



BACHELOR THESIS & COLLOQUIUM - ME141502

*DEVELOPMENT OF A ROUTINE FOR THE TRANSMISSION OF STATIC
PARAMETERS OF A SHIP MODEL INTO A SYSTEM FOR THE
SIMULATION OF DYNAMIC POSITIONING*

RIKA CITRA
NRP. 04211441000006

SUPERVISOR :
Prof. Dr.-Ing. Matthias Markert
Dr.-Ing. Bettina Kutschera

DOUBLE DEGREE PROGRAM
DEPARTMENT OF MARINE ENGINEERING
FACULTY OF MARINE TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

“This Page Intentionally Left Blank”



BACHELOR THESIS & COLLOQUIUM-ME 141502

**DEVELOPMENT OF A ROUTINE FOR THE TRANSMISSION OF STATIC
PARAMETERS OF A SHIP MODEL INTO A SYSTEM FOR THE SIMULATION
OF DYNAMIC POSITIONING**

**RIKA CITRA
NRP. 04211441000006**

**SUPERVISOR :
Prof. Dr.-Ing. Matthias Markert
Dr.-Ing. Bettina Kutschera**

**DOUBLE DEGREE PROGRAM
DEPARTMENT OF MARINE ENGINEERING
FACULTY OF MARINE TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018**

“This Page Intentionally Left Blank”



SKRIPSI-ME 141502

**PENGEMBANGAN SEBUAH RUTINITAS UNTUK TRANSMISI PARAMETER
STATIS PADA MODEL KAPAL KE SYSTEM UNTUK SIMULASI DYNAMIC
POSITIONING**

**RIKA CITRA
NRP. 04211441000006**

**SUPERVISOR :
Prof. Dr.-Ing. Matthias Markert
Dr.-Ing. Bettina Kutschera**

**DOUBLE DEGREE PROGRAM
DEPARTMENT OF MARINE ENGINEERING
FACULTY OF MARINE TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018**

“This Page Intentionally Left Blank”

**Development of A Routine for The Transmission of Static Parameters
of a Ship Model into A System for The Simulation of Dynamic
Positioning**

BACHELOR THESIS

Submitted to Comply One of the Requirement to Obtain a Bachelor of Engineering
on

Department of Marine Engineering

Faculty of Marine Technology

Institut Teknologi Sepuluh Nopember

Surabaya

Prepared By

RIKA CITRA

NRP. 04211441000006

Approved By 1st Supervisor and 2nd Supervisor:

1. Prof. Dr.-Ing Matthias Markert

(*Dr. Markert*)

2. Dr.-Ing. Bettina Kutschera

(*B. Kutschera*)

Warnemünde,

July 2018

“This Page Intentionally Left Blank”

APPROVAL FORM

DEVELOPMENT OF A ROUTINE FOR THE TRANSMISSION OF STATIC PARAMETERS OF A SHIP MODEL INTO A SYSTEM FOR THE SIMULATION OF DYNAMIC POSITIONING

BACHELOR THESIS

Submitted to Comply One of The Requirement to Obtain a Bachelor
Engineering Degree

on

Bachelor Program Department of Marine Engineering
Faculty of Marine Technology
Institut Teknologi Sepuluh Nopember

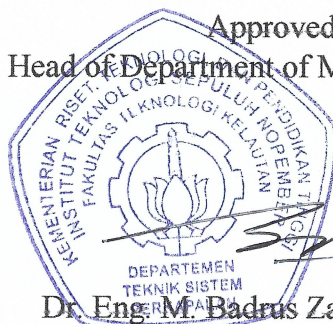
Prepare by :

RIKA CITRA

NRP. 04211441000006

Approved by

Head of Department of Marine Engineering



Dr. Eng. M. Badrus Zaman., ST., MT.

NIP. 197708022008011007

“This Page Intentionally Left Blank”

APPROVAL FORM

DEVELOPMENT OF A ROUTINE FOR THE TRANSMISSION OF STATIC PARAMETERS OF A SHIP MODEL INTO A SYSTEM FOR THE SIMULATION OF DYNAMIC POSITIONING BACHELOR THESIS

Submitted to Comply One of The Requirement to Obtain a Bachelor
Engineering Degree

on

Bachelor Program Department of Marine Engineering
Faculty of Marine Technology
Institut Teknologi Sepuluh Nopember

Prepare by :

RIKA CITRA

NRP. 04211441000006

Approved by

Representative of Hochschule Wismar in Indonesia

Dr.-Ing. Wolfgang Busse

“This Page Intentionally Left Blank”

DECLARATION OF HONOR

I hereby who signed below declare that :

This bachelor thesis has written and developed independently without any plagiarism act, and confirm consciously that all data, concepts, design, references, and material in this report own by Department of Marine Engineering ITS which are the product of research study and reserve the right to use for further research study and its development.

Name : Rika Citra

NRP : 04211441000006

Bachelor Thesis Title : Development of A Routine for The Transmission of Static Parameters of a Ship Model into A System for The Simulation of Dynamic Positioning

Department : Marine Engineering

If there is plagiarism act in the future, I will fully responsible and receive the penalty given by ITS according to the regulation applied.

Rostock, July 2018

Dimas Darmawan

“This Page Intentionally Left Blank”

Assignment for Bachelor Thesis

Student: Rika Citra
Subject: Development of a routine for the transmission of static parameters of a ship model into a system for the simulation of Dynamic Positioning

Supervising Professor:	Prof. Dr.-Ing. Markert	Hochschule Wismar
Assistant Supervisor:	Dr.-Ing. Bettina Kutschera	DNV GL

Date of issue: .2018
Filing date: .2018

Task

To adapt a dynamic positioning simulation system to a specific ship model, its settings should be transferred. For this purpose, all static parameters of the ship model are to be transferred from the ship handling simulator to the existing DP system. The transfer of the data and its control should be realized by MatLab and Simulink functions or procedures.

The following aspects should be considered in particular:

1. Analysis of the information inside the parameters, needed from the simulator.
2. Development of interface functionality for the data exchange in the stages:
 - Load in the parameter file directly in MatLab.
 - Read in the parameter file via UDP between two IPCs.
3. Prepare MatLab programs for the execution and control of the data exchange.
4. Test the program for data exchange in MatLab and Simulink.

The supervising Professor reserves the right to widen or narrow down the scope of the task as he sees fit while it is being processed. Contacts with other institutions and companies may only be established in agreement with the supervisors. The publication of the work or parts of it requires prior permission of the supervisors. The work shall be prepared in accordance with the applicable guidelines of Hochschule Wismar for academic and scientific work.

At least two consultations with the supervising professor are required as part of the processing.

The final version of the thesis is to be submitted in a generally accepted electronic format (such as PDF or similar) and in four printed copies at the Organization Office in Rostock-Warnemünde, Germany.

Prof. Dr.-Ing. M. Markert

Abstract

This bachelor thesis focus is transfer a data between a ship handling simulator to the I-PC via UDP. This project using Matlab to create some code that able to read a data from the simulator. The first step is to sorted a static parameter from the *.dat files and type it one by one in Matlab editor. Adding some used constants and missing variable to the editor. The next step is creating some codes with necessary functions. The code created so the users can open and select any data from the computer and read the selected data automatically from the file. Then creating a new code that will only process the certain variable with for-loop and the result have to be the same with manually code. In this project, there are six types of ships from the simulator. Four ships without POD-units thruster and two with POD-units thruster. This project has not finish yet because in the latest step of this project meet an obstacle as the result of matrix data does not came out as expected. but the actual result it is showed only one of the available value of the loop.

Keywords: Ship Handling Simulator, Matlab, Simulink, Transfer Data

Abstrak

Tugas akhir ini fokus pada data transfer antara simulator kapal dan I-PC via udp. Proyek ini menggunakan Matlab untuk membuat koding yang dapat membaca data dari simulator kapal. Tahap awal pada tugas akhir ini yaitu dengan mensortir static parameters dari file *.dat dan menuliskannya pada editor Matlab satu per satu. Langkah selanjutnya dengan membuat kosing dengan function yang dibutuhkan. Koding ini dibuat agar user dapat membuka dan memilih data/file dari computer dan membaca data dari file yang telah dipilih secara otomatis. Kemudian membuat koding baru yang hanya akan memproses variabel tertentu dengan for-loop function dan hasilnya akan sama dengan kode yang dibuat secara manual. Pada proyek ini, ada enam tipe kapal dari simulator. Empat kapal tanpa POD-units thruster dan dua dengan POD-units thruster. Proyek ini belum dikatakan selesai karena pada tahap terakhir ditemukan sebuah hambatan dimana hasil dari matrix data tidak sesuai dengan ekspektasi tetapi hasil yang muncul hanya menunjukkan satu dari banyaknya loop value.

Kata kunci: Ship Handling Simulator, Matlab, Simulink, Transfer Data

Preface

This Bachelor thesis is my completion of marine engineering program at Institut Teknologi Sepuluh Nopember Surabaya. This Bachelor thesis has been made at the Department of Maritime Study, Faculty of Engineering Science, Hochschule Wismar, Rostock, during Summer semester 2018.

First of all, I would like to say thank you for Allah SWT, because of Him, I am able to do this research in Hochschule Wismar in Rostock, Germany. I would like to say to all the people who help me with this bachelor thesis. My supervisor Prof. Dr.-Ing. Matthias Markert who gave me this opportunity to do my Bachelor thesis in Rostock, Germany and also my second supervisor Dr.-Ing. Bettina Kutschera who gave me a lot of knowledge and advice on this project. My friends who came together for also doing their Bachelor thesis and all master degree students from Indonesia, thanks for all the support, discussions, advices and help for this thesis.

I would like to say thank you for my family, who always support me no matter where I am. Thank you for the lectures of marine engineering program at Institut Teknologi Sepuluh Nopember Surabaya for all the knowledge and advice for this four years of my study.

It has been such an honor and a great experience to able doing my Bachelor thesis in Rostock, Germany for one semester.

Rostock, 06 July 2018

Rika Citra

Table of Content

APPROVAL FORM	vii
APPROVAL FORM	ix
APPROVAL FORM	xi
DECLARATION OF HONOR	xiii
Assignment for Bachelor Thesis	xv
Abstract.....	xvii
Abstrak.....	xviii
Preface.....	xix
Table of Content.....	xx
List of Table and Figures.....	xxii
Abbreviations	xxiii
1 Introduction.....	1
1.1 Backgrounds	1
1.2 Research Problems.....	2
1.3 Research Limitations	2
1.4 Research Objectives	2
1.5 Research Benefits	3
2 Theoretical Background	5
2.1 Ship Simulator	5
2.2 Dynamic Positioning System	6
2.2.1 Dynamic Positioning Controller	9
2.3 Matlab	10
2.4 Simulink	13
2.4.1 xPC Target	15
2.5 Industrial PC	17
2.6 User Datagram Protocol (UDP)	18
2.7 BIOS	20
3 Methodology	21
3.1 Schematic Diagram of Works	21
3.2 Detail of Works.....	22
3.2.1 Problem Identification.....	22
3.2.2 Literature Study.....	22
3.2.3 Data Collecting.....	22
3.2.4 Developing the Program.....	22

3.2.5	Software Simulation	22
3.2.6	Conclusion and Suggestion	23
4	Analysis	25
4.1	Static Parameters from The Simulator	25
4.2	Missing Variable from The Data	31
4.3	Developingthe Data Interface for Data Transfer	32
4.4	Analysis for The Error Occur in The Transfer Data.....	41
5	Result and Conclusion	43
5.1	Result.....	43
5.2	Conclusion	43
	Bibliography	45
	Appendix 1: Static Parameters	47
	Appendix 2: 0000_FFFF_1_1_1_Baltic-2013_Container_obj_40_597.dat ..	53
	Appendix 3: 0000_TTFF_2_2_2_MS-Europa_Passenger_obj_40_474.dat (with POD-Thruster).....	63
	Appendix 4:The following constants in a special condition that depends on the value of Indential_PropUnits and IdenticalRudderSystems.....	75
	Appendix 5: Code for Reading and Transfer Data Automatically	77
	Appendix 6: Example of Vector Data.....	83
	Appendix 7: Example of Matrix Data	85

List of Table and Figures

Figure 2.1 Ship Handling Simulator	5
Figure 2.2 Dynamic Positioning Basic Principle	6
Figure 2.3 Schematic Drawing of Dynamic Positioning	7
Figure 2.4 Dynamic Positioning System.....	8
Figure 2.5 Dynamic Positioning Controller	9
Figure 2.6 Structure of the controller	10
Figure 2.7 Matlab Editor	11
Figure 2.8 Matlab workspace	12
Figure 2.9 Simulink Workspace.....	14
Figure 2.10 Simulink Block Diagram	15
Figure 2.11 xPC Target.....	16
Figure 2.12 Industrial PC	18
Figure 2.13 UDP Mechanism.....	19
Figure 3.1 Methodology Flowchart	21
Figure 4.1 List of complete variables.....	26
Figure 4.2 The result of manually codes data (1)	28
Figure 4.3 The result of manually codes data (2)	29
Figure 4.4 Choosing the file.....	33
Figure 4.5 The result of data reading	34
Figure 4.6 The result of automatically codes on vector data type	37
Figure 4.7 The result of automatically codes on Matrix data type	38
Figure 4.8 Workspace result	41
Table 4.1 Table of Missing Value.....	31

Abbreviations

BIOS	:Basic Input/ Output System
CPU	:Central Processing Unit
RAM	:Random-Access Memory
CMOS	:Configuration Memory Operating System
LTE	:Long-Term Evolution
FPGA	:Field-Programmable Gate Array
ACIS	:Application-Specific Integrated Circuit

“This Page Intentionally Left Blank”

1 Introduction

1.1 Backgrounds

Dynamic positioning system has been used for 58 years since its inception, with the growth of oil industry. And today Dynamic positioning has been used on varied application, one of them is installed on the vessel. This technology has found its way into all aspects of the marine industry.

Some operation at sea require that marine vessels has to keep constant position and heading. It can be done by mooring if conditions allow. DP is defined by the (IMCA, 2003), (International Marine Contractors Association) as: “A system which automatically controls a vessel’s position and heading exclusively by means of active thrust”.

Many marine working ships have been equipped with dynamic positioning system (DPS) to maintain stable working conditions. DPS consisting of position measurement system, control system and thruster system.

Ship simulator are the facilities and an important instrument in the research of ship control and they provide a real situation of the ship condition and situation with various model of ships. This can be used for training of Dynamic Positioning Operators, for which a ship simulator must be used. The simulator instructor will set the condition of the sea and the people who train will try to operate the ship with the condition of the operator has set. With these features there are a lot of parameters that considered for controlling the ship simulator.

Ship simulator has more than 100 ships with different characteristics. And every type of ships has their own condition which define their motion when sailing and when keeping a predefined position. With the same wind force and the wave, the impacts are different between the ships model.

One of the main topic of this bachelor thesis is ship handling. Ship handling is the art of proper control of the ship while underway, especially in harbors, around the docks and piers. The basic thing to be understood in ship handling is to know and anticipate how a ship behaves under all circumstances.

Basically, the ship handling simulator is a facility that provide a real situation of handling the ship on the sea with different types of ship and different conditions. Here trainees can train even dangerous situations without risking damages for

persons, ship or environment. Another main advantage of simulators is that same situations can be repeated several times for training purposes.

By create the bridge between the ship handling simulator with dynamic positioning system, we can know the ship position and movement based on the data input and the ship types in the simulator.

In this bachelor thesis, Matlab software is used to creating a routine for the transmission of static parameters of a ship model into a system for the simulation of Dynamic Positioning and transfer the data given from the simulation to the industry PC (I-PC).

1.2 Research Problems

Based on background mention above, it can be concluding some problems of this final project is:

- a. How to transfer a data between ship handling simulator and Industry PC (I-PC)?
- b. How to create a code that able to transfer the data manually and automatically from the simulator?

1.3 Research Limitations

This final project can be focused and organized, with limitation on problem which are:

- a. Research object is the data result from Ship Handling Simulator in Warnemünde.
- b. In this research using Matlab program to developed the interface function for data exchange

1.4 Research Objectives

Based on problems mention above, the objectives of this final project are:

- a. To develop a communication between the existing control model on an I-PC (industry PC) and the ship simulator is done via xPC-target.
- b. Able to transfer the parameters to the DP (Dynamic Positioning) Simulator manually and automatically.

1.5 Research Benefits

Based on problems mention above, the Benefits of this final project is:

- a. Provides a software that able to transfer a data from ship handling simulator to dynamic positioning system.
- b. Able to transfer manually and automatically the variable data from the simulator.

“This Page Intentionally Left Blank”

2 Theoretical Background

2.1 Ship Simulator

Simulator is a replication of the operation of a real-world works or process. A simple version of a real vessel's and environmental dynamics. While Ship Handling Simulator provides experience in the basic operation of various types of ships in a various range of weather and sea conditions and in different port call operation. Ship simulator divided by two classifications. First is real-time simulators and fast-time simulators. Real-time simulators are controlled by human while the fast-time simulator is controlled by computer or it can be referred as an autopilot. There is some example of maritime simulator such as simulator for engine room, cargo handling operation, shore side operation, and Electronic Chart Display and Information System (ECDIS).

The simulator provides experience of basic operation of various types of ships and permit the trainees to train in different techniques. The simulator helps trainees to understand each of vessel characteristic behavior patterns. Simulator also given by maritime schools and academies as part of basic training.



Figure 2.1 Ship Handling Simulator [1]

<https://www.prescient.com.sg/wp-content/uploads/2014/01/header020.png>

The size of the simulator itself depends on the purpose. Range from a normal PC size to the actual size of the full-size bridge. The full mission bridge simulators, there are a realistic vessel's bridge simulator and control console, screen with 360-degree view of ship's environment.

The simulator consists a software that give a realistic condition of the ship and also the environment that replicate the vessel dynamic behavior and the system of the vessel. The simulator also replicates the maritime environment so the person can control and relate with simulated surroundings.

Ship simulators have to be manually prepared depends on the exercise goal. The user's goal from practicing with the simulator that users can understand the effect of various situation and condition on the ship's maneuverability such as wind, current, shallow water, river bank, loaded and ballast condition, increasing the efficiency of the bridge by understanding of bridge management principle, knowing how to handle the navigation emergency situation. To archive all of the goal, the preparation of the exercise includes the selection of the sea area, environmental parameters such as day time, wind, wave height, etc. The simulators can be used for various maritime application scenarios.

2.2 Dynamic Positioning System

Certain operations at sea require that marine vessels keep constant position and heading. This can be done by mooring if condition allow. In such cases, Dynamic Positioning System (DPS) is an appropriate method for maintain the position of the vessel. Figure 2 shows the basic principle of ship movement.

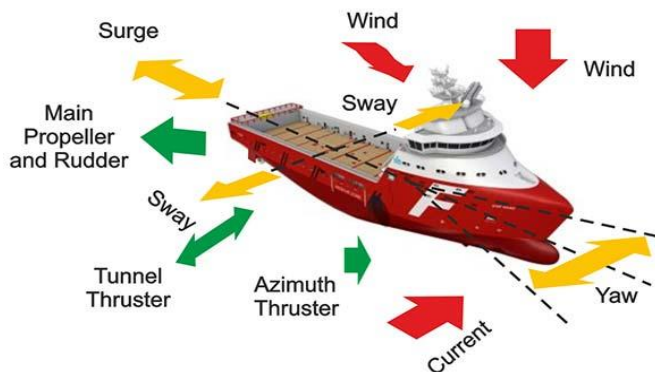


Figure 2.2 Dynamic Positioning Basic Principle [2]

<http://www.offshoreengineering.com/images/Courses/Dynamic-Positioning/what-is-dynamic-positioning.jpg>

Dynamic Positioning System (DPS) can be defined as a computer-controlled system designed to automatically maintain a vessel's position and heading by using its own propellers and thrusters [3]. It is needed by vessel that need to be dynamically positioned. Examples of vessel types that used a DPS such as ships and semi-submersible mobile offshore drilling units (MODU), oceanographic research vessels, cable layer ships and cruise ships. The computer program

contains a mathematical model of the vessel that includes information of the wind and current drag of the vessel and the location of the thrusters.

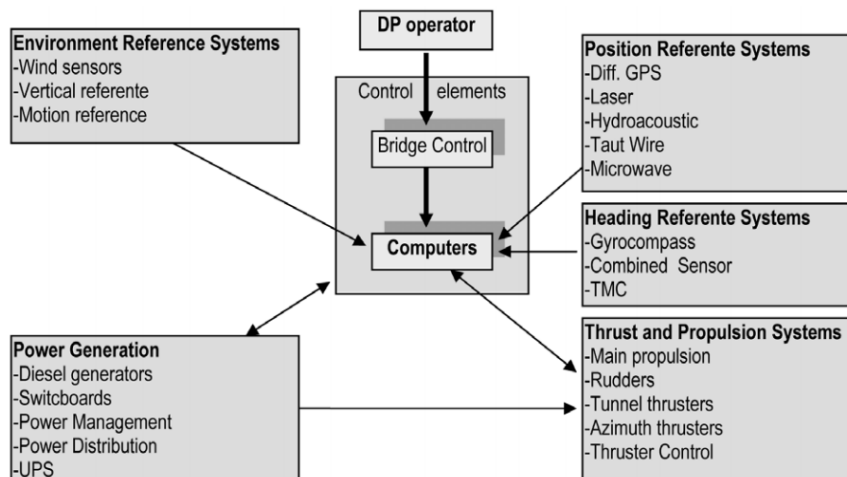


Figure 2.3 Schematic Drawing of Dynamic Positioning [3]

“Introduction to ship dynamic positioning systems”, by C. S. Chas and R. Ferreiro, 2008, Journal of Maritime Research, Vol. V. No. 1, pp. 80. Copyright 2008 by SEECMAR

On the ship there is a need for sensor to measure the velocities and direction of the wind, waves and current. The power and direction of the thrusters will be calculated with a mathematical model to stay course. The ship that equipped with dynamic positioning not always have azimuth thrusters, a combination of twin screws at the back with conventional rudder blades plus a bow and/or stern thrusters also allows for straight sideways movements. There are two Dynamic Positioning System available, first is based on PID regulator and the other one is based on model control. While the first type can only correct after the ship gets off course, the model control type can predict deviation.

A DPS is a complex system composed of several sensors, control and filtering algorithms, and propellers. The sensors are used to measure the position and heading of the floating vessel either as absolute or as relative values, while the algorithms are responsible for calculating the forces to be delivered by each propeller to counteract environmental forces, such as wind, waves, and current loads.

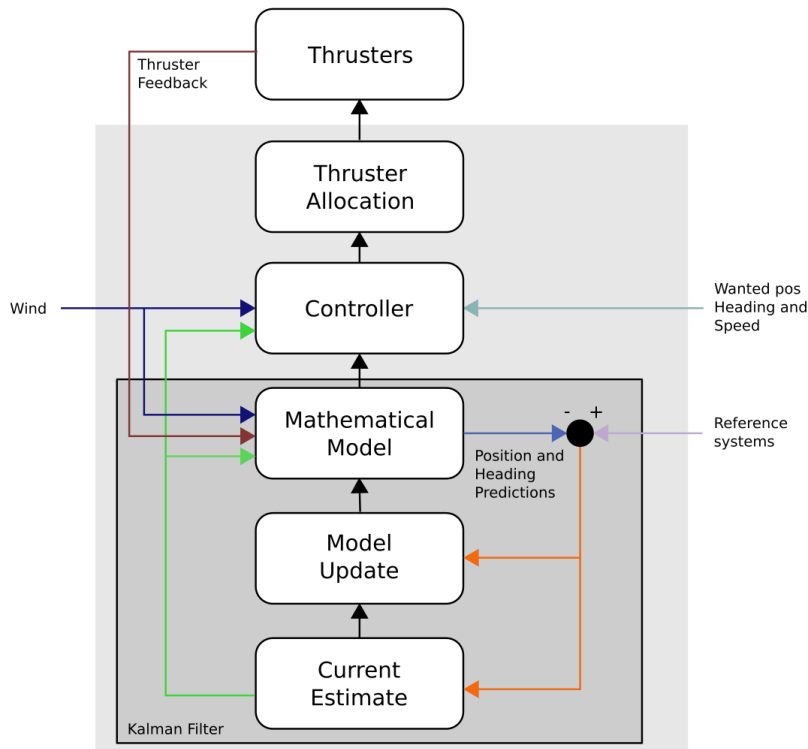


Figure 2.4 Dynamic Positioning System [4]
*“Dynamic Positioning Simulator Interim Report” by Jalitha Wills, 2007,
 Rotterdam, TU Delft.*

Based on *Dynamic Positioning Simulator Interim Report* by Jalitha Wills, Dynamic positioning (DP) has many possible operation modes [4], such as:

- Manual or joystick mode, where the operator has full control of the vessel;
- Auto-Heading mode, where the system keeps the required heading automatically;
- Auto-Position mode, where the system keeps the required position automatically;
- Auto area position mode, the system maintains the specified area automatically while using minimum power;
- Auto track mode, where the vessel follows a specified track described by a set of way points;
- Autopilot mode, where the vessel steer automatically along a pre-defined course;
- Follow target mode; where the vessel follows the constantly moving target.

2.2.1 Dynamic Positioning Controller

Dynamic positioning controller is consisting of display, buttons and the joystick. Performs with close loop control. All operator commands and information from sensors will be interpreted and it will provide the correct signal for the system.

Dynamic positioning system controller able to calculates the result force by the thruster or the propellers in able the ship to remain on the position.

Based on Kongsberg dynamic positioning system controller, there are several of the following modes with special characteristics [5]:

- **High precision control**
This characteristic provides high accuracy station-keeping in any weather condition at the expense of power consumption and exposure to wear and tear machinery and thruster.
- **Relaxed control**
This control used to control the thruster more smoothly with the expense of station-keeping accuracy.
- **Green DP control**
Kongsberg DP control developed a GreenDP control which is unique dynamic positioning to reduces fuel consumption and CO2 emission by as much as 20 percent.



Figure 2.5 Dynamic Positioning Controller [6]

Abschlussbericht für das FuE-Verbundvorhaben: Entwicklung eines schiffstypenunabhängigen adaptiven regelungstechnischen Kerns für eine Simulation von Dynamischen Positionier-Systemen (Kurztitel DP-SIM) P.12-16 by Bernhardt, F. & Kutschera, B., 2015, Germany



Figure 2.6 Structure of the controller[6]

Abschlussbericht für das FuE-Verbundvorhaben: Entwicklung eines schiffstypenunabhängigen adaptiven regelungstechnischen Kerns für eine Simulation von Dynamischen Positionier-Systemen (Kurztitel DP-SIM) P.12-16 by Bernhardt, F. & Kutschera, B., 2015, Germany

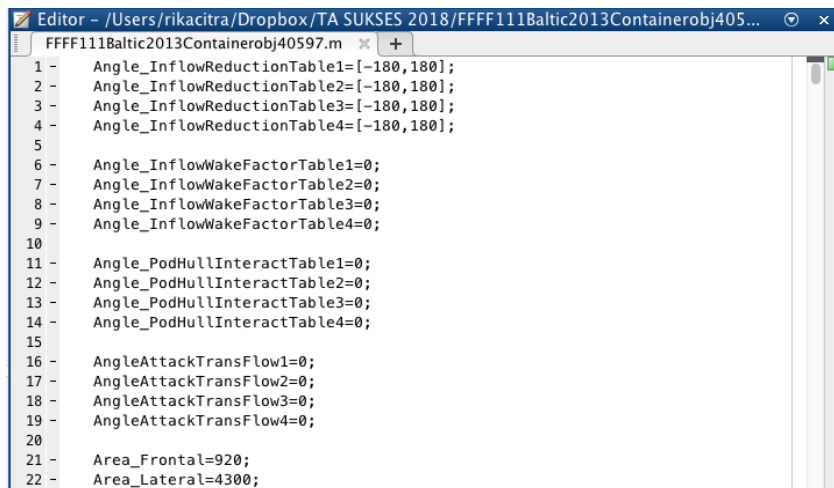
2.3 Matlab

Matlab is a program for technical computing, that usually used by the engineer or the scientist. Where Matlab offers an easy-used environment to solves a problem and provides a solution and showing it in mathematical notation. Matlab itself is a proprietary programming language developed by Mathworks. Matlab provides a lot of functions. This program allows matrix manipulations, plotting of function and data, implementation of algorithms, creating user interfaces and interfacing with programs written in other languages.

As Francisco Javier Campos from AT4 wireless said “MATLAB is a universal language that makes it easy to exchange algorithms and test results across our team. Our physical layer model in MATLAB and Simulink enabled us to better understand the LTE specifications, and Model-Based Design enabled us to verify that our FPGA implementation conformed to those specifications.”

That a lot of uses of Matlab include:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building



```
Editor - /Users/rikacitra/Dropbox/TA SUKSES 2018/FFFF111Baltic2013Containerobj405...
FFFF111Baltic2013Containerobj40597.m
1 - Angle_InflowReductionTable1=[-180,180];
2 - Angle_InflowReductionTable2=[-180,180];
3 - Angle_InflowReductionTable3=[-180,180];
4 - Angle_InflowReductionTable4=[-180,180];
5
6 - Angle_InflowWakeFactorTable1=0;
7 - Angle_InflowWakeFactorTable2=0;
8 - Angle_InflowWakeFactorTable3=0;
9 - Angle_InflowWakeFactorTable4=0;
10
11 - Angle_PodHullInteractTable1=0;
12 - Angle_PodHullInteractTable2=0;
13 - Angle_PodHullInteractTable3=0;
14 - Angle_PodHullInteractTable4=0;
15
16 - AngleAttackTransFlow1=0;
17 - AngleAttackTransFlow2=0;
18 - AngleAttackTransFlow3=0;
19 - AngleAttackTransFlow4=0;
20
21 - Area_Frontal=920;
22 - Area_Lateral=4300;
```

*Figure 2.7 Matlab Editor
Personal Document*

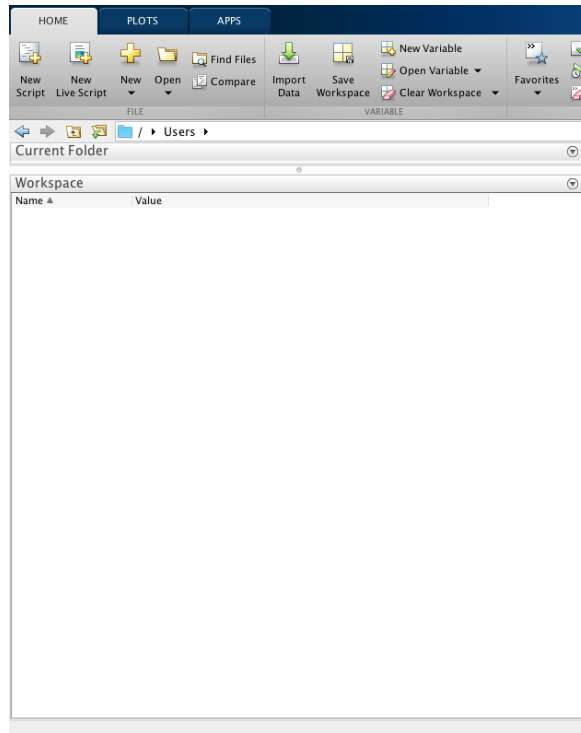


Figure 2.8 Matlab workspace
Personal Document

Matlab system consist of five main parts [7]:

1. The Matlab language
This Matlab system language with control flow statements, data structures, functions, input/output, and object-oriented programming features.
2. The Matlab working environment
This is a set of tools and facilities for Matlab users and programmers. This system provides users a facility to manage a variable on a workspace and also allow to importing and exporting a data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.
3. Handle graphics
This system provides a high-lever command for two-dimensional and three-dimensional graphics, image processing, animation and presentation graphics. It also allowed users to customize the graphics appearance and build a complete Graphical User Interfaces.

4. Mathematical function library

This system is a collection of computational algorithm such as sum, sine, cosine, and other complex arithmetic. This system also has a collection of more complex function such as matrix invers, matrix eigenvalues, Bessel function and fast Fourier transform.

5. Application Program Interface (API)

This system allowed users to write C and Fortran programs. This system provides a facility for calling routines from Matlab (dynamic linking), Calling Matlab as computational engine, and for reading and writing MAT-files.

For this thesis, Matlab is used for develop the interface data and also to design the control system by Simulink.

To work on this project, to make a connection between industry PC to the ship simulator is done via xPC-target.

xPC target is a solution that allows to archive real-time performance of the Simulink model on independent “target” computers. xPC target can archive sample rates approaching 50 Hz, depending on processor performance level, model size, and Input/Output complexity.

2.4 Simulink

Simulink is a program for simulation and model-based design for dynamic and embedded system. Developed by Mathworks as an add-on in Matlab. Simulink provides an interactive graphical design and customizable set of block libraries that allows user to design, simulate, control, implement, and test varies of time varying system such as communications, controls, signal processing, video processing, and image processing.

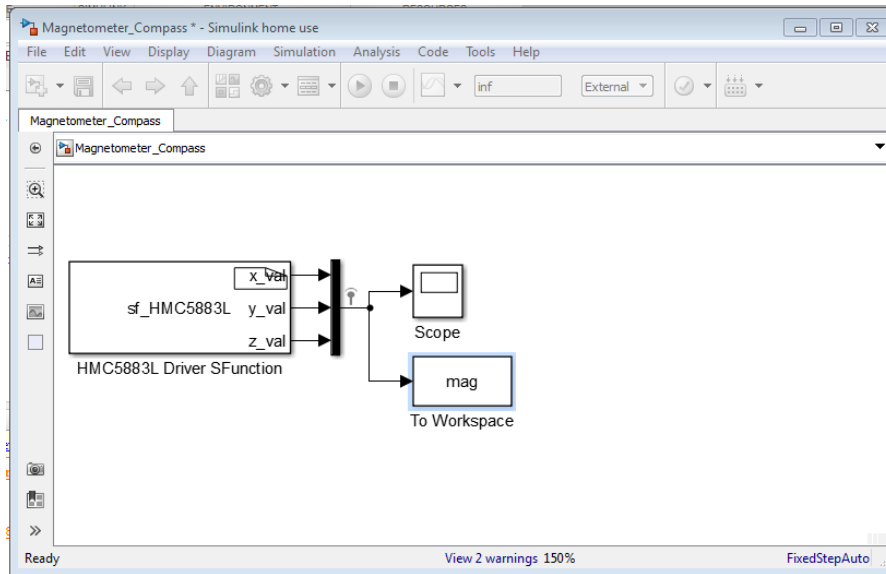


Figure 2.9 Simulink Workspace[8]

https://www.mathworks.com/matlabcentral/answers/uploaded_files/41017/model.PNG

By using Simulink, users can test a system without creating or buying an expensive prototype and also saving time and less risky. There are some project models that can be done in Simulink [9], such as:

- **Wireless communication**
This model allowed users to create a quick model manipulate signals, transforms, and filter. And also verify compliance to standards like LTE.
- **Motor and power control**
This model used in electric vehicles, renewable energy and industrial automation. By allowed users to create a system level model for electric motor, power converters, and battery storage system.
- **Control system**
Users used this model for verify a control design and generating a code for prototyping and production automatically.
- **Signal processing**
This model allowed users to create a model and simulate a digital signal processing system. Providing a rage of test signal, waveforms, filters types and architectures and scopes for dynamic visualization.
- **Robotics**
Users can design and creating a controller by using blocks for ground vehicles, manipulator, ROS access, and collecting and analyzing sensor

data. This model also allowed users to run the algorithm and automatically generate the code on the hardware.

- Advance driver assistance systems
This model is used for creating and simulate the vehicle and environment and perform sensor fusion and control development.
- Image processing and computer vision
A hardware model done the simulation by test benches using frame-to-pixel stream conversion and vision algorithm designed for FPGAs and ASICs.
- Internet of things
Users can createand test a smart device and deploy the model to edge nodes on hardware.

Simulink also provides a way to create a new model in contrast to text based-programming language.

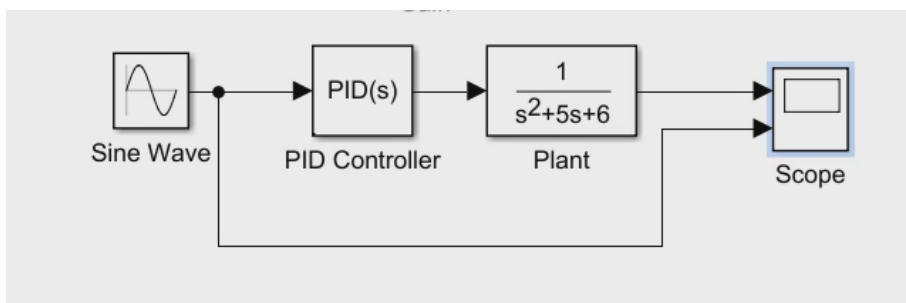


Figure 2.10 Simulink Block Diagram [10]

https://www.mathworks.com/content/dam/mathworks/videos/g/Getting-Started-with-Simulink-Part-2-Adding-a-Controller-and-Plant-to-the-Simulink-Model.mp4/_jcr_content/renditions/PArt_2_thumb.png

2.4.1 xPC Target

xPC Target is one of many solutions for prototyping, testing, and deploying real-time system using standard PC hardware. Using a PC, separated from a host PC for running real-time application as their environment. For this environment, users can use desktop pc as the host with Matlab, Simulink, and Stateflow.

xPC target enables to connect Simulink and Stateflow model to physical system and execute them in real time on PC-compatible hardware.

xPC Target allows you to use PC-compatible hardware with Intel, AMD, and other x86-compatible CPUs as the real-time target PC such as desktop computer,

a rack-mount or industrial computer, a PC/104 or PC/104+, CompactPCI, or an all-in-one embedded PC computer.

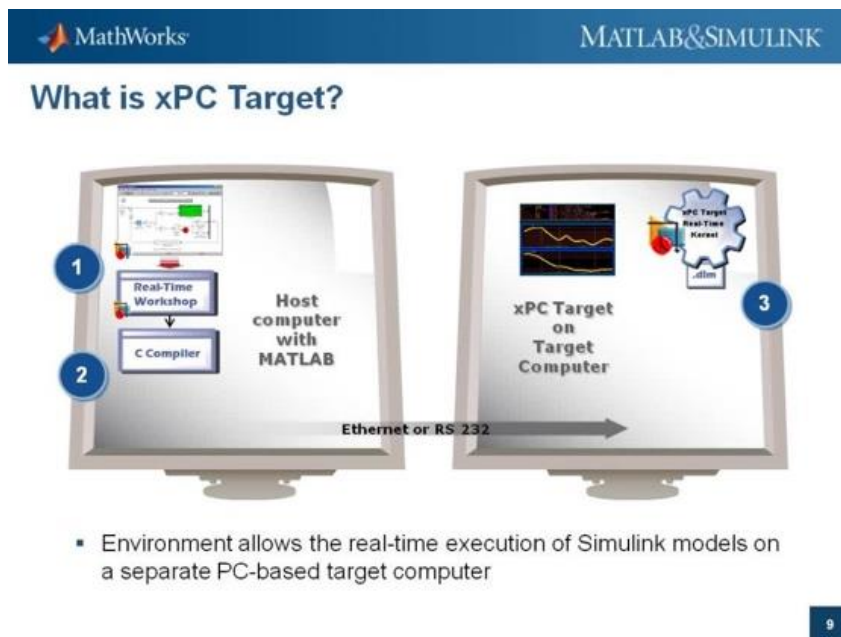


Figure 2.11 xPC Target [11]

<https://www.mathworks.com/videos/introduction-to-xpc-target-and-xpc-target-turnkey-68908.html>

xPC Target allows the user to use your Simulink and State flow models further into your design process by:

- Providing real-time target and I/O capabilities on any PC-compatible system
- Eliminating the need to customize or write any code

By using xPC Target, using a high performance mainstream environment the user can solve many specific rapid controller prototyping applications or hardware-in-the-loop simulations.

There are three requirements for xPC Target [12]:

1. xPc Target needs a special hardware.
xPC Target software requires a host PC, Target PC and for Input/Output target PC also must have Input/Output board supported by xPC Target.

2. xPC target needs a special software.
The xPC Target software requires either a Microsoft Visual C/C++ compiler (version 5.0 or 6.0) or a Watcom C/C++ compiler (version 10.6 or 11.0). In addition, xPC Target requires, MATLAB, Simulink, and Real-Time Workshop.
3. xPC target needs requirements for Embedded option.
The xPC Target Embedded Option is a separate product that requires an additional license from The MathWorks. With this additional license, you can deploy an unlimited number of real-time applications for stand-alone operation.

This option allows you to boot the target PC from an alternate device other than a floppy disk drive such as a hard disk drive or flash memory. It also allows you to create stand-alone applications on the target PC independent from the host PC.

2.5 Industrial PC

Industrial PC is an x86 PC-based computing platform for industrial application also a term of desktop computer. The technical characteristics and features of the IPC is basically the same such as microprocessor and RAM type, storage media, interface port, performance, etc. Industrial PC is a strong system and it is fit for use on the factory. Industrial PC design and manufactured in lower volumes than usual PC.

Industrial PC are specially designed for harsh factory environments such as extremes of temperature, dust, humidity, vibration, and power surges. Industrial PC can be used up to 45°C and over. The Industrial PC cooling fans is designed with special filters to keep out the dust or similar contaminants. Industrial PC also provided with IP/NEMA protection. On such as industrial PC, the Dynamic Positioning Control is running.



Figure 2.12 Industrial PC [13]

SIMATIC Industrial PC SIMATIC Box PC 627, SIEMENS, 2006

2.6 User Datagram Protocol (UDP)

User Datagram Protocol or UDP is part of Internet Protocol. With UDP, Computer applications can send any message in a form of data gram to others computer in other network.

UDP itself an alternative communications protocol to Transmission Control Protocol (TCP) to establishing low-latency and loss tolerating connections between applications on the Internet.

The UDP mechanism is the same with TCP. UDP has a channel that used to connecting between the host to send any information. This channel called port UDP. To able to connected with UDP protocol, first application in computer must be provide an IP address and UDP port number from the host target.

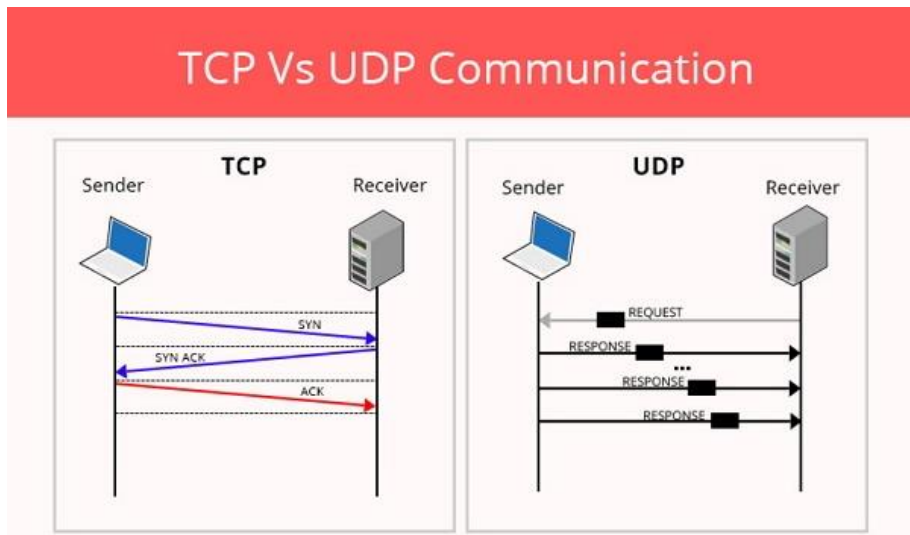


Figure 2.13 UDP Mechanism [14]

<https://www.nesabamedia.com/wp-content/uploads/2017/12/Cara-Kerja-UDP.jpeg>

UDP port is used as multiplexed message queue. It means UDP port able to work with accepting a couple of message at the same time. Every port of UDP has a unique identification number but has its own distribution.

UDP done a simple communication with minimum mechanism. There is a checksum process to keep the data integrity. UDP used for simple communication such as query DNS (Domain Name System), NTP (network Time Protocol), DHCP (Dynamic Host Configuration Protocol), and RIP (Routing Information Protocol).

On query DNS, computer asked some information of a data from a domain to the DNS server. The data can be a web server address, mail server address and the other data that connected with domain. DNS server will reply by giving the information that client asked.

On NTP, client will ask a time information to NTP server. NTP server will reply immediately. So, the client will have a very accurate system.

On DHCP, Client will call DHCP server to ask an IP address to personal used. DHCP will give the IP address information so the users can use the IP address.

On RIP, RIP server will broadcast the routing information to other routers.UDP also fit to send an information that need speed than the accuracy. For example, are audio streaming and video streaming.

2.7 BIOS

BIOS is an acronym for *Basic Input Output System* in IBM PC or compatible computer system (a computer based on the intel x86 processor) refers to a software routine that compatible to doing such as:

1. Initialization (ignition) as well as testing of hardware
2. Load and run operating system
3. Set some basic configuration in the computer
4. Helps the operating system and application in the hardware setup process by using BIOS Runtime System

BIOS provides a low-key communication interface and can control many types of hardware (such as keyboards). BIOS is made by using assembly language used by the machine.

Windows PCs has commonalities of BIOS sequence depending on the computer manufacture. Depends on the computer manufacture, usually BIOS begin with CMOS check for specialize setting for the computer. Then, BIOS will load the computer's device drivers and computer hardware interrupts.

BIOS component information items:

- CPU
- RAM
- Hard Drive
- Optical Drive

3 Methodology

3.1 Schematic Diagram of Works

This is a diagram showed the chart flow of working on this project, the explanation on the flow chart will be explained on point 3.2 on this chapter:

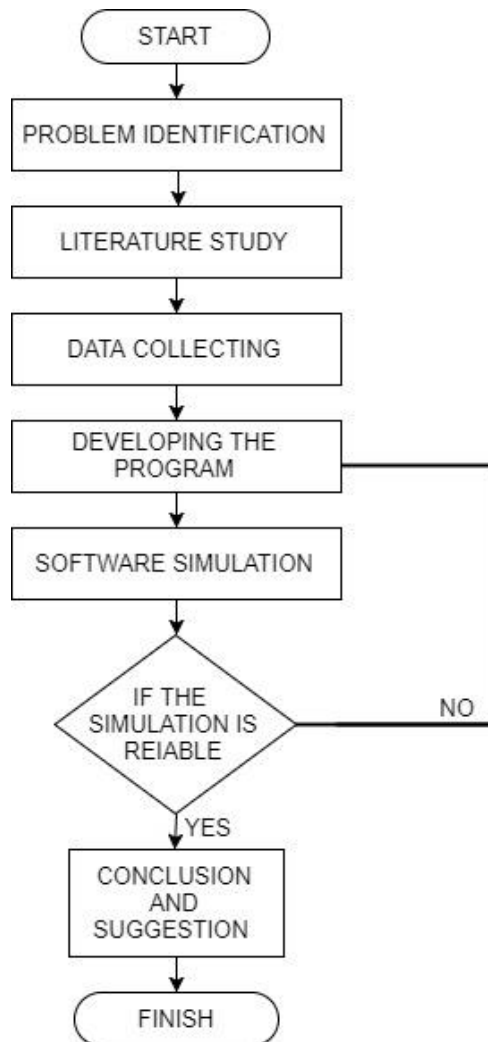


Figure 3.1 Methodology Flowchart
Personal Document

3.2 Detail of Works

3.2.1 Problem Identification

This is the first stage to construct the thesis. In this stage, problems are being identify and questions are being prepared to determine the specific objective of this thesis. To solve the problems and questions of this thesis it will be done by collect some information about the problem.

3.2.2 Literature Study

The first step of this research is literature study. Literature study is performed to explaining the basic theory, general and specific reference, and obtaining various other supporting information relating the final project. Literature study is done by reading papers and journals, related previous thesis, and literature books that related to the topic and supporting this thesis topic.

3.2.3 Data Collecting

The next stage is collecting data. Data is being collected by gather the information from the ship handling simulation. In this bachelor thesis, the data that will be collect is the ship static positioning parameters, which define the simulation of the ships movement based on 100 different values that already exist in the simulator and also the Simulink model for Dynamic Positioning System.

3.2.4 Developing the Program

After collecting data, data is used to develop the interface function. This interface function is used to design an application using a Matlab and Simulink application. The next step, the interface function is created, by using a Matlab program application it can be design. By write the data manually in the Matlab editor and automatically by creating some code on Matlab to sorted the data on the *.dat files and make it possible for Simulink to read the data needed for the simulation. On this stage it the application it will be adjusted depends on the needs to calculating the optimization.

3.2.5 Software Simulation

The last stage of this thesis, after the application is made, the application will be test if the application can transfer the data from ship handling simulator to the dynamic positioning system or not.

3.2.6 Conclusion and Suggestion

At this step as the last step of this research, conclusion and suggestion are given. The aim of this research is to create some program to transfer data between ship handling simulator in Warnemünde and DP (Dynamic Positioning) console.

“This Page Intentionally Left Blank”

4 Analysis

Based on the background of this project, this project is used to developing a routine transmission. There are two types of methods to developing the routines. First by writing and sorting the data manually by doing it in the Matlab editor and second by writing a code to open and the files into Matlab then automatically transfers and sorted the data into the Matlab. This chapter will explain the process of this project.

4.1 Static Parameters from The Simulator

The input signals for the process control divided into static and dynamic data. But of this main focus of this project only the static parameter of the ship's data, so the static parameter must be separated from the dynamic parameter. The static values are needed for the simulation of the ship movement in the navigator ship simulation. List of static parameters for this project are listed in Appendix 1. From the .dat files there are five rows indicates the data from the simulator. For example 14 0 Length 217,5 1.

First row (14) is indicating the ID number of the constant and this ID can change between different *.dat files.

Second row (0) is indicating a sub-ID of the constant.

Third row (Length) is indicating a fix constant name. This is a unique identifier between different *.dat files.

Fourth row (217,5) is a value for the constant.

And the fifth row (1) is indicating the conversion factor. For example, 0514444 stands for a speed given in knots = nautical miles/ hour.

On the fourth row, there is a ****@List@Definition@**** this is indicating that the constant is not only one value but this constant is a vector.

1	BSC2224				
2	1	0	DefaultName	Baltic@2013	1
3	2	0	DefaultCallSign	DQFS	1
4	3	0	C3	n-a	1
5	5	0	RadarShapeType_SubClass	1	1
6	6	0	C141	n-a	1
7	8	0	ImageFile	dsr_baltic.bmp	1
8	9	0	VisualSystemModNo_List	**@List@Definition@**	1
9	9	1	VisualSystemModNo_List	20	1
10	10	0	SymbolShape_SubClass	4	1
11	11	0	xPosNavSideLights	0	1
12	12	0	SymbolSwitchScale	1.5	1
13	13	0	C13	n-a	1
14	14	0	Length	217.5	1
15	15	0	Height	36.8	1
16	16	0	Beam	32.2	1
17	17	0	C17	n-a	1
18	18	0	RadarAntennaOffset	0	1
19	20	0	RadarReflectionCoefficient	1	1
20	21	0	RadarBlindSecPos_List	**@List@Definition@**	1
21	21	1	RadarBlindSecPos_List	0	1
22	22	0	RadarBlindSecAp_List	**@List@Definition@**	1
23	22	1	RadarBlindSecAp_List	0	1
24	23	0	C23	n-a	1
25	24	0	AnchorSpeedDown	0.3	1
26	25	0	AnchorSpeedUp	0.5	1
27	27	0	VisualSystemModNoDesc_List	**@List@Definition@**	1
28	27	1	VisualSystemModNoDesc_List	DSR@Baltic	1
29	28	0	MaxSpeed	20	0.514444
30	29	0	MinSpeed	-5	0.514444
31	30	0	C30	n-a	1

Figure 4.1 List of complete variables
Personal Document

This step the data will be transferred one by one by typing the data in the Matlab editor. The doubled variable also manually eliminated and missing value also added manually. This step as the first step of this project as the result of this step will be the base of the other method. The static parameters of 0000_FFFF_1_1_1_Baltic-2013_Container_obj_40_597.dat that already been sort and added the missing value are listed on the Appendix 2.

There is some variable that need a data conversion for the POD-units as a propulsion drive. The two constants Angle_InflowReductionTable1 and Factor_InflowReductionTable1 have a doubled value inside. Because Simulink did not accept of the value not monotonously increasing, the data must be reducing to create a value that monotonously increase.

For example, the data from Prince-Richard-4POD (obj_40_403.dat) the value are: Angle_InflowReductionTable1 = [-180 -180 -90 -90 -75.5 -75.5 -68.1 -68.1 -54.5 -54.5 -50 -50 -41.2 -41.2 -38.7 -38.7 -35 -35 0 0 35 35 38.7 38.7 41.2 41.2 50 50 54.5 54.5 68.1 68.1 75.5 75.5 90 90 180 180]

```
Factor_InflowReductionTable1 =[1 1 1 1 0.45 0.45 0.45 0.45 0.3 0.3 0.3 0.3 0.34
0.34 0.34 0.34 0.1 0.1 0.1 0.1 0.1 0.1 0.34 0.34 0.34 0.3 0.3 0.3 0.3 0.45 0.45 0.45
0.45 1 1 1 1]
```

The result from the function above must be the following to have a strictly monotonously increasing function:

```
Angle_InflowReductionTable1 = [-180 -90 -75.5 -68.1 -54.5 -50 -41.2 -38.7 -35
0 35 38.7 41.2 50 54.5 68.1 75.5 90 180]
```

```
Factor_InflowReductionTable1 = [1 1 0.45 0.45 0.3 0.3 0.34 0.34 0.1 0.1 0.1 0.34
0.34 0.3 0.3 0.45 0.45 1 1]
```

If the length of Angle_InflowReductionTable1 and Factor_InflowReductionTable1 is smaller than 1 or if two following values are not identical, the value of these variable will be:

```
Angle_InflowReductionTable1 = [-180 180];
Factor_InflowReductionTable1 = [1 1; 1 1];
```

For the constant for propeller system value such as Identical_PropUnits and IdenticalRudderSystems the value will show “Y” or “N”. If the value showed “Y” or “y” then the following constant (2, 3, 4) will follow the value of the first unit.

For example, if "Identical_PropUnits" == "Y" then:
 AngleAttackTransFlow2 = AngleAttackTransFlow1;
 AngleAttackTransFlow3 = AngleAttackTransFlow1;
 AngleAttackTransFlow4 = AngleAttackTransFlow1;

```
Identical_PropUnits='Y';
```

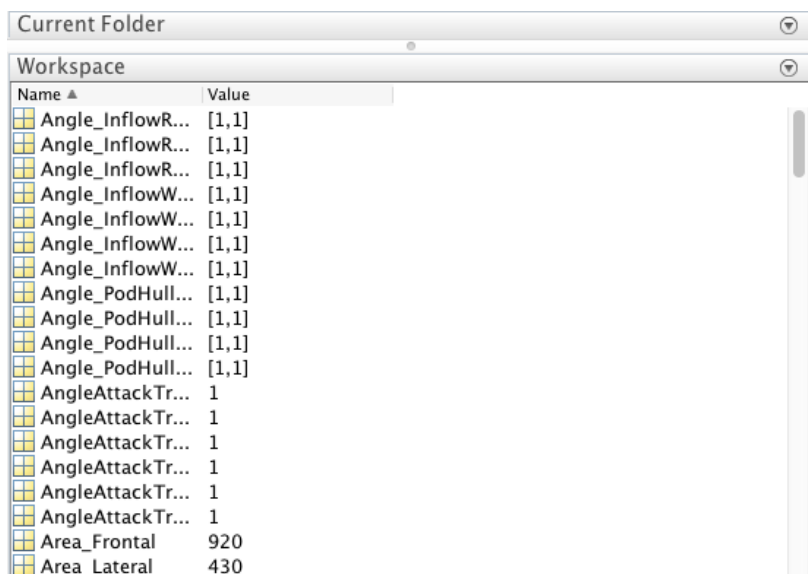
```
%Usually there are four constants on the list, for
this case this variable will follow the first
constant value if Identcal_PropUnit equal Y or y.
This will happen if the instructor personel only
input the propeller system values once, if they enter
for the constant "Identical_PropUnits" the value "Y"
```

or "y". This is a simplification for the instructors, when they design a new vessel model.

```
AngleAttackTransFlow1=0;
AngleAttackTransFlow2=0;
AngleAttackTransFlow3=0;
AngleAttackTransFlow4=0;
```

So, the value for AngleAttackTransFlow1 is copied to AngleAttackTransFlow2/3/4.

And this applies to the following constants will be written in Appendix 4. This is also applied for rudder system too.



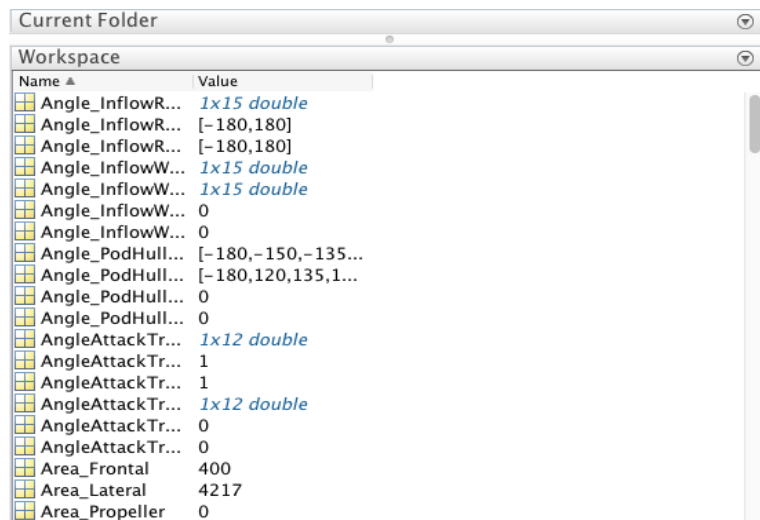
Name	Value
Angle_InflowR...	[1,1]
Angle_InflowR...	[1,1]
Angle_InflowR...	[1,1]
Angle_InflowW...	[1,1]
Angle_InflowW...	[1,1]
Angle_InflowW...	[1,1]
Angle_InflowW...	[1,1]
Angle_PodHull...	[1,1]
Angle_PodHull...	[1,1]
Angle_PodHull...	[1,1]
Angle_PodHull...	[1,1]
AngleAttackTr...	1
AngleAttackTr...	1
AngleAttackTr...	1
AngleAttackTr...	1
AngleAttackTr...	1
AngleAttackTr...	1
Area_Frontal	920
Area_Lateral	430

Figure 4.2 The result of manually codes data (1)

Personal Document

There are on a ship data variable files that have POD-unit thruster and they have a different value on Angle_PodHullInteractTable, Factor_PodHullInteractTable, RPM_PodHullInteractTable. If the ship has POD-unit, they will have a value on that three variables. The list of from that variables, will create *Spalten*, *Zeilen* and

Data variable on the list. For the list of ship that has POS-unit thruster listed on Appendix 3.



Name	Value
Angle_InflowR...	1x15 double
Angle_InflowR...	[-180,180]
Angle_InflowR...	[-180,180]
Angle_InflowW...	1x15 double
Angle_InflowW...	1x15 double
Angle_InflowW...	0
Angle_InflowW...	0
Angle_PodHull...	[-180,-150,-135...
Angle_PodHull...	[-180,120,135,1...
Angle_PodHull...	0
Angle_PodHull...	0
AngleAttackTr...	1x12 double
AngleAttackTr...	1
AngleAttackTr...	1
AngleAttackTr...	1x12 double
AngleAttackTr...	0
AngleAttackTr...	0
Area_Frontal	400
Area_Lateral	4217
Area_Propeller	0

Figure 4.3 The result of manually codes data (2)

Personal Document

Data value taken from a matrix form of Factor_PodHullInteractTable. *Spalten* value taken from a set data of RPM_PodHullInteractTable and *Zeilen* value taken from a set data of Angle_PodHullInteractTable.

The thread below showed the manually code function for Data, *Spalten* and *Zeilen*.

```
Angle_PodHullInteractTable1=[-180,-150,-135,-
120,180];
Angle_PodHullInteractTable2=[-180,120,135,150,180];
Angle_PodHullInteractTable3=0;
Angle_PodHullInteractTable4=0;

Zeilen1=[-180,-150,-135,-120,180];
Zeilen2=[-180,120,135,150,180];
Zeilen3=[-180,180];%this value showed if
Angle_PodHullInteractTable equals 0
```

```

Zeilen4=[-180,180];

Factor_PodHullInteractTable1=[1,1,0,0,0,0,0,0,1,1];
Factor_PodHullInteractTable2=[1,1,0,0,0,0,0,0,1,1];
Factor_PodHullInteractTable3=0;
Factor_PodHullInteractTable4=0;

Data1=[1,1;0,0;0,0;0,0;1,1];
Data2=[1,1;0,0;0,0;0,0;1,1];
Data3=[1,1;1,1]; %this value showed if
Factor_PodHullInteractTable equals 0
Data4=[1,1;1,1];

RPM_PodHullInteractTable1=[-100,100,-100,100,-
100,100,-100,100,-100,100];
RPM_PodHullInteractTable2=[-100,100,-100,100,-
100,100,-100,100,-100,100];
RPM_PodHullInteractTable3=0;
RPM_PodHullInteractTable4=0;

Spalten1 = [-100,100];
Spalten2 = [-100,100];
Spalten3 = [-100,100];%this value showed if
RPM_PodHullInteractTable equals 0
Spalten4 = [-100,100];

```

The value of *Spalten*, *Zeilen* and data are used as the auxiliary variable in Simulink. *Spalte* is German word for column and *Zeile* is German word for row.

If the length of Angle_PodHullInteractTable1 is smaller than 1 or if the values inside this constant are not a numeric format, than the auxiliary values will be defined as:

Spalten1 = [-100 100];

Zeilen1 = [-180 180];

Data1 = [1 1; 1 1];

4.2 Missing Variable from The Data

In this data there is a missing value that are not provided by the data from the simulator. This data has to be added by manually or created from the variable from the simulator. This missing value has a function as a converter or adding value for the Simulink simulation.

Table 4.1 Table of Missing Value

Description of The Variables	Variable Name	Unit
conversion factor knots -> meter/second	kn2ms=0.51445	$\frac{m/s}{kn}$
angular velocity	rpm2rads=2pi/60	1
angular velocity	rads2rpm=60/2pi	1
water density	rho_W=1025	kg/m ³
air density	rho_L=1.2	kg/m ³
initial x-position of vessel movement	X_pos_ini = 0	m
initial y-position of vessel movement	Y_pos_ini = 0	deg
initial course angle of vessel	psi_ini = 0	kn
initial speed over ground of vessel (in x-direction)	u_g_ini = 0	Kn
initial speed over ground of vessel (in y-direction)	v_g_ini = 0	Deg/s
initial angular speed over ground of vessel (around z-direction)	r_g_ini = 0	Deg
start ruder angle	delta_ini_1 = 0	Deg
start ruder angle	delta_ini_2 = 0	Deg
start ruder angle	delta_ini_3 = 0	Deg
start ruder angle	delta_ini_4 = 0	Deg

start rotational speed of 1. main engine	RPM1_ini = 0	U/min
start rotational speed of 2. main engine	RPM2_ini = 0	U/min
start rotational speed of 3. main engine	RPM3_ini = 0	U/min
start rotational speed of 4. main engine	RPM4_ini = 0	U/min
start pitch of 1. propeller (pitch = angle of the propeller fluke)	Pitch1_ini = 0	%
start pitch of 1. propeller (pitch = angle of the propeller fluke)	Pitch2_ini = 0	%
start pitch of 1. propeller (pitch = angle of the propeller fluke)	Pitch3_ini = 0	%
start pitch of 1. propeller (pitch = angle of the propeller fluke)	Pitch4_ini = 0	%

4.3 Developing the Data Interface for Data Transfer

The next stage it will be creating a code that the user can choose any desired file and directly transfer to the workspace and readable in Matlab. By creating some codes on Matlab. The aim of this step that users can select any original *.dat file and this selected can be read automatically without code one by one link of the files location and with this code it is easier to operate if it is required a lot of data to run with, without creating a lot of codes for the files. First, creating a code that allowed users to select a data and read the data into Matlab.

Data Reading

```
[filename,pathname] = uigetfile('*.','All Files (*.*)','MultiSelect','on');
fid = fopen(fullfile(pathname,filename));
if fid == -1
    disp('Error, check file name')
else
    S= textscan(fid,'%f %f %s %s %f','Delimiter',' ');
end
```


The result of this code, Matlab will be able to read the selected file. *If* and *else* function in this code will determine if the files/fid is equal -1 it will display in the command windows as “Error, check file name” means that the *fopen* cannot open the desired file. Else if the value of the fid showed 3 or more the file will be able to be read by *textscan* function. This function designed as `S= textscan(fid,'%f %f %s %s %f','Delimiter',' ').%f %f %s %s %f` (five conversion specifier) indicated that there are five rows that want to be read. %f is a conversion specifier for numeric inputs and %s is a conversion specifier for nonnumeric characters. Delimiter acts as a separator for the characters.

Textscan function is able to read a data and open a text file and convert the file into a cell array. Here f indicates the data input type is Floating-point number and s indicates the data input is text array and the data will be read as a cell array of character vector. The result of this code that Matlab can read every column of the data. By typing `S{columns's number}` on the command window, it will show all the rows in the selected column.

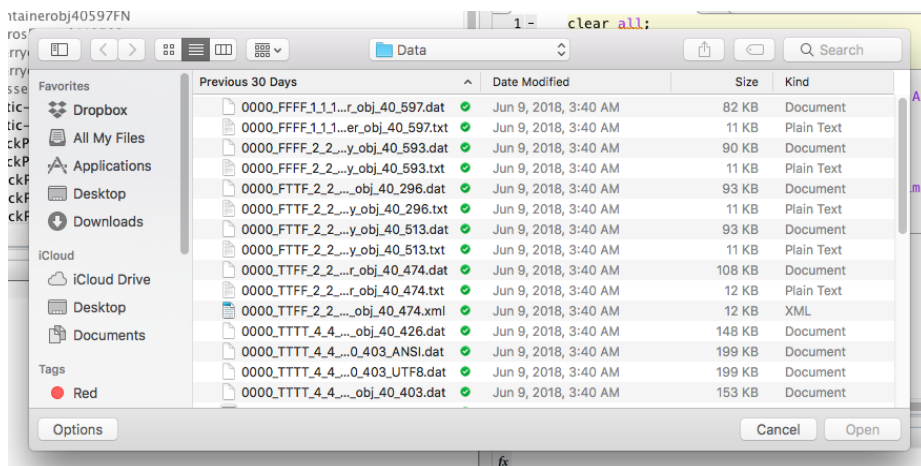


Figure 4.4 Choosing the file
Personal Document

This figure above showed the result of *fopen* function in the codes. By this function, users able to choose any data in any location in the computer and select the one that needed and in this case is a *.dat files.

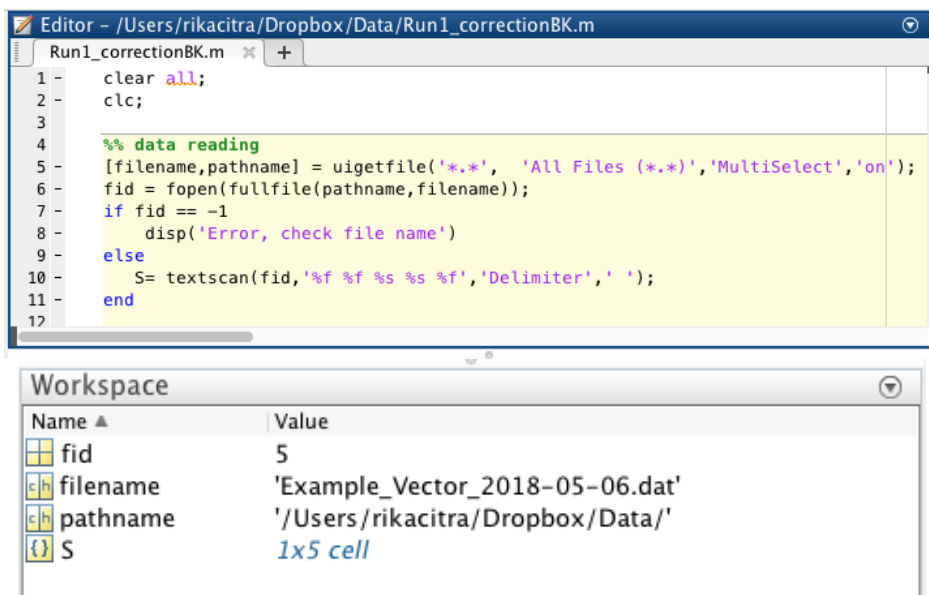


Figure 4.5 The result of data reading
Personal Document

Then from the selected data, variables from the file will be read automatically by this code. Here the data will be read by some for-loop one by one every variables name. the aim of this loop is that the result must able to showed the multiple value in one variable in one time without creating a code one by one for each rows of the files. every file has different order of data and this code must be created for each files of the simulator data. *i* and *ind* Indicate as a loop indexespecially *ind* is an index for rows and *Ergebnis* indicates as result.

By this code, it is automatically identifying the row that has a ****@List@Definition@**** and the row that has n-a value on the file.

```

for i = 1:58
for ind = 1:20
    H(i,ind)= 1 / (ind+i-1);
end
end
Ergebnis{1} = (1:1:max(S{1}))';
temp = strtok(char(S{3}(ind)));
if ~strcmp(char(S{4}(ind)), 'n-a') && ~((temp(1,1) ==
'C') && (isstrprop(temp(1,2), 'digit'))) &&
~strcmp(char(S{4}(ind)), '**ListDefinition**')

```

```

        Ergebnis{2}(i,1) = S{3}(ind);
    if ~isnan(str2double(S{4}(ind)));
        Ergebnis{4}(i,1) =
str2double(S{4}(ind));
        Ergebnis{3}(i,1) = {' '};
        current_name = char(Ergebnis{2}(i));
        current_val = Ergebnis{4}(i,1);
        cmd = sprintf('%s =
current_val;', current_name);
        eval(cmd);
    elseif isnan(str2double(S{4}(ind)));
        Ergebnis{3}(i,1) = S{4}(ind);
        Ergebnis{4}(i,1) = 0;
        current_name =
char(Ergebnis{2}(i,1));
        current_val =
char(Ergebnis{3}(i,1));
        cmd = sprintf('%s =
current_val;', current_name);
        eval(cmd);
    else disp('Fehler1')
end

elseif strcmp(char(S{4}(ind)), '**ListDefinition**')
    Ergebnis{2}(i,1) = S{3}(ind);
    Ergebnis{3}(i,1) = {' '};
    Ergebnis{4}(i,1) = 0;
    current_name =
char(Ergebnis{2}(i,1));
    current_val = Ergebnis{4}(i,1);
    cmd = sprintf('%s =
current_val;', current_name);
    eval(cmd);
elseif ((temp(1,1) == 'C') &&
(isstrprop(temp(1,2), 'digit')) ||
strcmp(char(S{4}(ind)), 'n-a'))
    Ergebnis{2}(i,1) = {' '};
    Ergebnis{3}(i,1) = {' '};
    Ergebnis{4}(i,1) = 0;
else
    disp('Fehler2')

```

```
end
```

The loop for-running form for *i* showed 1 to 20 on the code above, the loop means that there are 58 rows(data) on the file that will read at first step of the loop. Then the next step the code will loop a specific row of variable with *ind*.

The loop for-runningform for *ind*showed1 to 20 on the code above as a row index, the loop means that this code only focused on “WindResistCoeff_Cx_List” variable on the Example_Vector_2018-05-06.dat file. On the file “WindResistCoeff_Cx_List” are listed from the first row until the twentieth row.

The first *if* used to read the variable value. Not all the data value on the filecontains number but it can contain n-a, character, digits, or ****ListDefinition****. This code will determine what kind of type of the value by *if*function.

The function creating a variable from the workspace by selecting some part of the file. In this code, the focus only on column 3 and 4 and the column 3 has the variable name and column four has the variable value.

Fehler1 indicates if there is an error on the first *elseif*, if the result not categorize as n-a, character, digit or ****ListDefinition****. The second *elseif*create a value of ****ListDefinition**** as nothing. It means, this kind of value did not have any use for the program. Fehler2 it belongs to the second *elseif*.

The result value need to be transfer from *str* (string) to *double* (double precision value) because *str* represent a real of complex numeric value. *Str* can be a character vector, a cell array or string array. Meanwhile *double*in Matab is a default number.

The result of this code must be the same as the manually transferred value. Each of variable will be process by this code and create the aim result, by using the example data of vector and matrix variable data to try if the code works.On this project first,these codes are tried on vector and matrix type of variables with different files from the original files to tried the code functions.

First picture below showed the result on the vector type of variables. In this case the code is written one by one for each variable value even though the variable has a same name with multiple value. This method is not efficient because the actual condition of coding, the actual files has a lot of data and more than one variable has a lot of value in one variable name. The data attached in Appendix 6.

Workspace	
Name ▲	Value
ans	0
cmd	'WindResistCoeff_Cx_List = current_val21;'
current_name	'Length'
current_name2	'WindResistCoeff_Cx_List'
current_val	142
current_val10	-0.2380
current_val11	-0.7380
current_val12	-0.7150
current_val13	-0.7380
current_val14	-0.2380
current_val15	-0.1430
current_val16	-0.0480
current_val17	0
current_val18	0.0480
current_val19	0.1430
current_val2	0.7150
current_val20	0.2380
current_val21	0.8100
current_val3	0.8100

Figure 4.6 The result of automatically codes on vector data type

Personal Document

The second picture below showed the result of data reading from matrix type of variable data. In this case the result is showed that only one value of the variable showed on the Matlab workspace. The result should be able to create the same result as the manually codes.

Workspace	
Name ▲	Value
Angle_PodHull...	180
ans	100
cmd	'Factor_PodHullInteractTable1 = current_val;'
current_name	'Factor_PodHullInteractTable1'
current_val	1
Ergebnis	1x4 cell
Factor_PodHul...	1
fid	6
filename	'Example_Matrix_2018-05-06 (1).dat'
H	58x58 double
i	58
ind	58
pathname	'/Users/rikacitra/Dropbox/Data/'
RPM_PodHullIn...	100
S	1x5 cell
temp	'Factor_PodHullInteractTable1'

Figure 4.7 The result of automatically codes on Matrix data type
Personal Document

The problem occurs with the for-loop that the loop only want loop once on the desired row. Ind here indicates loop for the row on the file, in this file for Factor_PodHullInteractTable1 has a value from row two until row twenty. From the basic principal of loop, it should work as a loop and it has to proceed every value in each desired row, in this case is row two to row twenty. But from the workspace, the result only showed the row twenty from the data that attached in Appendix 5.

Because this matrix data has an automatically multiplied their values from the simulator that actually not necessary for creating a missing value. The unnecessary data must be eliminating from the desired result for creating a missing value for *Zeilen*, *Spalten* and *Data*.

```
% Simplify the real data from the file

Factor_PodHullInteractTable1 =
reshape(Factor_PodHullInteractTable1, [], 3);

RPM_PodHullInteractTable1(1:seqperiod(RPM_PodHullInteractTable1));

Angle_PodHullInteractTable1=
unique(Angle_PodHullInteractTable1);

% Creating Zeilen, Spalten and Data

Zeilen1 = Angle_PodHullInteractTable1';

Spalten1 = RPM_PodHullInteractTable1;

Data1 = Factor_PodHullInteractTable1;
```

The result of this thread will be creating a data which is used to form *Zeilen*, *Spalten* and *Data*. The example below showed the original value of the data.

```
RPM_PodHullInteractTable1 = [-100 50 100 -100 50 100 -100 50 100 -100 50
100 -100 50 100 -100 50 100];
```



```

1 0 0
1 0 0
1 1 1
1 1 1]

```

```

%Zeilen

if Angle_PodHullInteractTable1 == 0;
    Zeilen1 = [-180 180];
elseif Angle_PodHullInteractTable1 > 0;
    Zeilen1 = Angle_PodHullInteractTable1;
end

%Spalten

if RPM_PodHullInteractTable1 == 0;
    Spalten1 = [-100 100];
elseif RPM_PodHullInteractTable1 > 0;

    Spalten1 = RPM_PodHullInteractTable1;
end

%data

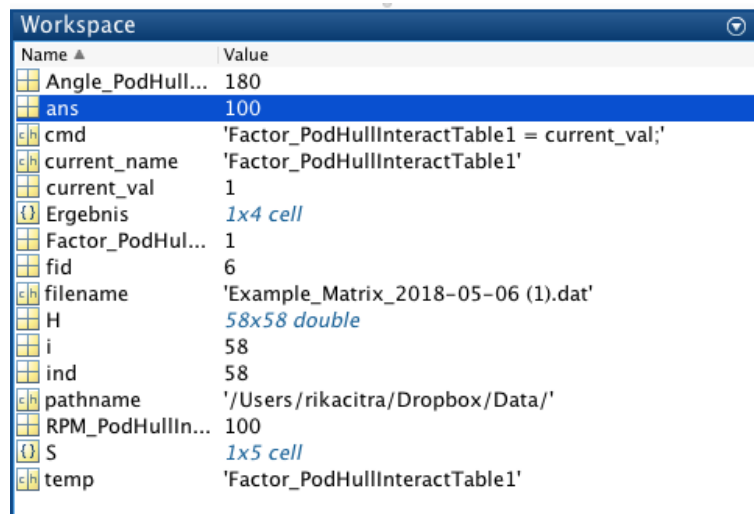
if Factor_PodHullInteractTable1 == 0;
    Data1 = [1 1;1 1];
elseif Factor_PodHullInteractTable1 > 0;
    Data1 = Factor_PodHullInteractTable1;
End

```

This code created to automatically decide if the data has a value of POD-unit thruster or not. When they did not have a value for RPM_PodHullInteractTable, Angle_PodHullInteractTable, and Factor_PodHullInteractTable, the value for *Zeilen*, *Spalten* and *Data* will be shown as Zeilen1 = [-180 180], Spalten1 = [-100 100], Data1 = [1 1;1 1].

4.4 Analysis for The Error Occur in The Transfer Data

On this Project, there is a problem with the transferring matrix type of data into the Matlab. From the codes that already design and written on the workspace as the thread on 4.3 this thread is used to transfer the data using for-loop so the variable that has multiple data can be transferred at once and creating a matrix variable on Matlab.



Name	Value
Angle_PodHull...	180
ans	100
cmd	'Factor_PodHullInteractTable1 = current_val;'
current_name	'Factor_PodHullInteractTable1'
current_val	1
Ergebnis	1x4 cell
Factor_PodHul...	1
fid	6
filename	'Example_Matrix_2018-05-06 (1).dat'
H	58x58 double
i	58
ind	58
pathname	'/Users/rikacitra/Dropbox/Data/'
RPM_PodHullIn...	100
S	1x5 cell
temp	'Factor_PodHullInteractTable1'

Figure 4.8Workspace result
Personal Document

From the figure above, the result of code on Appendix 7, of Angle_PodHullInteractTable1, Factor_PodHullInteractTable1, and RPM_PodHullInteractTable1 it is only showed one values from the file of Example_Matrix_2018-05-06 (1).dat. The wanted result from the code is the that the value of the variable above can show all of the value as the same as the manually type in the editor.

The problem from the code is with the function. The function has many flaws and the loop itself does not has any problem with the code.

The reason why the result is not coming out as expected because the loop is placed before the open file code. The file is read after the loop so the codes only read *i* and *ind* last iteration value as the value for the rest of the code. There is some code that it overwritten so it is wasting time of the calculating the code. for example, on the Appendix 7 about if *~isnan*:

```
if ~isnan(str2double(S{4}(ind)));  
%...  
elseif isnan(str2double(S{4}(ind)));  
%...  
end
```

the if *~isnan* code it is to prove if the value is true or false. But in this case *isnan* is already true value.

5 Result and Conclusion

5.1 Result

The aim result of this project is to able to transfer the data from the simulator to the Dynamic Positioning console that will be installed in the simulator.

1. The latest step of this project is creating the code that able to select and transfer a variable from a *.dat files into the Matlab workspace. The project must be stopped in the third step of this project because of the problem with the codes.

The result from the code that already created to transfer the data did not come out as expected. The result must show multiple of value of the variable but the actual result only showed one value of the variable. This problem makes this project stopped on the third point of the task of this bachelor thesis.

2. The code that already done is able to transfer data manually and automatically from the files. Manually data is archived by sorted and typed one by one the variable and the value inside the data. But there is some problem with transfer data automatically code. This code should be transferred the whole data with the same name variable but in this case the codes only want to transfer only one value of variable at the time. The code able to transfer data one by one for every value. But this method is not efficient for this kind of project because this project has a lot variable and mostly the variable has more than one value inside the variable. The naming also became difficult because in Matlab naming for the variable is important.

5.2 Conclusion

The conclusion of this report is that with Matlab, the variables from the files able to be transferred manually by sorted the data and typed the data in the editor and/or automatically by creating a code. This project has not completed yet because this project meets an obstacle on the third point of the assignment, as preparing Matlab programs for the execution and control of the data exchange. On this step, the variable must be read and transfer automatically from the *.dat files into the Matlab. But this step, the code that designed only transferred one of the exist value in the variable

“This Page Intentionally Left Blank”

Bibliography

1. Ltd, P. S. & T. P., 2014. *Full Mission Ship Handling Simulator (FMSHS)*. [Online]
Available at: https://www.prescient.com.sg/?page_id=57
[Access on 10.01.2018].
2. ENGINEERING, O., 2016. *Offshore Engineering*. [Online]
Available at: <http://www.offshoreengineering.com/education/dynamic-positioning-dp/what-is-dynamic-positioning>
[Access on 05.01.2018].
3. Chas, C. S. & García, R. F., 2008. Introduction to ship dynamic positioning systems. *Journal of Maritime Research*, Band 5, p. 79.
4. Wills, J., 2007. Dynamic Positioning Simulator Interim Report, Delft: TUDelft.
5. Kongsberg, kein Datum *Positioning systems, DP / thruster / joystick*. [Online]
Available at:
<https://www.km.kongsberg.com/ks/web/nokbg0240.nsf/AllWeb/14E17775E088ADC2C1256A4700319B04?OpenDocument>
[Access on 10.06.2018].
6. Bernhardt, F. & Kutschera, B., 2015. *Abschlussbericht für das FuE-Verbundvorhaben: Entwicklung eines schiffstypenunabhängigen adaptiven regelungstechnischen Kerns für eine Simulation von Dynamischen Positionier-Systemen (Kurztitel DP-SIM)*, s.l.: s.n.
7. MATLAB, 2005. *MATLAB® The Language of Technical Computing*. [Online]
Available at:
<https://www.mn.uio.no/astro/english/services/it/help/mathematics/matlab/getstart.pdf>
[Access on 06.01.2018].
8. Seg, M., 2015. *Simulink: To Workspace block, collects only one or two data points (Running in External mode)*. [Online]
Available at:
<https://www.mathworks.com/matlabcentral/answers/258050-simulink-to-workspace-block-collects-only-one-or-two-data-points-running-in-external-mode>
[Access on 06.01.2018].
9. Mathworks, kein Datum *Simulink*. [Online]
Available at: <https://www.mathworks.com/products/simulink.html>
[Access on 06.01.2018].
10. Carone, M. & Gotika, P., kein Datum *Getting Started with Simulink, Part 2: Adding a Controller and Plant to the Simulink Model*. [Online]

Available at: <https://www.mathworks.com/videos/getting-started-with-simulink-part-2-adding-a-controller-and-plant-to-the-simulink-model-1508442594866.html>

[Access on 07.01.2018].

11. McKay, B., kein Datum *Introduction to xPC Target and xPC Target Turnkey*. [Online]

Available at: <https://www.mathworks.com/videos/introduction-to-xpc-target-and-xpc-target-turnkey-68908.html>

[Access on 07.01.2018].

12. The MathWorks, I., 2000. *xPC Target For Use with Real-Time Workshop*. [Online]

Available at:

http://radio.feld.cvut.cz/matlab/pdf_doc/xpc/xpc_target_ug.pdf

[Access on 09.01.2018].

13. SIEMENS, 2006. *SIMATIC Industrial PC SIMATIC Box PC 627*. [Online]

Available at:

https://cache.industry.siemens.com/dl/files/221/21451221/att_69636/v1/ba_boxpc627_e.pdf

[Access on 15.01.2018].

14. Aliya, N., kein Datum *Nesaba Media*. [Online]

Available at: <https://www.nesabamedia.com/pengertian-udp-beserta-fungsi-dan-cara-kerjanya/>

Appendix 1: Static Parameters

Description of the variables	Variable name (with indexes for the propulsion units 1 to 4)	Unit
Vector with interpolation points of the POD rotation angle for which a reduction factor for the inflowing water jet is given	Angle_InflowReductionTable1..4	deg [-180..180]
Vector with interpolation points of the POD rotation angle for which a wake fraction is given	Angle_InflowWakeFactorTable1..4	deg [-180..180]
Vector with interpolation points of the POD rotation angle for which a factor for the cross current component of the propulsion unit is given	AngleAttackTransFlow1..4	deg [-180..180]
Propeller area	Area_Propeller1..4	m ²
Rudder area	Area_Rudder1..4	m ²
If the propulsion unit is to be considered a POD	CalcAzimuthPropulsion1..4	Y/N
If the transverse flow of the propulsion unit is to be calculated for the POD drive	CalcTransverseFlow1..4	Y/N
Empirical factor for reducing the rudder inflow angle	Coeff_AngleReduct1..4	-
Factor for the force component of the cross-flow in the propeller for advance	Coeff_Drag_Ahead1..4	-
Factor for the force component of the cross flow at the propeller for return	Coeff_Drag_Astern1..4	-

Influence of the tapering of the propeller jet	Coeff_RaceFactor	-
Thrust coefficient	Coeff_Thrust1..4	-
Matrix with factors influencing how much the hull influence reduces the POD thrust (line by line for var. Angle of rotation from lines, column by column for var. Speeds from columns)	Data1..4	[0..1]
Maximum rudder angle	Delta_Max1..4	Deg
Vector with interpolation points for the engine telegraph command points	EOTOrder_List1..4	%*10 [-1000..1000]
Vector with propeller gradients for the interpolation points from EOTOrder_List	EOTPitch_List..4	- [-100..100]
Vector with main engines rotation speed points for the interpolation points from EOTOrder_List	EOTRPM_List1..4	RPM
Efficiency of the propeller	Eta_Propeller1..4	% [0..100]
Vector with reducing factors for the inflowing water jet in PODs from Angle_InflowReduction Table	Factor_InflowReductionTable1..4	- [0..1]
If the wake factor is to be calculated as a constant, or a value table of angles and wake factors is given	FlagUseWakeFactorList1..4	Y/N
Tuning factor for force calculation of the cross flow component (If factor = 0 no cross current is calculated)	ForceCorrectionTransFlow1..4	[0..1]

Thrust coefficient for the propeller in POD drives for zero ahead thrust	K0TransFlow1..4	-
Length of the ship (construction waterline)	LengthWaterLine	m
Number of main drives (max 4)	NoPropUnits	-
Number of transverse jet drives (max.6)	NoThrusters	-
Rated power of the main drive	Power_Nom1..4	kN
Type of main drive (0 = propeller, 1 = Voith-Schneider drive)	PropUnitType1..4	-
Direction of rotation of the propellers, (+ 1 = clockwise, -1 = counterclockwise)	RotSense1..4	-
Nominal speed of the main engine for advance	RPM_FullAhead1..4	RPM
Nominal speed of the main engine for return	RPM_FullAstern1..4	RPM
Vector with interpolation points for the rudder angle	RudderCoeff_DeltaEffective_List 1..4	deg [-180..180]
Vector with factors for the drag coefficient of the rudder for rudder inflow angles from RudderCoeff_DeltaEffective_List	RudderCoeff_Drag_Normal_List 1..4	[0..1]
Vector with factors for the lift of the rudder for the rudder inflow angles of RudderCoeff_DeltaEffective_List	RudderCoeff_Lift_Normal_List 1..4	[0..1]
Vector with the rudder efficiencies for the flow velocities from	RudderEfficiencyValue_List	% [0..100]

RudderEfficiencyVelocity		
Vector with interpolation points for the percentage flow velocities at the rudder relative to the max. ship speed	RudderEfficiencyVelocity_List	% [0..100]
Vector with factors for the cross-flow component of the POD for the angles of rotation from AngleAttackTransFlow	ShapeTransFlow1..4	-
Vector with column interpolation points of the percentage nominal rated speeds in advance for the data matrix of the hull influencing factors for PODs	Spalten1..4	% [-100..100]
Nominal ship speed	Speed_Nom	kn
Rated power of the lateral thruster units	Thrust_Nom1..6	kN
Vector with interpolation points for the engine telegraph command point for the lateral thruster units	ThrusterEOTOrder_List1..66	%*10 [-1000... 1000]
Vector with the percentage thrust related to the max. Rated output of the lateral thruster units from ThrusterEOTOrder_List	ThrusterEOTThrust_List1..6	% [-100..100]
Angle in which the lateral thruster unit is mounted on the ship (usually 90 °)	ThrusterMaximumAngle1..6	deg
If the main drive should be calculated as a variable pitch propeller	VariablePitch1..4	Y/N

(N = fixed pitch propeller)		
Vector with wake factors for PODs for angles of rotation from Angle_InflowWakeFactorTable	Wake_InflowWakeFactorTable1..4	% [0..1]
Constant wake factor if the wake factor is not specified as a value table in Wake_InflowWakeFactorTable	WakeFactor1..4	% [0..1]
Ship-related x-coordinate of the center of gravity (COG)	xCentrGrav	m
Ship-related x-coordinate of the propeller	xPos_Propeller1..4	m
Ship related x-coordinate of the rudder	xPos_Rudder1..4	m
Ship related x-coordinate of the lateral thruster units	xPos_Thruster1..6	m
Ship-related y-coordinate of the propeller	yPos_Propeller1..4	m
Ship related y-coordinate of the rudder	yPos_Rudder1..4	m
Ship related y-coordinate of the lateral thruster units	yPos_Thruster1..6	m
Vector with line interpolation points for the rotation angles for the data matrix of the hull influencing factors for PODs	Zeilen1..4	[-180..180]

“This Page Intentionally Left Blank”

Appendix 2: 0000_FFFF_1_1_1_Baltic-2013_Container_obj_40_597.dat

```
% Static Parameters from the file
Angle_InflowReductionTable1=[-180,180];
Angle_InflowReductionTable2=[-180,180];
Angle_InflowReductionTable3=[-180,180];
Angle_InflowReductionTable4=[-180,180];

Angle_InflowWakeFactorTable1=0;
Angle_InflowWakeFactorTable2=0;
Angle_InflowWakeFactorTable3=0;
Angle_InflowWakeFactorTable4=0;

Angle_PodHullInteractTable1=0;
Angle_PodHullInteractTable2=0;
Angle_PodHullInteractTable3=0;
Angle_PodHullInteractTable4=0;

AngleAttackTransFlow1=0;
AngleAttackTransFlow2=0;
AngleAttackTransFlow3=0;
AngleAttackTransFlow4=0;

Area_Frontal=920;
Area_Lateral=4300;

Area_Propeller1=38.5;
Area_Propeller2=0;
Area_Propeller3=0;
Area_Propeller4=0;

Area_Rudder1=55;
Area_Rudder2=0;
Area_Rudder3=0;
Area_Rudder4=0;

BDN_RPM_Cmd_Eng_List1=[-100,-70,-25,0];
BDN_RPM_Cmd_Eng_List2=-100;
BDN_RPM_Cmd_Eng_List3=-100;
BDN_RPM_Cmd_Eng_List4=-100;

BDN_Time_List1=[0,5,16,35];
BDN_Time_List2=0;
BDN_Time_List3=0;
BDN_Time_List4=0;

BeamWaterLine=32.2;
```

```
BIN_RPM_Cmd_Eng_List1=[0,-25,-70,-100];  
BIN_RPM_Cmd_Eng_List2=-100;  
BIN_RPM_Cmd_Eng_List3=-100;  
BIN_RPM_Cmd_Eng_List4=-100;
```

```
BIN_Time_List1=[0,5,90,1800];  
BIN_Time_List2=0;  
BIN_Time_List3=0;  
BIN_Time_List4=0;
```

```
CalcAzimuthPropulsion1='FALSE';  
CalcAzimuthPropulsion2='FALSE';  
CalcAzimuthPropulsion3='FALSE';  
CalcAzimuthPropulsion4='FALSE';
```

```
CalcTransverseFlow1='FALSE';  
CalcTransverseFlow2='FALSE';  
CalcTransverseFlow3='FALSE';  
CalcTransverseFlow4='FALSE';
```

```
Coeff_AngleReduct1=0.5;  
Coeff_AngleReduct2=0;  
Coeff_AngleReduct3=0;  
Coeff_AngleReduct4=0;
```

```
Coeff_Drag_Ahead1=0.04;  
Coeff_Drag_Ahead2=0;  
Coeff_Drag_Ahead3=0;  
Coeff_Drag_Ahead4=0;
```

```
Coeff_Drag_Astern1=0.12;  
Coeff_Drag_Astern2=0;  
Coeff_Drag_Astern3=0;  
Coeff_Drag_Astern4=0;
```

```
Coeff_Friction1=0.03;  
Coeff_Friction2=0;  
Coeff_Friction3=0;  
Coeff_Friction4=0;
```

```
Coeff_RaceFactor=0.5;
```

```
Coeff_Thrust1=2.2;  
Coeff_Thrust2=0;  
Coeff_Thrust3=0;  
Coeff_Thrust4=0;
```

```
Coeff_Torque1=1.8;  
Coeff_Torque2=0;  
Coeff_Torque3=0;  
Coeff_Torque4=0;
```

```

% Missing Parameters
kn2ms      = 1.852/3.6;           % [kn in m/s] = 0,514444
ms2kn      = 1/kn2ms;           % [m/s in kn]
r2d        = 180/pi;            % [rad in grad]
d2r        = 1/r2d;             % [grad in rad]
rpm2rads   = 2*pi/60;           % [U/min in rad/s]
rads2rpm   = 1/rpm2rads;        % [rad/s in U/min]
g = 9.81;                          % [m/s^2] acceleration of gravity
rho_W = 1025;                     % [kg * m^-3] density of salt water
rho_L = 1.2;                     % [kg/m^3] density of air

X_pos_ini = 0;                   % [m] initial x-position of vessel
movement
Y_pos_ini = 0;                   % [m] initial y-position of vessel
movement
psi_ini = 0;                     % [deg] initial cours angle of
vessel
u_g_ini = 0;                     % [kn] initial speed over ground of
vessel (in x-direction)
v_g_ini = 0;                     % [kn] initial speed over ground of
vessel (in y-direction)
r_g_ini = 0;                     % [deg/s] initial angular speed over
ground of vessel (around z-direction)
delta_ini_1 = 0;                 % [deg] start ruder angle
delta_ini_2 = 0;                 % [deg] start ruder angle
delta_ini_3 = 0;                 % [deg] start ruder angle
delta_ini_4 = 0;                 % [deg] start ruder angle
RPM1_ini = 0;                   % [U/min] start rotational speed of
1. main engine
RPM2_ini = 0;                   % [U/min] start rotational speed of
2. main engine
RPM3_ini = 0;                   % [U/min] start rotational speed of
3. main engine
RPM4_ini = 0;                   % [U/min] start rotational speed of
4. main engine
Pitch1_ini = 0;                 % [%] start pitch of 1. propeller
(pitch = angle of the propeller fluke)
Pitch2_ini = 0;                 % [%] start pitch of 1. propeller
(pitch = angle of the propeller fluke)
Pitch3_ini = 0;                 % [%] start pitch of 1. propeller
(pitch = angle of the propeller fluke)
Pitch4_ini = 0;                 % [%] start pitch of 1. propeller
(pitch = angle of the propeller fluke)

% Static Parameters from the file (cont.)
Data1=[1,1;1,1];
Data2=[1,1;1,1];
Data3=[1,1;1,1];
Data4=[1,1;1,1];

```

```

Delta_Max1=35;
Delta_Max2=0;
Delta_Max3=0;
Delta_Max4=0;

DraughtBow=11;
DraughtStern=11;

EOTOrder_List1=[-1000,0,1000];
EOTOrder_List2=0;
EOTOrder_List3=0;
EOTOrder_List4=0;

EOTPitch_List1=[100,100,100];
EOTPitch_List2=0;
EOTPitch_List3=0;
EOTPitch_List4=0;

EOTRPM_List1=[-100,0,100];
EOTRPM_List2=0;
EOTRPM_List3=0;
EOTRPM_List4=0;

Eta_Propeller1=70;
Eta_Propeller2=0;
Eta_Propeller3=0;
Eta_Propeller4=0;

Factor_InflowReductionTable1=[1,1];
Factor_InflowReductionTable2=[1,1];
Factor_InflowReductionTable3=[1,1];
Factor_InflowReductionTable4=[1,1];

Factor_PodHullInteractTable1=0;
Factor_PodHullInteractTable2=0;
Factor_PodHullInteractTable3=0;
Factor_PodHullInteractTable4=0;

FDN_RPM_Cmd_Eng_List1=[100,80,75,25,0];
FDN_RPM_Cmd_Eng_List2=0;
FDN_RPM_Cmd_Eng_List3=0;
FDN_RPM_Cmd_Eng_List4=0;

FDN_Time_List1=[0,1800,2300,2450,2650];
FDN_Time_List2=0;
FDN_Time_List3=0;
FDN_Time_List4=0;

FIN_RPM_Cmd_Eng_List1=[0,25,70,90,100];
FIN_RPM_Cmd_Eng_List2=0;
FIN_RPM_Cmd_Eng_List3=0;

```



```

FIN_RPM_Cmd_Eng_List4=0;

FIN_Time_List1=[0,5,90,1800,3600];
FIN_Time_List2=0;
FIN_Time_List3=0;
FIN_Time_List4=0;

FlagUseWakeFactorList1='N';
FlagUseWakeFactorList2='N';
FlagUseWakeFactorList3='N';
FlagUseWakeFactorList4='N';

ForceCorrectionTransFlow1=0;
ForceCorrectionTransFlow2=0;
ForceCorrectionTransFlow3=0;
ForceCorrectionTransFlow4=0;

Identical_PropUnits='N';
IdenticalRudderSystems='N';

Inertia1=1891;
Inertia2=0;
Inertia3=0;
Inertia4=0;

K0TransFlow1=0;
K0TransFlow2=0;
K0TransFlow3=0;
K0TransFlow4=0;

Length=217.5;
LengthWaterLine=206.2;
Mass=49359;

NoPropUnits=1;
NoRudderSystems=1;
NoThrusters=1;

Kzz2=0.07;
Nrp=-0.01;
Nrr=0.24;
Nrtr=0.04150;
Nur=-0.15;
Nuv=-0.55;
Nv4r2=0.55331;
Nvp=-0.06;
Nvr=0.55331;
Nvrt=0.08300;
Nvv=1;
Nvvt=0.27666;
N5v3rt=0.08853;

```

```
PitchChangeRate1=0;
PitchChangeRate2=0;
PitchChangeRate3=0;
PitchChangeRate4=0;

Power_Nom1=17000;
Power_Nom2=10;
Power_Nom3=10;
Power_Nom4=10;

PropUnitType1=0;
PropUnitType2=0;
PropUnitType3=0;
PropUnitType4=0;

RateOfTurn_Max_RudderSys1=5.2;
RateOfTurn_Max_RudderSys2=0;
RateOfTurn_Max_RudderSys3=0;
RateOfTurn_Max_RudderSys4=0;

RateOfTurn_Min_RudderSys1=4;
RateOfTurn_Min_RudderSys2=0;
RateOfTurn_Min_RudderSys3=0;
RateOfTurn_Min_RudderSys4=0;

RudderAngleNeutral=0.34;

RotSense1=1;
RotSense2=-1;
RotSense3=-1;
RotSense4=-1;

RPM_Ahead_List1=[0,50,70,90,100];
RPM_Ahead_List2=0;
RPM_Ahead_List3=0;
RPM_Ahead_List4=0;

RPM_Astern_List1=[-100,-50,0];
RPM_Astern_List2=-100;
RPM_Astern_List3=-100;
RPM_Astern_List4=-100;

RPM_FullAhead1=99;
RPM_FullAhead2=0;
RPM_FullAhead3=0;
RPM_FullAhead4=0;

RPM_FullAstern1=-88;
RPM_FullAstern2=0;
RPM_FullAstern3=0;
```

```

RPM_FullAstern4=0;

RPM_PodHullInteractTable1=0;
RPM_PodHullInteractTable2=0;
RPM_PodHullInteractTable3=0;
RPM_PodHullInteractTable4=0;

RudderCoeff_DeltaEffective_List1=[-180,-150,-135,-90,-45,-32,-
25,-10,0,10,25,32,45,90,135,150,180];
RudderCoeff_DeltaEffective_List2=0;
RudderCoeff_DeltaEffective_List3=0;
RudderCoeff_DeltaEffective_List4=0;

RudderCoeff_Drag_Normal_List1=[0,0.26,0.4,0.6,0.1,0.07,0.07,0,0.
07,0.07,0.1,0.46,0.6,0.4,0.26,0];
RudderCoeff_Drag_Normal_List2=0;
RudderCoeff_Drag_Normal_List3=0;
RudderCoeff_Drag_Normal_List4=0;

RudderCoeff_Lift_Normal_List1=[0,0.9,0.4,0,-0.32,-1.5,-0.85,-
0.56,0,0.56,0.85,1.5,0.32,0,-0.4,-0.9,0];
RudderCoeff_Lift_Normal_List2=0;
RudderCoeff_Lift_Normal_List3=0;
RudderCoeff_Lift_Normal_List4=0;

RudderEfficiencyValue_List=[100,100,100];
RudderEfficiencyVelocity_List=[0,50,100];

ShapeTransFlow1=0;
ShapeTransFlow2=0;
ShapeTransFlow3=0;
ShapeTransFlow4=0;

Spalten1 = [-100,100];
Spalten2 = [-100,100];
Spalten3 = [-100,100];
Spalten4 = [-100,100];

Speed_Nom=19;

SpeedThreshold_RudderEngine1=10;
SpeedThreshold_RudderEngine2=0;
SpeedThreshold_RudderEngine3=0;
SpeedThreshold_RudderEngine4=0;

Thrust_Nom1=170;
Thrust_Nom2=0;
Thrust_Nom3=0;
Thrust_Nom4=0;
Thrust_Nom5=0;
Thrust_Nom6=0;

```

```

ThrusterEfficiencyValue_List1=[100,100,33,0,0];
ThrusterEfficiencyValue_List2=0;
ThrusterEfficiencyValue_List3=0;
ThrusterEfficiencyValue_List4=0;
ThrusterEfficiencyValue_List5=0;
ThrusterEfficiencyValue_List6=0;

```

```

ThrusterEfficiencyVeloc_List1=[0,10,30,50,100];
ThrusterEfficiencyVeloc_List2=0;
ThrusterEfficiencyVeloc_List3=0;
ThrusterEfficiencyVeloc_List4=0;
ThrusterEfficiencyVeloc_List5=0;
ThrusterEfficiencyVeloc_List6=0;

```

```

ThrusterEOTOrder_List1=[-1000,0,1000];
ThrusterEOTOrder_List2=0;
ThrusterEOTOrder_List3=0;
ThrusterEOTOrder_List4=0;
ThrusterEOTOrder_List5=0;
ThrusterEOTOrder_List6=0;

```

```

ThrusterEOTThrust_List1=[-100,0,100];
ThrusterEOTThrust_List2=0;
ThrusterEOTThrust_List3=0;
ThrusterEOTThrust_List4=0;
ThrusterEOTThrust_List5=0;
ThrusterEOTThrust_List6=0;

```

```

ThrusterMaximumAngle1=90;
ThrusterMaximumAngle2=90;
ThrusterMaximumAngle3=90;
ThrusterMaximumAngle4=90;
ThrusterMaximumAngle5=90;
ThrusterMaximumAngle6=90;

```

```

Torque_Ahead_List1=[30,50,90,90,100];
Torque_Ahead_List2=0;
Torque_Ahead_List3=0;
Torque_Ahead_List4=0;

```

```

Torque_Astern_List1=[-100,-50,-37];
Torque_Astern_List2=-100;
Torque_Astern_List3=-100;
Torque_Astern_List4=-100;

```

```

VariablePitch1='N';
VariablePitch2='Y';
VariablePitch3='Y';
VariablePitch4='Y';

```

```

Veloc_u_th=0;

Wake_InflowWakeFactorTable1=0;
Wake_InflowWakeFactorTable2=0;
Wake_InflowWakeFactorTable3=0;
Wake_InflowWakeFactorTable4=0;

WakeFactor1=0.2;
WakeFactor2=0;
WakeFactor3=0;
WakeFactor4=0;

WindResistCoeff_AlphaCN_List=[0.02,-0.1];
WindResistCoeff_AlphaCx_List=-0.3;
WindResistCoeff_AlphaCy_List=-0.9;

WindResistCoeff_CN_List=[0.02,-0.1];
WindResistCoeff_Cx_List=-0.3;
WindResistCoeff_Cy_List=-0.9;

xBowWaterLine=190;
xCentrGrav=87;

xPos_Propeller1=-10.4;
xPos_Propeller2=0;
xPos_Propeller3=0;
xPos_Propeller4=0;

xPos_Rudder1=-15.5;
xPos_Rudder2=0;
xPos_Rudder3=0;
xPos_Rudder4=0;

xPos_Thruster1=177.4;
xPos_Thruster2=0;
xPos_Thruster3=0;
xPos_Thruster4=0;
xPos_Thruster5=0;
xPos_Thruster6=0;

yPos_Propeller1=0;
yPos_Propeller2=0;
yPos_Propeller3=0;
yPos_Propeller4=0;

yPos_Rudder1=0;

yPos_Rudder2=0;
yPos_Rudder3=0;
yPos_Rudder4=0;

```

```
yPos_Thruster1=0;
yPos_Thruster2=0;
yPos_Thruster3=0;
yPos_Thruster4=0;
yPos_Thruster5=0;
yPos_Thruster6=0;

Xu4=-0.436;
Xup=-0.05;
Xuth=0;
Xuu=-0.03092;
Xuvvv=-200;
Xvr=0.1;
Y4v2rt=0.55331;
Yrp=-0.36;
Yrr=0.27666;
Yrrt=0.10375;
Yur=0.24;
Yuv=-2;
Yvp=-0.94683;
Yvr=1.8;
Yvrt=0.55331;
Yvv=3.31987;
Yvvt=0.82997;
Yvvvr=2.21324;

Zeilen1=[-180,180];
Zeilen2=[-180,180];
Zeilen3=[-180,180];
Zeilen4=[-180,180];
```

Appendix 3: 0000_TTFF_2_2_2_MS- Europa_Passenger_obj_40_474.dat (with POD-Thruster)

```
% Static Parameters from the file
Angle_InflowReductionTable1=[-180,-90,-60,-45,-35,-30,-
20,0,20,30,35,45,60,90,180];
Angle_InflowReductionTable2=[-180,-90,-60,-45,-35,-30,-
20,0,20,30,35,45,60,90,180];
Angle_InflowReductionTable3=[-180,180];
Angle_InflowReductionTable4=[-180,180];

Angle_InflowWakeFactorTable1=[-180,-90,-60,-45,-40,-20,-
10,0,10,20,40,45,60,90,180];
Angle_InflowWakeFactorTable2=[-180,-90,-60,-45,-40,-20,-
10,0,10,20,40,45,60,90,180];
Angle_InflowWakeFactorTable3=0;
Angle_InflowWakeFactorTable4=0;

Angle_PodHullInteractTable1=[-180,-150,-135,-120,180];
Angle_PodHullInteractTable2=[-180,120,135,150,180];
Angle_PodHullInteractTable3=0;
Angle_PodHullInteractTable4=0;

AngleAttackTransFlow1=[-180,-150,-120,-90,-70,-
25,35,70,90,120,150,180];
AngleAttackTransFlow2=[-180,-150,-120,-90,-70,-
35,25,70,90,120,150,180];
AngleAttackTransFlow3=0;
AngleAttackTransFlow4=0;

Area_Frontal=400;
Area_Lateral=4217;

Area_Propeller1=38.4;
Area_Propeller2=38.4;
Area_Propeller3=0;
Area_Propeller4=0;

Area_Rudder1=15;
Area_Rudder2=15;
Area_Rudder3=0;
Area_Rudder4=0;

BDN_RPM_Cmd_Eng_List1=[-100,0];
BDN_RPM_Cmd_Eng_List2=[-100,0];
BDN_RPM_Cmd_Eng_List3=-100;
BDN_RPM_Cmd_Eng_List4=-100;
```

```
BDN_Time_List1=[0,105];
BDN_Time_List2=[0,105];
BDN_Time_List3=0;
BDN_Time_List4=0;

BeamWaterLine=24;

BIN_RPM_Cmd_Eng_List1=[0,-40,-68,-100];
BIN_RPM_Cmd_Eng_List2=[0,-40,-68,-100];
BIN_RPM_Cmd_Eng_List3=-100;
BIN_RPM_Cmd_Eng_List4=-100;

BIN_Time_List1=[0,40,140,300];
BIN_Time_List2=[0,40,140,300];
BIN_Time_List3=0;
BIN_Time_List4=0;

CalcAzimuthPropulsion1='TRUE';
CalcAzimuthPropulsion2='TRUE';
CalcAzimuthPropulsion3='FALSE';
CalcAzimuthPropulsion4='FALSE';

CalcTransverseFlow1='TRUE';
CalcTransverseFlow2='TRUE';
CalcTransverseFlow3='FALSE';
CalcTransverseFlow4='FALSE';

Coeff_AngleReduct1=0.1;
Coeff_AngleReduct2=0.1;
Coeff_AngleReduct3=0.5;
Coeff_AngleReduct4=0;

Coeff_Drag_Ahead1=0.04;
Coeff_Drag_Ahead2=0.04;
Coeff_Drag_Ahead3=0;
Coeff_Drag_Ahead4=0;

Coeff_Drag_Astern1=0.06;
Coeff_Drag_Astern2=0.06;
Coeff_Drag_Astern3=0;
Coeff_Drag_Astern4=0;

Coeff_Friction1=0.3;
Coeff_Friction2=0.3;
Coeff_Friction3=0;
Coeff_Friction4=0;

Coeff_RaceFactor=1;

Coeff_Thrust1=2.5;
Coeff_Thrust2=2.5;
```



```

Coeff_Thrust3=0;
Coeff_Thrust4=0;

Coeff_Torque1=2;
Coeff_Torque2=2;
Coeff_Torque3=0;
Coeff_Torque4=0;

% Missing Parameters
kn2ms      = 1.852/3.6;           % [kn in m/s] = 0,514444
ms2kn      = 1/kn2ms;           % [m/s in kn]
r2d        = 180/pi;            % [rad in grad]
d2r        = 1/r2d;             % [grad in rad]
rpm2rads   = 2*pi/60;           % [U/min in rad/s]
rads2rpm   = 1/rpm2rads;        % [rad/s in U/min]
g = 9.81;                        % [m/s^2] acceleration of gravity
rho_W      = 1025;              % [kg * m^-3] density of salt water
rho_L      = 1.2;               % [kg/m^3] density of air

X_pos_ini = 0;                  % [m] initial x-position of vessel
movement
Y_pos_ini = 0;                  % [m] initial y-position of vessel
movement
psi_ini = 0;                    % [deg] initial cours angle of
vessel
u_g_ini = 0;                    % [kn] initial speed over ground of
vessel (in x-direction)
v_g_ini = 0;                    % [kn] initial speed over ground of
vessel (in y-direction)
r_g_ini = 0;                    % [deg/s] initial angular speed over
ground of vessel (around z-direction)
delta_ini_1 = 0;                % [deg] start ruder angle
delta_ini_2 = 0;                % [deg] start ruder angle
delta_ini_3 = 0;                % [deg] start ruder angle
delta_ini_4 = 0;                % [deg] start ruder angle
RPM1_ini = 0;                   % [U/min] start rotational speed of
1. main engine
RPM2_ini = 0;                   % [U/min] start rotational speed of
2. main engine
RPM3_ini = 0;                   % [U/min] start rotational speed of
3. main engine
RPM4_ini = 0;                   % [U/min] start rotational speed of
4. main engine
Pitch1_ini = 0;                 % [%] start pitch of 1. propeller
(pitch = angle of the propeller fluke)
Pitch2_ini = 0;                 % [%] start pitch of 1. propeller
(pitch = angle of the propeller fluke)
Pitch3_ini = 0;                 % [%] start pitch of 1. propeller
(pitch = angle of the propeller fluke)
Pitch4_ini = 0;                 % [%] start pitch of 1. propeller
(pitch = angle of the propeller fluke)

```

```

% Static Parameters from the file
Data1=[1,1;0,0;0,0;0,0;1,1];
Data2=[1,1;0,0;0,0;0,0;1,1];
Data3=[1,1;1,1];
Data4=[1,1;1,1];

Delta_Max1=180;
Delta_Max2=180;
Delta_Max3=35;
Delta_Max4=0;

DraughtBow=6.01;
DraughtStern=6.03;

EOTOrder_List1=[-1000,-500,-300,-100,0,100,300,500,1000];
EOTOrder_List2=[-1000,-500,-300,-100,0,100,300,500,1000];
EOTOrder_List3=[-1000,-500,-100,0,318,682,841,927,1000];
EOTOrder_List4=0;

EOTPitch_List1=[100,100,100,100,100,100,100,100,100];
EOTPitch_List2=[100,100,100,100,100,100,100,100,100];
EOTPitch_List3=[-80,-50,-20,0,10,30,50,70,100];
EOTPitch_List4=0;

EOTRPM_List1=[-74.4,-50.44,-30.267,-
7.715,0,7.715,30.267,50.44,100];
EOTRPM_List2=[-74.4,-50.44,-30.267,-
7.715,0,7.715,30.267,50.44,100];
EOTRPM_List3=[100,100,100,100,100,100,100,100,100];
EOTRPM_List4=0;

Eta_Propeller1=75;
Eta_Propeller2=75;
Eta_Propeller3=0;
Eta_Propeller4=0;

Factor_InflowReductionTable1=[1,1,1,0.8,0.7,0.4,0.2,0.1,0.1,0.2,
0.4,0.7,0.8,1,1];
Factor_InflowReductionTable2=[1,1,0.8,0.7,0.4,0.2,0.1,0.1,0.2,0.
4,0.7,0.8,1,1,1];
Factor_InflowReductionTable3=0;
Factor_InflowReductionTable4=0;

Factor_PodHullInteractTable1=[1,1,0,0,0,0,0,0,1,1];
Factor_PodHullInteractTable2=[1,1,0,0,0,0,0,0,1,1];
Factor_PodHullInteractTable3=0;
Factor_PodHullInteractTable4=0;

FDN_RPM_Cmd_Eng_List1=[100,0];
FDN_RPM_Cmd_Eng_List2=[100,0];

```

```

FDN_RPM_Cmd_Eng_List3=0;
FDN_RPM_Cmd_Eng_List4=0;

FDN_Time_List1=[0,105];
FDN_Time_List2=[0,105];
FDN_Time_List3=0;
FDN_Time_List4=0;

FIN_RPM_Cmd_Eng_List1=[0,7.74,30.35,50.6,100];
FIN_RPM_Cmd_Eng_List2=[0,7.74,30.35,50.6,100];
FIN_RPM_Cmd_Eng_List3=0;
FIN_RPM_Cmd_Eng_List4=0;

FIN_Time_List1=[0,1,10,20,30];
FIN_Time_List2=[0,1,10,20,30];
FIN_Time_List3=0;
FIN_Time_List4=0;

FlagUseWakeFactorList1='N';
FlagUseWakeFactorList2='N';
FlagUseWakeFactorList3='N';
FlagUseWakeFactorList4='N';

ForceCorrectionTransFlow1=1;
ForceCorrectionTransFlow2=1;
ForceCorrectionTransFlow3=0;
ForceCorrectionTransFlow4=0;

Identical_PropUnits='FALSE';
IdenticalRudderSystems='FALSE';

Inertia1=400;
Inertia2=400;
Inertia3=0;
Inertia4=0;

K0TransFlow1=0.4;
K0TransFlow2=0.4;
K0TransFlow3=0;
K0TransFlow4=0;

Length=198.6;
LengthWaterLine=179.86;
Mass=16358;

NoPropUnits=2;
NoRudderSystems=2;
NoThrusters=2;

Kzz2=0.07;
Nrp=-0.05128;

```

```
Nrr=0.13021;
Nrtr=0.05209;
Nur=-0.16713;
Nuv=-0.36193;
Nv4r2=0.69447;
Nvp=-0.01043;
Nvr=0.69447;
Nvrt=0.26043;
Nvv=1.04170;
Nvvt=0.34723;
N5v3rt=0.27779;

PitchChangeRate1=0;
PitchChangeRate2=0;
PitchChangeRate3=1;
PitchChangeRate4=1;

Power_Nom1=6650;
Power_Nom2=6650;
Power_Nom3=10;
Power_Nom4=10;

PropUnitType1=0;
PropUnitType2=0;
PropUnitType3=0;
PropUnitType4=0;

RateOfTurn_Max_RudderSys1=5;
RateOfTurn_Max_RudderSys2=5;
RateOfTurn_Max_RudderSys3=2;
RateOfTurn_Max_RudderSys4=0;

RateOfTurn_Min_RudderSys1=3.5;
RateOfTurn_Min_RudderSys2=3.5;
RateOfTurn_Min_RudderSys3=2;
RateOfTurn_Min_RudderSys4=0;

RudderAngleNeutral=0.34;

RotSense1=1;
RotSense2=-1;
RotSense3=1;
RotSense4=-1;

RPM_Ahead_List1=[0,100];
RPM_Ahead_List2=[0,100];
RPM_Ahead_List3=0;
RPM_Ahead_List4=0;

RPM_Astern_List1=[-100,0];
RPM_Astern_List2=[-100,0];
```

```

RPM_Astern_List3=-100;
RPM_Astern_List4=-100;

RPM_FullAhead1=168.5;
RPM_FullAhead2=168.5;
RPM_FullAhead3=0;
RPM_FullAhead4=0;

RPM_FullAstern1=-168.5;
RPM_FullAstern2=-168.5;
RPM_FullAstern3=0;
RPM_FullAstern4=-1000;

RPM_PodHullInteractTable1=[-100,100,-100,100,-100,100,-100,100,-
100,100];
RPM_PodHullInteractTable2=[-100,100,-100,100,-100,100,-100,100,-
100,100];
RPM_PodHullInteractTable3=0;
RPM_PodHullInteractTable4=0;

RudderCoeff_DeltaEffective_List1=[-180,-150,-135,-90,-60,-40,-
30,-15,0,15,30,40,60,90,135,150,180];
RudderCoeff_DeltaEffective_List2=[-180,-150,-135,-90,-60,-40,-
30,-15,0,15,30,40,60,90,135,150,180];
RudderCoeff_DeltaEffective_List3=[-30,0,30];
RudderCoeff_DeltaEffective_List4=0;

RudderCoeff_Drag_Normal_List1=[0,0.26,0.4,0.6,0.46,0.3,0.1,0.07,
0,0.07,0.1,0.3,0.46,0.6,0.4,0.26,0];
RudderCoeff_Drag_Normal_List2=[0,0.26,0.4,0.6,0.46,0.3,0.1,0.07,
0,0.07,0.1,0.3,0.46,0.6,0.4,0.26,0];
RudderCoeff_Drag_Normal_List3=[0.1,0,0.1];
RudderCoeff_Drag_Normal_List4=0;

RudderCoeff_Lift_Normal_List1=[0,0.45,0.2,0,-0.1,-0.9,-1.6,-
0.9,0,0.9,1.6,0.9,0.1,0,-0.2,-0.45,0];
RudderCoeff_Lift_Normal_List2=[0,0.45,0.2,0,-0.1,-0.9,-1.6,-
0.9,0,0.9,1.6,0.9,0.1,0,-0.2,-0.45,0];
RudderCoeff_Lift_Normal_List3=[0.4,0,-0.4];
RudderCoeff_Lift_Normal_List4=0;

RudderEfficiencyValue_List=[100,100,100];
RudderEfficiencyVelocity_List=[0,50,100];

ShapeTransFlow1=[0,-0.5,-0.86,-1,-0.3,-
0.05,0,0.05,0.3,1,0.86,0.5,0];
ShapeTransFlow2=[0,-0.5,-0.86,-1,-0.3,-
0.05,0,0.05,0.3,1,0.86,0.5,0];
ShapeTransFlow3=0;
ShapeTransFlow4=0;

```

```

Spalten1 = [-100,100];
Spalten2 = [-100,100];
Spalten3 = [-100,100];
Spalten4 = [-100,100];

Speed_Nom=21.4;

SpeedThreshold_RudderEngine1=6;
SpeedThreshold_RudderEngine2=6;
SpeedThreshold_RudderEngine3=0.1;
SpeedThreshold_RudderEngine4=0;

Thrust_Nom1=70;
Thrust_Nom2=70;
Thrust_Nom3=100;
Thrust_Nom4=0;
Thrust_Nom5=0;
Thrust_Nom6=0;

ThrusterEfficiencyValue_List1=[100,100,80,100];
ThrusterEfficiencyValue_List2=[100,100,80,100];
ThrusterEfficiencyValue_List3=[100,100,33,66];
ThrusterEfficiencyValue_List4=0;
ThrusterEfficiencyValue_List5=0;
ThrusterEfficiencyValue_List6=0;

ThrusterEfficiencyVeloc_List1=[0,30,50,100];
ThrusterEfficiencyVeloc_List2=[0,30,50,100];
ThrusterEfficiencyVeloc_List3=[0,10,30,100];
ThrusterEfficiencyVeloc_List4=0;
ThrusterEfficiencyVeloc_List5=0;
ThrusterEfficiencyVeloc_List6=0;

ThrusterEOTOrder_List1=[-1000,0,1000];
ThrusterEOTOrder_List2=[-1000,0,1000];
ThrusterEOTOrder_List3=[-1000,0,1000];
ThrusterEOTOrder_List4=0;
ThrusterEOTOrder_List5=0;
ThrusterEOTOrder_List6=0;

ThrusterEOTThrust_List1=[-100,0,100];
ThrusterEOTThrust_List2=[-100,0,100];
ThrusterEOTThrust_List3=[-100,0,100];
ThrusterEOTThrust_List4=0;
ThrusterEOTThrust_List5=0;
ThrusterEOTThrust_List6=0;

ThrusterMaximumAngle1=90;
ThrusterMaximumAngle2=90;
ThrusterMaximumAngle3=90;
ThrusterMaximumAngle4=90;

```

```

ThrusterMaximumAngle5=90;
ThrusterMaximumAngle6=90;

Torque_Ahead_List1=[100,100];
Torque_Ahead_List2=[100,100];
Torque_Ahead_List3=0;
Torque_Ahead_List4=0;

Torque_Astern_List1=[-100,-100];
Torque_Astern_List2=[-100,-100];
Torque_Astern_List3=-100;
Torque_Astern_List4=-100;

VariablePitch1='FALSE';
VariablePitch2='FALSE';
VariablePitch3='Y';
VariablePitch4='Y';

Veloc_u_th=0;

Wake_InflowWakeFactorTable1=[0,0,0,0,0.025,0.05,0.08,0.1,0.08,0.05,0.025,0,0,0,0];
Wake_InflowWakeFactorTable2=[0,0,0,0,0.025,0.05,0.08,0.1,0.08,0.05,0.025,0,0,0,0];
Wake_InflowWakeFactorTable3=0;
Wake_InflowWakeFactorTable4=0;

WakeFactor1=0.2;
WakeFactor2=0.2;
WakeFactor3=0.2;
WakeFactor4=0;

WindResistCoeff_AlphaCN_List=[-180,-140,-80,-40,0,40,80,140,180];
WindResistCoeff_AlphaCx_List=-0.8;
WindResistCoeff_AlphaCy_List=[-180,-140,-100,-45,-20,20,45,100,140,180];

WindResistCoeff_CN_List=[0,-0.143,-0.04,-0.128,0,-0.128,-0.04,0.143,0];
WindResistCoeff_Cx_List=-0.8;
WindResistCoeff_Cy_List=[0,0.86,0.9,0.86,0.4,-0.4,-0.86,-0.9,-0.86,0];

xBowWaterLine=13.2;
xCentrGrav=-67.13;

xPos_Propeller1=-155;
xPos_Propeller2=-155;
xPos_Propeller3=0;
xPos_Propeller4=0;

```

```
xPos_Rudder1=-157;
xPos_Rudder2=-157;
xPos_Rudder3=13.7;
xPos_Rudder4=0;

xPos_Thruster1=19.74;
xPos_Thruster2=14.2;
xPos_Thruster3=-3.6;
xPos_Thruster4=0;
xPos_Thruster5=0;
xPos_Thruster6=0;

yPos_Propeller1=-5;
yPos_Propeller2=-5;
yPos_Propeller3=0;
yPos_Propeller4=0;

yPos_Rudder1=-5;
yPos_Rudder2=-5;
yPos_Rudder3=0;
yPos_Rudder4=0;

yPos_Thruster1=0;
yPos_Thruster2=0;
yPos_Thruster3=0;
yPos_Thruster4=0;
yPos_Thruster5=0;
yPos_Thruster6=0;

Xu4=-0.45;
Xup=-0.05;
Xuth=-362.42709;
Xuu=-0.04932;
Xuvvv=-200;
Xvr=0.81999;
Y4v2rt=0.69447;
Yrp=-0.02312;
Yrr=0.34723;
Yrrt=0.13021;
Yur=0.49955;
Yuv=-1.25639;
Yvp=-0.8199;
Yvr=2.08340;
Yvrt=0.69447;
Yvv=4.16681;
Yvvt=1.077787;
Yvvvr=2.77787;

Zeilen1=[-180,-150,-135,-120,180];
Zeilen2=[-180,120,135,150,180];
```



```
Zeilen3=[-180,180];  
Zeilen4=[-180,180];
```

“This Page Intentionally Left Blank”

Appendix 4: The following constants in a special condition that depends on the value of Identical_PropUnits and IdenticalRudderSystems

Identical_PropUnits:

- AngleAttackTransFlow1
- Angle_InflowWakeFactorTable1;
- Angle_InflowReductionTable1;
- Area_Propeller1;
- BDN_Time_List1;
- BDN_RPM_Cmd_Eng_List1;
- BIN_Time_List1;
- BIN_RPM_Cmd_Eng_List1;
- Coeff_Drag_Ahead1;
- Coeff_Drag_Astern1;
- Coeff_Friction1;
- Coeff_Thrust1;
- Coeff_Torque1;
- EOTOrder_List1;
- EOTPitch_List1;
- EOTRPM_List1;
- Eta_Propeller1;
- ForceCorrectionTransFlow1;
- FDN_Time_List1;
- FDN_RPM_Cmd_Eng_List1;
- FIN_Time_List1;
- FIN_RPM_Cmd_Eng_List1;
- Inertia1;
- K0TransFlow1;
- PitchChangeRate1;
- RPM_FullAhead1;
- RPM_FullAstern1;
- RPM_Astern_List1;
- RPM_Ahead_List1;
- ShapeTransFlow1;
- Torque_Astern_List1;
- Torque_Ahead_List1;
- Wake_InflowWakeFactorTable1;

IdenticalRudderSystems:

- Angle_InflowWakeFactorTable1;
- Angle_InflowReductionTable1;
- Area_Rudder1;
- Coeff_AngleReduct1;
- Delta_Max1;
- Delta_Max_Low1;
- Factor_InflowReductionTable1;
- FlagUseWakeFactorList1;
- RateOfTurn_Max_RudderSys1;
- RateOfTurn_Min_RudderSys1;
- RudderCoeff_DeltaEffective_List1;
- RudderCoeff_Lift_Normal_List1;
- RudderCoeff_Drag_Normal_List1;
- SpeedThreshold_RudderEngine1;
- WakeFactor1;

Appendix 5: Code for Reading and Transfer Data Automatically

```
% Data Reading
[filename,pathname] = uigetfile('*..*', 'All Files
(*.*)','MultiSelect','on');
fid = fopen(fullfile(pathname,filename));
if fid == -1
    disp('Error, check file name')
else
    S= textscan(fid,'%f %f %s %s %f','Delimiter',' ');
end

% Data Transfer
for i = 1:58
    for ind = 1:20
        H(i,ind)= 1 /(ind+i-1);
    end
end
Ergebnis{1} = (1:1:max(S{1}))';
temp = strtok(char(S{3}(ind)));
if ~strcmp(char(S{4}(ind)),'n-a') && ~((temp(1,1)=='C') &&
(isstrprop(temp(1,2),'digit')) &&
~strcmp(char(S{4}(ind)),'**ListDefinition**'))
    Ergebnis{2}(i,1) = S{3}(ind);
if ~isnan(str2double(S{4}(ind)));
    Ergebnis{4}(i,1) = str2double(S{4}(ind));
    Ergebnis{3}(i,1) = {' '};
    current_name = char(Ergebnis{2}(i));
    current_val = Ergebnis{4}(i,1);
    cmd = sprintf('%s = current_val;',current_name);
    eval(cmd);
elseif isnan(str2double(S{4}(ind)));
    Ergebnis{3}(i,1) = S{4}(ind);
    Ergebnis{4}(i,1) = 0;
    current_name = char(Ergebnis{2}(i,1));
    current_val = char(Ergebnis{3}(i,1));
    cmd = sprintf('%s = current_val;',current_name);
    eval(cmd);
else disp('Fehler1')
end

elseif strcmp(char(S{4}(ind)),'**ListDefinition**')
    Ergebnis{2}(i,1) = S{3}(ind);
    Ergebnis{3}(i,1) = {' '};
    Ergebnis{4}(i,1) = 0;
    current_name = char(Ergebnis{2}(i,1));
    current_val = Ergebnis{4}(i,1);
    cmd = sprintf('%s = current_val;',current_name);
    eval(cmd);
```

```

elseif ((temp(1,1)== 'C') && (isstrprop(temp(1,2),'digit')) ||
strcmp(char(S{4}(ind)), 'n-a'))
    Ergebnis{2}(i,1) = {' '};
    Ergebnis{3}(i,1) = {' '};
    Ergebnis{4}(i,1) = 0;
else
    disp('Fehler2')
end

Angle_PodHullInteractTable1=
unique(Angle_PodHullInteractTable1);

% Take one example from file Example_Matrix_2018-05-06, if the
loop run successfully it will show the result as
Angle_PodHullInteractTable1= [-180.0 -180.0 -180.0 -155.0 -155.0
-155.0 -145.0 -145.0 -145.0 125.0 -125.0 -125.0 -115.0 -115.0 -
115.0 180.0 180.0 180.0].
% Then if the "Angle_PodHullInteractTable1" would be read in
successfully, the following necessary processing for
"Angle_PodHullInteractTable1" creation would be:
Angle_PodHullInteractTable1= unique(Angle_PodHullInteractTable1)
and it will create a result as Angle_PodHullInteractTable1=[-
180,-150,-135,-120,180].

```

RPM_PodHullInteractTable1

```

% Data Transfer
for i = 1:58;
for ind = 21:39;
    H(i,ind)= 1 / (ind+i-1);
end
end

Ergebnis{1,1} = (1:1:max(S{1,1}))';
temp = strtok(char(S{1,3}(ind,1)));
if ~strcmp(char(S{1,4}(ind(1,1),1)), 'n-a') && ~((temp(1,1)=='
'C') && (isstrprop(temp(1,2),'digit')) &&
~strcmp(char(S{1,4}(ind(1,1),1)), '**ListDefinition**'))
    Ergebnis{1,2}(i,1) = S{1,3}(ind,1);
if ~isnan(str2double(S{1,4}(ind(1,1),1)));
    Ergebnis{1,4}(i,1) =
str2double(S{1,4}(ind(1,1),1));
    Ergebnis{1,3}(i,1) = {' '};
    current_name = char(Ergebnis{1,2}(i,1));
    current_val = Ergebnis{1,4}(i,1);
    cmd = sprintf('%s = current_val;', current_name);
    eval(cmd);
elseif isnan(str2double(S{1,4}(ind(1,1),1)));
    Ergebnis{1,3}(i,1) = S{1,4}(ind(1,1),1);
    Ergebnis{1,4}(i,1) = 0;

```

```

        current_name = char(Ergebnis{1,2}(i,1));
        current_val = char(Ergebnis{1,3}(i,1));
        cmd = sprintf('%s = current_val;',current_name);
        eval(cmd);
else disp('Fehler1')
end

elseif strcmp(char(S{1,4}(ind(1,1),1)), '**ListDefinition**')
    Ergebnis{1,2}(i,1) = S{1,3}(ind,1);
    Ergebnis{1,3}(i,1) = {' '};
    Ergebnis{1,4}(i,1) = 0;
    current_name = char(Ergebnis{1,2}(i,1));
    current_val = Ergebnis{1,4}(i,1);
    cmd = sprintf('%s = current_val;',current_name);
    eval(cmd);
elseif ((temp(1,1)== 'C') && (isstrprop(temp(1,2),'digit'))) ||
strcmp(char(S{1,4}(ind(1,1),1)), 'n-a')
    Ergebnis{1,2}(i,1) = {' '};
    Ergebnis{1,3}(i,1) = {' '};
    Ergebnis{1,4}(i,1) = 0;
else
    disp('Fehler2')
end

RPM_PodHullInteractTable1(1:seqperiod(RPM_PodHullInteractTable1)
);

% One example from file Example_Matrix_2018-05-06, if the loop
run successfully it will show the result as
RPM_PodHullInteractTable1= [-100.0 50.0 100.0 -100.0 50.0 100.0
-100.0 50.0 100.0 -100.0 50.0 100.0 -100.0 50.0 100.0 -100.0
50.0 100.0].
% Then if the "RPM_PodHullInteractTable1" would be read in
successfully, the following necessary processing for "
RPM_PodHullInteractTable1" creation would be:
Angle_PodHullInteractTable1= unique(Angle_PodHullInteractTable1)
and it will create a result as RPM_PodHullInteractTable1=[-
100,100,-100,100,-100,100,-100,100,-100,100].

```

Factor_PodHullInteractTable1

```

% Data Transfer
for i = 1:58;
for ind = 40:58;
    H(i,ind)= 1 /(ind+i-1);
end
end

Ergebnis{1,1} = (1:1:max(S{1,1}))';
temp = strtok(char(S{1,3}(ind,1)));

```

```

if ~strcmp(char(S{1,4}(ind(1,1),1)), 'n-a') && ~((temp(1,1)=='
'C') && (isstrprop(temp(1,2), 'digit')) &&
~strcmp(char(S{1,4}(ind(1,1),1)), '**ListDefinition**'))
    Ergebnis{1,2}(i,1) = S{1,3}(ind,1);
if ~isnan(str2double(S{1,4}(ind(1,1),1)));
    Ergebnis{1,4}(i,1) =
str2double(S{1,4}(ind(1,1),1));
    Ergebnis{1,3}(i,1) = {' '};
    current_name = char(Ergebnis{1,2}(i,1));
    current_val = Ergebnis{1,4}(i,1);
    cmd = sprintf('%s = current_val;', current_name);
    eval(cmd);
elseif isnan(str2double(S{1,4}(ind(1,1),1)));
    Ergebnis{1,3}(i,1) = S{1,4}(ind(1,1),1);
    Ergebnis{1,4}(i,1) = 0;
    current_name = char(Ergebnis{1,2}(i,1));
    current_val = char(Ergebnis{1,3}(i,1));
    cmd = sprintf('%s = current_val;', current_name);
    eval(cmd);
else disp('Fehler1')
end

elseif strcmp(char(S{1,4}(ind(1,1),1)), '**ListDefinition**')
    Ergebnis{1,2}(i,1) = S{1,3}(ind,1);
    Ergebnis{1,3}(i,1) = {' '};
    Ergebnis{1,4}(i,1) = 0;
    current_name = char(Ergebnis{1,2}(i,1));
    current_val = Ergebnis{1,4}(i,1);
    cmd = sprintf('%s = current_val;', current_name);
    eval(cmd);
elseif ((temp(1,1)=='C') && (isstrprop(temp(1,2), 'digit'))) ||
strcmp(char(S{1,4}(ind(1,1),1)), 'n-a')
    Ergebnis{1,2}(i,1) = {' '};
    Ergebnis{1,3}(i,1) = {' '};
    Ergebnis{1,4}(i,1) = 0;
else
    disp('Fehler2')
end

Factor_PodHullInteractTable1 =
reshape(Factor_PodHullInteractTable1, [], 3);

% One example from file Example_Matrix_2018-05-06, if the loop
run successfully it will show the result as
Factor_PodHullInteractTable1= [1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0
0.0 1.0 0.0 0.0 1.0 1.0 1.0 1.0 1.0].
% Then if the "Factor_PodHullInteractTable1" would be read in
successfully, the following necessary processing for
"Factor_PodHullInteractTable1" creation would be:

```



```
Angle_PodHullInteractTable1= unique(Angle_PodHullInteractTable1)
and it will create a result as
Factor_PodHullInteractTable1=[1,1,0,0,0,0,0,0,1,1].
```

if statement for zeilen, spalten und data

```
Zeilen1 = Angle_PodHullInteractTable1;

Spalten1 = RPM_PodHullInteractTable1;

Data1 = Factor_PodHullInteractTable1;

% Creating Zeilen, Splalten and Data

%Zeilen
if Angle_PodHullInteractTable1 == 0;
    Zeilen1 = [-180 180];
elseif Angle_PodHullInteractTable1 > 0;
    Zeilen1 = Angle_PodHullInteractTable1\';
end

%Spalten
if RPM_PodHullInteractTable1 == 0;
    Spalten1 = [-100 100];
elseif RPM_PodHullInteractTable1 > 0;
    Spalten1 = RPM_PodHullInteractTable1;
end

%Data
if Factor_PodHullInteractTable1 == 0;
    Data1 = [1 1;1 1];
elseif Factor_PodHullInteractTable1 > 0;
    Data1 = Factor_PodHullInteractTable1;
end
```

“This Page Intentionally Left Blank”

Appendix 6: Example of Vector Data

```

14 0 Length 142.0 1
64 0 WindResistCoeff_Cx_List **@List@Definition@** 1
64 1 WindResistCoeff_Cx_List 0.715 1
64 2 WindResistCoeff_Cx_List 0.810 1
64 3 WindResistCoeff_Cx_List 0.238 1
64 4 WindResistCoeff_Cx_List 0.143 1
64 5 WindResistCoeff_Cx_List 0.048 1
64 6 WindResistCoeff_Cx_List 0.0 1
64 7 WindResistCoeff_Cx_List -0.048 1
64 8 WindResistCoeff_Cx_List -0.143 1
64 9 WindResistCoeff_Cx_List -0.238 1
64 10 WindResistCoeff_Cx_List -0.738 1
64 11 WindResistCoeff_Cx_List -0.715 1
64 12 WindResistCoeff_Cx_List -0.738 1
64 13 WindResistCoeff_Cx_List -0.238 1
64 14 WindResistCoeff_Cx_List -0.143 1
64 15 WindResistCoeff_Cx_List -0.048 1
64 16 WindResistCoeff_Cx_List 0.0 1
64 17 WindResistCoeff_Cx_List 0.048 1
64 18 WindResistCoeff_Cx_List 0.143 1
64 19 WindResistCoeff_Cx_List 0.238 1
64 20 WindResistCoeff_Cx_List 0.810 1
64 21 WindResistCoeff_Cx_List 0.715 1

```

“This Page Intentionally Left Blank”

Appendix 7: Example of Matrix Data

```

1518 0 Angle_PodHullInteractTable1 **@List@Definition@** 1
1518 1 Angle_PodHullInteractTable1 -180.0 1
1518 2 Angle_PodHullInteractTable1 -180.0 1
1518 3 Angle_PodHullInteractTable1 -180.0 1
1518 4 Angle_PodHullInteractTable1 -155.0 1
1518 5 Angle_PodHullInteractTable1 -155.0 1
1518 6 Angle_PodHullInteractTable1 -155.0 1
1518 7 Angle_PodHullInteractTable1 -145.0 1
1518 8 Angle_PodHullInteractTable1 -145.0 1
1518 9 Angle_PodHullInteractTable1 -145.0 1
1518 10 Angle_PodHullInteractTable1 -125.0 1
1518 11 Angle_PodHullInteractTable1 -125.0 1
1518 12 Angle_PodHullInteractTable1 -125.0 1
1518 13 Angle_PodHullInteractTable1 -115.0 1
1518 14 Angle_PodHullInteractTable1 -115.0 1
1518 15 Angle_PodHullInteractTable1 -115.0 1
1518 16 Angle_PodHullInteractTable1 180.0 1
1518 17 Angle_PodHullInteractTable1 180.0 1
1518 18 Angle_PodHullInteractTable1 180.0 1
1522 0 RPM_PodHullInteractTable1 **@List@Definition@** 1
1522 1 RPM_PodHullInteractTable1 -100.0 1
1522 2 RPM_PodHullInteractTable1 50.0 1
1522 3 RPM_PodHullInteractTable1 100.0 1
1522 4 RPM_PodHullInteractTable1 -100.0 1
1522 5 RPM_PodHullInteractTable1 50.0 1
1522 6 RPM_PodHullInteractTable1 100.0 1
1522 7 RPM_PodHullInteractTable1 -100.0 1
1522 8 RPM_PodHullInteractTable1 50.0 1
1522 9 RPM_PodHullInteractTable1 100.0 1
1522 10 RPM_PodHullInteractTable1 -100.0 1
1522 11 RPM_PodHullInteractTable1 50.0 1
1522 12 RPM_PodHullInteractTable1 100.0 1
1522 13 RPM_PodHullInteractTable1 -100.0 1
1522 14 RPM_PodHullInteractTable1 50.0 1
1522 15 RPM_PodHullInteractTable1 100.0 1
1522 16 RPM_PodHullInteractTable1 -100.0 1
1522 17 RPM_PodHullInteractTable1 50.0 1
1522 18 RPM_PodHullInteractTable1 100.0 1
1526 0 Factor_PodHullInteractTable1 **@List@Definition@** 1
1526 1 Factor_PodHullInteractTable1 1.0 1

```

1526 2 Factor_PodHullInteractTable1 1.0 1
1526 3 Factor_PodHullInteractTable1 1.0 1
1526 4 Factor_PodHullInteractTable1 1.0 1
1526 5 Factor_PodHullInteractTable1 1.0 1
1526 6 Factor_PodHullInteractTable1 1.0 1
1526 7 Factor_PodHullInteractTable1 1.0 1
1526 8 Factor_PodHullInteractTable1 0.0 1
1526 9 Factor_PodHullInteractTable1 0.0 1
1526 10 Factor_PodHullInteractTable1 1.0 1
1526 11 Factor_PodHullInteractTable1 0.0 1
1526 12 Factor_PodHullInteractTable1 0.0 1
1526 13 Factor_PodHullInteractTable1 1.0 1
1526 14 Factor_PodHullInteractTable1 1.0 1
1526 15 Factor_PodHullInteractTable1 1.0 1
1526 16 Factor_PodHullInteractTable1 1.0 1
1526 17 Factor_PodHullInteractTable1 1.0 1
1526 18 Factor_PodHullInteractTable1 1.0 1

AUTHOR'S BIOGRAPHY



The Author's name is Rika Citra, born on 06 September 1996 in Jakarta. As the first child of two. Derived from a simple family with a Father named Suwito and Mother named Ribut Riawati. However, fortunate to have a formal education at SDI Al-Azhar Kelapa Gading Jakarta and also, she continued her study at SMPI Al-Azhar Kelapa Gading Jakarta, and SMAN 68 Jakarta. In 2014, author proceed to pursue bachelor degree at Department of Marine Engineering (Double Degree Program with Hochschule Wismar), Faculty of Marine Engineering, Institut Teknologi Sepuluh Nopember Surabaya. During the study period, Author did activity in campus organization, HIMASIKAL (2014-2015). The Autor also joined in several event organizers such as Consumption Committee of Marine Icon (2014-2016), Decoration Committee of Homecoming Day 2014, and Head Committee of Malam Guyub 2015.