^T - \eta S(\mathbf{v}_{(b/n)_{tk}}^n)] \quad (5.94)$$

Where $\mathbf{q}_{tk} = \begin{bmatrix} \eta \\ \epsilon \end{bmatrix}$

With sensors like the pressure sensor that measures in the local world frame, the equations become relatively much simpler. For a 1 DOF vertical position estimate, like a pressure sensor, the following equation can be used.

$$z_k = h(z_{(b/n)_{tk}}^n) + w_k = z_{(b/n)_{tk}}^n + w_k \quad (5.95)$$

With this \mathbf{H}_x is stated as follows

$$\mathbf{H}_x = \begin{bmatrix} \mathbf{0}_{2 \times 2} & \mathbf{H}_{z_{1 \times 1}} & \mathbf{0}_{3 \times 13} \end{bmatrix} \quad (5.96)$$

With \mathbf{H}_z as

$$\mathbf{H}_z = \frac{\partial z_{(b/n)_{tk}}^n}{\partial z_{(b/n)_{tk}}^n} \Big|_{z_{(b/n)_{tk}}^n = z_{(b/n)_k}^n} = \mathbf{I}_{1 \times 1} \quad (5.97)$$

5.7.9 ESKF injection and reset step

In order for the filter

After the update step, the error state $\delta \mathbf{x}$ has no longer a mean of zero. The nominal states must then be updated with the observed error state. This is done in the ESKF injection step. This is stated as follows.

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k \oplus \delta \hat{\mathbf{x}} \quad (5.98)$$

After the injection step, the nominal states have been updated, there is of no interest of keeping the error state in the next cycle of the ESKF. Therefore it has to be reset. The ESKF reset step is stated as follows.

$$\delta \hat{\mathbf{x}}_k = \mathbf{0} \quad (5.99)$$

$$\mathbf{P}_k = \mathbf{G} \mathbf{P}_k \mathbf{G}^T \quad (5.100)$$

where \mathbf{G} is calculated as follows.

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_{6 \times 6} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{3 \times 3} - S(\frac{1}{2} \delta \hat{\theta}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{6 \times 6} \end{bmatrix} \quad (5.101)$$

Where S is the skew symmetric matrix.

5.8 Nonlinear Observer

Together with the extended Kalman filters and error-state Kalman filters discussed above, there are nonlinear state estimators/observers that are based on nonlinear stability theory. These observers may have the advantage of having a semiglobal or sometimes global convergence even with potentially large inaccurate initial estimates and unknown noise statistics [42]. Compared to the EKF, some nonlinear observers does not rely on linearization for covariance calculation [42]. Also compared to both the EKF and ESKF, they may be the best choice for low-cost applications, because of the relatively large computational cost of the Kalman filters [3]. Where for the EKF the linearization and solving the Riccati equation makes it computationally expensive and for the ESKF the need for exponential mapping may give slow performance.

The nonlinear observer presented in this thesis is a modification of the nonlinear observer based on the paper [3]. This is based on a nonlinear attitude observer which is feedback interconnected to an translational motion observer (TMO). The TMO has many similar steps as the EKF, and it solves the Riccati equation.

5.8.1 Assumptions and sensor configuration

The nonlinear observer presented assumes the following sensors and sensor measurements are available.

- A non-biased measurement \mathbf{a}_{acc}^b coming from the IMU accelerometer.
- A biased measurement ω_{gyro}^b coming from the IMU gyro.
- A full velocity measurement \mathbf{v}_{sensor}^b coming from a GNSS receiver with the following measurement $\mathbf{R}_e^b(\mathbf{q})\mathbf{v}_{GNSS}^e$ or a DVL measurement \mathbf{v}_{dvl}^b or some other sensor that may be transformed or rotated into $\mathbf{v}_{b/n}^b$.

Together with this the following assumptions are also assumed.

- A bound \mathbf{M}_{gyro} on the magnitude of the gyro bias \mathbf{b}_{gyro}^b .
- The angular velocity $\omega_{b/n}^b$ and the time derivative of \mathbf{a}_{acc}^b are uniformly bounded.

5.8.2 Attitude estimation

The first sub-problem will be on estimating the \mathbf{q} and gyro bias \mathbf{b}_{gyro}^b , by the use of two hypothetical pair of vector measurements in the $\{b\}$ frame, $\nu_1^b = \mathbf{a}_{acc}^b$ and $\nu_2^b = \mathbf{v}_{sensor}^b$. These must have the property that

$$\|\mathbf{a}_{acc}^b \times \mathbf{v}_{sensor}^b\| > 0 \quad (5.102)$$

Where \times is the cross-product. With this the attitude observer state equations are written as follows.

$$\dot{\hat{\mathbf{q}}} = \frac{1}{2} \hat{\mathbf{q}} \otimes \left(\begin{bmatrix} 0 \\ \omega_{imu}^b \end{bmatrix} - \begin{bmatrix} 0 \\ \mathbf{b}_{gyro}^b \end{bmatrix} + \begin{bmatrix} 0 \\ \hat{\sigma} \end{bmatrix} \right) \quad (5.103)$$

$$\dot{\hat{\mathbf{b}}}_{imu}^b = Proj(\mathbf{b}_{gyro}^b, -k_i \hat{\sigma}) \quad (5.104)$$

Where Proj denotes a projection that restricts the gyro bias estimate \mathbf{b}_{gyro}^b such that $\|\mathbf{b}_{gyro}^b\| < \mathbf{M}_{gyro}$ [3] and where $\hat{\sigma}$ is an injection term that is defined as follows

$$\begin{aligned} \hat{\sigma} &= S(k_1 \mathbf{a}_{acc}^b) \mathbf{R}_n^b sat_{M_f}(\hat{\mathbf{a}}_{acc}^n) + S(k_2 \mathbf{v}_{sensor}^b) \mathbf{R}_n^b \mathbf{v}_{sensor}^n \quad \text{if } f_{bi} == \text{true} \\ \hat{\sigma} &= S(k_1 \mathbf{a}_{acc}^b) \mathbf{R}_n^b sat_{M_f}(-\mathbf{g}^n) + S(k_2 \mathbf{v}_{sensor}^b) \mathbf{R}_n^b \mathbf{v}_{sensor}^n \quad \text{otherwise} \end{aligned} \quad (5.105)$$

where S is the skew symmetric matrix, k_1 and k_2 are gain tuning parameters and the f_{bi} is the feedback interconnection between the TMO and NLO. Since \mathbf{a}_{acc}^n is not directly measured, it has to be estimated, leaving it to $\hat{\mathbf{a}}_{acc}^n$. The sat_{M_f} is a saturation bound based on the magnitude bound \mathbf{M}_f . Also if one were to use a GNSS sensor the velocity \mathbf{v}_{sensor}^n would be directly measurable. But if the measurements were coming from a sensor that measured velocity in the $\{b\}$ frame only, like a DVL. The following $\hat{\sigma}$ has to be used.

$$\begin{aligned}\hat{\sigma} &= S(k_1 \mathbf{a}_{acc}^b) \mathbf{R}_n^b sat_{M_f}(\hat{\mathbf{a}}_{acc}^n) + k_2 S(S(\mathbf{a}_{acc}^b) \mathbf{v}_{sensor}^b) \mathbf{R}_n^b S(sat_{M_f}(\hat{\mathbf{a}}_{acc}^n)) \hat{\mathbf{v}}_{b/n}^n \quad \text{if } f_{bi} == \text{true} \\ \hat{\sigma} &= S(k_1 \mathbf{a}_{acc}^b) \mathbf{R}_n^b sat_{M_f}(-\mathbf{g}^n) + k_2 S(S(\mathbf{a}_{acc}^b) \mathbf{v}_{sensor}^b) \mathbf{R}_n^b S(sat_{M_f}(\hat{\mathbf{a}}_{acc}^n)) \hat{\mathbf{v}}_{b/n}^n \quad \text{otherwise}\end{aligned}\quad (5.106)$$

Where $\hat{\mathbf{v}}_{b/n}^n$ is the velocity estimate given in equation 5.115 summarized below.

The discrete time version can be found by using Runge Kutta method 4 (RK-4). The resulting equations are stated below.

$$\hat{\mathbf{q}} = \frac{1}{2} \hat{\mathbf{q}} \otimes \mathbf{e}^{\Delta t \left(\begin{bmatrix} 0 \\ \omega_{imu}^b \end{bmatrix} - \begin{bmatrix} 0 \\ \hat{\mathbf{b}}_{gyro}^b \end{bmatrix} + \begin{bmatrix} 0 \\ \hat{\sigma} \end{bmatrix} \right)} \quad (5.107)$$

For embedded systems where calculations of the matrix exponential may be computationally expensive, the quaternion estimation \mathbf{q} can be estimated with the following equation

$$\hat{\mathbf{q}} = \frac{1}{2} \mathbf{q} \otimes \begin{bmatrix} \cos\left(\frac{\Delta t}{2}(\omega_{imu}^b - \mathbf{b}_{gyro}^b + \hat{\sigma})\right) \\ \sin\left(\frac{\Delta t}{2}(\omega_{imu}^b - \mathbf{b}_{gyro}^b)\right) \Delta t(\omega_{imu}^b - \mathbf{b}_{gyro}^b + \hat{\sigma}) \end{bmatrix} \quad (5.108)$$

One important notice is that the quaterion has to be normalized in either of these cases.

The discrete time version of the gyro bias estimation $\hat{\mathbf{b}}_{gyro}^b$ is not straightforward because it is based on the projection term. In order to handle this the discrete time version is calculated as follows.

$$\begin{aligned}\hat{\mathbf{b}}_{gyro}^b &= -(\mathbf{I}_{3x3} - \frac{\hat{\mathbf{b}}_{gyro}^b \hat{\mathbf{b}}_{gyro}^{bT}}{\hat{\mathbf{b}}_{gyro}^b \hat{\mathbf{b}}_{gyro}^{bT}}) k_i \hat{\sigma} \quad \text{if } \hat{\mathbf{b}}_{gyro}^{bT} \hat{\mathbf{b}}_{gyro}^b > M_{gyro}^2 \text{ and } -\mathbf{b}_{gyro}^{bT} k_i \hat{\sigma} > 0 \\ \hat{\mathbf{b}}_{gyro}^b &= \hat{\mathbf{b}}_{gyro}^b - k_i \hat{\sigma} \quad \text{otherwise}\end{aligned}\quad (5.109)$$

5.8.3 Position, velocity and NED acceleration estimation

In the paper [3], the position and velocity estimates are dependent on position and velocity measurements coming directly from a GNSS reciever. In order to use this nonlinear observer independent on absolute position and velocity measurements, some modifications on the position and velocity observer equations had to be made. These are stated below. The TMO then becomes as follows.

$$\dot{\hat{\mathbf{p}}}_{b/n}^n = \hat{\mathbf{v}}_{b/n}^n \quad (5.110)$$

$$\dot{\hat{\mathbf{v}}}_{b/n}^n = \hat{\mathbf{a}}_{acc}^n + \mathbf{g}^n \quad (5.111)$$

$$\dot{\hat{\xi}} = -\mathbf{R}_b^n(\hat{\mathbf{q}}) S(\hat{\sigma}) \mathbf{a}_{acc}^b \quad (5.112)$$

$$\hat{\mathbf{a}}_{acc}^n = \mathbf{R}_b^n(\hat{\mathbf{q}}) \mathbf{a}_{acc}^b + \hat{\xi} \quad (5.113)$$

By using numerical integration with the implicit Runge-kutta of order 4 (RK4), the discrete time version can be found. These are stated below as follows.

$$\hat{\mathbf{p}}_{b/n}^n = \hat{\mathbf{p}}_{b/n}^n + \hat{\mathbf{v}}_{b/n}^n \Delta t + \Delta t \xi_{b/n}^n + \mathbf{R}_b^n((\hat{\mathbf{q}})(\frac{1}{2}\Delta t^2 - \frac{1}{6}\Delta t^3 S(\sigma))\mathbf{a}_{acc}^b) + \mathbf{g}^n \frac{1}{2}\Delta t^2 \quad (5.114)$$

$$\hat{\mathbf{v}}_{b/n}^n = \Delta t \xi^n + \mathbf{R}_b^n(\hat{\mathbf{q}})((\Delta t - \frac{1}{2}\Delta t^2 S(\sigma))\mathbf{a}_{acc}^b) + \mathbf{g}^n \Delta t \quad (5.115)$$

$$\xi^n = \xi^n - \mathbf{R}_b^n(\hat{\mathbf{q}})\Delta t S(\sigma)\mathbf{a}_{acc}^b \quad (5.116)$$

$$\mathbf{a}_{acc}^n = \mathbf{R}_b^n(\hat{\mathbf{q}})\mathbf{a}_{acc}^b + \xi \quad (5.117)$$

5.8.4 The prediction step

With the discrete time version of the attitude, position, velocity and NED acceleration kinematics, it is now possible to form the prediction step of the observer. These equations are calculated directly from the numerical integration step.

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u} + \mathbf{D} \quad (5.118)$$

where $\hat{\mathbf{x}} = [\hat{\mathbf{p}}_{b/n}^n, \hat{\mathbf{v}}_{b/n}^n, \xi^n]^T$ and \mathbf{F} , \mathbf{B} and \mathbf{D} are calculated as follows.

$$\mathbf{F} = \begin{bmatrix} \mathbf{I} & \Delta t \mathbf{I} & \frac{1}{2}\Delta t^2 \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \Delta t \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \quad \mathbf{B} = \mathbf{R}_b^n(\mathbf{q}) \begin{bmatrix} \frac{\Delta t}{2} & \frac{\Delta t^3}{6} \\ \Delta t & \frac{\Delta t^2}{2} \\ \mathbf{0} & \Delta t \end{bmatrix} \quad \mathbf{D} = \mathbf{g}^n \begin{bmatrix} \frac{\Delta t}{2} \\ \Delta t \\ \mathbf{0}_{3 \times 1} \end{bmatrix} \quad (5.119)$$

The \mathbf{u} is dependent on the feedback interconnection term f_{bi} . It is defined as follows.

$$\mathbf{u} = \begin{cases} \begin{bmatrix} \mathbf{a}_{acc}^b \\ -S(\sigma)\mathbf{a}_{acc}^b \end{bmatrix} & \text{if } f_{bi} == \text{true} \\ \begin{bmatrix} \mathbf{a}_{acc}^b \\ \mathbf{0}_{3 \times 1} \end{bmatrix} & \text{otherwise} \end{cases} \quad (5.120)$$

Also if the feedback interconnection is true f_{bi} then \mathbf{F} is defined as follows.

$$\mathbf{F} = \begin{bmatrix} \mathbf{I} & \Delta t \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \quad (5.121)$$

The nonlinear observer uses a uncertainty measure similiar to the EKF, which is calculated as follows.

$$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_k\mathbf{F}^T + \mathbf{Q} \quad (5.122)$$

where \mathbf{P} is the uncertainty matrix and \mathbf{Q} is the discrete time version of the process noise co-variance, which is calculated using Tustins' integration.

$$\mathbf{Q} = \frac{1}{2}(\mathbf{F}\mathbf{G}_{k|k-1}\mathbf{D}\mathbf{G}_{k|k-1}^T\mathbf{F}^T + \mathbf{G}_k\mathbf{D}\mathbf{G}_k^T)\Delta t \quad (5.123)$$

Here \mathbf{D} is the continuous time process covariance matrix, which is used in the tuning of the filter. $\mathbf{G}_{k|k-1}$ and \mathbf{G}_k are Tustins matrices defined as follows

$$\mathbf{G}_{k|k-1} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{R}_q^n(\mathbf{q})_{k|k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_q^n(\mathbf{q})_{k|k-1} \end{bmatrix} \quad \mathbf{G}_k = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{R}_q^n(\mathbf{q})_k & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_q^n(\mathbf{q})_k \end{bmatrix} \quad (5.124)$$

Where $\mathbf{0}$ represent a 3×3 zero matrix.

5.8.5 The update step

The update step of the observer is based on the innovation, Kalman gain and update step of the EKF.

The measurement model is defined similiar to the EKF as follows.

$$z_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k \quad (5.125)$$

Recall that \mathbf{H} is the measurement matrix, $\mathbf{H}\mathbf{x}$ is the measured states and \mathbf{w}_k is the noise.

The innovation is defined as follows

$$\nu_k = \mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1} \quad (5.126)$$

Where $\hat{\mathbf{z}}_{k|k-1}$ is defined as $\mathbf{H}\hat{\mathbf{x}}_{k|k-1}$.

The innovation covariance is defined as

$$\mathbf{S}_k = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R} \quad (5.127)$$

Where \mathbf{R} is the measurement covariance.

The Kalman gain is defined as follows

$$\mathbf{W}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T\mathbf{S}_k^{-1} \quad (5.128)$$

With this the current posterior estimate is the same as in the EKF, which is defined as follows

$$\begin{aligned} \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{W}_k\nu_k \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{W}_k\mathbf{H})\mathbf{P}_{k|k-1} \end{aligned} \quad (5.129)$$

For velocity measurements \mathbf{v}_{sensor}^b like the DVL measuring velocity relative to the world frame. The \mathbf{H} is defined as follows.

$$\mathbf{H} = [\mathbf{0}_{3 \times 3} \quad \mathbf{R}_n^b(\hat{\mathbf{q}}) \quad \mathbf{0}_{3 \times 3}] \quad (5.130)$$

where $\hat{\mathbf{z}}_k$ is equal to

$$\mathbf{z}_k = h(\mathbf{v}_{(b/n)}^n) + \mathbf{w}_k \quad (5.131)$$

$$= \mathbf{v}_{(b/bottom)k}^b + \mathbf{w}_k \quad (5.132)$$

$$= \mathbf{R}_n^b(\hat{\mathbf{q}})\mathbf{v}_{(b/n)}^n + \mathbf{w}_k \quad (5.133)$$

Where for the pressure sensor the \mathbf{H} is given as follows.

$$\mathbf{H} = [\mathbf{0}_{1 \times 2} \quad 1 \quad \mathbf{0}_{1 \times 6}] \quad (5.134)$$

5.9 Real-time aspects

There are different type of real-time aspects that have to be considered when using embedded hardware with relatively little computational power. Because of this limitation, sensor measurements may be coming into the estimation nodes in much higher frequency than the embedded hardware can calculate the estimates coming from the state estimators. With this the estimator node will only use the latest measurements and may potentially skip important in-between measurements coming from the high frequency sensors.

Also one critical real-time aspect to consider is time-synchronization of the sensors. If a measurement coming from a sensor is time-delayed, its accuracy to the real states may be off by a large error[43]. Especially is this true for high-speed vessels or aircraft. Thus to handle this is important for higher accuracy and PVA estimation.

A second real-time aspect that is important to handle, is sensor-synchronization. If two or more sensors or more sensors are used in the state estimators, their measurements are coming with different timestamps. If the sensors are not synchronized the state estimator may estimate with an older or a newer measurement than the other sensor measurement based on their timestamp.

Finally a third real-time aspect to consider is the execution time of the state estimators. This could be the time the state estimator uses from one estimate $k - 1$ to the next estimate k or it could be the time the state estimator uses in updating the states with the incoming measurements. With this it is possible to find the execution time of a specific state estimator on a specific CPU to find out how well the state estimator may perform on embedded hardware.

5.9.1 Time-synchronization

Time delayed measurements, as discussed above, can have big impact on the accuracy of the PVA estimation, and may give large errors. Following the source [43], there are several key error sources that causes these delayed measurements. These are listed in the table below.

General Error Source	Specific Error Source	Possible Mitigation
Internal hardware delay	A/D conversion	Specified by the manufacture
	Internal DSP processing	A constant delay may be calibrated
	Transmission to communication ports	
Data transmission delay	Communication board/protocol	A constant delay may be calibrated
Registration delay	Computer clock reading	Low level coding with real time clock
	Computer IRQ priorities	Multi I/O timing board

Table 1.3: Time synchronization error sources. Courtesy([43])

As seen in table 1.3 there are three main categories of general error sources, the internal hardware delay, the data transmission delay and registration delay. The internal hardware delay is the delays that occur in the respective hardware of the sensor. Some of these delays are due to the analog to digital conversion, the filtering process, like low pass or high pass filtering and the data-transmission from the sensor to the communication port to for example the OBC. As shown in the figure, these are either stated in the data-sheet or can be calibrated because of their constant behavior. However they are not always constant, and may vary in time. For example if an adaptive filtering method is used. For an IMU these delays may be large (50-60 ms for a navigation grade IMU [43]). The second general error is the data transmission delay. These delays are due to the communication board or protocol that is used in the communication between the sensor and for example the OBC. This delay is the amount of time required to forward or push all of a sensor packet bits into the communication wire[44]. This is normally a constant delay that can be calibrated and mathematically found. The equation is given as follows [44].

$$D_{trans} = \frac{N_{packet}}{R_{trans}} \quad (5.135)$$

Where D_{trans} is the transmission delay, N_{packet} is the number of bits in the sensor packet and R_{trans} is the rate of transmission (bits/s). This is like the internal filtering and transmission, a constant delay that can be calibrated.

The third source of error is the registration delay. These are related to the computer clock reading and computer interrupt request (IRQ). These delays as seen in the figure requires a more sophisticated approach with low level coding with real time clock and a multi I/O timing board, for example a Sentiboard, to keep these delays below 1 ms [43].

5.9.2 Sensor-synchronization and sensor-buffering

The first step in sensor-synchronization is to gather information about the sensor message. For the presented state estimators in this thesis, the most important ones are listed below as follows.

- The incoming measurement \mathbf{z}
- The measurement covariance \mathbf{R}
- The incoming timestamp t_{stamp} of the message.

Where the timestamp is based on when the software program received the message, or updated with timing-synchronization by a dedicated hardware timing board from for example a Sentiboard as discussed in the section above.

With this it is now possible to make a sensor-buffer for each sensor that is used in the state estimator. In software programs these buffers can for example be vectors, arrays, dictionaries or lists depending on the programming language used. The sensor-information can now be stored in the buffer as soon as they arrive the corresponding "callback" function in the software program with the latest measurement appended to the end of the buffer. One important note to this is that too many messages in the buffer can make the estimator use a too delayed measurement, because of too long execution time, which is not convenient. Thus the buffer needs to be "emptied" to only contain the latest message after a certain threshold M_{thres} of the size of the buffer. One alternative solution to this is to take the mean of a certain number M_{mean} of incoming measurements and timestamps and then push these to the end of the buffer.

With now having a buffered sensor-system it is now possible to make an algorithm such that sensor-synchronization can be made. There are several alternatives to this. One example is to add all the incoming messages from the sensors into a common buffer, sorting the messages based on the timestamps and then taking these measurements into the state estimators. This is shown clearer in the figure 5.9 below. This was added to Manta-2020.

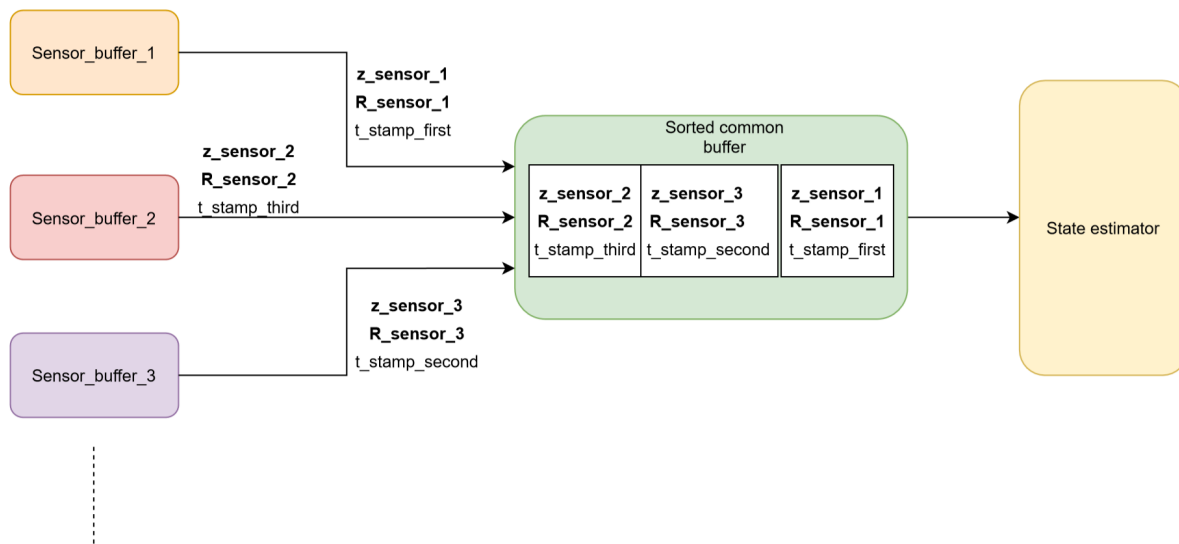


Figure 5.9: Common buffer sensor synchronization

5.9.3 Execution time

There are several methods to calculate the execution time. In for example C++11 or above, the package standard library package Chrono can be used. In Python, the Timeit library can be used. These are simple to use libraries that can be implemented anywhere in the software code to calculate the execution time or duration to a specific part in the code. For the estimators presented in this thesis, the most important execution time would be to consider the time between the estimated states and the time to update the filter with the sensor measurements. The method in this thesis would be to take the mean, standard deviation and the max value of a certain number of estimates. Then running the filters at least five times to calculate yet again their mean, standard deviation and max value. The following table shows the results for Manta-2020 in the prediction and the DVL and pressure updates steps for the ESKF and NLO.

Description	ESKF (ms)	NLO (ms)
Max value - Predict	2.938	1.471
Mean value - Predict	2.167	0.981
STD - Predict	0.171	0.115
Max value - DVL update	2.773	0.952
Mean value - DVL update	2.001	0.652
STD - DVL update	0.293	0.11
Max value - Pressure update	2.213	0.6877
Mean value - Pressure update	1.37	0.475
STD - Pressure	0.21	0.07

Table 5.2: Execution time of NLO and ESKF

Looking at table 5.2 it is seen that the NLO has the fastest execution time overall. This is when the ESKF has been adjusted to a third order approximation of the transition matrix. This was planned to be tested on the OBC during testing, but since of time constraints and COVID_19 this was not possible. The tests above was done on a laptop with a Intel core i5-9300H processor.

Experimental testing

In order to check the quality of the error-state Kalman filter, nonlinear observer and the extended-Kalman filter, experimental testing is needed. This will aim to see if the software implementations work as intended, and to point out what software needs to be redefined and debugged. It is also needed in order to see how the software implementations behaves to the hardware on AUV.

6.1 Preparation and COVID-19

A week before the physical experiments, a "preparation" period was conducted. Since this could not be done at NTNU because of the COVID-19 restrictions, one of the members of Vortex-NTNU offered a workplace at their home. Only five people were allowed to work at the same time, which meant that the author of this report and another master student from Vortex had to do much of the new hardware and electronic installment. This meant that "quick solutions" were implemented with little to no testing beforehand. Because of this, a bug not allowing us to launch the thrusters arised, meaning that all of the experimental testing had to be done without the thrusters and GNC system.

6.2 Experimental testing at the Marine Cybernetics laboratory

The experimental testing were done at the Marine Cybernetics laboratory (MC - lab) in Trondheim, Norway. It is a part of the Department of Marine Technology at NTNU where it is mainly used for master students and PhD candidates [45]. It consist of two rooms. One room with a basin of dimensions length, width and depth = 40m x 6.45m x 1.5m and a controller room. These are depicted in figure 6.1. The MC-lab also offered a Qualisys motion capture system, that could real-time 6DOF measure the AUV. Also in the basin-room, a move-able platform could be used to move along the water. This was extensively used during the experimental testing.



(a) The Marine Cybernetics testing basin and the Qualisys camera locations



(b) The Marine Cybernetics laboratory controller room. Figure from [45]

Figure 6.1: The Marine Cybernetics laboratory.



(a) Qualisys "above water" motion capture camera



(b) Qualisys underwater motion capture camera

Figure 6.2: The Qualisys motion capture system cameras

6.3 Calibration and set-up of the Qualisys motion capture systems

The MC-lab also offered a real-time 6DOF - tracking system for both underwater and "above water" environments. This was based on Qualisys camera-motion capture system. The main components in this system was the Qqus cameras, depicted in figure 6.2 and the Qualisys track manager (QTM) software. The Qqus cameras were based on an infrared (IR) technology that tracked the IR reflection from the motion balls that was fitted on a vehicle. For the "above water" system, the Qqus cameras were mounted on a towing carriage in a fixed angle and position. Figure 6.1 and 6.2a show this. For the underwater system, there were a total of 6 Qqus cameras that were mounted on a movable rail, numbered from 1 to 6. These cameras were also fixed in angle and position. Figure 6.2 shows a close up version of the camera, move-able rail and its

number. These cameras had to use internal light sources to get higher accuracy in the tracking of the IR reflections from the motion balls. This is clearer seen in figure 6.6. There were two main computers with dedicated QTM software in the control room. This meant that all the experiments could be supervised from the control room.

The accuracy of the Qualisys camera system were dependent on its tracking capabilities and calibration volume. Therefore Qualisys had to be setup and calibrated before starting the test scenarios.

6.3.1 "Above water" Qualisys camera setup

The first procedure was to calibrate and setup the "above water" Qualisys motion capture systems. This was done by first setting a fixed world coordinate frame, making a measurement volume, placing tracker balls on the AUV and then defining a 6DOF rigid body.

6.3.2 Fixed world frame setup

The first procedure to calibrate the "above water" cameras, was to make the world coordinate system and its respective origin. This was done by placing a fixed square with four tracking markers on it in the middle of the basin as seen in figure 6.3. Here the lower leftmost tracking ball is the origin of the world coordinate system, while the rightmost tracking ball is defined as the x-axis. The uppermost ball tracked the y-axis.

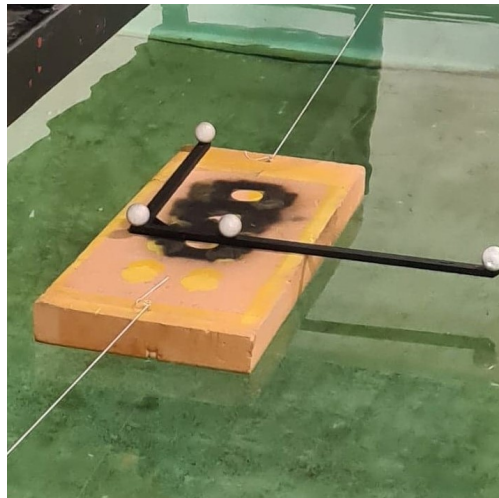


Figure 6.3: Qualisys "above water" world coordinate setup

6.3.3 Calibration volume setup

Next up was to make the calibration volume. This volume is the 3D tracking space in which the Qualisys cameras could accurately track the body frame based on the bone length tolerance. This space was also the only valid space the Qualisys could make a 3D coordinate system of the body frame of the AUV. In order to make the calibration volume, one had to use a rotating stick ("magic wand") with two tracking balls. This was a two person operation. One had to move a moveable rail backwards while another rotated the stick up and down while rotating. The procedure is seen in figure 6.4. With the underwater Qualisys camera system, spinning stick was also used. The procedure was almost the same, but now one had to spin it under the water surface instead. This is seen in figure 6.4



Figure 6.4: Qualisys "above water" calibration volume setup

After the calibration physical setup, the calibration volume was made. This is depicted in figure 6.4

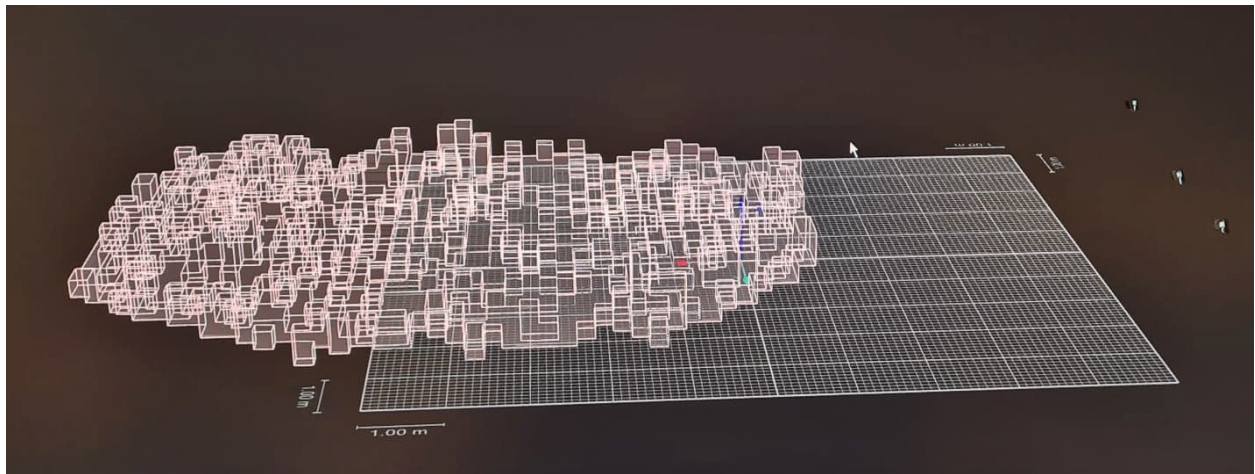


Figure 6.5: Qualisys "above water" calibration volume

6.3.4 Underwater Qualisys camera system

The underwater Qualisys camera system had a total of 6 cameras located underwater as seen in figure 6.6. The cameras emitted blue light in order to better identify the tracking balls that were placed on the AUV. Also under calibration the light sources from the basin room had to be lowered or turned off in order to remove unwanted light.

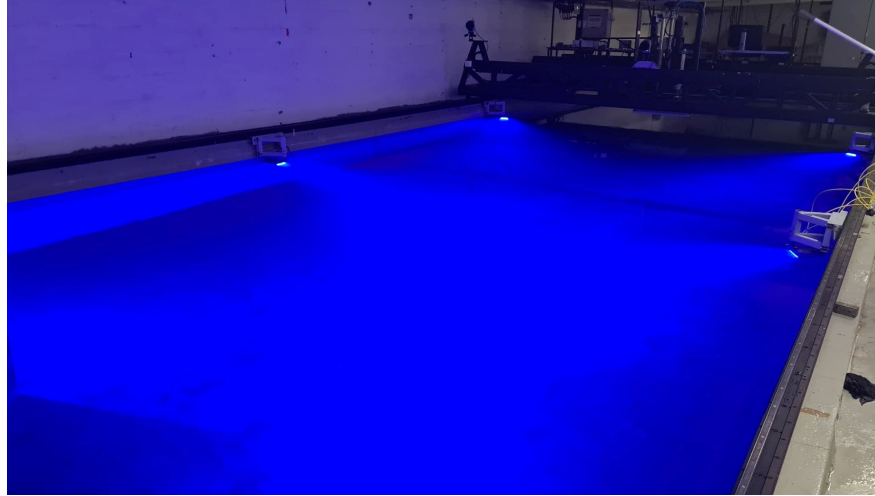


Figure 6.6: Qualisys underwater cameras locations

6.3.4.1 Calibration volume setup

The underwater calibration setup was almost the same as with the "above water" calibration. The only difference was that the spinning stick, had to be spun up and down underwater instead of above. The procedure is shown in figure 6.7



Figure 6.7: Qualisys underwater camera system physical calibration setup

For the underwater camera system the calibration volume is shown in figure 6.8. The coordinate system in the middle is defined as the world coordinate system. This was both set to an NED coordinate system, because the NLO and ESKF used this coordinate system as a fixed world frame.

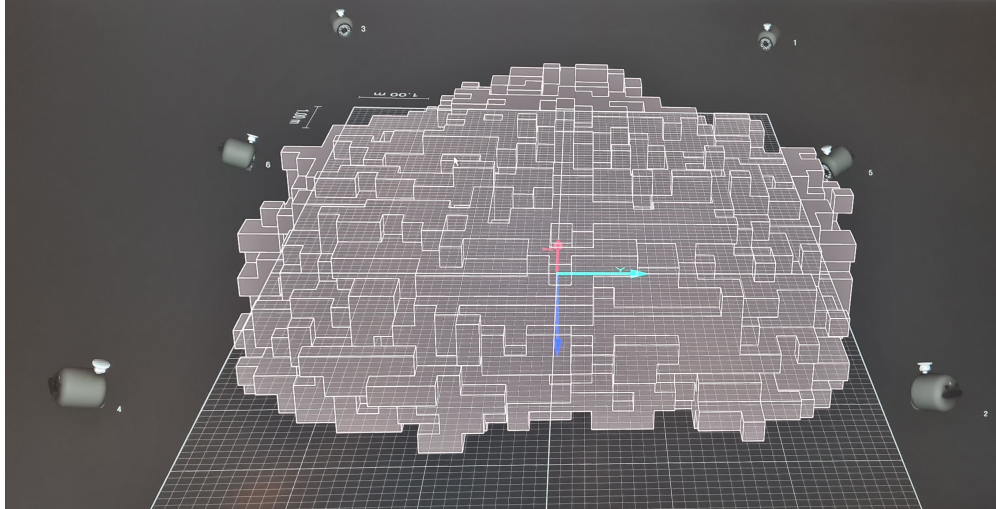


Figure 6.8: Qualisys underwater camera system calibration volume

Next up was to make the body frame coordinate system on the AUV. This was done by placing tracker balls on the AUV. A total of three tracker balls had to be seen by the cameras at all time. Therefore a total number of 6 tracker balls was used. The tracking balls setup on the AUV is seen in figure 6.9

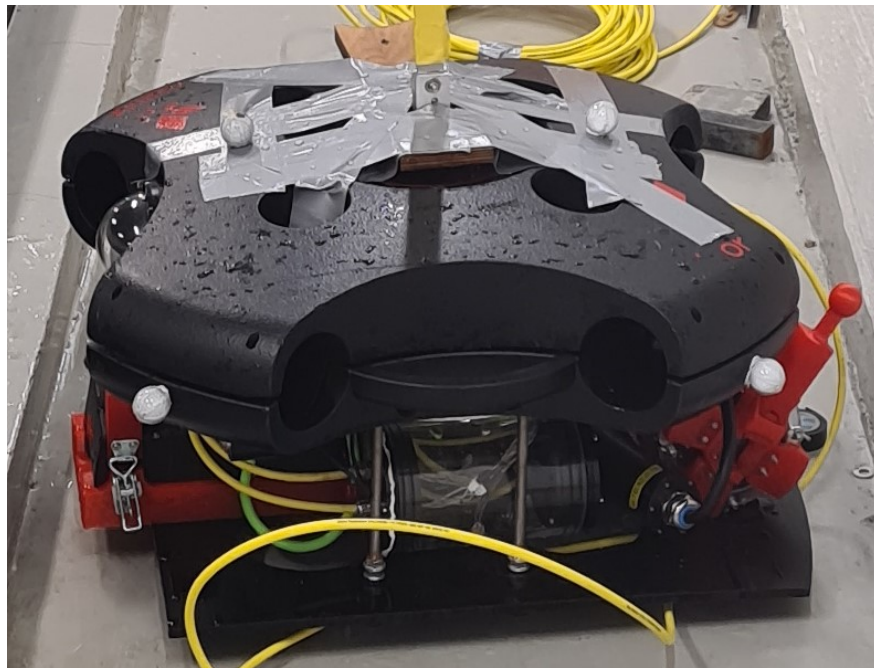
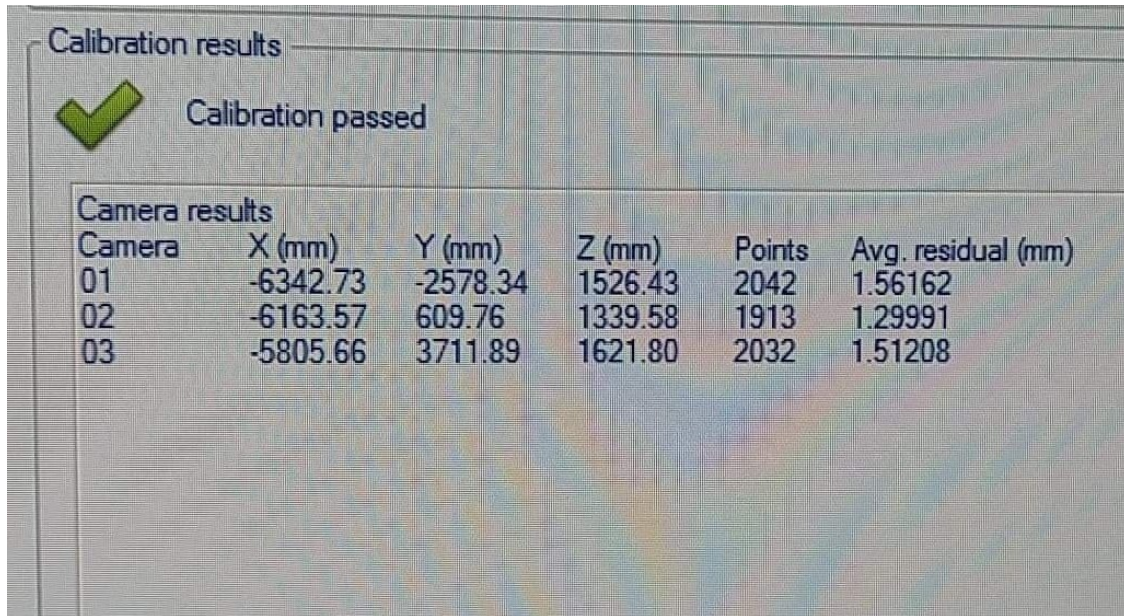


Figure 6.9: Manta-2020 tracking balls setup

6.4 Qualisys calibration results

The accuracy of the Qualisys system is based on the average residual of the calibration results. The calibration results from the "above water" system in figure 6.10. Here the "Id" is the identity of the Qqus camera. "X,Y,Z"

is the distance (in mm) between the origin of the fixed world frame to the optical center of the cameras [46]. The distance are then respectively given as a arm in the "X,Y,Z" direction[46]. The points cloumn represents the number of motion balls each camera has seen during calibration. The last column is the Average residual, which is the distance between the 2D position of the motion balls and the reference balls position coming from the fixed world coordinate system. This value could in mm tell how accurate the each camera will estimate the 6 DOF PVA. Since this exposure of this error was later discovered after the experimental testing weeks, only a picture of the "above water" calibration results were given. The COVID-19 restrictions made it hard to go back to the MC-lab and take a picture of the Qualisys underwater camera system.



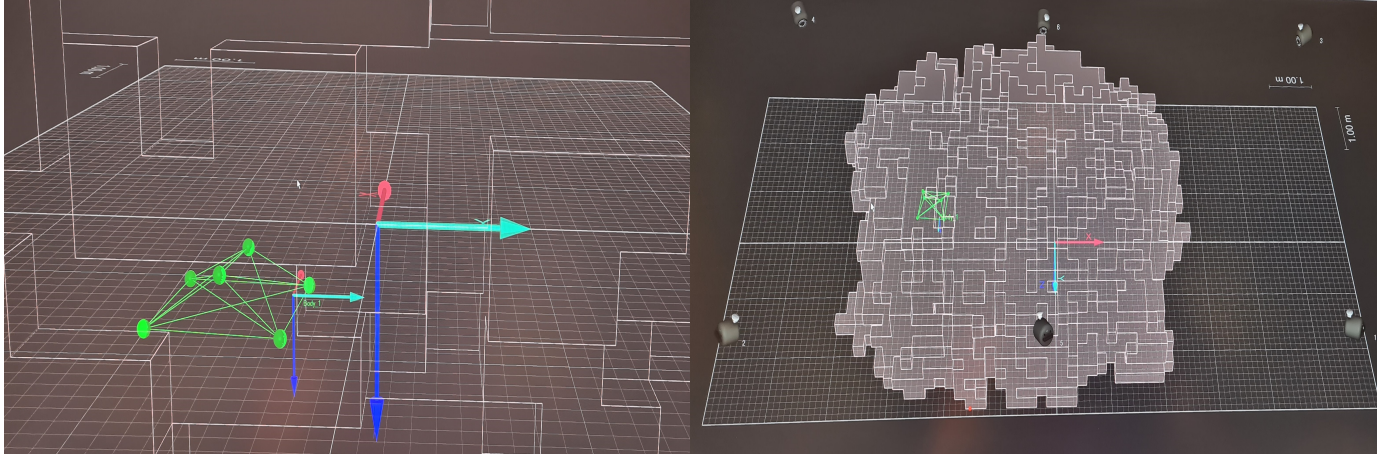
The image shows a software interface for Qualisys calibration. At the top, it says 'Calibration results' with a green checkmark and the text 'Calibration passed'. Below this is a table titled 'Camera results' with the following data:

Camera	X (mm)	Y (mm)	Z (mm)	Points	Avg. residual (mm)
01	-6342.73	-2578.34	1526.43	2042	1.56162
02	-6163.57	609.76	1339.58	1913	1.29991
03	-5805.66	3711.89	1621.80	2032	1.51208

Figure 6.10: Qualisys "above" water calibration results

6.5 Defining the 6 DOF rigid body frame

After the calibration setup was done, the 6DOF rigid body frame had to be made. This was done by placing Manta-2020 in the water with the tracker balls, and after three or more cameras had tracked more than three tracking balls, the body frame was made. These are depicted in figure 6.11.



(a) Qualisys 6DOF rigid body setup close-up

(b) Qualisys 6DOF rigid body setup zoomed-out

Figure 6.11: The Qualisys motion capture 6DOF rigid body setup

6.6 Testing scenarios

A total of 5 testing scenarios were chosen to test the comparison between the ESKF, NLO and EKF. Three of the tests were used with the "above water" Qualisys system, while the other two were used with the underwater Qualisys camera system. Since the Qualisys system could only give accurate and consistent 6 DOF in their respective measurement volume, some of the tests had to be adjusted. One important notice, is that none of the tests used the GNC system during testing with the reason being a hardware bug in the thruster configuration. This was hard to fix without having any people from the Vortex hardware team. All of the testing scenarios were then manually done. For the "above water" tests, a person in wet-suit moved the AUV while another person was giving tether cable and communicating with a person in the control room, who kept track of the Qualisys consistency, at the same time. This is clearer depicted in figure 6.12.

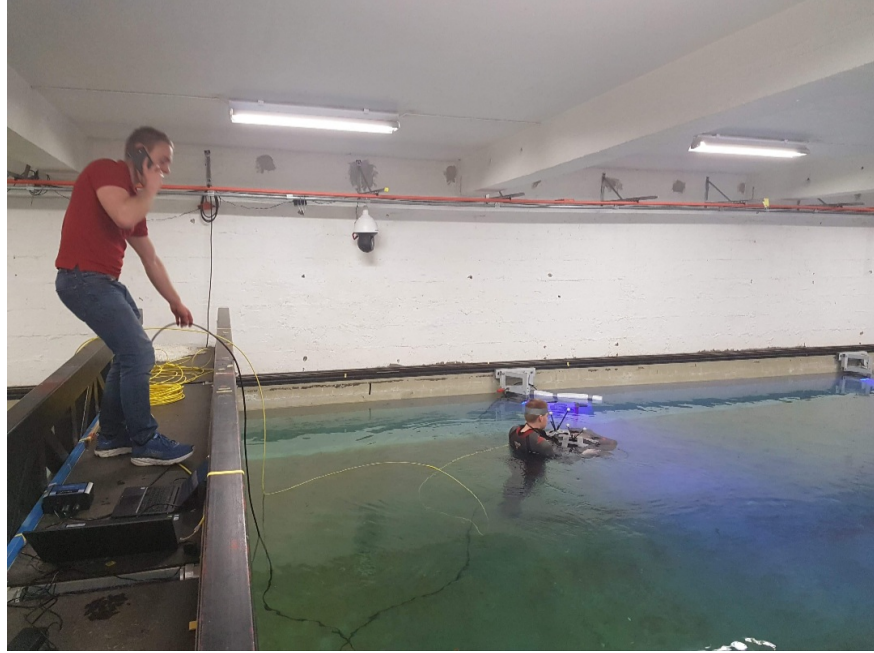


Figure 6.12: Manual testing procedure

For the underwater tests, another procedure was used. Now a added "stick" on top of the AUV was installed. This was done in order to carry out the maneuvers for the underwater tests. The installed stick is shown in the figure 6.13. The capture procedure was then manually done by having one person standing on the movable platform, one moving the platform and one keeping track of the Qualisys 6 DOF consistency.



Figure 6.13: Manta-2020 stick installment

6.6.1 Underwater testing scenarios

The first underwater testing scenarios was chosen to be a $L \times B = 3.0 \text{ m} \times 2.5 \text{ m}$ square with a depth of $z = -0.5 \text{ m}$ and a total of 8 waypoints. This test was originally planned with a variable z -value of $z = -0.5 \text{ m}$ to $z = -1.0 \text{ m}$, as shown in figure 6.14 However this was a hard test to accomplish, because of the inconsistency in the rigid body frame for the Qualisys underwater camera system. This was due to the cameras not having three or more tracker balls in their vision. The test was then redefined with a constant z -value of $z = -0.5 \text{ m}$, depicted in figure 6.15 This testing scenario was chosen because of the high accuracy in the 6 DOF tracking of Qualisys and how well the estimators tracked surge, sway, heave and yaw. This test was also a realistic maneuvering scenario for the AUV. Pitch and roll was tried to be kept as low as possible. The second underwater testing scenario was set to be a "sinus wave". With start on the water surface and moving -0.5 m in depth and 0.5 m in surge on each waypoint with a total of 11 waypoints and $L \times D = 3.0 \text{ m} \times 0.5 \text{ m}$. This testing scenario is seen in the figure 6.16. This testing scenario was chosen because it resembled a good comparison between the state estimators, and how well they could handle depth variations, and also see how the pressure-sensor worked.

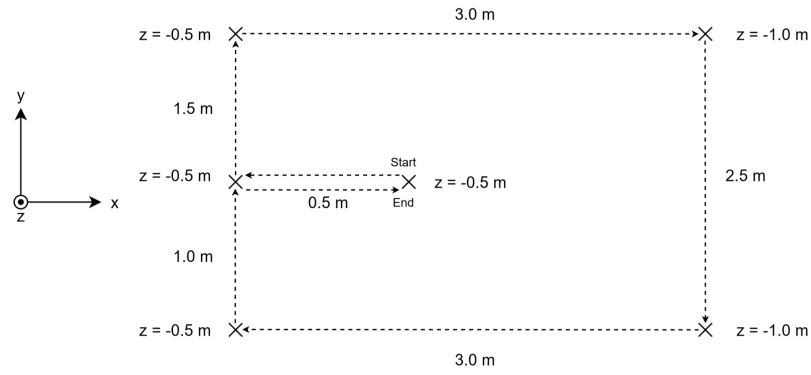


Figure 6.14: Underwater planned square-testing scenario

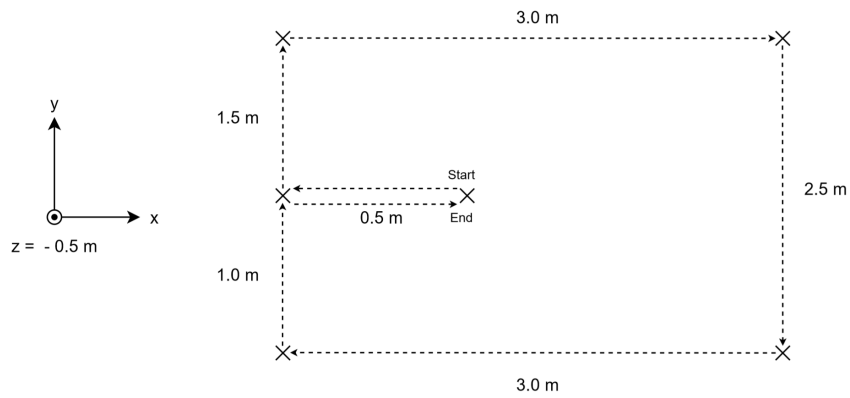


Figure 6.15: Underwater performed square-testing scenario

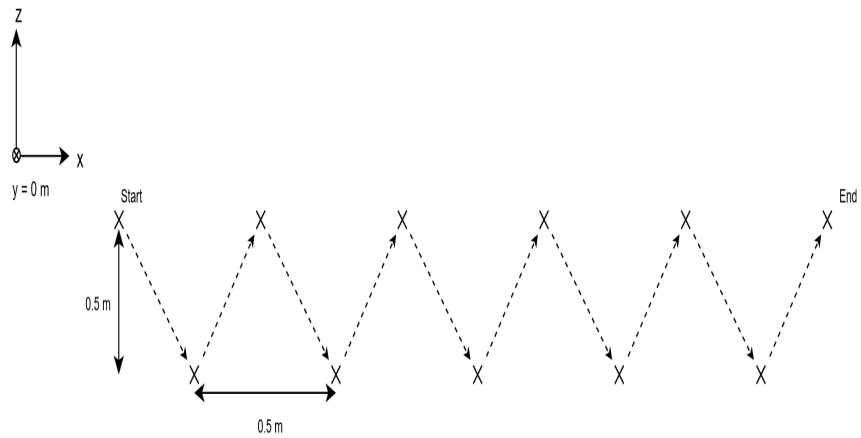


Figure 6.16: Underwater "sinus-wave" testing scenario

6.6.2 "Above water" testing scenarios

The "above water" testing scenarios were set to be a $L \times B = 6\text{ m} \times 3\text{ m}$ square, $L \times B = 5\text{ m} \times 3\text{ m}$ 30 min lap square and a "eight numbered" test with a horizontal and vertical distance of 3 meter apart from each waypoint. All of these tests are depicted in the figures 6.17, 6.18 and 6.19 respectively. The short-square test, was chosen mostly to set up a "method" of how the 30 min square test should be performed. Because of this no sensor measurements were recorded during the test-square. This would have been the backup test if the 30 min test showed to be infeasible. This was because it had to be done manually. The 30 min lap - square test was chosen to see the effect from the bias estimation from the ESKF and NLO and compare this to the EKF.

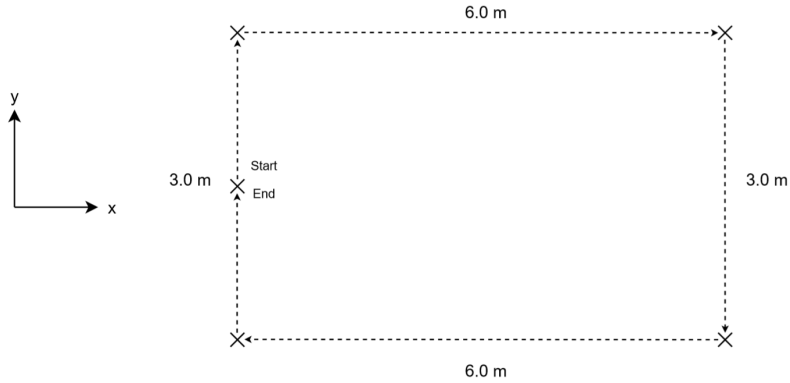


Figure 6.17: "Above water" square testing scenario

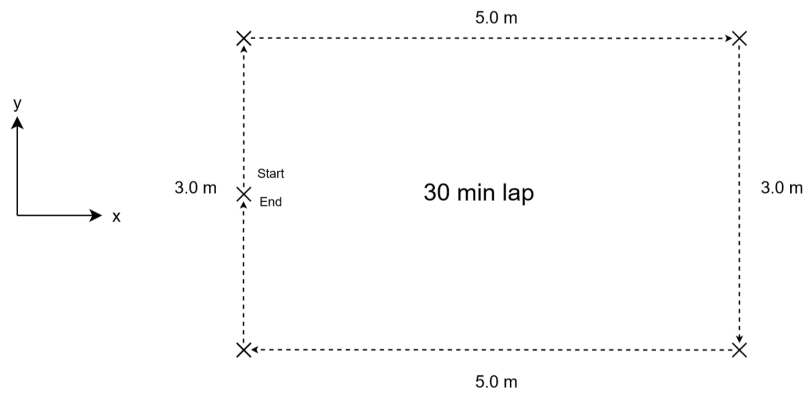


Figure 6.18: "Above water" square 30 min lap testing scenario

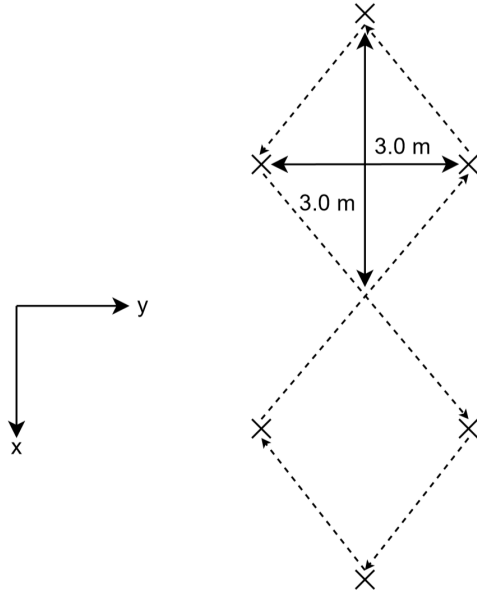


Figure 6.19: "Above water" eight number testing scenario

6.7 Recording and data gathering

The 6 DOF data from the Qualisys motion capture system was gathered by using the QTM software from one of the computers in the MC-lab control room. For both the "above water" and underwater system, the recorded data was stored in a *.mat* file with body frame NED position and (XYZ) Euler angles data. The recording frequency was also for both systems set to 125 hz to match the frequency output of the ESKF, NLO and IMU. This meant that no extrapolation or interpolation of the Qualisys data had to be done, which could have lead to more inaccurate comparisons. The only 6 DOF tracking parameter QTM offered was the bone length tolerance. This is the maximum separation between the lengths of the corresponding bones in a rigid body definition and a measured rigid body [46]. The default value for this value was $5mm$, and a higher value meant that the accuracy in attitude calculations lowered. This value had to be adjusted to $60mm$ and $70mm$ for the "above water" and underwater systems respectively. At these values the Qqus cameras could define a consistent 6 DOF rigid body frame.

QTM settings					
System	Frequency	Data	Tracking parameters	Value	Storage format
"Above water"	125 hz	NED{x,y,z},(XYZ) Euler angles	Bone length tolerance	60 mm	.mat {struct}
Underwater	125 hz	NED{x,y,z},(XYZ) Euler angles	Bone length tolerance	70 mm	.mat {struct}

Table 6.1: Qualisys tracking manager setup

The measurements coming from the DVL, IMU and pressure sensor were recorded with rosbags with their respective rostopic frequency. Specifically the rosbags gathered \mathbf{a}_{imu}^{acc} , ω_{imu}^{gyro} , $z_{b/w}^{pressure}$ and $\mathbf{v}_{bottom/b}^{dvl}$, as depicted

in figure 6.20. The data from the rosbags was then used directly to the NLO, ESKF and EKF.

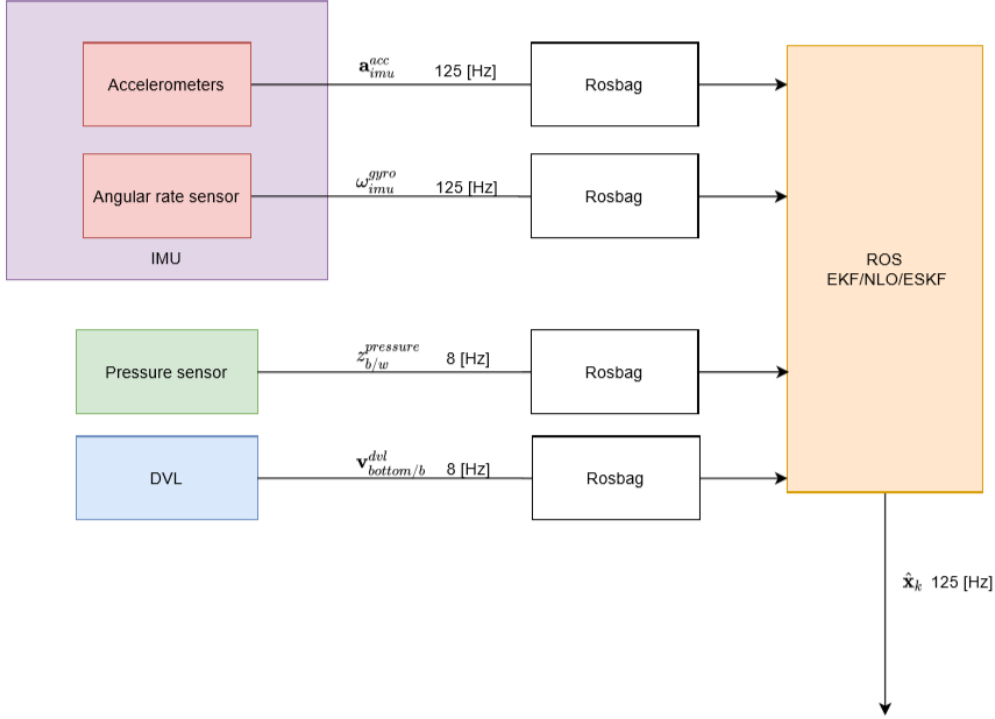


Figure 6.20: NLO, EKF and ESKF Sensor data gathering method

6.8 Tuning of filter parameters

For the AUV to run the testing scenarios satisfactory, tuning of the measurement noise covariances had to be done for the ESKF, NLO and EKF. In order to compare the filters, the same tuning parameters for the IMU, DVL and pressure sensor were set. The end results of the tuning parameters can be seen in section 6.9.7.

6.8.1 The EKF and NLO continuous time process noise covariance \mathbf{D}

The continuous time process noise covariance \mathbf{D} was set to match the process noise of the NLO for the corresponding states. Otherwise the default values of the in the robot_localization was used. The resulting \mathbf{D} matrix for the EKF was set as follows.

$$\mathbf{D}_{\text{ekf}} = \text{diag}(0.0001, 0.0001, 0.0001, 0.03, 0.03, 0.06, 0.001, 0.001, 0.001, 0.01, 0.01, 0.02, 0.01, 0.01, 0.015) \quad (6.1)$$

and for the NLO as follows.

$$\mathbf{D}_{\text{nlo}} = \text{diag}(0.0001, 0.0001, 0.0001, 0.001, 0.001, 0.001) \quad (6.2)$$

6.8.2 Tuning the gyros and accelerometers

The gyro and accelerometers measurement noise ($\sigma_{w_{gyro}^b}^2$, $\sigma_{w_{acc}^b}^2$) and bias driving noise variances ($\sigma_{b_{gyron}^b}^2$, $\sigma_{b_{accn}^b}^2$) were found using the Allan variance method described in section 3.1.4. The Allan variance plot for the gyroscopes and accelerometers were given in the STIM 300 datasheet. Thus the measurement noise variances $\sigma_{w_{acc}^b}^2$, $\sigma_{w_{gyro}^b}^2$, $\sigma_{b_{accn}^b}^2$ and bias noises $\sigma_{b_{gyron}^b}^2$ could be found.

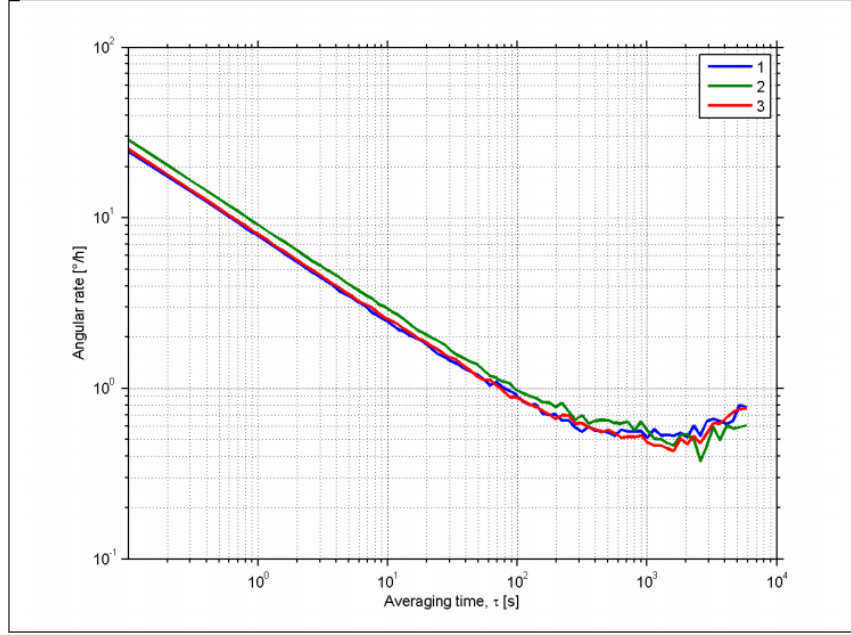


Figure 6.21: The allan variance of the gyros

	Bias instability	Angle Random Walk
X-axis	0.4°/h (at 1000 s)	8 °/√h
Y-axis	0.28°/h (at 1150 s)	8 °/√h
Z-axis	0.32°/h (at 1070 s)	9 °/√h

To find $\sigma_{w_{gyro}^b}^2$ the following equation is used.

$$\sigma_{w_{gyro}^b}^2 = \left(\frac{\pi}{180 * 60} \left(\frac{8 + 8 + 9}{3} \right) \right)^2 = 5.8761 * 10^{-6} \quad (6.3)$$

$\sigma_{b_{gyron}^b}^2$ is found as follows

$$\sigma_{b_{gyron}^b}^2 = \left(\frac{\pi}{180 * 3600 * \sqrt{\frac{1000+1150+1070}{3}}} \left(\frac{0.4 + 0.28 + 0.32}{3} \right) \right)^2 = 2.4331 * 10^{-15} \quad (6.4)$$

To find $\sigma_{w_{acc}^b}^2$ and $\sigma_{b_{accn}^b}^2$ the following accelerometer allan variance plot is used.

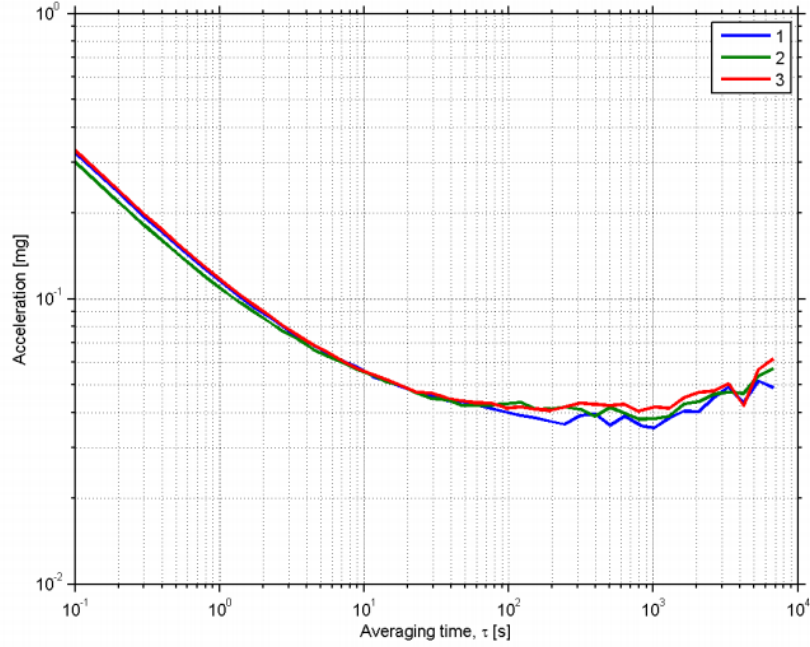


Figure 6.22: The allan variance of the accelerometers

	Bias instability	Angle Random Walk
X-axis	0.027 mg (at 1000 s)	0.12 mg/ \sqrt{h}
Y-axis	0.029 mg (at 800 s)	0.13 mg/ \sqrt{h}
Z-axis	0.031 mg (at 800 s)	0.13 mg/ \sqrt{h}

Using the same method as above $\sigma_{w_{acc}^b}^2$ and $\sigma_{b_{accn}^b}^2$ are found to be

$$\sigma_{w_{acc}^b}^2 = 3.4531 * 10^{-5} \quad (6.5)$$

$$\sigma_{b_{accn}^b}^2 = 6.2345 * 10^{-6} \quad (6.6)$$

6.8.3 Tuning the DVL

According to the data-sheet of the DVL, the variance $\sigma_{w_{dvl}^b}^2$ of the Gaussian measurement noise \mathbf{w}_{dvl} could be found from the instantaneous Doppler noise, which the data-sheet called figure of merit (FOM) [9]. This FOM value was scaled to represent a standard deviation of the Gaussian measurement noise of the bottom velocity tracking $\mathbf{v}_{b/bottom}^{dvl}$ and could be used directly as a measurement covariance \mathbf{R} in a Kalman filter or any statistical state observer. Since the FOM value gave instantaneous Doppler noise, the measurement noise $\mathbf{R}(t)$ changed at each cycle. These FOM values could be set directly in the covariance matrix in the driver of the DVL. Thus the measurement covariance matrix were set as follows for the NLO, ESKF and EKF.

$$\mathbf{R}_{dvl} = blkdiag(FOM_x^2, FOM_y^2, FOM_z^2) \quad (6.7)$$

One interesting note here is since this is a time varying measurement noise with instantaneous Doppler noise, the DVL will give accurate measurement covariances continuously, and thus means that in environments where the DVL measurements are unreliable, the covariances will be higher, and the state estimators can trust their prediction more.

6.8.4 Tuning the pressure sensor

The measurement noise variance $\sigma_{w_{pressure}^w}^2$ where tuned as following for the all state estimators.

$$\sigma_{w_{pressure}^w}^2 = 2.5125 \quad (6.8)$$

A more optimal method would have been to measure the variance by the definition of the variance. By calibrating the pressure sensor at land such that it at average gives a mean of 0 and then collect a set of data, setting a threshold for wild points, and then calculate the sample variance for the given data.

6.8.5 Tuning the Gauss-Markov bias model for ESKF

The acceleration and gyro bias reciprocal time constants ($p_{b_{acc}^b}, p_{b_{gyro}^b}$) were both initially set with a value of 1. This value tended to give discrete jumps and unsmooth behavior of the gyro and acceleration biases. By setting these to smaller values 10^{-3} ensured smooth changes of the biases.

6.8.6 Tuning of the k_1, k_2, k_i and M_{gyro} parameters for the NLO

For the NLO the tuning of the k_1 and k_2 were tuned to the following values.

$$k_1 = 0.1 \quad k_2 = 0.01 \quad (6.9)$$

Too large values of these seemed to make the NLO give unpredictable behavior with garbage like data.

For the k_i and M_{gyro} , it was tuned to the following.

$$k_i = 0.3 \quad M_{gyro} = 0.3 \quad (6.10)$$

Too large values of k_i seemed to render the gyro bias estimates with an non-smooth behavior. The tuning parameter M_{gyro} is the bound that restricts the gyro estimates. A low value of this was therefore preferred.

6.8.7 The resulting tuning parameters

Measurement covariances for EKF			
Description		Symbols	Values
Acceleration	\mathbf{R}_{acc}	$\sigma_{w_{acc}^b}^2 \mathbf{I}_{3x3}$	$3.4531 * 10^{-5} \mathbf{I}_{3x3}$
Gyro	\mathbf{R}_{gyro}	$\sigma_{w_{gyro}^b}^2 \mathbf{I}_{3x3}$	$5.8761 * 10^{-6} \mathbf{I}_{3x3}$
DVL	\mathbf{R}_{dvl}	$\text{diag}(FOM_x^2, FOM_y^2, FOM_z^2)$	changes on each cycle
Pressure sensor	$r_{pressure}$	$\sigma_{w_{pressure}^w}^2$	2.5125

Measurement covariances for ESKF			
Description		Symbols	Values
Acceleration	\mathbf{V}	$(\sigma_{w_{acc}^b}^2 / \Delta t) \mathbf{I}_{3 \times 3}$	$(3.4531 * 10^{-5} / 8 * 10^{-3}) \mathbf{I}_{3 \times 3}$
Acceleration bias	\mathbf{A}	$2p_{b_{acc}^b} \sigma_{b_{acc}^b}^2 \mathbf{I}_{3 \times 3}$	$2 * 10^{-3} * 6.2345 * 10^{-6} \mathbf{I}_{3 \times 3}$
Gyro	Θ	$(\sigma_{w_{gyro}^b}^2 / \Delta t) \mathbf{I}_{3 \times 3}$	$(5.8761 * 10^{-6} / 8 * 10^{-3}) \mathbf{I}_{3 \times 3}$
Gyro bias	Ω	$2p_{b_{gyro}^b} \sigma_{b_{gyro}^b}^2 \mathbf{I}_{3 \times 3}$	$2 * 10^{-3} * 2.4331 * 10^{-15} \mathbf{I}_{3 \times 3}$
DVL	\mathbf{R}_{dvl}	$\text{diag}(FOM_x^2, FOM_y^2, FOM_z^2)$	changes on each cycle
Pressure sensor	$r_{pressure}$	$\sigma_{w_{pressure}^w}^2$	2.5125

Accelerometer and gyro bias reciprocal time constants for ESKF		
Description		Value
Acceleration	$p_{b_{acc}^b}$	10^{-3}
Gyro	$p_{b_{gyro}^b}$	10^{-3}

Measurement covariances for NLO			
Description		Symbols	Values
DVL	\mathbf{R}_{dvl}	$\text{diag}(FOM_x^2, FOM_y^2, FOM_z^2)$	changes on each cycle
Pressure sensor	$r_{pressure}$	$\sigma_{w_{pressure}^w}^2$	2.5125

Tuning parameters for the NLO		
Description		Value
Acceleration injection gain	k_1	0.1
Velocity injection gain	k_2	0.01
σ injection gain	k_i	0.3
Gyro bias bound	M_{gyro}	0.3

6.9 Initialization

Since the rosbags where gathered of the sensors of all test scenarios as seen in the figure 6.20, the state estimators could be run by playing back the rosbags with its sensor data. With this it was not required to have Manta-2020. To resemble the most accurate comparison between the estimators, the initial estimate $\hat{\mathbf{x}}_0$ of all the filters where initialized to its zero state for all test scenarios. For the ESKF and NLO which initialized with quaternions, this was set to $\mathbf{q} = [1, 0, 0, 0]^T$. The initial error covariance \mathbf{P}_0 of the filters, where also tuned to be equal to in their respective states. For the EKF the following initial error covariance was used.

$$\mathbf{P}_0 = \text{blkdiag}(0.001\mathbf{I}, \mathbf{R}_{gyro}, \mathbf{R}_{dvl}, \mathbf{R}_{gyro}, \mathbf{R}_{acc}) \quad (6.11)$$

for the ESKF

$$\mathbf{P}_0 = \text{blkdiag}(0.001\mathbf{I}, \mathbf{R}_{dvl}, \Theta, \mathbf{A}, \Omega) \quad (6.12)$$

and at last for the NLO

$$\mathbf{P}_0 = \text{blkdiag}(0.001\mathbf{I}, \mathbf{R}_{dvl}, 0.0012\mathbf{I}) \quad (6.13)$$

Results and discussion

7.1 Experimental results

This chapter outlines all the experimental results from the MC-lab at Tyholt. These results present the performance for each of the implemented state estimators, EKF, ESKF and NLO on Manta-2020. The results include a comparison of the PVA estimation of the state estimators and Qualisys, the PVA errors, the bias estimates of ESKF and NLO and the NIS consistency analysis. For each testing scenario, a discussion based on these will be presented. The order will first be to discuss the "above water" testing scenarios and then the underwater ones. First off will be the eight number, then the 30 min square test, with the underwater sinus test after that and at last the underwater square test.

7.1.1 "Above water" eight number

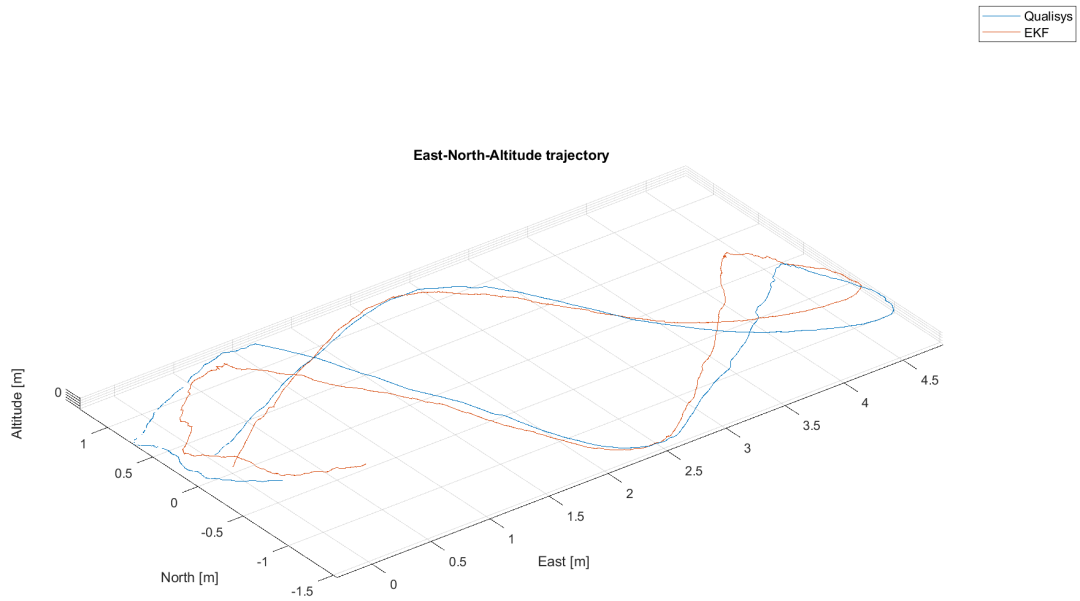


Figure 7.1: East-north-altitude 3D trajectory comparison between the EKF and Qualisys - eight number

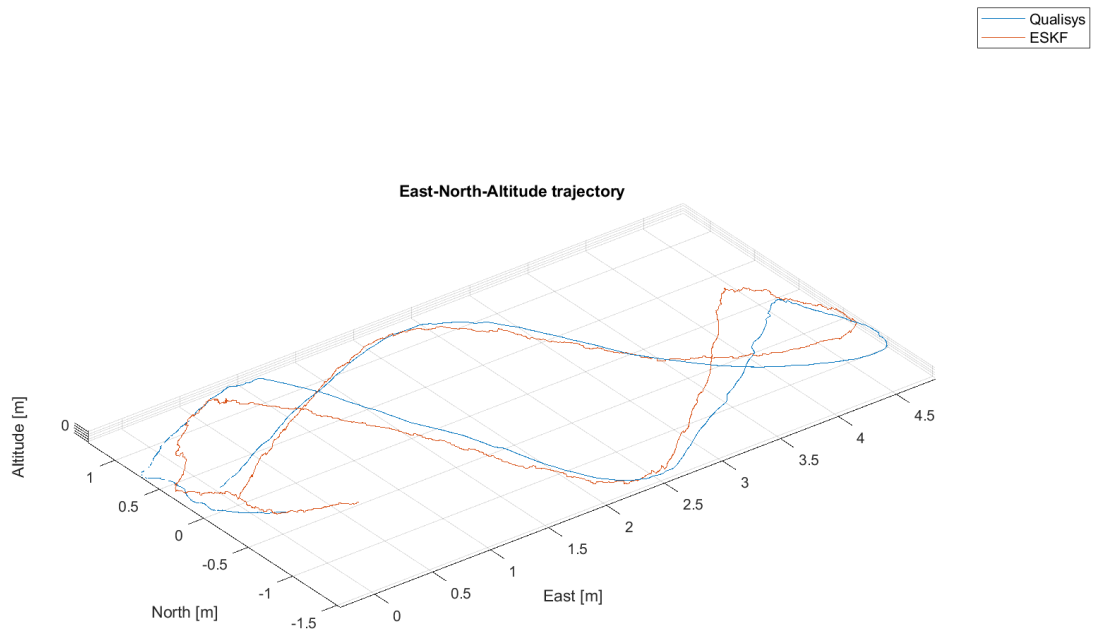


Figure 7.2: East-north-altitude 3D trajectory comparison between the ESKF and Qualisys - eight number

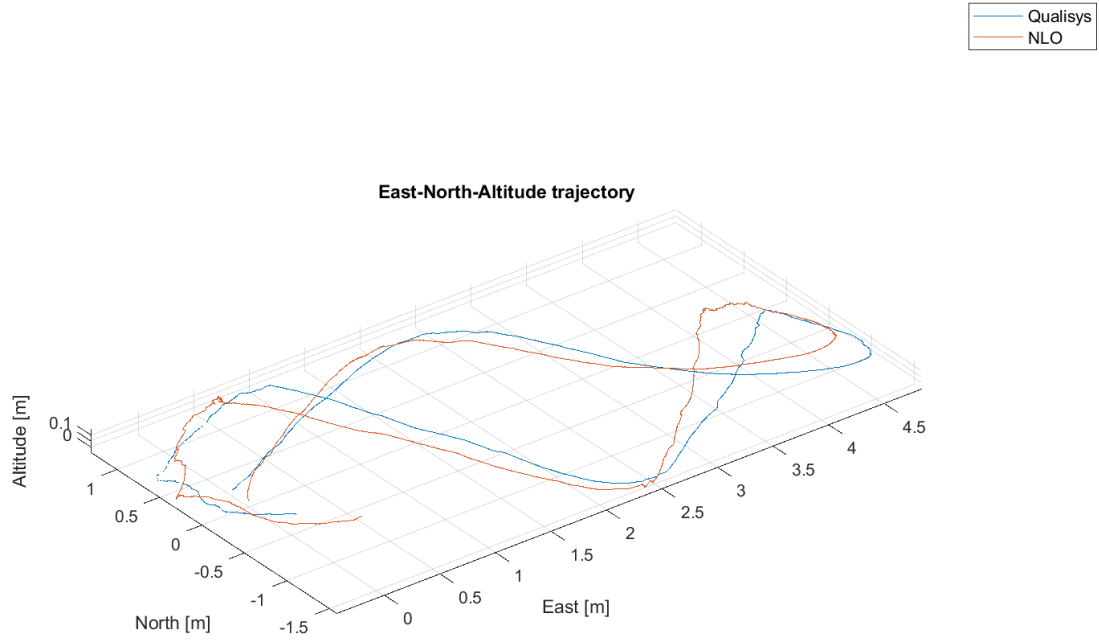


Figure 7.3: East-north-altitude 3D trajectory comparison between the NLO and Qualisys - eight number

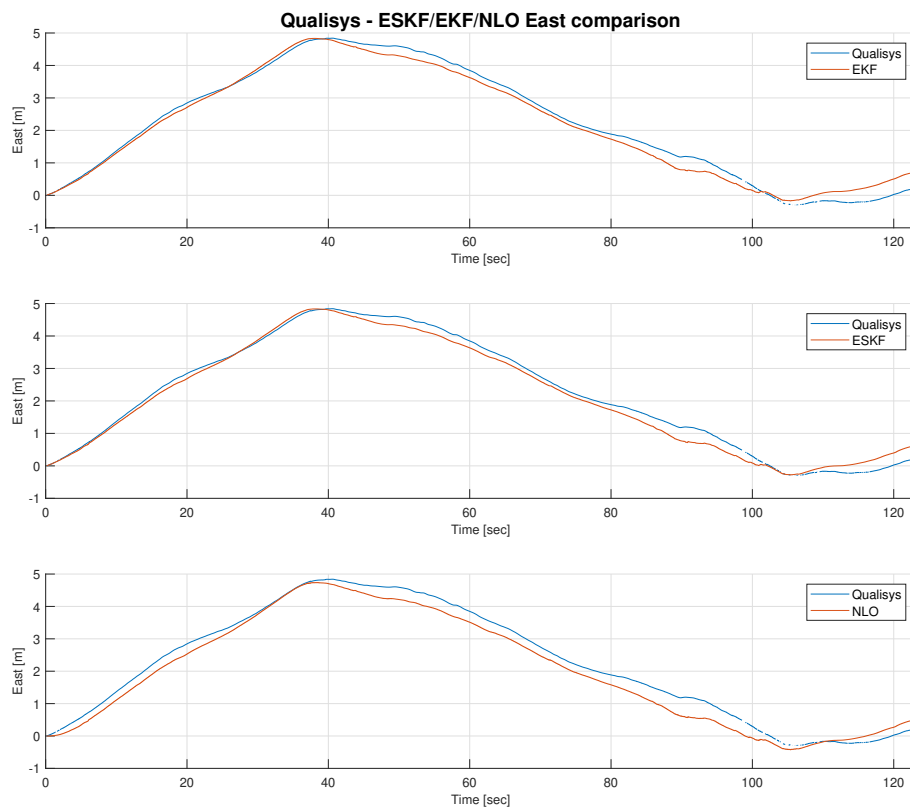


Figure 7.4: EKF/ESKF/NLO east trajectory comparison with Qualisys - eight number

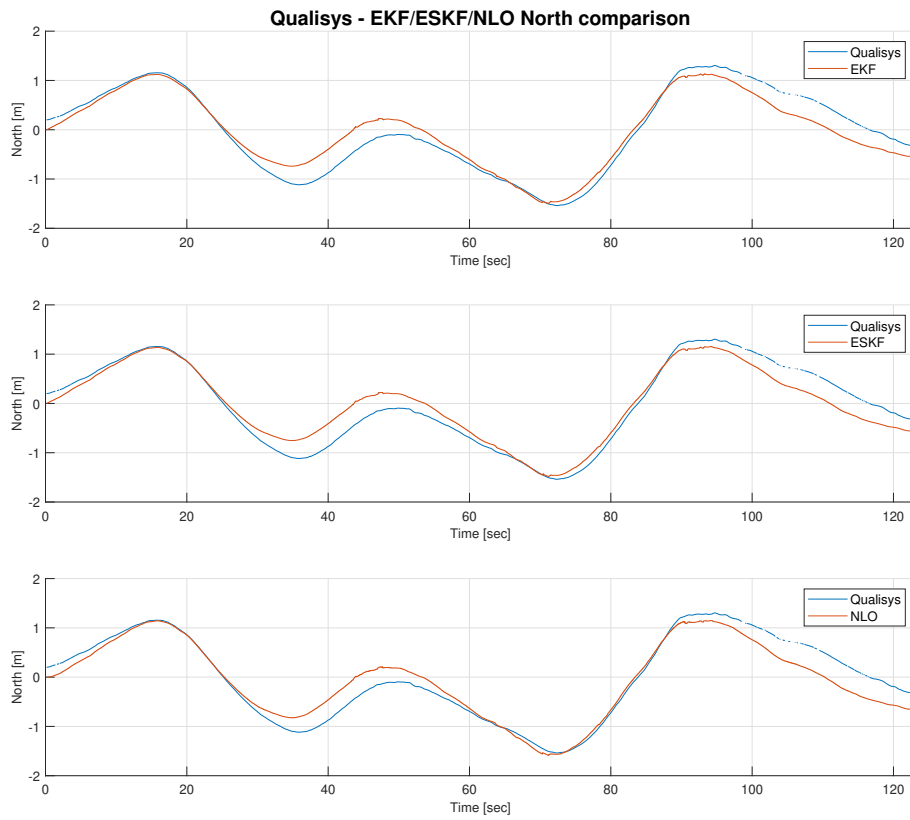


Figure 7.5: EKF/ESKF/NLO north trajectory comparison with Qualisys - eight number

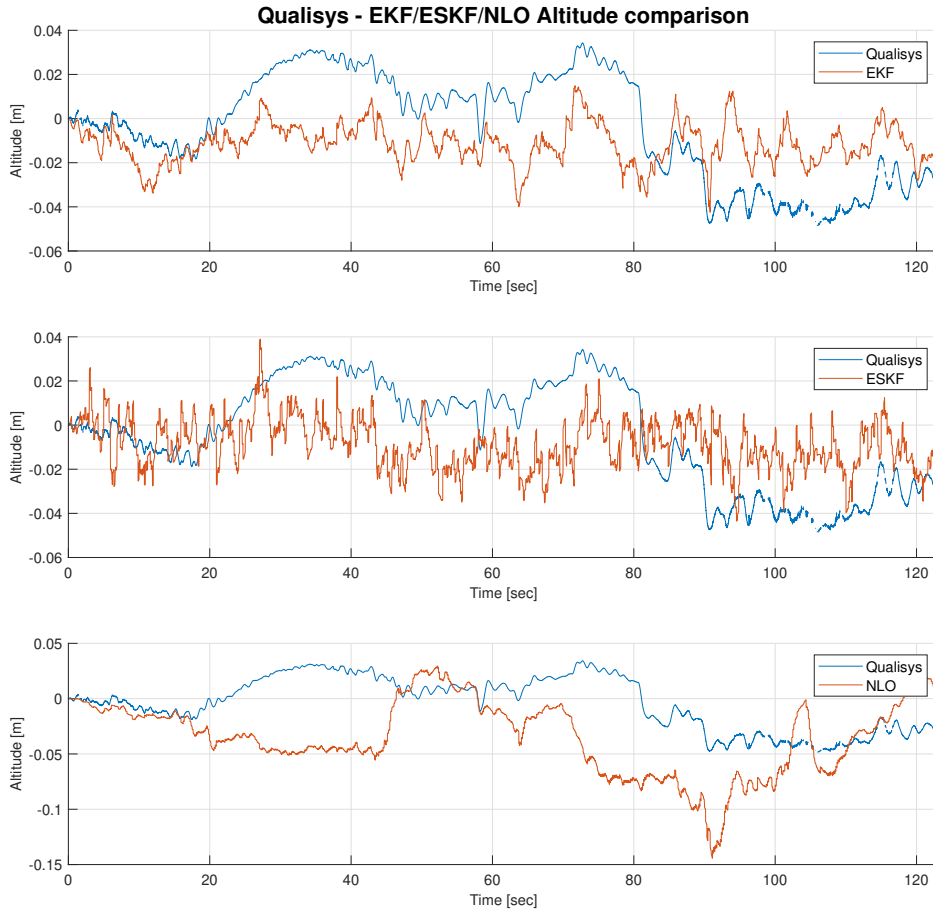


Figure 7.6: EKF/ESKF/NLO altitude trajectory comparison with Qualisys - eight number

Looking at the figures 7.1, 7.2 and 7.3 their East-north-altitude trajectory is represented with a closer look at their north, east and altitude comparison given in the figures 7.4, 7.5 and 7.6. These figures give a high overview of how the EKF, ESKF and NLO compares to Qualisys. By observation it is seen that all three estimators follow the Qualisys trajectory pretty well with a slight offset. This offset could be due to the initial alignment of the Qualisys world frame not matching the world frame of the AUV totally perfect or it could be the estimation itself. It is also seen that the ESKF has slightly more non-smooth behavior than the EKF and NLO. This can be verified by looking at altitude comparison in figure 7.6. Here the altitude position alters much between 0.03 m and -0.03 m . A reason for this could be that the ESKF uses the acceleration measurement coming from the IMU directly in the prediction step. When doing prolonged prediction with IMU only, it will as soon as possible drift. This is especially true if the gravity vector is not calculated or estimated totally correct. When the filter then gets updated with an pressure measurement, it will update its estimate most of its time close to that measurement based on its measurement covariance. This could be the reason for the up-and-down behavior of the ESKF. The NLO also has the acceleration and gyro measurements used directly in the prediction, but it is adjusted by the injection term σ , which takes the velocity behavior into account. Furthermore when it comes to the NLO and EKF, they both have continuous time process noise covariance \mathbf{D} that is not dependent on the variance of the measurement noises of the gyro

and accelerometer $\sigma_{w_{gyro}^b}^2, \sigma_{w_{acc}^b}^2$ and the bias driving noise variances $\sigma_{b_{gyro}^b}^2, \sigma_{b_{acc}^b}^2$. With this their continuous time process noise covariance \mathbf{D} can be tuned independently from these variances. With the exact same continuous time process noise covariance with the EKF and NLO, it is observed that the EKF resembles a better estimate than the NLO. This could come from that the gain tuning parameters k_1 and k_2 where not tuned that well.

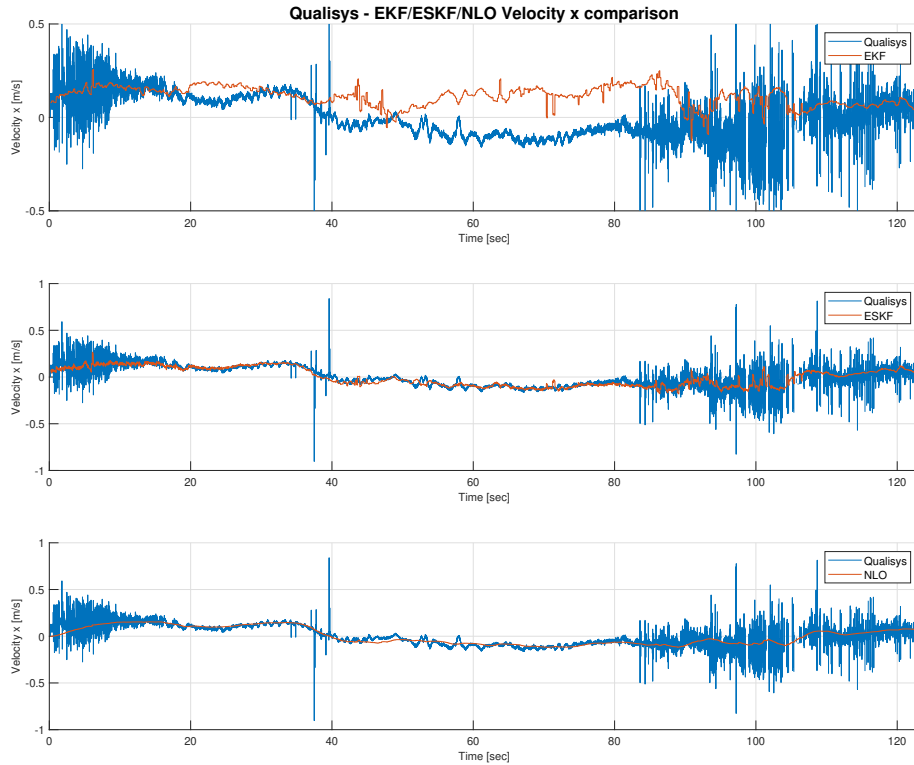


Figure 7.7: Velocity x comparison between the EKF, ESKF and NLO compared to Qualisys - eight number

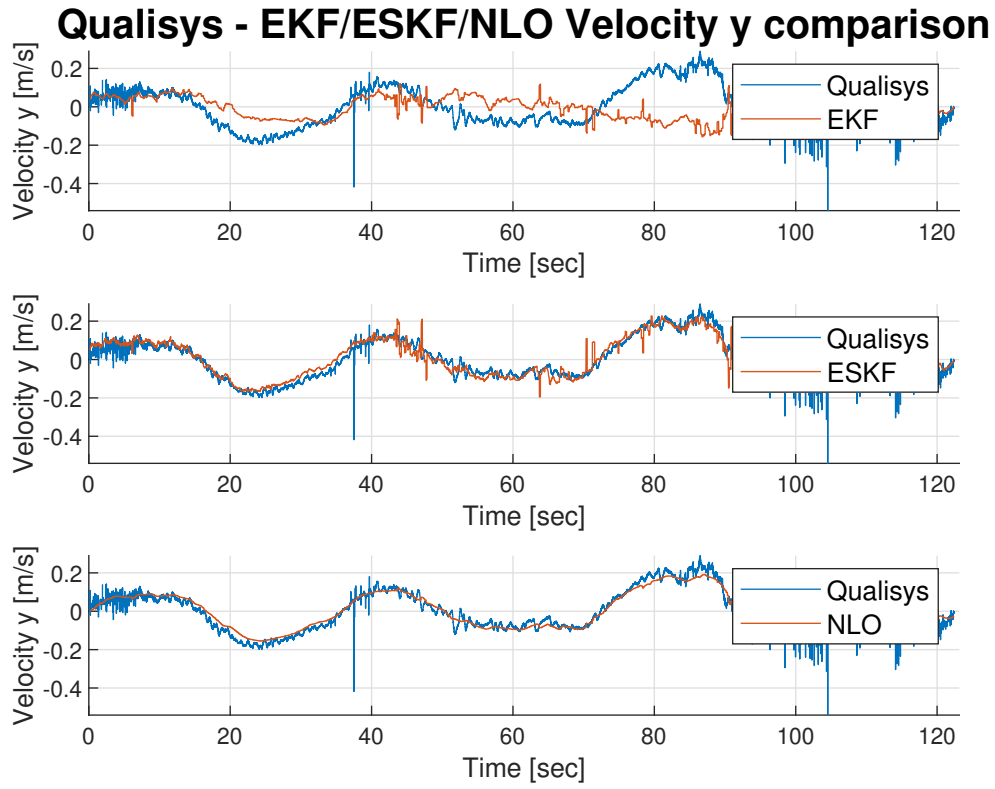


Figure 7.8: Velocity y comparison between the EKF, ESKF and NLO compared to Qualisys - eight number

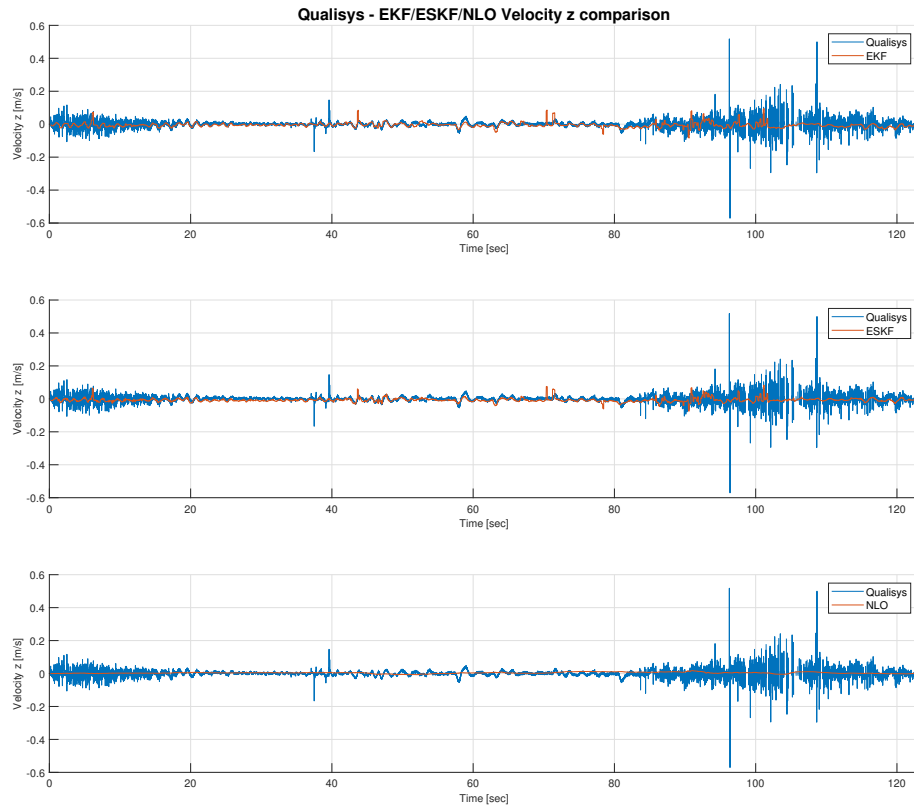


Figure 7.9: Velocity z comparison between the EKF, ESKF and NLO compared to Qualisys - eight number

Looking at the x and y velocities in the figures 7.7, 7.8, it is seen that the ESKF and NLO estimate the velocities of Qualisys very well, where the ESKF seems to have the best estimate compared to Qualisys. This is rather not that strange, because these states are directly observable by the DVL. But the behavior of the EKF seems a little strange. This could be because the velocity kinematic transition model of the EKF does not resemble the true dynamics of the AUV, or because the process noise is not correctly tuned. Looking at the z velocity figure 7.9 all of the state estimators estimates very close to Qualisys.

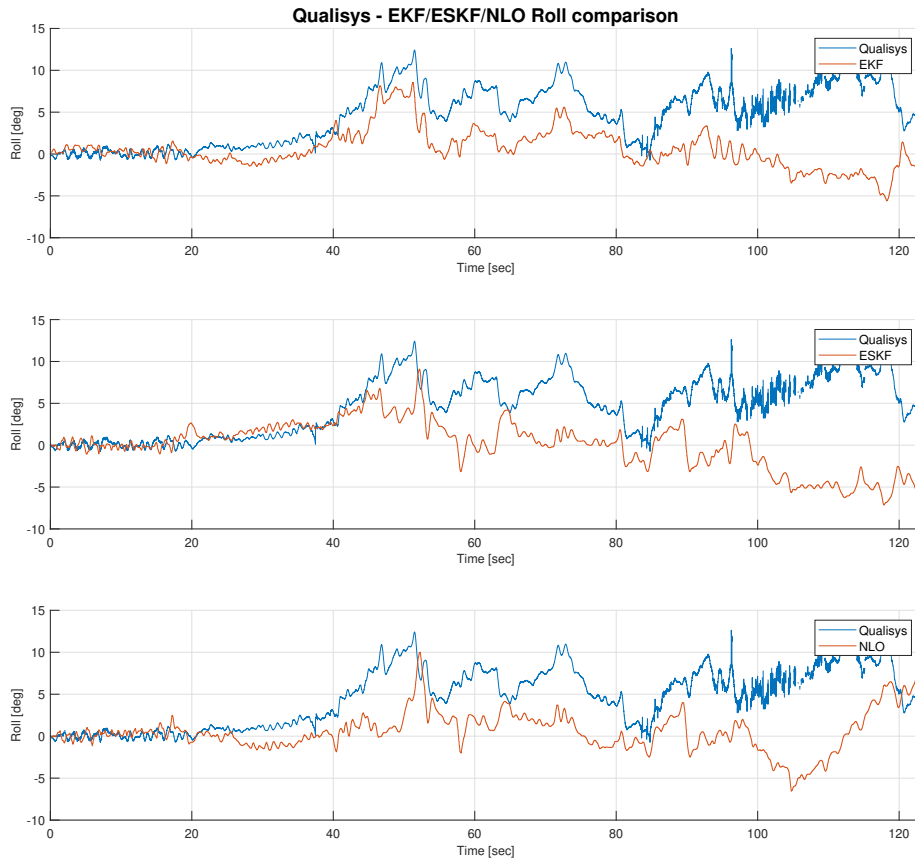


Figure 7.10: Roll comparison of EKF,ESKF and NLO compared to Qualisys - eight number

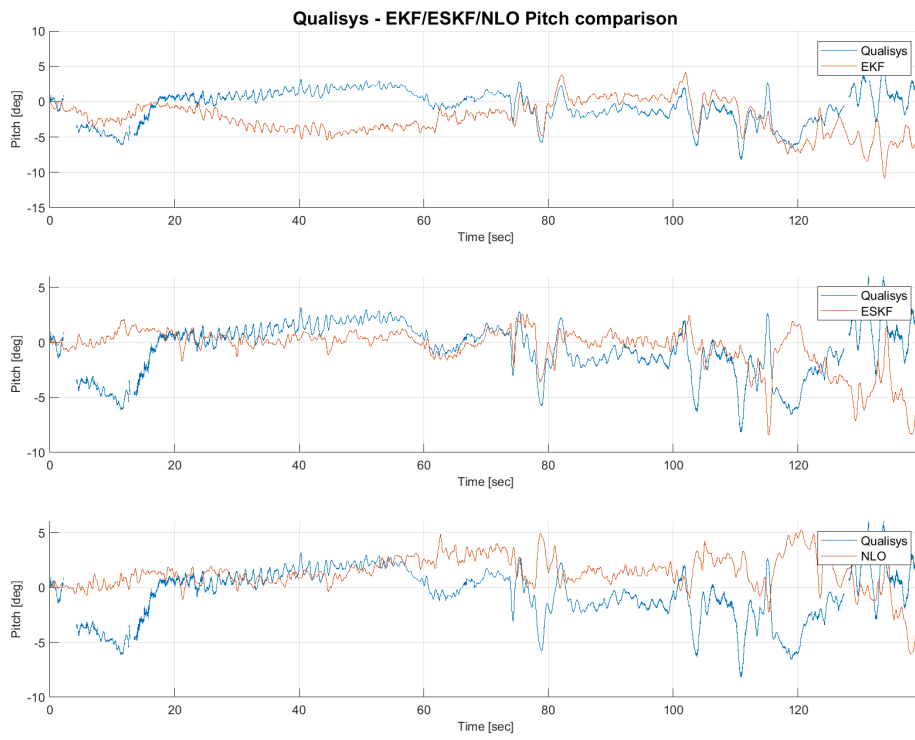


Figure 7.11: Pitch comparison of EKF,ESKF and NLO compared to Qualisys - eight number

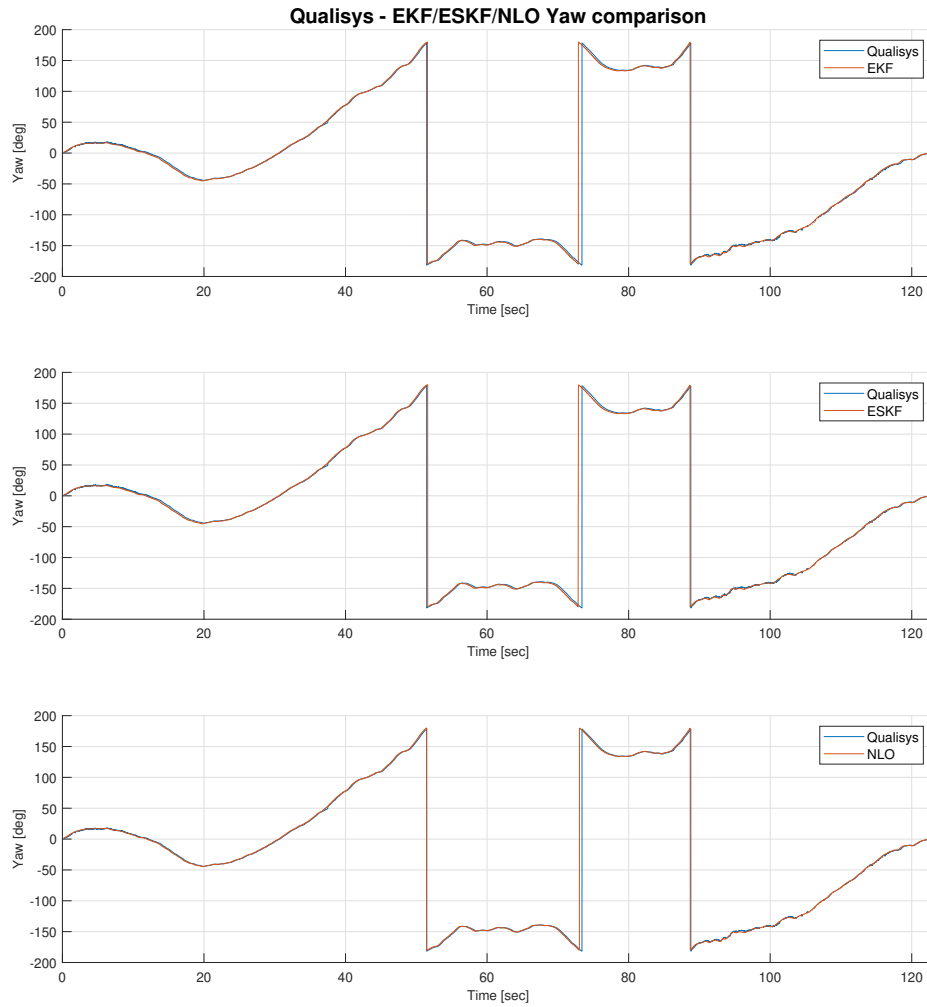


Figure 7.12: Yaw comparison of EKF,ESKF and NLO compared to Qualisys - eight number

Going over to the attitude comparison in the figures 7.10, 7.11 and 7.12 it is seen that the estimates of roll and yaw resembles the closest estimates compared to Qualisys, where the EKF seems to be the closest one. This seems a little strange because both of the ESKF and NLO have gyro bias estimation, where both the roll and pitch biases are observable because of the gravity vector of the accelerometer. This could be an implementation bug of the gyro bias in the ESKF and NLO by the author such that the roll and pitch estimates will produce errors, but then that would also have given corrupted yaw estimates, which is not the case as seen in figure 7.12. Another explanation could be that the kinematic prior model of the EKF resembles more the true behavior of the AUV, which will make the prediction closer to the true state of the AUV.

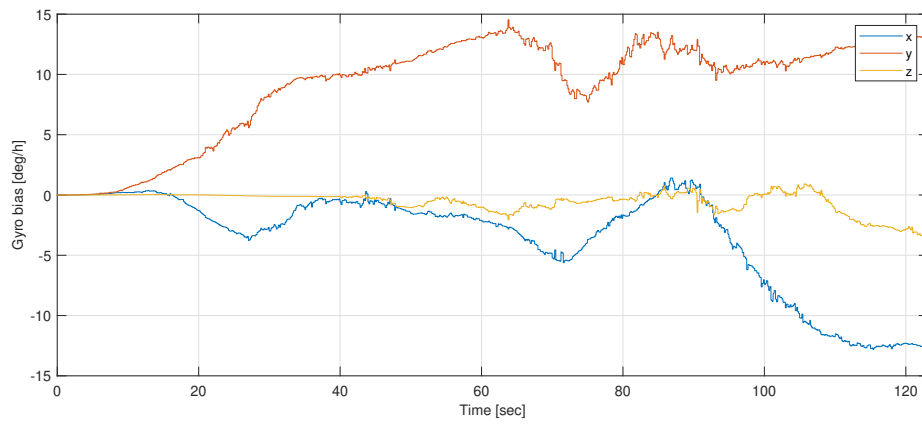
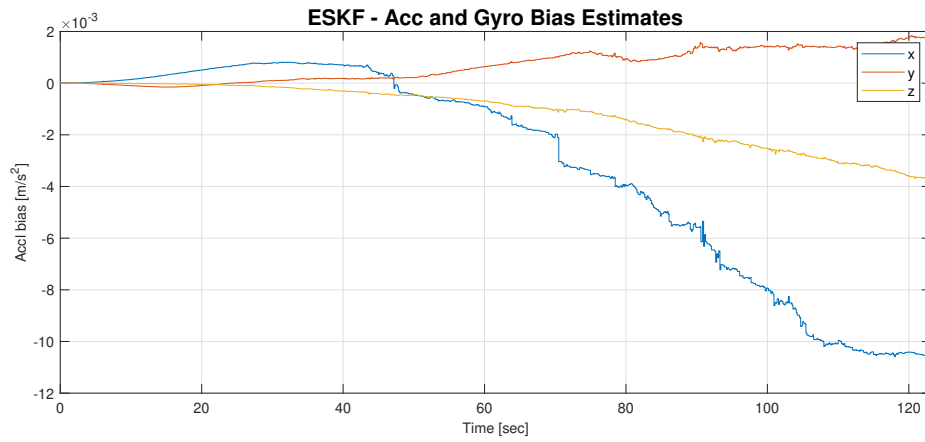


Figure 7.13: ESKF bias estimates - eight number

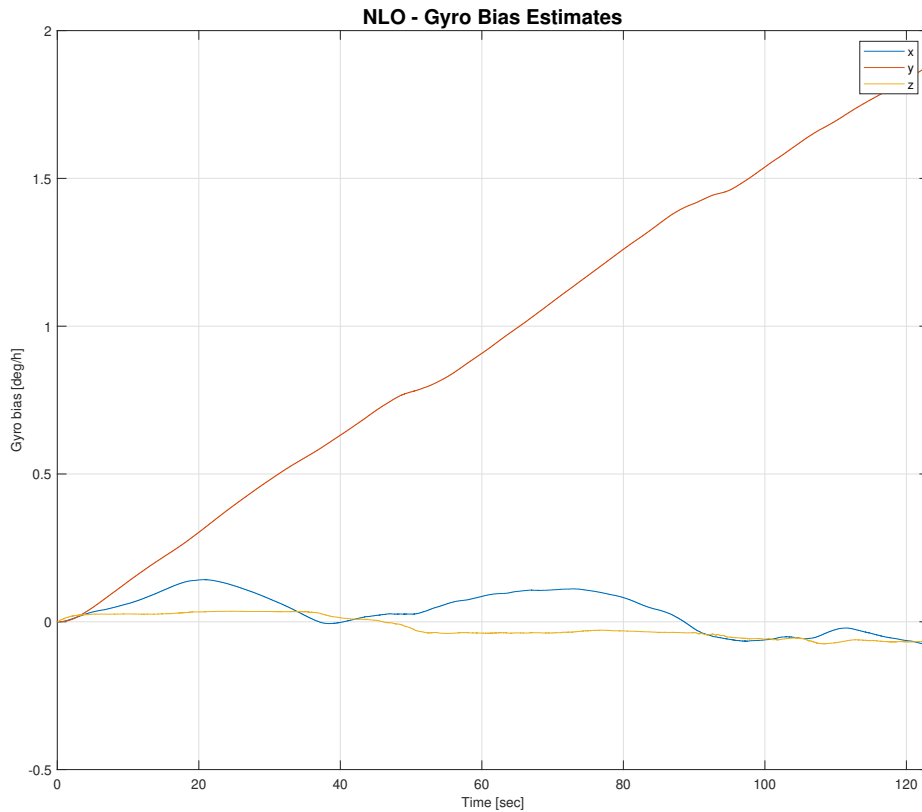


Figure 7.14: NLO bias estimates - eight number

Looking at the ESKF acceleration bias estimates in figure 7.13 it is seen that the x and y acceleration biases are non-observable, but the z-acceleration bias is observable. This is due to the pressure sensor giving absolute position in the world frame. To get observable acceleration bias in x and y, the need of for example a GNSS sensor is required. One would think that the DVL will render these biases observable, and it could have done so under specific maneuvers. But since it measures velocity in $\{b\}$ frame the rotation matrix $\mathbf{R}_b^n(\mathbf{q})$ must be calculated perfectly by the gyros. By the use of an inclinometer sensor that could measure roll and pitch together with for example a magnetometer that measures heading, this rotation matrix would be correctly estimated. This could have contributed in making these acceleration biases observable coming from a DVL sensor.

Looking at the gyro bias estimates of the ESKF and NLO in figure 7.13 and 7.14, it seems for the ESKF that the x and y gyro biases are non-observable, which is verified in the 30 min square test. This is rather strange, because these biases should be directly observable by having the accelerometer measuring specific force, using the gravity vector. This is most likely a bug in the implementation of the ESKF. For the NLO it seems that

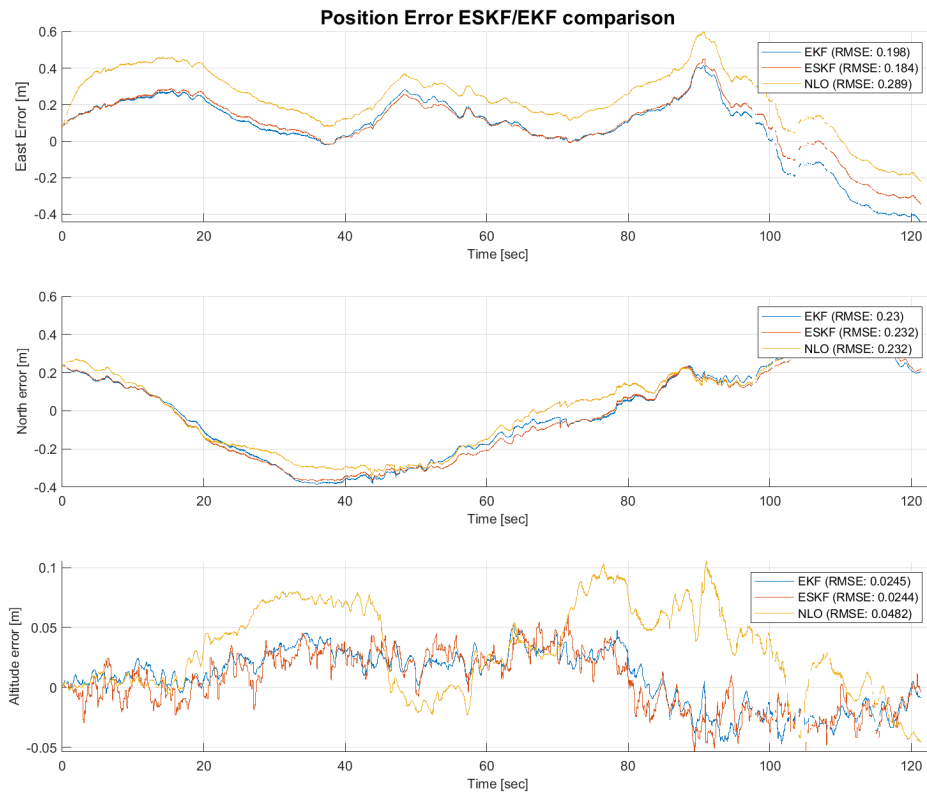


Figure 7.15: Position error of EKF,ESKF and NLO compared to Qualisys - eight number

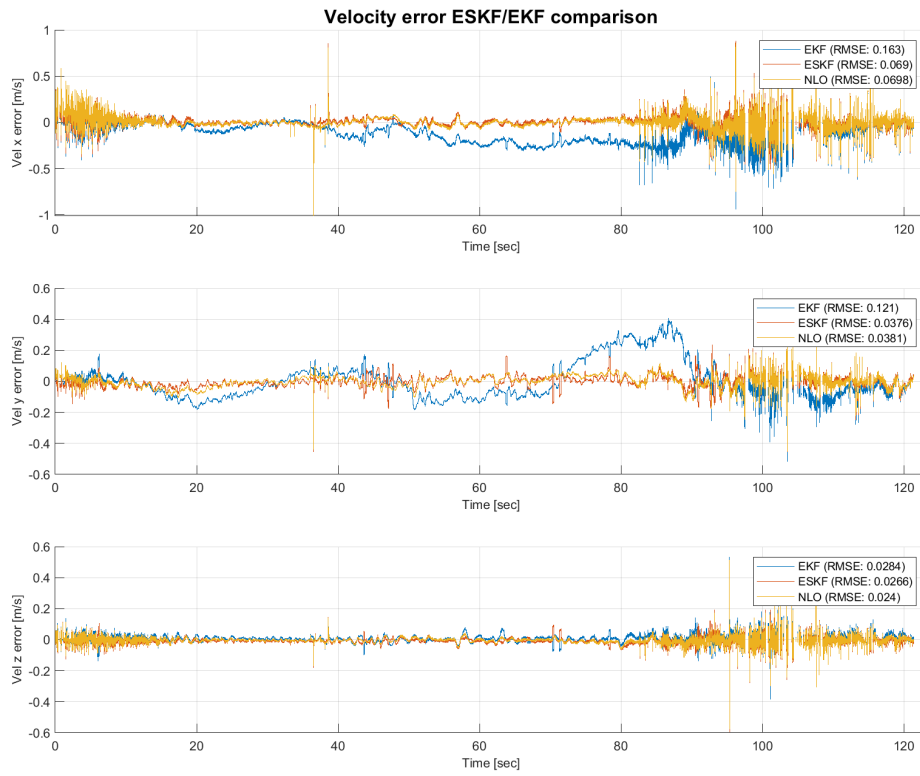


Figure 7.16: Velocity error of EKF,ESKF and NLO compared to Qualisys - eight number

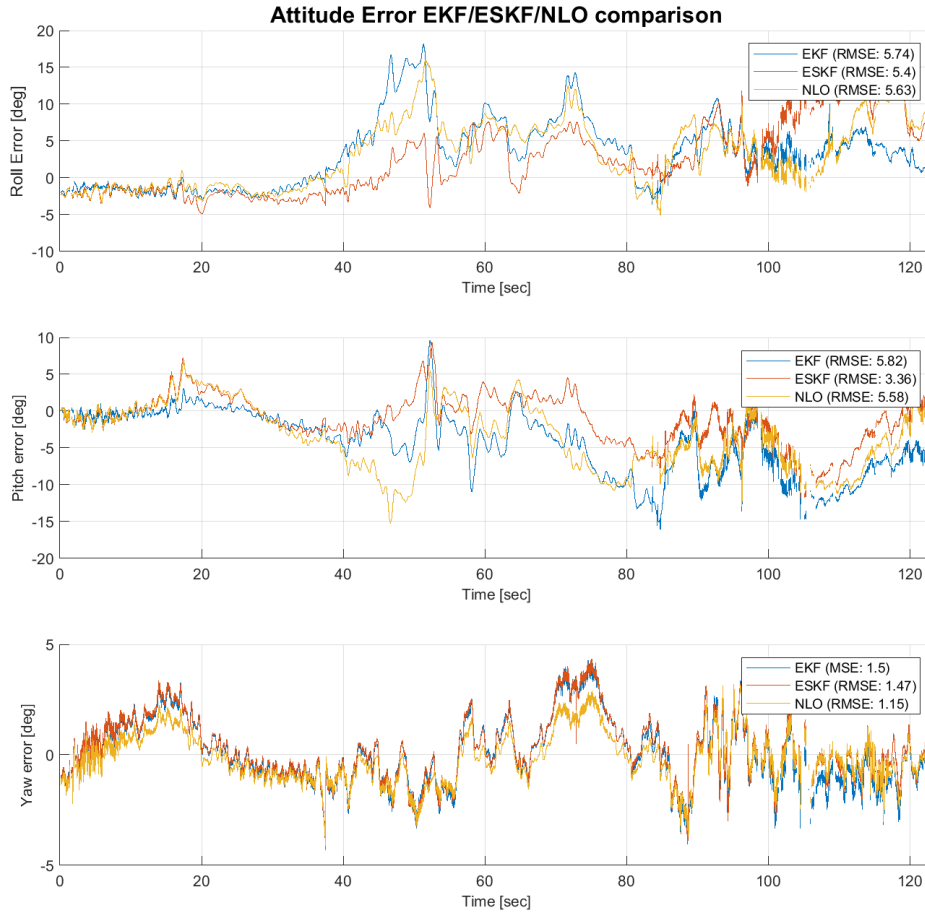


Figure 7.17: Velocity error of EKF,ESKF and NLO compared to Qualisys - eight number

Having discussed the position, velocity and attitude, their errors compared to Qualisys are also important to discuss. Looking at figure 7.15, 7.16 and 7.17 the errors compared to Qualisys are summarized with the RMSE for each state estimator. With these plots, we can find which state estimator that had the best overall performance compared to Qualisys.

Looking at the position errors, it is seen that the NLO performs the overall worst with an RMSE of 0.298, 0.232 and 0.0482 of east, north and altitude error respectively. This is the NLO has the largest error overall with a RMSE of 0.298, 0.232 and 0.0482. This could come from the tuning of k_1 and k_2 as discussed above. One interesting note here is that even when the ESKF had an alternating in their position estimates, it actually gives the most accurate position estimates overall with RMSE of 0.184, 0.232 and 0.0244 for the east, north and altitude error respectively. This was not expected. This accounts also for the velocity and attitude errors. Especially for the attitude errors, it is actually seen that the NLO and ESKF have smaller RMSE than the EKF even when it seemed to have larger errors on the attitude plots in the figures 7.10, 7.11 and 7.12.

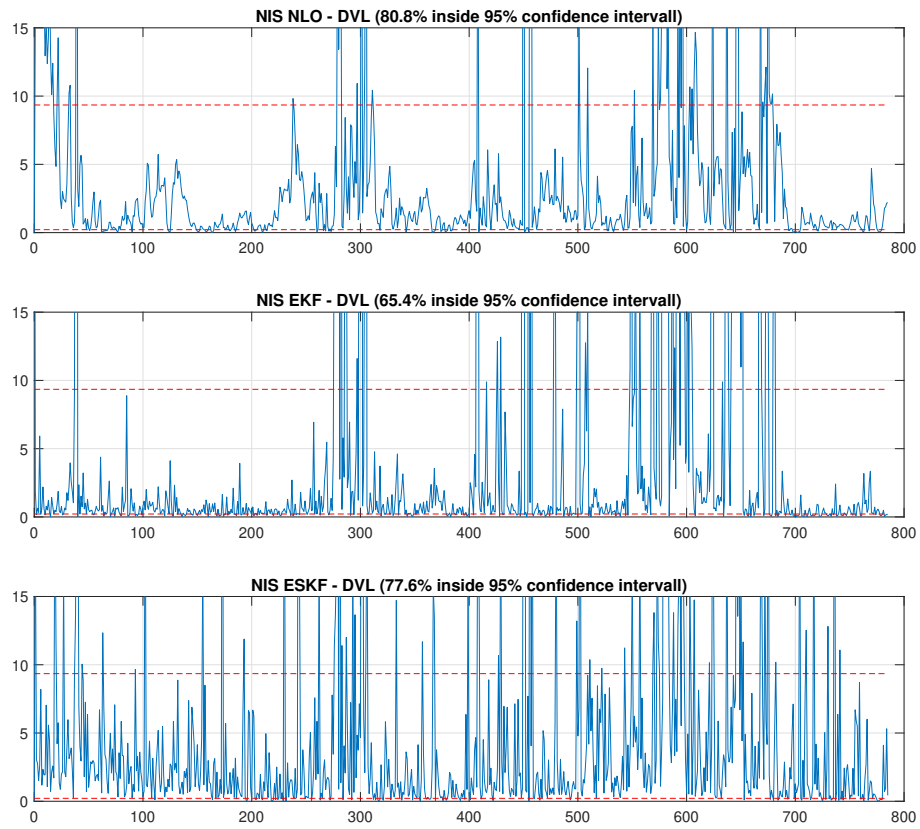


Figure 7.18: NIS DVL - EKF/ESKF/NLO comparison - eight number

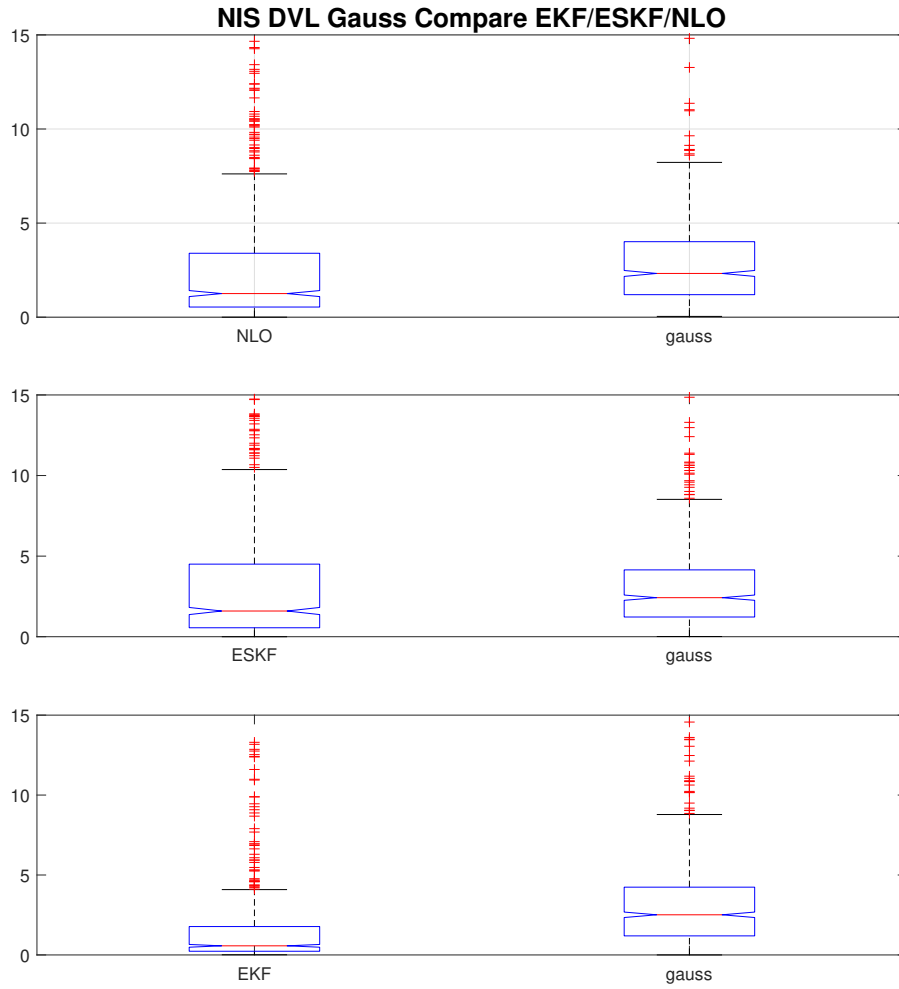


Figure 7.19: Gauss compare DVL - EKF/ESKF/NLO comparison - eight number

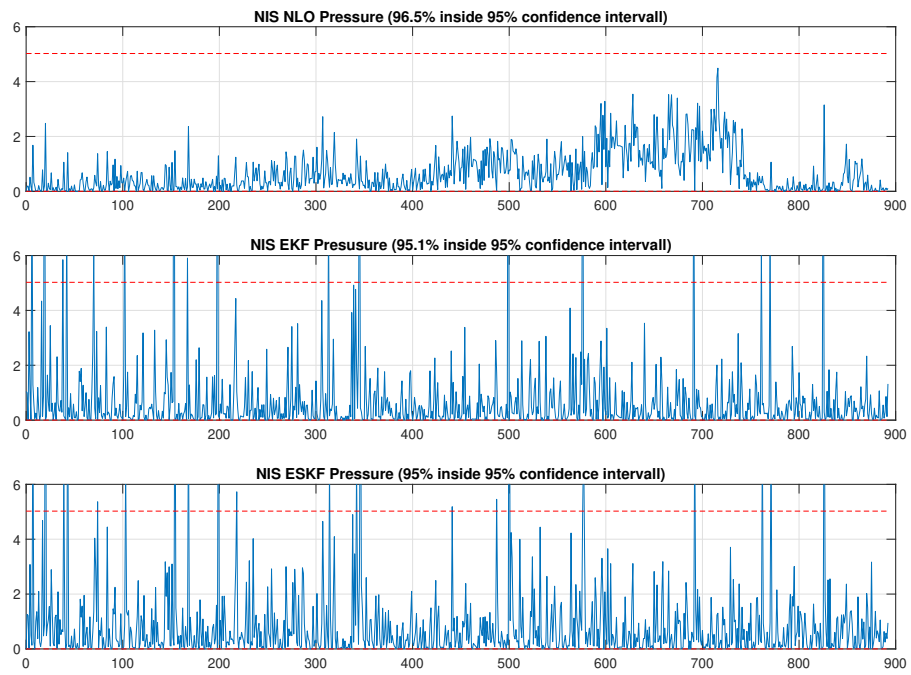


Figure 7.20: NIS Pressure - EKF/ESKF/NLO comparison - eight number

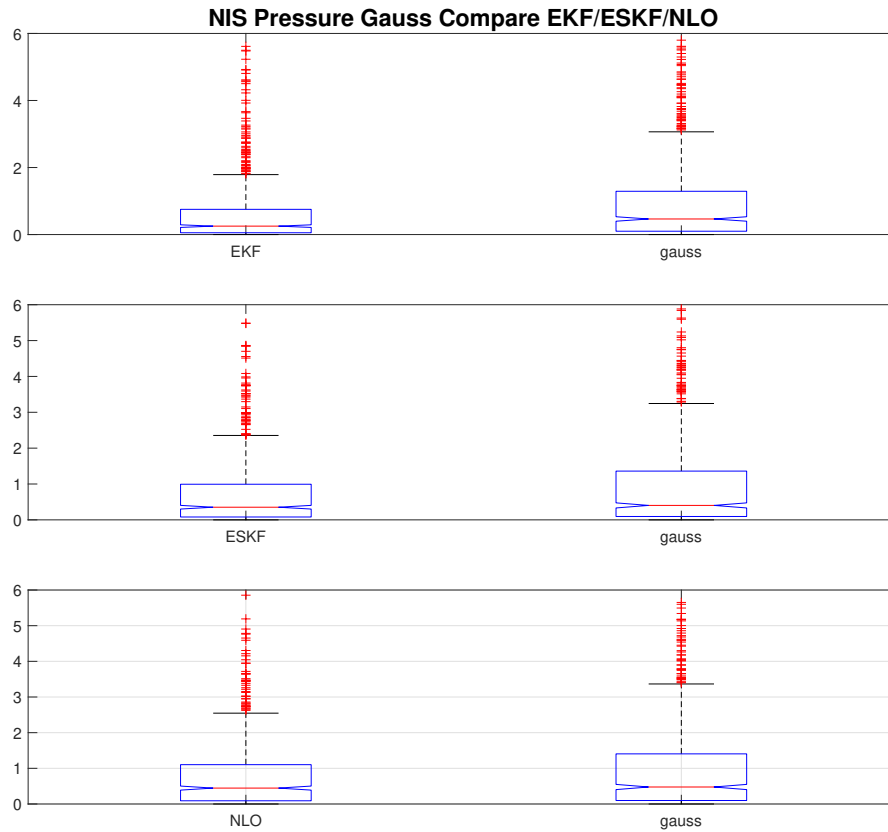


Figure 7.21: Gauss compare Pressure - EKF/ESKF/NLO comparison - eight number

In terms of filter consistency the NIS figures are represented in the 7.18, 7.19, 7.20 and 7.21. The plots of 7.18 and 7.20 represent the NIS values together with their respective upper and lower bound for the inverse chi-square distribution. The total percent of how many NIS values are inside this bound is also given. The figures 7.19 and 7.21 represent a box plot of the NIS values compared to series of the norm of a three dimensional zero-mean unit-variance Gaussian. The most optimal solution would be that the NIS test results resembles a gaussian distribution completely. This would mean that the estimation error is modelled as white noise, which indicates optimal estimation. For NIS/NEES tests results which have larger number of NIS values than its respective Gaussian distribution itself, can lead to divergence. As seen in the DVL Gauss comparison between the ESKF, EKF and NLO the EKF the NLO may seem to resemble the best comparison to the Gauss. The EKF will not diverge, but could have been better tuned with lower values in the DVL measurement covariance \mathbf{R} .

The NIS pressure tests on the other hand, seems very well tuned, where all of the state estimators have over 90 % inside their 95 % confidence interval. Compared to the Gauss all of them are close to the zero-mean Gauss. This indicates that the measurement covariance of the pressure sensor has been well tuned.

7.1.2 "Above water" 30 min square

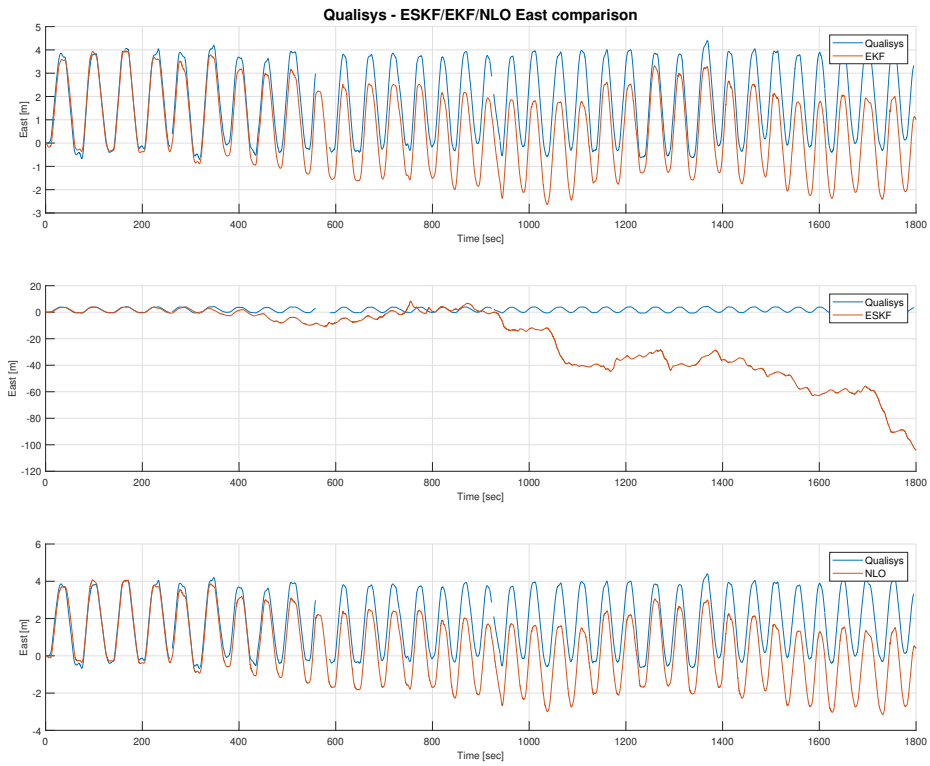


Figure 7.22: EKF/ESKF/NLO east trajectory comparison with Qualisys - 30 min square

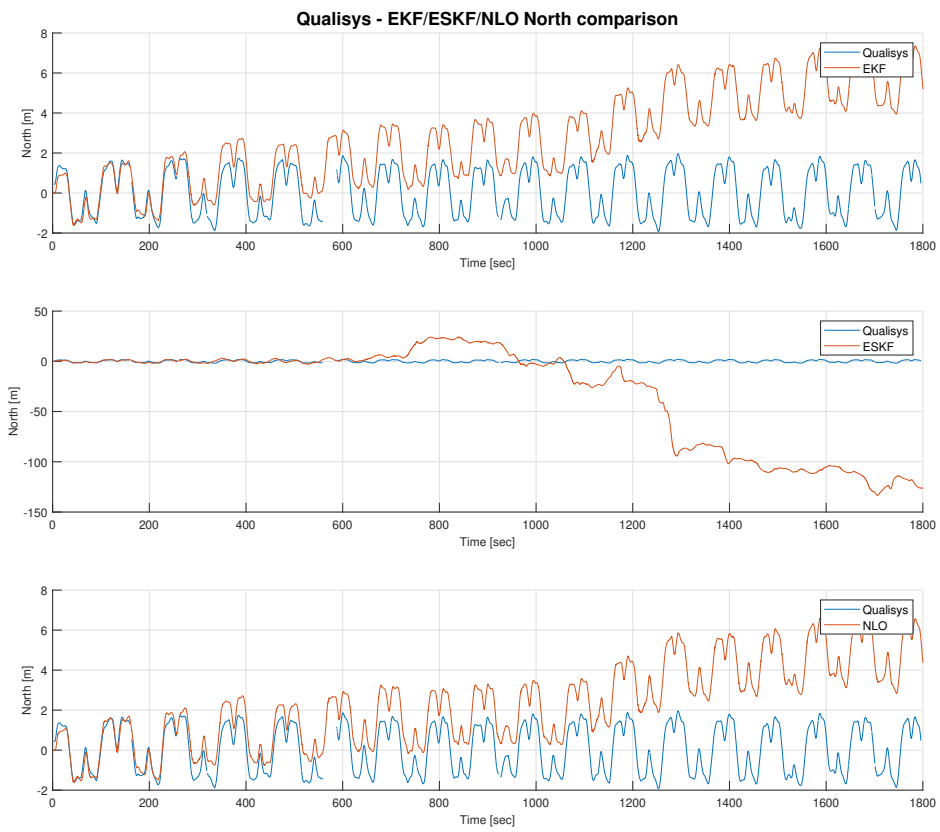


Figure 7.23: EKF/ESKF/NLO north trajectory comparison with Qualisys - 30 min square

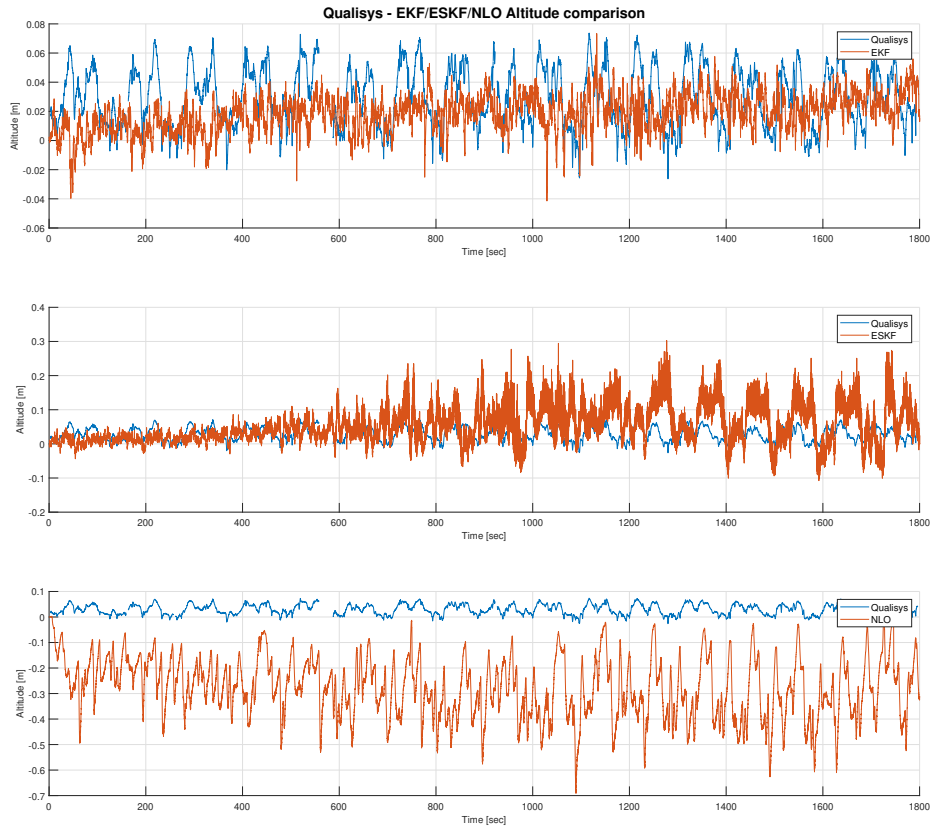


Figure 7.24: EKF/ESKF/NLO altitude trajectory comparison with Qualisys - 30 min square

Looking at the position estimates in the figures 7.22, 7.23, it is seen that the EKF, ESKF and NLO compare close to Qualisys between 0 and 200 seconds. After that they begin to drift. This is expected because the AUV does not have any sensor that gives direct measurements of horizontal position in the NED frame, like a GNSS receiver, which can update their horizontal position. The drift does not seem to be that large though. The EKF and NLO end up with roughly 2 meters drift in the east position and 5 meters drift in the north position.

For the ESKF, the filter begins to diverge completely at around 900 seconds (15 minutes). As seen in equation This could be due to a bug in the authors C++ implementation. However this seems strange because that would have most likely given inaccurate estimates much earlier. A more reasonable explanation could be that the IMU has been slightly moved during the test period. Since the IMU is not bolted or tightly attached to the AUV, the person moving Manta-2020 could have made a slight offset in its original position. This would have caused a mounting error that was not corrected for. This could have led to that the prolonged prediction of the ESKF giving very inaccurate estimates. But this seem yet again rather stranger, because then the EKF and NLO would also have been affected.

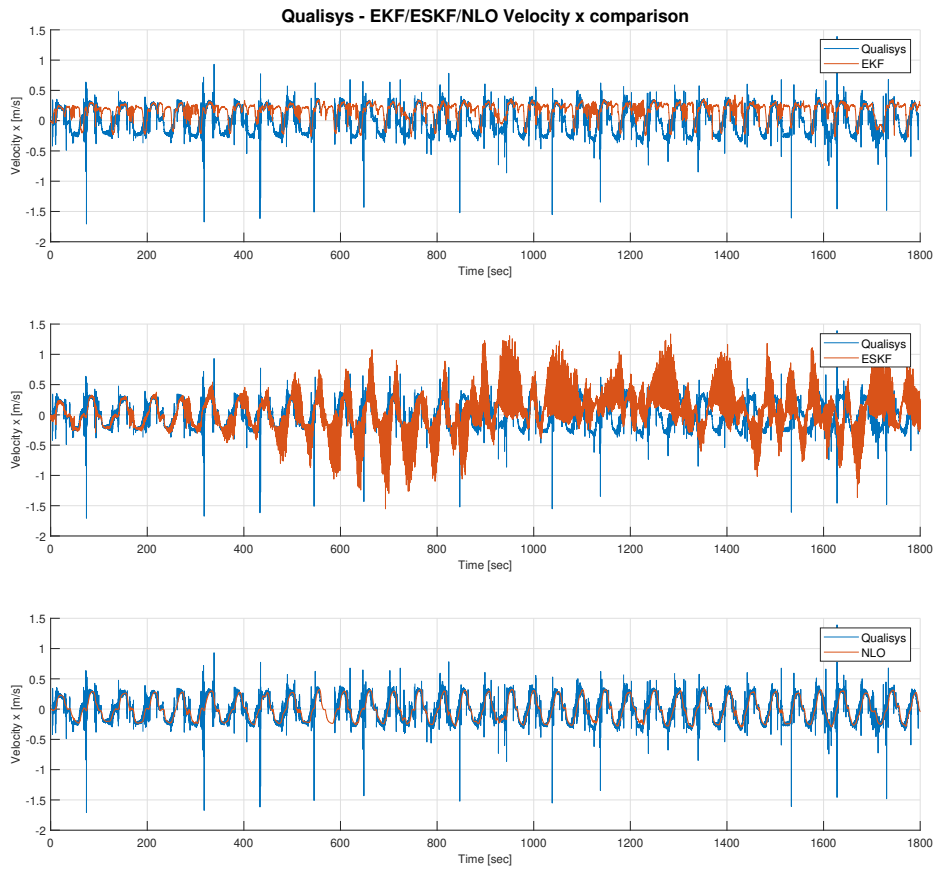


Figure 7.25: Velocity x comparison between the EKF, ESKF and NLO compared to Qualisys - 30 min square

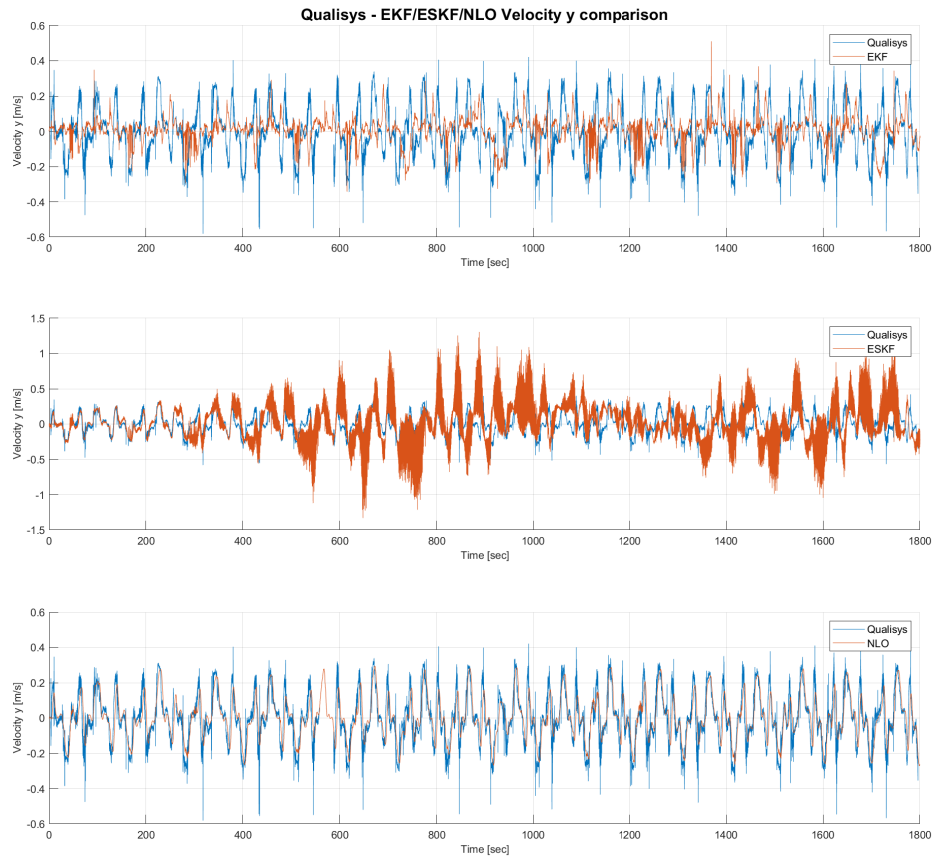


Figure 7.26: Velocity y comparison between the EKF, ESKF and NLO compared to Qualisys - 30 min square

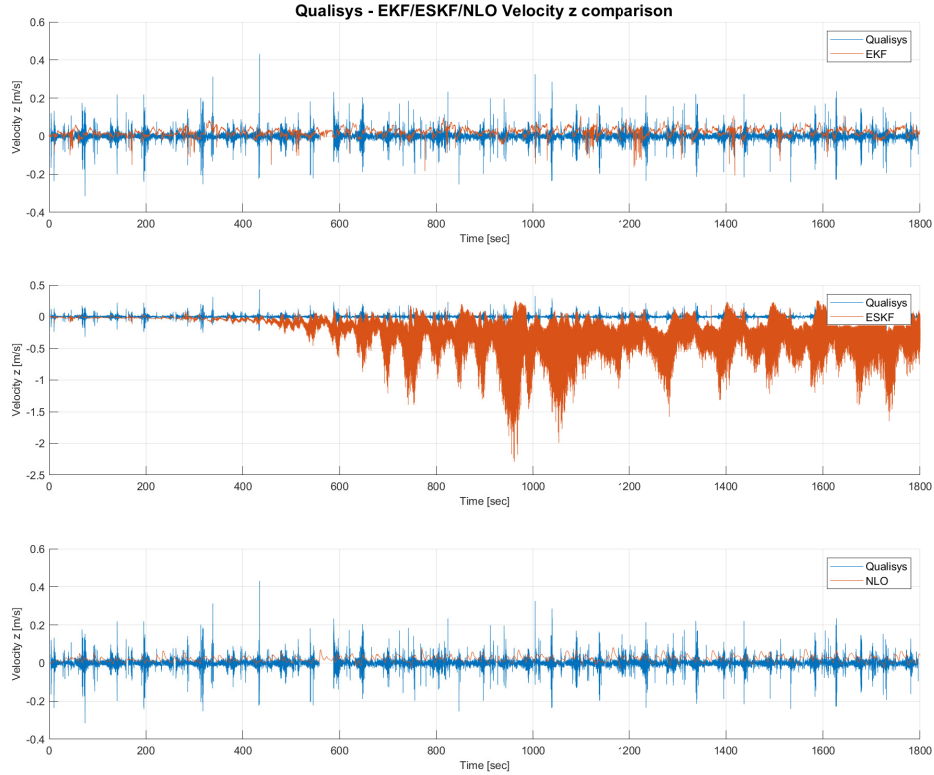


Figure 7.27: Velocity z comparison between the EKF, ESKF and NLO compared to Qualisys - 30 min square

Going over to the velocity plots in the figures 7.25, 7.26 and 7.27, the velocities of the NLO gives the most accurate estimation over time compared to the ESKF and EKF. However the ESKF seems to estimate more accurately in the first 250 seconds.

One interesting note is that velocities of the ESKF do not diverge completely, but rather begins to oscillate considerably. One reason for this could be that the prolonged prediction estimates have become divergent, but each time a DVL measurement comes into the filter, the velocity estimates becomes updated, and these estimates then come close to the measurement. This is most likely true, because the measurement covariance of the DVL \mathbf{R}_{dvl} has FOM values that are in the $10^{-4} - 10^{-7}$ range.

For the velocities of the NLO, they seem to have a "smoothing" behavior. This may be due to the tuning of the k_1 and k_2 in the injection term σ . This is because the velocity prediction is directly dependent on this injection term. Low values of these therefore gives little contribution to the acceleration term. Also the estimates of the attitude seems to be very accurate, which contribute in having an accurate $\mathbf{R}_b^n(\mathbf{q})$ matrix, which is important for the prediction step.

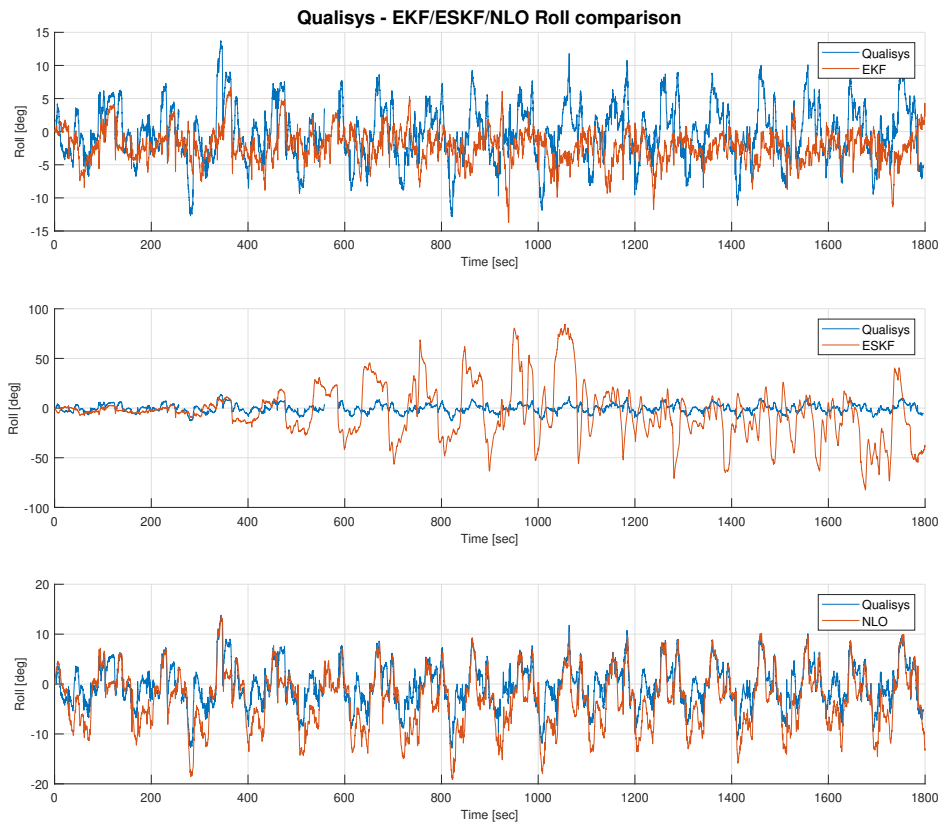


Figure 7.28: Roll comparison between the EKF, ESKF and NLO compared to Qualisys - 30 min square

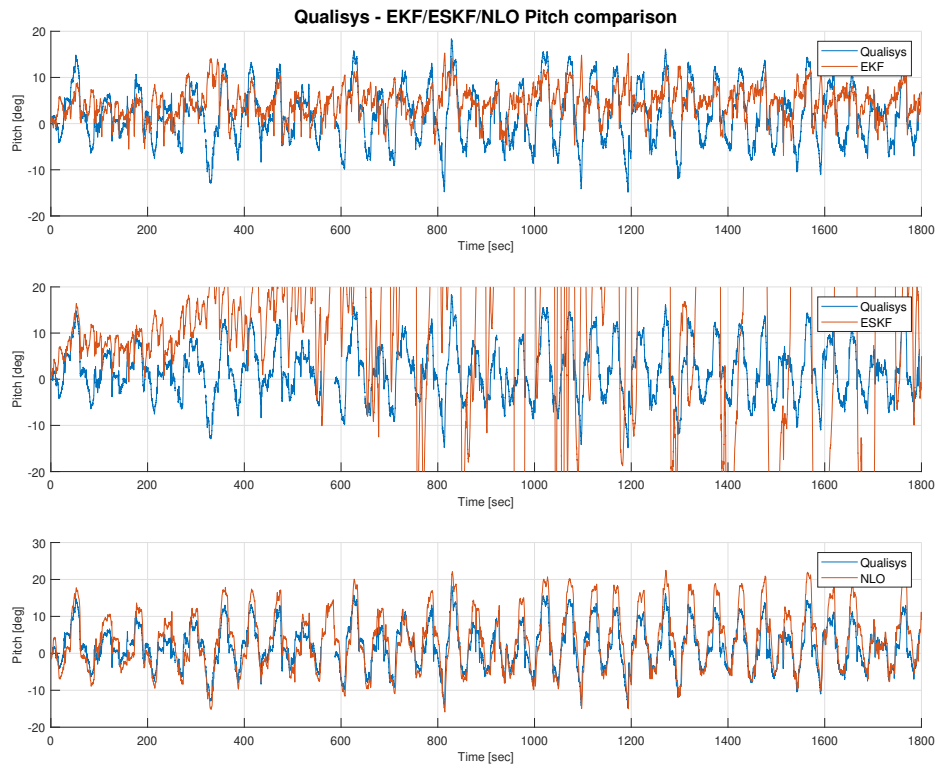


Figure 7.29: Pitch comparison between the EKF, ESKF and NLO compared to Qualisys - 30 min square

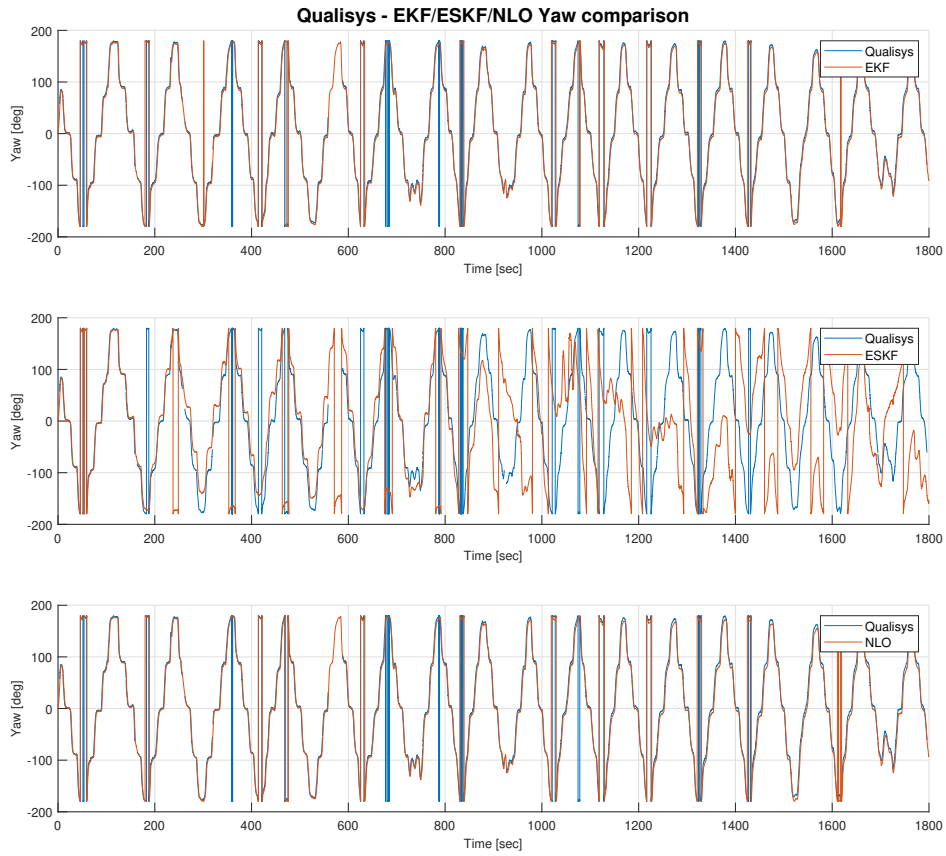


Figure 7.30: Yaw comparison between the EKF, ESKF and NLO compared to Qualisys - 30 min square

For the attitude estimates in figure 7.28, 7.29 and 7.30, the most interesting notes are that the yaw estimation is very accurate even for a long duration test of 30 minutes. There are no sensor that give measurements of heading directly, like a magnetometer.

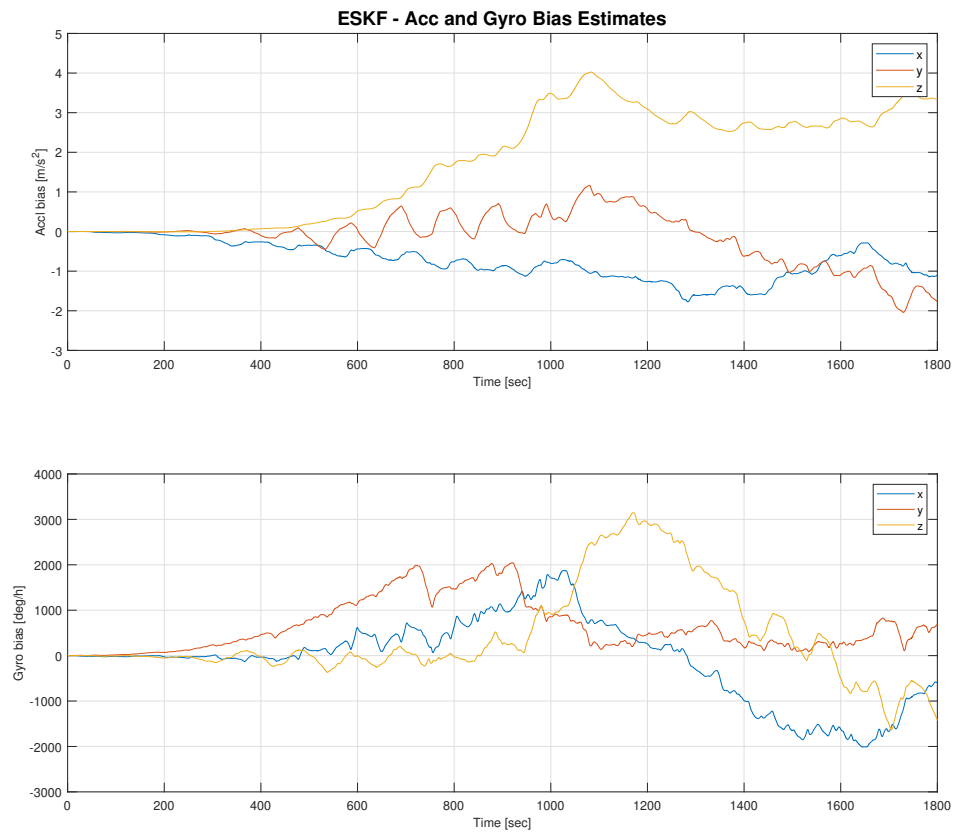


Figure 7.31: ESKF bias estimates - 30 min square

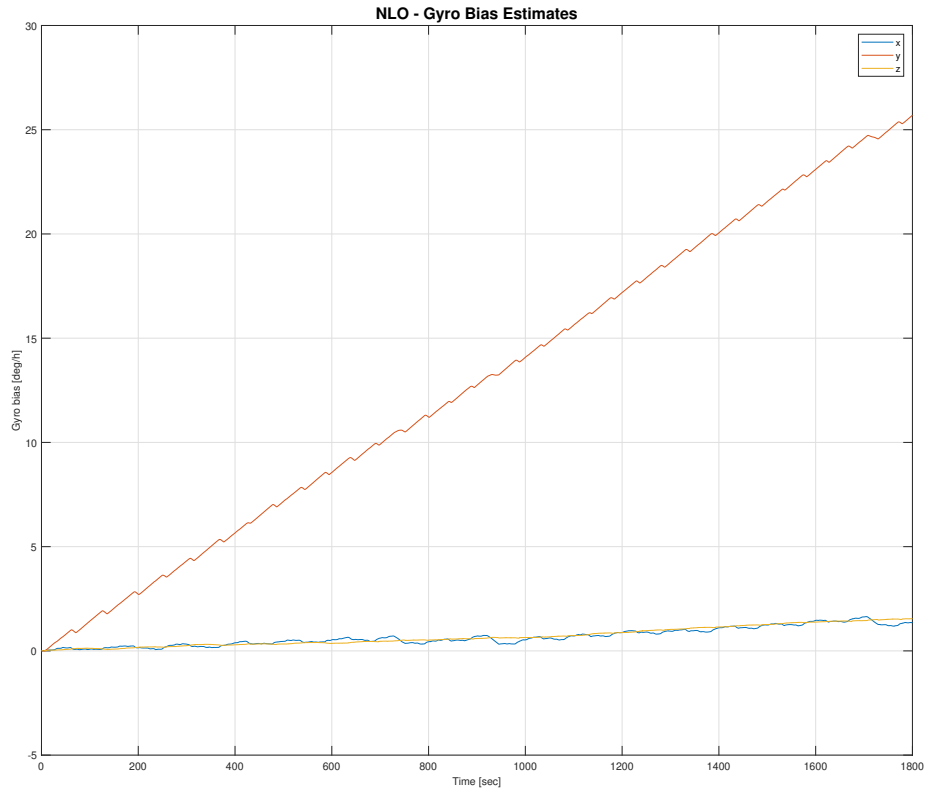


Figure 7.32: ESKF bias estimates - 30 min square

For the bias estimates of this test, given in the figures 7.31 and 7.32, the acceleration and gyro bias estimates of the ESKF becomes divergent after 200 seconds. The values alternatives well beyond 1000 deg/h for the gyro biases and between -2 and $3 \frac{m}{s^2}$ for the acceleration biases. This is a result from that the error covariance has diverged, giving inconsistent behavior of all the states. For the gyro biases of the NLO the z and x values seem to move slowly towards a steady state value. This should have been the x and y value, and not the z value, because of the z value, as described above, should not be observable. This could be due to a wrong setup of the figure. If this is the case, the non-observable bias should be the z value. Here it is seen that it grows continuously in a linear fashion. This will however be bounded by the tuning parameter \mathbf{M}_{gyro} .



Figure 7.33: Position error of EKF,ESKF and NLO compared to Qualisys - 30 min square

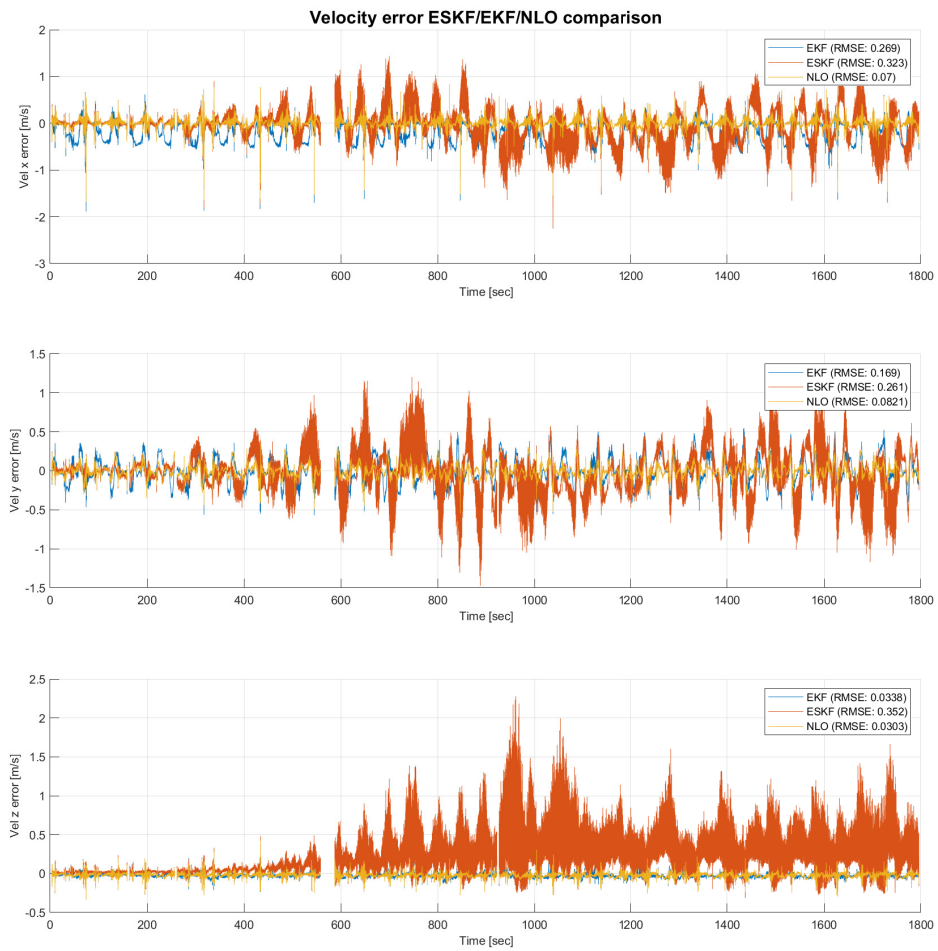


Figure 7.34: Velocity error of EKF,ESKF and NLO compared to Qualisys - 30 min square



Figure 7.35: Attitude error of EKF,ESKF and NLO compared to Qualisys - 30 min square

Looking at the figures of the PVA errors. The most reasonable would be to look at the comparison between the NLO and EKF. For the position errors given in the figure 7.33, it is observed that the NLO has a RMSE of 2.86 and the EKF has 3.38. One reason for this could be that the y velocities RMSE for the NLO (0.0821) is much smaller than the EKF rmse (0.169). However the EKF have lower RMSE (1.41) in the north position than the NLO (1.7). A reason for this could be that the process noise covariance for the position was well tuned for the EKF and not for the NLO.

Looking at the attitude error, it is observed that the NLO have lower RMSE values for roll and pitch than the EKF. A reason for this could be the gyro bias estimation of the NLO. This could also explain why the NLO has much lower north RMSE than the EKF. This is because an accumulating roll and pitch error, will make for a larger drift in east and north positions.

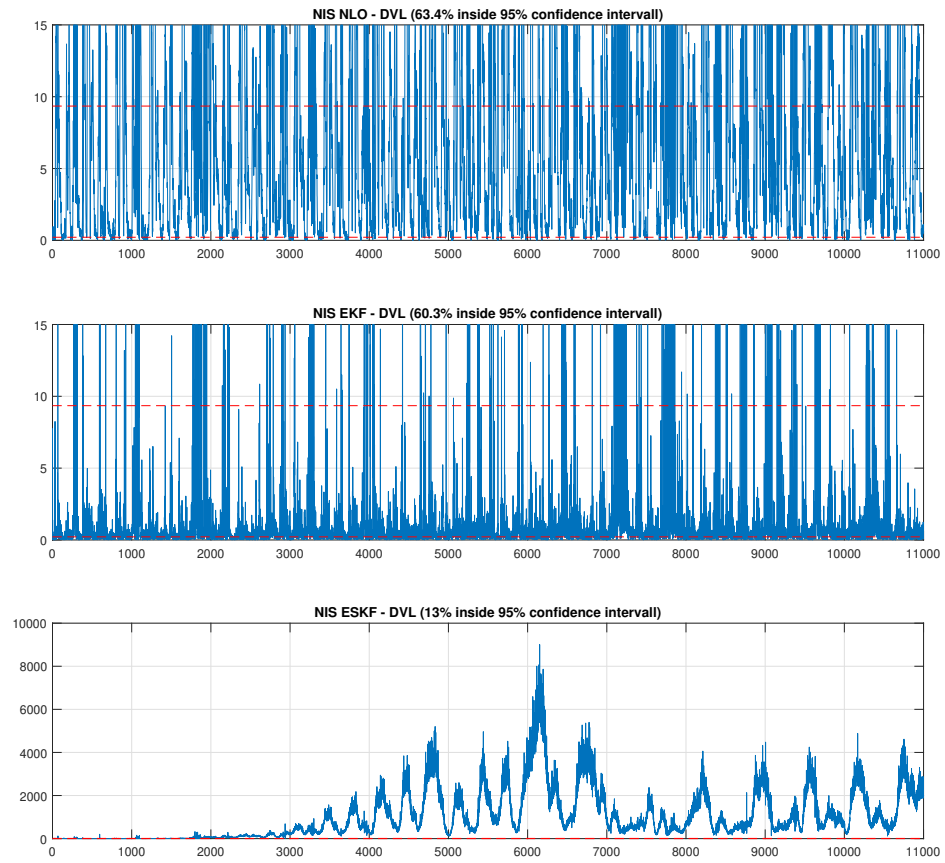


Figure 7.36: NIS DVL - EKF/ESKF/NLO comparison - 30 min square

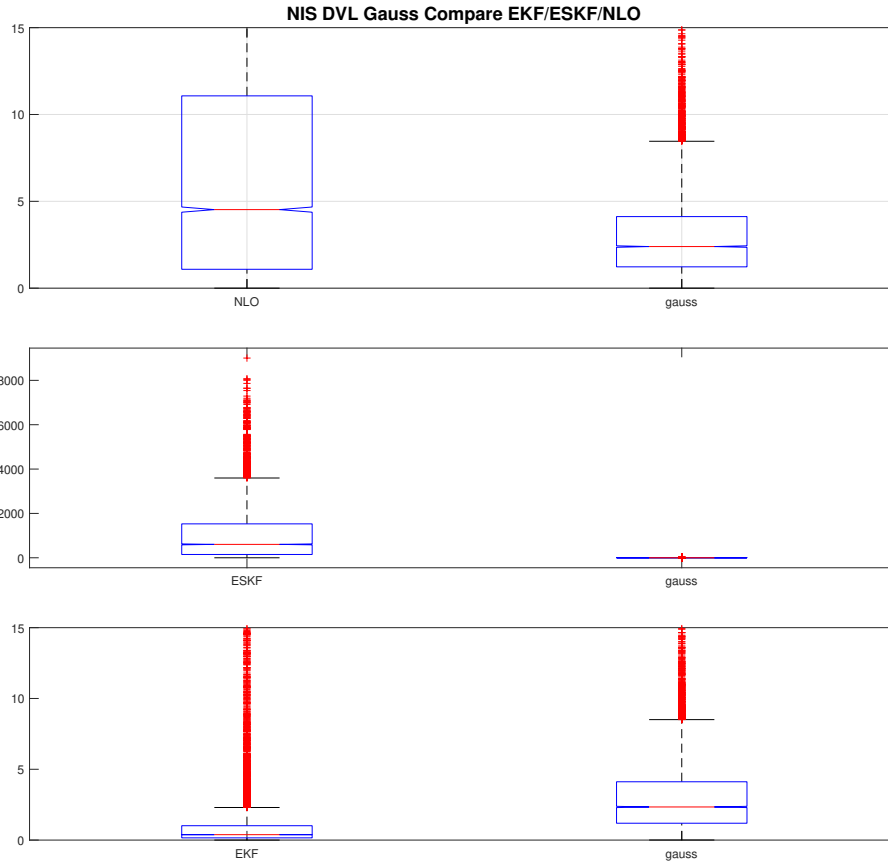


Figure 7.37: Gauss compare DVL - EKF/ESKF/NLO comparison - eight number

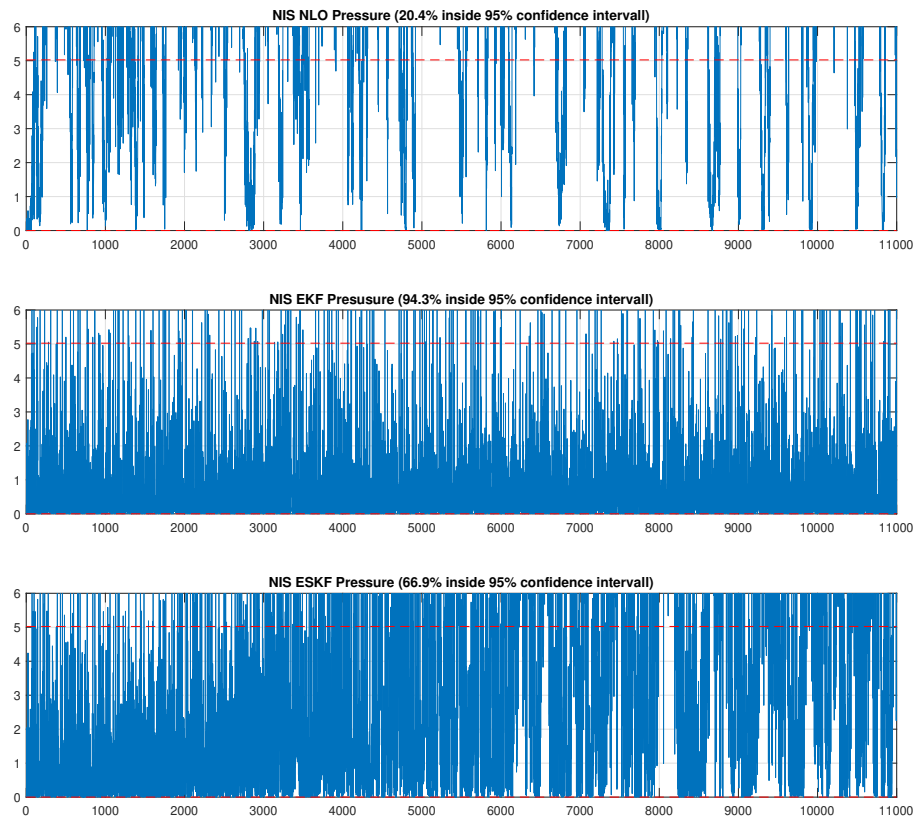


Figure 7.38: NIS pressure - EKF/ESKF/NLO comparison - 30 min square

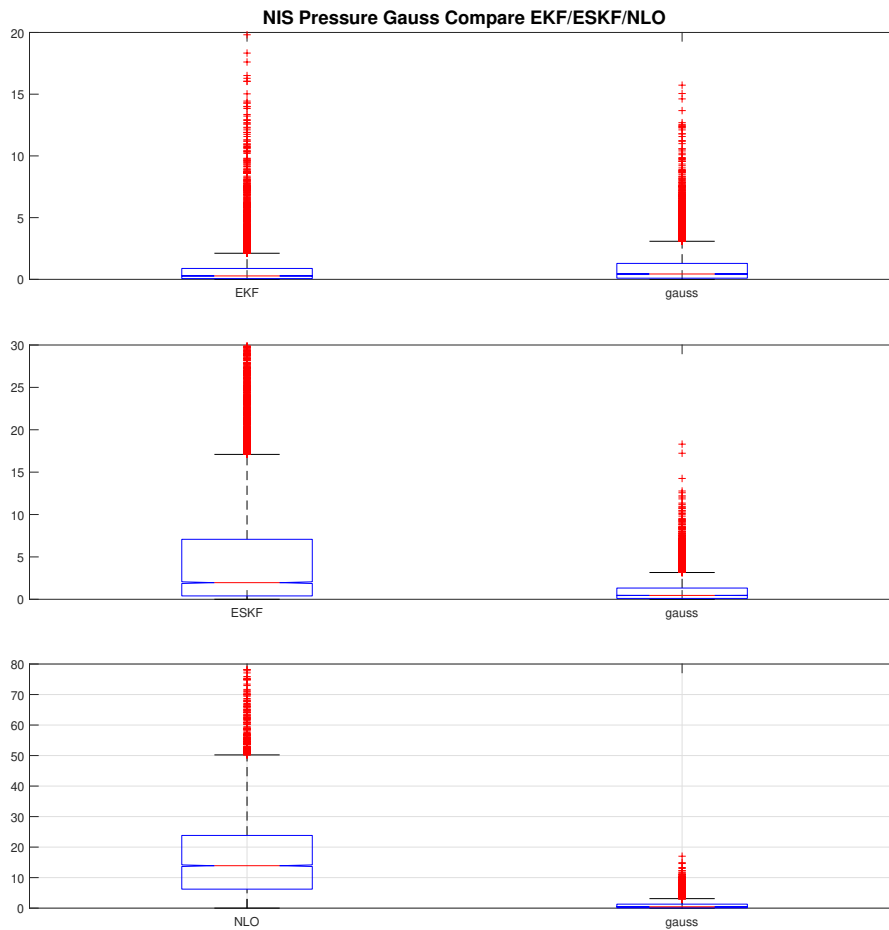


Figure 7.39: Gauss compare pressure - EKF/ESKF/NLO comparison - square

Going over to the NIS dvl plots in figure 7.36, 7.37, it is observed that the NIS of the NLO and EKF have over 60 % of their values inside the 95 % confidence interval. However, the DVL NIS of the NLO has many more outliers than the EKF as seen in the Guass compare figure 7.37. This means that the tuning of the continuous time process noise \mathbf{D} of the NLO have too small values for the velocity estimates. For the EKF on the other hand the NIS values are much lower and many of the values are below the lower limit. This means that continuous time process noise for the EKF have too large values for the velocity estimates. As discussed above, the latter is preferred, because the former can lead to divergence. The same holds true for the NIS of the pressure, seen in the figures 7.38 and 7.39.

7.1.3 Underwater sinus

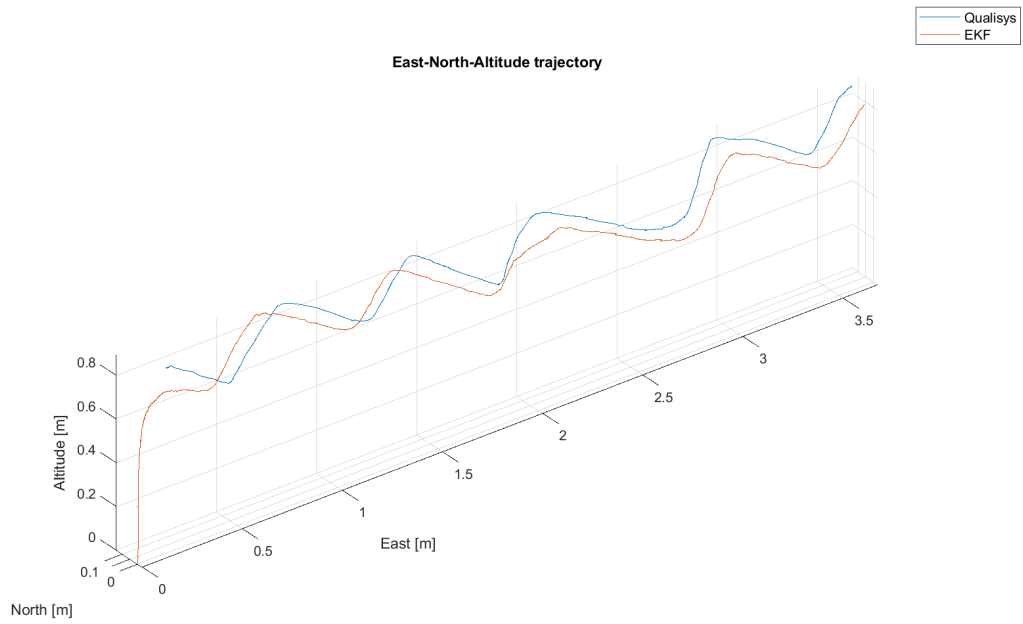


Figure 7.40: East-north-altitude 3D trajectory comparison between the EKF and Qualisys - underwater sinus

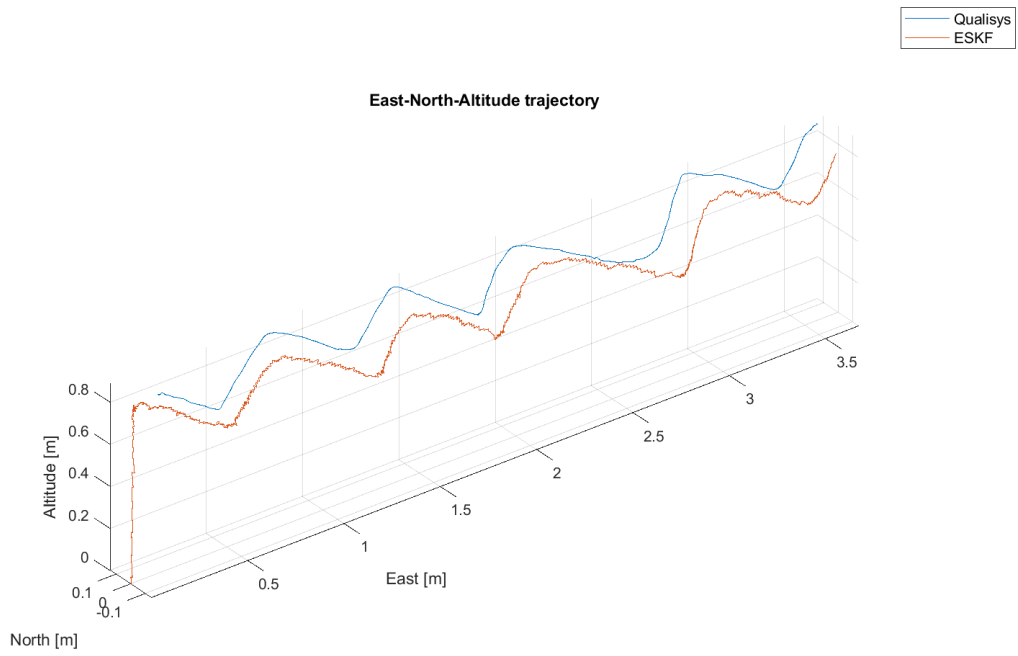


Figure 7.41: East-north-altitude 3D trajectory comparison between the ESKF and Qualisys - underwater sinus

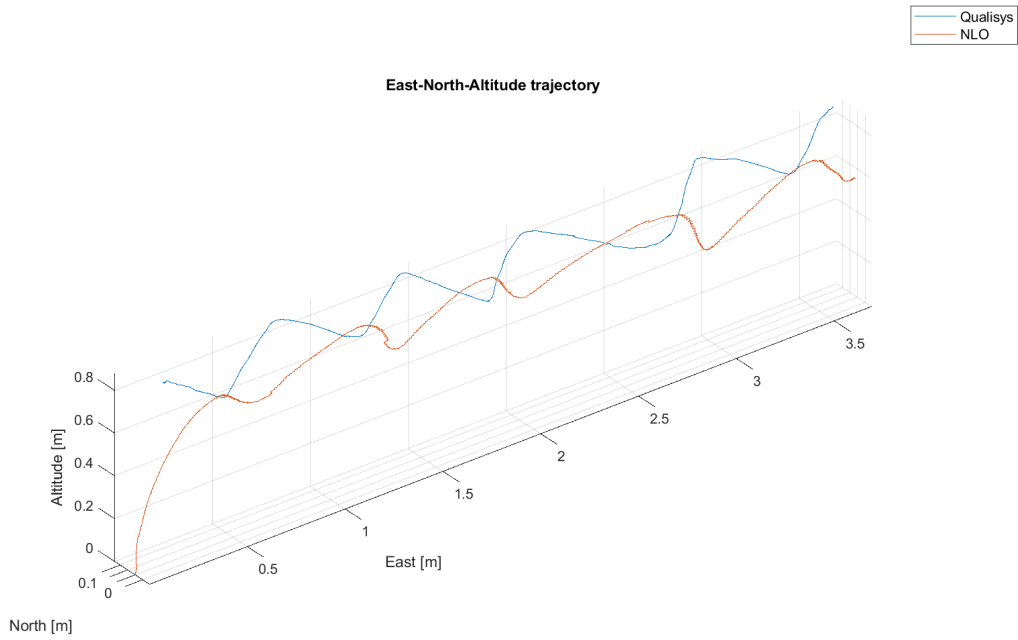


Figure 7.42: East-north-altitude 3D trajectory comparison between the NLO and Qualisys - underwater sinus

Looking at the east-north-altitude trajectory in the figures 7.40, 7.41 and 7.42, it is observed that the NLO seems to have a long transient period before reaching up from its initial estimates, compared to the ESKF and EKF. Also it is observed that the ESKF seems to have the fastest convergence to the Qualisys altitude trajectory. This could be because the k_1 and k_2 were tuned to low, giving too little contribution in the velocity estimates. It could have also been due to that the continuous time covariance matrix \mathbf{D} is tuned to low, such that the "trust" in prediction is higher. This could very well be the case, because the AUV was initially totally still in the water before starting the tests.

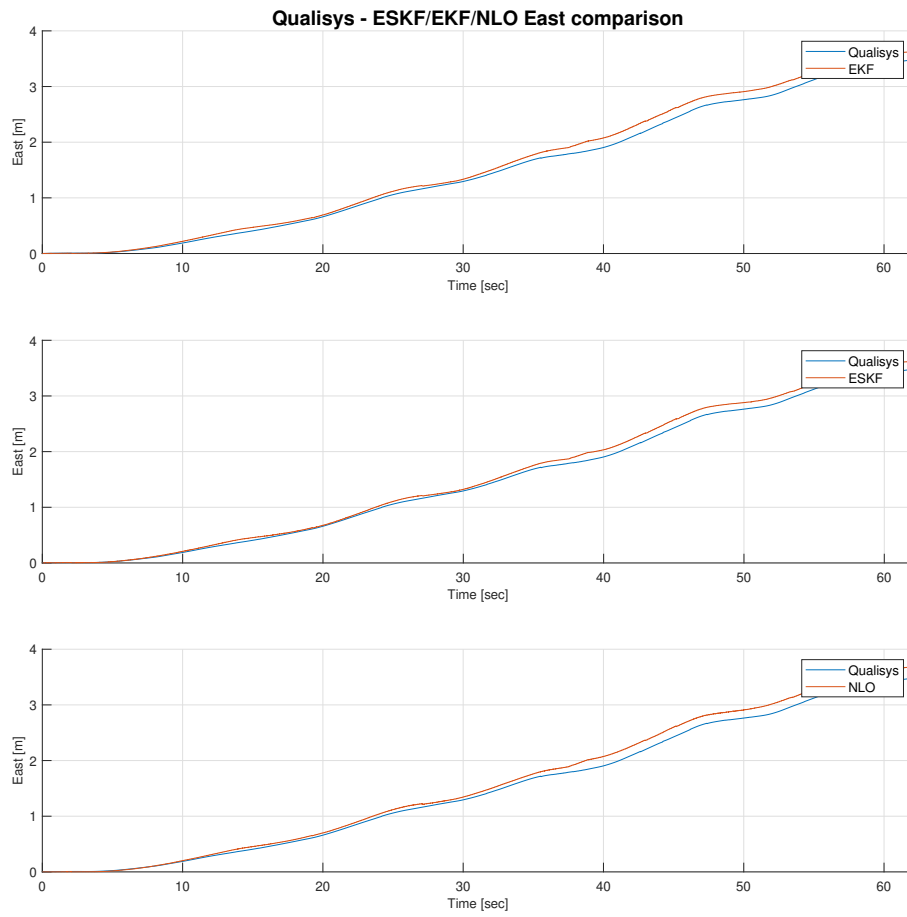


Figure 7.43: EKF/ESKF/NLO east trajectory comparison with Qualisys - underwater sinus

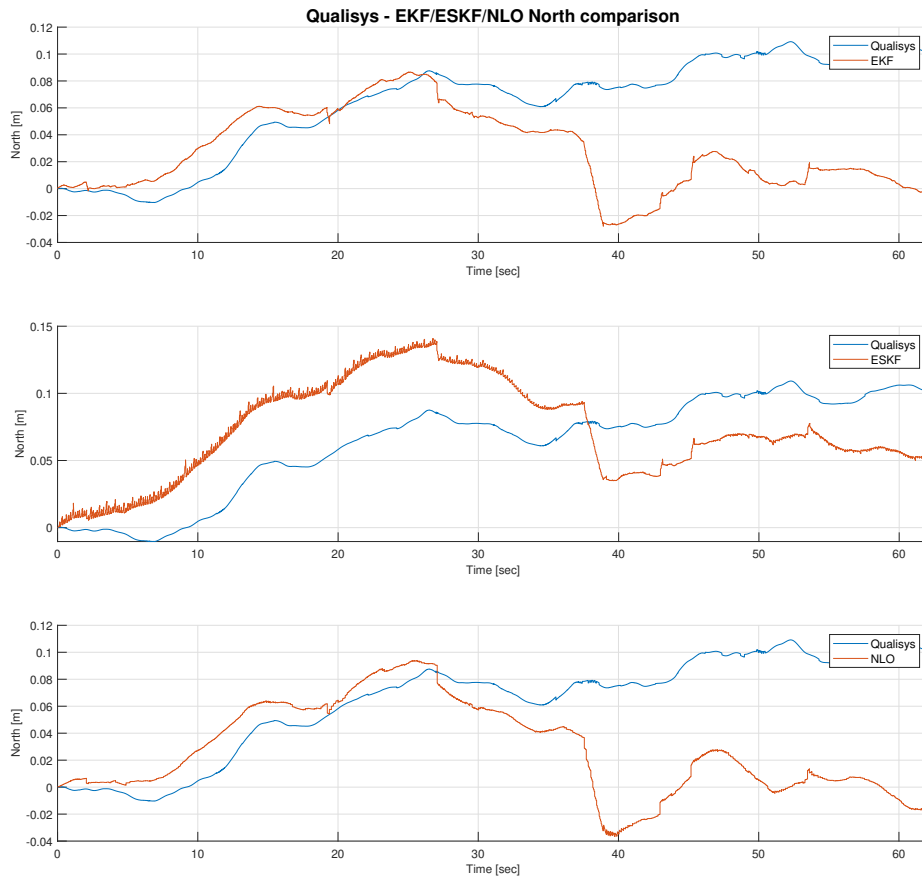


Figure 7.44: EKF/ESKF/NLO north trajectory comparison with Qualisys - underwater sinus

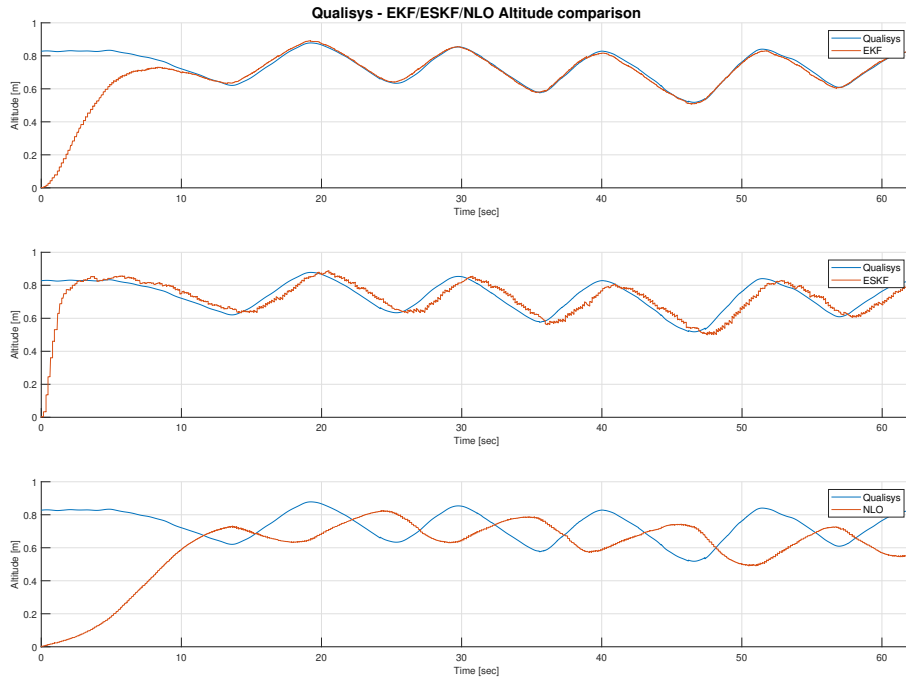


Figure 7.45: EKF/ESKF/NLO altitude trajectory comparison with Qualisys - underwater sinus

Going over to the position estimates in the figures 7.43, 7.44 and 7.45, it is observed that the north comparison is especially inaccurate for all the state estimators and it occurs a spike at the time of $t \approx 38$ seconds. This could be due to the range of the DVL not giving measurement close to the pool bottom. This will alter the positions estimates. The east estimates seem also to be affected by this. Making a slight offset to the Qualisys estimates after $t \approx 38$ seconds. For the altitude positions is verified from the discussion above that the NLO has a large transient behavior compared to the ESKF and EKF. The ESKF seems to be the one with the lowest transient period. This could come from the fact that the nominal state kinematics are based on the acceleration and gyro measurements from the IMU. And since Manta-2020 was almost standing still when starting the test, the contribution of the prediction estimate $h(\mathbf{x}_k)$ in the update step (given in the equations 5.81 - 5.83) with the pressure measurement could have been very low. This means that z-position would converge quickly to the pressure measurement. How quickly this goes is, as discussed in the eight test section, based on the measurement covariance \mathbf{R}_{pres} .

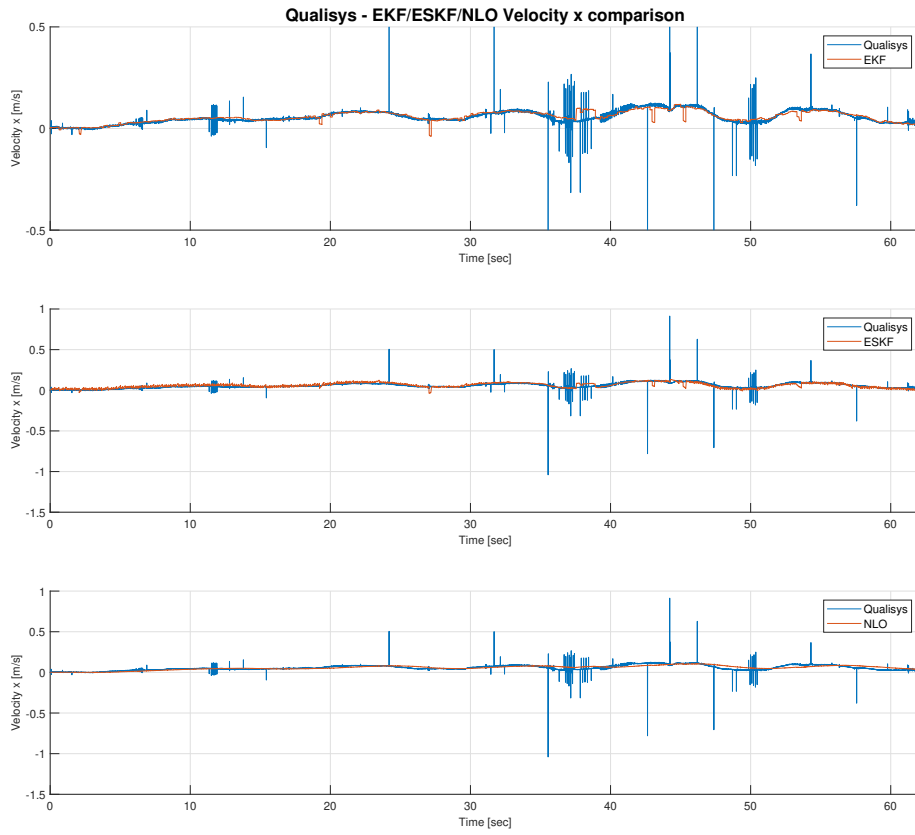


Figure 7.46: Velocity x comparison between the EKF, ESKF and NLO compared to Qualisys - underwater sinus

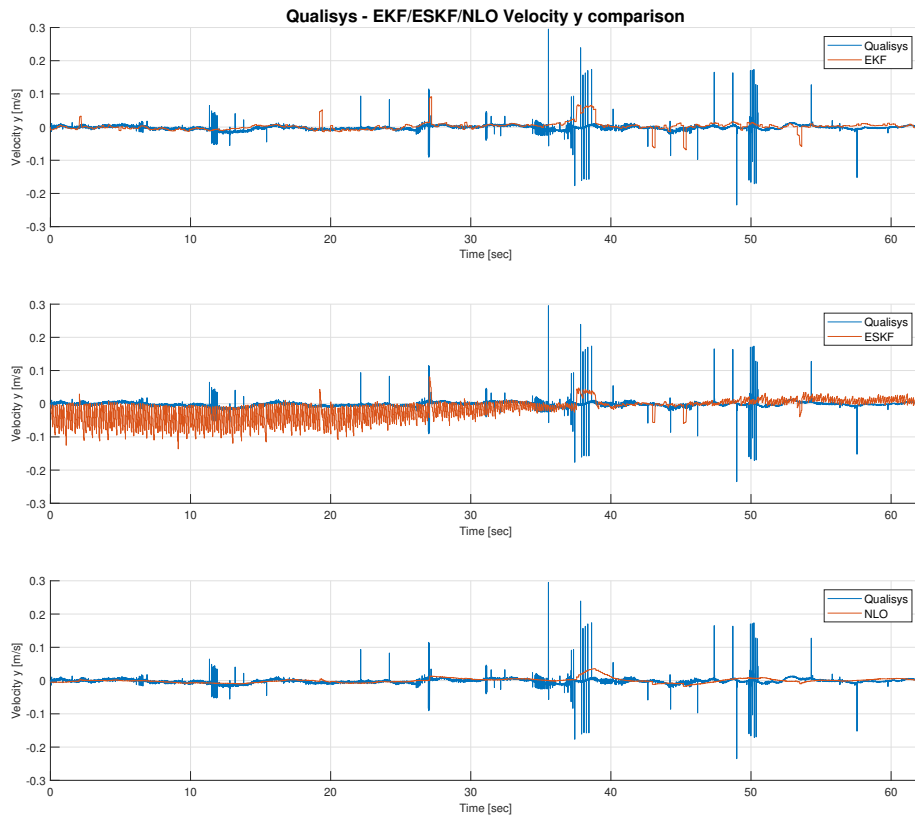


Figure 7.47: Velocity y comparison between the EKF, ESKF and NLO compared to Qualisys - underwater sinus

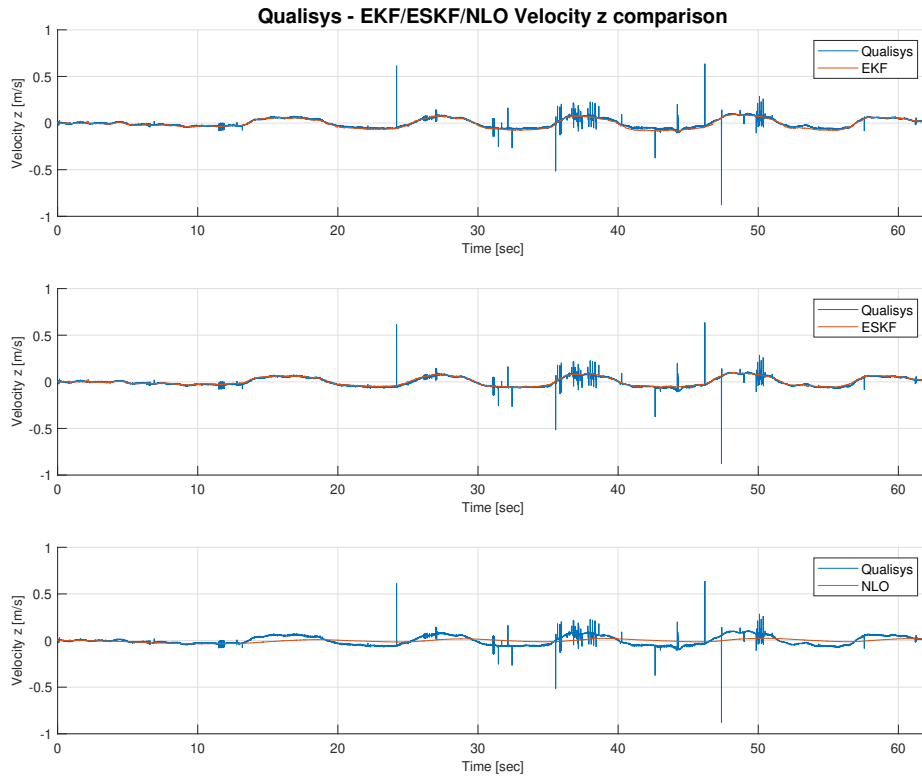


Figure 7.48: Velocity z comparison between the EKF, ESKF and NLO compared to Qualisys - underwater sinus

Looking at the velocity figures 7.46, 7.47, 7.48, it is observed that the x velocity has a spike around 38 seconds. This confirms the discussion of the position plots above, that this could be due to inaccurate DVL measurements. The interesting note here is that the y velocity of ESKF has very oscillating behavior. As discussed in the 30 min test, this could be due to the implementation bug. This behavior however, was not before 400 seconds. Furthermore this behavior did not show up in the eight test or the underwater square (discussed below) test either. This is also the reason for the spiked behavior of the ESKF in the first 38 seconds in the north comparison plot.

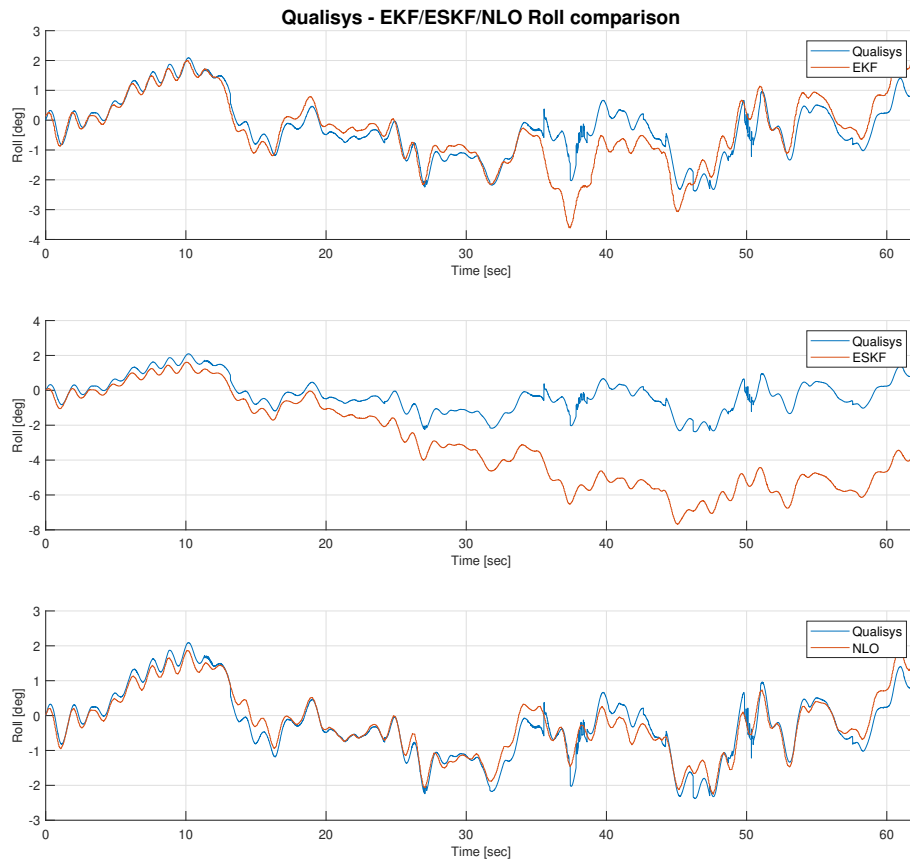


Figure 7.49: Roll comparison between the EKF, ESKF and NLO compared to Qualisys - underwater sinus

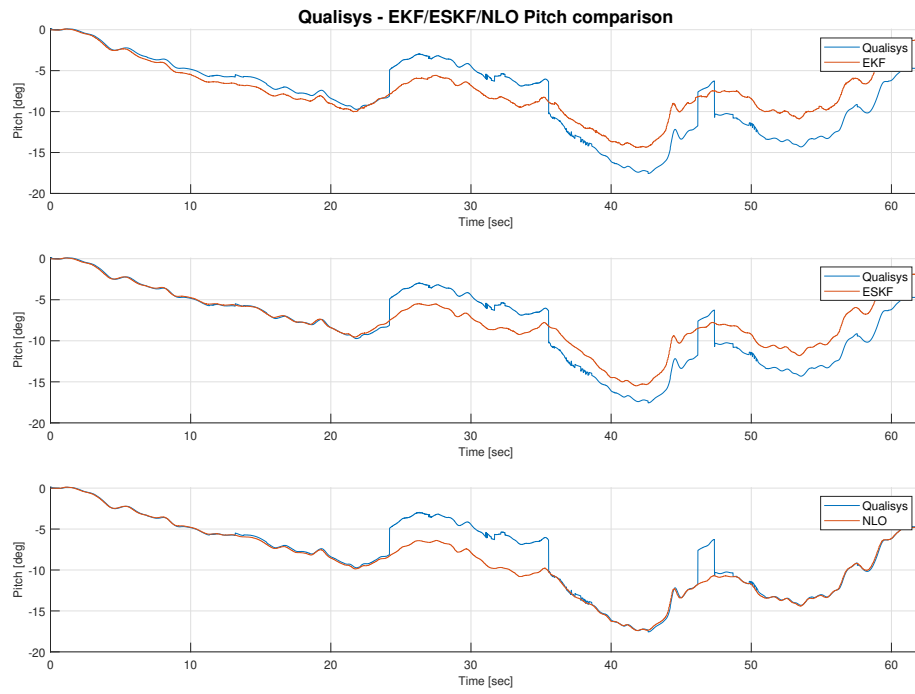


Figure 7.50: Pitch comparison between the EKF, ESKF and NLO compared to Qualisys - underwater sinus

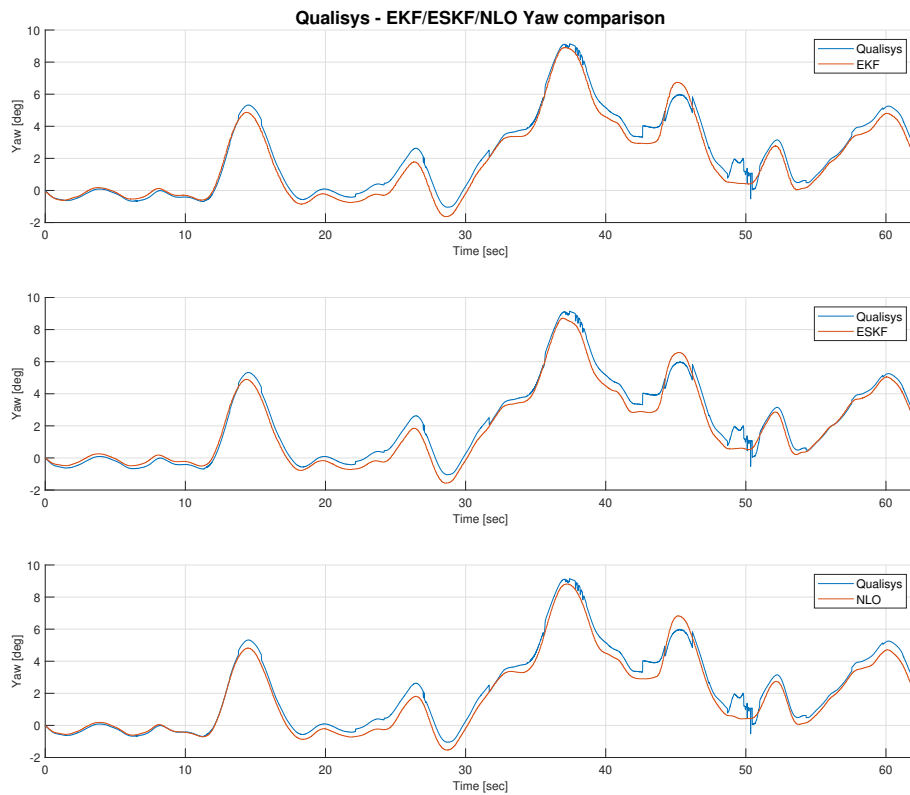


Figure 7.51: Yaw comparison between the EKF, ESKF and NLO compared to Qualisys - underwater sinus

Observing the attitude figures 7.49, 7.50 and 7.51 one very interesting note here is the that when the corrupt measurements of the DVL comes at around 38 seconds the roll and pitch estimates of the EKF and especially the ESKF, seems to get more inaccurate and altered. This is not the case for the NLO. The reason for this is that the NLO has its attitude feedback interconnected with the translational motion observer described in section 5.8. This means that the update on attitude is like its "seperated" from the behavior of the translation motion observer, which could explain why it does not alter the NLO pitch and roll estimates. The yaw estimates of ESKF and EKF seem to not be altered though. The spikes occurring on the Qualisys estimates could be due to the cameras loosing track of the balls.

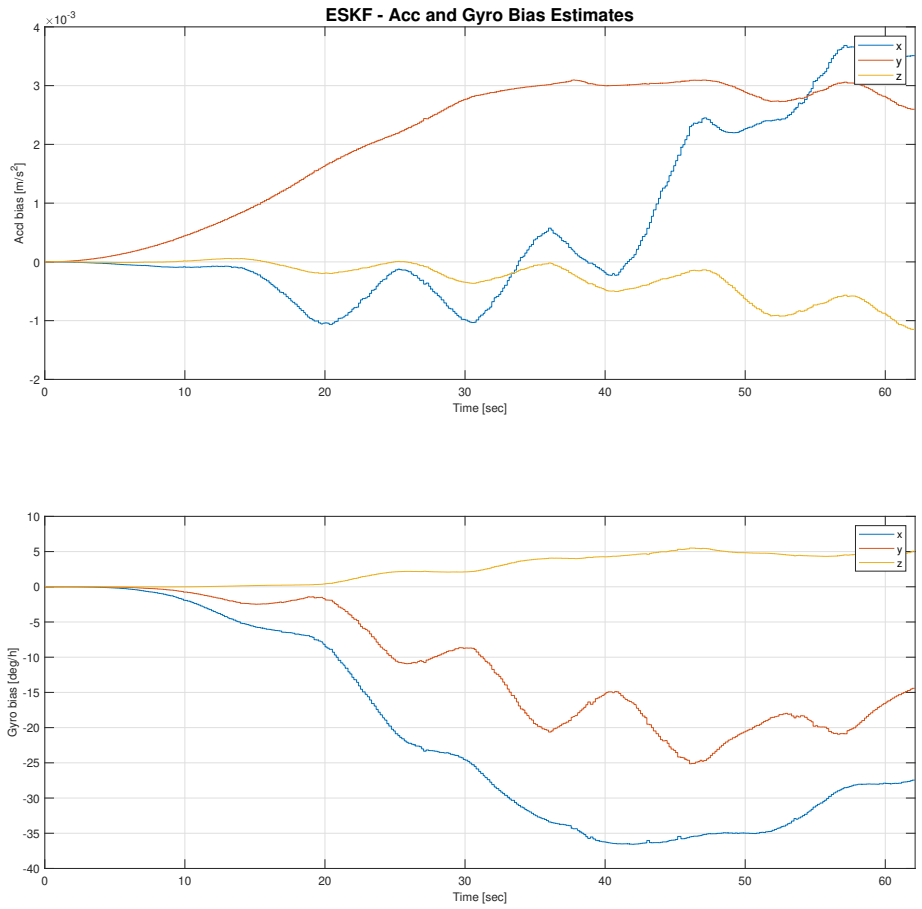


Figure 7.52: ESKF bias estimates - underwater sinus

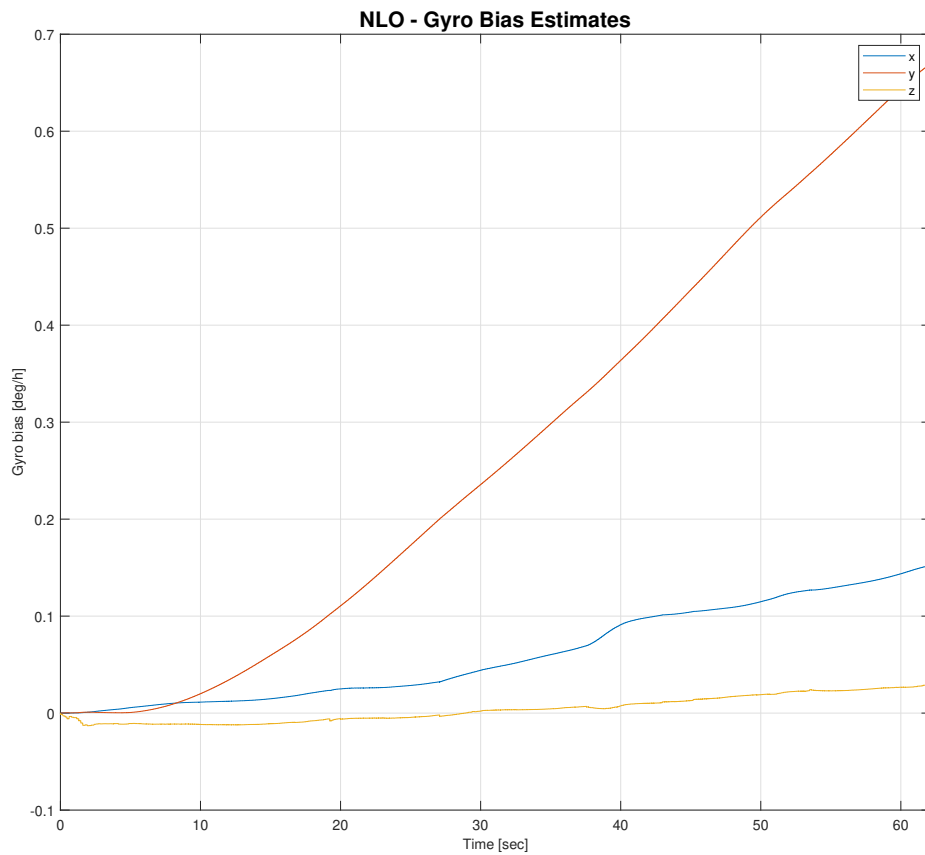


Figure 7.53: NLO bias estimates - underwater sinus

The figures 7.52 and 7.53 shows the bias plots of the test. The interesting note here is that the z acceleration bias of the ESKF is oscillating. This is could be due to the up-and-down motion of the AUV and that the measurements coming from the pressure sensor are moving up and down. However it seems like it is not altering its transient behavior.

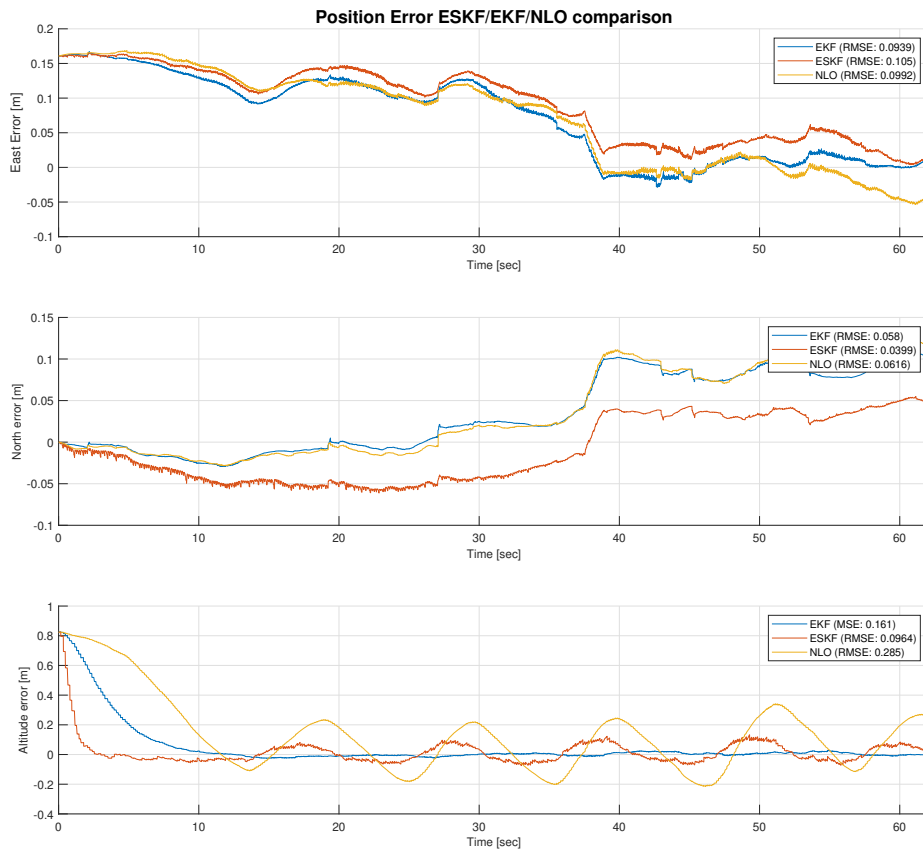


Figure 7.54: Position error of EKF,ESKF and NLO compared to Qualisys - underwater sinus

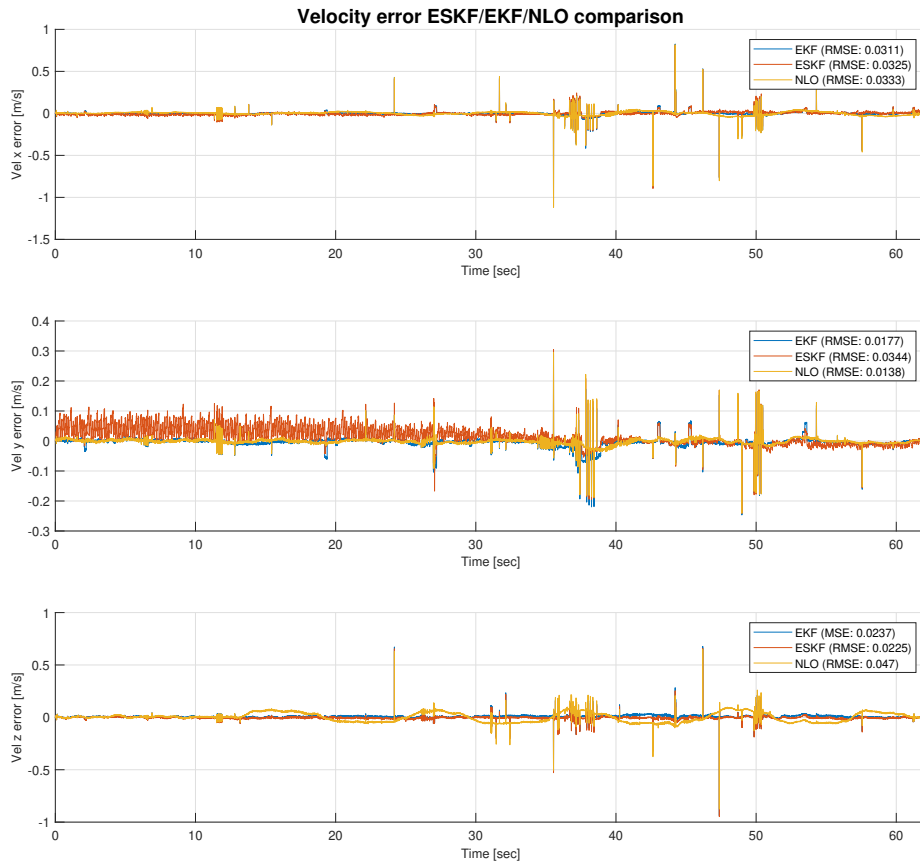


Figure 7.55: Velocity error of EKF,ESKF and NLO compared to Qualisys - underwater sinus

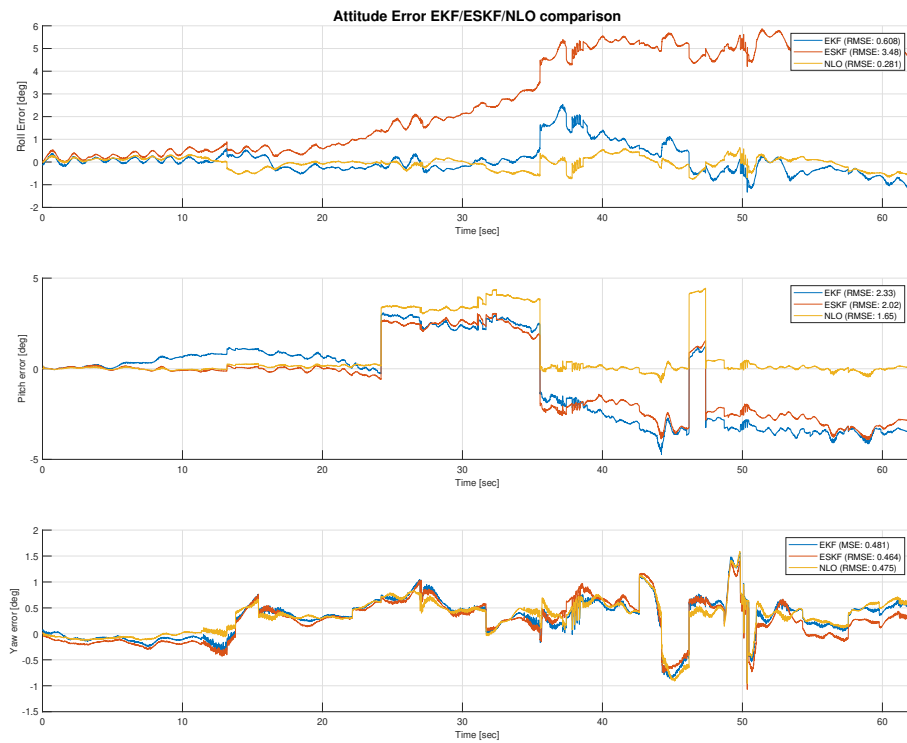


Figure 7.56: Attitude error of EKF,ESKF and NLO compared to Qualisys - underwater sinus

Going over to the position, velocity and attitude error in 7.54, 7.55 and 7.56, the overall RMSE of the ESKF is greater than the NLO and EKF. This is expected since the velocity estimation had an oscillating behavior with the reasons discussed above. Furthermore the overall RMSE of the NLO is actually greater than the EKF even though the pitch and roll RMSE estimates are much lower relatively speaking. This could be because of the slow transient behavior of the NLO.

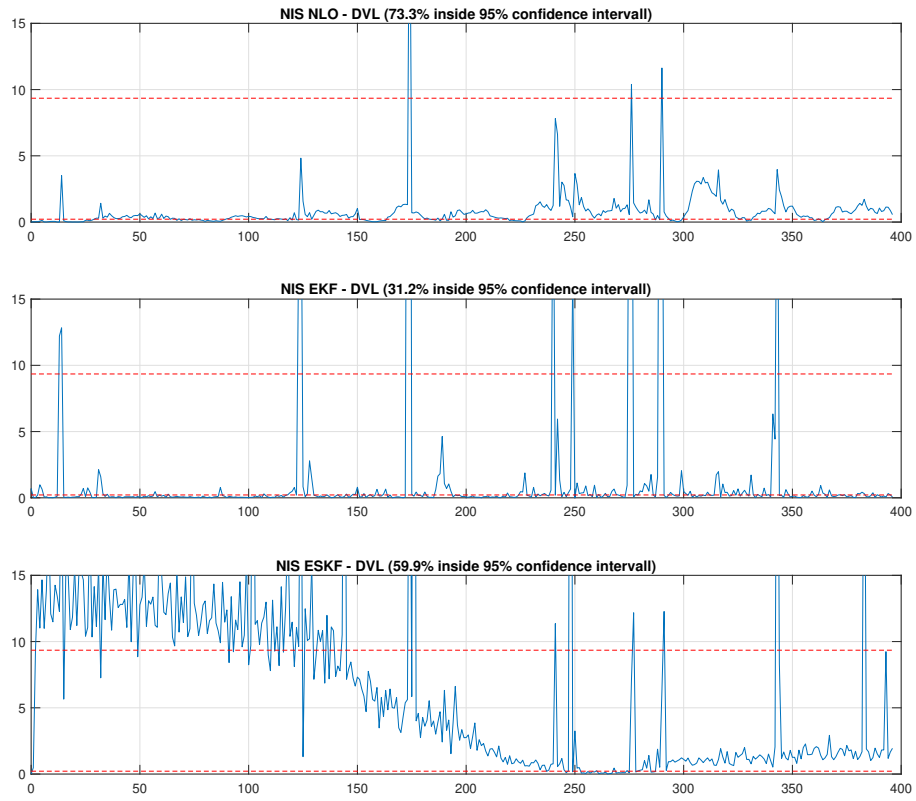


Figure 7.57: NIS DVL - EKF/ESKF/NLO comparison - underwater - sinus

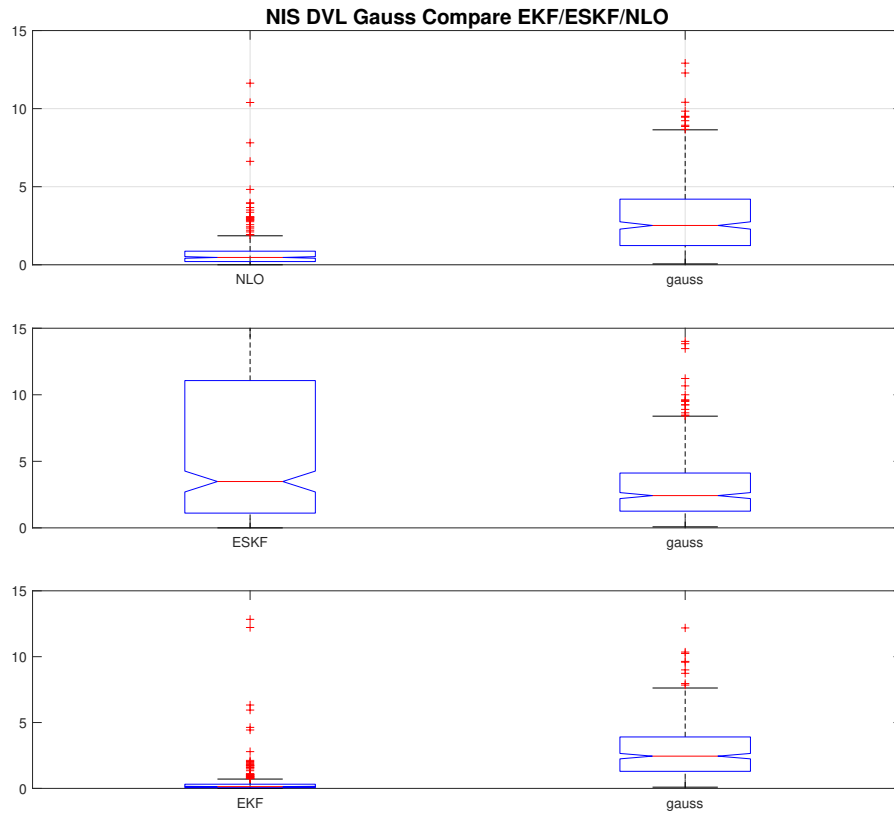


Figure 7.58: Gauss compare DVL - EKF/ESKF/NLO comparison - underwater sinus

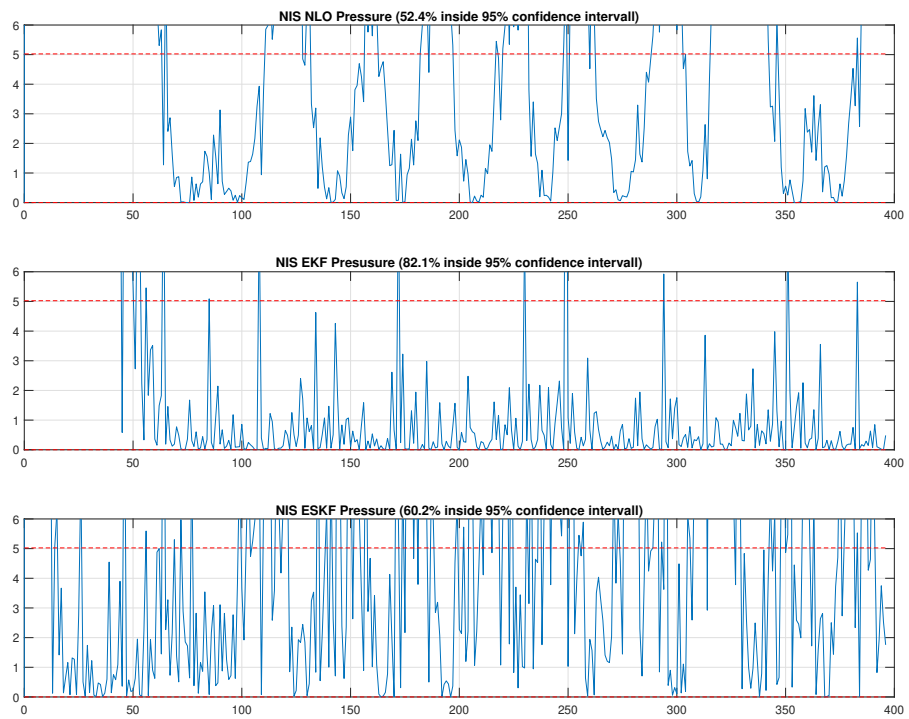


Figure 7.59: NIS pressure - EKF/ESKF/NLO comparison - underwater square

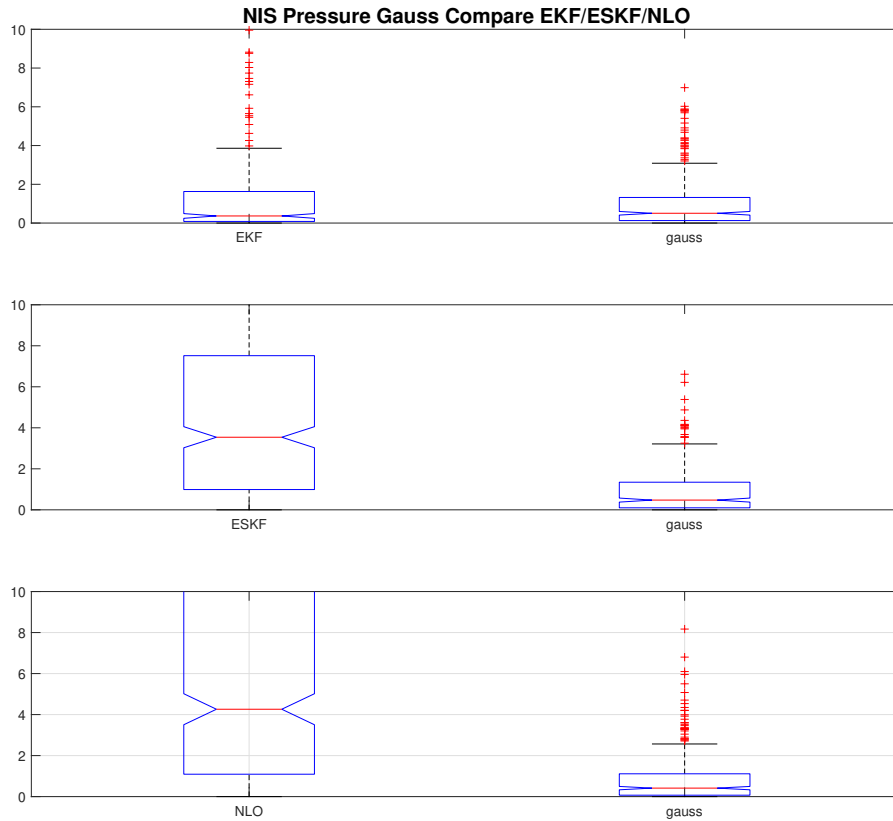


Figure 7.60: Gauss compare pressure - EKF/ESKF/NLO comparison - underwater sinus

Moving over to the DVL NIS plots in 7.57, it is observed that a lot of the values are under the lower limit for the NLO and EKF. This usually means as discussed previously, that the continuous process noise covariance matrix \mathbf{D} has too large values. Tuning this would have maybe led to more accurate behavior compared to Qualisys. Observing the NIS values for the ESKF, most of the values are above the upper limit and then goes towards the lower limit. This is rather expected because of the oscillating behavior of the velocity y measurement. This is resembled further in the gauss comparison figure 7.58.

For the pressure NIS values in figure 7.59, the NIS values of the ESKF and EKF have the most values inside the upper and lower limit with 82.1 % and 60.2 % respectively. However for the NLO only 52.4 % are inside the confidence interval. This means yet again that the tuning of the continuous time process noise covariance for the position is not well tuned. Also looking at the 7.60, the NLO may diverge if this test was to be prolonged.

7.1.4 Underwater square

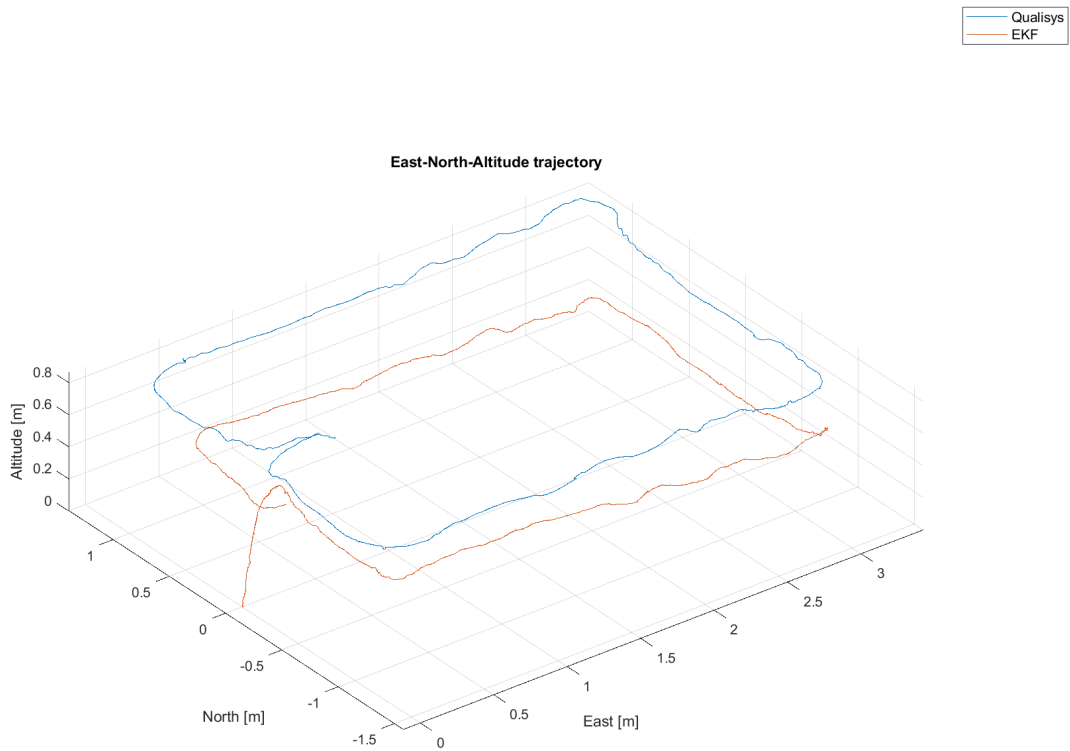


Figure 7.61: East-north-altitude 3D trajectory comparison between the EKF and Qualisys - underwater square

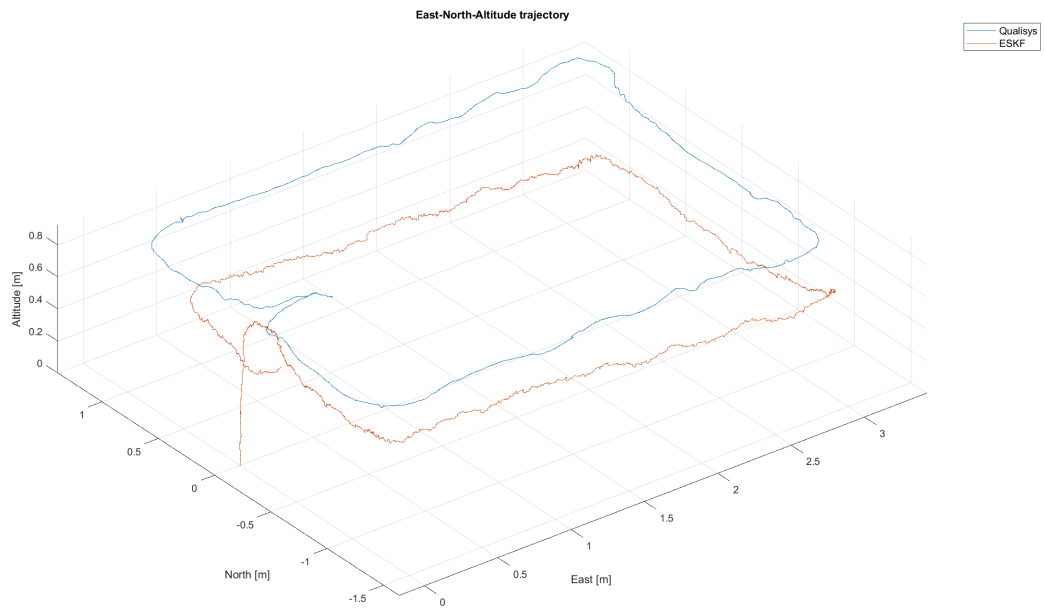


Figure 7.62: East-north-altitude 3D trajectory comparison between the ESKF and Qualisys - underwater square

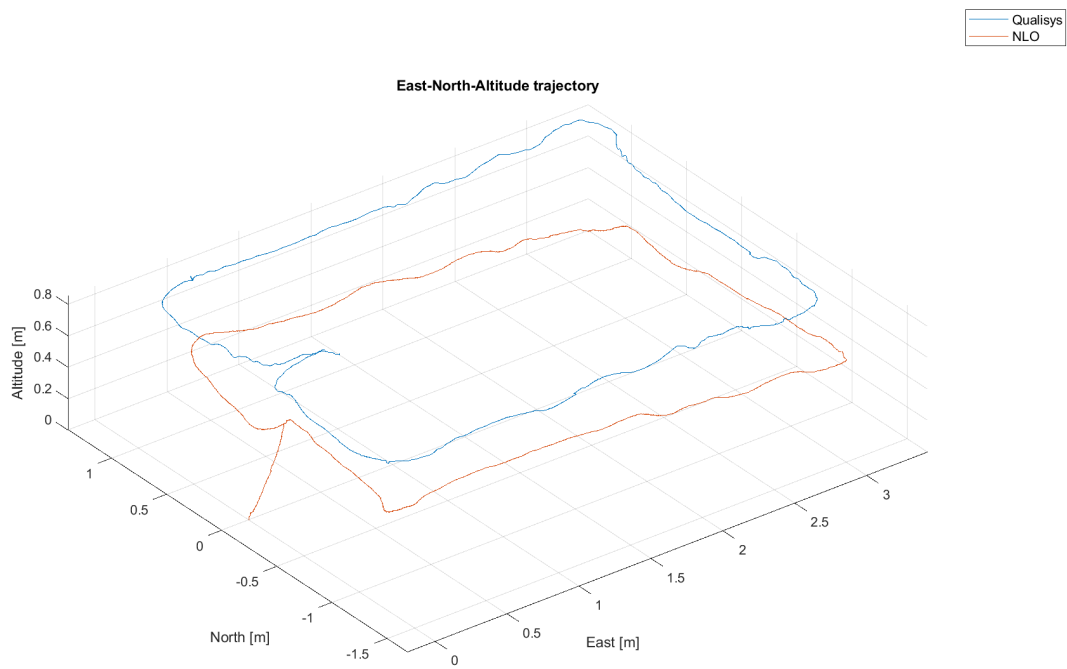


Figure 7.63: East-north-altitude 3D trajectory comparison between the NLO and Qualisys - underwater square

Comparing the figures 7.61, 7.62 and 7.42 it is seen that the NLO has a slow transient behavior like in the underwater sinus test. This usually means, that the k_1 and k_2 were tuned to low, as discussed for the underwater sinus test. Also in this case the ESKF has the fastest convergence.

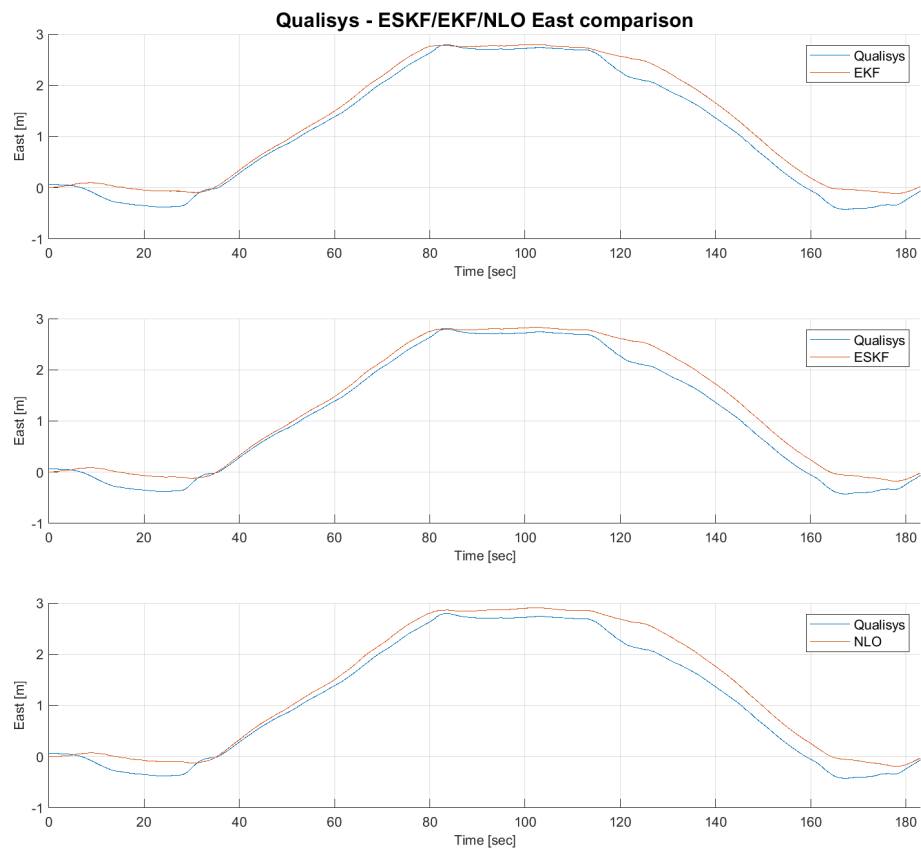


Figure 7.64: EKF/ESKF/NLO east trajectory comparison with Qualisys - underwater square

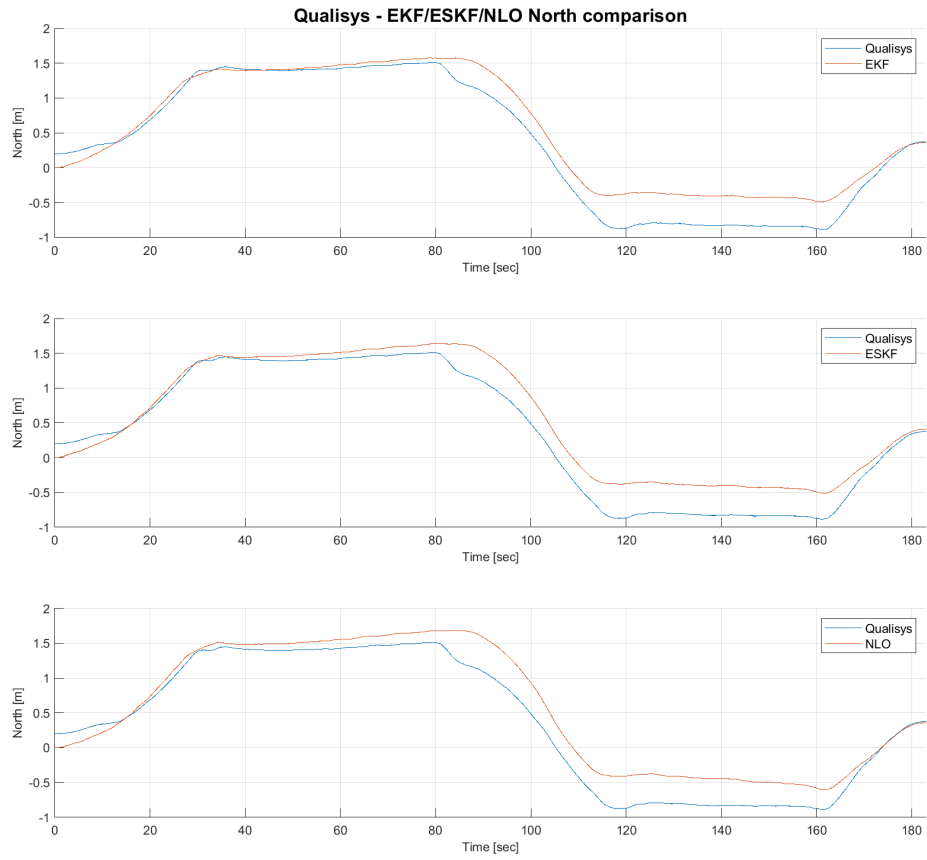


Figure 7.65: EKF/ESKF/NLO north trajectory comparison with Qualisys - underwater square

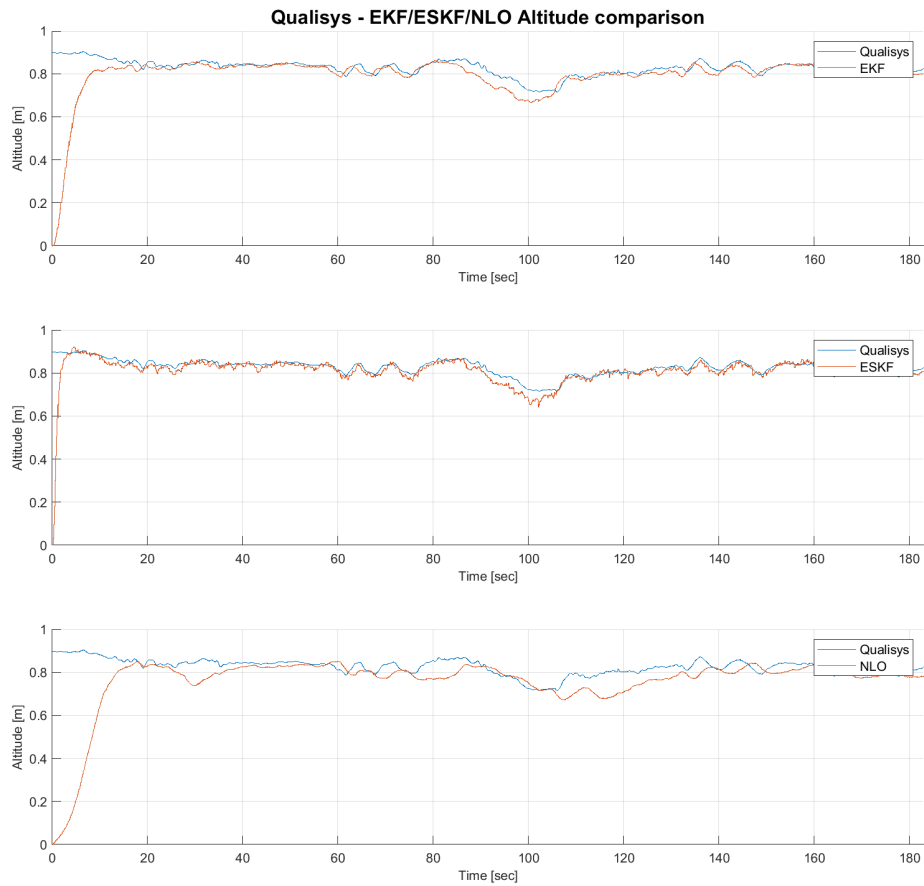


Figure 7.66: EKF/ESKF/NLO altitude trajectory comparison with Qualisys - underwater square

For the east, north and altitude figures given in 7.64, 7.65 and 7.66 the most interesting note here is that the transient behavior of the altitude of all three estimators seems to be faster than the underwater sinus test. Still the ESKF has the fastest behavior as seen in the altitude comparison.

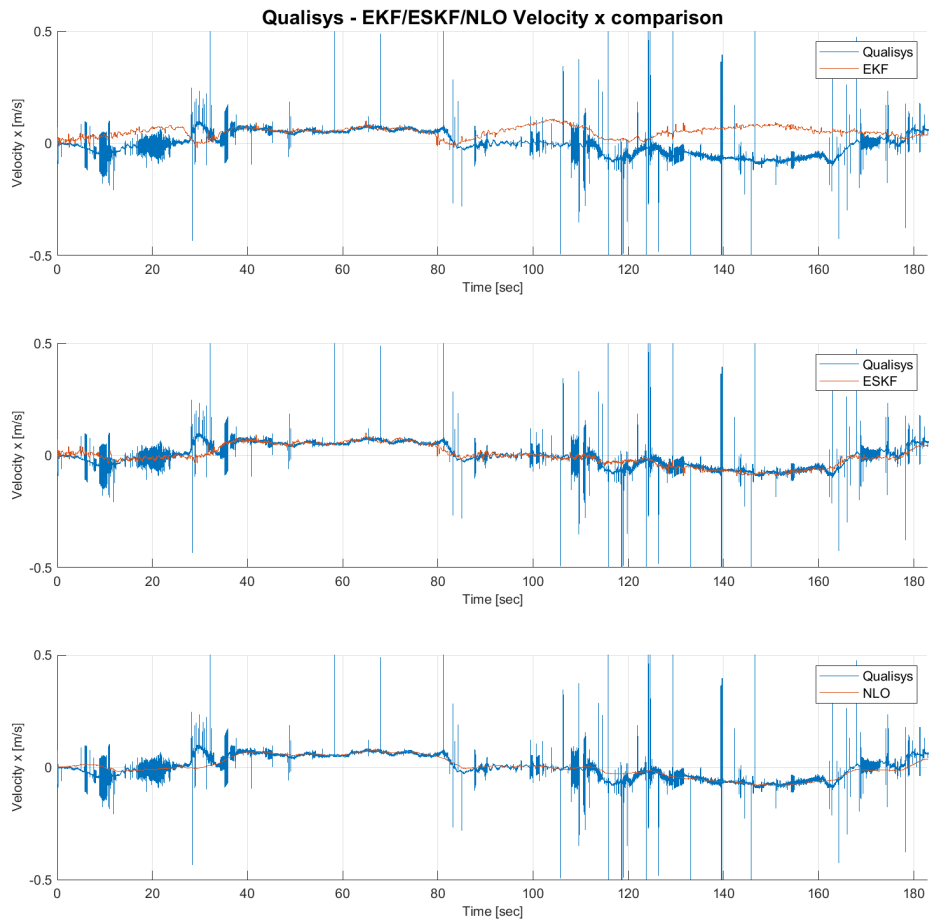


Figure 7.67: Velocity x comparison between the EKF, ESKF and NLO compared to Qualisys - underwater square

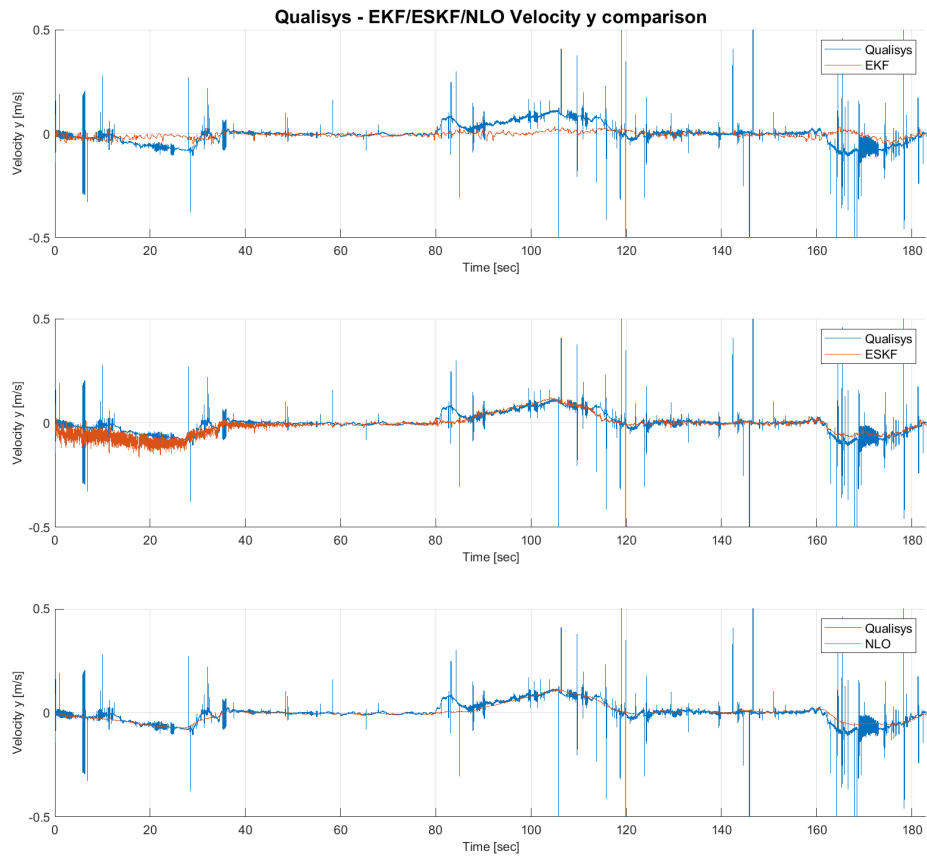


Figure 7.68: Velocity y comparison between the EKF, ESKF and NLO compared to Qualisys - underwater square

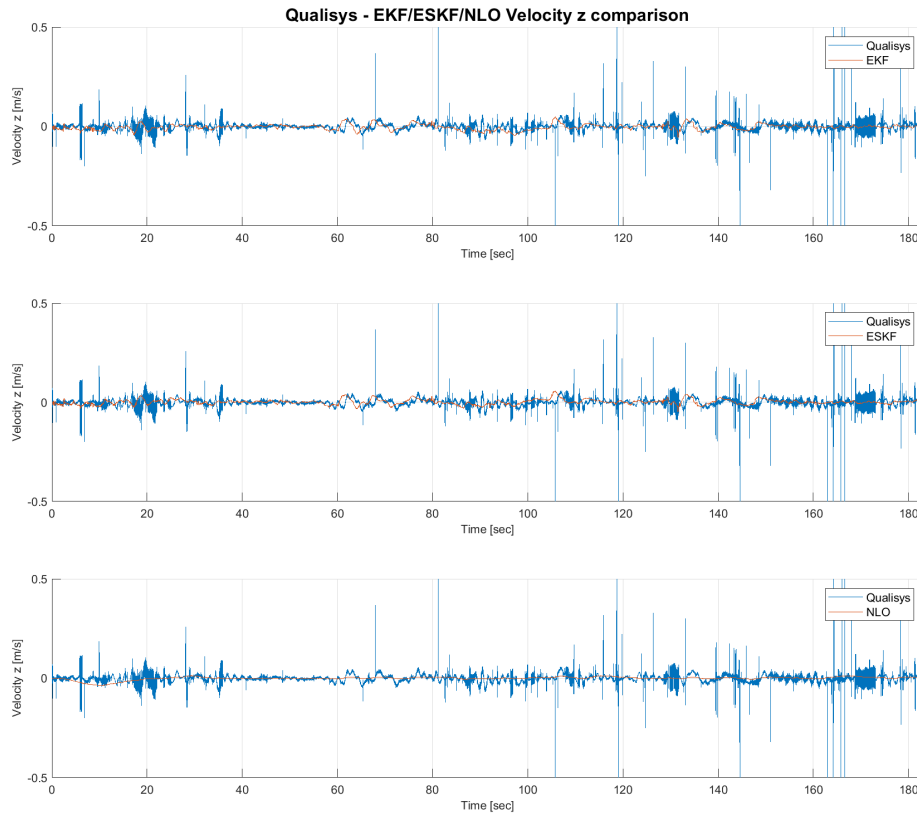


Figure 7.69: Velocity z comparison between the EKF, ESKF and NLO compared to Qualisys - underwater square

Moving over to the velocity figures in 7.67, 7.47 and 7.48 it seems like the y velocity of ESKF still has the same oscillating behavior issue as with the underwater-sinus test. Because of this, this behavior may be due to the person holding the pole while moving Manta-2020, which can give slight movement in velocity. This also give some spikes in the accelerometer. This could have been due to the wild point filter was not tuned for such small movements.



Figure 7.70: Roll comparison between the EKF, ESKF and NLO compared to Qualisys - underwater square

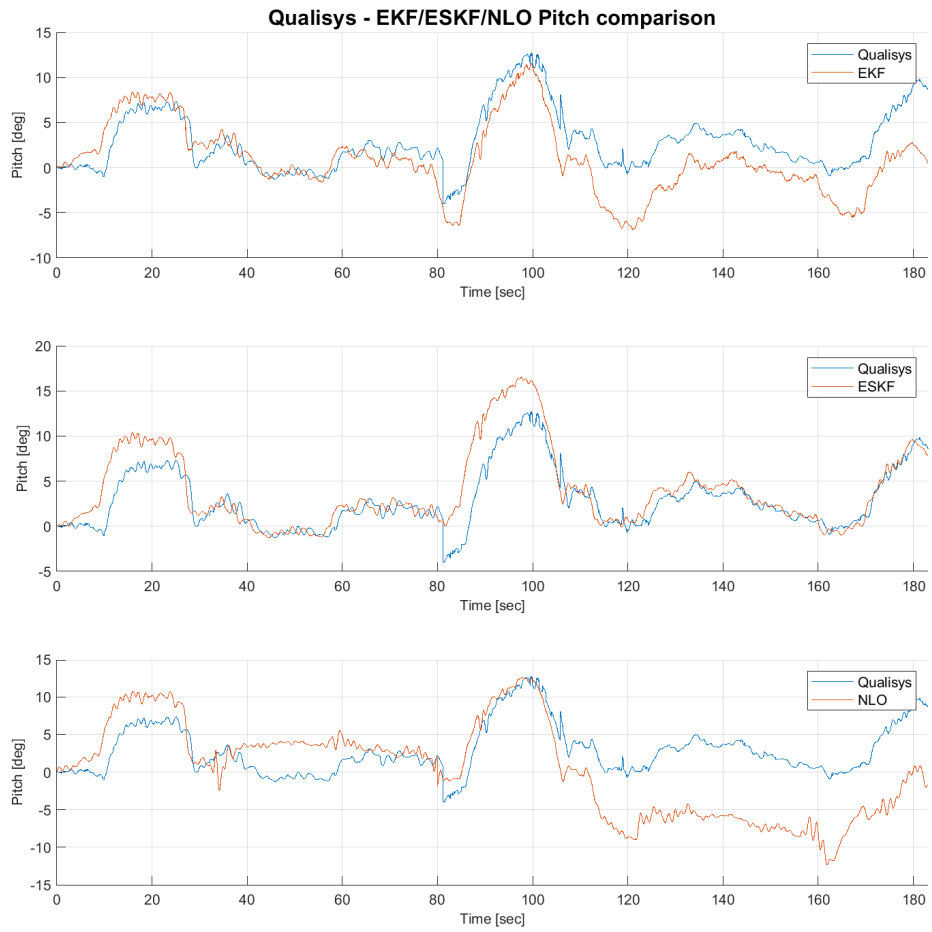


Figure 7.71: Pitch comparison between the EKF, ESKF and NLO compared to Qualisys - underwater square

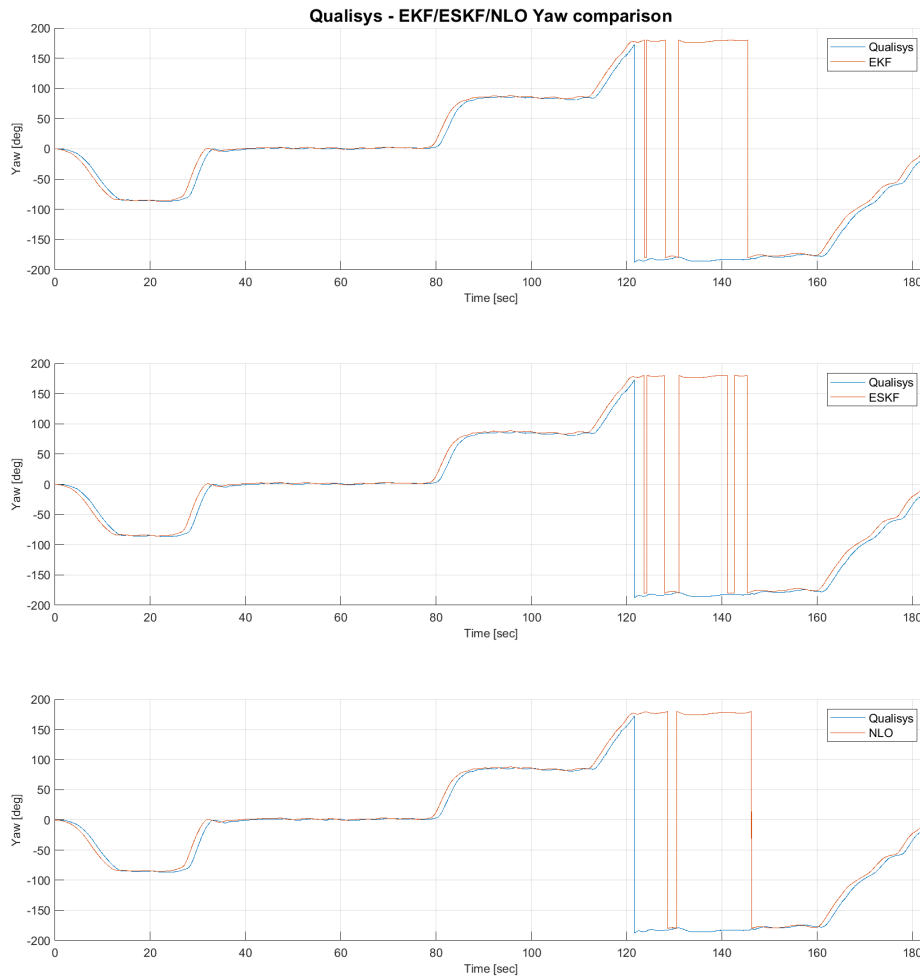


Figure 7.72: Yaw comparison between the EKF, ESKF and NLO compared to Qualisys - underwater square

Going over to the attitude plots in the figures 7.70, 7.71 and 7.72 it is observed that the NLO has almost the worst performing roll and pitch estimates compared to Qualisys. In comparison to the underwater sinus test this was not the case. This was probably due to the spike occurring for the y velocity that altered the pitch and yaw estimates of the EKF and ESKF.

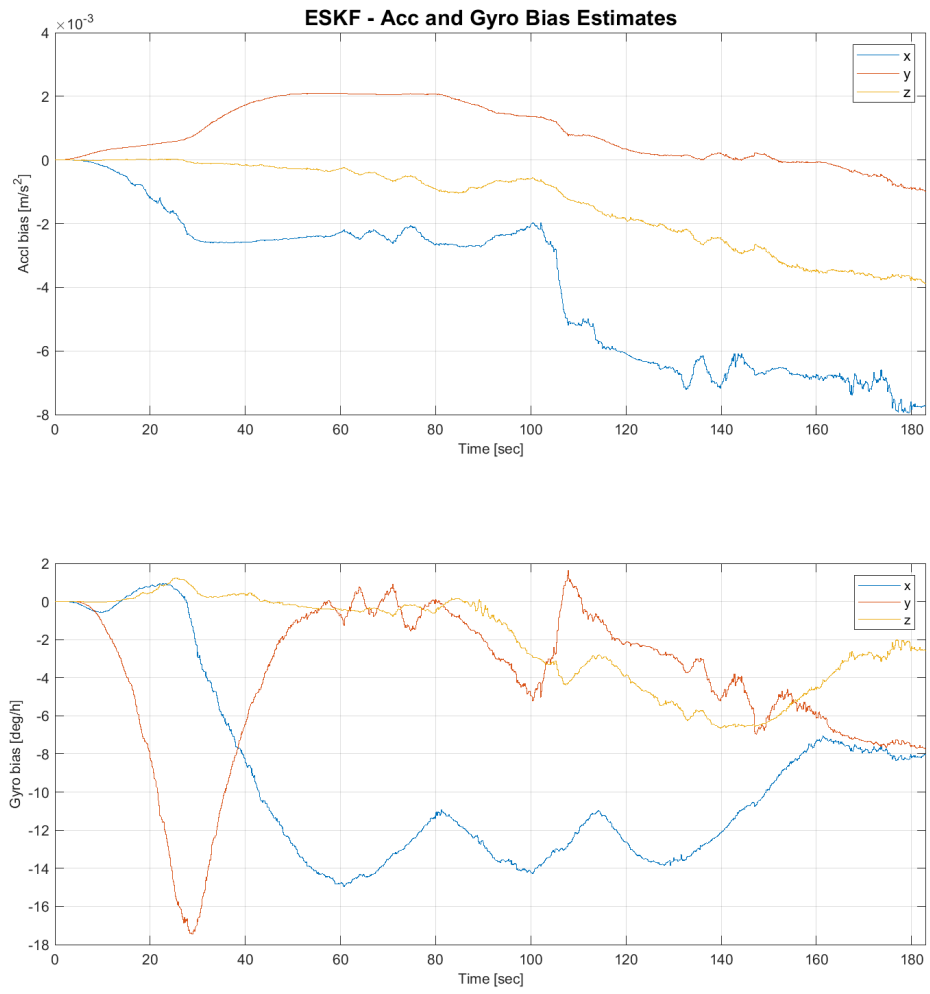


Figure 7.73: ESKF bias estimates - underwater square

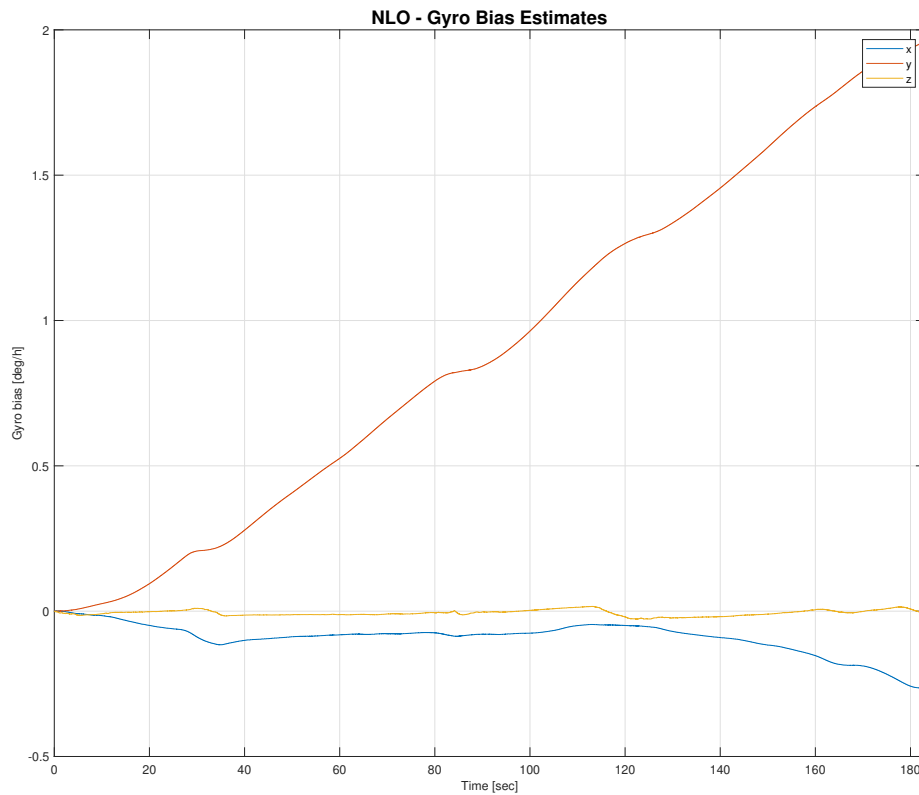


Figure 7.74: NLO bias estimates - underwater square

Shown in figure 7.73 and 7.74, there is a very different behavior of the bias estimates than the earlier test experiments. For example the y gyro bias for the ESKF. This moves first down to roughly -17 deg/h before moving towards again. This seems like a very unpredictable behavior. This could have been due to the unsteady moving of the person moving Manta-2020 during the test. This seems like a logical explanation because the roll and pitch estimates of Qualisys are very oscillating. This issue could have been solved by tuning the ω threshold in the wild point filter. For the gyro bias estimates of the NLO, the same behavior of the y value are the same, with reasons discussed in the previous experiments. This time, however, the x value seems to not "lock" itself to the z value. This could also be due to the wild point filtering not taking effect.

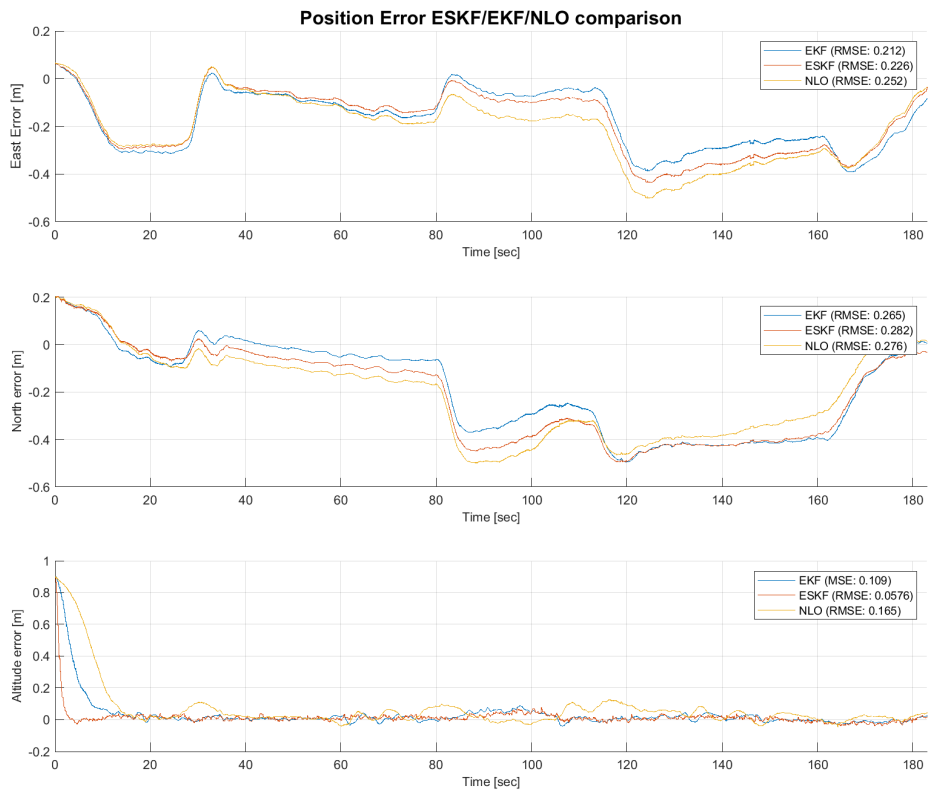


Figure 7.75: Position error of EKF,ESKF and NLO compared to Qualisys - underwater square

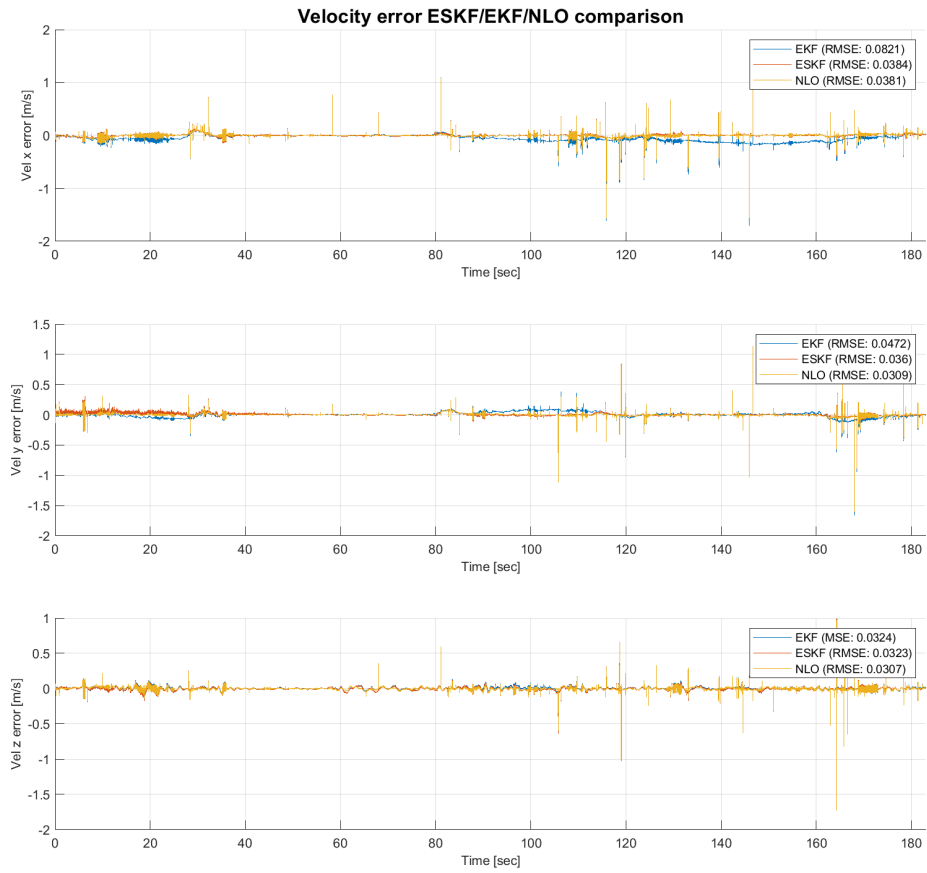


Figure 7.76: Velocity error of EKF,ESKF and NLO compared to Qualisys - underwater square

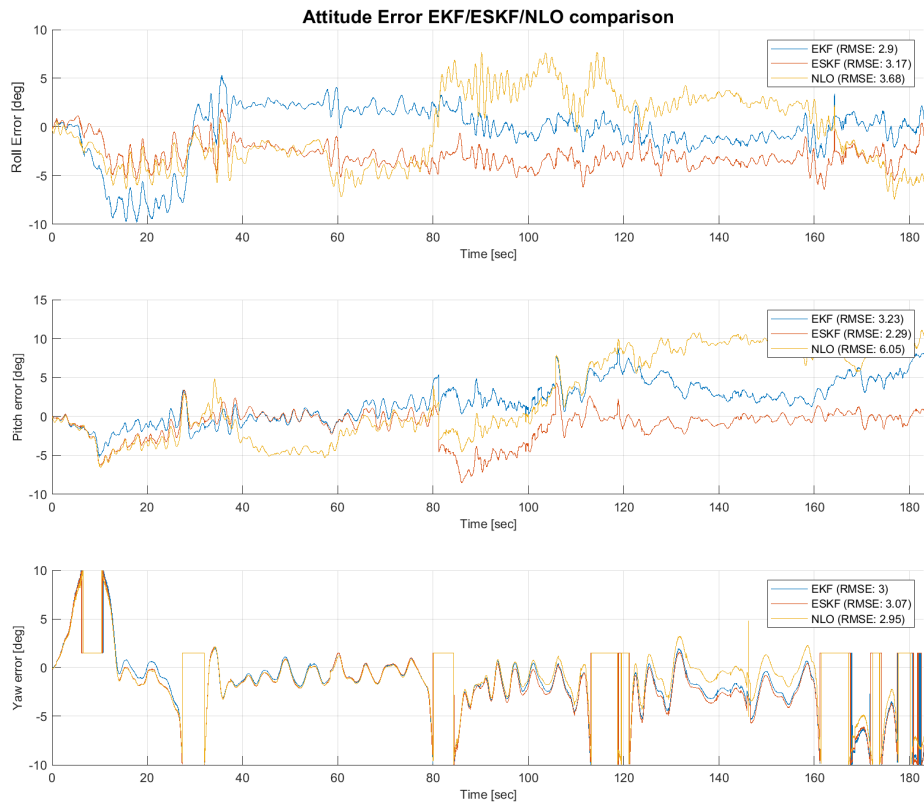


Figure 7.77: Attitude error of EKF,ESKF and NLO compared to Qualisys - underwater square

Observing the position error in figure 7.75, the EKF have the lowest RMSE for the horizontal positions. For the altitude, the relatively large RMSE for the NLO is because of the slow transient behavior. For the ESKF the unexpected behavior of the gyro bias estimates, could be the reason why its north and east positions could have contributed in greater RMSE than the EKF and NLO. However this does not seem to be the case, because of the low pitch and roll RMSE of the ESKF in figure 7.77. A more logical explanation would have been that the oscillating behavior of the y velocity of would have contributed to this error. But this seems to not be the case. Looking at figure 7.76 this seems also to not be the case. The ESKF has a smaller y velocity RMSE than the EKF. Then this position error could be due to tuning of the ESKF.

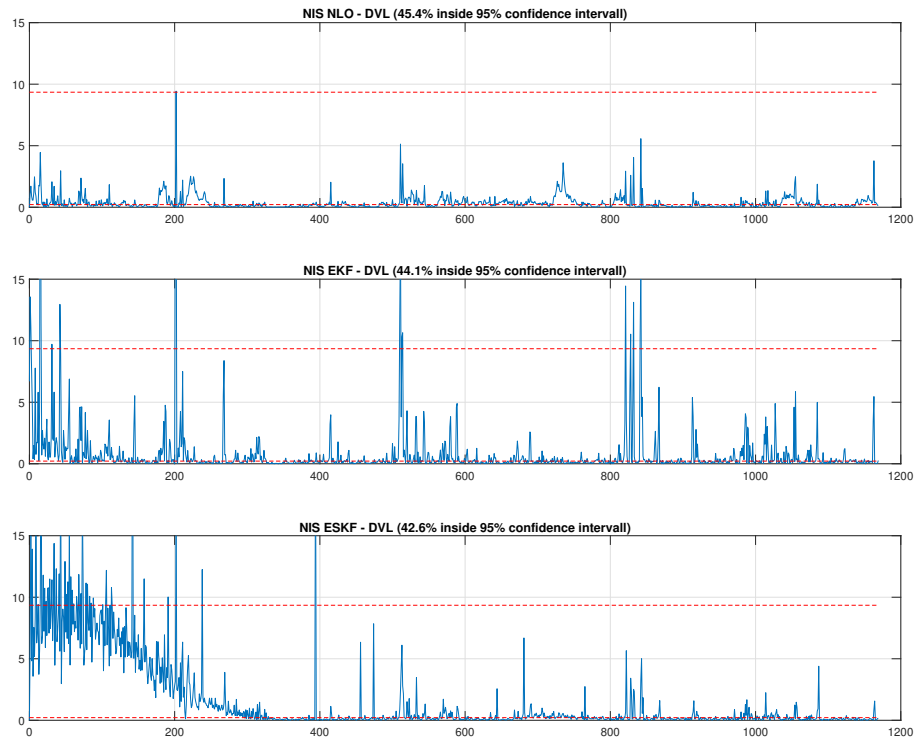


Figure 7.78: NIS DVL - EKF/ESKF/NLO comparison - underwater square

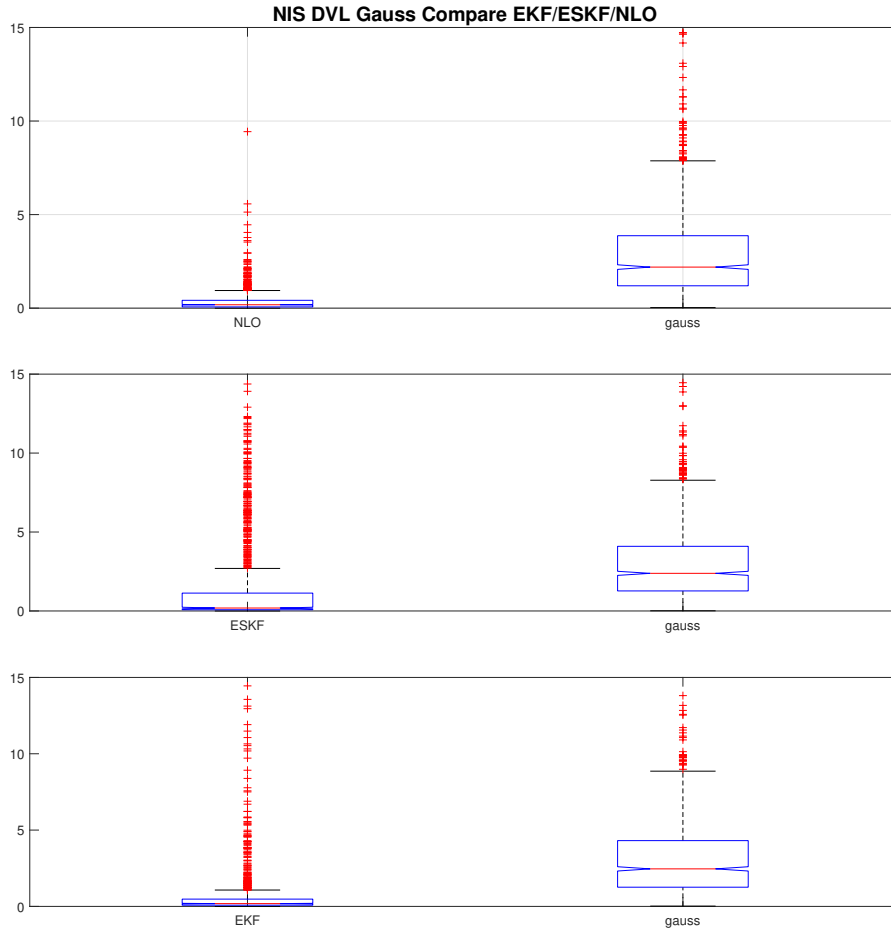


Figure 7.79: Gauss compare DVL - EKF/ESKF/NLO comparison - underwater square

Looking at the NIS DVL test in figure 7.78, the values in this test are most of the time well below the lower limit. This includes also the ESKF after ca 300 seconds. This is further seen in the Gauss compare figure 7.79. Here all three estimators are well below the Gauss, meaning that they will not diverge, but they are not estimating optimally. Also comparing this to the NIS compare in the underwater sinus test, the ESKF had much more of its values above the upper limit. This is due to the prolonged period of the oscillating behavior of the velocity in that test. In this test most of the dominant behavior of this stopped at around 40 seconds compared to almost the whole period of the sinus test.

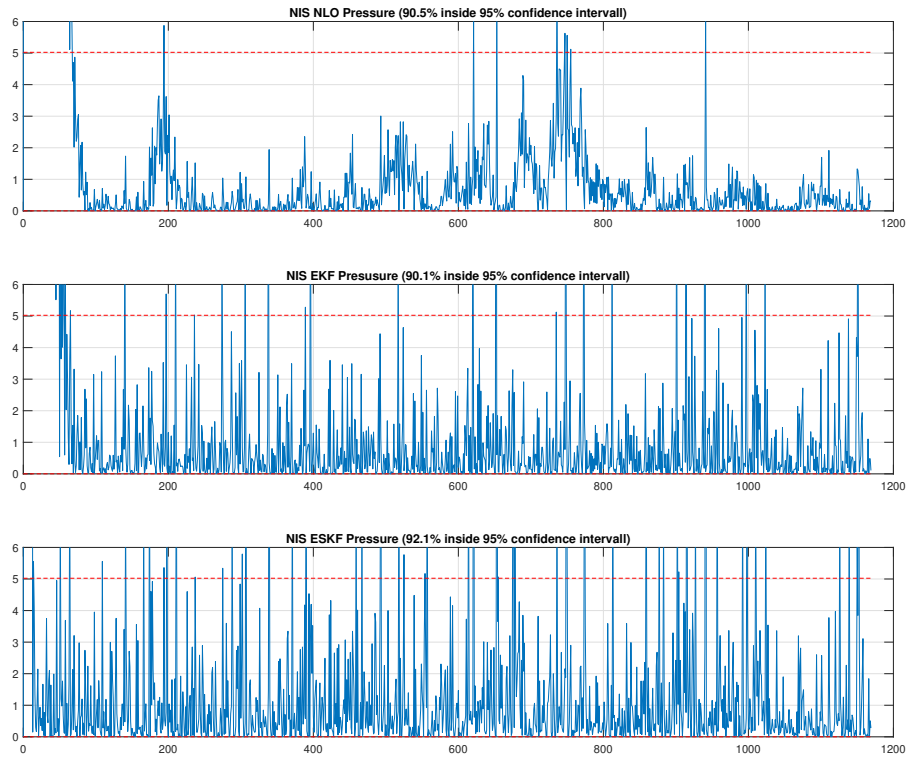


Figure 7.80: NIS pressure - EKF/ESKF/NLO comparison - underwater square

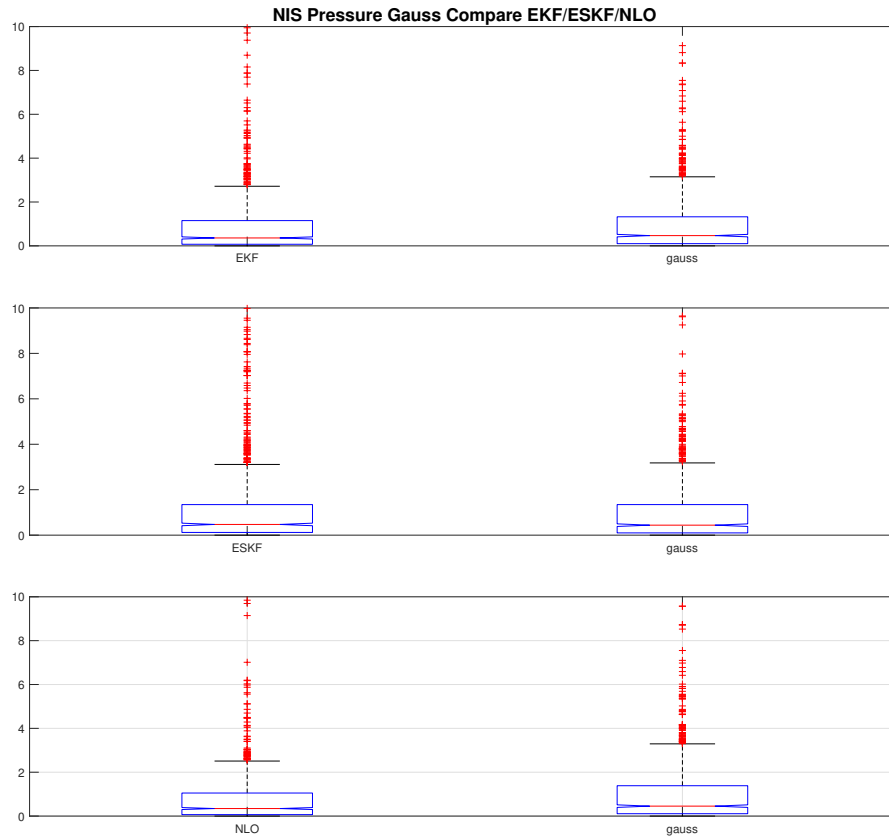


Figure 7.81: Gauss compare pressure - EKF/ESKF/NLO comparison - underwater square

For the NIS pressure values given in the figures 7.80 most of the values are well inside the 95 % confidence interval for the ESKF and NLO unlike the values for the underwater sinus test. This is most likely due to the alternating z values in the sinus test, and that the difference between the predicted state and the measurement of the pressure sensor are relatively large. Thus lowering the $\mathbf{r}_{pressure}$ and increasing the continuous time process noise covariance of the filters **D**.

For the Gauss pressure comparison in figure 7.81, all of the state estimators seem to resemble a very close distribution to the zero-mean Gaussian. This means the estimation errors of the filters are close to a modeled white noise, which indicates an optimal solution.

Conclusion

A real-time ESKF, NLO and EKF were successfully implemented and experimentally tested on Manta-2020. The tests included both underwater and "above" water tests scenarios from the MC-lab at Tyholt. This included also a 30 min test scenario to see if the filters were stable. Unfortunately the ESKF did not seem to satisfy this. This could have been because of an implementation bug, or more likely because of an mounting change in the IMU during the test. The NLO and EKF, on the other hand, prove to be stable for this test.

Looking at the results the EKF shows to have overall a NIS Gaussian that is closest to the zero-mean Gaussian. The NLO proves to be robust for corrupted velocity measurements of the DVL. The ESKF shows to have the fastest convergence of the altitude and give overall the most accurate velocity estimation. The filters were tuned to have the same initial state estimate $\hat{\mathbf{x}}_0$, error covariance \mathbf{P}_0 for the respective states and measurement covariances \mathbf{R}_{acc} , \mathbf{R}_{gyro} , \mathbf{R}_{dvl} and $r_{pressure}$ from the DVL, IMU and the pressure sensor.

The implemented sensor-synchronization proved to be successfully working for the ESKF and NLO. In addition the ESKF and NLO C++ implementations, successfully estimated real-time on Manta-2020. The wild point filter served to be working, but not tuned well for the behavior of Manta-2020.

Further work

To increase the accuracy of the state estimators, there are several aspects that can be further analyzed. For example in regards to safety and redundancy. Some safety features can include sensor redundancy, logging of measurements and more sophisticated signal processing that accounts for a frozen signal or signals with high or low variance as seen in figure 3.2. Other aspects of future work can include

- More sophisticated methods of aligning the Qualisys data with the ESKF/EKF data
- Error handling in the filters and in case of dropped measurements from the wild point filter. This could be warnings to the user, shutting down the ros-node if wrong filter parameters are used, etc...
- Coupling visual mapping with state estimation
- Newton-Laplace or Gauss-Newton optimization
- Implementation of a magnetometer, which can provide heading measurements.
- time synchronization with a timing board. For example a Sentiboard.

Appendix A

Links

Sensor STIM300 Driver: <https://github.com/vortexntnu/stim300>

Nortek DVL1000 Driver: https://github.com/vortexntnu/dvl_1000

ESKF implementation: <https://github.com/oyvind1501/ESKF>

NLO implementation: https://github.com/oyvind1501/nonlinear_observer

Wild point filter: https://github.com/oyvind1501/imu_wild_point_filter

Bibliography

- [1] Øyvind Denvik, “Project thesis - sensor fusion for an autonomous underwater vehicle,” Dec. 2019.
- [2] J. Solà, “Quaternion kinematics for the error-state kalman filter,” <http://www.iri.upc.edu/people/jsola/JoanSola/objectes/notes/kinematics.pdf>, October 12, 2017.
- [3] T. A. J. Haavard Fjaer Grip, Thor I. Fossen and A. Saberi, “Nonlinear observer for gnss-aided inertial navigation with quaternion-based attitude estimation,” pp. 272 – 279, Jun. 2013.
- [4] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion control*. Wiley, A John Wiley & Sons, Ltd, 2011.
- [5] S. K. R. Solberg, “Manta v1: A deliberative agen software architecture for autonomous underwater vehicles,” Master’s thesis, NTNU, Feb. 2020.
- [6] T. H. Bryne, “Ttk5 kalman filtering and navigation,” Sep. 2019.
- [7] D. Nguyen, “Signal processing - signal quality checking and fault detection,” *Department of Marine Technology - NTNU*.
- [8] Sensoror, *DATASHEET STIM300 Inertia Measurement Unit*, Sensoror-AS, May 2013. [Online]. Available: <https://www.sensoror.com/media/1132/ts1524r9-datasheet-stim300.pdf>
- [9] Nortek, *NORTEK MANUALS, DVL Integrator’s guide*, Nortek AS, 2017. [Online]. Available: <http://www.nortek.no/lib/manuals/dvl-integrators-guide>
- [10] J. Kelly and S. Waslander, “Lesson 5: Limitations of the ekf - module 2: State estimation - linear and nonlinear kalman filters.” [Online]. Available: <https://www.coursera.org/lecture/state-estimation-localization-self-driving-cars/lesson-5-limitations-of-the-ekf-OCrZc>
- [11] R. B. Wynn, V. A. Huvenne, T. P. L. Bas, B. J. Murton, D. P. Connelly, B. J. Bett, H. A. Ruhl, K. J. Morris, J. Peakall, D. R. Parsons, E. J. Sumner, S. E. Darby, R. M. Dorrell, and J. E. Hunt, “Autonomous underwater vehicles (auvs): Their past, present and future contributions to the advancement of marine geoscience,” *Marine Geology*, vol. 352, pp. 451 – 468, 2014, 50th Anniversary Special Issue. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0025322714000747>
- [12] I. Robonation, “<https://robonation.org/about/>,” 2020, accessed: 09.05.2020.
- [13] Robonation, *23-rd annual robosub 2020 Task ideas*, Robonation inc, 2020. [Online]. Available: <https://robonation.org/app/uploads/sites/4/2019/10/RoboSub-2020-Task-Ideas.pdf>

- [14] —, “22-nd annual international robosub competition - mission and scoring,” Naval Information Warfare Center Pacific, Tech. Rep., Jun. 2019.
- [15] I. RoboNation, Aug. 2014, accessed: 09.05.2020. [Online]. Available: [youtube.com/watch?v=UYG_r7b2Uic](https://www.youtube.com/watch?v=UYG_r7b2Uic)
- [16] *Autonomous Underwater Vehicle - AUV The HUGIN family*, Kongsberg, accessed 29.06.2020. [Online]. Available: <https://www.kongsberg.com/globalassets/maritime/km-products/product-documents/hugin-family-of-auvs>
- [17] O. K. H. K. V. Bjørn Jalving, Kenneth Gade, “A toolbox of aiding techniques for the hugin auv integrated inertial navigation system,” pp. 1146 – 1153, 2003.
- [18] K. S. A. W. R. S. Bjorn Jalving, Kenneth Gade, “Dvl velocity aiding in the hugin 1000 integrated inertial navigation system,” *Norwegian Defence Research Establishment (FFI)*. [Online]. Available: https://www.navlab.net/Publications/DVL_Velocity_Aiding_in_the_HUGIN_1000_Integrated_Inertial_Navigation_System.pdf
- [19] R. Mills, “Webinar - hugin superior: An introduction to superior productivity,” 2020. [Online]. Available: <https://www.kongsberg.com/maritime/products/marine-robotics/autonomous-underwater-vehicles/AUV-hugin/>
- [20] J. McCarthy, “Introduction to theoretical kinematics,” *MIT press*, 1990.
- [21] H. Edmund Brekke, “Fundamentals of sensor fusion,” First Edition, October 13, 2019.
- [22] O. J. Woodman, “An introduction to inertial navigation,” University of Cambridge - computer laboratory, Cambridge CB3 0FD United Kingdom, techreport 696, Aug. 2007.
- [23] N. TR8350, “Department of defense world geodetic system 1984 - its definition and relationships with local geodetic systems,” Tech. Rep., 2000.
- [24] K. Gade, “Introduction to inertial navigation and kalman filtering.” Tutorial for IAIN World Congress, Stockholm, Sweden, Oct. 2009.
- [25] W. Flenniken, J. Wall, and D. Bevely, “Characterization of various imu error sources and the effect on navigation performance,” in *Ion Gns*, 2005, pp. 967–978. [Online]. Available: <https://pdfs.semanticscholar.org/1f07/13f093a208f11752c669e53ef06e2654803c.pdf>
- [26] B. Sæther, “Development of a robust navigation system and adaptive motion control for milliampère,” Master’s thesis, NTNU, Jun. 2019.
- [27] I. Blue Robotics, *T200 thrusters*, accessed 09.05.2020. [Online]. Available: <https://bluerobotics.com/store/thrusters/t200-thruster/>
- [28] Sensoror, *Product Brief STIM300 Inertial Measurement Unit*, Sensoror-AS, Aug. 2017. [Online]. Available: <https://www.sensoror.com/products/inertial-measurement-units/stim300/>
- [29] *Low-light HD USB camera*, Blue Robotics, accessed 04.06.2020. [Online]. Available: <https://bluerobotics.com/store/sensors-sonars-cameras/cameras/cam-usb-low-light-r1/>
- [30] *FLIR Blackfly S USB3*, FLIR, 2020, accessed 04.06.2020. [Online]. Available: <https://www.flir.com/products/blackfly-s-usb3/>
- [31] *Technical Reference FLIR Blackfly S BFS-U3-16S7*, FLIR Integrated Imaging Solutions, Nov. 2019, accessed 04.06.2020.
- [32] J. G. Jean Christophe Lawson, Robert Leydier, “High-speed inter-chip usb electrical specification,” Universal Serial Bus Specification Supplement, techreport, Sep. 2007.

- [33] Nortek, “Dvl1000,” accessed 03.06.2020. [Online]. Available: <https://www.nortekgroup.com/products/dvl-1000-300m>
- [34] L. L. Michailas Romanovas, Ralf Ziebold, “A method for imu/gnss/doppler velocity log integration in marine applications,” *Institute of Communications and Navigation, German Aerospace Centre (DLR), Neustrelitz, Germany*, p. 8, Oct. 2015.
- [35] *Odroid XU4 - Hardware specification*, Hardkernel CO, Ltd, accessed 09.05.2020. [Online]. Available: <https://wiki.odroid.com/odroid-xu4/hardware/hardware>
- [36] *Jetson TX2 module*, NVIDIA corporation, accessed 05.06.2020. [Online]. Available: <https://developer.nvidia.com/embedded/jetson-tx2>
- [37] O. S. R. Foundation, “Ros/concepts,” accessed 07.06.2020. [Online]. Available: <http://wiki.ros.org/ROS/Concepts>
- [38] R. documentation, “tf.” [Online]. Available: http://wiki.ros.org/tf#static_transform_publisher
- [39] T. Moore and D. Stouch, “A generalized extended kalman filter implementation for the robot operating system,” in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.
- [40] D. S. Thomas Moore, “A generalized extended kalman filter implementation for the robot operating system,” Sep. 2016. [Online]. Available: https://github.com/cra-ros-pkg/robot_localization/blob/melodic-devel/doc/robot_localization_ias13_revised.pdf
- [41] R. documentation, “tf.” [Online]. Available: http://wiki.ros.org/tf#static_transform_publisher
- [42] A. Valibeygi, M. H. Balaghi I., and K. Vijayaraghavan, “A comparative study of extended kalman filter and an optimal nonlinear observer for state estimation,” in *2017 American Control Conference (ACC)*, 2017, pp. 5211–5216.
- [43] J. Skaloud, “Problems in direct-georeferencing by ins/dgps in the airborne enviroment,” *Swiss Federal Institute of Technology (EPFL), Institute of Geomatics, CH-1015 Lausanne, Switzerland*, 1999.
- [44] J. F. Kurose and K. W. Ross, *Computer networking: A top-down approach*, 7th ed. Pearson: 7 edition, 2016.
- [45] D. of Marine Technology, “Marine cybernetics laboratory (mc-lab),” accessed 10.06.2020. [Online]. Available: <https://www.ntnu.edu/imt/lab/cybernetics>
- [46] *QTM MARINE MANUAL*, Qualisys AB, Packhusgatan 6, Gotenburg, SWEDEN, Nov. 2008, accessed 18.06.2020. [Online]. Available: https://home.hvl.no/ansatte/gste/ftp/Marinlab_files/Manualer_utstyr/QTM%20Marine%20manual%2020081211.pdf