NTNU

Norwegian University of
Science and Technology

Fault Detection for Position Mooring Using Statistical Analysis

Andreas Bro Kolstø

Marine Technology
Submission date: June 2018
Supervisor: Vahid Hassani, IMT

Norwegian University of Science and Technology
Department of Marine Technology

Abstract

The standard approach for detecting mooring line failures for position moored vessels is by measuring the tension in the mooring lines, either directly or indirectly. The sensors measuring the line tensions are often hard to maintain due to being mounted under water, and determining if a loss of tension is a line failure or a sensor failure can be difficult. This thesis investigates a different approach, using statistical analysis of only the position of the vessel to detect mooring line failures.

This is achieved by creating a mathematical model of the vessel motion for each of the possible failure scenarios, in addition to the no failure scenario. For each scenario a passive observer is implemented based on the appropriate mathematical model. The observers estimate the states of the vessel. The quality of the estimated states will depend on how the actual behaviour of the vessel compares to what is predicted by the model assuming a certain failure scenario.

Two different methods are used to analyse these estimates to determine which scenario is believed to be correct: dynamic hypothesis testing (DHT) and maximum likelihood estimation (MLE). In short, DHT calculates the probability of each scenario being true, and MLE calculates the likelihood of each scenario. In simulations both methods show promising results in their ability to detect mooring line failures. Failures are detected within a couple of minutes, when subjected to both wave and current disturbances. However, an implementation error in the simulator used causes the results to not be directly transferable to a real world scenario.

Sammendrag

Den vanligste metoden for å detektere ankerlinebrudd på forankrede fartøy er å måle strekket i selve ankerlinene, enten direkte eller indirekte. Noen av utfordringene med dette er at måleinstrumentene som måler strekk ofte er vanskelige å vedlikeholde fordi de er montert under vann. I tillegg kan det ofte være vanskelig å være sikker på om en måling som viser at det ikke er noe strekk i ankerlinen er forårsaket av et brudd eller bare sensorsvikt. Denne oppgaven ser på en annen metodikk for å detektere ankerlinebrudd, ved å bruke statistisk analyse av kun posisjonen til fartøyet.

Dette oppnås ved å lage en matematisk modell av fartøyet for hver av de mulige sviktscenarioene, i tillegg til scenarioet hvor alle linene er intakte. For hvert scenario implementeres en passiv observator basert på den aktuelle matematiske modellen. Observatorene estimerer tilstanden til fartøyet, basert på antagelsen om at det gitte scenarioet er riktig. Kvaliteten på dette estimatet vil avhenge av hvordan den reelle bevegelsen til fartøyet er sammenlignet med hva modellen for det aktuelle sviktscenariet forutser.

To forskjellige metoder ble anvendt for å analysere disse estimatene for å avgjøre hvilket scenario som var korrekt: dynamisk hypotesetesting og maksimal sannsynlighetsestimering. Kort fortalt vil begge metodene, for hvert scenario, beregne sannsynligheten for at det gitte scenariet er riktig. Dette kan brukes til å finne det mest sannsynlige scenarioet. I simuleringer virker begge metodene lovende. De klarer å detekere ankerlinebrudd i løpet av få minutter når fartøyet er utsatt for både bølge- og strømningslaster. Dessverre gjør en feil i implementeringen av simulatoren at disse resultatene ikke er direkte overførbare til den virkelige verden.

Contents

Abstract i

Sammendrag iii

List of figures vii

List of tables x

1 Introduction 1

2 Existing and considered solutions 3
2.1 Existing Fault Detection Systems for Position Mooring 3
2.1.1 Direct Tension Measurement 4
2.1.2 Indirect Tension Measurement 4
2.1.3 Sonar 5
2.1.4 Visual Inspection 6
2.1.5 Offset Monitoring 6
2.2 Systems Under Consideration 6
2.2.1 Hull Mounted Sonar 6
2.2.2 Seabed Sonar 7
2.2.3 Depth Sensor 7
2.2.4 Response Learning System 7
2.3 Dynamic hypothesis testing 9

3 Theory 11
3.1 Probability 11
3.2 Gaussian Probability Density Function 12
3.3 System Model 12
3.3.1 Estimating Mooring Forces 14
3.4 Passive Observer 15

3.5 Dynamic Hypothesis Testing 16
3.6 Maximum Likelihood Estimation 18

4 Methods **21**
4.1 System configuration 23
4.2 Simulator 25
4.3 PI controller 25
4.4 Estimation of error covariance 26
4.5 Tuning the passive observer 26
4.6 Implementation in MATLAB 28
4.6.1 S-functions 28
4.6.2 Dynamic hypothesis testing 29
4.6.3 Maximum likelihood estimation 30
4.6.4 Passive Observer 32
4.6.5 Initialisation 32

5 Results **35**
5.1 No breakage 35
5.2 No breakage with current 40
5.3 Line 6 breaking 44
5.4 Line 6 breaking with current 47
5.5 Line 7 breaking 49
5.6 Line 7 breaking with current 54
5.7 Line 8 breaking 56
5.8 Line 8 breaking with current 59

6 Discussion **63**
6.1 Performance 63
6.1.1 General performance 63
6.1.2 Performance with current 64
6.2 Limitations 65
6.3 Simulator implementation error 66

7 Conclusion **67**
7.1 Further work 68

Bibliography **69**

Appendices **73**

A MATLAB code **75**
A.1 Dynamic Hypothesis Testing 75

A.2 Maximum Likelihood Estimation 79
A.3 Initialisation 83
A.4 Observer initialization 84
A.5 Generating error covariances 87

List of Figures

2.1 Depth sensor arrangement 8
2.2 Flowchart of response learning system 9

3.1 Reference frames 15

4.1 Flowchart MATLAB/Simulink implementation 22
4.2 System overview 24
4.3 MLE and DHT blocks in Simulink 29
4.4 Unbounded negative log-likelihoods 32
4.5 Implementation of passive observer in Simulink 33

5.1 Vessel position, no line breakages 36
5.2 Time plot of vessel position, no line breakages 37
5.3 Position residuals, no line breakages 37
5.4 Estimated scenarios, no line breakages 38
5.5 Hypothesis probabilities, no line breakages 39
5.6 Negative log-likelihoods, no line breakages 40
5.7 Vessel position with current, no line breakages 41
5.8 Estimated scenarios with current, no line breakages 42
5.9 Hypothesis probabilities with current, no line breakages 43
5.10 Vessel position, line 6 break 44
5.11 Position residuals, line 6 break 45
5.12 Estimated scenarios, line 6 break 45
5.13 Vessel position with current, line 6 break 47
5.14 Estimated scenarios with current, line 6 break 48
5.15 Negative log-likelihoods with current, line 6 break 49
5.16 Vessel position, line 7 break 50
5.17 Time plot of vessel position, line 7 break 51
5.18 Estimated scenarios, line 7 break 52

5.19 Negative log-likelihoods, line 7 break 53
5.20 Hypothesis probabilities, line 7 break 53
5.21 Vessel position with current, line 7 break 54
5.22 Estimated scenarios with current, line 7 break 55
5.23 Negative log-likelihoods with current, line 7 break 56
5.24 Vessel position, line 8 break 57
5.25 Estimated scenarios, line 8 break 58
5.26 Vessel position with current, line 8 break 59
5.27 Estimated scenarios with current, line 8 break 60
5.28 Negative log-likelihoods with current, line 8 break 61

List of Tables

4.1 Principle dimensions of simulated FPSO 23
4.2 Position of mooring line anchors 23
4.3 Disturbance parameters 25

Chapter 1

Introduction

Many marine operations that are carried out by floating installations or vessels, like floating production vessels for oil and gas extraction or ROV vessels, require that the vessel keeps its position fixed. For shorter duration operations the use of dynamic positioning (DP) systems is commonplace. Vessels fitted with such systems are able to keep their position by using only the thrusters on the vessel. However some operations, especially within oil and gas, require the vessel to stay stationary for several years. In these cases a mooring system is often used, as it is a passive system as opposed to the active DP system. Some vessels will combine the best of both worlds, using a position mooring system to keep a fixed position, and a DP system with thrusters to give more damping or more fine tuned position control to the system. These types of systems are referred to as thruster assisted position mooring systems (TAPM).

Any TAPM or position mooring system will obviously be affected by a failure of any of the mooring lines. Such a failure causes a loss of tension on the vessel, which could lead to a loss of position. For an oil and gas vessel with risers or a drill string that could break, a position loss could cause a huge environmental disaster. In severe weather conditions a break in one line could lead to a cascading failure in the other lines, potentially causing the vessel to drift and put the crew at risk. This makes it critical to be able to detect mooring line failures. Since a TAPM vessel has thrusters available it is possible to use them to counteract the loss of a mooring line. If the failure is discovered quickly the DP system can be used to avoid serious consequences.

Most of the current systems available for mooring line monitoring are based on measuring the tension in the mooring lines. This can be done directly be fitting

a tension sensor to the line, or indirectly for example by measuring the angle of the line and calculating the tension. Other types of systems use sonar to detect the position and/or angle of the lines. In common for almost all these existing systems is that they rely on sensors mounted under water. This means that both installation and maintenance requires expensive and complicated operations involving divers or ROVs, and makes it difficult to determine if an alarm signal is caused by a sensor failure or a line failure. In addition, according to a survey of floating production systems (FPS) in the north sea, there are many vessels that don't have any monitoring of their mooring lines.

Because existing mooring line solutions are expensive and many vessels don't have any, there is potential for alternative solutions. This thesis investigates the possibility of detecting mooring line failures using statistical analysis. The goal is to be able to detect whether or not any mooring lines have failed by analysing only the position of the vessel. This removes the need for underwater sensors, and the associated installation and maintenance costs. By analysing the behaviour of the vessel compared to what behaviour is expected for different failure scenarios, the statistical analysis methods will produce the probabilities for whether or not the failure scenarios are true. If this approach is shown to work, it will eliminate the need for underwater sensors, greatly simplifying maintenance. Because it is a software solution rather than a hardware solution installation costs are also kept low, and it should be possible to retrofit the system to vessels that currently have no form of mooring line monitoring.

This thesis will implement two different statistical methods: dynamic hypothesis testing and maximum likelihood estimation. They will be used to try to detect single line failures for a TAPM vessel.

Chapter 2

Existing and considered solutions

2.1 Existing Fault Detection Systems for Position Mooring

There are a variety of systems available for monitoring the status of mooring lines. Most of these are based on measuring, or estimating, the tension in each line. Some systems are only able to detect whether the mooring lines are present or not, whilst more sophisticated systems are able to estimate the condition of the mooring lines by analysing the time history of the line measurements. Almost all of the sensor systems currently in use include sensors installed under water, requiring diving or ROV operations to install and maintain. It also means the sensors are exposed to sea loads, and they must therefore be built to withstand tough conditions.

One survey showed that only 50% of floating production systems (FPSs) in the North Sea could monitor line tensions in real time. While 67% of FPSs were able to measure offsets from the no-load equilibrium position, only 22% of FPSs had line failure alarms. This indicates that the level of instrumentation is not as high as might be expected for such a heavily regulated sector. (Noble Denton Europe Limited, 2006)

2.1.1 Direct Tension Measurement

Perhaps the simplest and most obvious way to detect if a mooring line has failed is to directly measure the tension in the mooring line itself. This can be done with several different methods. For example a load measuring shackle can be attached to the mooring line, which measures the tension and transmits it to the vessel. Another method utilizes the fact that when tension is placed on a chain link, the link will deform. This causes the two parallel sides to pull towards each other. By measuring the distance between them it is possible to estimate the tension. The accuracy will be affected by the wear and tear on the chain, as the stress pattern of the links will change. However, unlike the inline shackle, it is suited for retrofit instalment. (GL Noble Denton, 2017)

A problem with direct tension measurement is how to interpret a loss of tension, as this could just as easily be a sensor failure as a line breakage. To avoid false alarms, it is possible to look at the signals from all the mooring lines. In an actual failure scenario, one would expect to see changes in the signals from the other lines as well, as they would be loaded differently. This could be used to determine if there was an actual breakage or just a sensor failure (GL Noble Denton, 2017). However this is not necessary an easy task. In one real world scenario it took the two weeks of processing the data from the other lines to determine if the failure was genuine (Noble Denton Europe Limited, 2006).

Another problem with tension measurement, that is also a concern for many other methods, is that some line breakages are harder to detect. Obviously if the line fails close to the vessel the tension goes to near zero. But if it fails along the part of the line that lies on the seabed, or in the touchdown zone, it is not always so clear. There is a lot of friction from the chain itself dragging through the mud on the seabed, even if it is not attached to the anchor. This means that there will still be a significant amount of tension measured at the vessel, making detection more difficult. This is especially the case when the weather is pushing the vessel towards the failed line, as one would expect a lower tension on the line in that scenario. Such failures might not have large consequences at the outset, but it leaves the vessel vulnerable without the knowledge of the crew. (GL Noble Denton, 2017)

2.1.2 Indirect Tension Measurement

A different approach to monitoring the state of the mooring lines is measuring the angle of the line as it leaves the vessel. Because of the relationship between the tension on the line and the shape of its catenary, it is possible to calculate

the tension from the measured angle by use of catenary equations. The angle of the cable is usually measured using an electronic inclinometer attached to the mooring line near the vessel. (GL Noble Denton, 2017)

It is also possible to estimate the tension in a chain by using the fact that as carbon steel is tensioned, its electromagnetic properties change. By placing a sensor on a chain link this property can be measured and sent back to the vessel. Obviously the sensor needs to be calibrated for the particular metallurgic composition of the chain link it is attached to. However the properties of the link might change over time with wear and tear. (GL Noble Denton, 2017)

Both of these methods are suited for retrofit instalment, as neither requires changing the existing mooring lines. However they also share many of the weaknesses of direct tension measurement.

2.1.3 Sonar

Using sonar imagery can be an effective way to get the status of the mooring lines, without installing sensors on every line separately. By deploying a 360 degree sonar beneath the vessel the positions of all mooring lines (and any risers) can be logged over time. This makes it possible to develop a model of where the lines are expected to be during normal conditions. This model can be used to create alarm conditions for both missing (i.e. broken) lines, as well as lines that are not in the correct position due to loss of tension caused by a breakage further down the line. (GL Noble Denton, 2017)

Some vessels have a deployable sonar probe on board, that can be used in fair weather conditions to check the status of the lines. Unlike the permanent installation this system will not be able to develop an expected position for the lines, and will therefore depend more on the skill of the operator for good interpretation of the data. It will also leave the crew without certain knowledge about the state of the lines between inspections. However, it is cheap to install and deploy this technology. (Noble Denton Europe Limited, 2006)

As with the direct tension measurement, this methodology will not be able to detect breakages in the mud, as in that case the line will still be present and under tension at the inspected location near the vessel. (Noble Denton Europe Limited, 2006)

2.1.4 Visual Inspection

For vessels where the connection points of the mooring lines are above the waterline, a simple visual inspection is possible. If there are any missing or untensioned lines, this will be immediately apparent. Visual inspection is also possible for lines attached underwater, by use of a mini ROV deployed from the vessel. This is however limited to fair weather. The accuracy of visual inspections highly depends on the skills of the inspector, as there are some failure scenarios that will only lead to very subtle changes at the vessel connection point. (Noble Denton Europe Limited, 2006) (GL Noble Denton, 2017)

2.1.5 Offset Monitoring

If a vessel is equipped with a positioning system, for example a GNSS system like GPS, it is possible to look at the vessels offset from its equilibrium position to determine if a line might be broken. However, unless the vessel is moored in very deep water, it is difficult to distinguish the offset caused by a broken line from the normal influence from environmental disturbances. The wind, wave, and current disturbances can also mask a line breakage, by providing a load that can make up for some of the missing force from the mooring line, thus reducing the offset. The largest value of this kind of system is perhaps giving the crew of the vessel an indication of the state of the system, so that if any suspicious behaviour is observed further action can be taken to determine if there is a failure. (Noble Denton Europe Limited, 2006)

2.2 Systems Under Consideration

2.2.1 Hull Mounted Sonar

An alternative to having a sonar probe or 360 degree sonar beneath the vessel is having a hull mounted sonar. In this system one or more sonar heads are angled down towards the mooring lines a short distance out from the connection point. This gives the sonar a strong echo, as the distance to the line is short, and the angle of incidence is close to perpendicular. Knowing the position of the sonar head it is possible to calculate the position and angle of the mooring line. As with the inclinometer, it is then possible to estimate the tension in the line. (GL Noble Denton, 2017)

One of the main advantages of this system is that it has lower maintenance costs. There is no need for a battery pack, as the system receives power from the vessel. The measurement signal can also be sent over fixed wires, which will not be as exposed to the elements as in other systems. This should reduce the need for diving or ROV operations to only the installation of the system. (GL Noble Denton, 2017)

2.2.2 Seabed Sonar

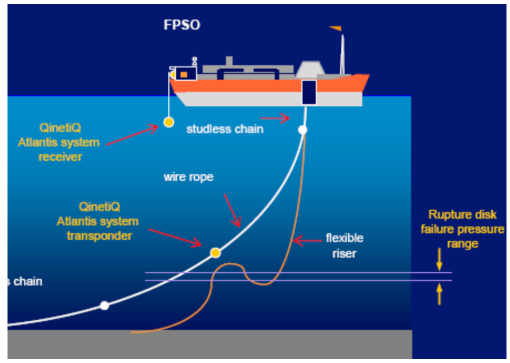
A sonar system deployed to the seabed is able to cover all mooring lines simultaneously. Unlike most other systems, it is able to provide a catenary shape for the mooring line, due to its distance from the line. This can be used to calculate the tension, and can also be used to detect a change in catenary profile associated with a failure in the mud (Noble Denton Europe Limited, 2006). Being placed at the seabed means the array will not be subjected to any environmental loads caused by winds or waves. It can also have a larger battery life than other sensors, as there are few practical restrictions on size. (GL Noble Denton, 2017)

2.2.3 Depth Sensor

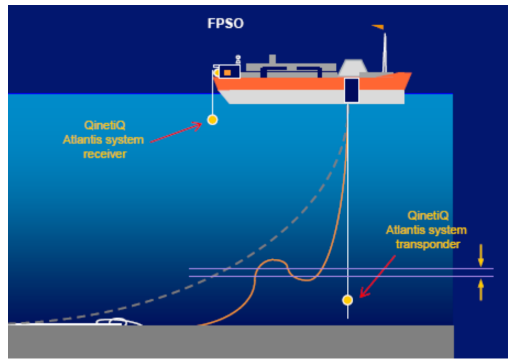
A passive method for detecting line breakages is using a depth activated sensor attached to the mooring line. The sensor unit is fitted with a seawater battery, which is activated when a rupture disk fails because of the external pressure. The pressure the rupture disk fails at determines the activation depth. If the line fails above the sensor, it will fall to the seabed along with the remaining line. If the line fails below it will lose its catenary shape, causing the sensor to swing down to a lower depth and being activated. This is shown in figure 2.1. This technology is unable to give information on the state of the mooring lines. However, its passive nature makes it very low maintenance. (GL Noble Denton, 2017)

2.2.4 Response Learning System

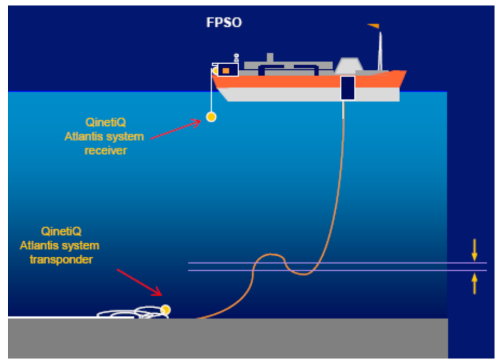
An automatic line failure detection system proposed by Noble Denton Europe Limited (2006) is the response learning system (RLS). Figure 2.2 shows an overview of the system methodology. This system uses a mathematical model of the vessel and mooring lines to predict the expected response of the vessel due to the measured applied environment, like environmental disturbances. By measuring the error between expected and actual behaviour, it is possible to revise the mathematical model coefficients to better represent the real behaviour. This



(a) Initial arrangement



(b) Failure below unit



(c) Failure above unit

Figure 2.1: Depth sensor arrangement, demonstrating ability to detect failures above and below sensor unit. (GL Noble Denton, 2017)

is the learning part of the method. An alarm is sounded when the motion of the vessel does not match the predicted behaviour. The major advantage of this approach is that there is no need for underwater sensors, and therefore no need for expensive maintenance using ROVs or divers. If there is a systems failure it can be easily diagnosed and repaired by the crew. It is also a relatively simple retrofit to existing vessels, not requiring expensive intervention work like with the inline sensors. (Noble Denton Europe Limited, 2006)

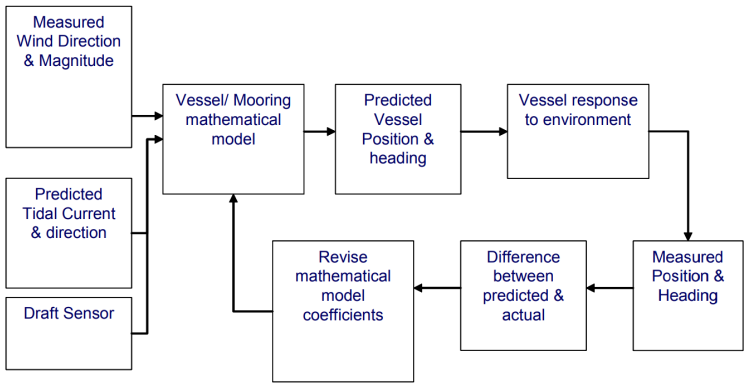


Figure 2.2: Flow chart showing the methodology of a response learning system. (Noble Denton Europe Limited, 2006)

2.3 Dynamic hypothesis testing

Hassani et al. (2018) proposes a statistical analysis method for detecting mooring line failures for thruster assisted position mooring vessels, when no line tension measurements are available. The method uses the dynamic hypothesis testing (DHT) approach, which involves creating a series of hypotheses of what the true state of the position mooring system is. An example hypothesis could be that mooring line 3 has failed, or that all mooring lines are intact. One hypothesis is created for each failure scenario that is evaluated. Using DHT it is possible to calculate the probability of each hypothesis being true by creating a mathematical model for each hypothesis, where the model assumes the given hypothesis is true. The actual behaviour of the vessel is then compared to the models expected behaviour. This is done by calculating the probability of getting the last position measurement, given the known previous measurements and control inputs, and assuming the given hypothesis to be true. By doing this calculation for all the

hypotheses it can be calculated which hypothesis has the highest probability of being correct, given the observed behaviour of the vessel.

In many ways this approach is similar to the response learning system. However the DHT approach does not revise the mathematical model itself, instead analysing the output from multiple predetermined models.

As this approach is also used in this thesis, the details of the DHT method can be seen in section 3.5. The main differences are that Hassani et al. (2018) uses a simpler system with only 4 mooring lines, whilst the system in this thesis uses 8 mooring lines. In addition this thesis uses a passive observer, whilst Hassani et al. (2018) uses a Kalman filter.

Chapter 3

Theory

3.1 Probability

Probability of an intersection of two variables:

$$P(A \cap B) = P(A \mid B) \cdot P(B) \tag{3.1}$$

Bayes rule:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{3.2}$$

The Conditional Probability Theorem can be found using equation (3.1):

$$\begin{aligned} P(A \cap B \mid C) &= \frac{P(A \cap B \cap C)}{P(C)} \\ &= \frac{P(A \mid B \cap C) \cdot P(B \cap C)}{P(C)} \\ &= \frac{P(A \mid B \cap C) \cdot P(B \mid C) \cdot P(C)}{P(C)} \\ &= P(A \mid B \cap C) \cdot P(B \mid C) \end{aligned} \tag{3.3}$$

Total Probability Theorem:

$$P(A) = \int_B P(A \mid B) \cdot P(B)\, dB \tag{3.4a}$$

$$P(A \mid B) = \int_C P(A \mid B \cap C) \cdot P(C \mid B)\, dC \tag{3.4b}$$

This section is taken from Walpole et al. (2014).

3.2 Gaussian Probability Density Function

The probability density function for a Gaussian vector $\boldsymbol{x}$ is given by (Ash, 1996)

$$\frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k\left|\boldsymbol{\Sigma}\right|}} \tag{3.5}$$

Where $\mu = \mathrm{E}[\boldsymbol{x}]$ is the expected value of $\boldsymbol{x}$, $\boldsymbol{\Sigma} = \mathrm{E}[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T]$ is the covariance matrix of $\boldsymbol{x}$, and k is the length of $\boldsymbol{x}$.

3.3 System Model

Fossen (2011) proposes a simplified 3 degree of freedom (DOF) model for dynamic positioning (DP). The three degrees used are surge, sway, and yaw, as the other three DOFs can be neglected for this case. The model is given as

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A}_w\boldsymbol{\xi} + \boldsymbol{E}_w\boldsymbol{w}_1 \tag{3.6a}$$

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} \tag{3.6b}$$

$$\dot{\boldsymbol{b}} = -\boldsymbol{T}^{-1}\boldsymbol{b} + \boldsymbol{w}_2 \tag{3.6c}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} = -\boldsymbol{D}\boldsymbol{\nu} + \boldsymbol{R}^\top(\psi)\boldsymbol{b} + \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{w}_3 \tag{3.6d}$$

$$\boldsymbol{y} = \boldsymbol{\eta} + \boldsymbol{C}_w\boldsymbol{\xi} + \boldsymbol{v} \tag{3.6e}$$

Where $\boldsymbol{\eta} = [N, E, \psi]^\top$ is the position of the vessel in a 3 DOF North-East-Down (NED) frame, and $\boldsymbol{\nu} = [u, v, r]^\top$ is the velocity. The state vector $\boldsymbol{\xi} \in \mathbb{R}^6$ and the matrices $\boldsymbol{A}_w \in \mathbb{R}^{6\times6}$, $\boldsymbol{C}_w \in \mathbb{R}^{3\times6}$, and $\boldsymbol{E}_w \in \mathbb{R}^{3\times6}$ describe the wave model, such

that $\boldsymbol{\eta}_\omega = \boldsymbol{C}_w\boldsymbol{\xi}$ is the wave response model of the vessel in surge, sway, and yaw. $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{D}$ is the damping matrix, $\boldsymbol{R}(\psi)$ is the rotation matrix in yaw, and $\boldsymbol{\tau}$ and $\boldsymbol{\tau}_{\text{wind}}$ are the loads applied to the vessel due to the thrusters and wind, respectively. The bias term $\boldsymbol{b}$ contains the effect of ocean currents, as well as any unmodeled nonlinear dynamics, and $\boldsymbol{T}$ is a diagonal matrix of positive time biases. The vector $\boldsymbol{y}$ are the measured states, while $\boldsymbol{w_i}(i = 1, 2, 3)$ ans $\boldsymbol{v}$ are zero-mean Gaussian noise vectors representing model uncertainty and measurement noise respectively.

The wave model parameters $\boldsymbol{A}_w$, $\boldsymbol{C}_w$, and $\boldsymbol{E}_w$ are given by Fossen and Strand (1999) as

$$\boldsymbol{A}_w = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{\Omega}_{21} & \boldsymbol{\Omega}_{22} \end{bmatrix} \tag{3.7}$$

$$\boldsymbol{C}_w = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{3.8}$$

$$\boldsymbol{E}_w = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\Sigma}_2 \end{bmatrix} \tag{3.9}$$

Where $\boldsymbol{\Omega}_{21}$, $\boldsymbol{\Omega}_{22}$, and $\boldsymbol{\Sigma}_2$ are given by

$$\boldsymbol{\Omega}_{21} = \text{diag}\left(\omega_{01}^2, \omega_{02}^2, \omega_{03}^2\right) \tag{3.10}$$

$$\boldsymbol{\Omega}_{22} = \text{diag}\left(2\zeta_1\omega_{01}, 2\zeta_2\omega_{02}, 2\zeta_3\omega_{03}\right) \tag{3.11}$$

$$\boldsymbol{\Sigma}_2 = \text{diag}\left(\sigma_1, \sigma_2, \sigma_3\right) \tag{3.12}$$

Where ω_{0i} is the dominating wave frequency, ζ_i is the relative damping ratio, and σ_i is a parameter related to the wave intensity, all defined for $i = (1, 2, 3)$.

As this model is applied to a TAPM problem and not a DP problem, it needs to be modified slightly. As the wind force is not a measurable parameter in the simulation software used, $\boldsymbol{\tau}_{\text{wind}}$ disappears. The effect of wind forces will instead be lumped into the bias term, along with the other unmodeled dynamics. In place of the wind forces, the load from the mooring lines $\boldsymbol{\tau}_m$ is added. So equation (3.6d) now becomes

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} = -\boldsymbol{D}\boldsymbol{\nu} + \boldsymbol{R}^\top(\psi)\boldsymbol{b} + \boldsymbol{\tau} + \boldsymbol{\tau}_m + \boldsymbol{w}_3 \tag{3.13}$$

3.3.1 Estimating Mooring Forces

A horizontal-plane spread mooring model can be formulated as (Ren et al., 2015)

$$\boldsymbol{\tau}_m = -\boldsymbol{R}^\top(\psi)\boldsymbol{g}_{mo}(\boldsymbol{\eta}) - \boldsymbol{d}_{mo}(\boldsymbol{\nu}) \tag{3.14}$$

Where $\boldsymbol{d}_{mo}(\boldsymbol{\nu}) \in \mathbb{R}^3$ is the damping effect of the mooring lines, and $\boldsymbol{g}_{mo}(\boldsymbol{\eta}) \in \mathbb{R}^3$ is the restoring force component. Assuming fixed mooring line length, the damping effects can be linearised as $\boldsymbol{D}_{mo}\boldsymbol{\nu}$ (Ren et al., 2015). The linearised mooring damping matrix $\boldsymbol{D}_{mo}$ is estimated to be in the order of magnitude of 10-20% of critical damping of the system (DNV, 2010).

The restoring force component is given by Ren et al. (2015)

$$\boldsymbol{g}_{mo}(\boldsymbol{\eta}) = \boldsymbol{T}(\boldsymbol{\beta})\boldsymbol{L}_p\boldsymbol{\tau}_H \tag{3.15}$$

Where $\boldsymbol{\tau}_H \in \mathbb{R}^M$ is a vector of the horizontal of component of the tension for each mooring line, and $\boldsymbol{L}_p \in \mathbb{R}^{M\times M}$ is the line breakage matrix. Here M is the number of mooring lines. When all lines are intact this becomes $\boldsymbol{L}_p = \boldsymbol{I}$. For a failure scenario where line number p is broken, the p^{th} element of the diagonal of $\boldsymbol{L}_p$ is zero. $\boldsymbol{T}(\boldsymbol{\beta}) \in \mathbb{R}^{3\times M}$ is the mooring line configuration matrix, which maps the forces from each line to the body-fixed frame. It is given by

$$\boldsymbol{T}(\boldsymbol{\beta}) = \begin{bmatrix} \cos(\beta_1) & \dots & \cos(\beta_M) \\ \sin(\beta_1) & \dots & \sin(\beta_M) \\ \bar{x}_1\sin(\beta_1) - \bar{y}_1\cos(\beta_1) & \dots & \bar{x}_M\sin(\beta_M) - \bar{y}_M\cos(\beta_M) \end{bmatrix} \tag{3.16}$$

Where $\boldsymbol{\beta} \in \mathbb{R}^M$ are the angles between the x-axis (see figure 3.1) and mooring line number i, and $\bar{x}_i$ and $\bar{y}_i$ are the distances from the turret connection point to the anchors of each line, in x and y direction respectively.

The horizontal tension $\boldsymbol{\tau}_H$ is calculated by using a lookup table, based on the distance from the connection point to the anchor. This lookup table was created as a part of the TAPM simulator used for testing. See Ren (2015).

The final expression for the mooring line forces then becomes

$$\boldsymbol{\tau_m} = -\boldsymbol{R}(\psi)^\top\boldsymbol{T}(\boldsymbol{\beta})\boldsymbol{L}_p\boldsymbol{\tau}_H - \boldsymbol{D}_{mo}\boldsymbol{\nu} \tag{3.17}$$

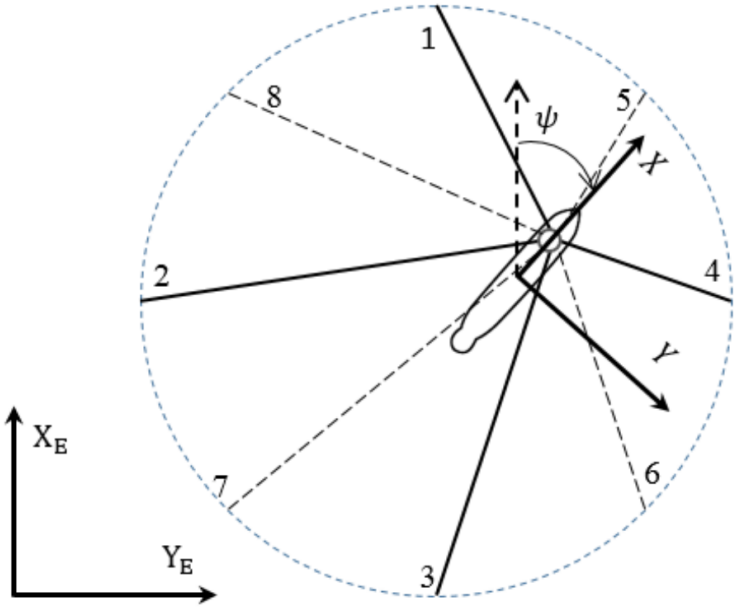


Figure 3.1: Reference frames

3.4 Passive Observer

The system model for the nonlinear passive observer is found by applying two assumptions to the DP model in equation (3.6):

- $\boldsymbol{w} = \boldsymbol{0}$ and $\boldsymbol{v} = \boldsymbol{0}$. That is, the Gaussian white noise terms for the disturbances and measurement noise are omitted from the observer model.

- $\boldsymbol{R}(y_3) = \boldsymbol{R}(\psi)$. This implies that the yaw angle disturbance caused by waves is small, giving $y_3 = \psi + \psi_w \approx \psi$. This is a good assumption, as this angle will normally be less than 5 degrees even in extreme weather conditions (Fossen, 2011).

Thus the nonlinear passive observer model is given by Fossen (2011) as

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A}_w \boldsymbol{\xi} \tag{3.18a}$$
$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu} \tag{3.18b}$$
$$\dot{\boldsymbol{b}} = -\boldsymbol{T}^{-1}\boldsymbol{b} \tag{3.18c}$$
$$\boldsymbol{M}\dot{\boldsymbol{\nu}} = -\boldsymbol{D}\boldsymbol{\nu} + \boldsymbol{R}^\top(\psi)\boldsymbol{b} + \boldsymbol{\tau} \tag{3.18d}$$
$$\boldsymbol{y} = \boldsymbol{\eta} + \boldsymbol{C}_w \boldsymbol{\xi} \tag{3.18e}$$

On the basis of this model, the passive observer is described by Fossen (2011) as

$$\dot{\hat{\boldsymbol{\xi}}} = \boldsymbol{A}_w \hat{\boldsymbol{\xi}} + \boldsymbol{K}_1(\boldsymbol{\omega}_0)\tilde{\boldsymbol{y}} \tag{3.19a}$$
$$\dot{\hat{\boldsymbol{\eta}}} = \boldsymbol{R}(\psi)\hat{\boldsymbol{\nu}} + \boldsymbol{K}_2\tilde{\boldsymbol{y}} \tag{3.19b}$$
$$\dot{\hat{\boldsymbol{b}}} = -\boldsymbol{T}^{-1}\hat{\boldsymbol{b}} + \boldsymbol{K}_3\tilde{\boldsymbol{y}} \tag{3.19c}$$
$$\boldsymbol{M}\dot{\hat{\boldsymbol{\nu}}} = -\boldsymbol{D}\hat{\boldsymbol{\nu}} + \boldsymbol{R}^\top(\psi)\hat{\boldsymbol{b}} + \boldsymbol{\tau} + \boldsymbol{R}^\top(\psi)\boldsymbol{K}_4\tilde{\boldsymbol{y}} \tag{3.19d}$$
$$\hat{\boldsymbol{y}} = \hat{\boldsymbol{\eta}} + \boldsymbol{C}_w\hat{\boldsymbol{\xi}} \tag{3.19e}$$

Where $\boldsymbol{K}_1(\boldsymbol{\omega}_0) \in \mathbb{R}^{6\times 3}$ and $\boldsymbol{K}_{2,3,4} \in \mathbb{R}^{3\times 3}$ are gain matrices. Here $\boldsymbol{\omega}_0 = [\omega_{01}, \omega_{02}, \omega_{03}]^\top$ is the vector of the wave spectra peak frequencies, with ω_{0i} being the frequency for surge, sway, and yaw for $i = (1, 2, 3)$ respectively.

The passive observer used is implemented in the Marine Systems Simulator (MSS), see Fossen and Perez (2004).

3.5 Dynamic Hypothesis Testing

Dynamic hypothesis testing calculates the probability that a given hypothesis $\mathcal{H} = \mathcal{H}_i$ is true, given the measurements up to and including that point in time $Y(t+1)$. This can be expressed as

$$\Pr\{\mathcal{H} = \mathcal{H}_i \mid Y(t+1)\} \tag{3.20}$$

Using equation (3.1) and (3.2), and writing $Y(t+1) = Y(t) + y(t+1)$ where $y(t+1)$ is the most recent measurement, this can be rewritten as:

$$\Pr\{\mathcal{H} = \mathcal{H}_i \mid Y(t+1)\} = \frac{\Pr\{\mathcal{H} = \mathcal{H}_i \cap y(t+1) \cap Y(t)\}}{\Pr\{y(t+1) \cap Y(t)\}} \tag{3.21}$$

Using equation (3.1) again now gives

$$\Pr\{\mathcal{H} = \mathcal{H}_i \mid Y(t+1)\} = \frac{\Pr\{\mathcal{H} = \mathcal{H}_i \cap y(t+1) \mid Y(t)\} \cdot \Pr\{Y(t)\}}{\Pr\{y(t+1) \mid Y(t)\} \cdot \Pr\{Y(t)\}} \tag{3.22}$$

$$= \frac{\Pr\{\mathcal{H} = \mathcal{H}_i \cap y(t+1) \mid Y(t)\}}{\Pr\{y(t+1) \mid Y(t)\}} \tag{3.23}$$

Using equation (3.3) to rewrite the numerator, and equation (3.4b) to rewrite the denominator gives

$$\Pr\{\mathcal{H} = \mathcal{H}_i \mid Y(t+1)\} = \frac{\Pr\{y(t+1) \mid \mathcal{H} = \mathcal{H}_i \cap Y(t)\} \cdot \Pr\{\mathcal{H} = \mathcal{H}_i \mid Y(t)\}}{\int_H \Pr\{y(t+1) \mid Y(t) \cap \mathcal{H}\} \cdot \Pr\{\mathcal{H} \mid Y(t)\}\, dt} \tag{3.24}$$

Because there is a finite number N of discrete hypotheses, the denominator can be simplified to

$$\Pr\{\mathcal{H} = \mathcal{H}_i \mid Y(t+1)\} = \frac{\Pr\{y(t+1) \mid \mathcal{H} = \mathcal{H}_i \cap Y(t)\} \cdot \Pr\{\mathcal{H} = \mathcal{H}_i \mid Y(t)\}}{\sum_{k=1}^{N} \Pr\{y(t+1) \mid Y(t) \cap \mathcal{H} = \mathcal{H}_k\} \cdot \Pr\{\mathcal{H} = \mathcal{H}_k \mid Y(t)\}} \tag{3.25}$$

$Z(t) = \{Y(t), U(t)\}$ is introduced as the time history of both the measurements $Y(t)$ and the input signal $U(t)$. Using the complete history $Z(t)$ instead of just the measurement $Y(t)$ now gives

$$\Pr\{\mathcal{H} = \mathcal{H}_i \mid Z(t+1)\} = \frac{\Pr\{z(t+1) \mid \mathcal{H} = \mathcal{H}_i \cap Z(t)\} \cdot \Pr\{\mathcal{H} = \mathcal{H}_i \mid Z(t)\}}{\sum_{k=1}^{N} \Pr\{z(t+1) \mid Z(t) \cap \mathcal{H} = \mathcal{H}_k\} \cdot \Pr\{\mathcal{H} = \mathcal{H}_k \mid Z(t)\}} \tag{3.26}$$

This can be further simplified by using the fact that the control signal $u(t)$ is deterministic, giving

$$\Pr\{\mathcal{H} = \mathcal{H}_i \mid Z(t+1)\} = \frac{\Pr\{y(t+1) \mid \mathcal{H} = \mathcal{H}_i \cap Z(t)\} \cdot \Pr\{\mathcal{H} = \mathcal{H}_i \mid Z(t)\}}{\sum_{k=1}^{N} \Pr\{y(t+1) \mid Z(t) \cap \mathcal{H} = \mathcal{H}_k\} \cdot \Pr\{\mathcal{H} = \mathcal{H}_k \mid Z(t)\}} \tag{3.27}$$

Looking at this expression, notice that the numerator and denominator are essentially the same, the only difference being that the numerator is for only the current hypotheses, while the denominator sums the expressions over all hypotheses. Also note that the expression $\Pr\{\mathcal{H} = \mathcal{H}_i \mid Z(t)\}$ is known, as that is the probability calculated in the previous time step. This means that for each hypothesis only $\Pr\{y(t+1) \mid \mathcal{H} = \mathcal{H}_i \cap Z(t)\}$ has to be calculated to calculate the next probability.

The dependence terms $\mathcal{H}$ and Z are deterministic, because they consist of a specific known hypothesis, the measurements (which are known), and the inputs (which are controlled and therefore also known). Because of the linearity of the model, and the Gaussian nature of the disturbance and noise, $\boldsymbol{y}$ is Gaussian as well. This means that $\Pr\{y(t+1) \mid \mathcal{H} = \mathcal{H}_i \cap Z(t)\}$ can be calculated using the Gaussian probability density function (see equation (3.5)).

For the measurement vector, the expected value is the same as the estimated value. That is $\boldsymbol{\mu} = \mathrm{E}[\boldsymbol{y}] = \hat{\boldsymbol{y}}_{\mathcal{H}_i}$, where $\hat{\boldsymbol{y}}_{\mathcal{H}_i} = \boldsymbol{C}\hat{\boldsymbol{x}}_{\mathcal{H}_i}$ can be found from the estimated states of the system. The notation $\tilde{\boldsymbol{y}}_{\mathcal{H}_i} = \boldsymbol{y} - \hat{\boldsymbol{y}}_{\mathcal{H}_i}$ is introduced in order to simplify equation (3.5). This gives

$$\Pr\{y(t+1) \mid \mathcal{H} = \mathcal{H}_i \cap Z(t)\} = \frac{\exp\left(-\frac{1}{2}\tilde{\boldsymbol{y}}_{\mathcal{H}_i}^T \boldsymbol{\Sigma}_{\mathcal{H}_i}^{-1} \tilde{\boldsymbol{y}}_{\mathcal{H}_i}\right)}{\sqrt{(2\pi)^k \left|\boldsymbol{\Sigma}_{\mathcal{H}_i}\right|}} \tag{3.28}$$

Where $\boldsymbol{\Sigma}_{\mathcal{H}_i}$ is the error covariance matrix, given hypothesis $\mathcal{H}_i$. Inserting this equation into equation (3.27) makes it possible to calculate the probabilities for each hypothesis.

This section is taken from Hassani et al. (2018).

3.6 Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) is a method for estimating the value of certain parameters in a model, given a series of observations. Such a parametric model can be described on state-space form as (Hassani et al., 2013)

$$\boldsymbol{x}(t+1) = \boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) + \boldsymbol{G}(\boldsymbol{\theta})\boldsymbol{w}(t) \tag{3.29a}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}(\boldsymbol{\theta})\boldsymbol{x}(t) + \boldsymbol{v}(t) \tag{3.29b}$$

Where $\boldsymbol{\theta}$ is the parameter (or parameters) that is to be estimated. The likelihood of $\boldsymbol{\theta}$ having a certain value is denoted as $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{Y}(t))$, where $\boldsymbol{Y}(t) \equiv \{\boldsymbol{y}(0), \boldsymbol{y}(1), \ldots, \boldsymbol{y}(t)\}$ is the time history of the measured output of the system.

The likelihood of the parameter given the observations is equal to the probability of getting those observations, given the parameter. This can be expressed as

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{Y}(t)) = p(\boldsymbol{Y}(t)|\boldsymbol{\theta}) \tag{3.30}$$

This means that for a given set of measurements $\boldsymbol{Y}(t)$, the maximum likelihood estimate for the parameter $\boldsymbol{\theta}$ is the value that maximizes the likelihood function $\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{Y}(t))$. As per equation (3.30), this is the same as finding the estimated parameter $\hat{\boldsymbol{\theta}}$ that gives the highest probability of getting the known measurements $\boldsymbol{Y}(t)$. The probability $p(\boldsymbol{Y}(t)|\boldsymbol{\theta})$ can be calculated as (Hassani et al., 2013)

$$p(\boldsymbol{Y}(t)|\theta) = \prod_{\tau=1}^{\tau=t} p(\boldsymbol{y}(\tau)|\boldsymbol{Y}(\tau-1), \boldsymbol{\theta}) \tag{3.31}$$

If the process is Gaussian, such that $\boldsymbol{Y}(t)$ is a Gaussian sequence, the probability of each measurement can be described using the Gaussian pdf (see equation (3.5)), giving

$$p(\boldsymbol{y}(\tau)|\boldsymbol{Y}(\tau-1), \boldsymbol{\theta}) = \frac{\exp\left(-\frac{1}{2}\tilde{\boldsymbol{y}}_\theta^\top(\tau)\Sigma_\theta^\top(\tau)\tilde{\boldsymbol{y}}_\theta(\tau)\right)}{(2\pi)^{\frac{k}{2}}\sqrt{\left|\Sigma_\theta(\tau)\right|}} \tag{3.32}$$

Where Σ_θ is the error covariance matrix given parameter $\boldsymbol{\theta}$, and $\tilde{\boldsymbol{y}}_\theta = \boldsymbol{y} - \hat{\boldsymbol{y}}_\theta$ is the estimation error. Combining this with equation (3.30) and (3.31) gives

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{Y}(t)) = \prod_{\tau=1}^{\tau=t} \frac{\exp\left(-\frac{1}{2}\tilde{\boldsymbol{y}}_\theta^\top(\tau)\Sigma_\theta^\top(\tau)\tilde{\boldsymbol{y}}_\theta(\tau)\right)}{(2\pi)^{\frac{k}{2}}\sqrt{\left|\Sigma_\theta(\tau)\right|}} \tag{3.33}$$

This equation can be used to calculate the likelihood for each value of $\boldsymbol{\theta}$ that is being evaluated, and whichever parameter gets the highest likelihood will be the best estimate. In practice it is easier to work with the log-likelihood function instead. Because the logarithm is a strictly increasing function, this does not change the maximum likelihood estimation. The log-likelihood function is given as

$$\ln\left(\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{Y}(t))\right) = \sum_{\tau=1}^{\tau=t} -\frac{k}{2}\ln(2\pi) - \frac{1}{2}\ln\left(\left|\Sigma_\theta(\tau)\right|\right) - \frac{1}{2}\tilde{\boldsymbol{y}}_\theta^\top(\tau)\Sigma_\theta^\top(\tau)\tilde{\boldsymbol{y}}_\theta(\tau) \quad (3.34)$$

The first part of this expression is constant, and as such can be neglected when maximizing the function as it will have no differentiating effect. As suggested by Hassani et al. (2013), it is now possible to simplify the expression by instead minimizing the negative log-likelihood function, denoted by $\mathcal{K}(\boldsymbol{\theta}|\boldsymbol{Y}(\boldsymbol{t}))$. This gives the final equation to be minimized as

$$\mathcal{K}(\boldsymbol{\theta}|\boldsymbol{Y}(\boldsymbol{t})) = \sum_{\tau=1}^{\tau=t} \frac{1}{2}\left(\ln\left(\left|\Sigma_\theta(\tau)\right|\right) + \tilde{\boldsymbol{y}}_\theta^\top(\tau)\Sigma_\theta^\top(\tau)\tilde{\boldsymbol{y}}_\theta(\tau)\right) \quad (3.35)$$

Chapter 4

Methods

In order to test the methods presented a TAPM vessel simulator implemented in Simulink was used. A simple PI controller was added to the simulator in order to keep the vessel roughly in position, and make it able to recover from disturbances. The position measurements from the simulator, along with the known thruster input given by the controller, were sent to a series of passive observers, one for each failure scenario being examined. These estimated the states, which were passed to the statistical methods DHT and MLE along with an estimated error covariance. The DHT and MLE blocks would then analyse this data to determine which of the failure scenarios were correct. Figure 4.1 shows a flowchart of the process.

There were 9 scenarios used in all the simulations. Scenario 1 through 8 were the failure of mooring line 1 through 8 respectively. Scenario 9 was the no breakage situation, where all mooring lines were still intact. This means that only single line failures were looked at.

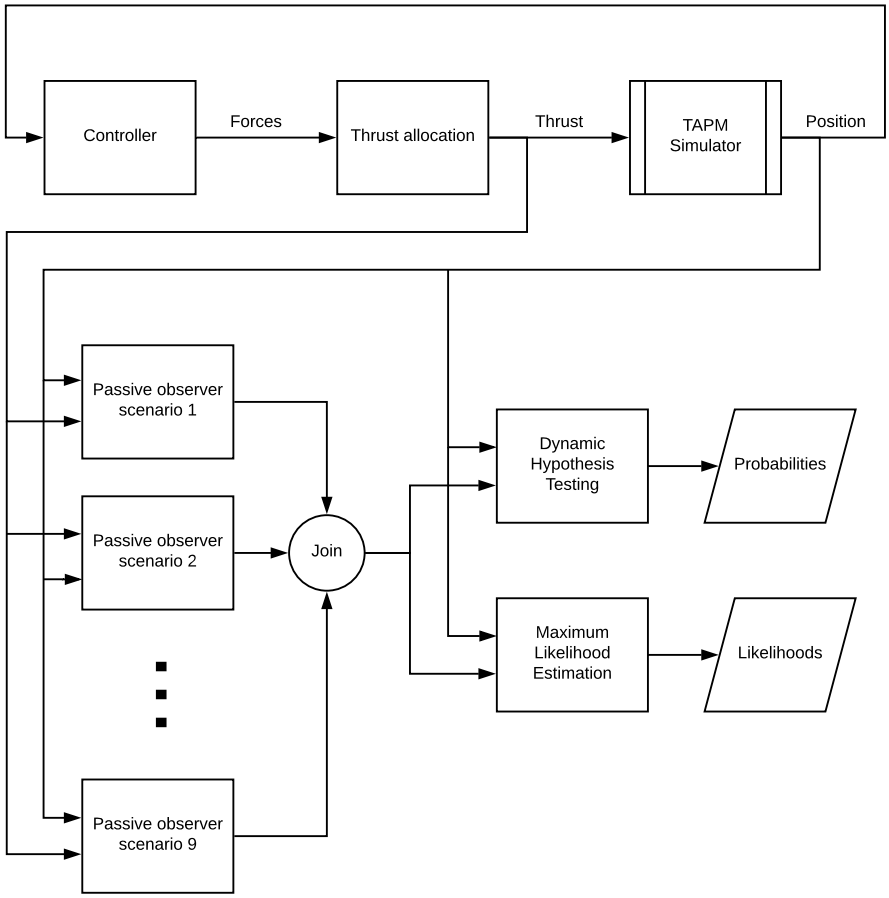


Figure 4.1: Flowchart MATLAB/Simulink implementation

4.1 System configuration

The simulated vessel is an FPSO with a length of 200 meters. The principle dimensions of the vessel are given in table 4.1.

Table 4.1: Principle dimensions of simulated FPSO

Parameter	Value
Length between perpendiculars	200 [m]
Breadth	44 [m]
Draught	12 [m]
Mass	$1.004 \cdot 10^8$ [kg]

The vessel is fitted with 8 mooring lines, evenly distributed around the mooring line turret of the vessel. Each line is a 2250 meter cable with a diameter of 0.08 meters. The position of the mooring line anchors are given in table 4.2.

Table 4.2: Position of mooring line anchors

Line	x	y	z
1	1950	0	-1000
2	0	1950	-1000
3	-1950	0	-1000
4	0	-1950	-1000
5	1378.9	1378.9	-1000
6	-1378.9	1378.9	-1000
7	-1378.9	-1378.9	-1000
8	1378.9	-1378.9	-1000

The vessel is fitted with three thrusters in a normal configuration: two thrusters at the aft of the vessel and a bow thruster. All three thrusters are fixed. The thrust force is controlled by the PI controller that was implemented in the simulation.

Figure 4.2 shows the position of the anchors, the direction of the waves and current, as well as the orientation of the vessel.

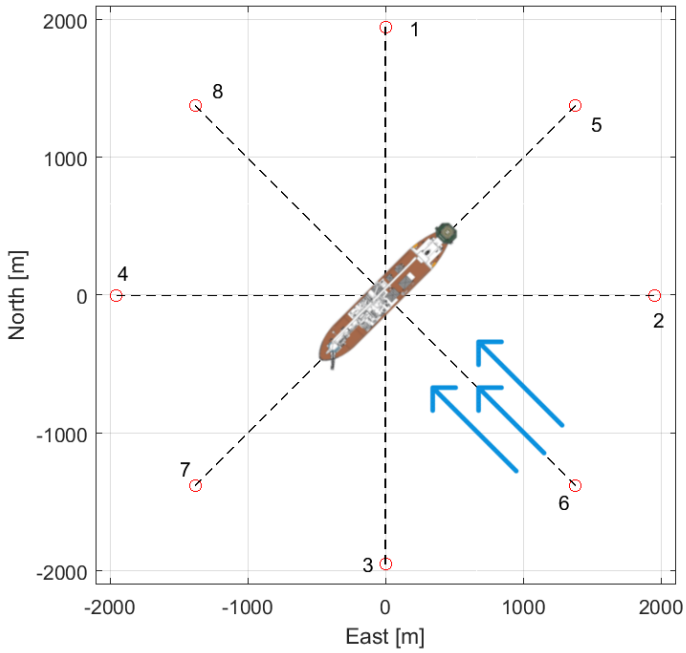


Figure 4.2: Overview of the system configuration, showing mooring lines and anchors. The direction of the wave and current disturbances are shown by the blue arrows. Vessel not to scale

4.2 Simulator

The TAPM vessel simulator used was provided by Zhengru Ren, and was developed for his master thesis (see Ren (2015)). The inner workings of the simulator are beyond the scope of this thesis. The simulator takes the desired thruster forces for each thruster as input, and outputs the position of the vessel and the mooring line tensions.

A number of parameters can be adjusted for each simulation. The initial position and velocity of the vessel can be set, and measurement noise for the output variables can be turned on or off. The position of the anchor lines can be set, as well as how many of them are intact at the start of the simulation. It is also possible to set which line or lines will break, and at what time. The breakage is treated as if the tension in the broken mooring line completely disappears, i.e. a failure at the connection to the vessel. This is the same way as it is being modelled in the observers, i.e. a diagonal element of $\boldsymbol{L}_p$ is set to zero in equation (3.15).

The simulator implements two different environmental disturbances: waves and current. The wave forces are both first and second order forces, where each one can be switched on or off independently. In addition the mean wave direction and significant wave height can be controlled. Similarly, the current force can be switched on or off, and the direction and current velocity can be controlled. The values of the disturbances can be seen in table 4.3.

Table 4.3: Disturbance parameters

Parameter	Symbol	Value
Significant wave height	H_s	2.5 [m]
Dominant wave frequency	$\boldsymbol{\omega}_0$	$\frac{2\pi}{7}$ [rad/s]
Current velocity	ν_c	0.1 [m/s]

In addition, the directions of both disturbances can be seen in figure 4.2. All the disturbance parameters were kept constant for all simulations.

4.3 PI controller

A simple PI controller was implemented using the PID controller block provided by the MSS software package (see Fossen and Perez (2004)). This was done in order to ensure the vessel stayed near the equilibrium point of the system, even in the case of large disturbances like current or line failures. It was also to test that the observers and statistical analyses were able to handle an input of the

thruster forces τ. This controller is in no way optimal, and uses a simple thrust configuration matrix for thrust allocation.

The tuning parameters for the controller were manually tuned to the following values:

$$\boldsymbol{K}_p = \text{diag}\,(100000, 1000000, 1000) \tag{4.1}$$
$$\boldsymbol{K}_i = \text{diag}\,(1000, 1000, 100) \tag{4.2}$$
$$\tag{4.3}$$

Figure 5.2 shows the response of the system given an initial offset of 2 meters from the equilibrium in both x and y direction. It shows that the vessel quickly moves to the equilibrium position, and while there is a slight overshoot the vessel quickly reaches a stable condition at the equilibrium. This performance is satisfactory for for the given purpose of the controller.

4.4 Estimation of error covariance

The statistical methods need the error covariance matrices for the estimated states in order to work. However, this is not calculated online by the passive observer, unlike for example a Kalman filter. In order fix this, an estimate of the covariances is needed. This essentially meant setting $\boldsymbol{\Sigma}_\theta(\tau) \approx \boldsymbol{\Sigma}_\theta$ in equation (3.35).

The estimation of $\boldsymbol{\Sigma}_\theta$ was achieved by running a 10000 second simulation for each failure scenario, with the controller activated. The resulting estimation errors were recorded, and the built in MATLAB function `cov()` was used to calculate the covariance matrix for each signal. These matrices were then stored, and fetched by Simulink to be passed to the statistical analysis block.

The code generating the error covariances can be seen in appendix A.5.

4.5 Tuning the passive observer

The observer gain matrices for the passive observer given by equation (3.19) are given by Fossen (2011) to be on the following diagonal form:

$$\boldsymbol{K}_1(\boldsymbol{\omega}_0) = \begin{bmatrix} \text{diag}\,(K_{11}(\omega_{01}), K_{12}(\omega_{02}), K_{13}(\omega_{03})) \\ \text{diag}\,(K_{14}(\omega_{01}), K_{15}(\omega_{02}), K_{16}(\omega_{03})) \end{bmatrix} \tag{4.4a}$$

$$\boldsymbol{K}_2 = \text{diag}\,(K_{21}, K_{22}, K_{23}) \tag{4.4b}$$

$$\boldsymbol{K}_3 = \text{diag}\,(K_{31}, K_{32}, K_{33}) \tag{4.4c}$$

$$\boldsymbol{K}_4 = \text{diag}\,(K_{41}, K_{42}, K_{43}) \tag{4.4d}$$

This shows that $\boldsymbol{K}_1$ is a function of the wave spectrum peak frequency $\boldsymbol{\omega}_0$. In fact, each K_{1i} can be calculated from the wave parameters. The constants are chosen in order for the passive observer to achieve the desired filtering effect, with a notch effect to filter out the wave disturbances. This gives the following expressions (Fossen, 2011)

$$K_{1i}(\omega_{0i}) = -2(\zeta_{ni} - \lambda_i)\frac{\omega_{ci}}{\omega_{0i}} \tag{4.5a}$$

$$K_{1(i+3)}(\omega_{0i}) = 2\omega_{0i}(\zeta_{ni} - \lambda_i) \tag{4.5b}$$

Where ζ_{ni} determines the notch effect, ω_{ci} is the filter cutoff frequency, and λ_i is the relative damping ratio of the wave spectrum. Typical values are $\zeta_{ni} = 1.0$ and $\lambda_i = 0.1$ (Fossen, 2011), and these are the values that were used in the simulations. For the filter cutoff frequency a value of $\omega_{ci} = 1.2255\omega_{0i}$ was chosen, as per examples in both Fossen (2011) and Fossen and Perez (2004).

In addition, $\boldsymbol{K}_2$ is given simply by the cutoff frequency (Fossen, 2011):

$$K_{2i} = \omega_{ci} \tag{4.6}$$

The values for $\boldsymbol{K}_3$ and $\boldsymbol{K}_4$ were taken from example 11.6 in Fossen (2011). They were given as

$$\boldsymbol{K}_3 = 0.1 \cdot \boldsymbol{K}_4 \tag{4.7}$$

$$\boldsymbol{K}_4 = \text{diag}\,(0.1, 0.1, 0.01) \tag{4.8}$$

Because the values suggested by Fossen (2011) worked well from the start, they were not tuned further.

4.6 Implementation in MATLAB

4.6.1 S-functions

It is possible to program custom Simulink blocks through a standard interface known as S-functions. Using S-functions has several advantages compared to using MATLAB code blocks or other solutions. The main reason is the possibility for the block to have internal states that are remembered between each simulation time step. It is also possible to easily control the dimensions of the input and output ports of the block. This makes it much easier to detect any mistakes, especially in combination with the fact that S-functions allow setting breakpoints in the code. This is not possible in MATLAB code blocks in Simulink, and having this possibility greatly simplifies debugging. In this project, level 2 S-functions have been used.

Each S-function consists of several functions, that are called by Simulink at different points in the simulation. Some of these callback functions are listed below:

- **`setup`:** Specifies the number of inputs, outputs, states, parameters, and other characteristics of the S-function. First function to be run.
- **`PostPropagationSetup`:** Specify the number and sizes of the internal state vectors, and create run-time parameters. Called only once, before simulation starts.
- **`Start`:** Initialize the internal state vectors. Run once before simulation start.
- **`CheckParameters`:** Checks the validity of the dialog box parameters given to the block. Run once before simulation start.
- **`Update`:** Update the internal state vectors. Is run each time step to calculate the new state.
- **`Outputs`:** Computes the signals that the block emits. Is run each time step.
- **`Terminate`:** Perform any actions required at termination of the simulation. Is only called at the end of the simulation.

There are many more callback functions that could be used for different types of blocks, but these are the main ones needed for this task. (Mathworks, 2017)

4.6.2 Dynamic hypothesis testing

The dynamic hypothesis testing block is implemented using a level 2 S-function. The block takes three inputs, the measurements $\mathbf{y}$, the estimated states $\hat{\mathbf{x}}$ from the observer, and the error covariance matrix Σ_θ. In addition it has dialogue box parameter, where a struct that describes the model used is provided. It has only one output and one internal state, both being a vector of the probabilities for each hypothesis being tested. An overview of the statistical analysis blocks can be seen in figure 4.3.

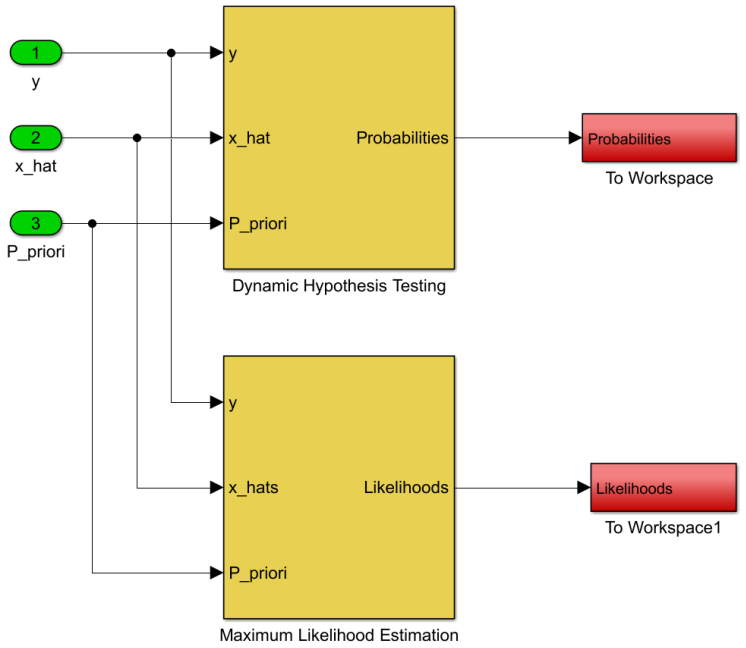


Figure 4.3: MLE and DHT blocks in Simulink

The probabilities are not known at the start of the simulation, so the internal state is initialized with an equal probability for each of the hypotheses. This is done in the `Start` function of the S-function.

To calculate the probabilities, equations (3.27) and (3.28) are used. From equation (3.27) it can be seen that the denominator is just the sum of all the numerators that need to be calculated. Therefore only the numerators are computed, and then summed at the end to get the denominator. Simple element-wise division then gives the probability vector for the hypotheses. This calculation is

done in the `Update` function, and the result is stored in the internal state. It is then provided to Simulink upon request using the `Outputs` function.

When the same hypothesis is true for a long duration of time, the probability of the other hypotheses will go to zero as the probability of the true hypothesis goes to one. Equation (3.27) shows that, for each time step, the probability for the newest measurement is multiplied with the existing probability from the previous timestep. This means that even if the newest measurement gives a high probability of a given hypothesis being true, if the previously known probability was infinitesimally small the new probability will be close to zero as well. This means that after a long time it will be almost impossible for the most probable hypothesis to change, because all other hypotheses are infinitesimally small. To counter this, a lower limit of 10^{-9} was implemented for the probabilities to stop them from becoming too small. This value was found to be a good compromise between switching hypothesis too quickly, causing oscillations between scenarios, and too slowly. The probabilities were then re-normalised so that the total probability remained 1.

The block is designed to be modular, and will calculate many of its parameters from the models that are provided to it. Unfortunately, this leads to some redundancy, as some parameters that are the same for all the models (like $\hat{\mathbf{y}}_{\mathcal{H}_i} = \mathbf{C}\hat{\mathbf{x}}_{\mathcal{H}_i}$, since $\mathbf{C}$ is the same for all hypotheses) are calculated multiple times. However, this trade off is worth it, as it makes the block completely independent of the use case.

The code implementing DHT can be seen in appendix A.1.

4.6.3 Maximum likelihood estimation

The maximum likelihood estimation algorithm is implemented in a level 2 S-function, in a very similar way to the DHT algorithm. The block takes the same three inputs ($\mathbf{y}$, $\hat{\mathbf{x}}$, and Σ_θ), and has one internal state and one output, both being the negative log-likelihood for each of the different failure scenarios.

The likelihoods are initialised to zero at the start of the simulation. They are then calculated using equation (3.35) for each timestep. By using the negative log-likelihood stored from the previous timestep, the equation only needs to be calculated for $\tau = t$. This represents the only new information, and can then be added to the stored value to get the updated $\mathcal{K}(\boldsymbol{\theta}|\boldsymbol{Y(t)})$. This essentially means modifying equation (3.35) to be as follows:

$$\mathcal{K}(\boldsymbol{\theta}|\boldsymbol{Y}(\boldsymbol{t})) = \overbrace{\sum_{\tau=1}^{\tau=t-1}\left(\frac{1}{2}\left(\ln\left(\left|\Sigma_\theta(\tau)\right|\right) + \tilde{\boldsymbol{y}}_\theta^\top(\tau)\Sigma_\theta^\top(\tau)\tilde{\boldsymbol{y}}_\theta(\tau)\right)\right)}^{\text{stored value}} + \frac{1}{2}\left(\ln\left(\left|\Sigma_\theta(t)\right|\right) + \tilde{\boldsymbol{y}}_\theta^\top(t)\Sigma_\theta^\top(t)\tilde{\boldsymbol{y}}_\theta(t)\right) \tag{4.9}$$

As with the DHT algorithm, the estimate of most likely failure scenario will become harder to change if one scenario stays true for a long period of time. This is because the likelihood for each scenario can grow unbounded, meaning likely scenarios will grow in a negative direction and unlikely scenarios will grow in a positive direction. This can be seen in figure 4.4. In order to limit this unbounded growth a forgetting factor C_{forget} was introduced:

$$\mathcal{K}(\boldsymbol{\theta}|\boldsymbol{Y}(\boldsymbol{t})) = \sum_{\tau=1}^{\tau=t-1}\left(\frac{1}{2}\left(\ln\left(\left|\Sigma_\theta(\tau)\right|\right) + \tilde{\boldsymbol{y}}_\theta^\top(\tau)\Sigma_\theta^\top(\tau)\tilde{\boldsymbol{y}}_\theta(\tau)\right)\right)\cdot C_{forget} + \frac{1}{2}\left(\ln\left(\left|\Sigma_\theta(t)\right|\right) + \tilde{\boldsymbol{y}}_\theta^\top(t)\Sigma_\theta^\top(t)\tilde{\boldsymbol{y}}_\theta(t)\right) \tag{4.10}$$

The forgetting factor was set as $C_{forget} = 0.999$, meaning the stored value would be weighted slightly less. However, because this forgetting factor is applied at every time step, the contribution from time step n will be weighted $(C_{forget})^{t-n}$ at time step t. Because $0 < C_{forget} < 1$, the contributions from older time steps will obviously have close to no effect on the total likelihood, and only the more recent contributions will be relevant. This effectively negates the unbounded growth, as older contributions are forgotten as new ones are added. As with the DHT minimum probability, the value of the forgetting factor was chosen to balance the time it took to switch estimated scenario upon a change, and oscillations between scenarios when there was no change.

The likelihoods for the same scenario with and without forgetting factor can be seen in figure 5.6 and 4.4 respectively. Comparing the two shows that the likelihoods grow linearly when there is no forgetting factor. When it is applied however the growth flattens out, and the likelihoods reach reasonably steady values.

The code implementing MLE can be seen in appendix A.2.

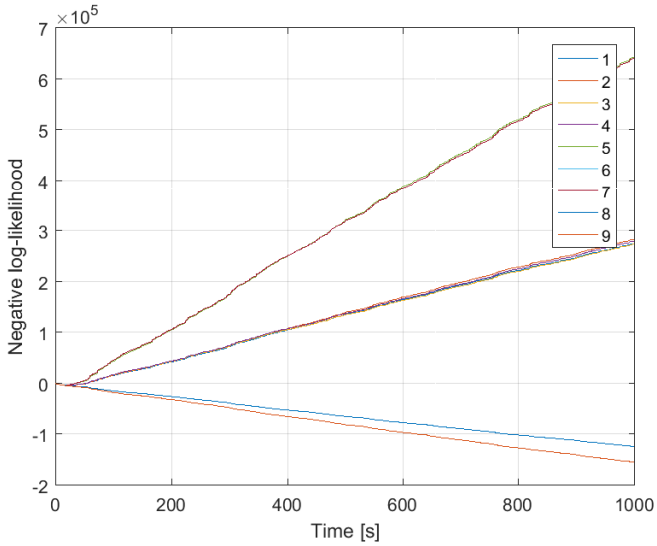


Figure 4.4: Unbounded negative log-likelihoods when there are no line breakages

4.6.4 Passive Observer

The passive observer is implemented in the Marine Systems Simulator (Fossen and Perez, 2004), and a block from that library is used. In addition, equation (3.17) describing the mooring line forces is implemented using a block diagram approach in Simulink (see figure 4.5). The mooring forces are added to the thruster forces before being passed to the passive observer block. The exact methodology is described in section 3.3.1.

4.6.5 Initialisation

The MATLAB function `initSim` initialises all the variables needed by the Simulink simulation. This includes loading the vessel and mooring line parameters, as well as initial conditions and anchor positions. In order to interface with the Simulink model the variables are moved to the base workspace using the function `putvar` (D'Errico, 2012). There they can be accessed by Simulink. Doing it this way ensures only the necessary parameters exist in the workspace, reducing the clutter.

A helper function `initPassiveObserver` is used to initialise the passive observer models. It outputs a cell array of nine structs, where each struct contains all the

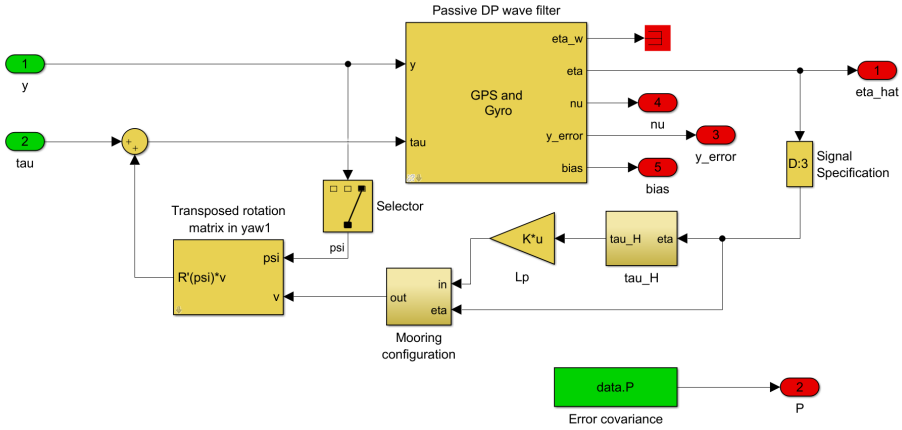


Figure 4.5: Implementation of passive observer in Simulink

model parameters needed for both the passive observer and statistical methods.

The code for `initSim` can be seen in appendix A.3, and `initPassiveObserver` can be seen in appendix A.4.

Chapter 5

Results

Simulations were run for four different failure scenarios:

- No line breakages
- Failure of line 6
- Failure of line 7
- Failure of line 8

All of these scenarios were simulated both with and without the current disturbance, while all were done with wave disturbance and measurement noise. Note that of the failures, line 6 and 8 are in line with the current disturbance. The current pushes the vessel towards line 8 and away from line 6. Line 7 is orthogonal to the direction of the current. This can be seen in figure 4.2. These particular failure scenarios were chosen to see how the system handles the effect of the current.

For all the simulations the initial conditions of the vessel are an initial position of $[N, E, \psi] = [2, 2, -2.36]$ and an initial velocity of $[u, v, r] = [0, 0, 0]$.

5.1 No breakage

Figure 5.1 shows the measured and estimated position of the vessel in a North-East coordinate system. The vessel can be seen to move from its starting position

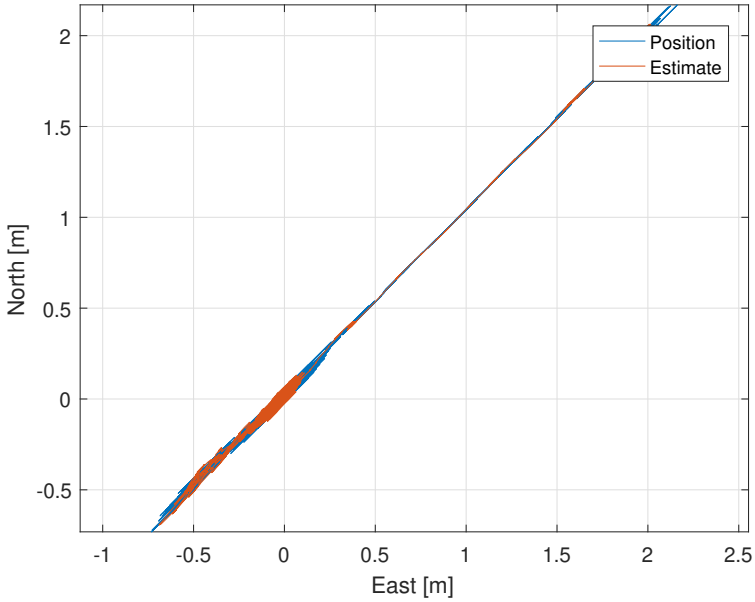


Figure 5.1: Position of the vessel when there are no line breakages

and almost straight towards the equilibrium point, with a little overshoot. Notice that the estimated position follows the measured position closely.

Figure 5.2 shows a time plot of each of the position coordinates. From this plot format it becomes evident that the vessel uses about 150 seconds so settle at the equilibrium coordinates. It is also noteworthy that the yaw angle hardly changes at all.

Figure 5.3 shows the residuals of the position estimate. That is, the difference between the measured position signal and the estimated position. It can be seen to be close to zero for all three degrees of freedom, with some noise in the order of magnitude of 0.1 meters and 0.01 radians.

Figure 5.4 shows which failure scenario the DHT and MLE methods estimate to be the most correct. It shows that both algorithms quickly converge to the correct scenario, and stay there for the duration of the simulation.

Figure 5.5 shows the probabilities of each of the nine hypotheses. It can be seen that for the first few seconds more than one of the hypotheses have spike, notably hypothesis 7. However after a few seconds the probability of hypothesis 9 goes to 1, and all others go to 0.

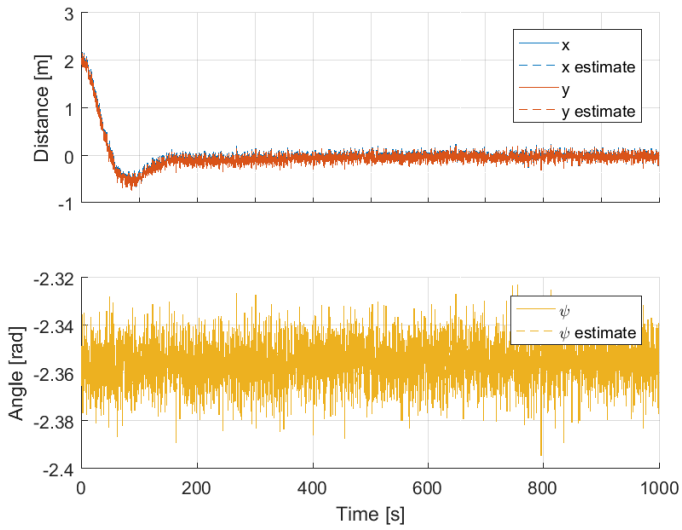


Figure 5.2: Time plot of the vessel position when there are no line breakages

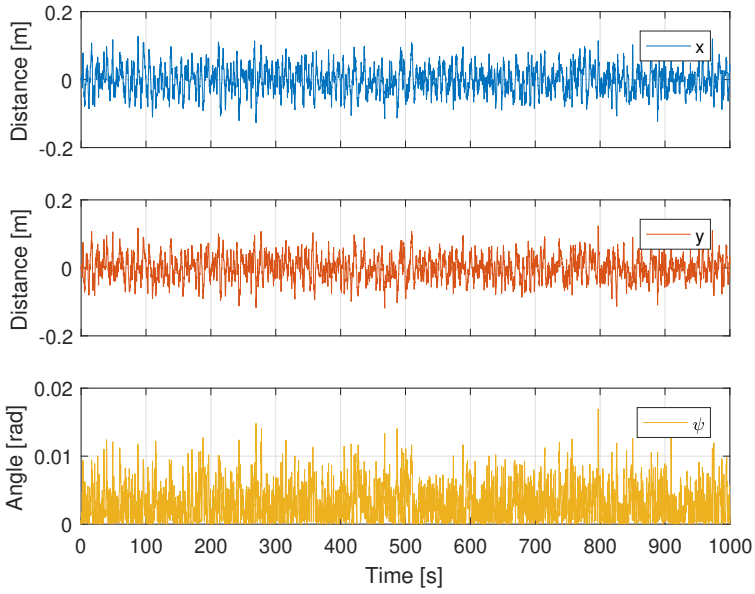


Figure 5.3: Position residuals when there are no line breakages

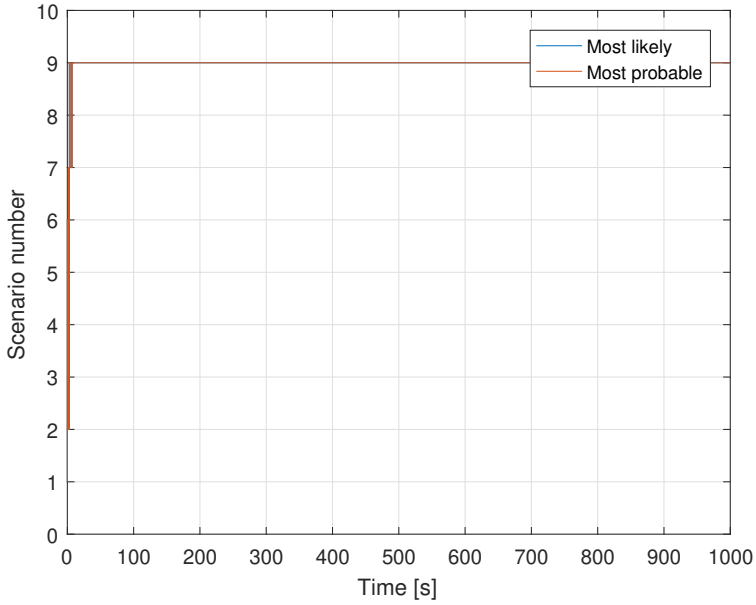


Figure 5.4: DHT and MLE estimates of correct scenario with no line breakages

Figure 5.6 shows the negative log-likelihoods for each failure scenario, and how they develop over time. It can be seen how the correct scenario, number 9, quickly grows to be the lowest among them. Note how the growth of all the likelihoods flatten out over time. This behaviour is caused by the forgetting factor (see section 4.6.3).

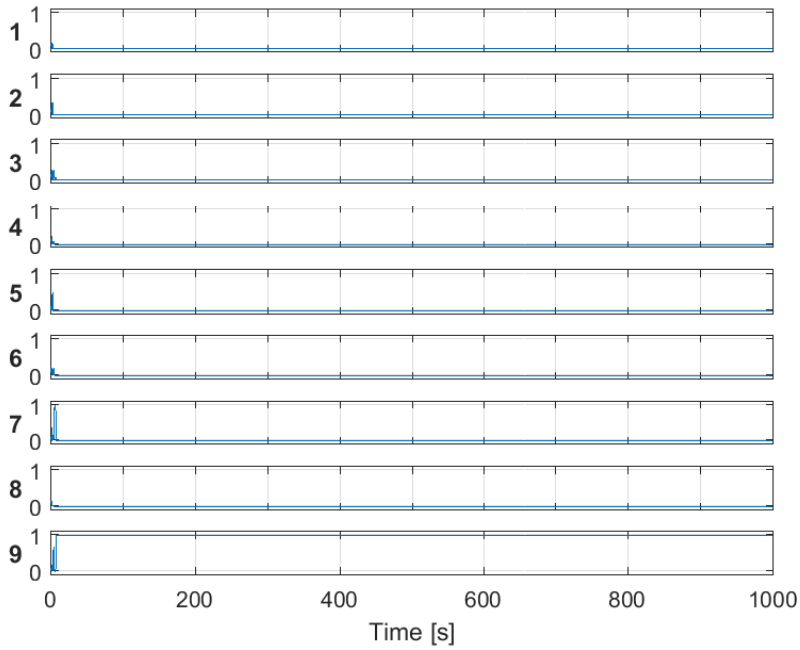


Figure 5.5: The probability for each of the different hypotheses when there are no line breakages

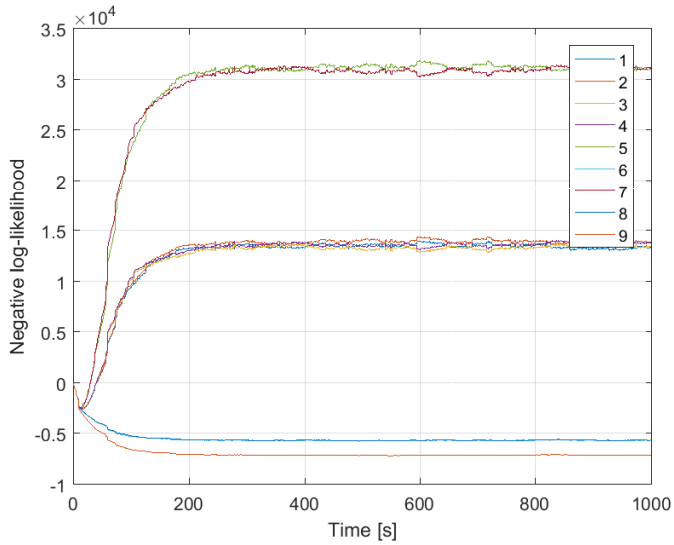


Figure 5.6: Negative log-likelihoods with no line breakages

5.2 No breakage with current

Figure 5.7 shows the position of the vessel in a North-East coordinate system with current load applied. Comparing to figure 5.1 we see that that the vessel no longer travels in a straight line towards the equilibrium point, as it is affected by the current pushing it to the north-west. Figure 5.7 also shows more clearly that the observer is in fact reducing the amount of noise in the position signal.

Figure 5.8 shows the DHT and MLE estimates of correct scenario, when there are no breakages and the current load is applied. Comparing to figure 5.4 where there is no current, we immediately see that the performance is not as good in this case. Both algorithms take longer to find the correct scenario, taking almost 100 seconds to stabilize. The MLE algorithm then gives a stable estimate for most of the simulation, while the DHT algorithm intermittently switches to the wrong hypothesis.

Figure 5.9 shows the probabilities for each of the hypotheses. It shows the same as figure 5.8, that most of the time hypothesis 9 is considered as the most probable, but intermittently it switches to hypothesis 6. This is especially apparent during the last section of the simulation.

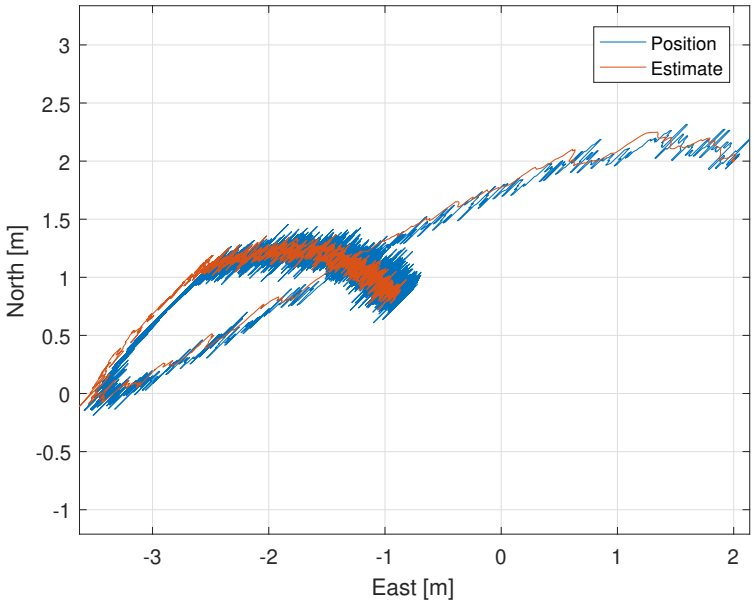


Figure 5.7: Position of the vessel position when there are no line breakages, with current load applied

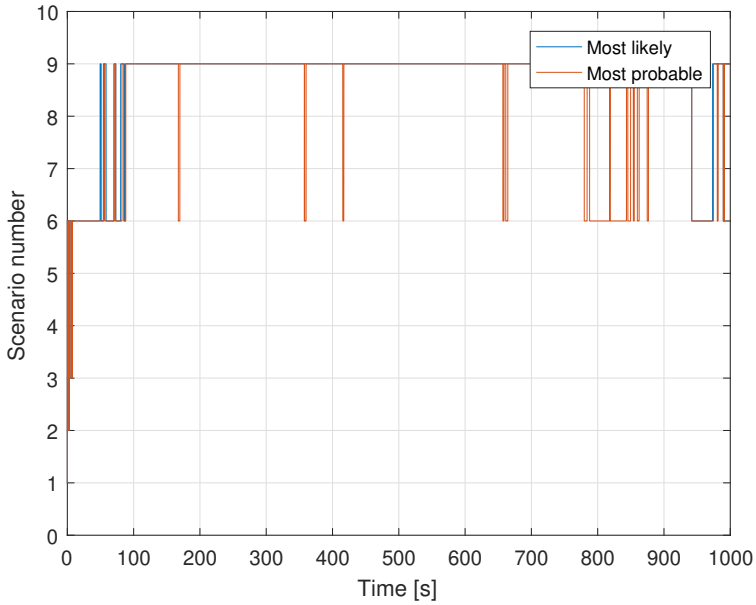


Figure 5.8: DHT and MLE estimates of correct scenario with no line breakages and current load applied

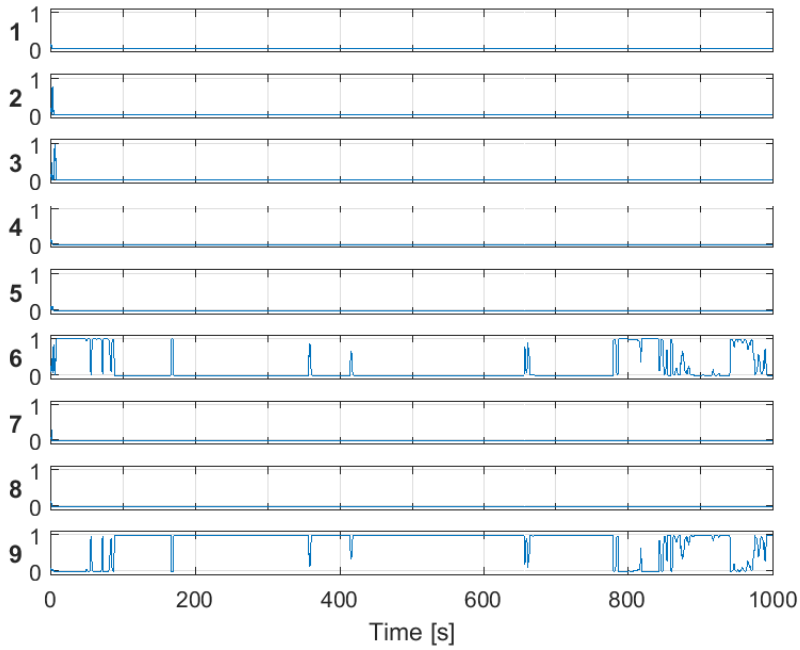


Figure 5.9: The probability for each of the different hypotheses when there are no line breakages and the current load is applied

5.3 Line 6 breaking

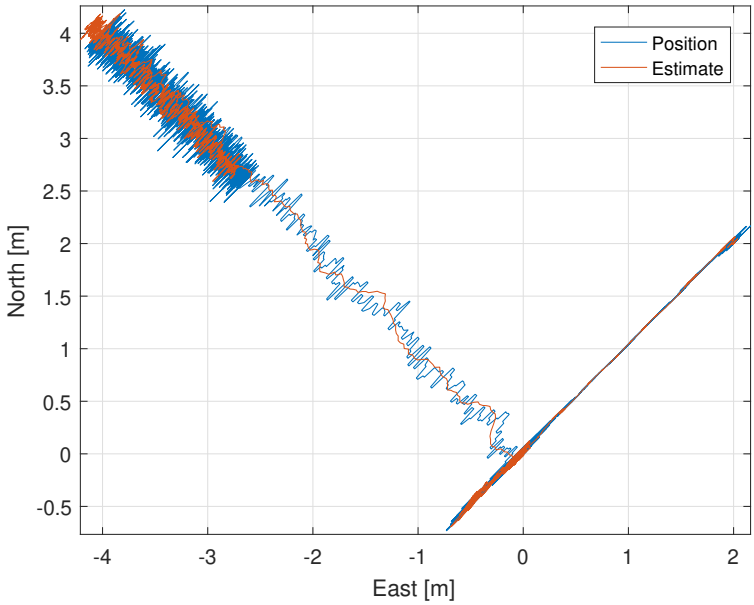
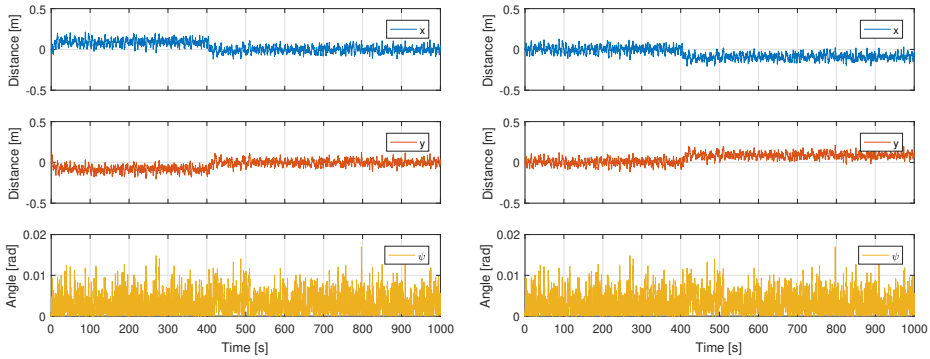


Figure 5.10: Position of vessel when line 6 breaks at $t = 400$ seconds

Figure 5.10 shows the position of the vessel when mooring line 6 breaks at $t = 400$ seconds. It can be clearly seen that up until that point the motion is very similar to figure 5.1. After the break the vessel moves in the opposite direction of mooring line 6. Note that the estimated position in this figure is from the observer that assumes scenario 9 to be correct. This can be seen at the upper left of the figure, where the estimate no longer aligns with the measured position because of this wrong assumption.

Figure 5.11 shows the difference in observer performance due to the assumed failure scenario. Figure 5.11a assumes that line 6 is broken, while figure 5.11b assumes all lines are intact. It can be seen in both plots that the residuals are close to zero when the observer is based on the correct scenario, and has a slight offset when it is wrong. At time $t = 400$ seconds both plots see a shift, as the correct scenario changes when line 6 breaks.

Figure 5.12 shows the DHT and MLE estimates of correct scenario when line 6 breaks at $t = 400$ seconds. It shows that the DHT estimate very quickly switches to the correct hypothesis, whilst the MLE algorithm takes a bit longer at around



(a) Observer assuming line 6 breaking being correct

(b) Observer assuming no breakages being correct

Figure 5.11: Residuals when line 6 breaks at $t = 400$ seconds

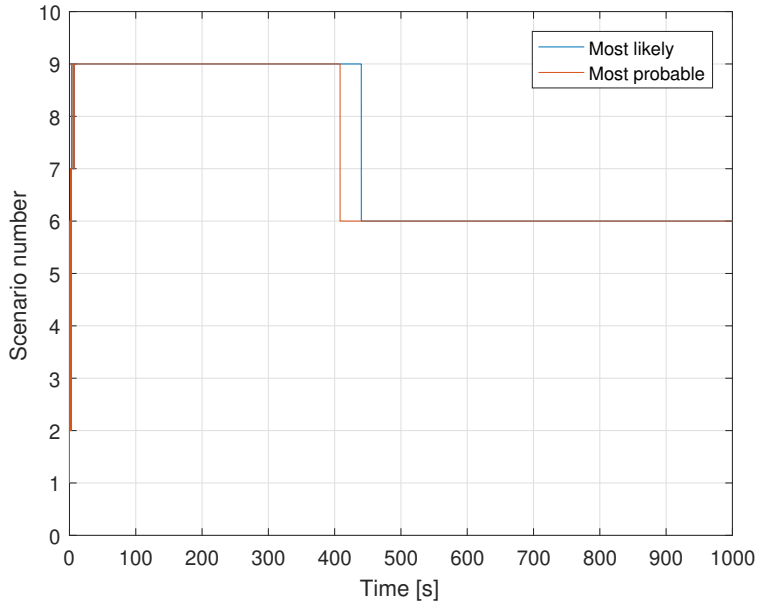


Figure 5.12: DHT and MLE estimates of correct scenario when line 6 breaks at $t = 400$ seconds

40-50 seconds.

5.4 Line 6 breaking with current

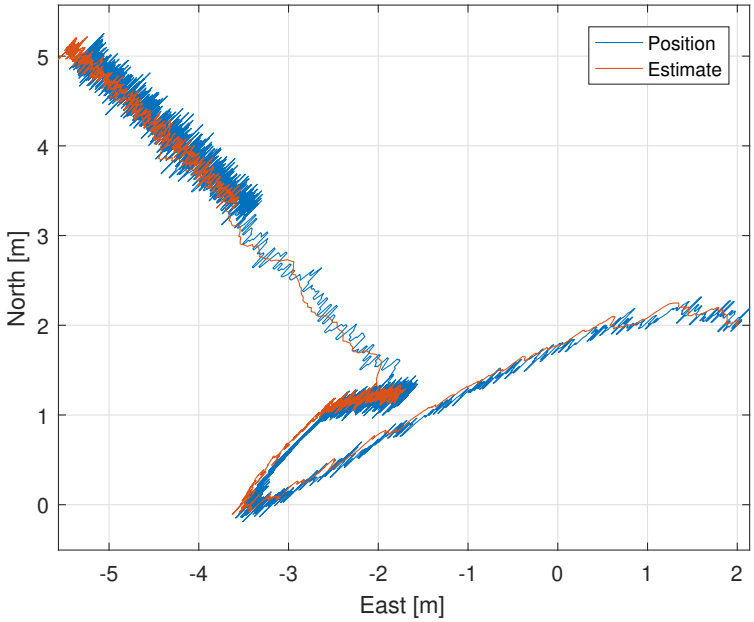


Figure 5.13: Position of vessel when line 6 breaks at $t = 400$ seconds, with current load applied

Figure 5.13 shows the position of the vessel when line 6 breaks at $t = 400$ seconds, and it is affected by current. Comparing with figure 5.10 it clearly shows the vessel being pushed further to the north-west by the current. Note that the estimated position is by an observer assuming no breakages.

Figure 5.14 shows the estimates for correct scenario using the DHT and MLE algorithms. It shows that with the current applied, it takes some time for the algorithms to determine the correct scenario. Both algorithms use almost 100 seconds. After that the estimate is stable, with the exception of a few spikes for the DHT estimate. When the line breaks a $t = 400$ seconds both methods almost immediately switch to the correct scenario and stay there.

Figure 5.15 shows the negative log-likelihoods of the different scenarios when line 6 breaks at $t = 400$ seconds, and the current load is applied. Note that the likelihoods of scenario 6 and 9 are almost intertwined until the break occurs, after which scenario 6 becomes the likeliest.

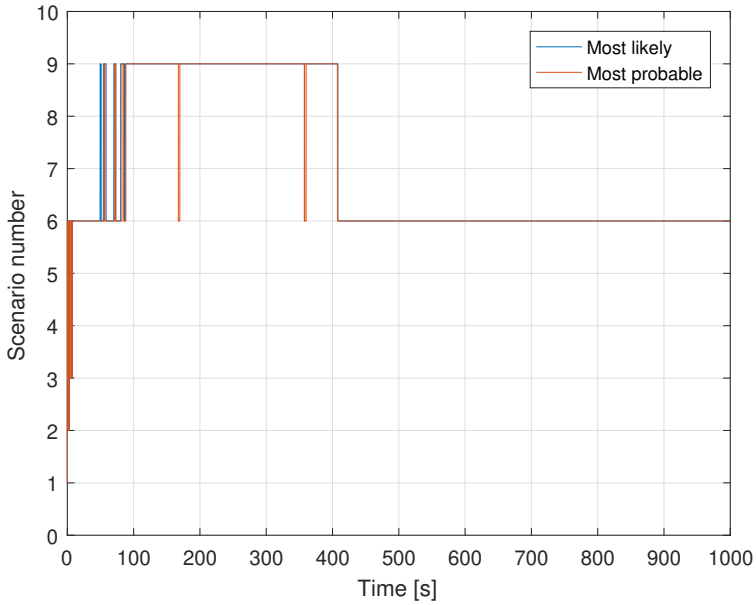


Figure 5.14: DHT and MLE estimates of correct scenario when line 6 breaks at $t = 400$ seconds, with current load applied

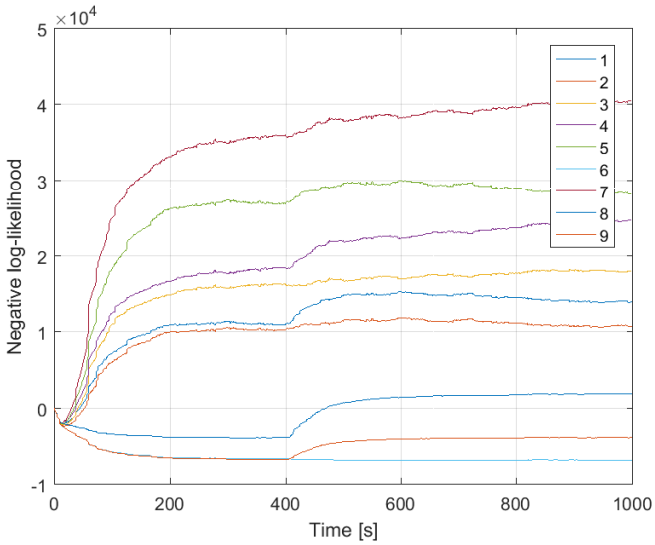


Figure 5.15: Negative log-likelihoods of all scenarios when line 6 breaks at $t = 400$ seconds, and current load is applied

5.5 Line 7 breaking

Figure 5.16 shows the position of the vessel when line 7 breaks at $t = 300$ seconds. The vessel can clearly be seen to move in the exact opposite direction of the failed line. Because the starting position and equilibrium are in line with the failing line's anchor the movement of the vessel is a straight line in this plot. See figure 5.17 for a time plot of the positions, where the motion becomes more clear. It shows that after the failure the vessel quickly drifts up to 35 meters away from the equilibrium, before the controller has time to react and move it back into position.

Figure 5.18 shows the DHT and MLE estimates of correct scenario. It shows that after line 7 breaks at $t = 300$ seconds, the DHT algorithm quickly determines that something has changed. It oscillates between the hypotheses for failure of line 3 and 4, the lines adjacent to line 7. After approximately 40 seconds it stabilizes on the correct hypothesis. The MLE method is slower to react, taking around 40 seconds to detect failure. It initially estimates scenario 3 to be correct, and then shortly changing to scenario 4 before settling on the correct scenario after approximately 80 seconds. This is slower than for DHT, but with less oscillations

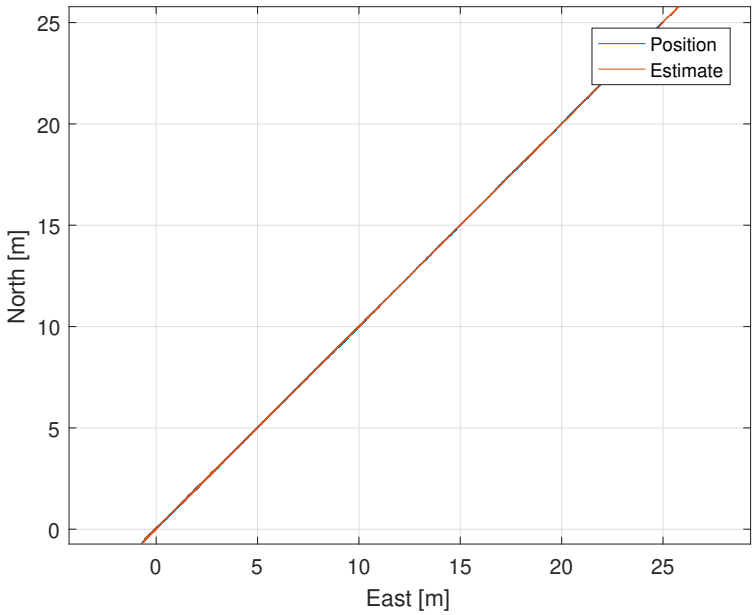


Figure 5.16: Position of the vessel when line 7 breaks at $t = 300$ seconds. Uses observer assuming no breakages.

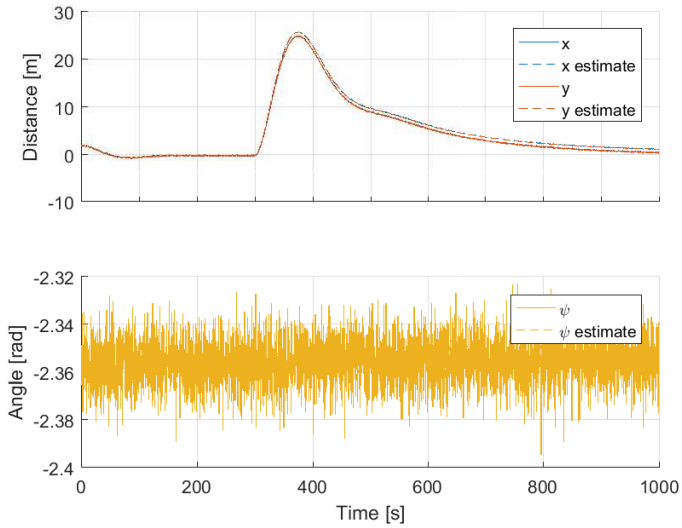


Figure 5.17: Time plot of vessel position when line 7 breaks at $t = 400$ seconds

between scenarios.

Figure 5.19 shows the negative log-likelihoods for all the failure scenarios. It shows that after the line break the correct scenario has a steeper descent than the other descending line (which is scenario 3 and 4, closely intertwined), but because it starts at a higher negative likelihood it takes time to overtake the other scenarios.

Figure 5.20 shows the probabilities for all the hypotheses when line 7 breaks at $t = 300$ seconds. It clearly shows the oscillation between hypothesis 3 and 4 right after the break occurs.

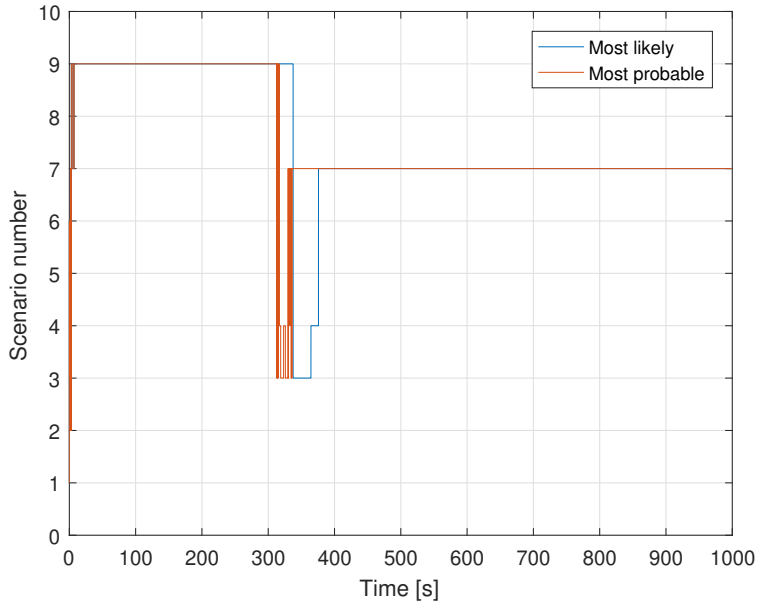


Figure 5.18: DHT and MLE estimates of correct scenario when line 7 breaks at $t = 300$ seconds

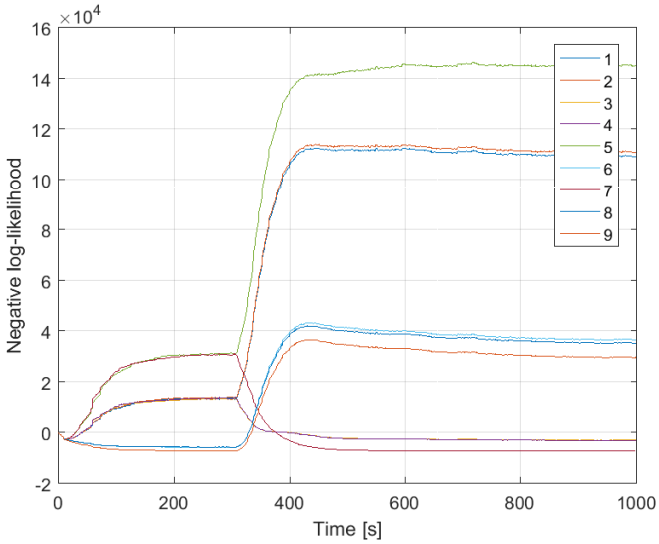


Figure 5.19: The negative log-likelihoods for all scenarios, when line 7 breaks at $t = 300$ seconds

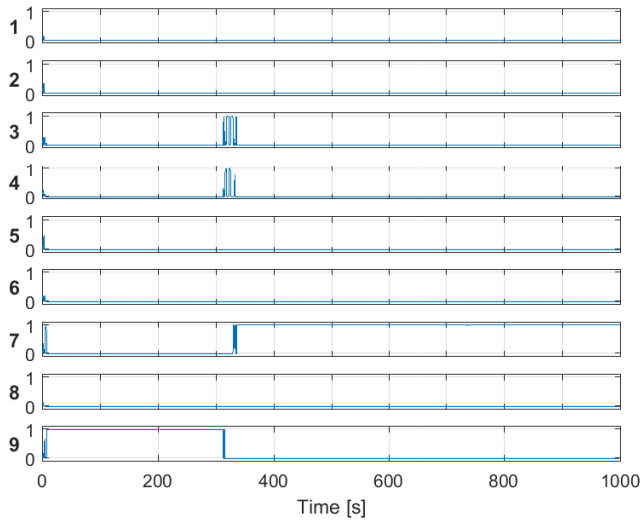


Figure 5.20: Probabilities of all hypotheses when line 7 breaks at $t = 300$ seconds

5.6 Line 7 breaking with current

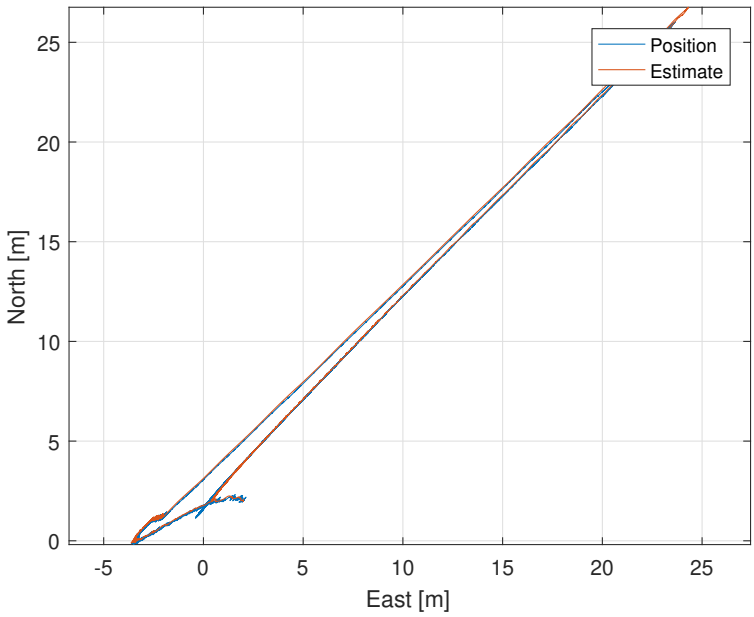


Figure 5.21: Position of the vessel when line 7 breaks at $t = 300$ seconds, and current is applied. Observer estimates assumes no breakages.

Figure 5.21 shows the position of the vessel when line 7 breaks at $t = 300$ seconds, with current load applied. Comparing with figure 5.16 it clearly shows the vessel being pushed to the north-west by the current. This movement also makes the moment when line 7 fails more apparent, as the vessel moves to the north-east.

Figure 5.22 shows the DHT and MLE estimates of correct scenario when line 7 fails at $t = 300$ seconds, with current load applied. As in figure 5.8 and 5.14 it shows that both algorithms estimate scenario 6 to be correct in the beginning, followed by oscillating between 6 and 9, before finally stabilizing on scenario 9. When the break occurs the behaviour is very similar to figure 5.18. The DHT algorithm changes very quickly, but to hypothesis 3, which is wrong. It changes to the correct hypothesis in the same time frame as in figure 5.18, but without oscillating between hypothesis 3 and 4. The MLE algorithm is again a bit slower, and also jumps to scenario 3 after the break. At around 100 seconds after the break it goes to the correct scenario. It is worth noting that the MLE estimate doesn't oscillate at all after the break, whilst the DHT estimate has occasional

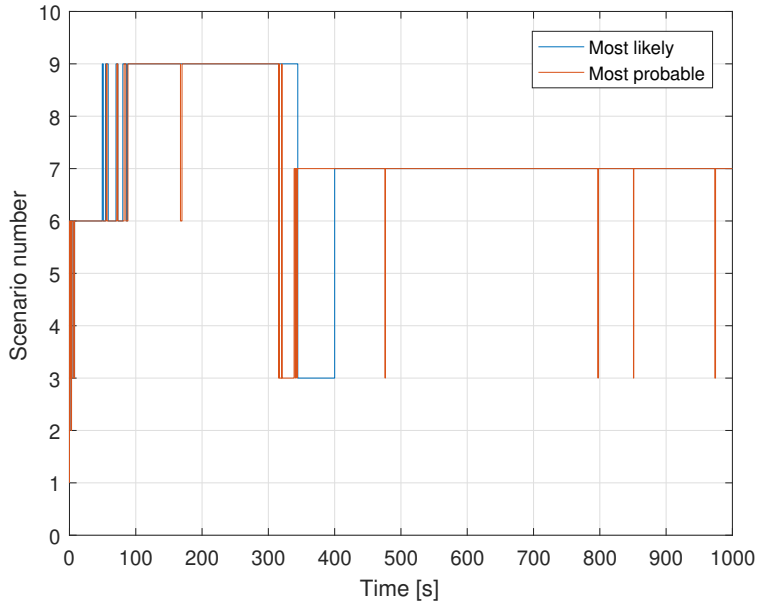


Figure 5.22: DHT and MLE estimates of correct scenario when line 7 breaks at $t = 300$ seconds, with current applied

spikes.

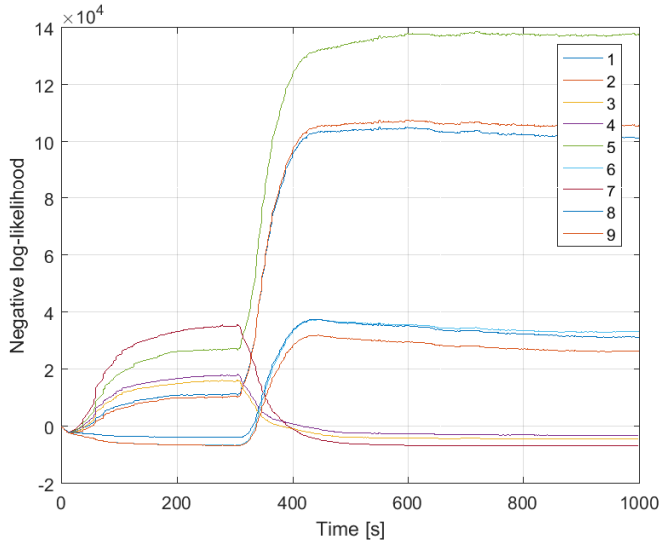


Figure 5.23: The negative log-likelihoods of all scenarios when line 7 breaks at $t = 300$ seconds, with current load applied

Figure 5.23 shows the negative log-likelihoods of all scenarios when line 7 breaks at $t = 300$ seconds, and current load is applied. It is very similar to figure 5.19. The most notable difference is that the likelihoods of scenario 3 and 4 are no longer intertwined.

5.7 Line 8 breaking

Figure 5.24 shows the position of the vessel when line 8 breaks at $t = 400$ seconds. It clearly shows the vessel moving in the opposite direction of the broken mooring line.

Figure 5.25 shows that both the DHT and MLE algorithms quickly determine the correct starting scenario, and are both able to detect the correct line failure. The MLE estimate is a bit slower to respond to the failure, taking almost 50 seconds. However, neither algorithm produce any oscillations between different scenarios, giving a very clear estimate.

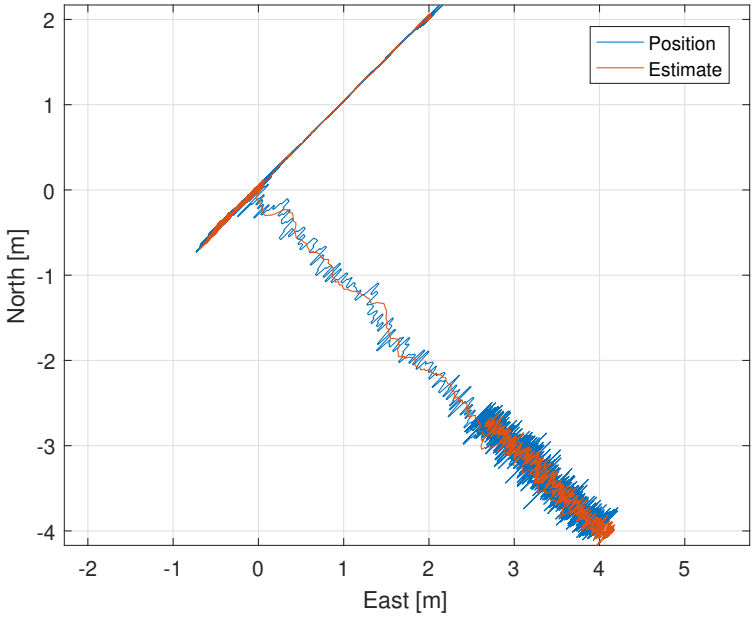


Figure 5.24: Position of the vessel when line 8 breaks at $t = 400$ seconds. Observer assumes no breakages.

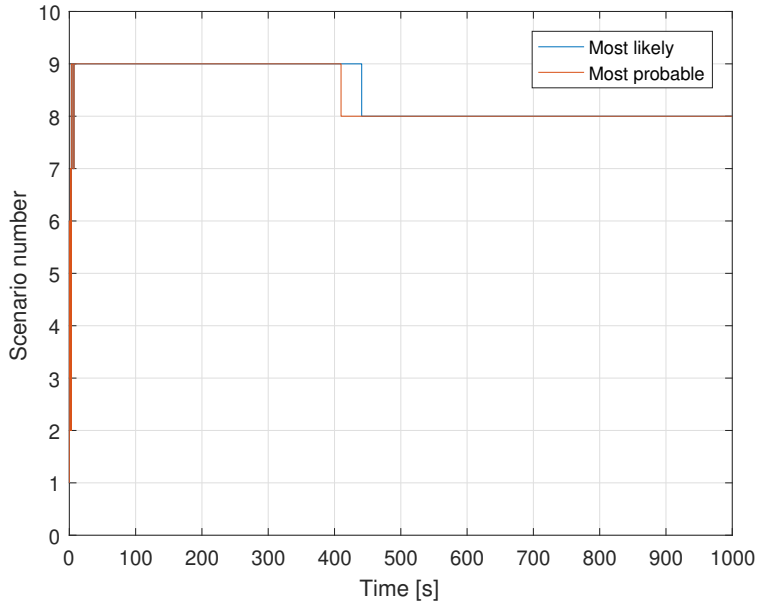


Figure 5.25: DHT and MLE estimates of correct scenario when line 8 breaks at $t = 400$ seconds

5.8 Line 8 breaking with current

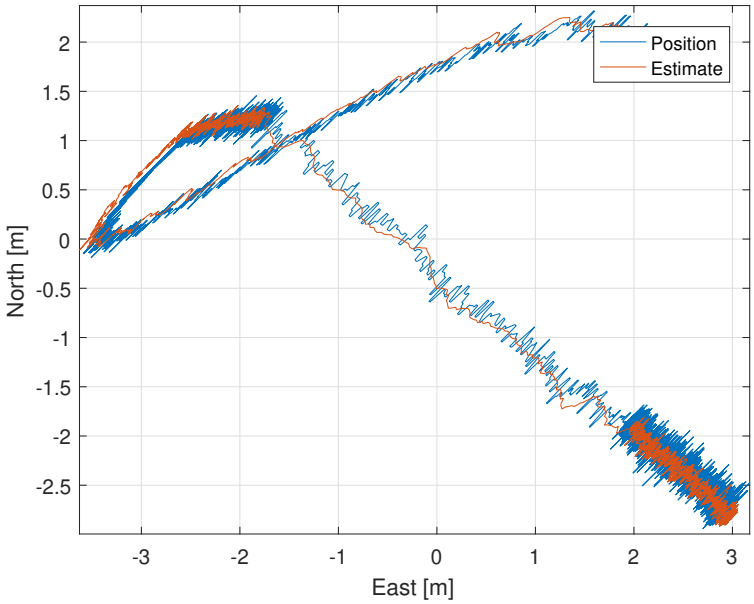


Figure 5.26: Position of the vessel when line 8 fails at $t = 400$ seconds, and current load is applied. Observer assumes no breakages.

Figure 5.26 shows the position of the vessel when line 8 breaks at $t = 400$ seconds, and the current load is applied. Note that the estimated position is from an observer assuming no line breakages. The motion of the vessel is clearly very similar to figure 5.24 where there was no current. The main difference is that the vessel is pushed further to the north-west before the failure. Also note that the offset from the equilibrium point never grows as large, as the current is pushing the vessel back towards the equilibrium point.

Figure 5.27 shows the DHT and MLE estimates of correct scenario, when current load is applied. It shows a similar picture to the other simulations with current, in that it takes a while for the algorithms to determine the correct scenario in the start. Even after the estimates settle there is a little bit of oscillation for the DHT method. After line 8 fails at $t = 400$ seconds, the DHT algorithm is able to quickly determine the correct scenario, much like in the no current simulation in figure 5.25. However, the MLE algorithm struggles a lot more than in the no current simulation, taking over 150 seconds to determine the correct failure scenario.

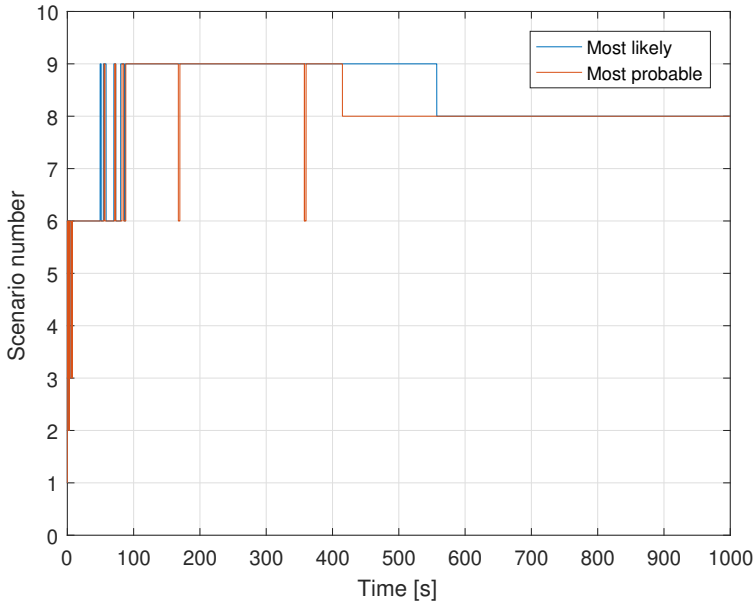


Figure 5.27: DHT and MLE estimates of correct scenario when line 8 breaks at $t = 400$ seconds, with current load applied

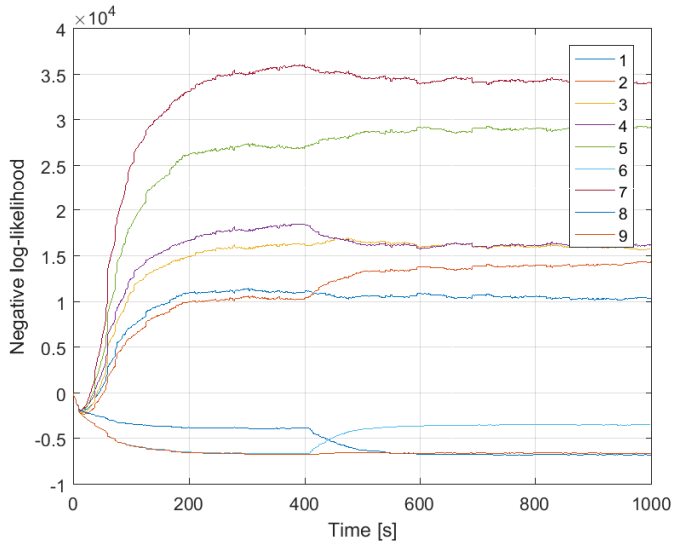


Figure 5.28: Negative log-likelihoods for all scenarios when line 8 fails at $t = 400$ seconds, and current load is applied

Figure 5.28 shows the negative log-likelihoods for all the scenarios, when line 8 fails at $t = 400$ seconds and the current load is applied. It shows that until the failure, there are two scenarios that have the lowest likelihoods, scenario 6 and 9, with scenario 9 being slightly more likely to be true. After line 8 breaks we see the negative log-likelihood of the now correct scenario 8 drop to become the smallest. However, while scenario 6 quickly rises to a higher steady level, scenario 9 sees almost no change in its likelihood after the failure. This prolongs the time it takes for the correct scenario to dip beneath its negative log-likelihood value.

Chapter 6

Discussion

6.1 Performance

6.1.1 General performance

In general the performance of the statistical methods is quite good. Both the DHT and MLE methods are able to detect line failures in a relatively short amount of time, with MLE usually being a bit slower than DHT. When there is a current load applied the performance deteriorates, introducing oscillations between multiple scenarios and longer reaction times. But even in these conditions both algorithms are able to detect the correct scenario.

The tunable parameters of the statistical methods, forgetting factor for MLE and minimum likelihood for DHT, greatly impact performance. These parameters are the main reason that MLE is a bit slower than DHT. If the forgetting factor is reduced to $C_{forget} = 0.99$ from $C_{forget} = 0.999$ the reaction time is on par with DHT. However, this causes larger oscillations in estimated scenario than DHT when current is applied. It is also possible to reduce oscillations for DHT by reducing the minimum probability, but again this comes at the cost of a longer reaction time to detect failures. The values that were used is viewed as a decent compromise between these two behaviours.

The good results were achieved despite the fact that the error covariance matrices used in the statistical methods were estimated offline. This was necessary since the passive observer does not calculate the error covariances. This shows that a

more complex observer utilising a Kalman filter is not necessary for this approach to work, although it is still quite possible it might provide better performance.

6.1.2 Performance with current

When the current load is applied the performance of the statistical methods suffer. Both the MLE and DHT estimates take a lot longer to settle on the no breakage scenario in the start of the simulations, often settling on scenario 6 first. This is probably because the current comes from the same direction as mooring line 6 is positioned in. The force from the current can therefore be interpreted as a loss of horizontal tension from mooring line 6, which makes a failure of that line appear to be likely. Even after both analysis methods settle on the correct scenario, they will sometimes briefly change to scenario 6. This is especially apparent for DHT, although this behaviour could probably be reduced by tuning the minimum probability to be smaller.

The current's effect on performance is also visible when a mooring line breaks, but here the effect is not always the same. Figure 5.14 shows that when line 6 breaks when there is current the estimates actually change faster compared to there being no current, as in figure 5.12. This is because the current makes a line 6 failure appear more likely in the MLE algorithm, and therefore the amount it needs to change to become the likeliest scenario is much smaller than if there was no current. This is very clear in figure 5.15. For the DHT algorithm the difference is not so pronounced, because the total probability must be 1, and all other scenarios than the most probable will therefore have quite small values.

When mooring line 8 breaks this effect is reversed: the MLE estimate is slower to change, which can be seen by comparing figure 5.25 with figure 5.27. This is caused by the current force compensating for the loss of mooring line tension, causing the loss of the mooring line to become less obvious. This is made apparent in figure 5.28, where the likelihood of scenario 9 can be seen to have almost no change after the break of line 8.

Both of these examples are caused by the fact that the MLE algorithm treats the likelihood of each scenario separately. Essentially this means that it is able to see multiple scenarios as likely at the same time. However, it is important to point out that in all the simulations with current the MLE algorithm always finds the correct scenario in the end. The direction of the current affects the response time, but doesn't make the algorithm fail.

6.2 Limitations

The model used does not take into account all of the loads on the vessel. In particular, it does not model the wind and current loads. These loads are gathered under the bias term, but there is no way of knowing if this term is correct as there is no way to measure the bias. In fact there are no measurements of any of the environmental loads in the observer model. Adding these measurements, either as an applied load or as states in the model with corresponding measurements in the $\boldsymbol{y}$ vector, would be an advantage.

The simulation software and observer model used only models simplistic failure scenarios, where the mooring lines are either fully functional or completely removed. This means that the DHT and MLE methods are not able to detect more complex failure scenarios, such as failures in the cable touchdown zone or on the seabed. These are the failures that are hardest to detect with existing technology, and if they are accounted for in the model it might be possible to detect them.

The simulations that were tested cover an extremely short time frame compared to the years a vessel can be moored at sea. This means that even though the simulations show that the DHT and MLE algorithms work, it has not been verified that they will continue to work for long durations. The tuning factor applied to the algorithms provide some assurance that they will remain unaffected by the amount of time passed, but it cannot be definitely demonstrated that they will not drift over time with these short simulations. Longer simulations, and ideally real world testing, will need to be performed to verify the performance.

Only one set of weather disturbance parameters were used in these simulations, in order for the results to be easily comparable. While these parameters represent a typical scenario, further testing is needed to ascertain the performance of the methods in all weather conditions.

The tuning of the forgetting factor for MLE and minimum probability for DHT greatly impact the performance of the two algorithms. It is hard to get this tuning right, as quick performance comes at the cost of less stable estimates. It might be necessary to retune these parameters for systems that are placed in different conditions. Not only might the different depths, mooring line configurations, weather conditions, etc. affect the performance, but different operators might have different requirements for the response time, and different levels of acceptability of false positives.

The DP controller used in the simulations is greatly simplified compared to commercial systems. In fact the controller is implemented and tuned only to be passable, and is not optimized in any way. While the model takes the thruster

forces into account in a very general way, it can not be guaranteed that the behaviour will stay the same when the controller changes.

The error covariance matrices used are calculated offline, using data from long simulations. While the performance using these estimated covariance matrices is good, it is quite possible that there are cases where the estimates do not reflect the real covariances. This could lead to erroneous estimates. Using an observer that is able to calculate the covariances online, like a nonlinear Kalman filter, might give a more robust performance.

6.3 Simulator implementation error

Towards the end of the work on this thesis an implementation error was discovered in the simulator software that was provided for simulating a TAPM vessel. The Simulink model that simulated the vessel adds measurement noise to the position. This was done by adding Gaussian white noise to each of the DOFs separately. The white noise generators used take a seed as an input. As random number generators are in reality pseudo random, this seed is what controls the output of the noise generator. If two generators are given the same seed they will produce the same pseudo random output. In the simulator provided, the seed for the different DOFs were set to the same value. This caused the noise in x and y direction to be identical, effectively reducing the possible noise values from a circle around the origin to a diagonal line. This is not realistic, and not the behaviour that was expected from using the simulator.

Fixing this error is very simple, just changing the seed for all the noise generators so they are different. However this obviously affected the performance of the observer and statistical analyses, as they were not tuned for this scenario. This caused the performance do decrease drastically. Unfortunately the error was discovered very close to the deadline, and there was simply no time to retune everything, redo the simulations, and rewrite the results and discussion of the thesis itself. The results seen in this thesis are therefore with this error still applied. The consequence of this is that even though the performance that was observed is good, it can not be interpreted as more than promising.

Chapter 7

Conclusion

The results presented are promising, showing that it is possible to detect simple mooring line failures using dynamic hypothesis testing and maximum likelihood estimation. In the simulations that were tested line failure detection would usually occur within a minute, while even in the worst scenario the failure was detected in less than three minutes. In general the performance is worse when the system is exposed to a current disturbance, compared to only a wave disturbance. However, due to an implementation error in the simulator, the results presented are not representative of a real world scenario.

Because the methodology used only requires the position measurement, which is usually already available on vessels with thruster assisted position mooring, it is well suited for a low-cost retrofit. Because of the simplistic failure modelling used this method will probably not be suitable to completely replace existing sensors. If it can be shown to work in a proper simulator it can however be a good supplement to existing technology, and especially well suited for vessels that currently have no mooring line monitoring.

Comparing MLE and DHT, the main difference is that DHT can only view one scenario as probable at the time, as the probabilities for all scenarios must sum to 1. MLE on the other hand treats all scenarios separately, and can therefore view multiple scenarios as likely. This can cause slower detection of failure for MLE, as the previous scenario is still viewed as likely after the failure in certain conditions. However the tuning of the statistical methods is the largest factor affecting the performance. The quicker response time of DHT can also be a downside, as it tends to temporarily change hypothesis away from the correct hypothesis in challenging conditions. This makes it difficult to determine which

of the methods are better.

7.1 Further work

Because of the error in implementing measurement noise in the simulator the results presented are not representative of a real world scenario. Correcting the mistake is luckily quite simple, so the time consuming work is retuning the observers and statistical methods. This will need to be done to validate the promising results.

The statistical methods need the error covariance matrix to estimate the correct scenario, because they depend on the Gaussian probability density function. This matrix is estimated offline because the passive observer cannot calculate it online. While the results are still good, using an observer that calculates the covariances online would be more correct, and might give better performance. A nonlinear variant of the Kalman filter is an obvious candidate.

The modelling of line failures is quite simplistic, assuming that the line is either fully connected or completely gone. This ignores more complex failure scenarios like failure on the seabed. These are the failures that are the most difficult to detect with current methods. Improving the model to include less obvious failure scenarios such as this would make it possible to test if these failures can also be detected using statistical methods.

The model could also be improved be incorporating more of the sensor measurements that might be available on a vessel. This is especially true for weather measurements like wind force and direction, or wave measurements if available. It could also be of interest to incorporate existing mooring line sensors, like tension measurement or sonar. Obviously this must be done in a robust fashion, so that problems with these sensors has a minimal effect on the performance of the statistical analysis.

The property of MLE that multiple scenarios can be viewed as likely is not very well utilized. The estimation simply looks at whichever scenario has the greatest likelihood, not taking into account if there are other scenarios that are just slightly less likely. Since this information is available, possible utilizations should be investigated. This could be a more intelligent selection of most likely scenario, or just presenting all scenarios considered likely in the operators interface.

Obviously testing in a simulation and in the real world are not the same thing. To validate this approach real world testing should be carried out. Because the system could be relatively simply installed as a retrofit the is not necessarily

prohibitive. The long term performance should also be investigated, as vessels are often moored for years at the time.

Bibliography

Ash, R. B. (1996). Lecture 21. the multivariate normal distribution. `https://faculty.math.illinois.edu/~r-ash/Stat/StatLec21-25.pdf`. 21.5 Finding the Density. Accessed on 2017-12-13.

D'Errico, J. (2012). putvar. `https://mathworks.com/matlabcentral/fileexchange/27106-putvar--uigetvar`. MathWorks file exchange.

DNV (2010). DNV-OS-E301 - Position Mooring.

Fossen, T. I. (2011). *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley.

Fossen, T. I. and Perez, T. (2004). Marine Systems Simulator (MSS). `http://www.marinecontrol.org`.

Fossen, T. I. and Strand, J. P. (1999). Passive nonlinear observer design for ships using lyapunov methods: full-scale experiments with a supply vessel. *Automatica*, 35(1):3–16.

GL Noble Denton (2017). Mooring failure detection systems for floating offshore installations. Research report 1097, Health and Safety Executive, UK.

Hassani, V., Pascoal, A. M., and Sørensen, A. J. (2018). Detection of mooring line failures using dynamic hypothesis testing. *Ocean Engineering*, 159:496–503.

Hassani, V., Sørensen, A. J., and Pascoal, A. M. (2013). Adaptive wave filtering for dynamic positioning of marine vessels using maximum likelihood identification: Theory and experiments. *IFAC Proceedings Volumes*, 46(33):203–208.

Mathworks (2017). *Write Level-2 MATLAB S-Functions*, 2016b edition. Accessed on 2017-12-13.

Noble Denton Europe Limited (2006). Floating production system: JIP FPS mooring integrity. Research report 444, Health and Safety Executive, UK.

Ren, Z. (2015). Fault tolerant control of thruster-assisted position mooring system. Master's thesis, NTNU.

Ren, Z., Skjetne, R., and Hassani, V. (2015). Supervisory control of line breakage for thruster-assisted position mooring system. *IFAC PapersOnLine*, 48(16):235–240.

Walpole, R. E., Myers, R. H., Myers, S. L., and Ye, K. (2014). *Probability and statistics for engineers and scientists*. Pearson Education, Harlow, 9th ed. edition.

Appendices

Appendix A

MATLAB code

A.1 Dynamic Hypothesis Testing

Listing A.1: `DHT.m`

```
function DHT(block)

%
% The setup method is used to set up the basic attributes of the
% S-function such as ports, parameters, etc. Do not add any other
% calls to the main body of the function.
%
setup(block);

end


function setup(block)

data = block.DialogPrm(1).Data;
num_states = data{1}.num_states;
num_measured_states = data{1}.num_measured_states;
num_hyp = length(data);

% Register number of ports
block.NumInputPorts  = 3;
block.NumOutputPorts = 1;

% Setup port properties to be inherited or dynamic
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;
```

```
27
28 % Override input port properties
29 % y measurement
30 block.InputPort(1).Dimensions   = [num_measured_states, 1];
31 block.InputPort(1).DatatypeID   = 0;  % double
32 block.InputPort(1).Complexity   = 'Real';
33 block.InputPort(1).SamplingMode = 'Sample';
34
35 % x_hat estimate
36 block.InputPort(2).Dimensions   = [num_states * num_hyp, 1];
37 block.InputPort(2).DatatypeID   = 0;  % double
38 block.InputPort(2).Complexity   = 'Real';
39 block.InputPort(2).SamplingMode = 'Sample';
40
41 % P_priori
42 block.InputPort(3).Dimensions   = [num_states * num_hyp, num_states
       ];
43 block.InputPort(3).DatatypeID   = 0;  % double
44 block.InputPort(3).Complexity   = 'Real';
45 block.InputPort(3).SamplingMode = 'Sample';
46
47 % Override output port properties
48 % Probabilities
49 block.OutputPort(1).Dimensions  = [num_hyp, 1];
50 block.OutputPort(1).DatatypeID  = 0; % double
51 block.OutputPort(1).Complexity  = 'Real';
52 block.OutputPort(1).SamplingMode = 'Sample';
53
54 % Register parameters
55 block.NumDialogPrms     = 1;
56
57 % Register sample times
58 block.SampleTimes = [-1 0];
59
60 % Specify the block simStateCompliance.
61 block.SimStateCompliance = 'DefaultSimState';
62
63 % Specifiy which functions are used
64 block.RegBlockMethod('CheckParameters', @CheckParameters);
65 block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
66 block.RegBlockMethod('Start', @Start);
67 block.RegBlockMethod('Outputs', @Outputs);     % Required
68 block.RegBlockMethod('Update', @Update);
69 block.RegBlockMethod('Terminate', @Terminate); % Required
70
71 end
72
73 % Check the validity of a MATLAB S-Function's parameters
74 function CheckParameters(block)
75
76 data = block.DialogPrm(1).Data;
77
```

```
78  if ~iscell(data)
79      error('Parameter is not a struct')
80  elseif length(data) < 1
81      error('Parameter is not long enough')
82  elseif ~isfield(data{1}, 'C')
83      error('The field C is missing from the parameter data')
84  elseif ~isfield(data{1}, 'R')
85      error('The field R is missing from the parameter data')
86  elseif ~isfield(data{1}, 'num_states')
87      error('The field num_states is missing from the parameter data')
88  elseif ~isfield(data{1}, 'num_measured_states')
89      error('The field num_measured_states is missing from the
        parameter data')
90  end
91
92  end
93
94
95  % Setup work areas and state variables
96  function DoPostPropSetup(block)
97  block.NumDworks = 1;
98
99  data = block.DialogPrm(1).Data;
100 num_hyp = length(data);
101
102 block.Dwork(1).Name             = 'probs';
103 block.Dwork(1).Dimensions       = num_hyp;
104 block.Dwork(1).DatatypeID       = 0;      % double
105 block.Dwork(1).Complexity       = 'Real'; % real
106 block.Dwork(1).UsedAsDiscState = true;
107
108 end
109
110 % Called once at start of model execution. If you have states that
        should
111 % be initialized once, this is the place to do it.
112 function Start(block)
113
114 data = block.DialogPrm(1).Data;
115 num_hyp = length(data);
116
117 % Assume equal probability for each hypothesis
118 block.Dwork(1).Data = 1/num_hyp * ones(num_hyp, 1);
119 end
120
121
122 % Called to generate block outputs in simulation step
123 function Outputs(block)
124
125 % Output the probabilities
126 block.OutputPort(1).Data = block.Dwork(1).Data;
127 end
```

```
128
129
130 % Called to update discrete states during simulation step
131 function Update(block)
132
133 % Find number of states
134 data = block.DialogPrm(1).Data;
135 num_hyp = length(data);
136 num_states = data{1}.num_states;
137 num_ms = data{1}.num_measured_states;
138 probs_old = block.Dwork(1).Data;
139 lower_limit = 10^-12;
140
141 y = block.InputPort(1).Data;
142
143 % Calculate the numerator for each hypotheses, and sum to get
        denominator
144 prob_top = zeros(num_hyp, 1);
145 prob_bot = 0;
146 for i = 1:num_hyp
147     C = data{i}.C;
148     R = data{i}.R;
149     P = block.InputPort(3).Data(1 + (i-1)*num_states : i*num_states,
          :);
150     x_hat = block.InputPort(2).Data(1 + (i-1)*num_states : i*
        num_states, :);
151
152     y_hat = C * x_hat;
153     y_tilde = y - y_hat;
154     S = C * P * C' + R;
155
156     prob_top(i) = probs_old(i) * exp(-0.5 * y_tilde' * inv(S) *
        y_tilde) / ...
157         sqrt((2*pi)^num_ms * det(S));
158     prob_bot = prob_bot + prob_top(i);
159
160 end
161
162 % Enforce a lower limit on the probabilities
163 prob = prob_top ./ prob_bot;
164 prob_adj = max(prob, lower_limit);
165 prob_norm = prob_adj / sum(prob_adj);
166
167 % Store the final probability vector in the internal state
168 block.Dwork(1).Data = prob_norm;
169
170 end
171
172 % Called at the end of simulation for cleanup
173 function Terminate(block)
174
175 end
```

A.2 Maximum Likelihood Estimation

Listing A.2: `MLE.m`

```
function MLE(block)

%
% The setup method is used to set up the basic attributes of the
% S-function such as ports, parameters, etc. Do not add any other
% calls to the main body of the function.
%
setup(block);

end


function setup(block)

data = block.DialogPrm(1).Data;
num_states = data{1}.num_states;
num_measured_states = data{1}.num_measured_states;
num_hyp = length(data);

% Register number of ports
block.NumInputPorts  = 3;
block.NumOutputPorts = 1;

% Setup port properties to be inherited or dynamic
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;

% Override input port properties
% y measurement
block.InputPort(1).Dimensions   = [num_measured_states, 1];
block.InputPort(1).DatatypeID   = 0;  % double
block.InputPort(1).Complexity   = 'Real';
block.InputPort(1).SamplingMode = 'Sample';

% x_hat estimate
block.InputPort(2).Dimensions   = [num_states * num_hyp, 1];
block.InputPort(2).DatatypeID   = 0;  % double
block.InputPort(2).Complexity   = 'Real';
block.InputPort(2).SamplingMode = 'Sample';

% P_priori
block.InputPort(3).Dimensions   = [num_states * num_hyp, num_states
    ];
block.InputPort(3).DatatypeID   = 0;  % double
block.InputPort(3).Complexity   = 'Real';
block.InputPort(3).SamplingMode = 'Sample';

```

```
47 % Override output port properties
48 % Negative Log-Likelihoods
49 block.OutputPort(1).Dimensions  = [num_hyp, 1];
50 block.OutputPort(1).DatatypeID  = 0; % double
51 block.OutputPort(1).Complexity  = 'Real';
52 block.OutputPort(1).SamplingMode = 'Sample';
53
54 % Register parameters
55 block.NumDialogPrms     = 1;
56
57 % Register sample times
58 block.SampleTimes = [-1 0];
59
60 % Specify the block simStateCompliance.
61 block.SimStateCompliance = 'DefaultSimState';
62
63 % Specifiy which functions are used
64 block.RegBlockMethod('CheckParameters', @CheckParameters);
65 block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
66 block.RegBlockMethod('Start', @Start);
67 block.RegBlockMethod('Outputs', @Outputs);     % Required
68 block.RegBlockMethod('Update', @Update);
69 block.RegBlockMethod('Terminate', @Terminate); % Required
70
71 end
72
73 % Check the validity of a MATLAB S-Function's parameters
74 function CheckParameters(block)
75
76 data = block.DialogPrm(1).Data;
77
78 if ~iscell(data)
79     error('Parameter is not a cell array')
80 elseif length(data) < 1
81     error('Parameter is not long enough')
82 elseif ~isfield(data{1}, 'C')
83     error('The field C is missing from the parameter data')
84 elseif ~isfield(data{1}, 'R')
85     error('The field R is missing from the parameter data')
86 elseif ~isfield(data{1}, 'num_states')
87     error('The field num_states is missing from the parameter data')
88 elseif ~isfield(data{1}, 'num_measured_states')
89     error('The field num_measured_states is missing from the
       parameter data')
90 elseif ~isfield(data{1}, 'forgetting_factor')
91     error('The field forgetting_factor is missing from the parameter
        data')
92 end
93
94 end
95
96
```

```
97  % Setup work areas and state variables
98  function DoPostPropSetup(block)
99  block.NumDworks = 1;
100
101 data = block.DialogPrm(1).Data;
102 num_hyp = length(data);
103
104 block.Dwork(1).Name             = 'Likelihoods';
105 block.Dwork(1).Dimensions       = num_hyp;
106 block.Dwork(1).DatatypeID       = 0;      % double
107 block.Dwork(1).Complexity       = 'Real'; % real
108 block.Dwork(1).UsedAsDiscState = true;
109
110 end
111
112 % Called once at start of model execution. If you have states that
        should
113 % be initialized once, this is the place to do it.
114 function Start(block)
115
116 data = block.DialogPrm(1).Data;
117 num_hyp = length(data);
118
119 % Accumulated likelihood starts at 0
120 block.Dwork(1).Data = zeros(num_hyp, 1);
121 end
122
123
124 % Called to generate block outputs in simulation step
125 function Outputs(block)
126
127 % Output the likelihoods
128 block.OutputPort(1).Data = block.Dwork(1).Data;
129
130 end
131
132
133 % Called to update discrete states during simulation step
134 function Update(block)
135
136 % Find number of states
137 data = block.DialogPrm(1).Data;
138 num_hyp = length(data);
139 num_states = data{1}.num_states;
140
141 likelihoods = block.Dwork(1).Data;
142
143 y = block.InputPort(1).Data;
144
145 for i = 1:num_hyp
146     C = data{i}.C;
147     R = data{i}.R;
```

```
148     P = block.InputPort(3).Data(1 + (i-1)*num_states : i*num_states,
          :);
149     x_hat = block.InputPort(2).Data(1 + (i-1)*num_states : i*
        num_states, :);
150     forget = data{i}.forgetting_factor;
151
152     y_hat = C * x_hat;
153     y_tilde = y - y_hat;
154     S = C * P * C' + R;
155
156     likelihoods(i) = likelihoods(i) * forget + 0.5 * (log(det(S)) +
        y_tilde' * inv(S) * y_tilde);
157 end
158
159 block.Dwork(1).Data = likelihoods;
160
161 end
162
163 % Called at the end of simulation for cleanup
164 function Terminate(block)
165
166 end
```

A.3 Initialisation

Listing A.3: `initSim.m`

```
function initSim()
%initialise all parameters needed for simulation

load('MLP.mat')
load('T_H_table.mat')
load('vessel.mat')
load('vesselABC.mat')

initial_eta = [2, 2, 0, 0, 0, -135/180*pi];
initial_velocity = zeros(1, 6);
initial_bias = zeros(1, 6);

epsilon_0_PassObs    =    [0;0;0;0;0;0];
eta_w_0_PassObs      =    epsilon_0_PassObs(1:3);
xi_w_0_PassObs       =    epsilon_0_PassObs(4:6);

anchor_pos = MLP.r_0;

putvar(MLP, T_H_table, vessel, vesselABC, initial_eta,
    initial_velocity,...
    initial_bias, epsilon_0_PassObs, eta_w_0_PassObs, xi_w_0_PassObs
    , anchor_pos)


%% Passive observer
passiveData = initPassiveObs(vessel, vesselABC, initial_eta,
    initial_velocity, initial_bias);
putvar(passiveData)


%% Controller
Kp = diag([100000, 1000000, 1000]);
Ki = diag([1000, 1000, 100]);
Kd = 0;
putvar(Kp, Ki, Kd)


end
```

A.4 Observer initialization

Listing A.4: `initPassiveObs.m`

```
function data = initPassiveObs(vessel, vesselABC, initial_eta,
    initial_velocity, initial_bias)

num_moor = 8;

%% Wave parameters

omega_0 = 2 * pi / 7 * ones(1, 3);
omega_c = 1.1 * ones(1, 3);

zeta_n = 1 * ones(1, 3);   % Fossen (2011) p. 317
lambda = 0.1 * ones(1, 3); % Fossen (2011) p. 317

OMEGA = diag(omega_0 .^ 2);
LAMBDA = diag(2 * zeta_n .* omega_0);

Aw = [zeros(3), eye(3);
    -OMEGA, -LAMBDA];
Cw = [zeros(3), eye(3)];


%% Observer Gains

K1_temp = zeros(1, 6);
for i = 1:3
    K1_temp(i) = -2 * (zeta_n(i) - lambda(i)) * (omega_c(i) /
    omega_0(i));
    K1_temp(i + 3) = 2 * omega_0(i) * (zeta_n(i) - lambda(i));
end

K1 = [diag(K1_temp(1:3));
    diag(K1_temp(4:6))];
K2 = omega_c;
K4 = diag([0.3, 0.3, 0.01]); % Zhengru: diag([0.3,0.3,0.01])
K3 = 0.1 * K4;


%% Vessel damping

freqIndex = 34;
DOF = [1, 2, 6];
D = vessel.B(DOF, DOF, freqIndex);
Dmo = 0.1 * D;
D = D + Dmo;


%% Bias time constants
```

```
46
47 Tb = diag(1000 * ones(1, 3));
48 invTb = inv(Tb);
49
50
51 %% Vessel mass
52
53 M_rigidBody = vessel.MRB(DOF, DOF);
54 M_addedMass = vesselABC.MA(DOF, DOF);
55 M = M_rigidBody + M_addedMass;
56 invM = inv(M);
57
58
59 %% Noises
60 measurement_var = 0.005;
61
62
63 %% Observer data for different hypotheses
64
65 x_hat_0 = transpose([initial_eta(DOF), initial_velocity(DOF),
       initial_bias(DOF)]);
66 load('custom\error_covariances.mat');
67
68 num_states = 3;
69 num_measured_states = 3;
70 num_hyp = num_moor + 1;
71 data = cell(1, num_hyp);
72 for i = 1:num_hyp
73     data{i}.Aw = Aw;
74     data{i}.Cw = Cw;
75     data{i}.C = eye(num_states);
76     data{i}.D = D;
77     data{i}.lambda = LAMBDA;
78     data{i}.omega_0 = omega_0;
79     data{i}.omega_c = omega_c;
80
81     Lp = diag(ones(1, num_moor));
82     if i <= num_moor % Line i is broken
83         Lp(i, i) = 0;
84     end
85     data{i}.Lp = Lp;
86
87     data{i}.K1 = K1;
88     data{i}.K2 = K2;
89     data{i}.K3 = K3;
90     data{i}.K4 = K4;
91
92     data{i}.M = M;
93     data{i}.invM = invM;
94     data{i}.Tb = Tb;
95     data{i}.invTb = invTb;
96
```

```
97      data{i}.H_num = i;
98      data{i}.x_hat_0 = x_hat_0;
99      data{i}.P = error_covariances{i};
100     data{i}.R = diag(measurement_var .* ones(1, num_measured_states)
        );
101     data{i}.forgetting_factor = 0.99;
102
103     data{i}.num_states = num_states;
104     data{i}.num_measured_states = num_measured_states;
105 end
106 end
```

A.5 Generating error covariances

Listing A.5: `generate_error_covariances.m`

```
function generate_error_covariances(runtime, generate_new)

error_covariances = cell(1, 9);

for hyp = 1:9
    filename = sprintf('custom/P_gen/y_error%d.mat', hyp);
    if generate_new
        fprintf('%s - Running simulation for hypothesis %d.\n',
    datestr(datetime('now')), hyp)
        controller = 1;
        breaktime = 0;
        [~, ~, ~, ~, y_error, ~, ~, ~, ~] = runSim(runtime,
    controller, hyp, breaktime, 1, 0);
        save(filename, 'y_error')
    else
        load(filename)
    end
    %calculate
    start = (hyp - 1) * 3 + 1;
    error = y_error(:, start:start+2);

    error_covariances{hyp} = cov(error);

end

save('custom/P_gen/error_covariances.mat', 'error_covariances')
fprintf('Remember to copy the new covariances from the P_gen folder
    to the custom folder\n')

end
```