TMR4150 - Marine Control Systems, Specialization Project: Development of a modularized control architecture for CS Enterprise I for path-following based on LOS and maneuvering theory

Nam Dinh Tran (dinh.ntran@gmail.com)

NTNU, Trondheim June 17, 2013

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

PROJECT DESCRIPTION SHEET

Name of the candidate: Dinh N. Tran

Field of study: Marine control engineering.

Thesis title (Norwegian): Utvikling av en modularisert kontroll-arkitektur for CS Enterprise I for banefølging basert på LOS- og manøvreringsteori.

Thesis title (English): Development of a modularized control architecture for CS Enterprise I for path-following based on LOS and maneuvering theory.

Background

The maneuvering control problem was defined in 2002, providing a novel framework for solving path-following problems for a wide variety of dynamical systems. By dividing the path-following problem into a geometric and dynamic task, the methodology provides means to construct intelligent control and guidance laws, with improved performance in terms of robustness and transient behavior. Within the field of marine technology, several applications have been reported in the literature, including pipe-laying operations, transit operations, and cooperative formation control for groups of surface vessels.

Recent developments have resulted in a generalization of the original maneuvering problem. Instead of focusing solely on one-dimensional paths, the objective is to ensure that the output of the controlled system converges to any desired manifold. This extension provides greater flexibility, effectively extending the possible applications of the design methodology. Although several applications already have been documented for marine vessels, including formation control, Line-Of-Sight (LOS)-based guidance and control, and extensions to path-following, few experiments have yet been conducted. It is therefore of high interest to establish a flexible experimental environment for testing general maneuvering controllers. This project aims at taking the first steps towards developing such a setup on CS Enterprise I (CSE1), a model ship purchased and built by NTNU to be used for demonstrations and student experiments at the Marine Cybernetics Laboratory (MC Lab).

Work description

1. Familiarization with the hardware and software of CSE1. Milestone: To get the control system made by Håkon Skåtun to run.
2. To propose a new modularized HW/SW architecture for CSE1. This should be clearly divided in control levels (real-time control loops; high-level control; HMI functions; etc.) Milestone: To get a Manual Thruster Control mode to run, also when outside the MC Lab.
3. Allocate time in the MC Lab for experiments in the spring 2013.
4. Perform a (literature) study on the maneuvering control framework. The goal should be to understand the main features of the framework, including how/why the filtered/unfiltered gradient optimization works.
5. Write a list with definitions and descriptions of relevant terms and concepts
6. Investigate and discuss how a general maneuvering control system can be implemented /modularized within the high-level control layer of the proposed SW architecture for CSE1.
7. To study the LOS-based maneuvering formation control design by C. F. L. Thorvaldsen. Perform modeling and simulation for CSE1 using a simple straight-line path and full-state feedback.
   Note: By using the formation controller for a single ship, the design effectively reduces to a generic maneuvering LOS controller.
8. Discuss and prepare for implementation of the LOS-based maneuvering controller, considering:
   8.1. need for parameter calculations and system identification,
   8.2. requirements for a sufficient guidance system for generating and presenting the desired path,
   8.3. specification of the LOS-based maneuvering controller,
   8.4. need of an exponentially stable observer, and
   8.5. HMI functions.

Guidelines
The scope of work may prove to be larger than initially anticipated. By the approval from the supervisor, described topics may be deleted or reduced in extent without consequences with regard to grading.

The candidate shall present his personal contribution to the resolution of problems within the scope of work. Theories and conclusions should be based on mathematical derivations and logic reasoning identifying the various steps in the deduction.

The report shall be organized in a rational manner to give a clear exposition of results, assessments, and conclusions. The text should be brief and to the point, with a clear language. The report shall be written in English (preferably US) and contain the following elements: Abstract, acknowledgements, table of contents, main body, conclusions with recommendations for further work, list of symbols and acronyms, references and (optional) appendices. All figures, tables, and equations shall be numerated. The original contribution of the candidate and material taken from other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a Harvard citation style (e.g. *natbib* Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct. Such practice is taken very seriously by the university and will have consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed upon.

The thesis shall be submitted in 3 printed copies, each signed by the candidate. The final revised version of this project description must be included. The report must appear in a bound volume or a binder according to the NTNU standard template. Computer code and a PDF version of the report should be included electronically.

A 15 min. presentation (conference style) on your status, intermediate results, and plan for completion is expected to be delivered at a scheduled time midway into the project period.

Start date: August, 2012 **Due date:** As specified by the administration.

Supervisor: Roger Skjetne
Co-advisor(s): Christoffer F. L. Thorvaldsen (ph.d. candidate)

Trondheim,

_____

Roger Skjetne
Supervisor

Summary

A literature study was done on the maneuvering framework, applications of it and line of sight. The maneuvering framework approach the problem as two separate tasks, *Geometric task* and *Dynamic task*. The primary objective is the geometric task, follow a reference target or a desired path, while the secondary objective is the dynamic task, being a time, speed or acceleration assignment. This separation allows flexibility in the design of the control law as the motions can be shaped by state feedback.

The software development was done in three stages, the first stage focused on the manual control, how it recognisances the input controller, received and the interaction between, PlayStation 3 controller, Windows XP, LabVIEW, SIT Manager, Real-Time Workshop and Simulink. The second stage concentrated on the 3D visualization of the system in LabVIEW and the communication between LabVIEW, Simulink and Qualisys. The third and final stage established the necessary framework for expanding the LabVIEW interface and Simulink model to accommodate multiple controller, plants and systems.Several of the bugs in Skåtun's programs were resolved, such as the shuffling of Qualisys data.

Towing of CyberShip Enterprise 1 was conducted for 0, 45 and 90 degrees and thruster force measurements were done to improve the system identification. 234 towing data files and 120 thruster force data files are yet to be processed.

Lastly plans for future work and suggestions were stated

Preface

The work presented in this report is a documentation on the preparation done for the master thesis that will be written in the fall of 2013 at the Norwegian University of Science and Technology (NTNU), under guidance of Professor Roger Skjetne, and Ph.D candidate Øivind Kåre Kjerstad.

First of all I would like to thank my supervisor Professor Roger Skjetne, this project report could not have been done without his support and guidance.

I would also like to express my gratitude to my co-advisors, Ph.d candidate Øivind Kåre Kjersta and former Ph.d candidate Christoffer F. L. Thorvaldsen for their support and follow-ups in solving practical to theoretical problems, to Torgeir Wahl Senior Engineer at Department of Marine Technology for teaching me the practical aspects related to the MCLAb, and to Mika Sundland for being my lab-partner.

Contents

List of Abbreviation

BtSix	Bluetooth Sixaxis
CSE1	CyberShip Enterprise 1
DOF	Degree of freedom
DP	Dynamic positioning
ESC	Electronic Speed Control
HIL	Hardware In the Loop
HMI	Human-Machine-Interface
IR	Infra-red
LOS	Line-Of-Sight
LV	LabVIEW
MCLab	Marine Cybernetics Laboratory
NTNU	Norwegian University of Science and Technology
PPJoy	Parallel Port Joystick
PS3	Playstation 3
PWM	Pulse Width Modulation
QS	Qualisys
QTM	Qualisys Track Manager
SIT	Simulation interface toolkit
VSP	Voith Schneider Propeller
WiFi	Wireless fidelity

Chapter 1

Introduction

The gyrocomapss invented in 1908 made it possible for feedback control system for heading and are known today as autopilots. The possibilities for automating are increasing as the global coverage of navigation systems develops. The maneuvering control problem was defined in 2002 and the maneuvering framework was formalized in Skjetne (2005). This framework separates the control problem into geometric and dynamic task, proving a methodology to design intelligent control and guidance laws, improving performance in robustness and transient behavior. Several application have been published for this framework, but few experiments have been conducted with it.

The objective of this project is to prepare and improve the infrastructure of CyberShip Enterprise 1 for trial and study of design using the maneuvering control framework.

1.1 Scope

The scope of this project is to give an in depth study of the infrastructure of CyberShip Enterprise 1, propose and implement improvements, modularize the control architecture and prepare it for a Line-of-Sight path-following controller

The remainder of this report is organized as follows:

Chapter 2: A summary of the literature study preformed related to path-following, maneuvering theory and Line-of-Sight guidance.

Chapter 3: An in depth study of the CyberShip Enterprise 1's infrastructure

Chapter 4: Step by step software development in LabVIEW and Simulink

Chapter 5: Review of system identification on CyberShip Enterprise 1

Chapter 6: Report on activities done related CyberShip Enterprise 1

Chapter 7: Identifying the different task that needs to be done and sketching out a schedule

Chapter 2

Literature study

This chapter contains summaries of the literature study done related to thesis, with a slightly different notation compared to the source material. It begins with a general introduction to *maneuvering*, covering problem statement, path parametrization and gradient optimization. Then continue with an application of *maneuvering* for formation control. The work presented in this chapter belongs to the respective authors. Section 2.2 are extracts from Skjetne (2005) and Section 2.3 are extracts from Thorvaldsen (2011)

2.1 Preliminaries

The notations and reference frames corresponds with what is found in Skjetne's and Thorvaldsen's work.

2.1.1 Notations

Time derivatives of *x(t)* are denoted as $\dot{x}$, $\ddot{x}$, $x^{(3)}$, ... , $x^{(n)}$, while partial differentiation: $\alpha^t(x,\theta,t) := \frac{\partial \alpha}{\partial t}$, $\alpha^{x^2}(x,\theta,t) := \frac{\partial^2 \alpha}{\partial x^2}$ and $\alpha^{\theta^n}(x,\theta,t) := \frac{\partial^n \alpha}{\partial \theta^n}$. The Euclidean vector norm $|x| := (x^T x)^{1/2}$ and stacking vectors into one is denoted as *col(x,y,z)* $:= [x^T, y^T, z^T]^T$.

2.1.2 Reference frames

The $\mathcal{E}$-frame is an earth-fixed tangent frame on the sea surface, with x-axis pointing north and y-axis pointing east.

The $\mathcal{B}$-frame moves and rotates with the vessel. The origin is in the principles plane of symmetry, with x-axis pointing from stern to bow, y-axis from port to starboard and z-axis top to bottom

The $\mathcal{R}$-frame has its origin at a point along a path. The x-axis is directed along the tangent vector and the y-axis in the orthogonal direction according the the right-hand-rule.

2.2 Maneuvering framework

In *maneuvering* the main objective is to converge and follow a desired path and the dynamic behavior is considered a secondary objective. These objectives are treated a two separate task. The primary task is to steer the vessel toward and along a desired path parametrized by a scalar path variable s. The secondary task uses s as a state variable to satisfy dynamic specifications along the path. This section is a summary of maneuvering problem found in Skjetne (2005), see Skjetne (2005) and Fossen (2002) for a more thorough treatment.

2.2.1 Path parametrization

There are many possibilities to parametrize a path. It can be continuous, discrete, or hybrid,a mixture of both. Different applications and control design will require different ways of parametrization. The parametrizations this subsection are exemplified for straight lines.

Discrete parametrization[1]

A *discrete parametrization* creates n line-segments l_i identified by an index $i \in \mathcal{I} = \{1, 2, \ldots, n\}$. Each segment have a local path reference frame $\mathcal{R}_i$ with origin attach at the segment's start-point $\boldsymbol{p}_i$ and x-axis pointed toward $\boldsymbol{p}_{i+1}$. Let $\boldsymbol{\epsilon}_1 :=$ col(1,0), $\boldsymbol{\epsilon}_2 :=$ col(0,1), and rotation of $\mathcal{R}_i$ in the inertial $\mathcal{E}$-frame, ψ_i is

$$\psi_i = \arctan\left(\frac{\epsilon_2^\top (p_{i+1} - p_i)}{\epsilon_1^\top (p_{i+1} - p_i)}\right) \tag{2.1}$$

$\mathcal{R} \mapsto \mathcal{E}$ is mapped by the use of the rotation matrix

$$\boldsymbol{R}_2(\psi_i) = \begin{bmatrix} \cos(\psi_i) & -\sin(\psi_i) \\ \sin(\psi_i) & \cos(\psi_i) \end{bmatrix} \tag{2.2}$$

For a position vector $\boldsymbol{p} \in \mathcal{E}$, let the local representation of the point $\boldsymbol{p}$ be expressed as a vector in the reference frame $\mathcal{R}_i$, $\boldsymbol{\eta}(\boldsymbol{p}, i) := \boldsymbol{R}_2(\psi_i)^\top (\boldsymbol{p} - \boldsymbol{p}_i) \in \mathcal{R}_i$. Then the projection of $\boldsymbol{p}$ into a line-segment l_i is $\boldsymbol{\epsilon}_1^\top \boldsymbol{\eta}(\boldsymbol{p}, i)$ and the cross-track error is the Euclidean distance from $\boldsymbol{p}$ to l_i, $|\boldsymbol{\epsilon}_2^\top \boldsymbol{\eta}(\boldsymbol{p}, i)|$. The discretely parametrized path by index i, which identifies all points along line-segment l_i is then

$$\mathcal{P} := \{\boldsymbol{p} \in \mathbb{R}^2 : \exists i \in \mathcal{I} \; s.t. \; |\boldsymbol{\epsilon}_2^\top \boldsymbol{R}_2(\psi_i)^\top (\boldsymbol{p} - \boldsymbol{p}_i)| = 0\} \tag{2.3}$$

[1] Here n is total number of path segment

Continuous parametrization[2]

To continuously parametrize a path of n way-points, by s, each point along the path must be uniquely determined by a specific value of $s \in [0, n)$. The path is defined as

$$\boldsymbol{p}_d(s) := \begin{cases} \boldsymbol{p}_1 + s(\boldsymbol{p}_2 - \boldsymbol{p}_1), & s \in [0, 1) \\ \boldsymbol{p}_2 + (s-1)(\boldsymbol{p}_3 - \boldsymbol{p}_2), & s \in [1, 2) \\ \quad\vdots & \\ \boldsymbol{p}_i + (s-i+1)(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i), & s \in [i-1, i) \\ \quad\vdots & \\ \boldsymbol{p}_{n-1} + (s-n+2)(\boldsymbol{p}_n - \boldsymbol{p}_{n-1}), & s \in [n-1, n) \end{cases} \tag{2.4}$$

and the path is given by

$$\mathcal{P} := \{\boldsymbol{p} \in \mathbb{R}^2 : \exists\ s \in [0, n)\ s.t.\ \boldsymbol{p} = \boldsymbol{p}_d(s)\}. \tag{2.5}$$

Hybrid parametrization[3]

For $n+1$ way-points, the hybrid parametrization identifies each line-segment l_i by an index $i \in \mathcal{I} = \{1, 2, \ldots, n\}$. Each segment is continuously parametrized by s for values in a fixed interval, e.g. $[0, 1)$. A line-segment i is then described as $\boldsymbol{p}_d(i, s) := \boldsymbol{p}_i + s(\boldsymbol{p}_{i+1} - \boldsymbol{p}_i)$, so that $\boldsymbol{p}_d(i, 0) = \boldsymbol{p}_i$ and $\lim_{s \nearrow 1} \boldsymbol{p}_d(i, s) = \boldsymbol{p}_{i+1}$. Each point along the path, except $\boldsymbol{p}_{n+1}$, is uniquely determined by a par $i \in \mathcal{I}$ and $s \in [0, 1)$. The path becomes

$$\mathcal{P} := \{\boldsymbol{p} \in \mathbb{R}^2 : \exists\ i \in \mathcal{I}\ and\ s \in [0, 1)\ s.t.\ \boldsymbol{p} = \boldsymbol{p}_d(i, s)\}. \tag{2.6}$$

2.2.2 $\mathcal{C}^r$ path generation from way-points

In path generation the desired path $\boldsymbol{p}_d(s)$ is divided into n subpaths $\boldsymbol{p}_{d,i}(s)$, $i \in \mathcal{I} = \{1, 2, \ldots, n\}$ between the way-points. Each is expressed as a polynomial in s of a certain order. Then the expressions for the subpaths are concatenated at the way-points to assemble the full path. The order of polynomials must be sufficiently high to ensure that the overall path is sufficiently differentiable at the way-points.

To exemplify, consider $\mathbb{R}^2$. Let $\mathcal{I} = \{1, 2, \ldots, n\}$ be a set of indices identifying each subpath. The overall desired curve is denoted $\boldsymbol{p}_d = \text{col}(x_d(s), y_d(s))$, $s \in [0, n)$, while $\boldsymbol{p}_{d,i}(s) = col(x_{d,i}(s), y_{d,i}(s))$, $i \in \mathcal{I}$, are the subpath and $\boldsymbol{p}_i = col(x_i, y_i)$, $i \in \mathcal{I} \cup \{n+1\}$, are the way-points.

A common choice is to let s be an integer value at each way-point, staring with $s = 0$ at way-point $\boldsymbol{p}_1$, and $s = 1 - i$ at way-point $\boldsymbol{p}_i$. The differentiability requirement, $\boldsymbol{p}_d(s) \in \mathcal{C}^r$ means that the connection of two subpaths must have the same value,

[2]Here n is total number of way-points
[3]Here n is total number of path segment

$$\begin{aligned}
\lim_{s\nearrow i-1} x_{d,i-1}(s) &= \lim_{s\searrow i-1} x_{d,i}(s) & \lim_{s\nearrow i-1} y_{d,i-1}(s) &= \lim_{s\searrow i-1} y_{d,i}(s) \\
\lim_{s\nearrow i-1} x^s_{d,i-1}(s) &= \lim_{s\searrow i-1} x^s_{d,i}(s) & \lim_{s\nearrow i-1} y^s_{d,i-1}(s) &= \lim_{s\searrow i-1} y^s_{d,i}(s) \\
&\vdots & &\vdots \\
lim_{s\nearrow i-1} x^{s^r}_{d,i-1}(s) &= \lim_{s\searrow i-1} x^{s^r}_{d,i}(s) & \lim_{s\nearrow i-1} y^{s^r}_{d,i-1}(s) &= \lim_{s\searrow i-1} y^{s^r}_{d,i}(s)
\end{aligned} \tag{2.7}$$

for $i \in \mathcal{I}$. An approach to parametrize is to consider a polynomial of order k, that is

$$\begin{aligned}
x_{d,i}(s) &= a_{k,i}s^k + a_{k-1,i}s^{k-1} + \cdots + a_{1,i}s + a_{0,i} \\
y_{d,i}(s) &= b_{k,i}s^k + b_{k-1,i}s^{k-1} + \cdots + b_{1,i}s + b_{0,i}
\end{aligned} \tag{2.8}$$

where the coefficients $a_{j,i}, b_{j,i}$ must be determined. Each subpath have $(k+1)\cdot 2$ unknowns, yields $(k+1)\cdot 2n$ unknowns for the entire path. An obvious way is is to set up a large set of $(k+1)\cdot 2n$ linear equations, $\boldsymbol{A\phi} = \boldsymbol{b}$, for the path and solve them in a single operation as $\boldsymbol{\phi} = \boldsymbol{A}^{-1}\boldsymbol{b}$. However, numerical problem will arise with increasing n.

The ill-conditioning problem is circumvented by using a hybrid parametrization. Each subpath $\boldsymbol{p}_i$ identified by an index $i \in \mathcal{I}$ is continuously parametrized within a fixed interval $\varepsilon \in [0, 1)$. This corresponds to the hybrid parametrization described in Section 2.2.1.

$\mathcal{C}^0$: Continuity at the way-points for $i \in \mathcal{I}$

$$\begin{aligned}
x_{d,i}(0) &= x_i & y_{d,i}(0) &= y_i \\
x_{d,i}(1) &= x_{i+1} & y_{d,i}(1) &= y_{i+1}
\end{aligned} \tag{2.9}$$

$\mathcal{C}^1$: The slopes at first and last way-points are chosen as:

$$\begin{aligned}
x^s_{d,1}(0) &= x_2 - x_1 & y^s_{d,1}(0) &= y_2 - y_1 \\
x^s_{d,n}(1) &= x_{i+1} - x_n & y^s_{d,n}(1) &= y_{i+1} - y_n
\end{aligned} \tag{2.10}$$

$\mathcal{C}^1$:The slopes at the intermediate way-points

$$\left.\begin{aligned}
x^s_{d,i}(0) &= \lambda_{\mathcal{C}}(x_{i+1} - x_{i-1}) \\
y^s_{d,i}(0) &= \lambda_{\mathcal{C}}(y_{i+1} - y_{i-1})
\end{aligned}\right\} \quad i = 2, \ldots, n \tag{2.11}$$

$$\left.\begin{aligned}
x^s_{d,i}(1) &= \lambda_{\mathcal{C}}(x_{i+2} - x_i) \\
y^s_{d,i}(1) &= \lambda_{\mathcal{C}}(y_{i+2} - y_i)
\end{aligned}\right\} \quad i = 1, \ldots, n-1 \tag{2.12}$$

where $\lambda_{\mathcal{C}} > 0$ is a design constant.

$\mathcal{C}^j$: Derivatives of order j ≥ 2 for $i \in \mathcal{I}$

$$\begin{aligned}
x^{s^j}_{d,i}(0) &= 0 & y^{s^j}_{d,i}(0) &= 0 \\
x^{s^j}_{d,i}(1) &= 0 & y^{s^j}_{d,i}(1) &= 0
\end{aligned} \tag{2.13}$$

The result is a hybrid parametrization of the path

$$\hat{\boldsymbol{p}}_d(i,\varepsilon) = \begin{bmatrix} x_{d,i}(\varepsilon) \\ y_{d,i}(\varepsilon) \end{bmatrix} \tag{2.14}$$

where $i \in \mathcal{I}$ and $\varepsilon \in [0,1)$. This does not conform with the requirement of a continuous parametrization. However

$$\lim_{\varepsilon \nearrow 1} \hat{\boldsymbol{p}}_d^{\varepsilon^j}(i-1,\varepsilon) = \lim_{\varepsilon \searrow 0} \hat{\boldsymbol{p}}_d^{\varepsilon^j}(i,\varepsilon) \tag{2.15}$$

holds for $j = 0,1,2,\ldots,r$ and $i = 2,3,\ldots,n-1$ and therefore fulfils the differentiability requirement,. A map can be constructed so that $s \mapsto \boldsymbol{p}_d(s)$ is $\mathcal{C}^r$. For $s \in [0,n)$, let $i = \lfloor s \rfloor + 1 \in \mathcal{I}$ and $\epsilon = s - \lfloor s \rfloor \in [0,1)$ where $\lfloor \cdot \rfloor$ is a floor operation. Then the desired $\mathcal{C}^r$ map

$$s \mapsto \boldsymbol{p}_d(s) := \hat{\boldsymbol{p}}_d(i(s), \varepsilon(s)) \tag{2.16}$$

which is continuously parametrized by s, see Figure 2.1

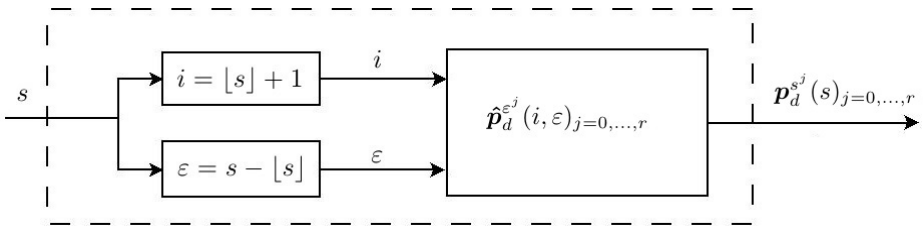


Figure 2.1: Block diagram showing the $\mathcal{C}^r$ map $s \mapsto \boldsymbol{p}_d(s)$. Courtesy:(Skjetne, 2005)

2.2.3 Dynamic assignments

According to Skjetne (2005) the maneuvering problem is divided into a *geometric task* and a *dynamic task*. The dynamic task satisfy one or more desired dynamic behaviors along the path and can be expressed as a time, speed, or acceleration assignment along the path. He defines those as:

1. A *time assignment* means to be at specific points along the path at specific time instants. For a continuous parametrized path $y_d(s)$, specific values of s_1, s_2, etc., must correspond to specific time instants t_1, t_2, etc. This can be done through a design function $v_t(\cdot)$ such that $s_1 = v_t(t_1)$ and $s_2 = v_t(t_2)$.

2. A *speed assignment* is to obtain a desired speed along the path for y. If $y_d(s)$ is a continuous parametrization this can be translated into a desired speed for $\dot{s}$. This desired speed may depend on the location along the path given by s (as for instance speed limits along a road), or it may explicitly depend on time. A natural choice is therefore to express the desired speed for $\dot{s}$ as a design function $v_s(s,t)$.

3. An *acceleration assignment* is to obtain a desired acceleration for y along the path. For a continuous parametrization $y_d(s)$ this can be expressed by a design function $v_a(\dot{s}, s, t)$ for $\ddot{s}$, which may depend on the speed $\dot{s}$ along the path in addition to s and t.

Furthermore, if the design functions v_t, v_s, or v_a express the desired dynamic behavior along the path when perfectly tracing it, then the time evolution of $s(t), \dot{s}(t)$, or $\ddot{s}(t)$ can additionally be shaped by the system states. This allows flexibility in the design that can be exploited to achieve favorable dynamic responses for a closed-loop system. In principle the functions could also depend explicitly on the system state x giving it more flexibility in the design.

One *level of flexibility* is added to the design when the system output y(t) converge to the reference system output $y_d(t)$ that next satisfy its desired objective in the limit. By increasing the *level of flexibility* by letting a system output y(t) converge to a reference system output $y_{d1}(t)$ which again converge to $y_{d2}(t)$ and so on, until the last reference system solves the maneuvering objective, the advantage of this may be that each level of flexibility will introduce an extra degree of freedom that in the design can be used constructively to solve specific problem at hand

2.2.4 The maneuvering problem statement

For a system output $\boldsymbol{y} \in \mathbb{R}^m$, the desired path is all points represented by the set:

$$\mathcal{P} := \{\boldsymbol{y} \in \mathbb{R}^m : \exists\ s \in \mathbb{R}\ s.t.\ \boldsymbol{y} = \boldsymbol{y}_d(s)\} \tag{2.17}$$

where $\boldsymbol{y}_d$ is continuously parametrized by s and both $\boldsymbol{y}_d(s)$ and $\mathcal{P}$ are called the desired path. Given the desired path and dynamic behavior, the maneuvering problem consist of the two tasks:

1. **Geometric Task:** For any continuous function $s(t)$, force the output $\boldsymbol{y}$ to converge to the desired path $\boldsymbol{y}_d(s)$

$$\lim_{t\to\infty} |\boldsymbol{y}(t) - \boldsymbol{y}_d(s(t))| = 0 \tag{2.18}$$

2. **Dynamic Assignment Task:** Satisfy one or more of the following assignments:

   2.1. **Time Assignment:** Force the path variable s to converge to a desired time signal, $v_t(t)$,

$$\lim_{t\to\infty} |s(t) - v_t(t)| = 0 \tag{2.19}$$

   2.2. **Speed Assignment:** Force the path speed $\dot{s}$ to converge to a desired speed $v_s(s, t)$,

$$\lim_{t\to\infty} |\dot{s}(t) - v_s(s(t), t)| = 0 \tag{2.20}$$

2.3. Acceleration Assignment: Force the path acceleration $\ddot{s}$ to converge to a desired acceleration $v_a(\dot{s}, s, t)$,

$$\lim_{t\to\infty} |\ddot{s}(t) - v_a(\dot{s}(t), s(t), t)| = 0 \tag{2.21}$$

Feasibility of geometric and dynamic task must be verified for the maneuvering problem to be properly solvable. Boundedness of the system states, such as restrictions in the control space may add limitations or constraints to the two tasks.

To validate the feasibility of the two task, let $\boldsymbol{y}_d(s) \in \mathcal{C}^r$ be the desired path in the geometric task. Differentiate $\boldsymbol{y}_d(s)$ r times with respect to time yields a continuous function

$$\boldsymbol{y}_d^{(r)} = \boldsymbol{\Phi}(s, \dot{s}, \ddot{s}, \ldots, s^{(r)}) \tag{2.22}$$

To ensure invariance of $\boldsymbol{y}_d(s)$ with respect to the system, there must then exist an admissible control input

$$\boldsymbol{u} = \boldsymbol{u}^* \tag{2.23}$$

such that the feasibility constraints differential equation,

$$\boldsymbol{\Phi}(s, \dot{s}, \ddot{s}, \ldots, s^{(r)}) - \boldsymbol{u}^* = 0 \tag{2.24}$$

has a solution s^*.

In other words the geometric task in the maneuvering problem is feasible for the system if $\boldsymbol{u} = \boldsymbol{u}^*$ exist. The dynamic task of the maneuvering problem is feasible for the system if there exist a $s^*(t) = v_t(t),\ \dot{s}^*(t) = v_s(s^*(t), t)$ or $\ddot{s}^* = v_a(\dot{s}^*(t), s^*(t), t)$

The *maneuvering control objective* is to construct a control law for the system that solves the maneuvering problem with repect to a feasible geometric task and feasible dynamic task, while keeping all system states bounded.

The *manuvering controller* is then a control law that solving the maneuvering control objective. In the special case when the dynamic task is a time assignment, that is satisfied identically, $s(t) = v_t(t),\ \forall t = 0$, the maneuvering controller becomes a *tracking controller*.

2.2.5 Gradient optimization

Gradient optimization ensures minimization and therefore improves performance. It rapidly positions to the most favourable point to converge by finding the fastest or steepest change.

A unfiltered gradient is faster than a filtered gradient, but it is more sensitive to measurement noise. While the filtered gradient smooths out noise the cost is a delay and the cut-off frequency is an important design parameter to mitigate state measurement noise versus bandwidth.

Using gradient optimization within the control structure, a separation of time scales can be induced between the task of selecting a point of the path closes to the state and the

task of driving the states toward the path. In addition to uniform global convergence the the path, achieved without separation of time scale, the separation of time scales allows achievement of *near foward incariance* of a path from a large range of initial conditions.

2.3 LOS-based maneuvering formation control

The control design in Thorvaldsen (2011) follows the generic maneuvering methodology to solve the formation control problem using the Line-of-Sight (LOS) algorithm presented in Skjetne et al. (2011). The incorporated LOS steering algorithm ensures feasible and predictable motion toward the path for formation.

The control law includes activation fucntions σ_k to enable prioritization and separation between task of establishing formation, and the task of getting the formate to converge to the specified path. This effectively divide an the task into two distinct phases by tuning the system. The designed formation control establish formation in their immediate vicinity prior to initiation of a collective moment to converge and follow a path. The intuitive notion that that the separation and prioritization of group coordinate and path-following increase safety. However a guarantee against collisions is not given, as the design do not explicitly incorporate collision avoidance capabilities. The design shows to require considerable inter-vessel communication during operations in general which could limit their applicability for large groups.

2.3.1 The LOS algorithm

Consider a vessel with generalized position vector $\boldsymbol{\eta} = col(\boldsymbol{p}, \psi)$, $\boldsymbol{p} = col(x, y)$, in the $\mathcal{E}$-frame, with dynamics given by

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}(\psi)\boldsymbol{\nu}, \tag{2.25}$$

where $\boldsymbol{\nu} = col(u, v, r)$. For a parametrized path given by the set

$$\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^2 : \exists s \in \mathbb{R} \; s.t. \; \boldsymbol{x} = \boldsymbol{p}_d(s)\}, \tag{2.26}$$

where $\boldsymbol{p}_d(s) := col(x_d(s), y_d(s))$ is a sufficiently smooth function, the LOS algorithm directs the course angle of the vessel in such a way that it guarantees convergence to the path $\mathcal{P}$ provided perfect tracking and non-zero lower bound on the magnitude of motion.

For a vessel not experiencing any sideslip (i.e. zero deviation between the course and heading angles), this translates to a reference signal ψ_{los} for the heading angle that accomplishes path following though positive surge motion. As the LOS algorithm actually will be used to direct the motion of an formation reference $\mathcal{F}$-frame in this design, sideslip is of no relevance in the following.

The orientation of a path-tangential reference frame with origo located at $\boldsymbol{p}_d(s)$ is given by

$$\psi_d(s) := \arctan\left(\frac{y_d^s(s)}{x_d^s(s)}\right) \tag{2.27}$$

The position of the vessel relative to the origin of the frame can be expressed in path-tangential coordinates according to

$$\boldsymbol{\epsilon}(\boldsymbol{p}, s) = col(\epsilon_1(\boldsymbol{p}, s), \epsilon_2(\boldsymbol{p}, s)) = \boldsymbol{R}_2(\psi_d(s))^\top(\boldsymbol{p} - \boldsymbol{p}_d(s)), \tag{2.28}$$

where $\epsilon_1(\boldsymbol{p}, s)$ and $\epsilon_2(\boldsymbol{p}, s)$ are termed the along-track and cross-track errors respectively, and

$$\boldsymbol{R}_2(\psi_d) = \begin{bmatrix} \cos(\psi_d(s)) & -\sin(\psi_d(s)) \\ \sin(\psi_d(s)) & \cos(\psi_d(s)) \end{bmatrix}. \tag{2.29}$$

Let the surge speed of the vessel be given by $u = U_d(t) > 0$ and define

$$V(\boldsymbol{p}, s) = \frac{1}{2}\boldsymbol{\epsilon}(\boldsymbol{p}, s)^\top\boldsymbol{\epsilon}(\boldsymbol{p}, s) \tag{2.30}$$

In accordance with Skjetne et al. (2011), the LOS algorithm can be summarized as

$$\psi_{los}(\boldsymbol{p}, s) = \psi_d + arctan\left(\frac{-\epsilon_2(\boldsymbol{p}, s)}{\Delta}\right), \tag{2.31}$$

$$\dot{s} = f_s^*(\boldsymbol{p}, s, t), \tag{2.32}$$

$$f_s^*(\boldsymbol{p}, s, t) = \frac{\Delta}{\sqrt{\Delta^2 + \epsilon_2(\boldsymbol{p}, s)^2}}\frac{U_d(t)}{|\boldsymbol{p}_d^s(s)|} - \omega_s(\boldsymbol{p}, s), \tag{2.33}$$

$$\begin{aligned} \omega_s(\boldsymbol{p}, s) &= \frac{\mu_s}{|\boldsymbol{p}_d^s(s)|}V^s(\boldsymbol{p}, s) \\ &= -\mu_s\frac{\boldsymbol{p}_d^s(s)^\top}{|\boldsymbol{p}_d^s(s)|}(\boldsymbol{p} - \boldsymbol{p}_d(s)) \\ &= -\mu_s\epsilon_1(\boldsymbol{p}, s), \end{aligned} \tag{2.34}$$

where $\Delta, \mu_s > 0$. The algorithm points the vessel toward a point located a distance Δ along the x-axis of the path-tangential $\mathcal{R}$-frame. Furthermore the dynamic update law for s incorporates gradient optimization in addition to a feedback term from the surge speed of the vessel.

The algorithm solves a maneuvering problem given by the geometric and dynamic task

$$\lim_{t\to\inf}|\boldsymbol{\eta}(t) - \boldsymbol{\eta}_d(t)| = 0 \tag{2.35}$$

$$\lim_{t\to\inf}|\dot{s}(t) - v_s(s(t), t| = 0 \tag{2.36}$$

2.3.2 Problem statement

The position and orientation of the $\mathcal{F}$-frame is explicitly stated in this design and allowed to evolve over the entire state space. The $\mathcal{F}$-frame is defined as

$$\boldsymbol{\zeta} := col(\boldsymbol{p}_\zeta, \zeta_3) \in \mathbb{R}^3 \tag{2.37}$$

where

$$\boldsymbol{p}_\zeta = col(x_\zeta, y_\zeta) \in \mathbb{R}^2 \tag{2.38}$$

contains the position and orientation of the $\mathcal{F}$-frame , and ζ_3 contains its orientation relative to the $\mathcal{E}$-frame.

Consider r vessels with the dynamics given by (2.25) that is to be controlled in formation, and assign to each of them a unique index $i \in \mathcal{I} = \{1, \ldots, r\}$. The desired formation is specified as the set of configuration vectors

$$\boldsymbol{l}_i = col(x_{ci}(t), y_{ci}(t), \psi_{ci}(t)), \ i \in \mathcal{I} \tag{2.39}$$

that is to be attained relative to the $\mathcal{F}$-frame. This yields a desired

generalized position vector for each vessel according to

$$\boldsymbol{\eta}_{di} = (\boldsymbol{\zeta} + \boldsymbol{R}(\zeta_3)\boldsymbol{l}_i, \ i \in \mathcal{I} \tag{2.40}$$

where

$$\boldsymbol{R}(\cdot) = \begin{bmatrix} \cos(\cdot) & -\sin(\cdot) & 0 \\ \sin(\cdot) & \cos(\cdot) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.41}$$

is the rotation matrix in yaw. When the vessels have established the formation by reaching their desired generalized positions , then the formation will utilize the LOS algorithm to pursue path-following. Using the maneuvering methodology this is achieved by letting the algorithm dictate the trajectory of $\boldsymbol{\zeta}$ when a certain manifold is reached. by defining

$$\bar{\boldsymbol{\eta}} := col(\boldsymbol{\eta}_1, \ldots, \boldsymbol{\eta}_r) \in \mathbb{R}^{3r}, \tag{2.42}$$

$$\bar{\boldsymbol{\nu}} := col(\boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_r) \in \mathbb{R}^{3r}, \tag{2.43}$$

$$\bar{\boldsymbol{\eta}}_d(\boldsymbol{\zeta}, t) := col(\boldsymbol{\eta}_{d1}(\boldsymbol{\zeta}, t), \ldots, \boldsymbol{\eta}_{dr}(\boldsymbol{\zeta}, t)) \in \mathbb{R}^{3r}, \tag{2.44}$$

a natural choice for the manifold is

$$\mathcal{A} = \{(\bar{\boldsymbol{\eta}}, \bar{\boldsymbol{\nu}}, \boldsymbol{\zeta}, s, t) : \bar{\boldsymbol{\eta}} = \bar{\boldsymbol{\eta}}_d(\boldsymbol{\zeta}, t), \zeta_3 = \psi_{los}(\boldsymbol{p}_\zeta, s)\}, \tag{2.45}$$

where t is included in the system state vector by adhering to

$$\dot{t} = 1 \qquad t(0) = t_0 \tag{2.46}$$

The explicit time is separated from the internal time-variable in the system, and denoted as t^* so that $t(t^*) = t^* + t_0,\ \forall t^* \geq 0$.

The desired dynamics for the $\mathcal{F}$-frame on the manifold $\mathcal{A}$ is given by the motion constraints. By defining

$$\boldsymbol{f}_p(\zeta_3, t) = \begin{bmatrix} cos(\zeta_3) \\ sin(\zeta_3) \end{bmatrix} U_d(t) \tag{2.47}$$

where $U_d(t)$ is a desired surge speed for the $\mathcal{F}$-frame. The control objectives expressed as the following geometric task:

$$\lim_{t^* \to \inf} |\bar{\boldsymbol{\eta}}(t^*) - \bar{\boldsymbol{\eta}}_d(\boldsymbol{\zeta}(t^*), t(t^*))| = 0 \tag{2.48}$$

$$\lim_{t^* \to \inf} |\zeta_3(t^*) - \psi_{los}(\boldsymbol{o}(t^*), s(t^*))| = 0 \tag{2.49}$$

and dynamic task:

$$\lim_{t^* \to \inf} |\dot{\boldsymbol{p}}_\zeta(t^*) - \boldsymbol{f}_p(\zeta_3(t^*, t(t^*))| = 0 \tag{2.50}$$

(2.48) is given a higher priority than the others due to separation of group coordination and path-following pursuit.

2.3.3 Control design

A backstepping procedure is used for the control design. Using maneuvering design the dynamics of $\boldsymbol{\zeta}$ and s would be chosen as

$$\dot{\boldsymbol{\zeta}} = \boldsymbol{f}^*_\zeta(\boldsymbol{\zeta}, s, t) - \boldsymbol{\omega}_\zeta \tag{2.51}$$

$$\dot{s} = f^*_s(\boldsymbol{p}_\zeta, s, t) - \omega_s \tag{2.52}$$

where $\boldsymbol{\omega}_\zeta$ and ω_s provides the Lyapunov function used the design, and $\boldsymbol{f}^*_\zeta(\boldsymbol{\zeta}, s, t)$ and $f^*_s(\boldsymbol{p}_\zeta, s, t)$ are nominal terms that would be chosen in accordance with the desired dynamics on the specified manifold. In this case $f^*_s(\boldsymbol{p}_\zeta, s, t)$ would be (2.33),

$$\boldsymbol{f}^*_\zeta(\boldsymbol{\zeta}, s, t) = \begin{bmatrix} \boldsymbol{f}_p(\zeta_3, t) \\ f_\zeta 3^*(\boldsymbol{\zeta}, s, t) \end{bmatrix}, \tag{2.53}$$

Chapter 3

Infrastructure around CyberShip Enterprise 1

This chapter discuss the infrastructure around CyberShip Enterprise 1 (CSE1). It is a part of the Marine Cybernetics Laboratory (MCLab) at the Norwegian University of Science and Technology (NTNU). For detailed information on Voith schneider propeller, MCLab and Qualisys see Skåtun (2011).

In Skåtun (2011) a hardware/software architecture was created with the use of LabVIEW and Simulink. Three control modes were implemented in the architecture manual thruster control, joystick control, and dynamic positioning (DP) control. He gave detailed documentation on the hardware, but spoke little of the software. The aim in this chapter is to fill out the blanks and state the information gathered so far.

3.1 Overview

Figure 3.1: Anchor Handling Tug AZIZ. Courtesy: Model Slipway (n.d.)

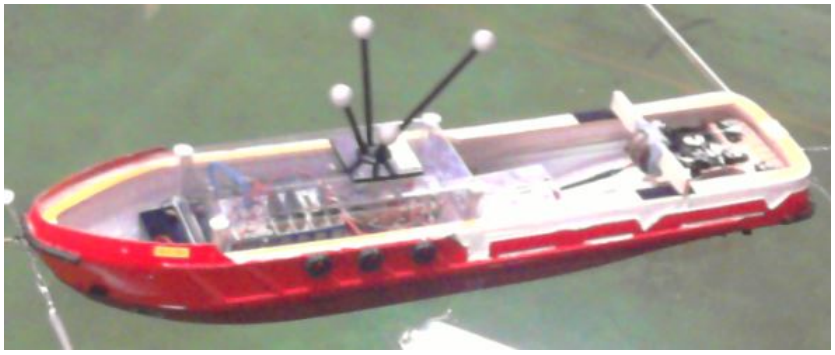


Figure 3.2: CyberShip Enterprise 1

Length	1.1 meter
Width	0.23 meter
Draft	0.09 meter
Freeboard	0.04 meter

Table 3.1: Main dimensions of CyberShip Enterprisee 1

CSE1 was built from the 1:50 scaled model boat, see Figure 3.1 and 3.2. Little remained of the original except for the hull and a few rubber accessories. The bow thruster was upgraded for more thrust power. The rudders and propellers were replaced with two voith schneider propellers for better thrust direction versatility. Plastic mouldings were added to increase the freeboard. When fully equipped it weigh 17 kg.

The Compact RIO and the Electronic Speed Controls (ESCs) for the VSPs are powered by the main 12V battery. The WiFi adapter and bow thruster are powered by the auxiliary 6V battery. The reason for this is that the Compact RIO and the ESCs needs at least 9V to operate, and the ESC for the bow thruster and the WiFi adapter are rated for 5V. Further on the Compact RIO generates Pulse Width Modulation (PWM) control signals to all the servos and ESCs (Skåtun, 2011), see Figure 3.3.

Table 3.2: CSE1 equipment list

CSE1 equipment	
Bow thruster	1
Voith schneider propeller	2
Electronic speed control	3
Servo	4
6 Volt Auxiliary battery	1
12 Volt Main battery	1
CompactRIO	1
WiFi adapter	1
Waterproof casing	1
Infra-red marker	4
Ballast weight	1

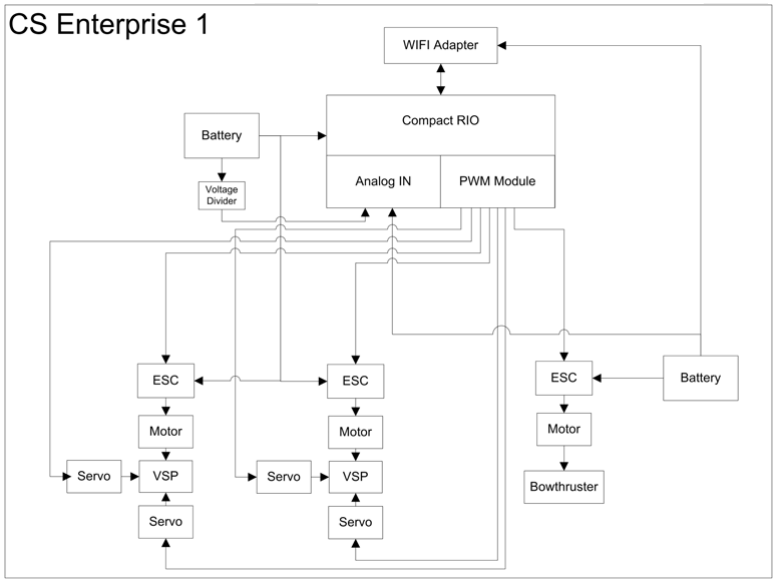


Figure 3.3: CyberShip Enterprisee 1 wiring topology. Courtesy:(Skåtun, 2011)

3.2 Hardware

In addition to the on-board equipment the infrastructure around CSE1 are one Windows XP laptop to run the programs, two PlayStation 3 (PS3) controllers and bluetooth dongle for manual and joystick control, three infrared (IR) cameras to track the IR markers on CSE1, One stationary Windows XP computer for qualisys track manager (QTM) and one extra 6V and 12V battery with chargers.

3.3 Signal flow

The signal flows are well described in Skåtun's thesis and he provides schematics on the hardware/software interaction, see Figure 3.4. The aim of this section is to go deeper into the details on the signal flow.

Playstation 3 controller

The operator interacts either directly with the laptop or through a PS3 controller. The controller communicate with the laptop through a bluetooth dongle. For the laptop to be able to understand the signals, two software are needed. BtSix is the communication port for the bluetooth controller. PP Joy recognise the input signals and make it available for the laptop. Further it can also be used to configure and tune the PS3 controller buttons.

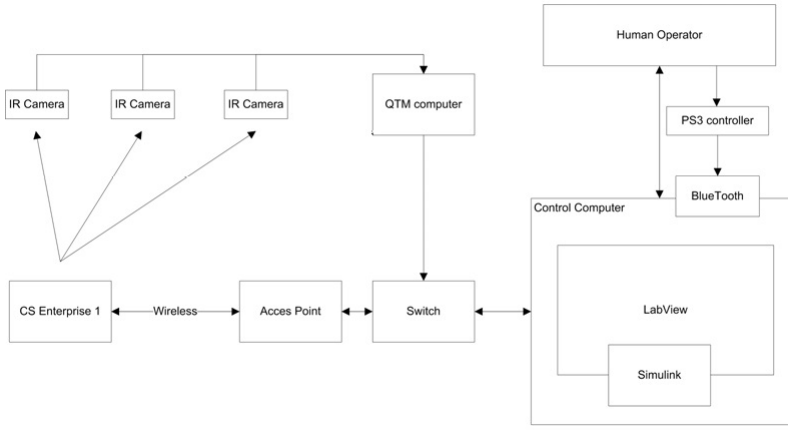


Figure 3.4: Hardware and software topology in the MCLab realated to CSE1. Courtesy:(Skåtun, 2011)

Simulink

The laptop runs in addition to the regular program, BtSix, PPJoy, LabVIEW, and dependent on the user, Matlab/Simulink. The model created in Simulink is implicitly used, since the RealTime Workshop[1] converts the Simulink model to a C/C++ dll-fil to make it usable for LabVIEW. Therefore Simulink will only communicate with LabVIEW.

LabVIEW

LabView is the Human-Machine-Interface (HMI) and where the deployment of CSE1 are done. It loads in the dll-file of the Simulink model, receives and process signals from the PS3 controller and Qualisys before "sending" it to Simulink. As well as interacting with the hardwares through SIT connection manager. It communicates with CSE1 wireless through the Ethernet, HIL-lab, to the IP adress 192.168.0.77.

SIT connection manger

SIT connection manager is the intersection for the entire architecture, except for the Qualisys, which is handled by a "background" vi-file, Base_Rate_Loop.vi. It is here almost all the mapping are done, LabView/Simulink communications and hardware/software e.g. Pulse Width Modulation control signals for the propellers.

Base_Rate_Loop.vi

The Base_Rate_Loop.vi receives the data from QTM, through QTMdriver.vi developed by Torgeir Wahl. It was in the QTMdriver.vi the source of Skåtun's problems laid.

[1] In newer Matlab versions RealTime Workshop is called Simulink Coder

The entire function of Base_Rate_Loop.vi is to initiate, read and close the QTMdriver.vi and it can be the cause of crashes. When it do not receive data in high enough frequency, it will just stop causing the entire run to halt. Figure 3.5 shows the correct set-up of Base_Rate_Loop.vi, if QTM is not sending data then it should be set to N/O instead of Init, Read and Close.

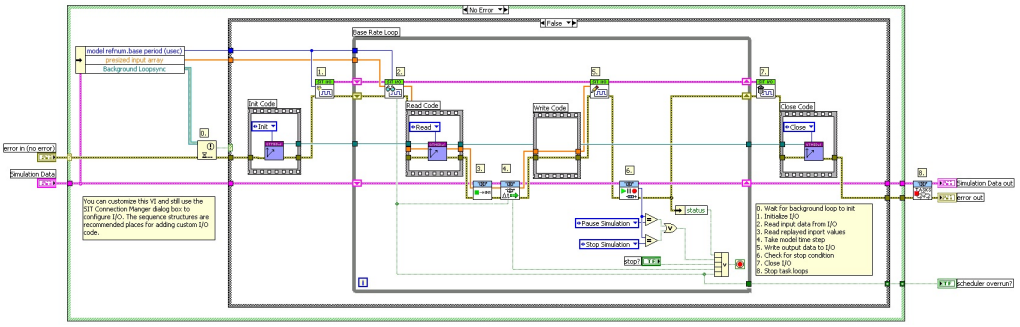


Figure 3.5: Base_Rate_Loop.vi for online QTM

QTMdriver.vi

The QTMdriver.vi is needed to process the data sent from QTM. It ask the QTM server for data and determines what data to pass along from QTM to LabVIEW.

There was a bug in the QTMdriver.vi that was used by Skåtun. It was defined to send out eight parameters when it was suppose to be nine parameters and in addition *DataOutIndex* began on one when it should have been zero. The effects of this were, it did send nine values despite the definition, but the values became scrambled in set of three. This is the reason Skåtun had extra switches to work around the problem.

QTM

The QTM uses three IR cameras to track the four IR markers on CSE1. It to calculate the six degrees of freedom (DOF) and passes it on in real-time. In addition it also controls several of the camera parameters.

3.4 Software

The software used when working with CSE1 are, LabVIEW, SIT connection manager, Matlab/Simulink/Real-Time Workshop, PPJoy, BtSix, and QTM. On a side note, is useful to have the signal and processing toolboxes that comes with Matlab and LabVIEW and in addition the MSS GNC toolbox.

Due to the sparse information in Skåtun (2011) much time were spent on understanding the logic behind the LabVIEW file CSEI.vi and Simulink model CSE1.mdl. Most of the time was used to understand random placement of object, thus began the reverse

engineering. This process can be summarized in three phases, Spartan-, Duplicate- and Expansion-phase, and produced three sets of "stable"[2] software for CSE1.

The one that is most reliable is the vi-file created under the Spartan phase, it will always run and does not crash. The vi-file in the Duplicate phase is free of delay and shuffling of parameters that plagued Skåtun's vi-file, but due to some unknown parameters, that is yet to be identified, it will sometime crash or simply refuse to run. The quick fix solution to this require a full system reboot and the problem simply vanishes. These unknowns are carries over to the vi-file in the Expansion phase, but it is worth noting that this events do not often occur.

3.5 Notes and miscellaneous

This section contains notes related settings, procedure and some troubleshooter for successfully deploy CSE1. To keep it a general the Simulink model name is referred to as *slmodel* and the labview file name as *lvfile*.

The Simulink model and LabVIW-vi can be created in matlab 2009b and LabVIEW 2010 or newer version just remember to save them as a 2009b mdl-file and 2010 vi-file.

3.5.1 QTM

QTM sends out nine parameters;

1. Frame number
2. Average error
3. x-position
4. y-position
5. z-position
6. yaw-rotation
7. pitch-rotation
8. roll-rotation
9. Residual

in that order. The positions are given in millimetres while the rotations are in degrees. The axis follows the right hand rule with positive z-axis points downward. Average error as the name implies is the average error of the four IR-markers. The measurement from QTM are not valid ff the residual number becomes -1.

[2] Able to run, but incomplete product

3.5.2 Simulink and qualisys

The QTM should have the same real-time sending frequency as the sampling frequency of the Simulink model. When the QTM frequency was 24 Hz and Simulink model had a sample time of 0.1 the lag was two minutes from the get-go. The reason for this might be a backlog of data from QS that Simulink must process chronologically. If the QS subsystem add an enabler then the lag increased to eight minutes.

By setting the real-time frequency of QTM to 10 Hz and Simulink sampling to 0.1 s the delay disappears. This implies that a QTM frequency higher than simulinks sampling time create delay. On a sidenote, if the QTM frequency is lower than simulinks sampling time then the Base_Rate_Loop.vi would crash.

3.5.3 Simulink model to dll-file

The solver option should be fixed step size, using ODE as solver method, do not use block reduction optimization and use Realtime Workshop to compile the nidll- and nidll_vxworks-file. Two folders should have been created *slmodel_nidll_rtw* and *slmodel_vxworks_rtw* in the current matlab folder.

After compiling the slmodel.mdl to a dll-files, go into the folder named *slmodel_nidll_rtw*. Find *slmodel_portsRead.txt* and confirm that the first input port is for Qualisys (width = 9). If not move SIT input block to make it so.

3.5.4 Connecting to CSE1

Most of the time connecting to CSE1 do not require much thought, but there are times when CSE1 will not respond to when pinging it. This led to an ad-hoc procedure that guarantees connection to CSE1.

1. Start with unplugging both batteries and the network cable from the ATC/LINK port.
2. Connect 12V battery.
3. Connect 6V battery.
4. Wait 5-10 seconds.
5. Connecting D-link network cable to the ATC/Link port.
6. See green light in the ATC/Link port, preferably blinking
7. Ping CSE1 in cmd.exe using the command: *ping 192.168.0.77*
8. If it do not respond wait 15-30 seconds before pinging again.
9. If second try failed. If wireless connection check that you are connected to the Ethernet *HIL-lab*, if direct cable from laptop to CSE1 check that it is properly connected.

Table 3.3: Hardware I/O mapping

PWM output to Model output	
PWM out 0 (DO 0)	Bow thruster
PWM out 1 (DO 1)	VSP speed port
PWM out 2 (DO 2)	VSP speed starboard
PWM out 3 (DO 3)	unplugged
PWM out 4 (DO 4)	Servo 1
PWM out 5 (DO 5)	Servo 2
PWM out 6 (DO 6)	Servo 3
PWM out 7 (DO 7)	Servo 4

Analog input to Model input	
AI 0	Bow thruster battery voltage
AI 1	Servo battery voltage
AI 2	Main battery voltage

3.5.5 Hardware I/O

An hardware I/O mapping can be found in the folder *mapping* and it is worth checking it if it have been a long time since running the LabVIEW file.

The device should be *RIO0 - cRIO-9113* and the *PWM out* channels with *Type* Frequency Hz should be 50.

3.5.6 Deploying CSE1

Other problems can occure even with correct setup of LabVIEW and Simulink and connection with CSE1. When deploying it will always ask for saving changes in case something goes wrong and it often does.

The first problem encountered is deployment time $t \to \inf$, or the feeling of it. The solution is simple force shut down LabVIEW and redeploy. This time the deployment time should take significantly less time, sometimes skipping it since it was deployed the first time, but did not respond.

Another problem is *screen freeze* after successful deployment. There can be several causes, but the most common causes are flat battery or loss of wireless connection. If it is flat batteries change and charge. When the wireless connection is lost two thing can be done. Ping it, wait for it to respond and reconnect by itself. Everything will work normally or force shut down of LabVIEW, rerun it and redeploy. Sometimes there are no need to redeploy since it only lost connection, but it can be stuck in the middle of a command requiring to override the previous one. If it ask for overriding press *OK* and move on.

3.5.7 Naming

The name of the files generated were named using abbreviation and shorthand form to make it into a compact information bundle.

Table 3.4: List of file name abbreviation

Shorthand	Extended
BP MC1	BluePrint ManualControl1
CA CSE1 V1e	ControlArchitecture CyberShipEnteprise1 Version1e
wo LOS2b	without LineOfSight2b
HMI CSE1	HumanMachineInterface CyberShipEnterprise1

Chapter 4

Software development

The software created by Skåtun is ill suited for anyone but him to use, but provides a good start point for a modularized control architecture. There was little information of the software development in Skåtun (2011) as mentioned previously, but having a finished product to reverse engineer is easier than to start from scratch. A secondary objective to do this work was to remove a delay that built up over time when using Skåtun's LabVIRW program.

This chapter will systematically go through the work process of software development. The control architecture software development begins by first explaining Skåtun's software then describe the step by step development through three phases. Spartan phase; keeping it to the bare minimum, Duplicate phase; implement all functions in Skåtun's work and Expansion phase; adding and improving functions. The figures in this chapter are to provide the reader illustrations of the development and not meant to be readable. For a better view see the corresponding mdl- and vi- files found in the digital folders attached.

4.1 LabVIEW

4.1.1 Front panel

CSEI.vi

Skåtun's *front panel* was brilliantly designed for the user. Three tabs, *Overview*, *3D Vis* and *Regulation Error* with a header for enabling various modes. See Figure 4.1, 4.2 and 4.3.

The *Overview* tab provides information about the PS3 controller, position of the analogue sticks, battery voltage, power and speed settings, direction, pitch and force the thrusters.

The *3D Vis* tab gives the user a virtual 3D window, a more intuitively representation of the systems behavior compared to using 2D graphs. It is here the input for the DP controller is placed, as well as the systems output.

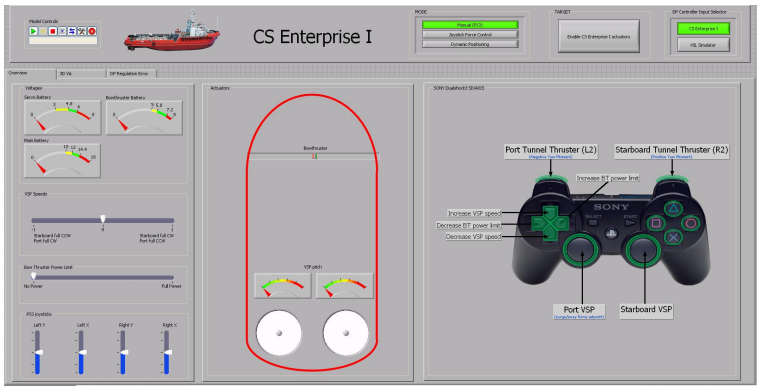


Figure 4.1: LabVIEW, front panel Overview tab of CSEI.vi Courtesy: Skåtun (2011)

Figure 4.2: LabVIEW, front panel 3D Vis tab of CSEI.vi Courtesy: Skåtun (2011)

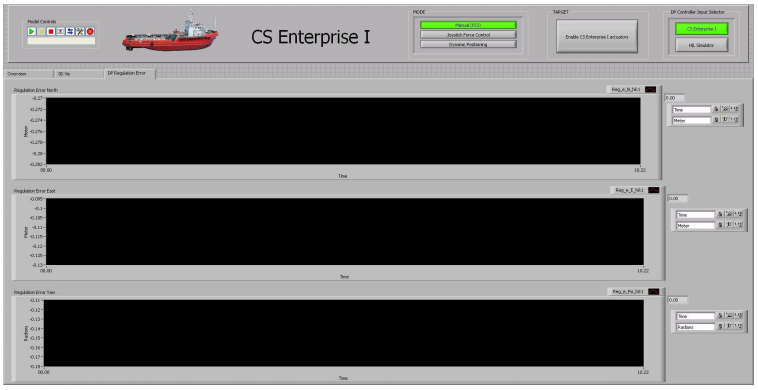


Figure 4.3: LabVIEW, front panel Regulation Error tab of CSEI.vi Courtesy: Skåtun (2011)

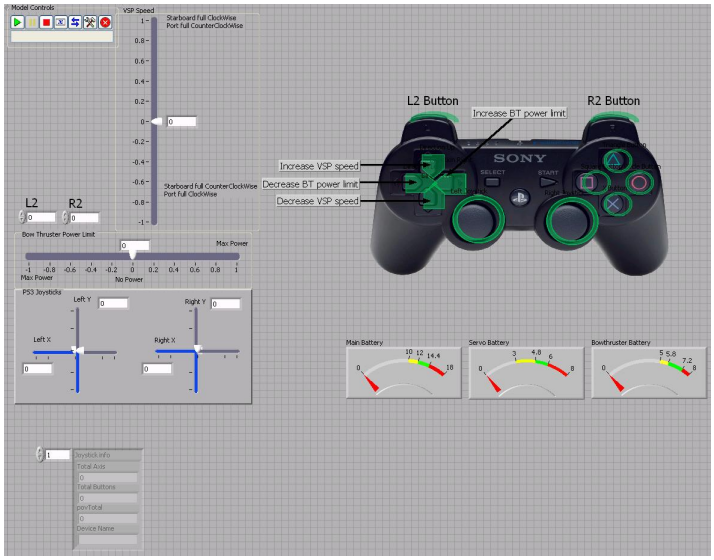


Figure 4.4: LabVIEW, front panel Spartan phase

The *Regulation Error* tab presents three graphs showing the development of the regulation error for north, east and yaw.

Spartan phase

The mantra in this phase was: *what do i need in an emergency?* All that is needed are the PS3 controller and some indicators for to steer CSE1. This meant reducing the Skåtun's front panel to the bare minimum meant sparing only parts the *Overview* tab. The only difference between this and Skåtun's layout, except the obvious, was converting the analogue indicators to a more intuitive arrangement of two crosses instead of four vertical bars, see Figure 4.4. And making the VSP speed limiter indicator vertical to reflect the up/down direction buttons.

Duplicate phase

In the *Duplicate* phase, the front panel from the previous phase is put in the frame from CSEI.vi. At this point the need the need of showing the pitch, direction and force of the thrusters was not on the radar. Therefore the *Overview* tab contained only the compact set from the Spartan phase, see Figure 4.5.

The *3D Display* tab remains mostly the same as CSEI.vi's, but with just a few tweaks in the layout, see Figure 4.6. The 3D window is more quadratic and two tabs are introduced to accommodate other controllers and system outputs. The DP input remain the same. The *Regulation Error* tab was not implemented here as this phase was a intermediate phase focusing more on the 3D display.

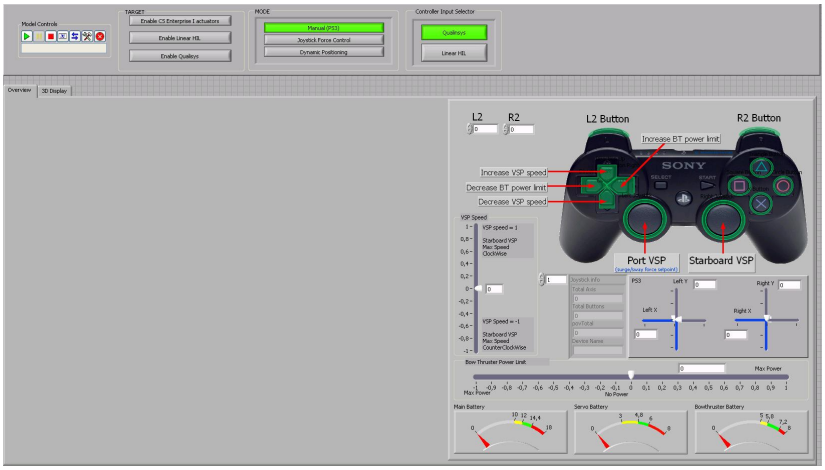


Figure 4.5: LabVIEW, front panel Overview tab Duplicate phase

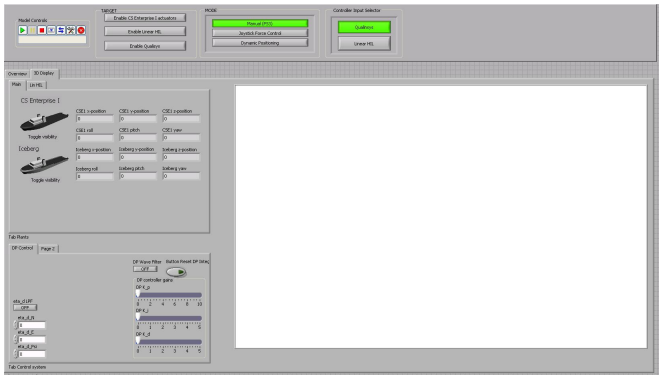


Figure 4.6: LabVIEW, front panel 3D Display tab Duplicate phase

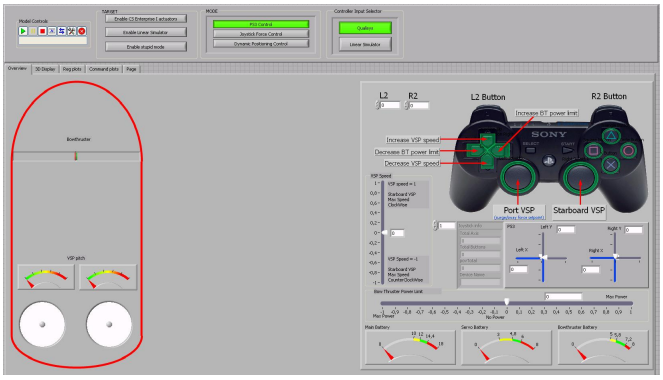


Figure 4.7: LabVIEW, front panel Overview tab Expansion phase

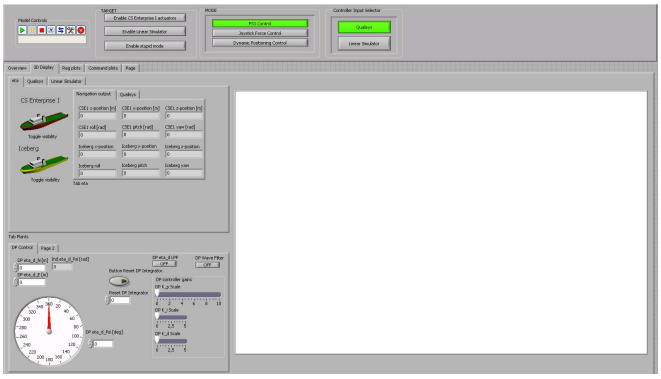


Figure 4.8: LabVIEW, front panel 3D display tab Expansion phase

Expansion phase

In the expansion phase the *Overview* tab contains all that was in CSEI.vi's with a different layout. There were plans to adding graphs displaying the $u_1, \ldots, u_5$ signals to the thrusters in the middle, but at that point seems redundant when having an entire tab for graphs, see Figure 4.7.

The *3D Display* tab have undergone small adjustments but the overall layout remains the same as in the duplicate phase. The system output tab was modified to better sort what was shown of inputs and outputs of the system. In the DP input tab, the heading input method was change from radians to degrees and a dial to make it more intuitive. In addition the tuning parameters slides were scaled so the numerical value matches the graphical representation, see Figure 4.8.

In this phase two tab were added compared with the *Duplicate* phase, *Reg plot* tab and *Command plots* tab. The *Reg plot* tab mimics the *regulation error* tab of CSEI.vi. It also displays the response and accommodates expansion for other controllers, see Figure 4.9.

The *Command plots* tab have a similar layout as the *Reg plot* tab, but instead of response and error it plots the control law output and corresponding $u_1, \ldots, u_5$ inputs to the thrusters, see Figure 4.10.

Figure 4.9: LabVIEW, front panel Reg plots tab Expansion phase

Figure 4.10: LabVIEW, front panel Command plots tab Expansion phase

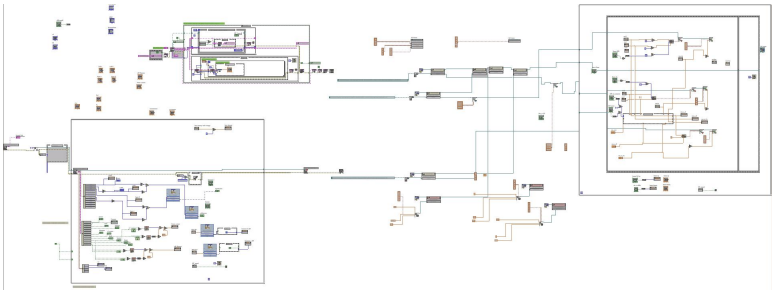


Figure 4.11: LabVIEW, block diagram of CSEI.vi Courtesy: Skåtun (2011)

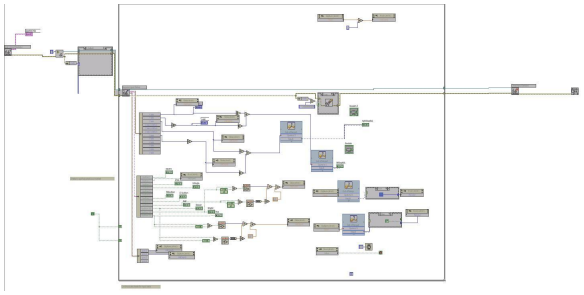


Figure 4.12: LabVIEW, block diagram *PS3 module* of CSEI.vi Courtesy: Skåtun (2011)

4.1.2 Block diagram

CSEI.vi

The block diagram in CSEI.vi largely consist of three main modules, *SIT*, *PS3 input* and *3D picture*. The thing that makes this it difficult to work with is the random placement of blocks and overcomplicated ways the structure and wiring were done. The block diagram is not as "smooth" to work with compared to the front panel. Seeing it for the first time can be daunting especially if you have never worked with LabVIEW before, see Figure 4.11.

The first discovery made, *do not touch the SIT block* and it is by far the most complicated set to work with. Luckly the *SIT* module is automatically generated by the SIT connection manager and is left as is. It is the most structured and compact of the three seen in Figure 4.11 at the upper left part.

The second discovery was, there were a lot of excessive component, cluttered wire paths, unnecessary structures and random placed blocks. For instance in Figure 4.12 the small group of blocks in the upper right corner of the while structure had no purpose what so ever. An example of the excessive structure can be seen in Figure 4.13, the case structure within the sequence structure.

The *PS3 module* scanned for devices connected to the computer. Each device is assigned an integer uniquely identifying them and it have to be manually inputted into the blue framed block seen in Figure 4.12 at the upper left part. The device integer is found by running the vi-file and scroll through the *Joystick info* indicator in the front panel, seen

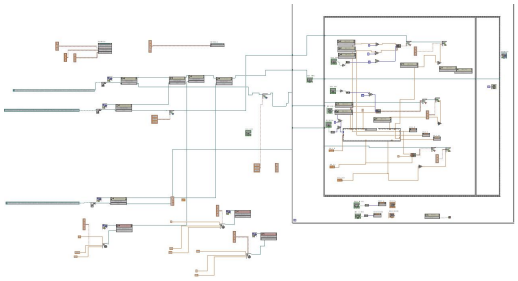


Figure 4.13: LabVIEW, block diagram *3D picture module* of CSEI.vi Courtesy: Skåtun (2011)

in Figure 4.4 at the bottom left and in Figure 4.5 and 4.7 in the middle of the right half, to the left of the analogue axis.

The signals are split into buttons and axis within the while-loop, that run as long as the simulation runs. All the buttons and analogue stick are then wired to a indicator to tell if they are responsive. To prevent indicator chattering of the analogue sticks a threshold is compared with the sum of x- and y-position of the sticks.

The direction pads are divided into up/down and right/left. The up/down pads are assigned to the VSP speed limiter by summing up the number of times up is pressed subtracting the number of times down is pressed and then multiplied with a step length for the incremental increase. Similarly with the right/left button assigned for the bow thruster power limiter.

The *3D picture module* work in principle by adding a parental object and then attach other objects to it as children. This principle was used in CSEI.vi, but heavily distorted by the poor arrangement of wires and blocks. It retrieves position data from the Simulink model via the *SIT module* with property nodes and sends them to the respective objects, see Figure 4.13.

CSEI.vi deviates from the normal convention with the y-axis points upwards, x-axis forward and z-axis starboard. The reason for this is the boat model used, was created with the y-axis pointing upwards and therefore the whole 3D window orientation become as it is. This requires keeping track of the 3D vector sequence, the camera setting vector becomes col(1,0,0) instead of col (0,0,1) and the water surface dimension vector is col(20,0.001,6.5) instead of col(20,6.5,0.001). In addition the boat model needed to be scaled to 0.001.

The *3D picture module* in CSEI.vi also contains a case structure that change what position and rotation are shown based on two boolean buttons for set-points from the hardware in the loop (HIL) model or from QTM.

Spartan phase

The *PS3* module was the main goal in the Spartan phase. All redundancy were striped from the block diagram giving the clean and structured *PS3 module*, see Figure 4.14. It have the same functions as the one found in CSEI.vi with one upgrade. To prevent indicator chattering of the analogue sticks instead of the sum of x- and y- stick position, a threshold consisting of a circle around the stick's origin is used. The threshold limit is

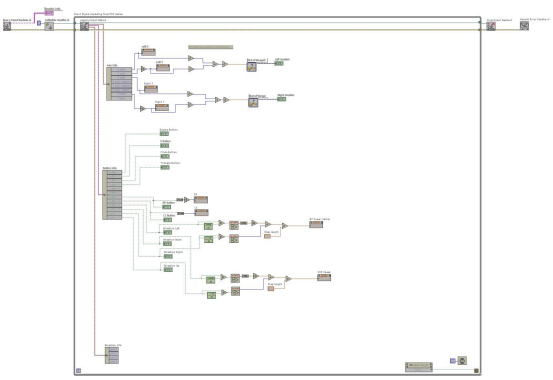


Figure 4.14: LabVIEW, block diagram sorted *PS3 module*

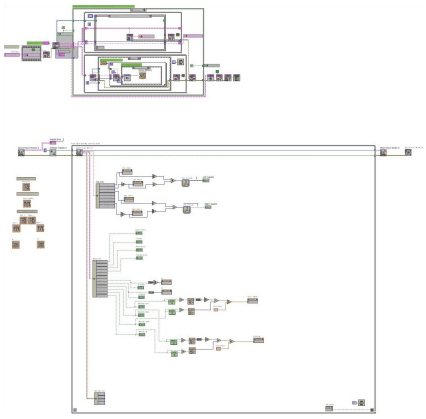


Figure 4.15: LabVIEW, block diagram Spartan phase

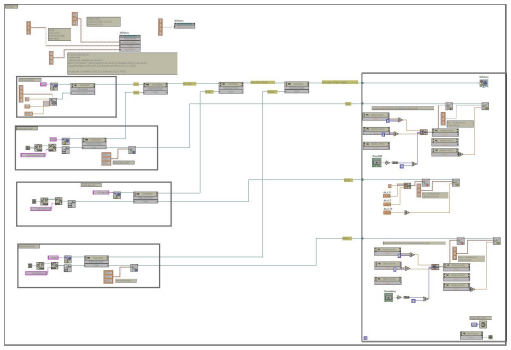


Figure 4.16: LabVIEW block diagram sorted *3D picture module*

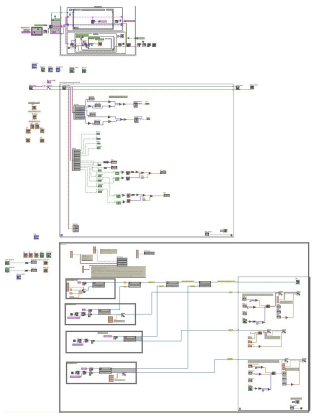


Figure 4.17: LabVIEW block diagram Duplicate phase

compared to the square root of the squared sum of x- and y-position of the sticks. The different indicators were then clustered together reflecting the layout of the front panel, see Figure 4.15.

Duplicate phase

The Duplication phase the target was to organize the *3D picture module*. The camera setting remains unchanged, as well as scale and dimensions. A good reference example is the solarsystem.vi found in the LabVIEW example folder under picture\3D Picture Control.

The water surface object was chosen as the parent object and the orientation is the same as in CSEI.vi. One of the boat models was swapped out for a iceberg model to accommodate Mika Sundland's project.

The module was structured and labelled to to easily see the pattern used for adding and manipulating the object, hook it on to the parent and orientate each object individually. The case and sequence structure were removed as they made things more complicated then needed. Instead frames with no function at all were added giving a visual structure and

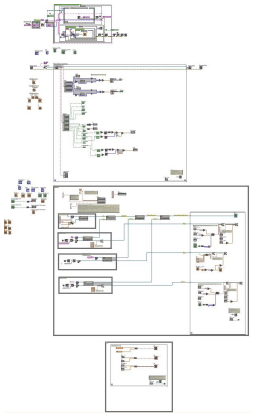


Figure 4.18: LabVIEW, block diagram Expansion phase

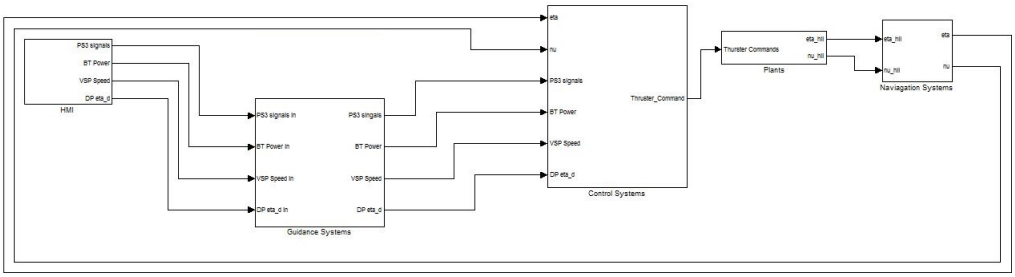


Figure 4.19: Simulink model, control architecture overview

all the "loose" blocks were clustered together, see Figure 4.16. With that the *3D picture module* was added to the block diagram yielding Figure 4.17.

Expansion phase

In the Expansion phase, several tab and other miscellaneous blocks was added to the block diagram in preparation for new controllers and models. The significant change to it is the plot module that plots two parameters in the same graph, receiving from the *SIT module*, seen at the bottom of Figure 4.18.

4.2 Simulink

This section will first give a general overview of the framework and then present the development module by module. Colour coding begin to show itself in the Duplicate phase and follows a similar code as the one found in the MSS GNC toolbox.

4.2.1 Overview

The control architecture is inspired by the modularity of closed-loop system in Skjetne (2005), five segregated modules, User interface, Guidance, Control, Plant, and Measurement. The framework was design using the LOS path-following controller presented in Skjetne et al. (2011) and the hybrid path generation in Skjetne (2005). This can be viewed in the mdl-file found under the Duplicate folder. Implemented this in Simulink means creating five subsystems, see Figure 4.19. The subsystems's names are self-given on what they contain but for completeness they are explicitly stated here. It should be noted that all switches are zero based.

User interface

The User interface module corresponds to the HMI subsystem. Here all inputs from LabVIEW are mapped to, except tuning gains, enablers and the QTM parameters. It then pass it along to the Guidance subsystem.

Guidance module

The Guidance module is the Guidance subsystem. In here the reference signals are constructed based on the inputs, but the in the latest model the signals flows right through to Control subsystem. However in the case of path-following, the path will be created here from the inputted way-points as well as the reference signals.

Control module

The Control module is the Control subsystem. Here the different control laws are implemented, each within their own subsystem and the controller used are chosen with the use of switches and enablers. It takes in the reference signals and control inputs and sends out the thruster commands to the Plant subsystem.

Plant module

The Plant module is the Plant subsystem. This subsystem contains the output port subsystem to the CSE1 hardware, as well as any linear or non-linear model to mimic the real world. The Plant subsystem sends the thruster command to CSE1 and maps them before sending those through the model subsystem. The subsystems within the Plant subsystem are equipped with enabler to enable and disable them at will. The subsystem models sends the output to the Navigation subsystem.

Measurement module

The Measurement module is the Navigation subsystem. Here the data from the Plant subsystem gets routed into a switch and there is also a subsystem handling the input data from the QTM. The switch decides which data to send as state feedback to the control law.

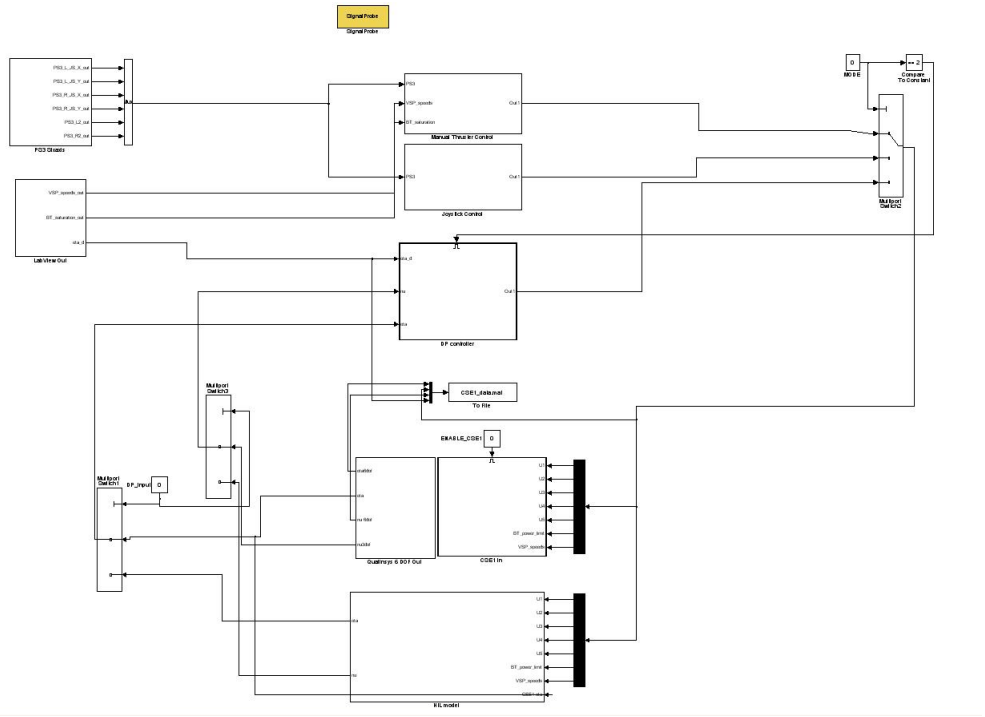


Figure 4.20: Simulink model, control architecture overview of CSEI.mdl Courtesy:Skåtun (2011)

4.2.2 Control architecture

CSEI.mdl

CSEI.mdl contain eight subsystems and three switches in the top level, see Figure 4.20. One subsystem for the direct PS3 signals, one for LabVIEW inputs, two subsystems for direct control, one for individual thrusters and one using the analogue sick, one for the DP controller, one for CSE1 inputs, one for a HIL model and the last one for the QTM outputs. One switch is used for choosing control mode, the other two are deciding the feedback source to the controller. At most it have three levels, found in the *DP controller* subsystem.

The subsystems are structured vertically, with switches on either side and inputs on the left-hand side. Giving it a bit unorthodox structure and the wiring path does not help to get a clear view if the signal flow.

Spartan phase

Following the mantra of emergency minimalism, but still consider modularity, CSEI.mdl was reduced to four subsystems, no switches, two mux and a demux block, see Figure 4.21. The model is crude, but it does the job. The PS3, LabVIEW and direct thruster control were keep as it and the *CSE1 in* subsystem was renamed to *Thruster Allocation*.

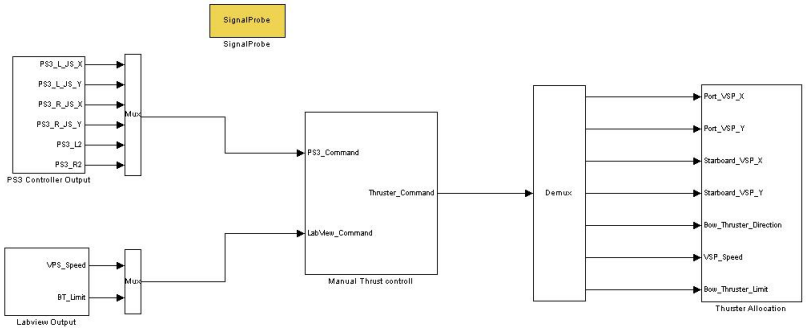


Figure 4.21: Simulink model, control architecture overview of Spartan phase

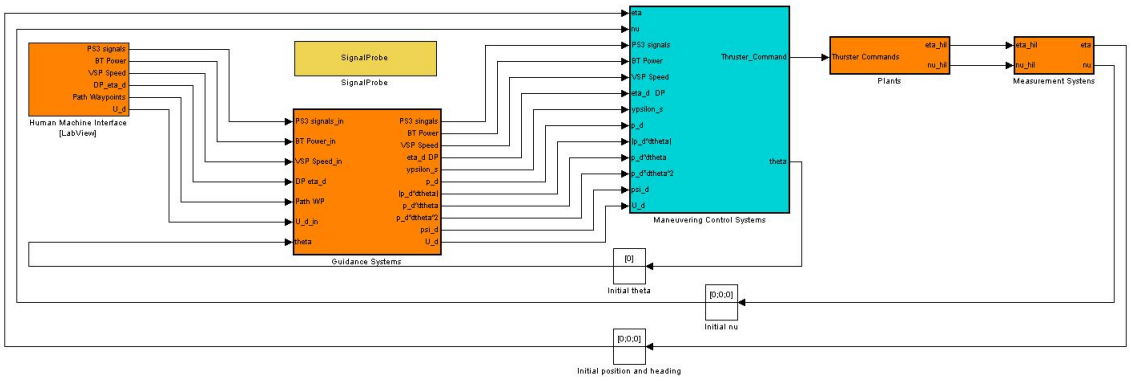


Figure 4.22: Simulink model, control architecture overview of Duplicate phase, stage a

Duplicate phase

It was in this phase the idea of color doing began as a means to quickly keep track of the different intermediate stages. Green as input, red as output, yellow for SIT related blocks and other colors for various stage progression. The combining and splitting of signal to vector or vice versa are now done within the subsystem where those are needed.

The Duplicate phase was done in two stages, first building it with a LOS path-following controller and then later removed it due to technical difficulty. Even without implementing the LOS path-following controller the framework is embedded into the architecture, see Figure 4.22.

The input subsystems for PS3 and LabVIEW, were placed within a subsystem dubbed *Human Machine Interface [LabView]*. A subsystem for constructing reference parameters, *Guidance Systems*, was created for the path generation and reference signals need for LOS path-following. All the control subsystem were placed in a subsystem called *Maneuvering Control Systems* with a feedback to the *Guidance Systems* block. The subsystem known in Spartan phase as *Thruster Allocation* was placed inside a subsystem entitled Plants and the QTM subsystem from known as *Qualinsys 6 DOF Out* was put under the *Measurements Systems* block sending feedback signals to the controllers.

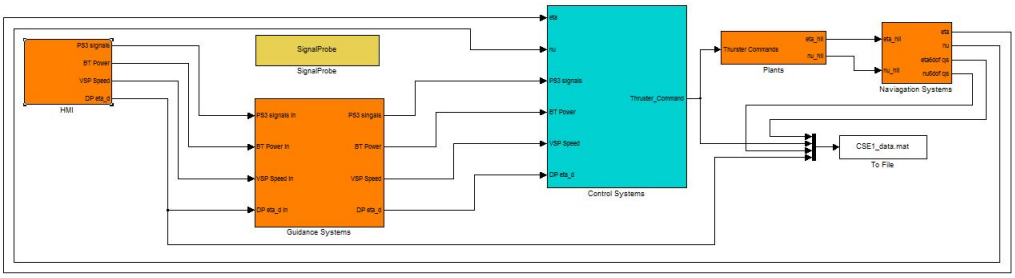


Figure 4.23: Simulink model, control architecture overview of Expansion phase

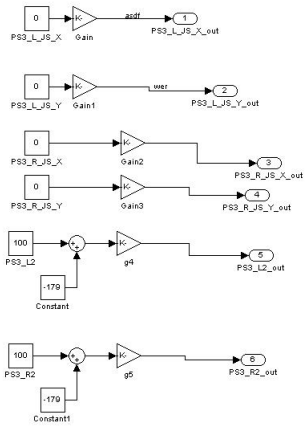


Figure 4.24: Simulink model, *PS3 Sixaxis* of CSEI.mdl Courtesy:Skåtun (2011)

In stage b of Duplicate phase the LOS path-following blocks were removed. The *Measurements Systems* block was renamed into *Navigation Systems*, the *Human Machine Interface [LabView]* was renamed to *HMI* and a To File block was added as a legacy from CSEI.mdl.

Expansion phase

The Expansion phase and stage b of the Duplicate phase overlap, therefore not much were done in the Expansion phase on the top layer. Since both of them are closely identical at the top level, the stage b Duplicate phase figure was omitted from the report, see Figure 4.23.

4.2.3 HMI

The *HMI* block is equivalent to the *PS3 Sixaxis* and *LabView Out* blocks in CSEI.mdl.

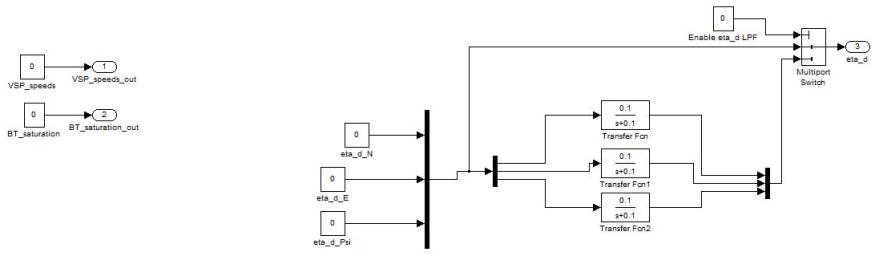


Figure 4.25: Simulink model, *LabView Out* of CSEI.mdl Courtesy:Skåtun (2011)

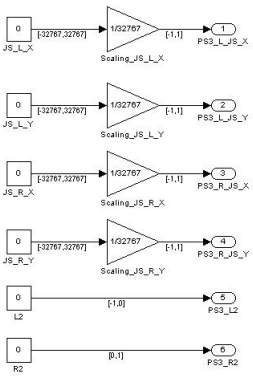


Figure 4.26: Simulink model, *PS3 Controller Output* in Spartan phase

CSEI.mdl

The *PS3 Sixaxis* block divided the analogue signals with 32 676 to convert the signal value to a intervall [-1,1] and something similar was done for the shoulder button signals L2/R2, see Figure 4.24

The *LabView Out* block send the VSP and bow thruster limiter straight through, but for the DP inputs are bundled and duplicated, where one is sent straight to a switch while the other goes through a filter before arriving at the switch, see Figure 4.25.

Spartan phase

The blocks are mostly keep as is. The PS3 subsystem was streamlined removing the offsets and conversion in *PS3 Sixaxis*, see Figure 4.26. In the LabVIEW block the DP inputs were removed leaving only two blocks and two output ports, see Figure 4.27.

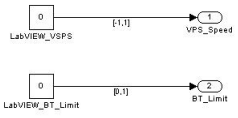


Figure 4.27: Simulink model, *Labview Output* in Spartan phase

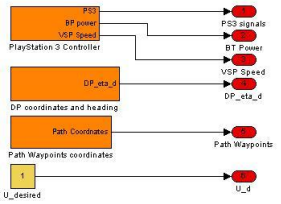


Figure 4.28: Simulink model, *HMI* in Duplicate phase

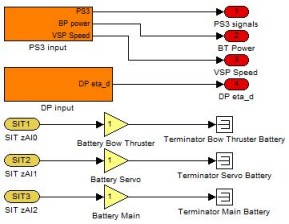


Figure 4.29: Simulink model, *HMI* in Expansion phase

Duplicate phase

The two blocks in Spartan phase are merged into a two levelled subsystem in the Duplicate phase. In addition two new subsystems were added, one for the DP inputs and one for the path way-point coordinates and a reference speed, see Figure 4.28.

Expansion phase

The Expansion phase removed all trace of the path and replaced it with the battery voltage input from SIT, see Figure 4.29. It also added a degrees to radians converter as the input from LabVIEW was switch from radiance to degrees with the dial for more intuitive inputting.

4.2.4 Guidance

The guidance is non-existing in CSEI.mdl and the Spartan phase. It is introduced in the Duplicate phase stage a to accommodate a LOS path-following controller, but serves no purpose beside being a place holder for when path generation is needed.

4.2.5 Plant

The plant corresponds to *CSE1 in* and *HIL model* from CSEI.mdl.

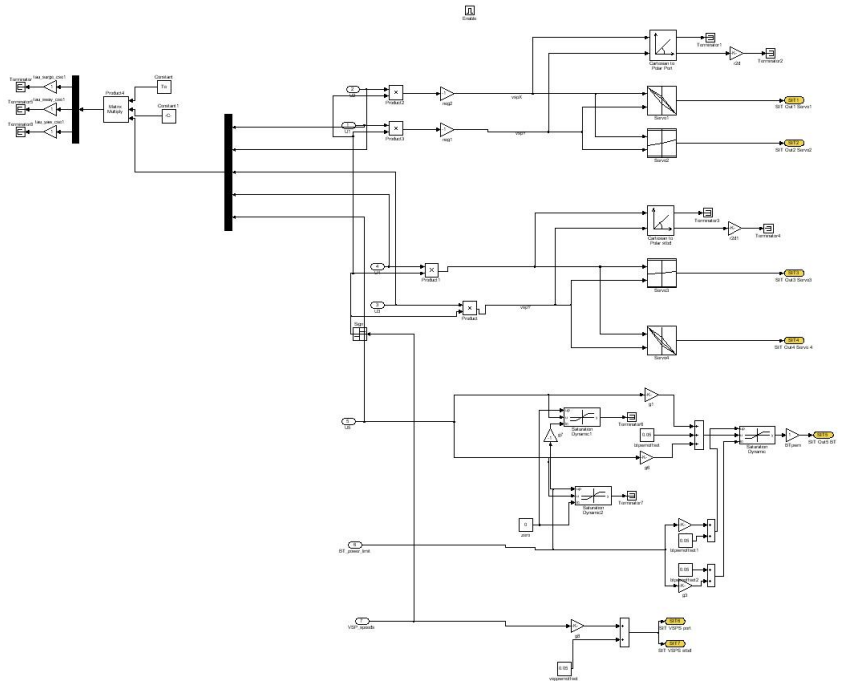


Figure 4.30: Simulink model, *CSE1in* of CSEI.mdl Courtesy:Skåtun (2011)

CSEI.mdl

CSE1 in's blocks can be divided into three groups, VSP (upper right), bow thruster (lower right) and LabVIEW (rest), see Figure 4.30 .

To transform the input signal $u_1, \ldots, u_4$ to commands the VSP can use, four lookup tables are used (one for each servo) to determine the position of the VSP pin to generate the desired thrust force and direction. Each of $u_1, \ldots, u_4$ are multiplied by the sign of the VSP speed limit, if the limit is zero then the commands are all zero. This means to be able to manipulate the VSP pitch angles the limit cannot be zero. u_1 and u_2 are multiplied by -1 since servo 1 and 2 use the same lookup table as servo 3 and 4. This switching of sign is because servo 1 and 2 have the mirrored position compared to servo 3 and 4. Most likely the lookup tables were developed only for servo 3 and 4.

The bow thruster part use three saturation dynamics to determine the signal sent and have two power offsets probably to neutralize some deviation.

The HIL model was made overly complicated, it transformed $\boldsymbol{u}$ back to $\boldsymbol{\tau}$ and sent it into a linear model, see Figure 4.31.

Spartan phase

The plant in this phase is only the *Thruster Allocation* block a light version of *CSE1 in*. Realizing that there are no overlap between the VSP and bow thruster the two were clearly separated as seen in Figure 4.32.

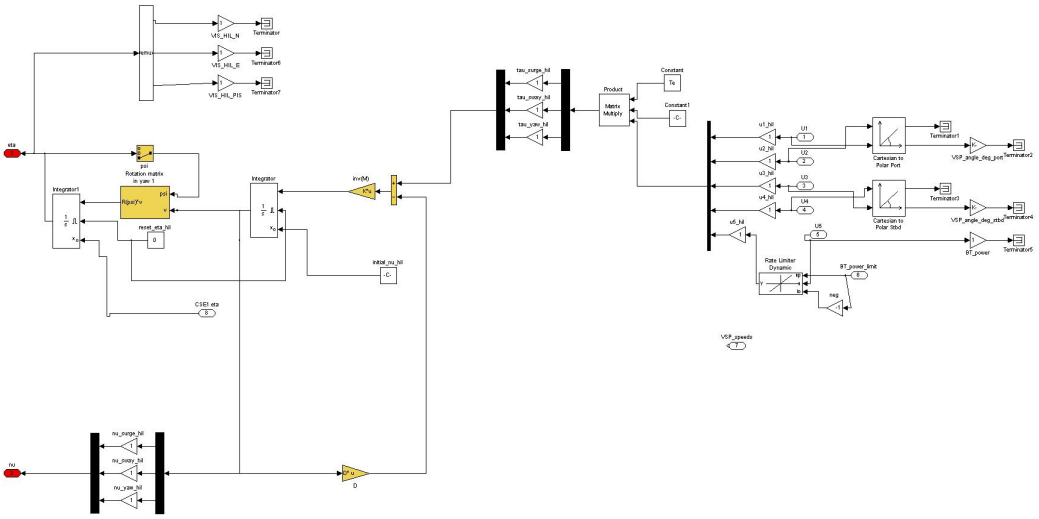


Figure 4.31: Simulink model, *HIL model* of CSEI.mdl Courtesy:Skåtun (2011)

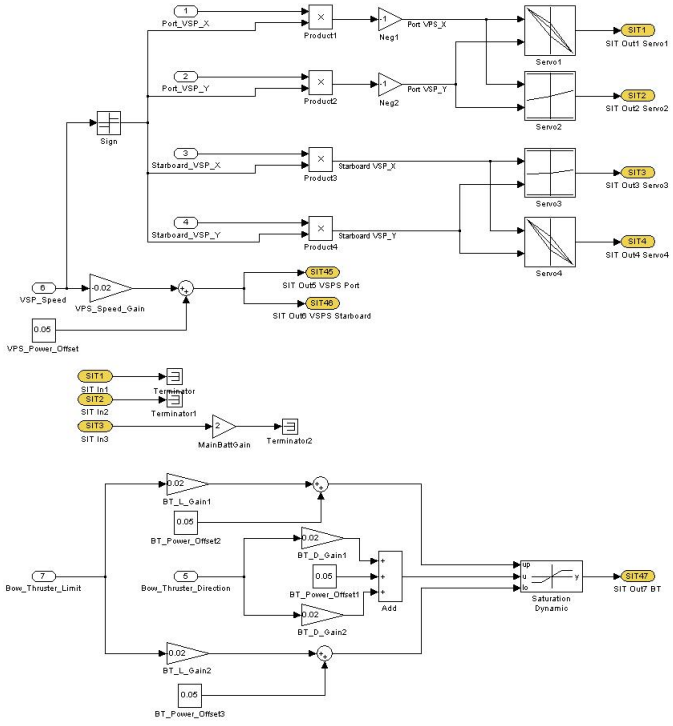


Figure 4.32: Simulink model, *Thruster Allocation* Spartan phase

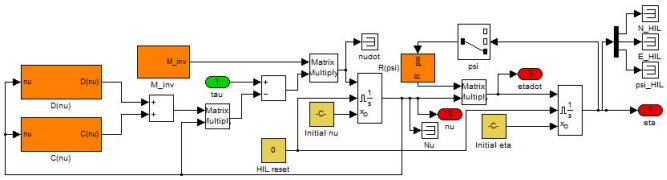


Figure 4.33: Simulink model, *non-linear vessel model* Duplicate phase stage a

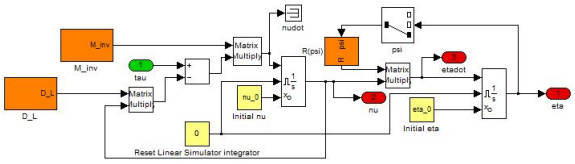


Figure 4.34: Simulink model, *linear vessel model* Duplicate phase stage b

Duplication phase

In the Duplicate phase the VSP and bow thruster components were further divided into two subsystems. A more structured and compact non-linear vessel model was added in stage a (see Figure 4.33)as the *HIL model*, but converted to a linear model in stage b the number of parameters needed to be identified became clear, see Figure 4.34.

Expansion phase

In the Expansionphase gain-blocks were added to accommodate the plots in LabVIEW for better keep track of the signals within the Simulink model.

4.2.6 Navigation

The navigation block is more or less just the Qualisys block and two switches, it is not seen in the Spartan phase as the manual control do not have any use for it.

CSEI.mdl

This block was made to work around the shuffling of signals caused by the bug in QTM-driver.vi. The signals are in groups of three and selected based on the Skåtun's ad-hoc approach to this problem. The positions are give in mm and the angle are in degrees and they are converted to meter and radians before being bundled and extracted as $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$, see Figure 4.35.

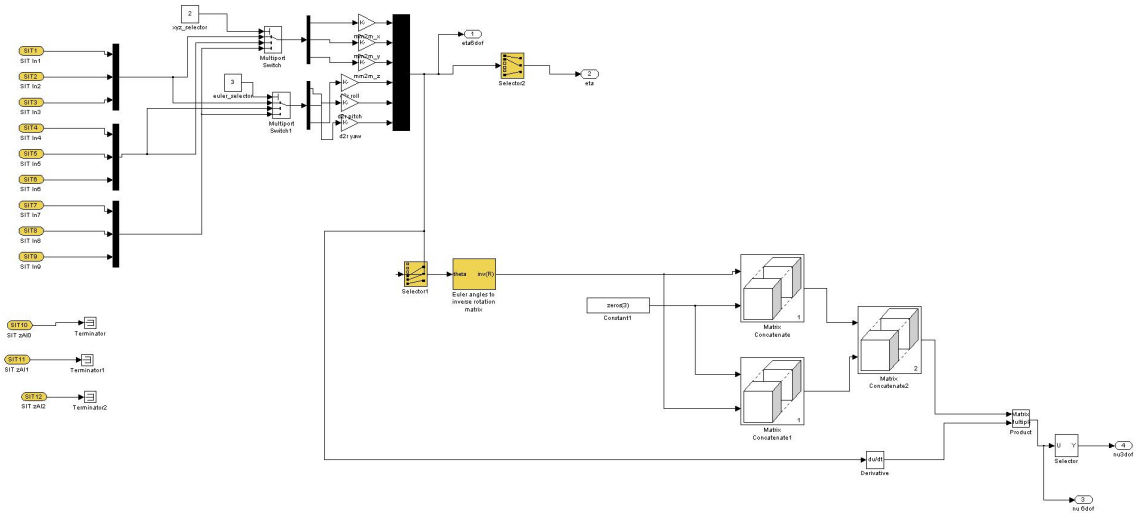


Figure 4.35: Simulink model, *Qualinsys 6 DOF Out* of CSEI.mdl Courtesy:Skåtun (2011)

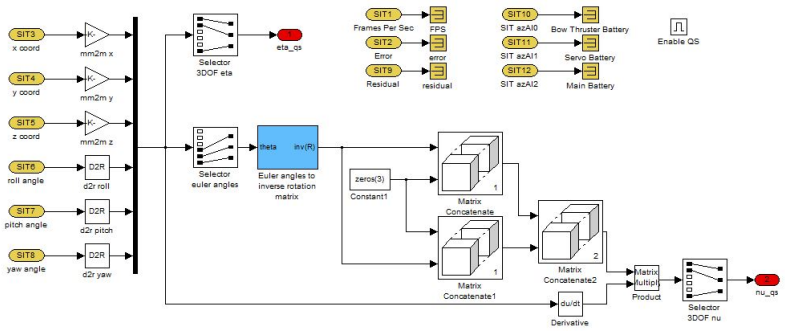


Figure 4.36: Simulink model, *Qualinsys* Duplicate phase

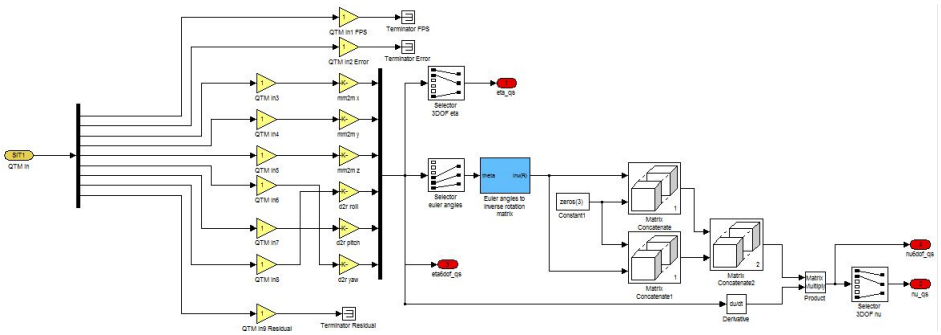


Figure 4.37: Simulink model, *Qualinsys* Expansion phase

Duplicate phase

The Duplicate phase was conducted without any knowledge on how the QTM worked, as in not knowing the name of the system (Qualinsys without the "n") or the route the data took to go from QTM to Simulink. Not much though when into this module in this phase. The workaround developed by Skåtun was removed and the working assumption was things will work out in the end, see Figure 4.36. Therefore it contains several mistakes that is corrected in the Expansion phase.

Expansion phase

With proper knowledge on the QTM the block was properly named *Qualisys 6 DOF* and solving the shuffling problem the workaround was never introduced back in. Gains were added to accommodate the signal surveillance implemented in LabVIEW, see Figure 4.37. A significant change is going from nine SIT input port to one. This is due to the modifications done the the QTMdriver.vi. When fixing the shuffling problem, a bundling of data was also implemented in the QTMdriver.vi by Torgeir Wahl.

Chapter 5

System identification

System identification of CSE1 is needed to create a better HIL model and to construct observes for it. Part of the system identification of CSE1 have been done by Skåtun in his thesis, where he determined two coefficients in the linear dampning matrix and used a gray box to find the remaining coefficients in that matrix as well as the coefficients in the mass and added mass matrix. The results of the system identification were not used since several of parameters that was determined had a variance several times the parameter value making it untrustworthy. Instead he opted to use CyberShip2's coefficients as the ships dimensions and hull have shared similarities.

This chapter takes aim to process Skåtun's work and suggest and plan improvement to his methodology of system identification.

5.1 Vessel Model

To keep it simple, a model suited for ship positioning can according to Fossen (2002) be expressed as

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{R}(\psi)\boldsymbol{\nu} \\ \boldsymbol{M}\dot{\boldsymbol{\nu}} &= \boldsymbol{\tau} - \boldsymbol{D}\boldsymbol{\nu} \end{aligned} \tag{5.1}$$

where $\boldsymbol{\eta} = col(\boldsymbol{p}, \psi)$ contains the north, east position and yaw angle of the vessel in the $\mathcal{E}$-frame, $\boldsymbol{p} = col(x, y)$, $\boldsymbol{\nu} = col(u, v, r)$ is a velocity vector decomposed in the $\mathcal{B}$-frame containing the velocity component in surge sway and yaw, and $\boldsymbol{\tau} = \boldsymbol{B}\boldsymbol{f}$ is the command force vector decomposed in the $\mathcal{B}$-frame.

The coefficients in the matrices follows the notion of The Society of Navel Architects and Marine Engineers,SNAME (1950). The rotation matrix

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.2}$$

The mass matrix

$$\boldsymbol{M} = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & mx_g - Y_{\dot{r}} \\ 0 & mx_g - N_{\dot{v}} & I_z - N_{\dot{r}} \end{bmatrix} \tag{5.3}$$
$$Y_{\dot{r}} = N_{\dot{v}}$$

The dampning matrix

$$\boldsymbol{D} = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix} \tag{5.4}$$

The thruster configuration matrix using the same notation as in Skåtun (2011)

$$\boldsymbol{B} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ l_{y1} & l_{x1} & l_{y2} & l_{x2} & l_{x3} \end{bmatrix} \tag{5.5}$$

$\boldsymbol{f} = col(f_1, f_2, f_3, f_4, f_5)$ is the command force vector.

5.2 Variable reduction

A reason for the high variance in Skåtun's graybox system identification method must be due to all the parameters he left as variables. The goal is to reduce the number of variable parameters.

He have in his thesis determined X_u and Y_v and found estimates for $X_{\dot{u}}$ and $Y_{\dot{v}}$. Removing the $\boldsymbol{K}$ matrix in Skåtun's extended thrust matrix $\boldsymbol{\tau_e}$ would reduce it by five. This is done by mapping the $\boldsymbol{f}$ to thruster inputs $\boldsymbol{u} = col(u_1, u_2, u_3, u_4, u_5)$ through lookup tables.

Another parameter that can be fixed is N_v by towing CSE1 in an angle and then find the difference in moment created in the stern and aft for each speed. Plot this into a graph and this would and conduct a linear regression similar to how Skåtun determined X_u and Y_v.

5.2.1 $\boldsymbol{f}$ to $\boldsymbol{u}$ mapping

To map the relationship between $\boldsymbol{f}$ and $\boldsymbol{u}$, each $u \in [-1, 1]$ with incremental step of 0.1 were measured individually and in par of $u_1 = u_3$ and $u_2 = u_4$ for comparison with the individual measurement. The VSP speed limit was set to 0.4 in all measurements to make them comparable to Skåtun's. u_5 was not measured since it proved to be linear and well defined in Skåtun (2011).

The setup for these measurement is the same as for towing CSE1 in a angle. Another reason to do the thruster force measurement is in his measurement the same force and direction could be obtained using two different u input and the documentation on the VSP measurements were lacking.

Table 5.1: Measurement equipment list

Measurement equipment	
2×4 plank	6
Eye bolt	6
Rotation free attachement hooks	15
Bar clamp	6
Carpender's rule	1
Force ring	3
Spring	3
Rope	6
Tape roll	1

Figure 5.1: Rope, hook, spring and force ring connections

5.3 Towing of CSE1

Towing of CSE1 was preformed for heading, 0, 45, and 90 degrees w.r.t. towing direction similar to what was done in Skåtun (2011). This was done for comparison purposes. The measurements were taken for -0.3 m/s to 0,3 m/s with a step increase of 0.01. In the interval, for the intervals -0.4 to -0.3 and 0.3 to 0.4 the increase was 0.02. For the interval -0.6 to -0.4 and 0.4 to 0.6 the increase was 0.5. This yields 78 times 3 = 234 data set to process.

5.4 Thruster force measurements

In addition to measuring the thruster force from the VSP, with VSP speed at 0.4. the measurement were taken individually for each of the VSP u's, from -1 to 1 with a step length of 0.1, and as a pair with u_1, u_3 and $u2, u4$ giving 120 data set to create a mapping in the form of a look-up table to replace the $\boldsymbol{K}$ matrix in Skåtun's thruster configuration matrix. See appendix for sketch note.

The set of measurement were done to check for effect of hull interaction and thruster on

Figure 5.2: Set-up for measurements

Figure 5.3: Towing configuration 90 degrees, pure surge

Figure 5.4: Towing configuration 0 degrees, pure sway

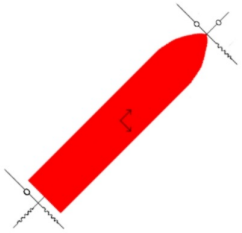


Figure 5.5: Towing configuration 45 degrees, yaw due to sway

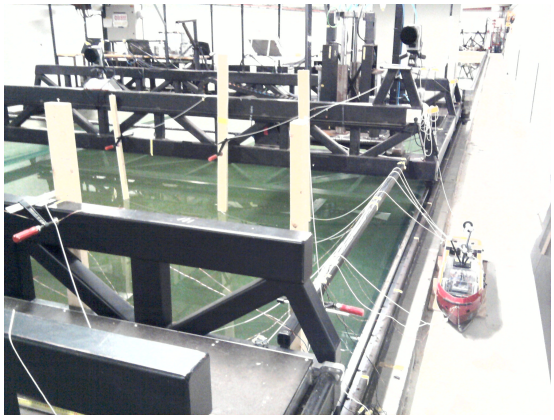


Figure 5.6: CSE1 ready for testing

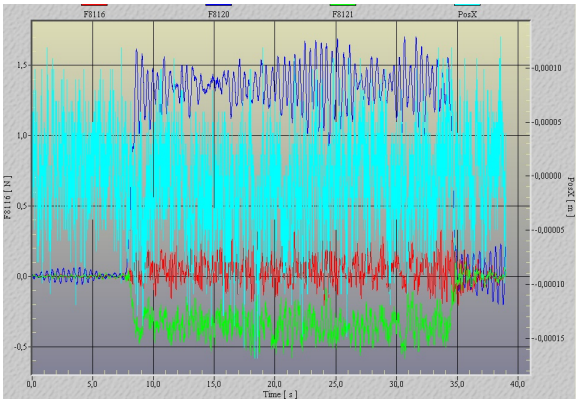


Figure 5.7: Thruster force measurements, $u_1 = 1.0$

thruster interaction. With the look-up table, when the control law demands a force or momentum the lookup table will map it to the appropriate u command, instead of trying to tune the gain matrix.

Chapter 6

Activities

This chapter summarize the different activities related to CSE1

6.1 MCLab

The MCLab was used during week 11, 12, 13 and 16 together with Mika Sundland. In week 11,12 and 13 Mika was there to fulfil the occupational safety and health requirement as well as assisting in the rigging. Originally the MCLab was booked for week 11 and 12. The plan was to test out the modularized LabVIEW and Simulink files and work out any bugs. This was completed in the middle of week 12 and it was decided to try and do system identification for the remaining week. However I did not have knowledge about how to operate the carriage or where to find the different equipments needed. In addition Torgeir Wahl was not at hand to help, little could be done with the last few days.

However no one had booked the MCLab for Easter, week 13 and Torgeir Wahl was available the first days of week 13. This presented an opportunity to do thorough towing experiments of CSE1 and the remaining days was used to plan the setup of the towing. The first rigging took a whole day to finish, since it had to be done twice since the first configuration would not give usable data as the rope would bend around the stern edge of the model. With each rigging, valuable experience was gained on doing minor practical things as fasting and calibrating of ropes. The whole Monday was used for rigging and calibrating, Tuesday towing CSE1 in a angle, Wednesday thruster force measurements, Thursday towing in sway, and Friday towing in surge.

The knowledge and experience earned during those weeks came to good use when it was Mika Sundland's turn to do his experiments in week 16. In that week I had a support role handling the QTM and assisting in using the Expansion phase files for his experiments.

It was during his experiments it was noticed that area of visibility of the IR markes for the cameras was 100 percent reliable within a area of nine $\times$ six meters. One of the causes of this might have been the barrel simulating the iceberg overshadowed the CSE1 markes, but it is worth noting when planing path following experiments. The dead zones are approximately one meter from either side of the basin walls and four meter from the cameras. QTM are able to extrapolate, but the most accurate area are from four meter

from camera to 13 meters from the camera. It is most likely that the length is increased with there are not barrel overshadowing the CSE1 markers.

6.2 DP mode

When testing the DP mode, CSE1 was able to easily converge to the desired position. However after initially obtaining the desired position it became unstable when trying to obtain desired heading, leading to spinning of vessel heading and divergence of desired position. It was noticed that it used primarily the bow thruster when trying to obtain the desired heading, confirming the findings in Skåtun (2011). The power of the bow thruster is underestimated leading to an constant overshooting of the heading.

6.3 Demonstrations and tours

CSE1 and it's HMI were demonstrated in the MCLab several in the course of the time I have been working with it. The majority of the demonstrations were of the drop-in variety while two were scheduled demonstrations.

The first time was to a group of students that won a dream day at NTNU and were shown around on the 18. april 2013. The other time was a part of a tour for a group from Serbia when they were visiting Marinteknisk Senter(MTS). The demonstrations involved explanation of the HMI, how the manual control steered CSE1 using a PS3 controller and then letting them try it.

Chapter 7

Future works

The main thing learned from working with this project is the need of planning. Poor planning leads to poor execution meaning mediocre report. The schedule should be written down with long term goal and short term milestone and deadlines. In addition to a daily log stating what was done and sketching the work for the next day.

7.1 Data processing

The first priority is to process the measurement of the towing and thruster forces, to determine a third coefficient in the damping matrix and set up the new thruster mapping with lookup table instead of a gain matrix. Test the lookup table setup in LabVIEW simulation.

7.2 Path generation

A workable path generation is needed. The first step is to construct it to parametrize a linear path made up of two way-points, then parametrization using three way-points and create two sub-paths with hybrid parametrization. From there expand to three, four and five sub-paths. The next step is to repeat the first step, but this time using a polynomial of order 2 instead. After that path generation of a elliptic path and a sin/cos path. The last step test and verify it in LabVIEW.

7.3 Controller

Attempts to implement a LOS path-following controller ended in failure and a simplified LOS path-following controller is need to be designed such that it can be more easily implemented in Simulink. Step two is to implement and test the controller in LabVIEW. The priority is to get a path-following controller.

Using a LOS means to only steer with positive surge speed and heading. The practical implication of this means the thrust direction from the VSP should be $\pm 45°$ with respect to the x-axis. This means $\{u_1, u_3\} \in [0, 1)$ and $\{|u_2|, |u_4|\} \in [0, 0.5)$ with the condition $\{|u_2|, |u_4|\} \leq \{u_1, u_3\}$.

7.4 MCLab

The MCLAb have been reserved 16.-21. September and 18.-23 November. The plan is to have have the path generation, path-following controller and lookup table configuration done by 14. September and test it out with CSE1 in the basin from the 16. September. If all go according to plan logging of the data will be used in Skåtun's graybox.

If the work done in September is successful then November will be used for testing out different types of controllers. If it was a failure then the work will continue on gaining a usable path-following controller

7.5 Manual thruster control

If time allows it then I want to replace BtSix and PPJoy with another program called MotionJoy. The the reason for this is that MotionJoy can read the shoulder buttons L2 and R2 to axis instead of buttons, meaning they become pressure sensitive. This will allow a better manual control of the bow thruster. As of now the choice is either no thrust or full throttle, with the pressure sensitivity L2, R2 becomes more like pedals on a car.

7.6 User manual

If there is time then a user manual on how to use CSE1 and infrastructure should be written so that when other people work with CSE1 they do not have to reverse engineer the files to be able to modify them. The work done in the last year should provide a good basis for this, but the challenge is to write it in a friendly structure.

7.7 Observer

As QTM signal can be lost and having experience problems with zero data a exponentially stable observer would be advantageous to perform dead reckoning until the QTM comes back on-line. The difficulty is to be able to find sufficiency good data for the observer as system identification of CSE1 are limited. The option of using CyberShip2's data might be a good alternative considering the circumstances.

Bibliography

Fossen, T. I. (2002), *Marine Control Systems: Guidance, Navigation, and Control of Ships, Rigs and Underwater Vehicles*, 1st edn, Marine Cybernetics, Trondheim, Norway.

Fossen, T. I. (2011), *Handbook of Marine Craft Hydrodynamics and Motion Control*, John Wiley & Sons Ltd., United Kingdom.

Model Slipway (n.d.), 'Anchor Handling Tug AZIZ'.
URL: *http://www.modelslipway.com/aziz.htm*

Skåtun, H. N. (2011), Development of a DP system for CS Enterprise I with voith schneider thrusters, Master's thesis, Norwegian University of Science and Technology.

Skjetne, R. (2005), The Maneuvering Problem, PhD thesis, Norwegian University of Science and Technology.

Skjetne, R., Jørgensen, U. and Teel, A. R. (2011), Line-of-sight path-following along regularly parametrized curves solved as a generic maneuvering problem, *in* 'Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference', Orlando, Florida, USA, pp. 2467–2474.

SNAME (1950), Nomenclature for treating the motion of a submerged body through a fluid, *in* 'Technical and Research Bulletin No. 1-5'.

Thorvaldsen, C. F. L. (2011), Formation control of marine vessels, Master's thesis, Norwegian University of Science and Technology.

Appendix A

Marine vessel definition

In this thesis the motion variables of a marine vessel are defined according to Figure A.1 and follows the SNAME (1950) notation. For a thorough treatment see Fossen (2002) and Fossen (2011).

Table A.1: The notation of SNAME (1950) for marine vessels. Courtesy: Fossen (2002)

DOF		force and moments	linear and angular velocities	positions and Euler angles
1	motion in the x-direction (surge)	X	u	x
2	motion in the y-direction (sway)	Y	v	y
3	motion in the z-direction (heave)	Z	w	z
4	rotation about the x-axis (roll)	K	p	ϕ
5	rotation about the y-axis (pitch)	M	q	θ
6	rotation about the z-axis (yaw)	N	r	ψ

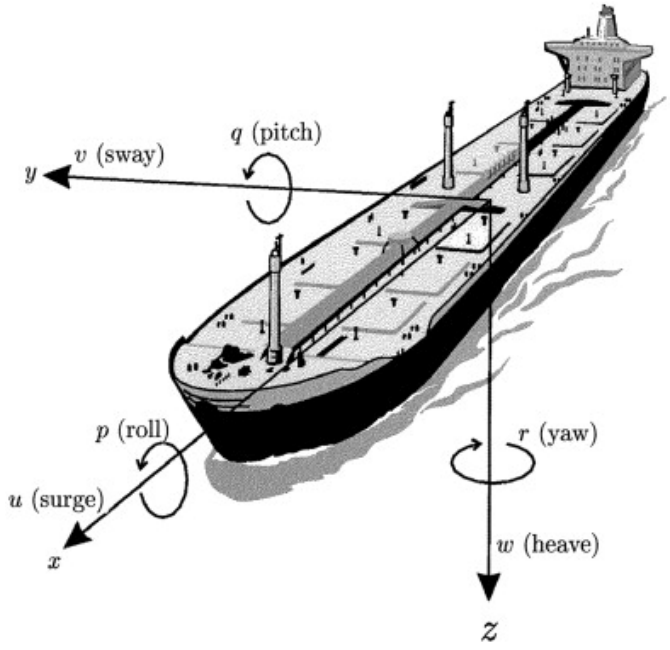


Figure A.1: Motion variables for a marine vessel. Courtesy: Fossen (2002)

Appendix B

Sketch note of towing and thruster force measurements

NCG. SIGNING

$$M\dot{\nu} + D_L\nu = f \quad , \dot{\nu} = 0$$

$$D_L\nu = f$$

$$D_L = \begin{bmatrix} -X_u & 0 & 0 \\ 0 & -Y_v & -Y_r \\ 0 & -N_v & -N_r \end{bmatrix} \quad , \nu = \begin{bmatrix} u \\ v \\ r \end{bmatrix}$$

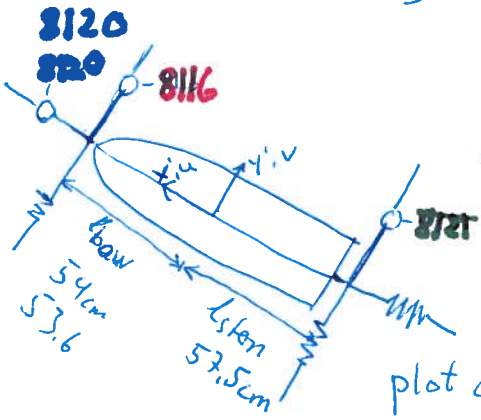
$$\rightarrow -X_u \cdot u = X$$

$$-Y_v \cdot v - \cancel{Y_r \cdot r}^{r=0} = Y$$

$$-N_v \cdot v - \cancel{N_r \cdot r}^{r=0} = N$$

$$M = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 \\ 0 & m - Y_{\dot{v}} & m x_g - Y_{\dot{r}} \\ 0 & m x_g - N_{\dot{v}} & I_z - N_{\dot{r}} \end{bmatrix} \quad , f = \begin{bmatrix} X \\ Y \\ N \end{bmatrix}$$

$\tau = f$



Pressure + Tension =

plot diff moment

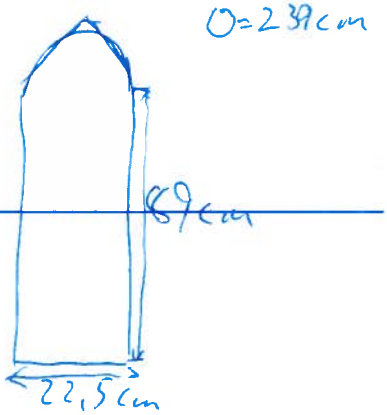
$$\tau = B f_{act} \quad , \tau = \begin{bmatrix} X \\ Y \\ N \end{bmatrix} \begin{matrix} [N] \\ [N] \\ [Nm] \end{matrix} \quad B = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ b_p & -l_{VSP} & -b_S & -l_{VSP} & l_{BT} \end{bmatrix} \quad f_{act} = \begin{bmatrix} F_{u_1} \\ F_{u_2} \\ F_{u_3} \\ F_{u_4} \\ F_{u_5} \end{bmatrix} \begin{matrix} [N] \\ [N] \\ [N] \\ [N] \\ [N] \end{matrix}$$

$B = T_e$

$f_{act} = B^+ \tau \quad [N]$

$$u_e = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix} \begin{matrix} - \\ - \\ - \\ - \\ - \end{matrix}$$

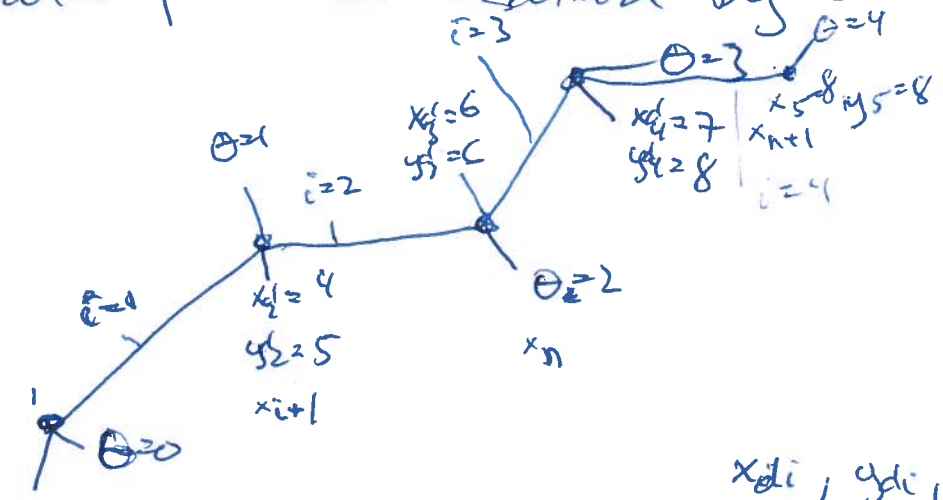
$\tau \rightsquigarrow f_{act} \rightsquigarrow u_e \rightarrow$ plant



Appendix C

Sketch note of path generation

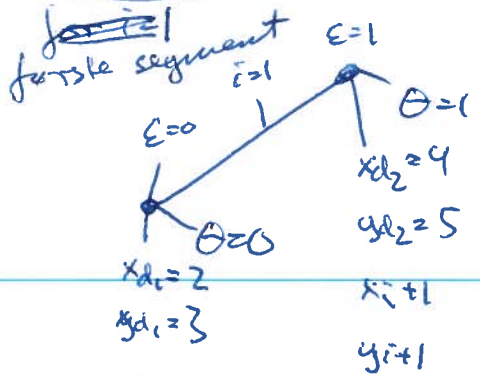
path parameterization by Θ



x_{di} , y_{di} , $\psi_{di} = \arctan2\left(\frac{y_i}{x_i}\right)$

$\tilde{P}_d^{\varepsilon}(i,\varepsilon) = a_i \varepsilon + b_i$

$\varepsilon \in [0,1)$



$\varepsilon = 0 \Rightarrow \begin{bmatrix} x_{d1}=2 \\ y_{d1}=3 \end{bmatrix}$ $\quad \varepsilon = 1 \Rightarrow \begin{bmatrix} x_{d2}=4 \\ y_{d2}=5 \end{bmatrix}$

$a_1 = ?$ $\quad b_1 = ?$

~~$\varepsilon \neq 0 \quad i=1$~~
~~$P_d = a_i \varepsilon + b_i$~~

$\tilde{P}_d^{\varepsilon}(1,0) = a_1 0 + b_1 \Rightarrow b_1 = \begin{bmatrix} x_{d1}=2 \\ y_{d1}=3 \end{bmatrix} = \begin{bmatrix} x_{d1} \\ y_{d1} \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$

$\tilde{P}_d^{\varepsilon}(1,1) = a_1 \cdot 1 + b_1 = \begin{bmatrix} x_{d2}=4 \\ y_{d2}=5 \end{bmatrix}$

$\Rightarrow a_1 + \begin{bmatrix} x_{d1}=2 \\ y_{d1}=3 \end{bmatrix} = \begin{bmatrix} x_{d2}=4 \\ y_{d2}=5 \end{bmatrix} \Rightarrow a_1 = \begin{bmatrix} x_{d2}-x_{d1} \\ y_{d2}-y_{d1} \end{bmatrix} = \begin{bmatrix} 4-2 \\ 5-3 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$

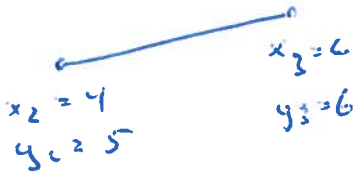
atan2(3,2) = 0.9828

....dot eller dot...

(1.1)

$a_2 = ?$ $b_2 = ?$

$\varepsilon = 0 \Rightarrow \begin{bmatrix} x_{d_2} = 4 \\ y_{d_2} = 5 \end{bmatrix}$ $\varepsilon = 1 \Rightarrow \begin{bmatrix} x_{d_3} = 6 \\ y_{d_3} = 6 \end{bmatrix}$



$\hat{p}_d^{\varepsilon}(2,0) = a_2 \cdot 0 + b_2 = \begin{bmatrix} x_{d_2} = 4 \\ y_{d_2} = 5 \end{bmatrix}$

$\Rightarrow b_2 = \begin{bmatrix} x_{d_2} = 4 \\ y_{d_2} = 5 \end{bmatrix}$

$\hat{p}_d^{\varepsilon}(2,1) = a_2 \cdot 1 + b_2 = \begin{bmatrix} x_{d_3} = 6 \\ y_{d_3} = 6 \end{bmatrix}$

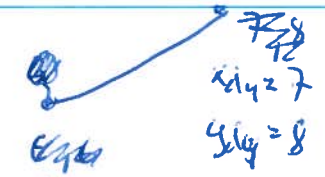
$= a_2 + \begin{bmatrix} x_{d_2} = 4 \\ y_{d_2} = 5 \end{bmatrix} = \begin{bmatrix} x_{d_3} = 6 \\ y_{d_3} = 6 \end{bmatrix}$

$\Rightarrow a_2 = \begin{bmatrix} x_{d_3} - x_{d_2} \\ y_{d_3} - y_{d_2} \end{bmatrix} = \begin{bmatrix} 6 - 4 \\ 6 - 5 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

---

a_3 b_3

$\varepsilon = 0 \Rightarrow p_d = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$ $\varepsilon = 1 \Rightarrow p_d = \begin{bmatrix} 7 \\ 8 \end{bmatrix}$



$p_d(3,0) = a_3 \cdot 0 + b_3 = \begin{bmatrix} 6 \\ 6 \end{bmatrix} \Rightarrow b_3 = \begin{bmatrix} 6 \\ 6 \end{bmatrix}$

$p_d^{\varepsilon}(3,1) = a_3 \cdot 1 + b_3 = \begin{bmatrix} x_{d_4} = 7 \\ y_{d_4} = 8 \end{bmatrix}$

$= a_3 + \begin{bmatrix} x_{d_3} \\ y_{d_3} \end{bmatrix} = \begin{bmatrix} x_{d_4} \\ y_{d_4} \end{bmatrix}$

$\Rightarrow a_3 = \begin{bmatrix} x_{d_4} - x_{d_3} \\ y_{d_4} - y_{d_3} \end{bmatrix} = \begin{bmatrix} 7 - 6 \\ 8 - 6 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

given $\theta = 0.5$ $\quad p_d^{\theta}(\theta) = ?$

$i = \lfloor \theta \rfloor + 1 = \lfloor 0.5 \rfloor + 1 = 0 + 1 = 1$

$\varepsilon = \theta - \lfloor \theta \rfloor = 0.5 - \lfloor 0.5 \rfloor = 0.5 - 0 = 0.5$

$$\tilde{p}_d^{\varepsilon}(1, 0.5) = a_1 0.5 + b_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix} 0.5 + \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

$$\Rightarrow p_d^{\theta}(\theta = 0.5) = \begin{bmatrix} x = 3 \\ y = 4 \end{bmatrix}$$

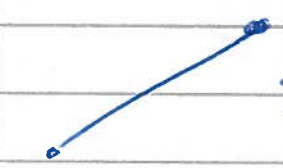
$\psi(\theta) = ?$ $\quad \psi(\theta = 0.5) = ?$

$\psi(\theta) = \arctan2\left(\frac{y'(\theta)}{x'(\theta)}\right)$ $\quad \psi(\theta = 0.5) = \arctan2\left(\frac{y'(\theta = 0.5)}{x'(\theta = 0.5)}\right)$

$\theta = 0.5 \Rightarrow i = 1, \varepsilon = 0.5$

$$\begin{bmatrix} x'(\theta = 0.5 \Rightarrow i = 1, \varepsilon = 0.5) \\ y'(\theta = 0.5 \Rightarrow i = 1, \varepsilon = 0.5) \end{bmatrix} = \tilde{p}_d^{i=1,\varepsilon}(i = 1, \varepsilon = 0.5) = a_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} x_{d_2} - x_{d_1} \\ y_{d_2} - y_{d_1} \end{bmatrix}$$

$$\Rightarrow \psi(\theta = 0.5) = \arctan2\left(\frac{2}{2}\right) = 0.7854$$



$\varepsilon = 0 \Rightarrow x(\varepsilon) = 2 \qquad \varepsilon = 1 \Rightarrow x(\varepsilon) = 4$

ε

$x = a\varepsilon + b$

$x(0) = a \cdot 0 + b = 2 \Rightarrow b = 2$

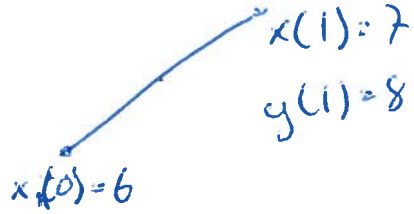
$x(1) = a \cdot 1 + b = 4$

$= a + 2 = 4 \Rightarrow 4 - 2 = a$

$\Rightarrow a = 2$

$\varepsilon = 0.5 \Rightarrow x(0.5) = 2 \cdot 0,5 + 2$

$= 1 + 2 = 3$

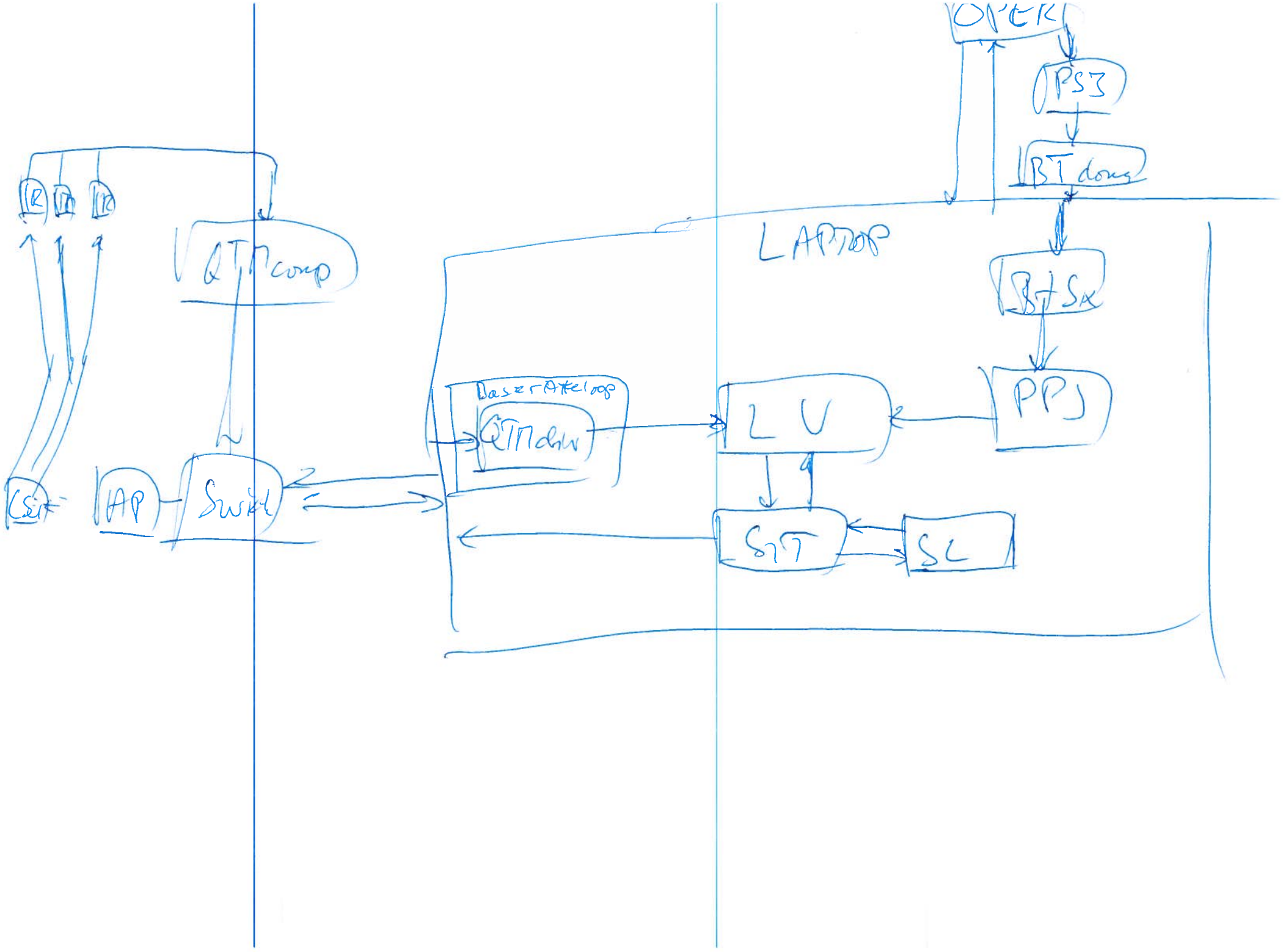


$\varepsilon = 0.5$ $a = 1$ $b = 6$

$x = 1 - 0{,}5 + 6 = 6.5$

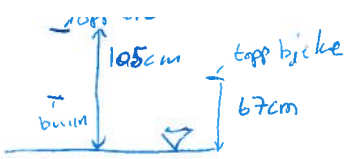
Appendix D

Sketch Signal flow

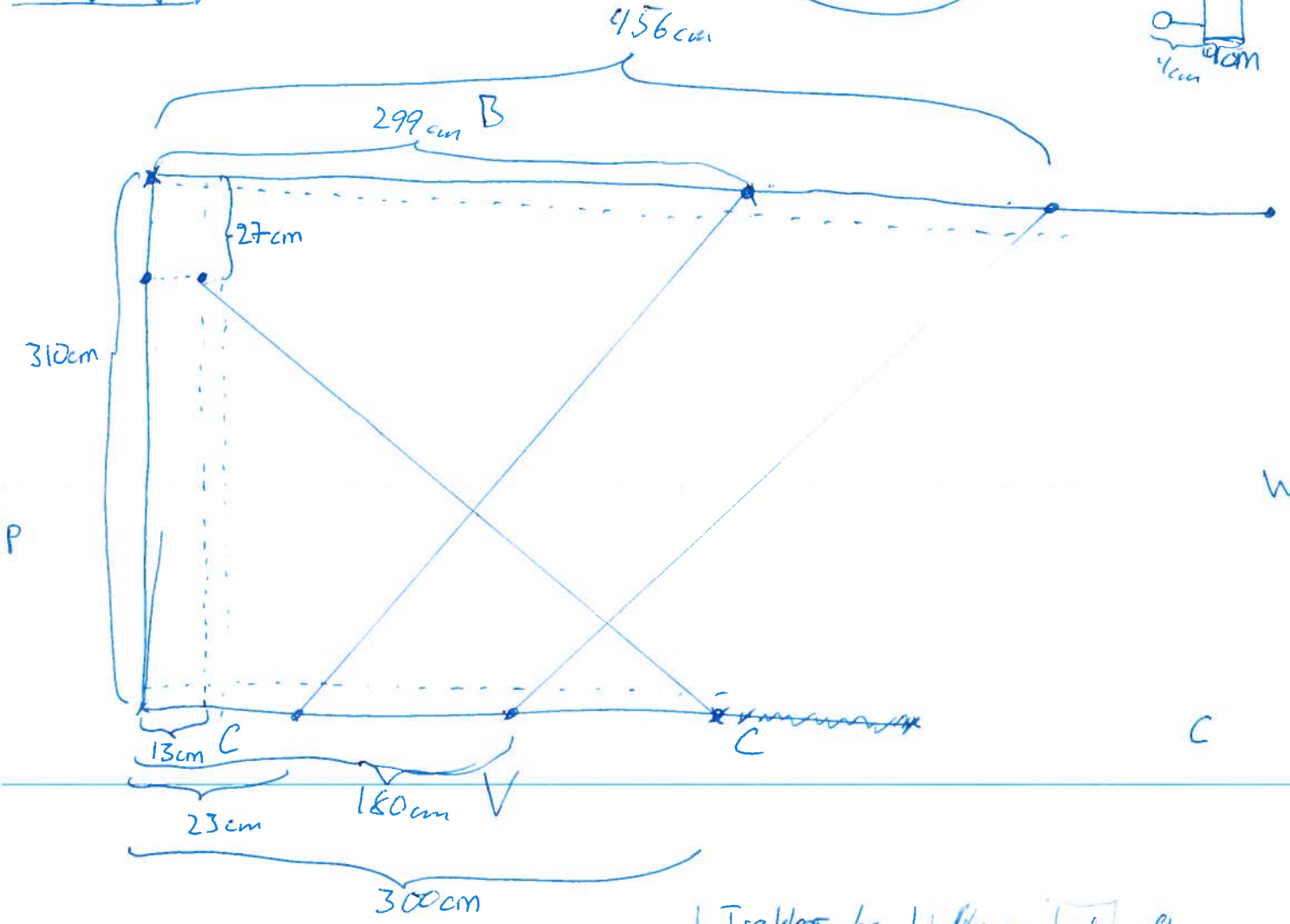

LAPTOP
PS3
BT dongle
BT Stack
PPJ
LU
SIT
SL
QTM comp
QTM drv
AP
Switch

Appendix E

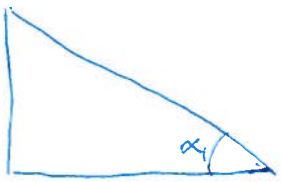
Towing 45 degrees plus dead zone map



45° isv

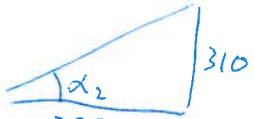


310
- 27
283



300
- 13
287

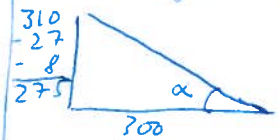
$\alpha_1 = \arctan\left(\frac{283}{287}\right) \approx 44.6°$



299
- 23
276

$\alpha_2 = 48.3°$

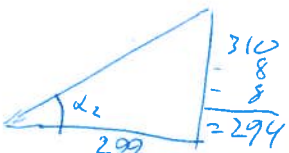
Trekket fra bjelke og feste - 8cm



300
- 13
- 8
279

(1)

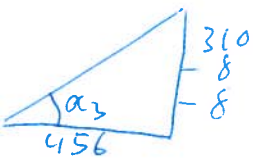
$\alpha_1 = \arctan\left(\frac{275}{279}\right) \approx 44.6°$



299
- 23
276

(1)

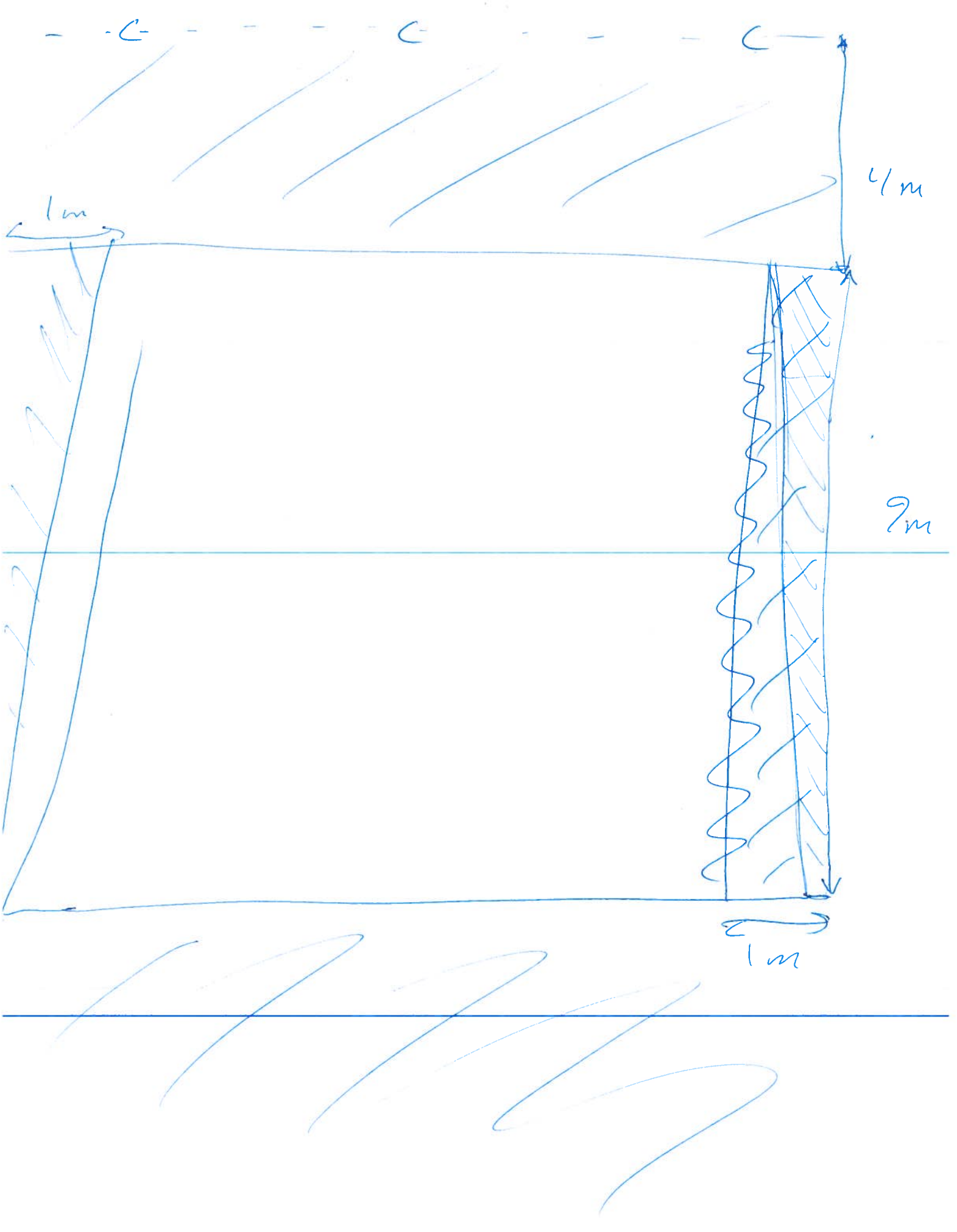
$\alpha_2 = 46.8°$



456
- 180
= 276

$\alpha_3 = 46.8°$

K4 DEAD ZONE

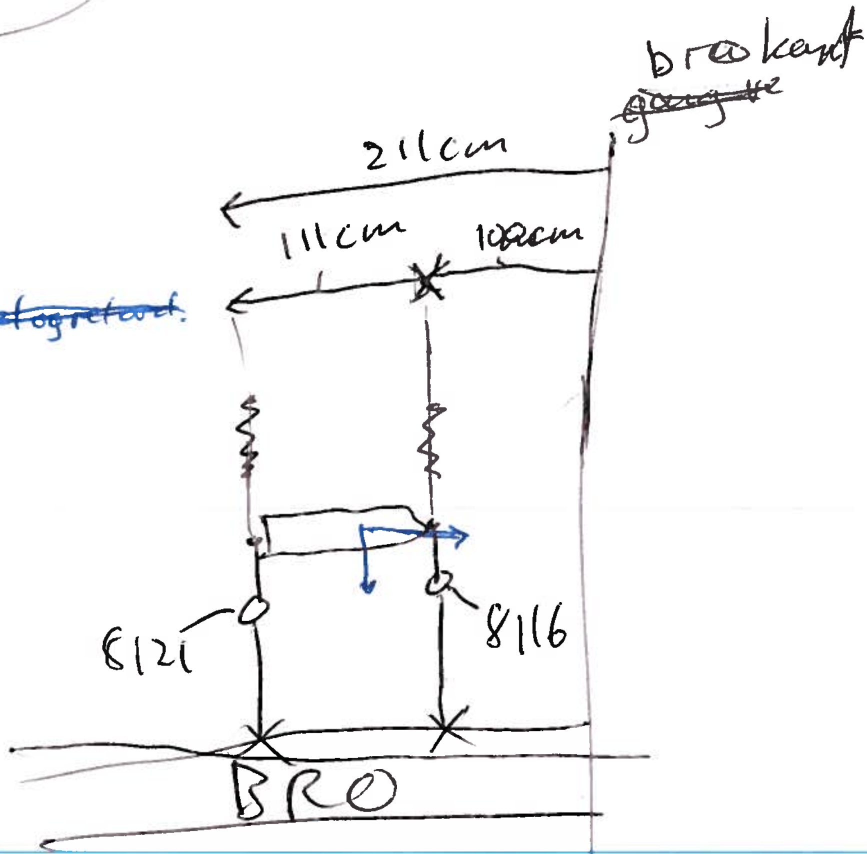


Appendix F

Towing 0 and 90 degrees

VOGN

ved høye taue hastigheter
med 90° ~~med~~ taue hastighet
retning ift heading 90°
~~Begynner skipet~~
vil skipet bevege
seg i surge retning
pga skrogutforming.
ved ca 0.45 m/s
~~begynner~~ kan
man se at den
osillerer i surge
retning når vognen
stopper.



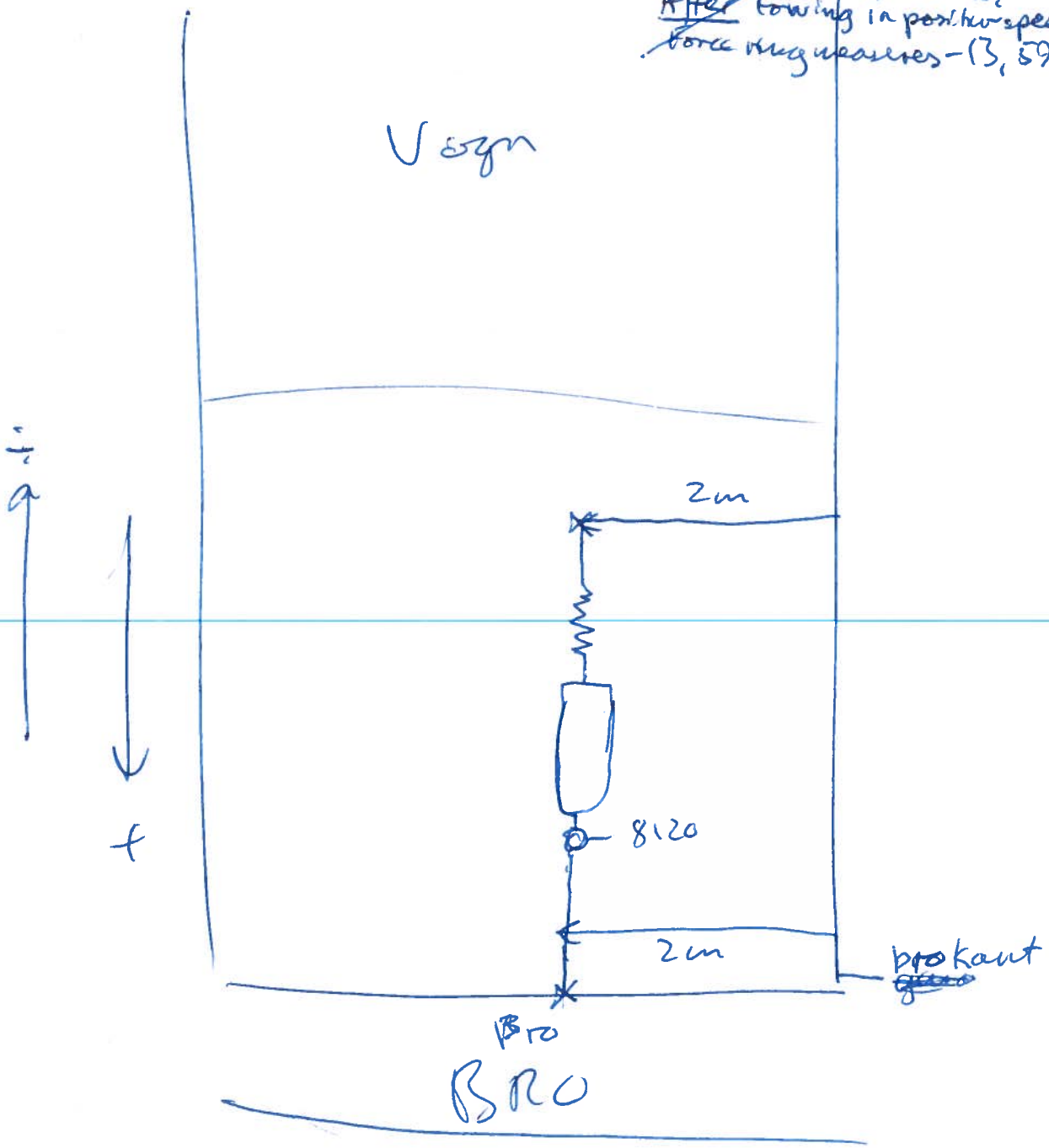
det gjør målingene
mer unøyaktig
fordi det
legger til mer
støy og kraftingen
får en vinkel ift surge
og vil da kunne
få inn noe av surgebevegelsene.

taue vogn
max akselerasjon 0.50 m/s² min 0.0 m/s² S-kurve [%] = 30
maks hastighet 2.00 m/s min 0.0 m/s

± 0.6 m/s osillerer ved tauing.

0 ⑨

Shift in zero measurement
When at the far end (further from MC WINDOW) after towing in neg speed direction
Forcering measures -13,55... to -13,54...
When at the close end (MC Room)
~~Forcering measures 13,~~
After towing in pos. tow speed direction
force neg measures -13,59... to -13,58



~~0,23 -> 0,24~~

kraft ranger tråler max 50 [N]

0.01 -> 0.06 enkeltvei
~~0.02 -> 0.0~~
~~opp til~~ 0.07 -> 0,10 enkeltvei

0.18 -> 0.19 enkeltvei
0.20 -> 0.21 -"-
~~1 stekte per~~

Appendix G

Measurements notes

MyComputer > (C:) > Carl Man Projects > Main > Data

MEG

> MGC Reg → MGCReg 9.0

Først I/O - Definitions

→ Definerer hvilken kanaler Forsterkeren skal bruke ift kraftmåler

3 kraftmåler + X pos. Stekk

Sample time sat til 2000 Hz

Kraftsignene tåler bare 50 N

> "Zero settings"

trykk "START" la gå i ca 30 sek

For hver runn gjøres dette

For målinger "Measurement" → trykk "Setup"

"Start" begynner målinger

Stop stopper målinger

Inn hver gang "Setup" blir trykket reuses alt tidligere data

Hastighets målaren er darling

Så det er bedre at hastigheten notees : Hl warm.

Kjøre vogn: ~~kavel~~ ~~Hovedmeny~~ "F1"

~~Konfig~~

HOVEDMENY "F1" trykk inntil den får det som
så
KONFIG. AV AKSER "F3"
→ AKSEKONFIG.

X-AKSE "F3"

Sett inn aks + hastighet

~~AKTIV~~ HA HOLDEK. JOG X AKTIVERT
"F4"

~~HOVEDMENY "F1"~~

→ AKTIVISERING AV AKSER "F4"

SJEKK AT X ER AKTIVERT "AV/PÅ" "F3"

VED HOLDEK. JOG X BEVEGER DEN SEG
I DEN RETNING ~~HVILE~~ INNTIL DU TRYKKER
AT DEN SKAL BEVEGE SEG I MOTSATT RETNING