

Shubham Garg

Design and Validation of a Course Controller for a Wave Powered Vehicle Using NMPC

Masteroppgave i Marine and Maritime Intelligent Robotics

Veileder: Asgeir Johan Sørensen

Medveileder: Tor Arne Johanssen

Januar 2022

Shubham Garg

Design and Validation of a Course Controller for a Wave Powered Vehicle Using NMPC

Masteroppgave i Marine and Maritime Intelligent Robotics
Veileder: Asgeir Johan Sørensen
Medveileder: Tor Arne Johanssen
Januar 2022

Norges teknisk-naturvitenskapelige universitet
Fakultet for ingeniørvitenskap
Institutt for marin teknikk



Kunnskap for en bedre verden

Design and Validation of a Course Controller for a Wave Powered Vehicle Using NMPC

Shubham Garg

June 8, 2023

Abstract

The emergence of a new class of green-energy-powered vehicles has introduced the capability to conduct large-scale spatiotemporal surveys without the need for energy-intensive motored propulsion. These vehicles can have endurance lasting months, thanks to their reliance on environmental forces for propulsion. However, this reliance on external forces makes them susceptible to loss of controllability, particularly in adverse weather conditions. As a consequence, there is a need for increased robustness in the onboard autonomy system to ensure the safe and effective operation of these vehicles at sea.

We propose using an Nonlinear Model Predictive Control based course control system to ensure a stable course over the entire operating region of the vehicle. We derived three system models with varying levels of fidelity along with two objective functions χ_d and \dot{v} to compute the optimal control input for the vehicle. The developed controllers were tested in simulation using both Model-In-the-Loop and Hardware-In-the-Loop methods.

The findings of the study demonstrated proficiency in maintaining course accuracy. The controller achieved a zero steady-state error under ideal conditions and was robust to noise and disturbances. In extreme situations, as the Speed Over Ground approached zero, the controller determined the optimal trajectory allowing for a stable course and minimum oscillations. The real-time performance was validated by integrating the controller with the onboard autonomy stack on an embedded platform using the IPOPT solver. The solver computation time varied, but consistently remained within the specified threshold, demonstrating the feasibility of online operations.

Sammendrag

Fremveksten av en ny klasse kjøretøyer med grønn energi har introdusert muligheten til å gjennomføre storskala undersøkelser av rom og tid uten behov for energikrevende motordrevet fremdrift. Disse kjøretøyene kan ha utholdenhet som varer i måneder, takket være deres avhengighet av miljøkrefter for fremdrift. Imidlertid gjør denne avhengigheten av eksterne krefter dem utsatt for tap av kontrollerbarhet, spesielt under ugunstige værforhold. Som en konsekvens er det behov for økt robusthet i autonomisystemet ombord for å sikre sikker og effektiv drift av disse kjøretøyene til sjøs.

Vi foreslår bruk av et Nonlinear Model Predictive Control-basert kurskontrollsystem for å sikre stabil kurs over hele kjøretøyets driftsområde. Vi utledet tre systemmodeller med varierende grad av troskap sammen med to objektive funksjoner χ_d og $\dot{\nu}$ for å beregne den optimale kontrollinngangen for kjøretøyet. De utviklede kontrollerene ble testet i simulering ved bruk av både Model-In-the-Loop- og Hardware-In-the-Loop-metoden.

Funnene fra studien demonstrerte ferdigheter i å opprettholde kursnøyaktighet. Kontrolleren oppnådde null steady state-feil under ideelle forhold og var robust mot både støy og forstyrrelser. Under ugunstige forhold, når Speed Over Ground nærmet seg null, bestemte kontrolleren den optimale banen som tillot stabil kurs og minimumssvingninger. Sanntidsytelsen ble validert ved å implementere kontrolleren på en innebygd plattform ved å bruke IPOPT-løseren. Løserberegningstiden varierte, men holdt seg konsekvent innenfor den spesifiserte terskelen, noe som demonstrerer muligheten for online operasjoner.

Contents

Abstract	iii
Sammendrag	v
Contents	vii
Figures	ix
Tables	xi
Acronyms	xiii
1 Introduction	1
1.1 Motivation	2
1.2 Literature Review	2
1.3 Scope	3
1.4 Key Aims and Objectives	3
1.5 Outline of Report	4
2 Background	5
2.1 Introducing the Vehicle: AutoNaut	5
2.2 PID Based Course Control System	7
2.3 Gain Scheduling Based Course Control System	7
2.4 Speed Prediction using Data Regression	8
3 Theory	9
3.1 System Modelling	9
3.1.1 Review of Principles of Dynamics	9
3.1.2 Generalized Nonlinear Dynamic Model in 3DOFs	13
3.1.3 Generalized nonlinear dynamic model in 2DOF	16
3.1.4 Generalized linear dynamic model in 3DOF	17
3.2 State Space Modeling	18
3.2.1 Linearized Dynamics in 3DOFs	19
3.2.2 Nonlinear Dynamics in 3DOFs	19
3.2.3 Nonlinear Dynamics in 2DOFs	20
3.3 Control Objective	21
3.4 Nonlinear Model Predictive Control	22
3.4.1 Definition	22
3.4.2 Dynamic Model and Discretization	24
3.4.3 Objective Function	27
3.4.4 Constraints	28
3.4.5 Problem Formulation	30

3.4.6	Sub-optimal NMPC	32
3.4.7	Tuning	33
4	Method	35
4.1	Software Framework: CasADi	35
4.1.1	Introduction	35
4.1.2	Symbolic Framework in CasADi	35
4.1.3	Interior Point Optimizer Solver	37
4.2	Model-In-the-Loop Simulation	37
4.2.1	Introduction	37
4.2.2	Simulation Design	38
5	Implementation	41
5.1	Software Package: acados	41
5.1.1	Why acados	41
5.1.2	Algorithm Implementations in acados	42
5.1.3	Workflow with a High-Level Language Interface	42
5.2	Software Toolchain: LSTS/DUNE	45
5.2.1	Introduction	45
5.2.2	Overview of the System Architecture	48
5.3	Hardware-In-the-Loop Simulation	49
5.3.1	Integration with onboard System	49
5.3.2	Hardware Setup	51
6	Results	53
6.1	Test Plan	53
6.2	Results from MIL Simulation	55
6.2.1	Ideal Conditions	55
6.2.2	Nominal Conditions with Noisy Observer	57
6.2.3	Adverse Conditions with Relaxed Constraints	60
6.3	Results from HIL Simulation	63
6.3.1	Nominal Conditions with Noisy Observer	63
7	Discussions	65
7.1	MIL Simulation	65
7.1.1	Ideal Conditions	66
7.1.2	Nominal Conditions with Noisy Sensors	67
7.1.3	Adverse Conditions with Soft Constraints	68
7.2	HIL Simulation	71
7.2.1	Nominal Conditions with Noisy Sensors	71
7.3	Future Work	73
8	Conclusion	75
	Bibliography	77
A	AutoNaut USV model parameters	85
B	Code	87
C	Additional Plots	89
C.1	Results from MIL Simulation	89

Figures

2.1	AutoNaut 5m Version. Credits: AutoNaut Documentation Wiki, NTNU	5
3.1	The three body-fixed DOFs and their interpretation in NED frame	11
4.1	Example NLP problem	36
4.2	Flowchart of the MIL simulation	38
5.1	acados Workflow using a high-level scripting language.	42
5.2	Course Angle using Acados Solver	43
5.3	Input Rudder Angle using Acados Solver	43
5.4	Example architecture implementation and possible switching between active/inactive controllers. Image taken from [56]	46
5.5	Neptus Operator Console. Image taken from [58]	47
5.6	Overview of System Design	49
5.7	Flowchart of the class NmpcDynamics and NmpcCourse	50
5.8	Setup for HIL Simulation	51
6.1	Summary of results in ideal conditions	55
6.2	Vehicle path in ideal conditions	56
6.3	Course angle in ideal conditions	56
6.4	Input rudder angle in ideal conditions	56
6.5	Solver computation time in ideal conditions	57
6.6	Summary of results in nominal conditions	57
6.7	Path in nominal conditions	58
6.8	Course angle in nominal conditions	58
6.9	Input rudder angle in nominal conditions	58
6.10	Solver computation time in nominal conditions	59
6.11	Speed Over Ground in nominal conditions	59
6.12	Sideslip angle in nominal conditions	59
6.13	Summary of results in adverse conditions	60
6.14	Path in adverse conditions	61
6.15	Course angle in adverse conditions	61
6.16	Input rudder angle in adverse conditions	61
6.17	Solver computation time in adverse conditions	62
6.18	Speed Over Ground in adverse conditions	62

6.19 Crab angle in adverse conditions	62
6.20 nonlinear4 with chi_d	63
6.21 nonlinear4 with dotv	64
6.22 linear4 with dotv	64
C.1 Heading angle in ideal conditions	89
C.2 SOG in ideal conditions	90
C.3 Surge speed in ideal conditions	90
C.4 Sway speed in ideal conditions	90
C.5 Sideslip angle in ideal conditions	91
C.6 Heading angle in nominal conditions	91
C.7 Surge speed in nominal conditions	92
C.8 Sway speed in nominal conditions	92
C.9 Heading angle in adverse conditions	93
C.10 Surge speed in adverse conditions	93
C.11 Sway speed in adverse conditions	93

Tables

2.1	Specifications for AutoNaut USV operated by NTNU	6
3.1	The notation of SNAME for marine craft. Credits: Handbook of Marine Craft Hydrodynamics and Motion Control	10
6.1	Default simulation parameters	54
6.2	Modified parameters for ideal conditions	55
6.3	Modified parameters for nominal conditions	57
6.4	Modified parameters for adverse conditions	60
A.1	Vehicle Parameters	85
A.2	Hydrodynamic Parameters	85
A.3	Rudder Parameters	86
A.4	Wind Model Parameters	86

Acronyms

- ADCP** Acoustic Doppler Current Profiler. 6, 48
- C2** Command and Control. 45, 47
- CG** Center of Gravity. 14
- DOF** Degree of Freedom. vii, 9, 11–13, 17, 20, 39, 49, 67
- DOFs** Degree of Freedoms. vii, ix, 11, 13, 17–20
- DUNE** DUNE Unified Navigation Environment. 45–47
- GNC** Guidance-Navigation-Control. 2
- GNSS** Global Navigation Satellite System. 7, 8, 12, 48
- HIL** Hardware-In-the-Loop. iii, v, viii, 3, 49, 53, 63, 71, 72, 75
- IMC** Inter-Module Communication. 45–49
- IPOPT** Interior Point Optimizer. iii, v, viii, 37, 45, 72, 75
- LSTS** Laboratório de Sistemas e Tecnologia Subaquática. 45, 46
- LTI** Linear time-invariant. 17
- MIL** Model-In-the-Loop. iii, v, viii, ix, 3, 4, 35, 37–39, 53, 55, 65, 71, 75, 89
- MPC** Model Predictive Control. 2, 22–25, 28, 29, 44
- NED** North-East-Down. ix, 9–12
- NLP** NonLinear Programming. 30, 31, 35, 36, 43
- NMPC** Nonlinear Model Predictive Control. iii, v, 2–4, 8, 9, 22, 31, 32, 38, 39, 41, 43, 50, 70, 72, 73, 75
- NTNU** Norges Teknisk-Naturvitenskapelige Universitet. 1, 6, 41

OCP Optimal Control Problem. 27, 29–31, 33, 35, 38, 41–45

ODE Ordinary Differential Equation. 25, 26, 36

ODEs Ordinary Differential Equations. 24, 35

OOP Object Oriented Programming. 49

PID Proportional-Integral-Derivative. 3

QP Quadratic Programming. 36, 42, 44

RK4 Runge-Kutta 4. 25, 26

RMSE Root Mean Square Error. 65, 67, 70, 72

RTI Real Time Iteration. 42

SOG Speed Over Ground. iii, v, ix, x, 8, 12, 44, 59, 60, 62, 68, 69, 72, 75, 90

SQP Sequential Quadratic Programming. 42, 72, 75

USV Unmanned Surface Vehicle. 5, 7

Chapter 1

Introduction

Water, the namesake of our blue planet, is believed to provide the crucial element that makes life possible on Earth. It regulates the Earth's climate and weather patterns and is a major source of food and oxygen for the planet. Historically, it has also played a significant role in accelerating humanity's economic and social growth [1]. It is no surprise then that unchecked exploitation of ocean resources over time has led to erratic weather events and long-term harm to ocean biochemistry. To safeguard this valuable resource, persistent and sustainable ocean monitoring is essential.

This has been realized by scientists leading to science-driven ocean monitoring of the upper water column becoming a mature field, as demonstrated in [2], [3] and [4]. However, most current state-of-the-art technologies make use of one or a combination of the three methods, which are (1) Ship-based sampling, (2) Satellite-based remote sensing and (3) Unmanned robotic platforms. Ship-based methods are not sustainable and do not scale well across space and time. Satellites on the other hand lack the fine resolution required for detailed studies. Finally, most robotic platforms currently in use are highly constrained due to their complexity, lack of autonomy or onboard energy limitations [5].

Investigative studies and surveys over large space-time scales require sustained operations while leaving a minimal footprint over the ecosystem being studied. In these situations, a new class of underwater or surface gliders are seen to be bridging the gap[6]. These are low-cost unmanned crafts powered by green energy. This makes it possible to carry out long missions without the need for physical human intervention. The vessel in use for this report is a surface glider called AutoNaut, developed in England by AutoNaut Ltd [7]. The autonomy system used in our 5m version of the commercially available AutoNaut was designed and developed at the Norges Teknisk-Naturvitenskapelige Universitet (NTNU), as described in [8].

1.1 Motivation

Long-term sustained and unattended autonomous operations require a robust approach toward the onboard autonomy system. We want to minimize deviations from the desired behavior and show resilience to varying environmental disturbances, design limitations, and systemic failures. This enforces additional constraints on the Guidance-Navigation-Control (GNC) system of the vessel that must account for different sea states, winds, and currents over the entire duration of the mission. This is further complicated by the limited control offered by vehicles that depend on the environmental forces for propulsion, such as the one being considered here.

A capable autonomy system would allow for uninterrupted and self-sufficient studies, surveys, and monitoring with minimal impact on the ecosystem, even in adverse weather conditions. This would greatly enhance the efficiency and reliability of the data collected while reducing the need for expensive human intervention. As conventional control strategies can not guarantee maneuverability and a stable course over the ground in all weather conditions [9], our research aims to explore Nonlinear Model Predictive Control (NMPC) techniques that consider the consequences of lack of motored propulsion and adverse weather conditions to ensure safe operations on the high seas.

1.2 Literature Review

Model Predictive Control (MPC) is a mature subject with well established principles and literature to refer to. A good starting point for learning about MPC technology can be [10]. It introduces both linear and nonlinear MPC while touching upon fields of system stability, the feasibility of the solution, estimation, and algorithmic implementation of a linear MPC controller. The reader may also be interested in [11] as it describes in depth how a linear MPC controller for state regulation or target tracking may be implemented in MATLAB™ using only basic built-in functions. For further literature on linear MPC, the reader may refer to the review papers [12] and [13].

For Nonlinear Model Predictive Control (NMPC), we refer to [14] and [15]. They form a comprehensive resource to discuss online NMPC-based control systems and include discussion on problem formulation, implementation through numerical methods, problem feasibility, and stability analysis for nonlinear systems. Finally, our implementation of the NMPC controller using the CasADi toolbox was inspired by [16].

1.3 Scope

In [9] and [17], the authors introduced and validated a robust PID-based course control strategy capable of sustained operations in calm waters with increasing instabilities as the ground speed of the vehicle approached zero. In our research, we aim to use optimal- and predictive control-based strategies to improve the course-keeping performance at very low or near-zero ground speeds.

The scope of this report is to design and implement a viable NMPC-based course controller for a wave-powered vehicle for operation in adverse weather conditions. Additionally, as the vehicle is anticipated to undertake operations lasting several weeks, the proposed solution's feasibility and robustness will be validated. The controller performance will be analyzed using Model-In-the-Loop (MIL) simulation in the numeric computing environment MATLABTM and real-life feasibility is studied using Hardware-In-the-Loop (HIL) simulation on an embedded platform.

We restrict the scope to NMPC-based controllers and do not explore alternate nonlinear and adaptive control strategies. Further, we do not discuss lower-level actuator control systems and higher-level systems that provide guidance references. We note that both linear and nonlinear state observer and parameter estimators may offer significant benefits by accounting for model inaccuracies however, they've been excluded due to time limitations. Finally, we assume that the reader has sufficient knowledge of the basics of modeling marine crafts and systems and Proportional-Integral-Derivative (PID) control.

Finally, we note that this thesis project is a continuation of the work done as part of the master course "TMR4510 - Marine Control Systems, Specialization Project".

1.4 Key Aims and Objectives

In this project, we aim to design a NMPC-based control system to minimize deviations in the cardinal direction in which AutoNaut is moving, also known as the course angle. To this end, the key objectives of the report are listed as follows:

1. Develop a model that captures the system dynamics and accounts for the nonlinear effects of passive actuators like the rudder, and the wave propulsion system, as well as external disturbances like winds and currents.
2. Develop a NMPC-based algorithm that can minimize the error in course angle, taking into account the real-time environmental conditions.
3. Evaluate the performance of NMPC-based controller(s) in comparison to conventional control solutions and under different environmental conditions.
4. Evaluate the feasibility of implementing the controller on an embedded platform and its ability to perform online optimization.

1.5 Outline of Report

The report is organized into seven chapters. Chapter 1 and 2 establish the context of the study while providing a balanced review of the pertinent existing work done on this subject. Chapter 3 lays out the necessary theoretical background on the system dynamics and the design of the control system. Chapter 4 explains the implementation of the proposed NMPC controller(s) in MATLAB™ and prepares the MIL simulation. Chapter 5 discusses the hardware implementation of the controller and integration with the existing framework onboard the vehicle. Results and discussions are laid out in Chapter 6 and 7. Finally, a conclusion and topics of additional interest are given in Chapter 8.

Chapter 2

Background

The chapter provides an overview of the existing studies and previous research done on designing a course controller for AutoNaut. By examining the existing state-of-the-art, we aim to identify specific shortcomings and gain insights into the proposed research direction.

2.1 Introducing the Vehicle: AutoNaut

AutoNaut is an Unmanned Surface Vehicle (USV) developed by AutoNaut Ltd [7] and is the focus of this study. Its unique hull design enables it to efficiently move through the water while capturing wave energy, providing significant advantages over traditional fuel-powered vehicles. Its renewable energy source allows for extended operations on the open ocean without the need for refueling, making it more environmentally friendly and sustainable. Furthermore, the ability of wave-powered vehicles like AutoNaut to access remote or challenging areas of the ocean can offer unparalleled opportunities for research and monitoring.

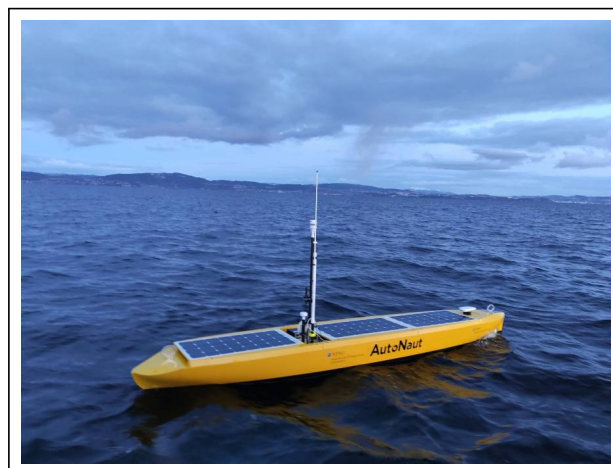


Figure 2.1: AutoNaut 5m Version. Credits: AutoNaut Documentation Wiki, NTNU

AutoNaut’s patented Wave Foil technology consists of four keel-mounted foils and utilizes the energy from the pitch and roll motions of the hull in waves to generate a forward thrust [8]. These spring-loaded foils are attached to the struts beneath the keel and utilize the lifting and dropping motion of the vehicle motion to generate forward propulsion. This technique was first developed by Linden and Einar Jacobsen as described in [18] and [19].

The majority of the oceans experience continuous weather patterns ensuring that waves (and hence propulsion) are almost always present. Even in relatively flat water, AutoNaut will move with a speed of half a knot due to the presence of wave undulation. However, waters near shore and complete flat conditions are problematic as the propulsion system is not able to generate significant thrust. Under such circumstances, AutoNaut may make use of the onboard electric thruster to generate forward thrust.

This project considers the use of the AutoNaut USV owned and operated by NTNU, which is a modified version of the commercial off the shelf vehicle. Specifically, the onboard autonomy system and the relevant hardware required for the same were developed in-house as described in [8]. The vehicle is depicted in fig. 2.1 and its specifications are listed in tbl. 2.1.

Table 2.1: Specifications for AutoNaut USV operated by NTNU

Parameter	Value
Hull Type	Monohull
Length	5 m
Beam	0.8 m
Draft	0.7 m
Displacement	230 Kg
Average Speed	Upto 2 Knots
Endurance	3+ Months; dependent on hull fouling
Transportation	Two-person portable, air freight
Propulsion	Wave foil technology, Electric Propulsion
Battery Capacity	4 × 70Ah 12V Lead Gel batteries
Solar Charging	300 Watt peak photovoltaic solar panels
Payload Volume	500 Litres
Payload Weight	130 Kg
Sensors	Marine GPS, Airmar weather station, Active radar transponder, Acoustic Doppler Current Profiler

2.2 PID Based Course Control System

One of the earlier works done on this subject is described in [9] where the authors study the model nonlinearities in the presence of adverse weather conditions to design a robust course control system for the AutoNaut. The paper lists the main challenges associated with this new class of vehicles and examines conditions where environmental disturbances exceed the steering and propulsion forces, resulting in reduced control and maneuverability. Their investigation leads to the derivation of a simple quasi-linear mathematical model which provides the basis for the design of a PID-based course controller. The controller is also validated through analysis and field trials.

The developed autopilot is a proportional-integral (PI) controller that has the capability to compensate for constant or gradually changing wind and current disturbances. Its robust tuning was optimized for conditions where the ground speed is at least 0.2 m/s. The experimental results were validated in Trondheim fjords and in the North Atlantic Ocean. However, in situations where the ground speed dropped below 0.2 m/s and approached zero, the vehicle experienced instability and loss of control.

The experiments revealed that as environmental forces exceeded the propulsion forces, the vehicle experienced significant oscillations in both the course and the rudder. A provisional solution to switch to heading control was proposed at the expense of significant steady-state course error.

2.3 Gain Scheduling Based Course Control System

The previous study introduced the parameter γ , which captured the main nonlinearities present in the system and highlighted some of the challenges associated with using this parameter in practice. These limitations include uncertainty in the value of the parameter γ which depends on ocean currents that may not be known, and the effect of unreliable course angle measurements from the GNSS when the USV's speed approaches zero. Despite the above limitations, γ proved to be a useful tool for designing a gain scheduling control strategy as shown in [17].

Gain Scheduling is a control strategy for nonlinear systems that uses a family of linear controllers, each designed to operate around a specific operating point [20]. The linear controllers are tuned using well-established linear control theory to provide satisfactory performance around their respective operating points. These operating points are characterized by a scheduling variable, which, in the case of this study is γ . The main contribution of this paper is the design and implementation of a gain scheduling-based course control system for a USV operating at low speeds.

The proposed controller(s) offered significant improvements over the existing controller as it reduced oscillations in both the course and rudder at low speeds. However, the controller was not able to eliminate them entirely and did not ad-

dress the singularity present when Speed Over Ground (SOG) becomes zero and the course angle becomes undefined. The model analysis also showed poor course tracking ability at low relative velocities due to the smaller bandwidth of the controller, compared to the expected frequency of the course reference. Therefore, the need for a more robust control strategy for very low relative velocities remains.

2.4 Speed Prediction using Data Regression

AutoNaut leverages renewable energy from the environment to undertake extended missions but its reliance on the environment renders it vulnerable to potential loss of controllability in adverse weather conditions. These unfavorable effects can be mitigated by careful mission planning to maximize the vehicle's SOG over the mission duration while ensuring compliance with mission objectives that may include time-sensitive transient events. However, accurately predicting the speed of the vehicle in varying weather conditions poses significant challenges.

The authors employed regression techniques first introduced in [21], [22], and [23] to develop a speed prediction model for AutoNaut. The model made use of environment properties of wind and currents obtained using onboard sensors and wave profile obtained using estimates from GNSS data as features in the prediction model. The feature vector θ along with the regression model g considered is expressed in Eq. 2.1

$$\begin{aligned} \theta &= [H_s \quad \omega_p \quad \cos \gamma_p \quad V_w \cos(\beta_w - \psi) V_c \cos \beta_c \quad 1] \\ g &: \mathbb{R}^7 \rightarrow \mathbb{R} \\ y &= g(\theta) = w^T \theta \end{aligned} \tag{2.1}$$

where w is the vector containing set of weights and $y \in \mathbb{R}$ is the speed of the vehicle.

The experimental results show good accuracy when the vehicle is operating in the speed range of 0.2m/s - 0.8m/s with performance degrading at both ends of the spectrum due to limited availability of data. Further, features not captured by the model such as the directional spread of the waves and local variation reduce model accuracy. Regardless, the proposed model still proves to be a reliable method of predicting the speed of the vehicle in the open ocean under normal operating conditions.

Since precise dynamics of the propulsion system are inherently complex, being able to approximate the vehicle speed allows us to develop credible models for the surge dynamics of AutoNaut. This has relevance in developing plant models for use in the NMPC controller.

Chapter 3

Theory

This chapter presents the theoretical framework for our proposed solution. It starts with a review of the fundamental concepts of dynamic modeling, which form the basis for deriving both linear and nonlinear models for the AutoNaut. Subsequently, we introduce our proposed objective functions aimed at minimizing course error, along with the application of NMPC theory that enables their implementation.

3.1 System Modelling

3.1.1 Review of Principles of Dynamics

In this section, we review the fundamental concepts of dynamics, which can be divided into two distinct fields: kinematics, which focuses on the motion and geometry of objects, and kinetics, which examines the forces and torques affecting objects in motion.

Vehicle Kinematics

In order to provide a solid foundation for our subsequent discussion, we introduce key fundamental concepts that are directly relevant to our work.

Motion Variables: The motion of a marine craft moving in six Degree of Freedom (DOF)s can be described by six independent coordinates that determine its position and orientation. In our case, the position is defined in North-East-Down (NED) coordinates while orientation is described by Euler Angles which include roll, pitch, and yaw. These Euler angles represent the orientation of the craft's body-fixed reference frame (BODY) in relation to the NED frame. In the case of marine craft, the six motion components in the BODY frame are referred to as surge, sway, heave, roll, pitch, and yaw. However, since AutoNaut operates on the water's surface, our focus is primarily on 3DOFs, namely surge, sway, and yaw as depicted in tbl. 3.1.

Table 3.1: The notation of SNAME for marine craft. Credits: Handbook of Marine Craft Hydrodynamics and Motion Control

DOF		BODY		NED
		Forces and Moments	Linear and Angular Velocities	Position and Euler Angles
1	Motion in the x_b -axis (surge)	X	u	x^n
2	Motion in the y_b -axis (sway)	Y	v	y^n
3	Rotation about the z_b -axis (yaw)	N	r	ψ

Reference Frames The motion of the marine craft is analyzed in the context of geographical reference frames. The reference frames relevant to our discussion are described below

1. **NED:** The North-East-Down coordinate system is denoted with $\{n\} = \{x_n, y_n, z_n\}$ where

x_n -axis points towards the true North.

y_n -axis points towards East.

z_n -axis points downwards normal to the Earth's surface.

2. **BODY:** The body-fixed coordinate system is denoted with $\{b\} = \{x_b, y_b, z_b\}$ with its origin o_b fixed to a point on the craft. The axis of the frame coincides with the principle axis of inertia and is defined as

x_b -axis is directed from aft to fore.

y_b -axis is directed to starboard.

z_b -axis is directed from top to bottom.

Reference Points A reference point is a defined location on the craft along which the equations of motion are expressed. The primary reference point is C_0 , which serves as the coordinate origin for both the body-fixed frame and the GNC systems.

The motion variables along with their respective reference frame and reference points are depicted in fig. 3.1.

Generalized Coordinates describe the configuration of the craft relative to some reference configuration. Generalized coordinates can be used to express both pose (position and orientation) and twist (linear and angular velocities) in both NED and body-fixed frames. For AutoNaut, the generalized coordinates are

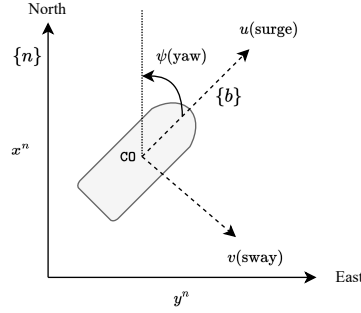


Figure 3.1: The three body-fixed DOFs and their interpretation in NED frame

expressed as

$$\begin{aligned}
 &\text{generalized position in } \{n\} \\
 &\quad \eta = [x^n \quad y^n \quad \psi] \\
 &\text{generalized velocity in } \{n\} \\
 &\quad \dot{\eta} = [\dot{x}^n \quad \dot{y}^n \quad r] \\
 &\text{generalized velocity in } \{b\} \\
 &\quad \nu = [u \quad v \quad r]
 \end{aligned} \tag{3.1}$$

Ocean currents are the continuous and directional movement of ocean waters produced by effects of gravity, wind friction, water density, etc. In our model, we consider the effects of constant and irrotational currents expressed as

$$\begin{aligned}
 &\text{generalized ocean current velocity in } \{b\} \\
 &\quad \nu_c = [u_c \quad v_c \quad 0] \\
 &\quad \dot{\nu}_c = [r v_c \quad -r u_c \quad 0]
 \end{aligned} \tag{3.2}$$

Relative velocity vector represents the velocity of the vehicle in three DOF relative to the water. This is expressed as

$$\begin{aligned}
 &\text{generalized relative velocity in } \{b\} \\
 &\quad \nu_r = \nu - \nu_c
 \end{aligned} \tag{3.3}$$

Transformations between BODY and NED frames can be expressed using the help of rotation matrices, denoted by $R_{a_1}^{a_2}$ where the rotation is taking place from $\{a_1\}$ to $\{a_2\}$. In case of 3DOFs transformation, the rotation from $\{b\}$ to $\{n\}$ can be expressed as

$$R_b^n = R(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

A frequently used application relevant to our work is the transformation of body-fixed frame velocity to NED frame velocity vector as

$$\dot{\eta} = R(\psi)v \quad (3.5)$$

We note that since rotation matrices are skew-symmetric, the inverse transformation can be obtained by simple transposition.

Definitions of Heading, Course and Crab Angles Understanding the interplay between the angular variables of course, heading, and sideslip is significant in maneuvering the AutoNaut within the horizontal plane. Their definitions, as noted in [24] are given below

Heading Angle The angle ψ from the x_n -axis (true North) to the x_b -axis of the craft, positive rotation about the z_n -axis by the right-hand screw convention. The heading angle is usually measured by using a magnetic, gyro, or a GNSS compass.

Crab Angle The angle β_c from the x_b -axis to the velocity vector of the craft, positive rotation about the z_b -axis by the right-hand screw convention. Mathematically, it is expressed as

$$\beta = \arcsin\left(\frac{v}{U}\right) = \arctan\left(\frac{v}{u}\right) \quad (3.6)$$

where $U = \sqrt{v^2 + u^2}$ is the Speed Over Ground (SOG) of the vehicle.

Course Angle The angle χ from the x_n axis (true North) to the velocity vector of the craft, positive rotation about the z_n axis by the right-hand screw convention. Mathematically, it is expressed as

$$\chi = \psi + \beta \quad (3.7)$$

Vehicle Kinetics

Using maneuvering theory as described in [24], the motion of a marine craft in six DOFs with the assumption of frequency-independent hydrodynamic forces can be expressed in matrix-vector format as

$$\mathbf{M}\dot{v}_r + \mathbf{C}(v_r)v_r + \mathbf{D}(v_r)v_r + g(\eta) + g_0 = \tau + \tau_{\text{wind}} + \tau_{\text{wave}} \quad (3.8)$$

where

- J_{Θ} is the BODY-NED transformation matrix.
- $\mathbf{M} = \mathbf{M}_{\text{RB}} + \mathbf{M}_{\text{A}}$ is the system inertia matrix
- $\mathbf{C}(v_r) = \mathbf{C}_{\text{RB}}(v_r) + \mathbf{C}_{\text{A}}(v_r)$ is the Coriolis-centripetal matrix
- $\mathbf{D}(v_r)$ is the damping matrix

- $g(\eta)$ is the forces and moment vector due to hydrostatics
- g_0 is the vector for pretrimming (ballast control).
- τ is the vector of control inputs.
- τ_{wind} is the vector of generalized wind forces.
- τ_{wave} is the vector of generalized wave-induced forces.

The model described above is useful for designing a variety of model-based control systems. However, unless additional assumptions and simplifications are made, the model is highly nonlinear and too complex due to coupling between different motion states.

3.1.2 Generalized Nonlinear Dynamic Model in 3DOFs

Motion for a surface craft, such as the AutoNaut, primarily occurs in the horizontal plane that can be described using the 3DOFs - surge, sway and yaw. The maneuvering model described in Eq. 3.8 involves additional DOFs that are heave, pitch, and roll, which introduce more complexities in terms of modeling, simulation, and control. By focussing on the reduced order model, we capture the essential dynamics of the craft and disregard the relatively negligible values of pitch and roll. This simplification also help reduce computational complexity.

Assuming the generalized coordinates as given in Eq. 3.1, Eq. 3.2 and Eq. 3.3, the generalized nonlinear model for a 3DOFs as described in sec. 6.5 in [24] can now be expressed as follows:

$$\begin{aligned} \dot{\eta} &= R(\psi)(v_r + v_c) \\ \mathbf{M}\dot{v}_r + \mathbf{C}(v_r)v_r + \mathbf{D}(v_r)v_r &= \tau_{\text{prop}} + \tau_{\text{dist}} + \tau_{\text{ud}} \end{aligned} \quad (3.9)$$

where

- τ_{prop} and τ_{dist} are forces due to propulsion and disturbance respectively.
- τ_{ud} are the disturbances due to higher-order wave components and other unmodeled dynamics.

By exploiting the operation conditions and design of the AutoNaut, we can further simplify the model matrices as described below.

The system inertia matrix, \mathbf{M} and the Coriolis-centripetal matrix, $\mathbf{C}(v_r)$ are composed of two terms each which represent the contribution of rigid-body and added-mass kinetics. For rigid body kinetics, we can assume the AutoNaut to have a homogeneous mass distribution and symmetry along the xz-plane. Further, the body-frame origin coordinate aligns with the centerline of the craft yielding $y_g = 0$. Using Eq. 6.7 in [24] gives us,

$$\mathbf{M}_{\text{RB}} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_g \\ 0 & mx_g & Iz \end{bmatrix} \quad \mathbf{C}_{\text{RB}}(v_r) = \begin{bmatrix} 0 & -mr & -mx_g r \\ mr & 0 & 0 \\ mx_g r & 0 & 0 \end{bmatrix} \quad (3.10)$$

where m is the mass of the vehicle, I_z is the moment of inertia along the z-axis and (x_g, y_g) is the Center of Gravity (CG) in BODY frame.

The added-mass effects can be estimated by decoupling the surge mode from the steering dynamics. Further, we neglect the effects of heave, pitch and roll due to the assumption of small motion which using Eq. 6.56 and Eq. 6.57 from [24] gives us,

$$\mathbf{M}_A = - \begin{bmatrix} X_{\dot{u}} & 0 & 0 \\ 0 & Y_{\dot{v}} & Y_{\dot{r}} \\ 0 & N_{\dot{v}} & N_{\dot{r}} \end{bmatrix} \quad \mathbf{C}_A(\mathbf{v}_r) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}}v_r + Y_{\dot{r}}r \\ 0 & 0 & -X_{\dot{u}}u_r \\ -Y_{\dot{v}}v_r - Y_{\dot{r}}r & X_{\dot{u}}u_r & 0 \end{bmatrix} \quad (3.11)$$

The vehicle experiences damping forces due to multiple sources such as potential damping, skin friction, wave drift damping, etc. These are summarised into two components – linear component represented by \mathbf{D} and nonlinear component given by $\mathbf{D}_n(\mathbf{v}_r)$. This relationship is given by

$$\mathbf{D}(\mathbf{v}_r) = \mathbf{D} + \mathbf{D}_n(\mathbf{v}_r) \quad (3.12)$$

Using our previous assumptions, nonlinear damping components can be neglected and the linear damping matrix for the AutoNaut can be expressed using Eq. 6.67 in [24] as

$$\mathbf{D} = - \begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix} \quad (3.13)$$

Force and Moment Analysis

The forces and moments acting on the vessel can be classified into three distinct groups: propulsion forces denoted by τ_{prop} , disturbance forces denoted by τ_{dist} , and unmodelled dynamics denoted by τ_{ud} .

The **propulsion forces** acting on the vehicle are a result of the wave-foil technology, onboard actuators, and the rudder installed on the stern strut. This relationship can be expressed as follows:

$$\tau_{\text{prop}} = \tau_{\text{foil}} + \tau_{\text{thrs}} + \tau_{\text{rudr}} \quad (3.14)$$

where

- τ_{foil} is the forward propulsion generated by wave-induced vessel motion.
- τ_{thrs} is the active propulsion generated by the onboard thruster meant for use in calm waters or in emergency scenarios.
- τ_{rudr} is the passive steering force generated by the rudder.

In Chapter 2.4, we discussed the influence of environmental factors, including winds, waves, and currents, on the speed of the vehicle. Modeling the precise relationship between these environmental factors and vehicle dynamics is however a challenging task due to its complexity. In this study, we use the speed prediction

model proposed in [25] to emulate surge dynamics. By restricting the impact of wave foils in the surge dimension, we can express the force generated by wave-foil technology using Eq. 2.1 as

$$\tau_{\text{foil}} = - \begin{bmatrix} g(x) \\ 0 \\ 0 \end{bmatrix} X_u \quad (3.15)$$

where X_u is the linear damping component in surge.

AutoNaut has an onboard thruster for use in very calm water and emergency scenarios. A core objective of this work is to optimize mission planning while reducing energy expenditure due to unnecessary rudder motion. Since thrusters are heavy energy sinks, in most cases we either want to either limit or completely prevent thruster use. Hence, the force exerted by the thruster is assumed to be zero.

$$\tau_{\text{thrs}} = 0 \quad (3.16)$$

The steering forces generated by the rudder can be described using Eq. 9.103 in [24]. Assuming a small rudder angle δ , we have

$$\begin{aligned} \tau_{\text{ruddr}} &= \begin{bmatrix} X_{\delta}^{\text{NL}} \sin(\delta) \\ Y_{\delta}^{\text{NL}} \cos(\delta) \\ N_{\delta}^{\text{NL}} \cos(\delta) \end{bmatrix} \\ \theta_{\delta} &= \rho U_R^2 A_R C_N \sin(\alpha_R) \\ X_{\delta}^{\text{nl}} &= -\frac{1}{2}(1 - t_R)\theta_{\delta} \\ Y_{\delta}^{\text{nl}} &= -\frac{1}{2}(1 + a_H)\theta_{\delta} \\ N_{\delta}^{\text{nl}} &= -\frac{1}{2}(x_R + a_H x_H)\theta_{\delta} \end{aligned} \quad (3.17)$$

where t_R is the thrust reduction factor, A_R is the area of the rudder, b is rudder height, U_R is the rudder inflow speed, a_H is the rudder force increase factor, x_H is the longitudinal coordinate of the additional lateral force, x_R is the longitudinal coordinate of the rudder position and C_N is the rudder coefficient, and α_R is the effective rudder angle.

The effective rudder angle α_R can be calculated as

$$\alpha_R = \delta - \beta_R \quad (3.18)$$

In practical scenarios, the drift angle at the rudder position β_R and the rudder inflow speed U_R are different from the sideslip angle β and relative speed of the vehicle U_r . However, for the sake of modeling convenience, we assume that they are equivalent, giving us

$$\beta_R \approx \beta \quad U_R \approx U_r \quad (3.19)$$

The **disturbance forces** acting on the vehicle are induced due to wind, waves, and currents. Currents are accounted in Eq. 3.9 explicitly and winds are introduced as a constant or slowly-varying wind force τ_{wind} which using Eq. 10.23 in [24] give us

$$\begin{aligned}\tau_{\text{dist}} = \tau_{\text{wind}} &= \begin{bmatrix} X_w \\ Y_w \\ N_w \end{bmatrix} \\ X_w &= -\frac{1}{2}c_x(\gamma_{rw})A_{F_w} \cos(\gamma_w)\rho_a V_{rw}^2 \\ Y_w &= \frac{1}{2}c_y(\gamma_{rw})A_{L_w} \sin(\gamma_w)\rho_a V_{rw}^2 \\ N_w &= \frac{1}{2}c_n(\gamma_{rw})A_{L_w} L_{oa} \sin(2\gamma_w)\rho_a V_{rw}^2\end{aligned}\quad (3.20)$$

where V_{rw} is the relative wind speed, γ_{rw} is the angle of attack, A_{F_w} and A_{L_w} are the forward and lateral projected areas, L_{oa} is the length overall and c_x, c_y, c_n are approximate wind coefficients assuming symmetry with respect to the xz - and yz -planes.

All unmodeled effects such as that of rotational currents, motion coupling, frequency-dependent wave-action, first-order wave-induced forces, and residual energy imparted by the wave-foils are summed in the term τ_{ud} . This component is treated as noise and is assumed to be small. A robust controller that explicitly deals with uncertainty can compensate for the effects of τ_{ud} .

3.1.3 Generalized nonlinear dynamic model in 2DOF

The model described by Eq. 3.9 considers the equations of motion acting in the horizontal plane. This motion can be further divided into two groups called the surge- and sway-yaw subsystems. The surge subsystem for AutoNaut can be written as

$$(m - X_{\dot{u}})\dot{u}_r - X_u u_r + \mathbf{N}(v_r) = \tau_{\text{prop}_x} + \tau_{\text{dist}_x} + \tau_{\text{ud}_x} \quad (3.21)$$

where the subscript x represents the x -component of the respective force vector described in Eq. 3.14 and $\mathbf{N}(v_r)$ represents the contribution of nonlinear and cross-coupled terms.

If we assume straight-line motion and neglect the contribution of wind forces, we observe that the surge speed depends explicitly on the environmental factors rendering the surge dynamics uncontrollable. Since these factors are either slowly-changing or bounded, we can in effect presume the speed to be an uncontrollable quasi-constant.

By employing the speed prediction model [25], we are able to parameterize the surge motion and anticipate the vehicle's steady state speed. This approach enables us to project forward in time by leveraging observed trends and making predictions about the future.

Ignoring surge and considering only the sway-yaw subsystem, we modify our generalized state vector η to $\eta^* = [y \ \psi]^T$ and rewrite the equations of motion as

$$\begin{aligned} \dot{\eta}^* &= \mathcal{R}(\psi)(\mathbf{v}_r^* + \mathbf{v}_c^*) \\ \mathbf{M}_2 \dot{\mathbf{v}}_r^* + \mathbf{C}_2(\mathbf{v}_r)\mathbf{v}_r^* + \mathbf{D}_2 \mathbf{v}_r^* + \mathbf{D}_{2n}(\mathbf{v}_r)\mathbf{v}_r^* &= \tau_{\text{prop}} + \tau_{\text{dist}} + \tau_{\text{ud}} \end{aligned} \quad (3.22)$$

where

- $\mathbf{v}_r^* = [v_r \ r]^T$ is the 2DOFs relative state velocity vector.
- $\mathbf{v}_c^* = [v_c \ 0]^T$ is the 2DOFs current velocity vector
- the 2 subscript refers to sway and yaw components of the hydrodynamic matrices.

We note that generalized force vectors τ_{prop} and τ_{dist} are only considered in y - and ψ -axis.

3.1.4 Generalized linear dynamic model in 3DOF

In the field of control theory, linear systems are often preferred due to their simpler properties compared to nonlinear systems. Linear time-invariant (LTI) systems, in particular, have been extensively studied, offering a wealth of knowledge on their behavior and making them valuable as a fundamental component for analysis [26]. Although linear systems may not perfectly capture the complexities of real-world systems, their simplicity and computational efficiency often make them a practical choice for implementation.

For the AutoNaut, the simplified 3DOFs linear model is derived by neglecting nonlinear and cross-coupled terms involving nonlinear damping and Coriolis effects. Additionally, the influence of wind forces is neglected, and the rudder force is linearized under the assumption of small rudder angles. These approximations hold valid at low speeds and are expressed as

$$\begin{aligned} \mathbf{C}(\mathbf{v}_r) &= \mathbf{D}_n(\mathbf{v}_r) = \mathbf{0} \\ \tau_{\text{wind}} &= 0 \\ \sin(\delta) &\approx \delta \end{aligned} \quad (3.23)$$

The linearized rudder force can now be expressed using Eq. 9.103 in [24] as

$$\begin{aligned} \tau_{\text{rudr}}^L &= - \begin{bmatrix} X_\delta^L \delta^2 \\ Y_\delta^L \delta \\ N_\delta^L \delta \end{bmatrix} \\ X_\delta^1 &= \frac{1}{2}(1 + t_R)\rho U_r^2 A_R C_N \\ Y_\delta^1 &= \frac{1}{4}(1 + a_H)\rho U_r^2 A_R C_N \\ N_\delta^1 &= \frac{1}{4}(x_R + a_H x_H)\rho U_r^2 A_R C_N \end{aligned} \quad (3.24)$$

where the subscript L denotes the linearized force. The linearized dynamics of AutoNaut can now be expressed as

$$\begin{aligned} \eta &= R(\psi)(v_r + v_c) \\ \mathbf{M}\dot{v}_r + \mathbf{D}v_r &= \tau_{\text{foil}} + \tau_{\text{rudr}}^L + \tau_{\text{ud}} \end{aligned} \quad (3.25)$$

where the superscript L denotes the linearised force component.

3.2 State Space Modeling

A state-space representation is a mathematical model that describes the behavior of a system in terms of its internal state variables and the inputs that affect them. This representation is useful for analyzing and designing control systems because it allows for a compact and computationally efficient representation of the system.

Consider a system with the state variable \mathbf{x} , input variable \mathbf{u} , and output variable \mathbf{y} . Its state-space representation in a linear framework takes the general form

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{aligned} \quad (3.26)$$

where \mathbf{A} and \mathbf{B} are the state matrix and input matrix respectively. \mathbf{C} and \mathbf{D} are the output and feed-through matrix respectively.

In the case of nonlinear systems, the states and transitions are defined by nonlinear mappings and take the form

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{u}) \\ \mathbf{y} &= h(\mathbf{x}, \mathbf{u}) \end{aligned} \quad (3.27)$$

where f and h are vectors consisting of nonlinear mappings $f_i(\cdot)$ and $h_i(\cdot)$ respectively.

In this section, we present both linear and nonlinear dynamics of the vehicle in nonlinear state-space form for the sake of convenience. For the models expressed in 3DOFs, the state \mathbf{x} and the input variable \mathbf{u} are chosen as

$$\mathbf{x} = \begin{bmatrix} \psi \\ u \\ v \\ r \end{bmatrix} \quad \mathbf{u} = [\delta] \quad (3.28)$$

For the models expressed in 2DOFs, the state variable \mathbf{x} is modified to \mathbf{x}_2 as

$$\mathbf{x}_2 = \begin{bmatrix} \psi \\ v \\ r \end{bmatrix} \quad (3.29)$$

We limit the scope of our project towards the development of the control system and assume that the full state is either directly observable or available using an appropriate estimator. The output mapping is then given as

$$h(\mathbf{x}, \mathbf{u}) = \mathbf{x} \quad (3.30)$$

3.2.1 Linearized Dynamics in 3DOFs

The linear model given in Eq. 3.25 can be rewritten as

$$\begin{aligned}\dot{\psi} &= r \\ \dot{v} &= \dot{v}_c + M^{-1}(\tau_{\text{foil}} + \tau_{\text{rudr}}^L - D v_r)\end{aligned}\quad (3.31)$$

where $\tau_{\text{ud}} = 0$.

We can now represent Eq. 3.31 in state-space form as described in Eq. 3.27 by expressing the mapping f as

$$\begin{aligned}\dot{\mathbf{x}} &= f^l(\mathbf{x}, \mathbf{u}) \\ &= \begin{bmatrix} 0 \\ v_c \\ -u_c \\ 0 \end{bmatrix} r + \begin{bmatrix} 1 & 0 \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} r \\ X_u u_r - X_\delta \delta^2 - g(x) X_u \\ Y_v v_r + Y_r r - Y_\delta \delta \\ N_v v_r + N_r r - N_\delta \delta \end{bmatrix} \\ &= \begin{bmatrix} f_1^l(\mathbf{x}, \mathbf{u}) \\ f_2^l(\mathbf{x}, \mathbf{u}) \\ f_3^l(\mathbf{x}, \mathbf{u}) \\ f_4^l(\mathbf{x}, \mathbf{u}) \end{bmatrix}\end{aligned}\quad (3.32)$$

where $\Delta = \mathbf{M}^{-1}$ and Δ_{ij} is the element located at i^{th} row and j^{th} column. The full expression is mentioned below for better readability.

$$\begin{aligned}\dot{\psi} &= f_1^l(\mathbf{x}, \mathbf{u}) = r \\ \dot{u} &= f_2^l(\mathbf{x}, \mathbf{u}) = v_c r + \Delta_{11}(X_u(u_r - g(x)) - X_\delta \delta^2) \\ \dot{v} &= f_3^l(\mathbf{x}, \mathbf{u}) = -u_c r + \Delta_{22}(Y_v v_r + Y_r r - Y_\delta \delta) + \Delta_{23}(N_v v_r + N_r r - N_\delta \delta) \\ \dot{r} &= f_4^l(\mathbf{x}, \mathbf{u}) = \Delta_{23}(Y_v v_r + Y_r r - Y_\delta \delta) + \Delta_{33}(N_v v_r + N_r r - N_\delta \delta)\end{aligned}\quad (3.33)$$

3.2.2 Nonlinear Dynamics in 3DOFs

We can rewrite the nonlinear model given in Eq. 3.9 as

$$\begin{aligned}\dot{\psi} &= r \\ \dot{v} &= \dot{v}_c + \mathbf{M}^{-1}(\tau_{\text{foil}} + \tau_{\text{rudr}} + \tau_{\text{wind}} - \mathbf{N}(v_r) v_r)\end{aligned}\quad (3.34)$$

where $\mathbf{N}(v_r) = \mathbf{D} + \mathbf{C}(v_r)$ is the matrix of net resistive forces.

$$\begin{aligned}
\dot{\mathbf{x}} &= f^{\text{nl3}}(\mathbf{x}, \mathbf{u}) \\
&= \begin{bmatrix} 0 \\ v_c \\ -u_c \\ 0 \end{bmatrix} r + \begin{bmatrix} 1 & 0 \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} r \\ -g(x)X_u - X_\delta^{\text{nl}} \sin(\delta) + X_w - N_{1,1:3} v_r \\ -Y_\delta^{\text{nl}} \cos(\delta) + Y_w - N_{2,1:3} v_r \\ -N_\delta^{\text{nl}} \cos(\delta) + N_w - N_{3,1:3} v_r \end{bmatrix} \\
&= \begin{bmatrix} f_1^{\text{nl3}}(\mathbf{x}, \mathbf{u}) \\ f_2^{\text{nl3}}(\mathbf{x}, \mathbf{u}) \\ f_3^{\text{nl3}}(\mathbf{x}, \mathbf{u}) \\ f_4^{\text{nl3}}(\mathbf{x}, \mathbf{u}) \end{bmatrix}
\end{aligned} \tag{3.35}$$

where $N_{i,1:3}$ is the i^{th} row of the matrix $\mathbf{N}(v_r)$. The full expression is obtained in practice using MATLAB™ symbolic framework and is expressed below.

$$\begin{aligned}
\dot{\psi} &= f_1^{\text{nl3}}(\mathbf{x}, \mathbf{u}) = r \\
\dot{u} &= f_2^{\text{nl3}}(\mathbf{x}, \mathbf{u}) = u_c - 1.0u + r v_c + g(\theta) - 3.5e \\
&\quad - 3 \delta^2 (53.0(u - 1.0u_c)^2 + 53.0(v - 1.0v_c)^2) \\
\dot{v} &= f_3^{\text{nl3}}(\mathbf{x}, \mathbf{u}) = 0.2r - 0.51v + 0.51v_c - 1.3e \\
&\quad - 3 \delta (100.0(u - 1.0u_c)^2 + 100.0(v - 1.0v_c)^2) - 8.9e \\
&\quad - 5 \delta (210.0(u - 1.0u_c)^2 + 210.0(v - 1.0v_c)^2) - 1.0r u_c \\
\dot{r} &= f_4^{\text{nl3}}(\mathbf{x}, \mathbf{u}) = 0.035v - 1.0r - 0.035v_c + 8.9e \\
&\quad - 5 \delta (100.0(u - 1.0u_c)^2 + 100.0(v - 1.0v_c)^2) + 4.6e \\
&\quad - 4 \delta (210.0(u - 1.0u_c)^2 + 210.0(v - 1.0v_c)^2)
\end{aligned} \tag{3.36}$$

The hydrodynamic parameters used to obtain the above equations are given in Appendix A.

3.2.3 Nonlinear Dynamics in 2DOFs

The 2DOFs nonlinear dynamic model in state-space form can be expressed by eliminating surge dynamics and introducing surge as a parameter in the vehicle model. This is expressed as

$$\begin{aligned}
\dot{\mathbf{x}}_2 &= f^{\text{nl2}}(\mathbf{x}_2, \mathbf{u}, \mathbf{p}) \\
&= \begin{bmatrix} f_1^{\text{nl2}}(\mathbf{x}_2, \mathbf{u}, \mathbf{p}) \\ f_2^{\text{nl2}}(\mathbf{x}_2, \mathbf{u}, \mathbf{p}) \\ f_3^{\text{nl2}}(\mathbf{x}_2, \mathbf{u}, \mathbf{p}) \end{bmatrix}
\end{aligned} \tag{3.37}$$

where $\mathbf{p} = u = g(\theta)$ is the called the surge parameter. The equations of motion for sway and yaw dynamics remain the same as expressed in Eq. 3.36.

3.3 Control Objective

In order to define the desired behavior of the system, we now formulate our control objective. This will allow us to specify the goal control system should satisfy to achieve its intended function. We present two cost functions

Minimizing the difference between current and desired course angle

Let us consider the difference function as

$$\delta_x(t) = \text{ssa}(\chi(t) - \chi_d(t)) \quad (3.38)$$

where ssa is the smallest signed angle as described in [24].

The function $\delta_x(t)$ is bounded between $[-\pi \ \pi]$ and has its minimum value when the current course angle is the same as the desired course angle. This is a straightforward way to align the current course with the desired course however, it can run into numerical difficulties due to the presence of the arctan function, which makes the function undefined as surge approaches zero.

Aligning the current and desired course angle vector

This method utilizes unit vectors to represent the current course angle and the desired course angle. When these vectors align, their dot product reaches its maximum value of unity. To convert the maximizing function into a minimizing function, we subtract the dot product from its maximum value. This enhances numerical stability as it avoids any inverse trigonometric functions. The proof of this function is presented below.

From Eq. 2.36 in [24], the horizontal motion of the marine craft can be described by

$$\begin{aligned} \dot{x}^n &= u \cos \psi - v \sin \psi = U \cos(\chi) \\ \dot{y}^n &= u \sin \psi + v \cos \psi = U \sin(\chi) \end{aligned} \quad (3.39)$$

where we have

$$\begin{aligned} \chi &= \psi + \beta_c \\ \beta_c &= \arctan\left(\frac{v}{u}\right) \\ U &= \sqrt{(u^2 + v^2)} \end{aligned} \quad (3.40)$$

Now the course vector can be expressed by normalizing the velocity vector v^n in the inertial frame as

$$\begin{aligned} \vec{\chi} &= \begin{bmatrix} \cos(\chi) \\ \sin(\chi) \end{bmatrix} \\ &= \frac{1}{U} v^n = \frac{1}{U} \begin{bmatrix} \dot{x}^n \\ \dot{y}^n \end{bmatrix} \end{aligned} \quad (3.41)$$

Given a desired course angle, we can represent the desired course vector as

$$\vec{\chi}_d = \begin{bmatrix} \cos \chi_d \\ \sin \chi_d \end{bmatrix} \quad (3.42)$$

We know by the nature of dot product that the dot product of two parallel vectors is zero. Since both are unit vectors, we have

$$\vec{\chi}_d \cdot \vec{\chi} = |\chi| \cdot |\chi_d| = |\chi| * |\chi_d| \cos \theta = 1 \quad (3.43)$$

when the current course vector aligns with the desired course angle. Hence, minimizing the expression below will align the vehicle in the direction of the desired course vector.

$$\delta_x = 1 - \vec{\chi}'_d \cdot \vec{\chi} \quad (3.44)$$

3.4 Nonlinear Model Predictive Control

With an understanding of our system dynamics and the desired control objective, we can now delve into the theory that underlies our proposed solution. This section provides a concise overview of the fundamental concepts of Model Predictive Control (MPC).

3.4.1 Definition

Model Predictive Control is a subset of predictive and optimal control theory that uses a mathematical model of the system to make predictions and optimize control actions over a given time horizon. MPC has been widely used in various industrial applications[27] from process applications to power electronics[28]. Due to advancements in digital electronics, MPC is now finding use in cutting-edge robotics such as humanoid robots[29] and unmanned drones[30].

MPC when extended to nonlinear systems is called NMPC where N stands for nonlinear. In this case, both the system model and the constraints may be specified using nonlinear equations. However, unlike linear MPC, no guarantees are made that the receding horizon open-loop optimal control solution will yield a stable system. In the texts that follow, MPC and NMPC have been used interchangeably.

Working of a MPC Controller

Let us consider a controlled process with a state is given by \mathbf{x}_n and measured at discrete time intervals t_n with $n = 0, 1, 2, 3, \dots$. Since the system is controlled, at each time interval, we can select a control input \mathbf{u}_n that influences the future behavior of the system. For such a system, MPC can be employed to address two types of problems outlined below.

Stabilization Problem: In this problem, we are given a reference state or trajectory (in time) $\mathbf{x}^{\text{ref}}(t)$ and the goal is to determine the control inputs that drive the current state of the system towards the reference state.

$$\mathbf{x}(t) \rightarrow \mathbf{x}^{\text{ref}}(t) \quad (3.45)$$

Regularization Problem: In this problem, the reference state is constant and zero. The goal is to keep the current state at origin.

$$\mathbf{x}(t) \rightarrow \mathbf{0} \quad (3.46)$$

MPC is an optimization-based method for feedback control of systems, which means that the desired control input $\mathbf{u}(t) \in \mathcal{U}$ is expressed a mapping of the current state $\mathbf{x}(t) \in \mathcal{X}$ as in the form

$$\begin{aligned} \mu : \mathcal{X} &\rightarrow \mathcal{U} \\ u(t) &= \mu(\mathbf{x}(t)) \end{aligned} \quad (3.47)$$

The primary objective of the MPC controller is to find the mapping μ which is accomplished by using two key ideas: *iterative online minimization* and *moving horizon*.

Iterative online minimization is a three-step process which computes the optimal input sequence for a given system state. We start by defining the system model to predict the behavior of the system at discrete time intervals $\mathbf{x}_n(k)$ with $k = 0, 1, 2, \dots, N$ for up to N intervals, starting at the given time t_n .

$$\begin{aligned} x_n(0) &= x(t_n) \\ x_n(k) &= x(t_n + kT_s) \end{aligned} \quad (3.48)$$

where T_s is the controller discretization time. For our proposed system(s), the dynamic model can be represented in state-space form using Eq. 3.27 as

Continuous Dynamics	Discrete Dynamics	
$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$	$\mathbf{x}_{k+1} = f_d(\mathbf{x}_k, \mathbf{u}_k)$	(3.49)
$\mathbf{y} = h(\mathbf{x}, \mathbf{u})$	$\mathbf{y}_k = h_d(\mathbf{x}_k, \mathbf{u}_k)$	

where mappings f_d and h_d are obtained by discretizing the continuous time non-linear dynamics.

Now we propose an arbitrary input control sequence $\mathbf{u}_n(k)$ for $k \in [0, N)$ and compute the predicted state trajectory $\mathbf{x}_n(k)$ for $k \in [0, N]$. For the chosen $\mathbf{u}_n(k)$, we can measure the deviation of the predicted state trajectory $\mathbf{x}_n(k)$ from the desired state trajectory $\mathbf{x}_n^{\text{ref}}(k)$ using a function known as the objective function. Over the entire predicted trajectory, the deviation is expressed as

$$J = \sum_{k=0}^{N-1} l(\mathbf{x}_n(k), \mathbf{x}_n^{\text{ref}}(k), \mathbf{u}_n(k), \mathbf{u}_n^{\text{ref}}(k)) + m(\mathbf{x}_n(N), \mathbf{x}_n^{\text{ref}}(N)) \quad (3.50)$$

where $\mathbf{u}_n^{\text{ref}}(k)$ is the reference input sequence often set as zero to minimize actuator use, and the mappings $l(\cdot)$ and $m(\cdot)$ are known as the stage and terminal cost respectively. A common choice for them is the quadratic least squares function

The final step in the optimization process is to use the principles of optimal control to determine the control sequence \mathbf{u}_n^* that minimizes the value of the cost function over the entire trajectory i.e. J . The sequence thus obtained is called the optimal control sequence and the state trajectory obtained using u^* is called the optimal state trajectory. The optimal control problem is now formulated as

$$\mathbf{u}^* = \min_{\mathbf{u}_n(k)} J \quad (3.51)$$

To get the desired feedback, we set the mapping μ to extract the first element of the sequence \mathbf{u}^* yielding

$$\mu(x_n) = \mathbf{u}^*(0) \quad (3.52)$$

This process is repeated iteratively for each time instant $t_n = t_1, t_2, t_3 \dots$ to yield $\mu(\mathbf{x}_1), \mu(\mathbf{x}_2), \mu(\mathbf{x}_3) \dots$ that forms the feedback controller and is called *iterative on-line optimization*.

At each time instant t_n , we compute predictions for a fixed number of intervals N , hence defining a window that starts at t_n and ends at $t_n + NT_s$. As we move forward in time, the window or the prediction horizon also moves accordingly. This feature is called *moving horizon* and is an important aspect of MPC.

3.4.2 Dynamic Model and Discretization

Most physical systems are mathematically expressed as non-linear continuous time Ordinary Differential Equations (ODEs). Since MPC uses the system dynamics to predict its future behavior, these ODEs need to be transformed into discrete-time difference equations through the process known as discretization, as referred in Eq. 3.49.

Discretization is an essential step in MPC problem formulation as it allows us to express the control problem in finite-dimension space, enabling the use of numerical optimization techniques. In this process, the continuous-time equations and associated constraints/functions are converted to a finite set of equations that can be solved efficiently using computational methods and implemented on digital systems. However, care should be taken as poor discretization techniques can lead to inaccurate system predictions leading to control problems. This is especially important for highly non-linear or unstable systems.

The most common methods for discretization using direct numerical approaches are (1) single shooting, (2) multiple shooting and (3) collocation as described in [31].

1. *Single-shooting* involves discretizing the control variables and subsequently employing a numerical routine to obtain the states sequentially as an initial value problem over the entire horizon. Although single shooting is straightforward, it can encounter difficulties when applied to highly nonlinear or unstable systems. Hence this approach is often limited to simple problems.

2. *Multiple-shooting* is an improvement over the single-shooting method which introduces additional degrees of freedom by dividing the prediction horizon into extra grid points with each grid point generating an initial value problem. This improves convergence and mitigates the error growth caused by inadequate initial data.
3. *Collocation* is another effective method that involves incorporating a set of collocation points using polynomial equations to approximate the model equations between those points. This technique ensures the satisfaction of the model equation at these intermediate points. By introducing these additional variables, direct collocation offers even more degrees of freedom compared to multiple shooting while reducing nonlinearity.

Both multiple shooting and collocation methods offer significant improvements over the single shooting method however multiple shooting is often preferred for problems with simple control but complicated dynamics and simple or no path constraints [32]. Due to this reason, we chose to use multiple shootings to discretize the MPC problem.

Multiple Shooting

The direct multiple shooting method, first introduced by [33] is a simultaneous approach as it reformulates the ODE to a set of nonlinear algebraic equality constraints that are solved simultaneously with the optimization problem. We proceed as follows.

Consider a uniform time grid parameterized by k for up to N intervals known as shooting nodes. The control sequence is then obtained by a simple piece-wise discretization of the control signal as follows

$$\begin{aligned} k &= 0, 1, 2 \dots N - 1 \\ \mathbf{u}_n(t) &= \mathbf{u}_k \quad \forall t \in [t_n, t_{n+1}] \end{aligned} \quad (3.53)$$

Each state \mathbf{x}_k in the grid is used as an initial value to solve the ODE for the time interval $[t_k, t_{k+1}]$ using a suitable integrator as follows

$$\mathbf{x}_{k+1} = \Phi(\mathbf{x}_k, \mathbf{u}_k) \quad (3.54)$$

where Φ is typically a simple and robust Runge-Kutta 4 (RK4) approximation.

By employing this approach, we incorporate the discretized system state variable, denoted as \mathbf{x}_k , as an unknown quantity within a nonlinear equality constraint. In addition to minimizing the objective function, we simultaneously solve for the system dynamics.

While multiple shooting introduces significantly more variables in the optimization problem, it is more well-posed for several reasons as it takes advantage of the favorable structural and numerical properties of the equations. Further, it facilitates easier evaluation of both the cost and constraint functions while benefiting from the availability of reliable initial guesses.

For the case of AutoNaut, the discretized dynamics can be formulated as

$$\begin{aligned} \text{given } \mathbf{x}_0 &= \mathbf{x}(0) \\ \mathbf{x}_{k+1} &= \Phi(f(\mathbf{x}_k, \mathbf{u}_k)) \end{aligned} \quad (3.55)$$

where $k \in [0 N)$, Φ is an appropriate RK4 integrator and f is the continuous-time dynamic models given by Eq. 3.9, Eq. 3.22 or Eq. 3.25.

A Note on Direct and Indirect Methods

A typical time-invariant continuous-time dynamic model is expressed using the ODE given by the function f which defines the evolution of the state $\mathbf{x}(t)$. The system also depends on input signal $\mathbf{u}(t)$ possessing in an infinite-dimensional entity as shown below

$$\begin{aligned} \mathbf{x}(t) &\in \mathcal{R}^n \quad \mathbf{u}(t) \in \mathcal{R}^m \\ \mathbf{x}(0) &\in \mathcal{R}^n \\ \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \mathbf{u}(t)) \end{aligned} \quad (3.56)$$

where n and m are state and input dimensions respectively.

Under the assumption that the involved functions satisfy the necessary regularity conditions, including continuity and smoothness, we can employ classical mathematical tools such as calculus of variations[34], Pontryagin's maximum principle[35], and dynamic programming[36]. These methods, known as indirect methods, are applicable only in specific cases as they may become impractical for larger systems. An illustrative example of this approach is the linear quadratic regulator, which is widely recognized.

In this work, we utilize the more popular and promising *direct methods* that are characterized by discretization and finite parameterization. It is advantageous to use direct methods due to the following reasons:

1. Generally, numerical integration is required to handle the system as exact closed-form solutions of the ordinary differential equations (ODEs) are typically not feasible, especially in nonlinear cases.
2. The infinite-dimensional unknown solution $\mathbf{u} \in [0 T]$ must be replaced by a finite number of decision variables to define a finite-dimensional optimization problem that can be effectively solved using numerical optimization techniques.
3. Measurements are often available only at specific sampling instants, leading to the availability of updated initial state $\mathbf{x}(0)$ only at those defined sampling points.
4. Due to the nature of physical systems, arbitrary changes in control commands are not possible with control adjustments only being realized at defined sampling instants.

3.4.3 Objective Function

For a nonlinear Optimal Control Problem (OCP) starting from a continuous time model and a finite horizon of length T , the objective or cost function can be expressed as

$$J(x(t), u(t)) = \int_0^T l(\mathbf{x}(t), \mathbf{u}(t), \mathbf{x}^{\text{ref}}(t), \mathbf{u}^{\text{ref}}(t)) dt + m(\mathbf{x}(T), \mathbf{x}^{\text{ref}}(T)) \quad (3.57)$$

where $t \in [0 \ T]$, $\mathbf{x}^{\text{ref}}(t)$ and $\mathbf{u}^{\text{ref}}(t)$ are the reference state and reference input trajectories respectively.

In the above formulation, the term l is known as the stage cost and m as the terminal cost which together define the objective function J . The objective function allows definition for time-varying reference trajectories and exogenous input signals as long as it satisfies the necessary regularity assumptions, such as continuity and smoothness but we limit our scope to time-invariant cost functions.

After discretization, as described in sec. 3.4.2, the objective function is expressed using a summation operator instead of the integral as

$$J = \sum_{k=0}^{N-1} l(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_k^{\text{ref}}, \mathbf{u}_k^{\text{ref}}) + m(\mathbf{x}_N, \mathbf{x}_N^{\text{ref}}) \quad (3.58)$$

where $T = NT_s$ with T is the prediction horizon, T_s is the discretization time and N is the number of intervals in the prediction horizon.

The exact formulation of both l and m can be rich and varied in literature and have a significant impact on both MPC performance and stability[37]. From a practical perspective, however, one of two typical choices are seen

$$\begin{aligned} l_2 \text{ norm: } l(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_k^{\text{ref}}, \mathbf{u}_k^{\text{ref}}) &= \|\mathbf{x}_k - \mathbf{x}_k^{\text{ref}}\|_Q^2 + \|\mathbf{u}_k - \mathbf{u}_k^{\text{ref}}\|_R^2 \\ l_1 \text{ norm: } l(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_k^{\text{ref}}, \mathbf{u}_k^{\text{ref}}) &= Q\|\mathbf{x}_k - \mathbf{x}_k^{\text{ref}}\|^1 + R\|\mathbf{u}_k - \mathbf{u}_k^{\text{ref}}\|^1 \end{aligned} \quad (3.59)$$

where Q and R are positive semi-definite matrices. The properties of the Q and R important for performance and stability are discussed further in [14].

For the case of AutoNaut, we consider two different cost formulations based on the vehicle state and the change in the input rudder angle. We ignore the absolute input trajectory as we are only concerned with rapid rudder changes. The objective function is then formulated as follows

$$\begin{aligned} l(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_k^{\text{ref}}, \mathbf{u}_k^{\text{ref}}) &= l^x(\mathbf{x}_k, \mathbf{x}_k^{\text{ref}}) + l^u(\mathbf{u}_k) \\ l^u(\mathbf{u}_k) &= \|\mathbf{u}_k - \mathbf{u}_{k-1}\|_R^2 \end{aligned} \quad (3.60)$$

where \mathbf{u}_{k-1} is the input signal at the previous interval with $\mathbf{u}_{-1} = 0$.

As described in sec. 3.3, we propose the use of two different objective functions. Considering the state vector \mathbf{x} as described in Eq. 3.28 and using Eq. 3.38

and Eq. 3.44, we can formulate them as

minimizing course difference

$$l^x(\mathbf{x}_k, \mathbf{x}_k^{\text{ref}}) = \|\text{ssa}(\chi_k^{\text{ref}} - \psi - \beta)\|_{Q_1}^2 \quad (3.61)$$

aligning course vector

$$l^x(\mathbf{x}_k, \mathbf{x}_k^{\text{ref}}) = \|1 - \vec{\chi}_k^{\text{ref}} \cdot \vec{\chi}_k\|_{Q_2}^2$$

where

- χ_k^{ref} is the trajectory obtained by using a first-order reference model with $\chi_0^{\text{ref}} = \chi_0$ and $\chi_N^{\text{ref}} = \chi_{ref}$
- χ_0 and χ_{ref} are the current and the desired course angle respectively
- Q_1 and Q_2 are the costs for the objective functions as expressed in Eq. 3.38 and Eq. 3.44 respectively.
- $\text{ssa}(\cdot)$ is the function to evaluate the smallest sign angle difference of the angles being subtracted.

3.4.4 Constraints

Constraints form an integral part of MPC Problem formulation and are arguably one of the key reasons behind the success of MPC controllers. In this text, we consider constraints on both state and input.

Considering the state $\mathbf{x} \in \mathcal{X}$ and input $\mathbf{u} \in \mathcal{U}$, we can define a nonempty state constraint set $X_k \subset \mathcal{X}$ and a nonempty input constraint set $U_k \subset \mathcal{U}$ such that all possible values of \mathbf{x}_k and \mathbf{u}_k for $k \leq N$ and $N \in \mathbb{R}$ lie in X_k and U_k respectively. These state and input values are known as admissible state and admissible input values. For an MPC controller, both state and input trajectories must necessarily be admissible to be feasible. Therefore, we have

$$\mathbf{x} \in X \subset \mathcal{X} \quad \mathbf{u} \in U \subset \mathcal{U} \quad (3.62)$$

The k notation signifies that the constraint set may change from one iteration to another, however, we drop the notation for the sake of simplicity.

For optimization purposes, there are properties that the constraint sets must follow. They should be compact and convex. Further, the assumption of viability [14], which ensures the feasibility of the problem, must hold.

Assumption of viability: For each $\mathbf{x} \in X$ there exists $\mathbf{u} \in U$ such that $f(\mathbf{x}, \mathbf{u}) \in X$

where $f(\mathbf{x}, \mathbf{u})$ specifies the system dynamics. Assuming the assumption of viability holds and the closed-loop MPC controller satisfies the desired constraints, the feedback $\mu(\cdot)$ as defined in Eq. 3.52 is well defined and admissible [14].

When formulating the OCP, constraints are specified in the form of equality and inequality constraint equations as follows

$$\begin{aligned}
 &\text{equality constraints} \\
 &\quad g(\mathbf{x}_k, \mathbf{u}_k) = 0 \\
 &\text{inequality constraints} \\
 &\quad h(\mathbf{x}_k, \mathbf{u}_k) < 0
 \end{aligned} \tag{3.63}$$

For the case of AutoNaut, we consider physical constraints that limit rudder actuation within a preset range and constraints on surge velocity which ensures the vehicle always faces the direction of travel.

$$\begin{aligned}
 \text{abs}(\delta) &< \delta_{\max} \\
 u_k &> 0
 \end{aligned} \tag{3.64}$$

where

- u_k is the surge velocity at interval k as defined in Eq. 3.28.
- δ is the rudder angle (input) and δ_{\max} is the maximum deflection possible on either side.

Slack Constraints

For a stable and robust MPC design, it is essential to ensure feasibility in all operating conditions [38]. Feasibility in this context assures that the class of predictions available to the MPC algorithm can satisfy all the constraints simultaneously. However, feasibility can be lost due to improper modeling, tight constraints, disturbances, large set-point changes, etc.

Since feasibility is required for any meaningful practical application, we should reformulate the MPC problem to achieve feasibility by relaxing constraints when needed. This is possible as often systems have additional constraints imposed on them that may be overly restrictive and unnecessary when the MPC is designed with care.

In the case of AutoNaut, the physical constraints imposed by the rudder cannot be disregarded. However, we have the flexibility to allow the surge velocity to be less than zero. Typically, the surge velocity of the vehicle is always greater than zero, as moving backward is undesirable. Nevertheless, in order to prevent infeasibility and avoid loss of control at very low speeds, it is preferable to relax this constraint. This relaxation can be achieved by introducing slack variables, as demonstrated below.

We start by introducing the slack variable s and rewriting the surge velocity constraint as

$$\begin{aligned}
 \mathbf{s}_k + u_k &> 0 \\
 \mathbf{s}_k &> 0
 \end{aligned} \tag{3.65}$$

where u_k is the surge speed at interval k with $k = 0, 1, 2, \dots, N$. The slack variables are then added to the cost function 3.60 as follows

$$\begin{aligned} l(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_k^{\text{ref}}, \mathbf{u}_k^{\text{ref}}, \mathbf{s}_k) &= l^x(\mathbf{x}_{\text{ref}}, \mathbf{x}_k) + l^u(\mathbf{u}_k) + l^s(\mathbf{s}_k) \\ l^s(\mathbf{s}_k) &= \|\mathbf{s}_k\|_S \end{aligned} \quad (3.66)$$

where $S \gg Q, R$ and is a very large number.

Under normal conditions, the surge is greater than zero and hence $s_k = 0$ is the optimal value. When the vehicle operates at very low speeds, the optimizer may assign s_k a positive value to satisfy the constraint in Eq. 3.65 and prevent infeasibility, at the expense of higher cost.

3.4.5 Problem Formulation

In this section, we combine the different elements of the OCP introduced in the earlier sections to construct the NonLinear Programming (NLP) problem. Let us rewrite the state and input vector \mathbf{x} and \mathbf{u} as expressed in Eq. 3.28

$$\mathbf{x} = \begin{bmatrix} \psi \\ u \\ v \\ r \end{bmatrix} \quad \mathbf{u} = [\delta] \quad (3.67)$$

In terms of the discrete-time state space model and constraints, the optimal control problem with discretization time T_s and a fixed-width moving horizon T is then formulated as follows

$$\begin{aligned} \mathbf{x}^*, \mathbf{u}^*, \mathbf{s}^* &= \min_{\mathbf{x}(\cdot), \mathbf{u}(\cdot), \mathbf{s}(\cdot)} J \\ &= \min_{\mathbf{x}(\cdot), \mathbf{u}(\cdot), \mathbf{s}(\cdot)} \sum_{k=0}^{N-1} \left[l^x(\mathbf{x}_k, \mathbf{x}_k^{\text{ref}}) + l^u(\mathbf{u}_k) + l^s(\mathbf{s}_k) \right] + m(\mathbf{x}_N, \mathbf{x}_N^{\text{ref}}) + m^s(\mathbf{s}_k) \end{aligned} \quad (3.68)$$

where

- $T = NT_s$
- \mathbf{x}^* and \mathbf{u}^* are the optimal state and optimal input sequence
- the cost on input $l^u(\mathbf{u}_k)$ is defined as in Eq. 3.60 as

$$l^u(\mathbf{u}_k) = \|\mathbf{u}_k - \mathbf{u}_{k-1}\|_R \quad (3.69)$$

- the stage and terminal costs on state $l^x(\mathbf{x}_k, \mathbf{x}_k^{\text{ref}})$ and $m(\mathbf{x}_N, \mathbf{x}_N^{\text{ref}})$ are defined as in Eq. 3.61 as

minimizing course difference

$$m(\mathbf{x}_N, \mathbf{x}_N^{\text{ref}}) = l^x(\mathbf{x}_k, \mathbf{x}_k^{\text{ref}}) = \|\text{ssa}(\chi_k^{\text{ref}} - \psi - \beta)\|_Q^2 \quad (3.70)$$

aligning course vector

$$m(\mathbf{x}_N, \mathbf{x}_N^{\text{ref}}) = l^x(\mathbf{x}_k, \mathbf{x}_k^{\text{ref}}) = \|1 - \vec{\chi}_k^{\text{ref}} \cdot \vec{\chi}_k\|_Q^2$$

- the costs on slack variables $l^s(\mathbf{s}_k)$ and $m^s(\mathbf{s}_k)$ are defined by Eq. 3.66

$$l^s(\mathbf{s}_k) = m^s(\mathbf{s}_k) = \|\mathbf{s}_k\|_S \quad (3.71)$$

- subject to constraints given by Eq. 3.55, Eq. 3.64 and Eq. 3.65

$$\begin{aligned} \text{given } \mathbf{x}_0 &= \mathbf{x}(0) \\ \mathbf{x}_{k+1} &= \Phi(f^i(\mathbf{x}_k, \mathbf{u}_k)) \\ \text{abs}(\delta_k) &< \delta_{\max} \\ \mathbf{s}_k + u_k &> 0 \\ \mathbf{s}_k &> 0 \end{aligned} \quad (3.72)$$

- where f^i represents the dynamics of the vehicle as given by Eq. 3.9 and Eq. 3.25.

We note that the OCP for the generalized 2DOF Nonlinear dynamics is obtained by considering the state variable \mathbf{x}_2 given by Eq. 3.29, dynamics given by Eq. 3.22 and removing the slack constraints from Eq. 3.72 and slack variables from the cost function in Eq. 3.71.

The optimal control input at time t is then calculated using the mapping as described in eq. 3.52

$$\mathbf{u} = \mu(\mathbf{u}^*) = \mathbf{u}^*(0) \quad (3.73)$$

We note that the notation of (t) has been dropped to simplify the problem.

Warm Start

The solution to a well-posed problem may still require significant computation time to be accurately determined. If this is the case, it might be prudent to inquire about a good initial guess that our optimizer can use to converge to the optimal solution. The initial guess may not need to be a full or partial optimal solution, but only reasonably accurate. Given that our NLP problem at any given time-step is closely related to the NLP problem at the previous time-step, we can construct our guess using the past solution. This is a valid assumption if we set our discretization period to be significantly smaller compared to the dynamics of the plant and the system. Assuming multiple shooting, the guess is constructed by

1. discarding the first element from the predicted state and input trajectory.
2. shifting the prediction state and input trajectory by one time step to the left.
3. repeating the last element of the state and input trajectory.

The first guess is created by setting the states and input vector as a randomized array between 0 and 1, not including zero.

The NMPC Algorithm

Based on the above, the NMPC algorithm implemented in this work is expressed in Algorithm 1.

Algorithm 1 NMPC Algorithm

```

1: for each sampling instant  $t_n$  with  $n = 0, 1, 2, 3 \dots$  do
2:   Measure the state  $\mathbf{x}(t_n) \in X$  of the system and environmental parameters.
3:   Construct the initial guess of the state and control sequence as described
   in Sec. 3.4.5.
4:   Set  $\mathbf{x}_0 = \mathbf{x}(t_n)$ .
5:   Solve the optimal control problem as described in Sec. 3.4.5.
6:   if the solver was successful then
7:     Obtain the optimal state and optimal control sequence  $\mathbf{x}^*$  and  $\mathbf{u}^*$ .
8:   else if the solver was unsuccessful then
9:     Obtain the sub-optimal control sequence  $\mathbf{u}^*$ .
10:  end if
11:  Define the NMPC feedback value using the mapping described in Eq. 3.73.
12: end for

```

3.4.6 Sub-optimal NMPC

To ensure efficient and reliable practical implementation, it is crucial to ascertain the maximum execution time of the NMPC controller. This information guarantees that the system meets the available computational requirements and avoids any potential resource exhaustion. Determining a strict bound on the number of iterations needed for the convergence of the nonlinear programming problem is crucial for estimating the worst-case execution time. However, finding a non-conservative bound is a challenging task because of two key reasons[15]

1. When constraints are present, efficient solvers for NMPC may encounter difficulties in converging to the optimal solution. Soft constraints can be relaxed to some extent, but hard constraints do not have the same flexibility and can result in infeasibility. In such cases, the controller fails to produce a viable solution.
2. In real-time applications, the system may face unpredictable states which often lead to the number of iterations at one interval being dramatically different from the next. In such cases, a bound suitable for one might not work for another, especially in edge cases.

A method for setting a hardbound is to apply a sub-optimal solution after the number of iterations exceeds the preset threshold. This is possible due to the fundamental result as shown in [15] which states that feasibility and descent (reduction in cost function compared to the control trajectory computed at the previous sample) is sufficient for asymptotic stability of NMPC provided that terminal constraints are included in the formulation.

The preset threshold may be obtained by looking at the average number of iterations needed in the normal operating range and the frequency specification of the controller.

3.4.7 Tuning

For good and robust performance, the optimization problem should be well-scaled. While this can mean many different things, in the case of tuning, it is advisable that the different components of the OCP are of a similar order. As per sec. 3.4.3, the objective function at any given stage can be expressed as

$$l_k = l_k^x(\mathbf{x}_k^{\text{ref}}, \mathbf{x}_k) + l_k^u(\mathbf{u}_k) \quad (3.74)$$

Assuming a maximum rotation speed of $r_\delta = \pm 0.1\pi \frac{\text{rad}}{\text{sec}}$, we can compute the maximum costs associated with the change in input as follows

$$\begin{aligned} r_\delta^{\text{max}} &= 0.1\pi \\ l_k^u(\mathbf{u}_k)^{\text{max}} &= \|\mathbf{u}_k - \mathbf{u}_{k-1}\|_R \\ &= \|r_\delta^{\text{max}} T_s\|_R \end{aligned} \quad (3.75)$$

where T_s is the discretization time. With a discretization time of 0.5 sec, we have $l_k^u(\mathbf{u}_k)^{\text{max}} \leq 0.024R$ and $l_k^u(\mathbf{u}_k)^{\text{min}} = 0$.

In this report, we consider two cost functions as described in sec. 3.4.3. The maximum possible difference between the current and desired course angle is $\Delta\chi^{\text{max}} = \pi$ rad, the associated costs with the state cost functions in Eq. 3.61 can then be computed as

minimizing course difference

$$\begin{aligned} l_k^x(\mathbf{x}_{\text{ref}}, \mathbf{x}_k)^{\text{max}} &= \|\Delta\chi^{\text{max}}\|_{Q_1} = 9.8Q_1 \\ l_k^x(\mathbf{x}_{\text{ref}}, \mathbf{x}_k)^{\text{min}} &= 0 \end{aligned} \quad (3.76)$$

aligning course vector

$$\begin{aligned} l_k^x(\mathbf{x}_{\text{ref}}, \mathbf{x}_k)^{\text{max}} &= \|(1 - \vec{\chi}_k \cdot \vec{\chi}_k^{\text{ref}})\|_{Q_2} = Q_2 \\ l_k^x(\mathbf{x}_{\text{ref}}, \mathbf{x}_k)^{\text{min}} &= 0 \end{aligned}$$

since both $\vec{\chi}$ and $\vec{\chi}_d$ are unit vectors, the maximum value $(1 - \vec{\chi} \cdot \vec{\chi}_d)$ can achieve is 1.

Based on these calculations, we start our tuning with $R_1 = R_2 = 10$, $Q_1 = 1$, and $Q_2 = 10$ as we place a higher priority on deviations of the course angle than the use of the rudder. As these values are for reference only, our final tuning parameters are obtained through trials and are given in sec. 6.

Chapter 4

Method

This chapter introduces the tools and framework used to develop Model-In-the-Loop (MIL) simulation framework used to validate the proposed controller.

4.1 Software Framework: CasADi

4.1.1 Introduction

CasADi is an open-source software tool[16] that provides a symbolic framework for expression handling, efficient algorithmic differentiation for derivative computation, and capabilities for solving systems of ODEs, differential-algebraic equations, NLP problems and OCPs. It's development was started by Joel Andersson and Joris Gillis at KU Leuven, Belgium.

CasADi is available for multiple programming languages, including C++, Python, and MATLAB/Octave, with consistent performance across these interfaces and a syntax similar to computer algebra systems. The core driving force behind CasADi is to provide the necessary tools and flexibility to build efficient OCP solvers tailored to the user's specific needs.

4.1.2 Symbolic Framework in CasADi

CasADi incorporates a powerful symbolic framework at its core, providing users with the ability to construct symbolic expressions using a MATLAB-inspired syntax where everything is treated as matrices. The data types available in CasADi offer different capabilities for symbolic expressions:

1. *SX*: The *SX* data type is central to CasADi's symbolic framework and represents matrices composed of symbolic expressions created through unary and binary operations.
2. *DM*: The *DM* data type shares similarities with *SX* but represents matrices with numerical values instead of symbolic expressions. It is mainly used for efficient storage of matrices in CasADi and as inputs and outputs of functions.

3. *MX*: The *MX* data type provides a more versatile matrix expression format. *MX* allows the construction of expressions using a variety of elementary operations, beyond just scalar unary and binary operations.

By leveraging the *SX*, *DM*, and *MX* data types, we can effectively model and solve a wide range of optimization problems, benefiting from the versatility and computational advantages provided by CasADi's symbolic capabilities.

Function objects

Function objects in CasADi provide a powerful way to encapsulate various types of functions, such as symbolic expressions, ODE/DAE integrators, QP solvers, NLP solvers, and more. To create a function object, we use the syntax:

```
f = functionname(name, arguments, ..., [options])
```

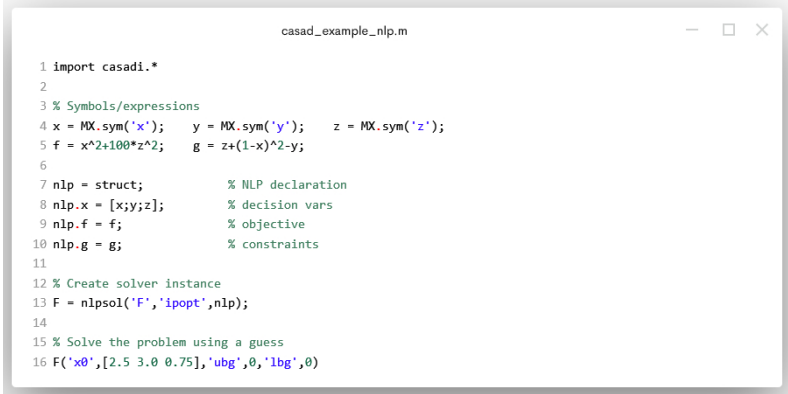
We have the flexibility to customize the behavior of the function object by passing an optional options structure. This allows for fine-tuning and adapting the function object according to specific requirements or preferences.

Example Problem

An example depicting the use of CasADi to solve a Nonlinear programming question is shown below. The example is taken from the CasADi website.

$$\begin{aligned} \min_{x,y,z} \quad & x^2 + 100z^2 \\ \text{subject to:} \quad & z + (1-x)^2 - y = 0 \end{aligned} \tag{4.1}$$

The above problem can be expressed using CasADi symbolic framework in MATLAB as



```
casad_example_nlp.m
1 import casadi.*
2
3 % Symbols/expressions
4 x = MX.sym('x'); y = MX.sym('y'); z = MX.sym('z');
5 f = x^2+100*z^2; g = z+(1-x)^2-y;
6
7 nlp = struct; % NLP declaration
8 nlp.x = [x;y;z]; % decision vars
9 nlp.f = f; % objective
10 nlp.g = g; % constraints
11
12 % Create solver instance
13 F = nlpsol('F','ipopt',nlp);
14
15 % Solve the problem using a guess
16 F('x0',[2.5 3.0 0.75],'ubg',0,'lbg',0)
```

Figure 4.1: Example NLP problem

4.1.3 Interior Point Optimizer Solver

Interior Point Optimizer (IPOPT) is an open-source optimization solver that is widely used in various scientific and engineering fields[39]. The solver is based on the primal-dual interior point method and is designed to solve nonlinear, non-convex optimization problems with inequality and equality constraints. IPOPT is a robust tool that can efficiently handle large-scale optimization problems. Its flexible and user-friendly interface allows user to define their optimization problem in a high-level modeling language like MATLAB or CasADi.

It can address the general nonlinear programming problems of the form

$$\begin{aligned} \min_{\mathbf{x} \in \mathcal{R}^n} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g^L(\mathbf{x}) \leq g(\mathbf{x}) \leq g^U(\mathbf{x}) \\ & \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U \end{aligned} \quad (4.2)$$

where

1. $\mathbf{x} \in \mathbb{R}^n$ are the optimization variables
2. \mathbf{x}^L and \mathbf{x}^U are the lower and upper bounds for \mathbf{x}
3. $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the objective function and is smooth.
4. $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ are the constraints with lower and upper bounds g^L and g^U

The mathematical implementation of IPOPT is based on an interior-point line-search filter method and can be found in [40], [41], and [42].

4.2 Model-In-the-Loop Simulation

This section introduces the Model-In-the-Loop (MIL) simulation setup used to test and validate the proposed controller using the numeric computing environment MATLAB™.

4.2.1 Introduction

MIL simulation is a powerful technique to validate and optimize the performance of complex systems [43]. It involves integrating and testing a system model within a simulation environment to assess its behavior, functionality, and performance before physical implementation. It finds wide use in various industries, including automotive, aerospace, robotics, and industrial automation.

In our setup, the simulation model is the mathematical representation of the system expressed using the CasADi framework and captures the system's dynamics, control actuators, and the environmental factors as described in Eq. 3.9. The developed framework in MATLAB™ provides inputs, simulates the system's response, and generates outputs for analysis.

The primary purpose of our simulation is to evaluate the controller's performance and validate its functionality at different operating conditions with a focus

on very low speeds. It is further useful to assess the impact of different objective functions and plant models with varying fidelity on NMPC performance. By leveraging MIL simulation, we were able to refine and optimize the controller iteratively, leading to an enhanced and robust system performance.

In our work, MIL simulation is an invaluable tool as most tools available for NMPC development make use of a high-level framework like MATLAB for design and development. With this capability, we can expedite the transition to real-world implementation, significantly enhancing the speed at which our developed solution can be packaged and deployed on an embedded platform.

4.2.2 Simulation Design

The simulation environment was designed using the numeric computation package MATLAB™. The system flowchart illustrating the basic logic flow of the developed environment is depicted in fig. 4.2.

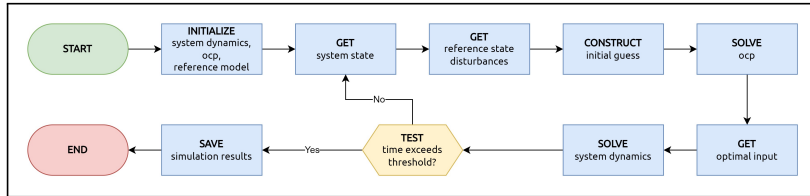


Figure 4.2: Flowchart of the MIL simulation

The simulation design is based on function-oriented design[44], which decomposes the model into interacting modules with well-defined functions, promoting modularity and clarity. The simulation begins by initializing key parameters set by the user, such as simulation run time, environmental disturbances (e.g., currents, winds), sensor noise, and trajectory generation configuration. Additionally, the user can specify MPC controller details, including the controller model, prediction horizon, and discretization time to initialize the optimal control problem.

Once initialized, the simulation enters a loop, starting from an initial state. During each iteration, the reference state and environmental disturbances are retrieved using functions `getReferenceState()`, `getCurrents()`, and `getWinds()`. To warm-start the optimal control problem, an initial guess is constructed using the function `initializeOCPGuess()`.

Next, utilizing the current state, reference state, and knowledge of environmental disturbances, the optimal input is obtained by solving the OCP through the function `solveCasadiOCP()`. The system dynamics are then simulated one time step forward using the function `ship()`¹, resulting in the evolved system state. This loop continues until the simulation concludes.

¹This function is obtained by modifying the assignment handout files provided in the course TTK4190, Guidance Navigation Control of Vehicles, taught by Thor Inge Fossen and Håkon Hagen Helgesen [45].

Upon completion of the simulation, the results are saved and plotted using functions like `saveSimData()` and `plotSimData()`, respectively. This allows for further analysis and visualization of the simulation outcomes.

A note of simulation and controller dynamic models

A model refers to a mathematical representation of the system and finds use in prediction, real-time simulation, decision-support systems, situational awareness as well as controller-observer design. The complexity of the model being used depends on the situation and design specifications. As per [24], models used in autonomous system design can be classified into three categories that are – (1) Simulation model, (2) Control design model, and (3) State estimator design model. Simulation models are often 6DOFs high-fidelity models for simulation of coupled motions in the time domain. Control design models on the other hand are usually designed using a simplified version of the simulation model.

In this work, we utilize a reduced-order 3DOFs nonlinear model using discretized dynamics based on Euler method integration for simulation and a similar model but using 4th Order Runge Kutta integration method for control design. Since both the simulation and controller may use the same model, any modeling errors in the simulation will propagate to the controller as well. Despite this limitation, testing NMPC in MIL simulations still provides significant benefits and has been widely recognized in the research community.

One advantage of using MIL simulations is the ability to evaluate the control performance in a realistic but controlled environment. MIL simulations allow for comprehensive testing and validation of the control strategy before deploying it in real-world applications [46]. This approach helps in identifying potential issues and improving the controller's performance and robustness by assessing the NMPC controller's behavior under different scenarios and operating conditions. The simulation environment allows for easy modification of system parameters, disturbances, and constraints, enabling thorough testing of the controller's response to various scenarios that may be challenging to replicate in physical experiments[47].

Furthermore, MIL simulations provide a platform for evaluating the NMPC controller's sensitivity to modeling errors and uncertainties. The controller can be tested with different levels of model fidelity, ranging from simplified models (linear models) to more complex and accurate representations (3DOFs nonlinear models) of the plant dynamics. This helps in understanding the impact of modeling assumptions and uncertainties on the controller's performance and assists in refining the model and controller design [48].

Chapter 5

Implementation

This chapter introduces the tools and framework used to design, develop and integrate the proposed NMPC controller with the existing software system onboard the vehicle. This helps us determine the feasibility of operating the controller for online and real-time applications.

5.1 Software Package: *acados*

acados is a collection of solvers for fast optimization applications on embedded hardware for efficient implementation of OCPs[49]. The core of is based on top of a high-performance linear algebra library that facilitates real-time and online implementation. The tool-chain emphasizes on values of modularity, flexibility, and rapid prototyping. This is realized by interfacing the high-performance core with higher-level languages such as Python and MATLAB allowing users to quickly put together different algorithm components that can be readily connected and interchanged.

acados is publicly available under the free and open-source BSD 2-Clause license. This license grants users the freedom to use, copy, modify, and distribute the software with source code, both for non-profit and commercial purposes, without any restrictions. It is developed by the Systems Control and Optimization Laboratory, at the University of Freiburg, led by Prof. Moritz Diehl.

5.1.1 Why *acados*

acados is a highly versatile open-source toolbox that has been proven effective in multiple projects both at NTNU and in the wider industry [50], [51] and [52]. By using *acados*, we can take advantage of its proven track record and efficient algorithms, thus reducing development time and ensuring reliable and effective results. Its open-source nature also means that we have access to a wealth of community support and development resources, further enhancing its value and justifying it as a tool for our optimization-based control system.

5.1.2 Algorithm Implementations in `acados`

`acados` combines together several software packages to handle the optimization problem. The core linear algebra library in `acados` is BLASFEO[53] and Quadratic Programming (QP) problems are by default solved by the HPIPM[54] library. The solver used in `acados` is a type of iterative method for solving constrained nonlinear optimization known as the Sequential Quadratic Programming (SQP) solver.

This method involves a series of QP sub-problems, each of which optimizes a quadratic model of the control objective subject to a linearization of the constraints. It is important to note that SQP methods are used on mathematical problems for which the objective function and the constraints are twice continuously differentiable. A typical SQP method features (1) numerical integration routines for continuous-time dynamics, (2) generation of first- and second-order sensitivities of objectives and constraints, (3) tools for approximating the Hessian matrix, and (4) an efficient QP solver. `acados` offers two different SQP-like methods, a full step SQP method and a specialized Real Time Iteration (RTI) routine with different algorithmic options.

5.1.3 Workflow with a High-Level Language Interface

OCPs are often coded in scripting languages as direct integration with the core C-based modules can be cumbersome and error-prone. `acados` provides an interface to two popular languages for scientific computing – Python and MATLAB/Octave. This section describes the process to set up the OCP using using the MATLAB interface as depicted in Fig. 5.1.

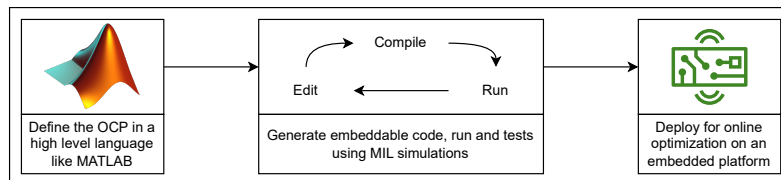


Figure 5.1: `acados` Workflow using a high-level scripting language.

The process of formulating the OCP can be divided into three parts that are the formulation of (1) the system dynamics, (2) the objective/cost function, and (3) the system constraints. The generally nonlinear functions that describe the various parts of the OCP are modeled using CasADi. This provides both the ability to quickly generate C functions from human-readable code and the ability to compare performance with other optimization tools interfaced with CasADi. The different parts of the OCP are then wrapped into a single object by calling the module `acados_ocp_model()`.

Once the OCP to be solved is described in the domain-specific language, an instance of the solver options module `acados_ocp_opts()` is used to define the solver-specific options. This can now be used, together with the model to create

an instance of the solver using `acados_ocp()` module. After defining the solver instance, a human-readable self-contained c project that makes use of templated code is generated. The generated project contains all the C code necessary for function and derivative evaluations generated through CasADi and the C code necessary to set up the NLP solver using the acados C interface.

With the workflow described above, it is possible to obtain a self-contained high-performance solver that can be easily deployed on embedded hardware starting from a description of the OCP in a high-level language.

Results and Discussions

This section depicts the result of the NMPC controller developed using acados. Fig. 5.2 and fig. 5.3 show the controlled course angle and the computed optimal rudder input respectively. The parameters used for the simulation are described and given in tbl. 6.1.

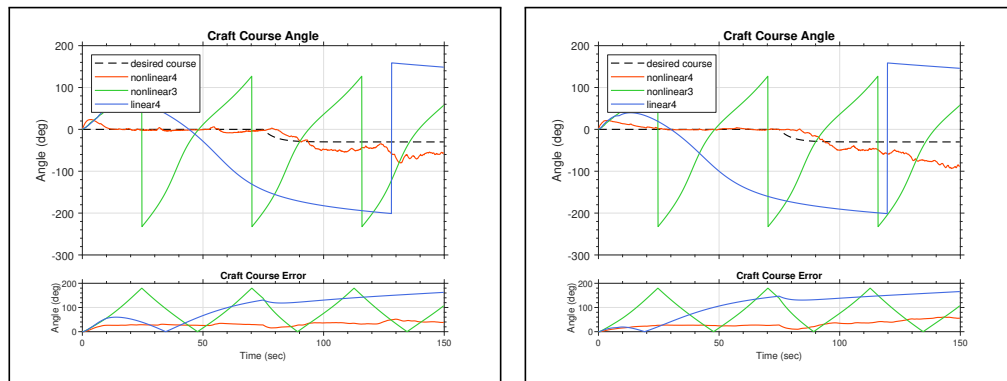
(a) χ_d (b) $\dot{\nu}$

Figure 5.2: Course Angle using Acados Solver

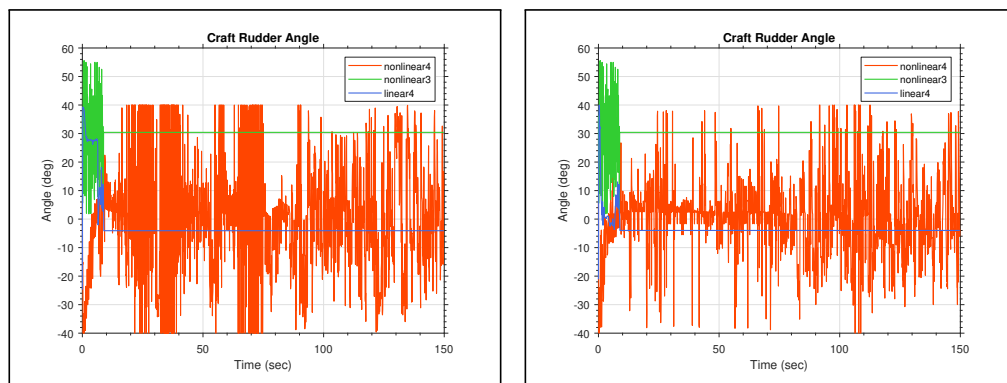
(a) χ_d (b) $\dot{\nu}$

Figure 5.3: Input Rudder Angle using Acados Solver

As we can see, the solver fails and is not able to converge to a stable trajectory. While the time required to solve the problem is low, any introduction of disturbances makes the solution diverge and the vehicle loses controllability.

Troubleshooting Controller Issues

In our efforts to troubleshoot controller issues, we explored various strategies outlined below. While some of these measures partially improved controller performance and mitigated convergence issues, they did not provide a complete resolution. We have documented these attempts for reference and to ensure a comprehensive record of our troubleshooting process.

Change of variables As discussed in [9], the course dynamics of the vehicle approach singularity when the SOG or the surge speed tends to zero. This causes the arctan function to get undefined. In order to transform the problem into a more well-behaved form that is easier to solve numerically, we modify the surge variable as shown below. This can help to shift the singularity away from the origin and make the problem easier to solve.

$$u_e = u + \epsilon \quad (5.1)$$

where ϵ is a small constant.

Regularization is another method to improve the numerical properties of the MPC optimization problem and is achieved by introducing penalty terms or constraints discouraging ill-posedness or behaviors leading to singularities.

The choice of regularization depends on the specific problem and the characteristics of the system being controlled. We introduce two methods to address solution convergence.

1. A *Barrier Cost* $l^b(\mathbf{x}_k)$ that penalizes control inputs that violate undesired behavior. In this case, the additional cost imposed by the barrier rapidly increases as the surge speed tends to zero but is zero otherwise. The modified cost function is given below

$$\begin{aligned} l(\mathbf{x}_k, \mathbf{x}_k^{\text{ref}}, \mathbf{u}_k) &= l^x(\mathbf{x}_k, \mathbf{x}_k^{\text{ref}}) + l^u \mathbf{u}_k + l^b(\mathbf{x}_k) \\ l^b(\mathbf{x}_k) &= \lambda \ln(u_k) \end{aligned} \quad (5.2)$$

where $\lambda > 0$ and we have used the simplified cost function without slack costs.

2. *Box Constraints* to regularize the OCP by introducing hard constraints that to avoid regions of singularity. This was done by constraining the surge speed to be always greater than ϵ .

The default **QP Solvers** employed by is the partial condensing HPIPM solve. We experimented with two additional QP solvers, namely full condensing HPIPM solver and qpOASES[55], but observed only marginal improvements. This suggests that either the QP being solved is ill-posed or that the issue does not stem from the solver itself.

Additional Solver Options: In the majority of cases, the default solver configuration, combined with a well-posed problem, is sufficient to effectively solve the OCP. However, despite our attempts to modify additional solver parameters such as Hessian Regularization, globalization, Exact Hessian, and Levenberg Marquardt, we did not achieve the desired outcome.

Reasons for Abandoning the Solver

Despite spending a month with little visible progress, we recognized the time limitations of the project and decided it was more worthwhile to explore alternatives. We opted to use the more robust but expensive IPOPT solver as described in Sec. 4.1.3.

5.2 Software Toolchain: LSTS/DUNE

The autonomy software onboard the AutoNaut utilizes a powerful open-source software toolchain developed by the Laboratório de Sistemas e Tecnologia Subaquática (LSTS) [56]. As an interdisciplinary research laboratory established in 1997, LSTS brings together experts from fields such as Electrical, Computer, and Mechanical Engineering. They specialize in the design, construction, and operation of unmanned underwater, surface, and air vehicles, including swarms.

5.2.1 Introduction

The LSTS toolchain is a comprehensive software framework specifically designed to support the operations of networked vehicle systems, enabling seamless communication and collaboration between autonomous systems and human operators. It addresses the unique challenges faced by autonomous systems operating in communication-restricted environments, such as dynamic network topologies, fluctuating bandwidth and latency, and uncertainties in state estimation.

The architecture of the LSTS toolchain adopts a flexible and layered approach, accommodating diverse mission scenarios. It incorporates a control hierarchy consisting of different layers (see Fig. 5.4), each responsible for encapsulating specific functional details and providing standardized interfaces for state retrieval and command execution. This layered approach fosters system interoperability at multiple levels of control, ensuring efficient coordination and cooperation among different components.

The LSTS toolchain comprises three essential software components. The first component is the Neptus shore-side Command and Control (C2) framework, which serves as the interface for operators to monitor and control the networked vehicle system. The second component is the onboard vehicle control software known as DUNE, which executes control algorithms and manages the vehicle's operations. Finally, the toolchain utilizes the Inter-Module Communication (IMC) protocol

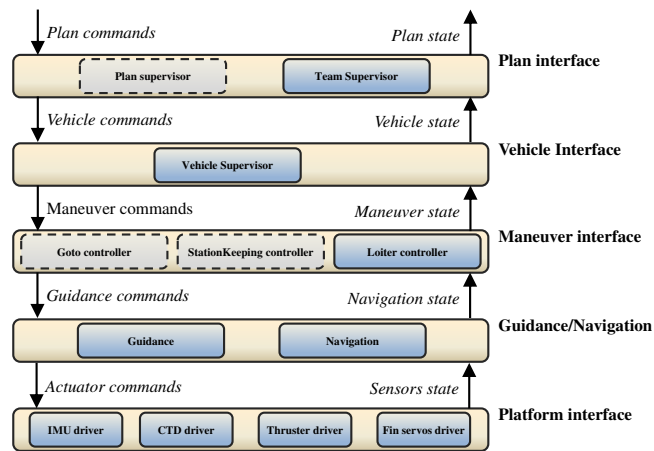


Figure 5.4: Example architecture implementation and possible switching between active/inactive controllers. Image taken from [56]

as a shared communication standard, enabling seamless data exchange and coordination between different modules and systems within the networked vehicle architecture.

Through its comprehensive and modular design, the LSTS toolchain empowers networked vehicle systems with robust and adaptable capabilities, facilitating efficient and effective mission execution in challenging environments.

DUNE

DUNE, which stands for DUNE Unified Navigation Environment, is a runtime environment designed for unmanned systems' on-board software [56]. It serves as a versatile platform for developing generic embedded software components that form the core of the system, encompassing various functionalities such as control, navigation, communication, sensor and actuator access, and more. Built in C++, DUNE offers a platform abstraction layer that is independent of both operating systems and hardware, ensuring portability across different architectures.

DUNE adopts a modular approach, isolating logical operations into individual tasks that run in separate threads of execution. These tasks follow a standardized life cycle, implementing specific methods for task execution and communication functions. Examples of common task operations include registering computational entities, updating task configurations, and consuming incoming messages.

DUNE categorizes tasks based on their base functions, allowing for a clear organizational structure. Task types include Sensors, Actuators, Estimators, Controllers, Monitors, Supervision, and Transports. These tasks communicate with one another through a message bus, where any task can forward IMC messages that can be consumed by one or all registered receivers. This flexible and scalable communication framework ensures effective information flow among tasks,

promoting system-wide coordination and collaboration.

Flexibility and adaptability are inherent features of DUNE. It can be deployed in different systems with varying configurations, providing users with the option to update configurations manually or through supervisor tasks. Additionally, users can define profiles that enable quick switching between pre-specified configurations, streamlining the process and enhancing the usability of DUNE in diverse operational scenarios.

Neptus

Neptus is a C2 software that serves as a tool for commanding and monitoring unmanned systems [57]. Developed in Java, Neptus is compatible with both Linux and Microsoft Windows operating systems. It utilizes the IMC protocol to establish communication and provides a coherent visual interface for commanding a wide range of IMC-based autonomous systems.

Two noteworthy features of Neptus are control abstraction and adaptability. Neptus implements an abstraction layer that simplifies the operator's interaction with different assets, without getting entangled in the specifics of each individual asset's capabilities. Additionally, Neptus provides flexibility by facilitating the rapid creation of derived tools and incorporating user-developed plugins. These features streamline command and control operations, across a diverse set of vehicles and mission plans.

The mission life-cycle in Neptus typically encompasses three distinct phases. The planning phase precedes mission execution and empowers operators to prepare mission plans and conduct preliminary simulations. During the execution phase, systems are readied for deployment, and mission execution, telemetry monitoring, and vehicle cooperation are carried out. Finally, in the review and analysis phase, collected data is processed, analyzed, and compiled for further dissemination, providing valuable insights and supporting informed decision-making.

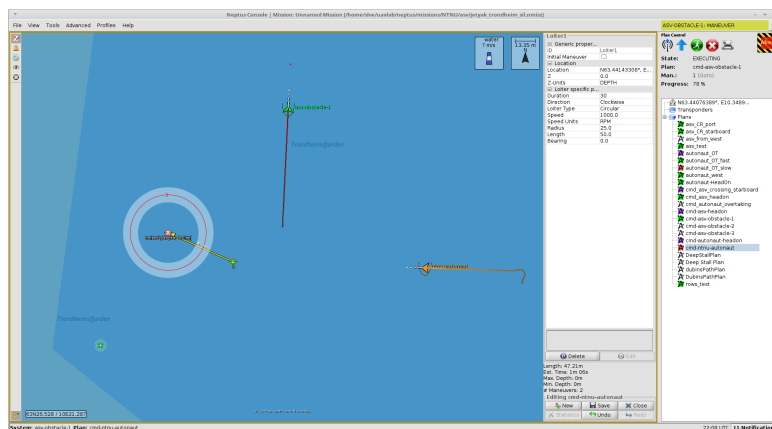


Figure 5.5: Neptus Operator Console. Image taken from [58]

Neptus provides access to its comprehensive feature set using the graphical user-friendly interface with the Neptus operator console, depicted in Fig. 5.5.

Inter-Module Communication Protocol

The Inter-Module Communication (IMC) is a versatile and flexible message-oriented protocol designed to facilitate seamless communication among various components in a distributed system. It serves as a means of inter-process, inter-vehicle, and operator-vehicle communication. IMC enables modular architecture by allowing different software components to operate independently while exchanging messages to interface with other modules.

One of the key advantages of IMC is its extensibility. It offers the capability to add new types of messages and events, both onboard and off-board, by maintaining a collection of protocol definitions in a single XML file. This enables developers to easily incorporate new functionality into the system without compromising compatibility. Furthermore, IMC is agnostic to the underlying communication systems and can seamlessly adapt to various communication technologies and protocols, making it highly adaptable to different operational environments.

IMC messages follow a structured format consisting of a header, payload, and footer. The header includes essential information such as a synchronization number, which allows for the detection of different byte order serializations and protocol versions. It also includes a message identifier, source, and destination, enabling efficient routing and processing of messages. The footer contains a checksum for ensuring message integrity and verification. The reader is referred to [59] and [60] for more information on IMC.

5.2.2 Overview of the System Architecture

The system architecture of AutoNaut, as illustrated in Fig. 5.6, provides an overview of the information flow and interactions within the system. The architecture encompasses various components and tasks responsible for processing sensor data, estimating the vehicle's navigation state and environmental parameters, generating mission plans, and controlling actuation.

At the edge of the system are the sensor controllers, which gather and process data from onboard sensors such as GNSS, Weather Station, ADCP, and IMU. This data is then utilized by the navigation tasks to estimate the vehicle's navigation state, including pose and twist, as well as environmental parameters like wind, currents, and waves.

The Estimated State information is subsequently consumed by several tasks, including Mission Supervisors, Vehicle Supervisors, Maneuver Controllers, and Guidance Controllers. These tasks utilize the Estimated State to plan future trajectories, ensuring collision-free navigation in accordance with the mission plan.

Once the desired course or heading is determined, the Guidance Controllers calculate the corresponding rudder angle and thruster actuation required to execute the planned trajectory. These actuation commands are then forwarded to

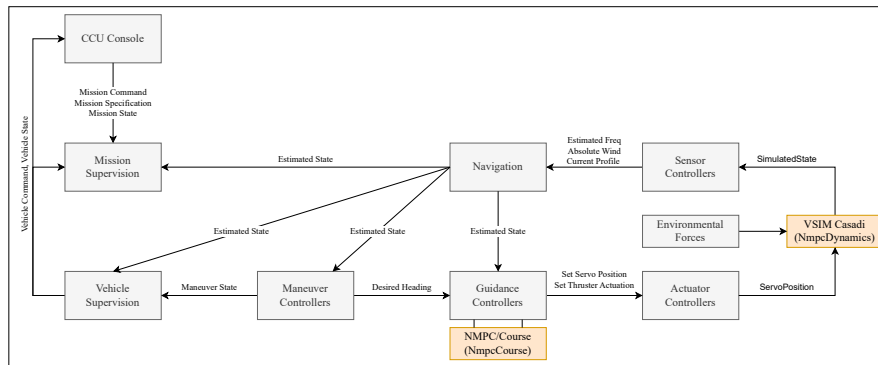


Figure 5.6: Overview of System Design

the actuation controllers for implementation. The communication and data exchange between the tasks occurs through the IMC bus, which facilitates the relay of relevant information.

For further insights into the design and construction of AutoNaut, we recommend referring to the source material [61] and [58] which provide detailed information on the development and implementation of the autonomy system on-board the AutoNaut.

5.3 Hardware-In-the-Loop Simulation

Hardware-In-the-Loop (HIL) is a powerful tool that allows for testing and validating complex control systems in a simulated environment that closely mimics the physical system [43]. HIL testing involves connecting a real-time simulation of the plant (hardware) to the controller (software) being developed, allowing for rapid iteration and verification of control system designs. By using HIL testing, engineers can reduce the cost and risk associated with testing on the physical system, and identify and correct issues earlier in the design process.

5.3.1 Integration with onboard System

In this project, we have developed and integrated two classes into the existing system of AutoNaut. These classes, namely `NmpeDynamics` and `NmpeCourse` were implemented using Object Oriented Programming (OOP) concepts.

The `NmpeDynamics` class focuses on simulating the vehicle dynamics in 3DOFs and is called within the `VSIM_CASADI` task. The task accepts actuator input and environmental conditions, such as rudder input from the `ServoPosition` message and data from `WaveProfile`, `AbsoluteWind` and `SingleCurrentCell` message. With this, the task uses the fourth-order Runge-Kutta method to compute the vehicle's state at the next time step. The computed state is then published at a user-defined rate using the `SimulatedState` message.

On the other hand, the `NmpcCourse` class is responsible for computing and publishing the optimal rudder angle for a given set of initial conditions and is called by the NMPC/Course Task. These initial conditions include the navigation state obtained from the `EstimatedState` message, environmental parameters from sources like `WaveProfile`, `AbsoluteWind` and `SingleCurrentCell`, and the desired course reference from the `DesiredHeading` message. As the online optimization process can be computationally expensive and the `AutoNaut` exhibits slow dynamics, the optimization frequency is slower than the publish rate. This is achieved by determining the optimal trajectory over the entire prediction horizon in the NMPC controller, which is then utilized until a time period is determined by the user. This approach balances computational resource utilization while ensuring access to optimal rudder angles. The optimization frequency and publish rate can be configured by the user through the `Solver Rate` and `Publish Rate` parameters.

The system flowcharts for both tasks are depicted in Fig. 5.7, illustrating the sequence of operations and data flow within each task.

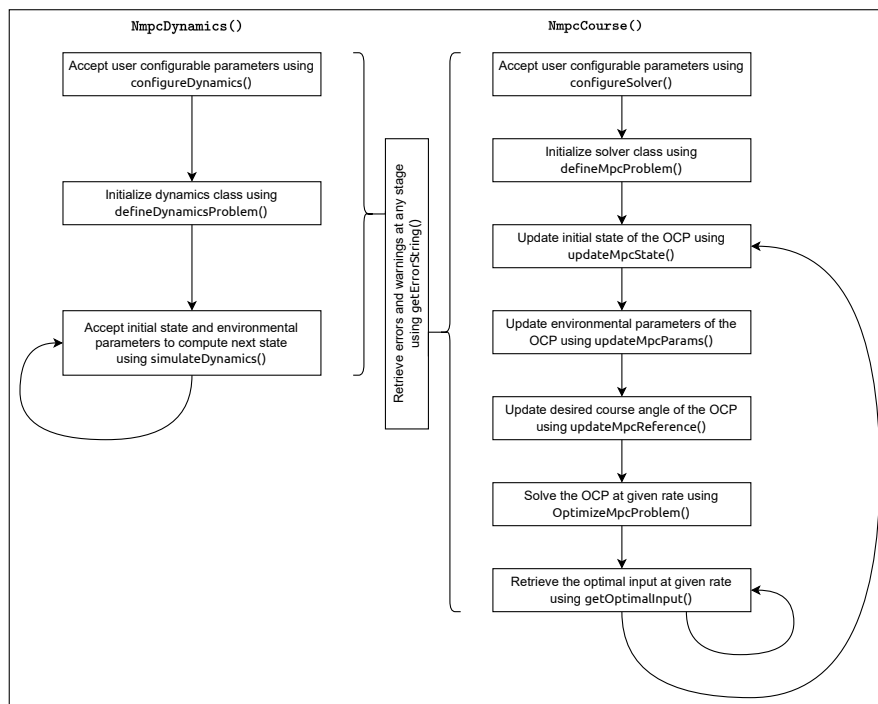


Figure 5.7: Flowchart of the class `NmpcDynamics` and `NmpcCourse`

5.3.2 Hardware Setup

The hardware design of the AutoNaut, which is discussed in [61], incorporates the TS-7800-V2 embedded system board[62] as the primary computation unit for the vehicle's control system. However, due to limited access to the actual vehicle, we implement our solution in a computationally comparable board, Raspberry Pi 4[63]. We simulate all the necessary systems on a single board. A depiction of the hardware setup can be found in Figure 5.8.

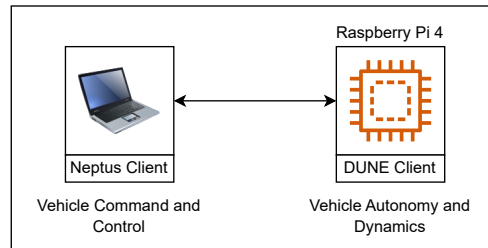


Figure 5.8: Setup for HIL Simulation

Chapter 6

Results

This section contains the result of MIL and HIL simulations as described in Sec. 4 and Sec. 5.

6.1 Test Plan

We are going to present three sets of results from MIL simulation and one set of results from HIL Simulation. In both cases, the simulation model is described by the discrete-time nonlinear equation of motions as given by Eq. 3.9. The simulation model integrates constant environmental forces of winds, waves, and currents as well.

To ensure an accurate assessment of the controller's performance, the simulation employs default parameters that represent a realistic scenario. Table 6.1 provides a comprehensive overview of all user-configurable parameters along with their corresponding nominal values. The sensor noise parameters control the standard deviation of zero-mean Gaussian noise incorporated into the state vector. Additionally, we introduce slack costs and constraints by setting $S \gg 0$. It is important to note that the default configuration does not employ the slack formulation as it is unnecessary when the vehicle is operating with speeds much greater than zero. Throughout the subsequent results, unless stated otherwise, all experiments utilize the default parameter values.

The simulation framework incorporates a reference model that generates a consistent course angle over a specified duration. Once a predefined threshold is reached, the angle is incrementally increased by a predetermined value. This iterative process persists until the simulation concludes. This comprehensive approach allows for a thorough assessment of performance under various conditions as the vehicle encounters both scenarios that are (1) where the propulsion force act with the environmental disturbances and (2) where the propulsion force acts against the environmental disturbances leading to very low speeds.

In this study, we analyze the performance of the system under three distinct conditions: ideal, nominal, and adverse. Under ideal conditions, the system operates without any modeling error, sensor noise, or external disturbances such as

Table 6.1: Default simulation parameters

S. No.	Parameters	Default Value	Description
Simulation Parameters			
1	h	0.1 sec	Discretization time for simulation dynamic model
2	Ns	3200	Simulation runtime (in ds)
3	x0	[0, 0.95, 0, 0, 0, 0]	Initial state
4	controller	'mpc'	Choose controller type from 'open', 'pid' and 'mpc'
Environmental Parameters			
5	currents.Vc	0.35 m/s	Current speed
6	currents.beta	90 degs	Current direction in NED
7	winds.Vw	5 m/s	Absolute wind speed
8	winds.beta	90 degs	Wind speed in NED
9	Hs	5	Wave strength
10	Tp	10	Peak wave time period
11	beta	90 degs	Wave direction
Sensor Parameters (noise.)			
12	mode	1	Flag to use noisy sensors
13	u	0.075 m/s	Noise parameter for surge speed
14	v	0.075 m/s	Noise parameter for sway speed
15	psi	1,5 degs	Noise parameter for heading angle
16	r	0.05 deg/sec	Noise parameter for heading rate
17	adcp.vel	0.075 m/s	Noise parameter for current speed
18	adcp.angle	1,5 degs	Noise parameter for current angle
19	wind.vel	1 m/s	Noise parameter for wind speed
20	wind.angle	5 degs	Noise parameter for wind angle
Reference Model			
21	deltaT	80 sec	Time to hold the desired course angle
22	deltaTh	90 degs	Incremented in desired angle at each time deltaTh
23	chi0	-45 deg	Initial desired course angle
24	Tchi	6 sec	First order reference model time constant
MPC Controller			
25	solver	'ipopt'	Choose solver from 'ipopt' or 'acados'
26	model_type	'nonlinear'	Choose model from 'linear' or 'nonlinear'
27	ref_type	'chi_d'	Choose objective function from 'chi_d' or 'dotv'
28	model_dim	4	Choose model dim from 3 or 4
29	Ts	0.5 sec	Discretization time for model dynamics
30	Tp	40 secs	Prediction horizon for MPC controller
31	Q	1.5	Cost for stage deviation
32	R	7.5	Cost for input deviation
33	S	0 (N/A)	Cost for slack variables
Post Processing Results			
34	plots	[1 2 3 4]	Choose data to plot. (1-heading, 2-full state, 3-path, 4-noisy states)
35	debug	false	Flag to plot results during runtime
36	save	false	Flag to save the results

currents and winds, which are assumed to be zero. Moving to nominal conditions, we introduce median expected currents, winds, and zero-mean white noise in the sensor, reflecting more realistic operating conditions. Finally, under adverse conditions, the external disturbances surpass the wave propulsion force, resulting in a significant reduction in the surge speed of the vehicle, often reaching near or below zero.

The dynamic models given by Eq. 3.9, Eq. 3.22 and Eq. 3.25 are named `nonlinear4`, `nonlinear3` and `linear4` respectively. The cost functions given by Eq. 3.38 and Eq. 3.44 are named `chi_d` and `dotv` respectively.

6.2 Results from MIL Simulation

This section only depicts results relevant to the discussion in chapter 7. Additional plots that may be of interest are given in the Appendix C.

6.2.1 Ideal Conditions

In an ideal scenario, we consider the vehicle to operate under conditions absent of any disturbances, with a perfect observer. This implies the absence of currents, winds, and any errors in the estimated state. The modified parameters for this condition are given in Tbl. 6.2. The results of the simulation are depicted graphically in fig. 6.1, 6.2, 6.3, 6.4 and 6.5.

Table 6.2: Modified parameters for ideal conditions

S. No.	Parameter	Default Value
1	mode	0
2	noise.currents.Vc	0
4	noise.winds.Vw	0
6	Q (chi_d)	10
7	R (chi_d)	1
8	Q (dotv)	0.5
9	R (dotv)	100

Model	Objective Function	RMSE (deg)	Mean Solver Time (sec)	Max Solver Time (sec)	Solver Success Rate (%)
<code>nonlinear4</code>	<code>chi_d</code>	14.318	0.022	0.323	100
<code>nonlinear3</code>	<code>chi_d</code>	14.749	0.019	0.167	100
<code>linear4</code>	<code>chi_d</code>	14.64	0.013	0.111	100
<code>nonlinear4</code>	<code>dotv</code>	14.448	0.025	2.478	100
<code>nonlinear3</code>	<code>dotv</code>	14.717	0.046	1.361	100
<code>linear4</code>	<code>dotv</code>	14.794	0.015	0.116	100

Figure 6.1: Summary of results in ideal conditions

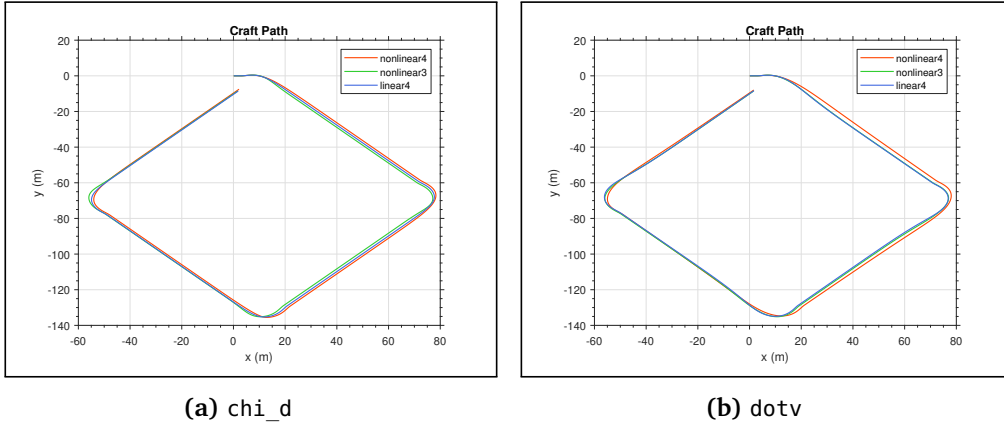


Figure 6.2: Vehicle path in ideal conditions

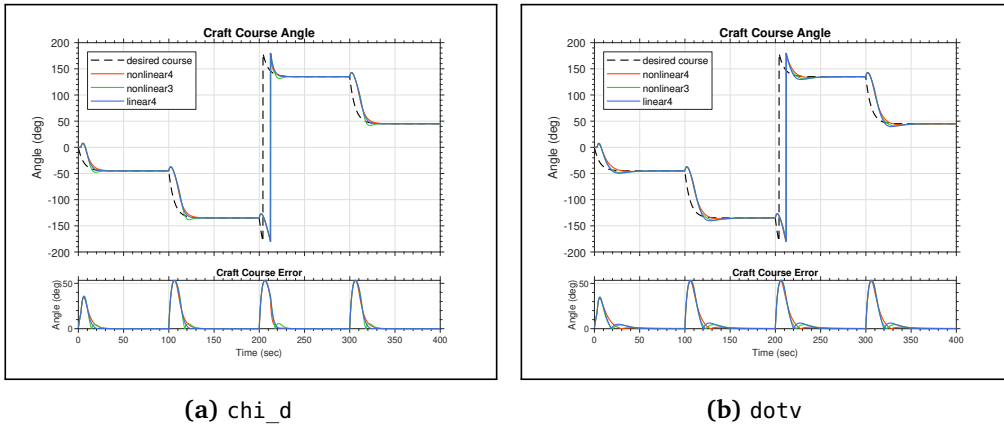


Figure 6.3: Course angle in ideal conditions

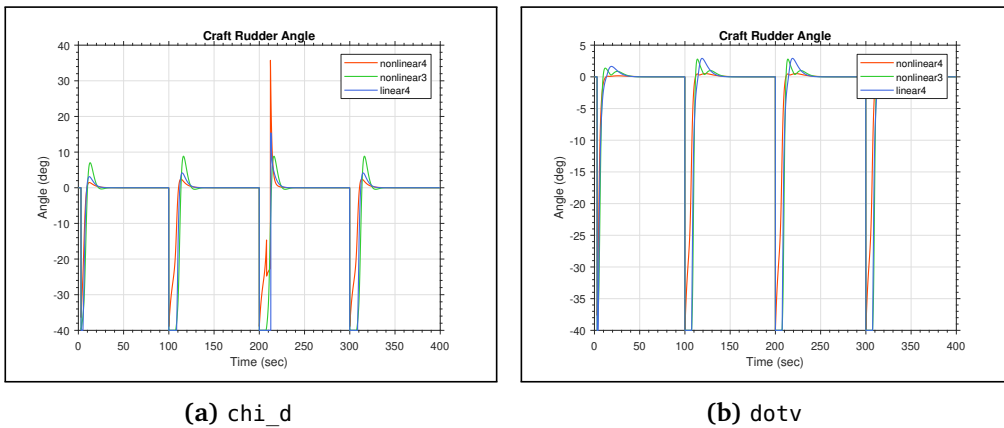


Figure 6.4: Input rudder angle in ideal conditions

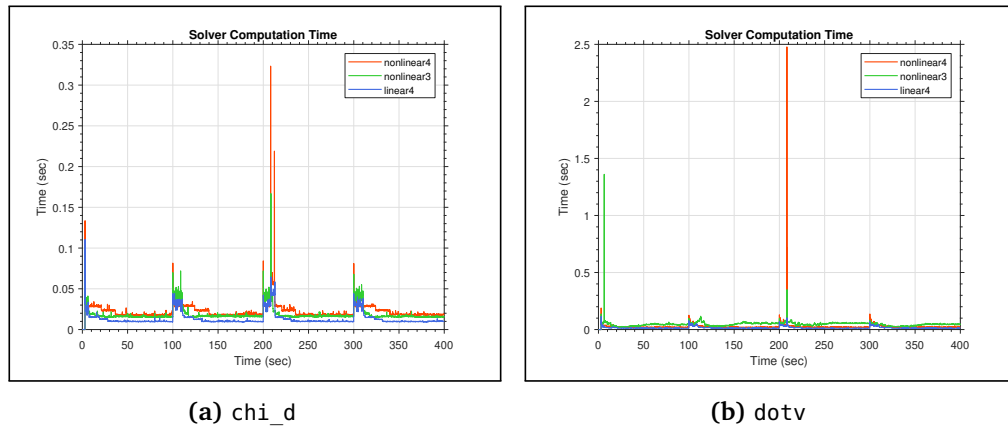


Figure 6.5: Solver computation time in ideal conditions

6.2.2 Nominal Conditions with Noisy Observer

In nominal conditions, we consider the vehicle to operate in an environment with constant median expected currents and winds with noisy sensors. The noisy state is constructed by adding Gaussian errors with zero mean and variance as given in Tbl. 6.1. The modified parameters for this condition are given in Tbl. 6.3. The results of the simulation are depicted graphically in fig. 6.6, 6.7, 6.8, 6.9, 6.10, 6.11 and 6.12.

Table 6.3: Modified parameters for nominal conditions

S. No.	Parameter	Default Value
6	Q (χ_d)	0.5
7	R (χ_d)	7.5
8	Q (\dot{v})	50
9	R (\dot{v})	7.5

Model	Objective Function	RMSE (deg)	Solver Time (sec)	Max Solver Time (sec)	Solver Success Rate (%)
nonlinear4	χ_d	24.838	0.035	0.176	100
nonlinear3	χ_d	26.551	0.023	0.108	100
linear4	χ_d	22.804	0.019	0.083	100
nonlinear4	\dot{v}	20.976	0.07	0.529	100
nonlinear3	\dot{v}	24.371	0.042	0.529	100
linear4	\dot{v}	20.274	0.026	0.169	100

Figure 6.6: Summary of results in nominal conditions

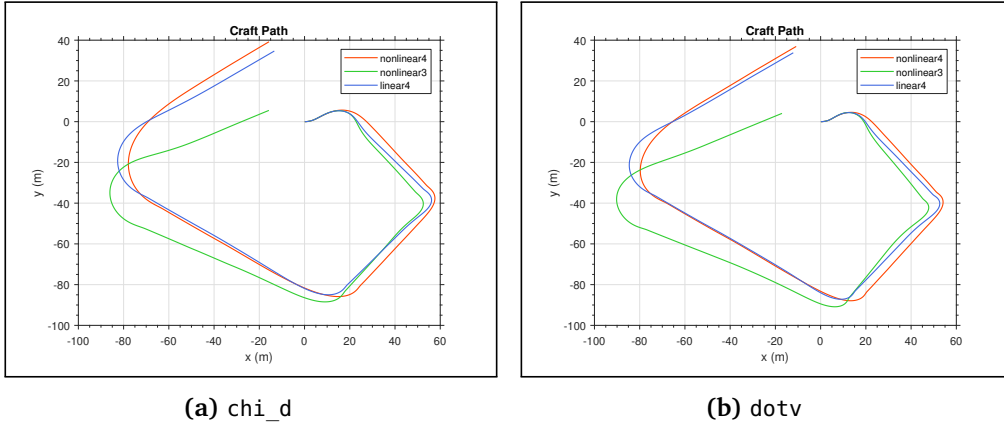


Figure 6.7: Path in nominal conditions

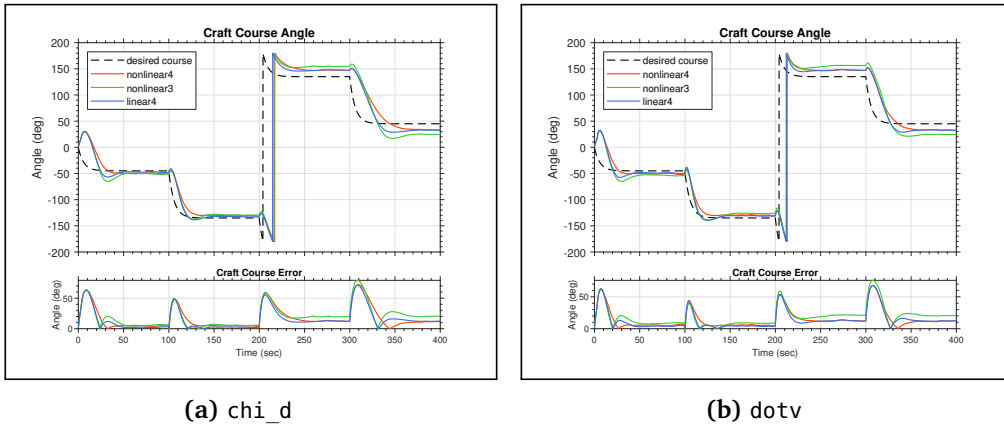


Figure 6.8: Course angle in nominal conditions

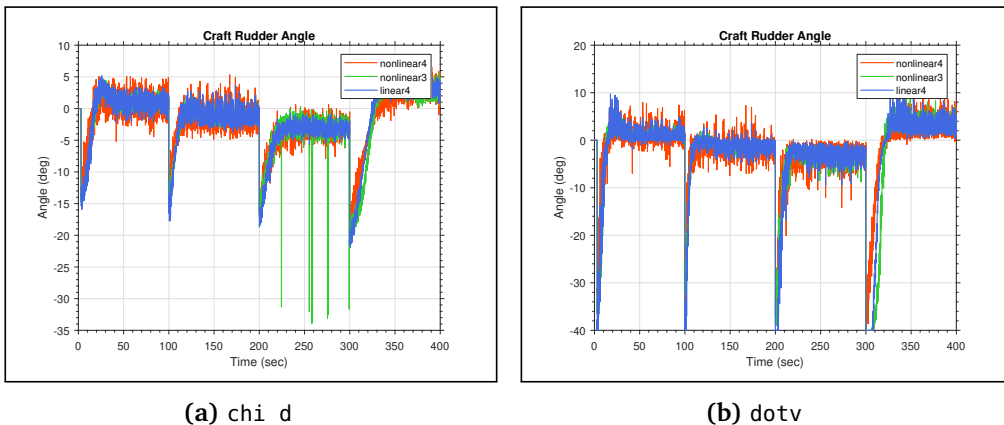


Figure 6.9: Input rudder angle in nominal conditions

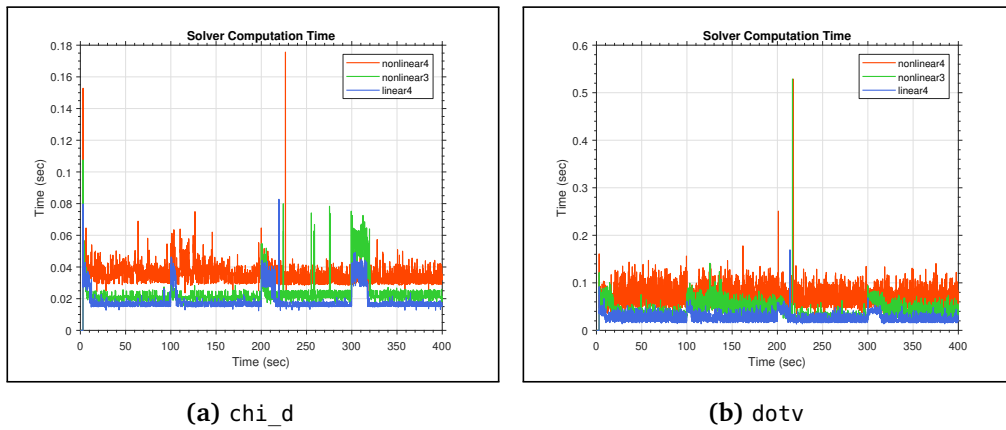


Figure 6.10: Solver computation time in nominal conditions

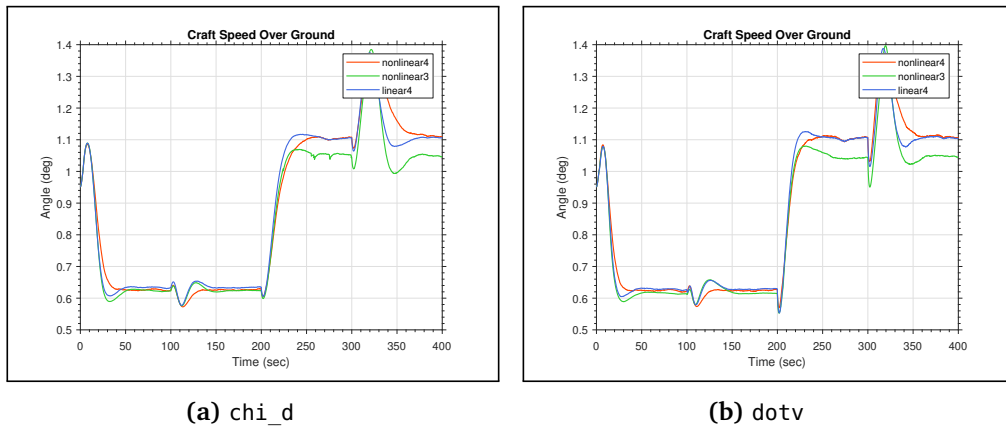


Figure 6.11: Speed Over Ground in nominal conditions

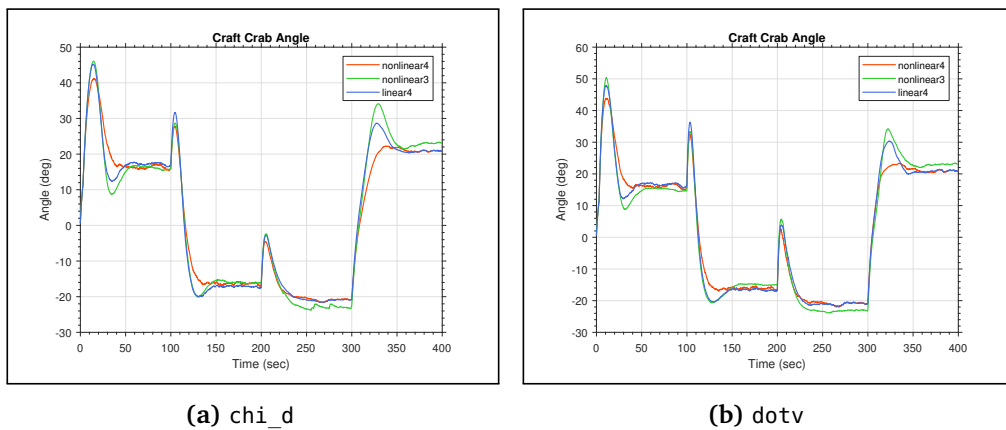


Figure 6.12: Sideslip angle in nominal conditions

6.2.3 Adverse Conditions with Relaxed Constraints

In adverse conditions, we consider the vehicle to operate in an environment with extreme currents and winds and with noisy sensors. Due to high disturbances, solving for an optimal course becomes difficult leading to increased solver time. To combat this, we restricted the solver rate to 2Hz. The rest of the modified parameters are given in Tbl. 6.4. For adverse conditions, it is useful to study the graph of SOG and crab angle (β_c) as well. The results of the simulation are depicted graphically in fig. 6.13, 6.14, 6.15, 6.16, 6.17, 6.18 and 6.19.

Table 6.4: Modified parameters for adverse conditions

S. No.	Parameter	Default Value
1	noise.currents.Vc	0.95 m/s
2	noise.winds.Vw	5 m/s
3	Ts	0.8
4	Q (chi_d)	4.75
5	R (chi_d)	7.5
6	S (dotv)	800
7	Q (dotv)	1.75
8	R (dotv)	7.5
9	S (dotv)	800

Model	Objective Function	RMSE (deg)	Solver Time (sec)	Max Solver Time (sec)	Solver Success Rate (%)
nonlinear4	chi_d	53.516	0.443	8.525	65.93
nonlinear3	chi_d	58.057	0.004	0.182	100
linear4	chi_d	50.492	0.021	4.863	98.58
nonlinear4	dotv	57.846	0.027	5.039	99.05
nonlinear3	dotv	84.661	0.005	0.425	100
linear4	dotv	60.124	0.009	0.275	100

Figure 6.13: Summary of results in adverse conditions

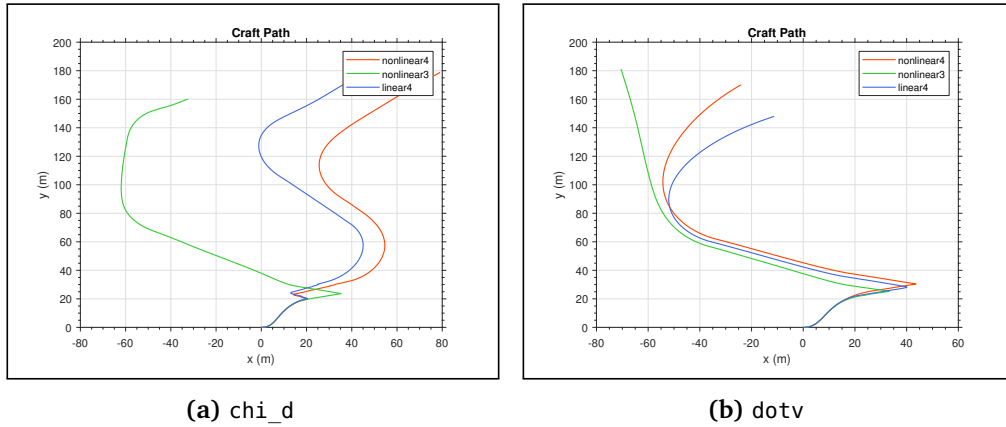


Figure 6.14: Path in adverse conditions

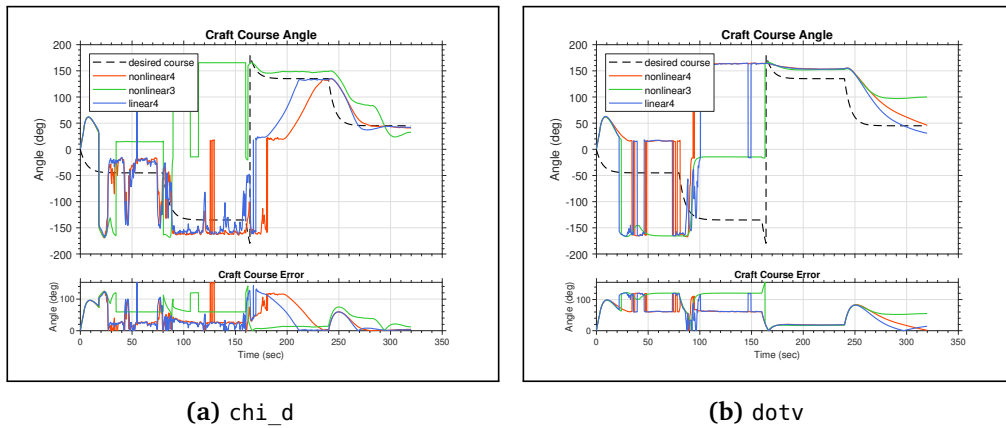


Figure 6.15: Course angle in adverse conditions

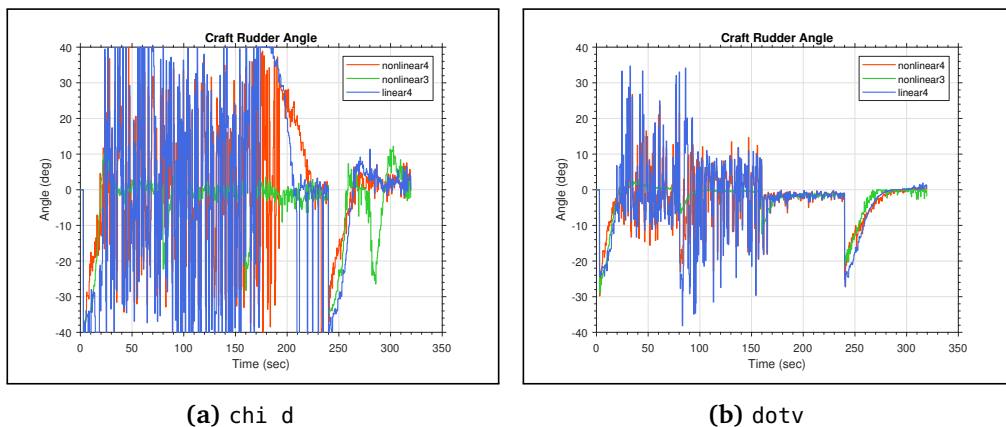


Figure 6.16: Input rudder angle in adverse conditions

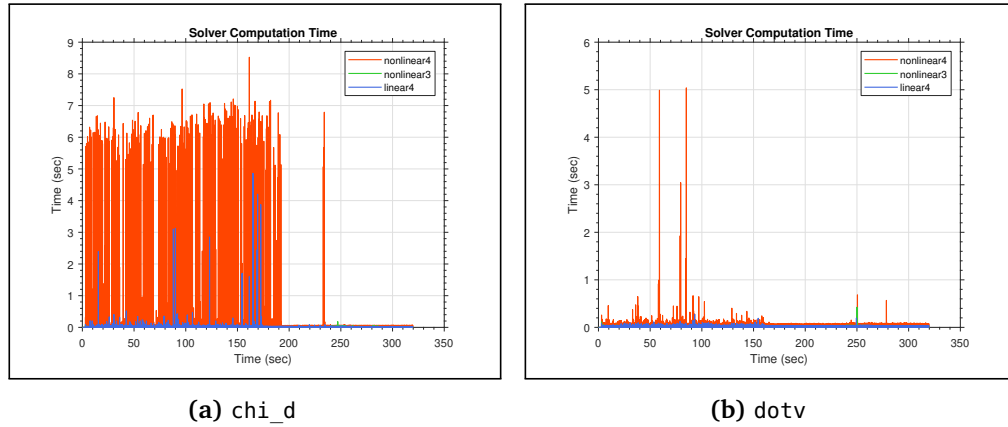


Figure 6.17: Solver computation time in adverse conditions

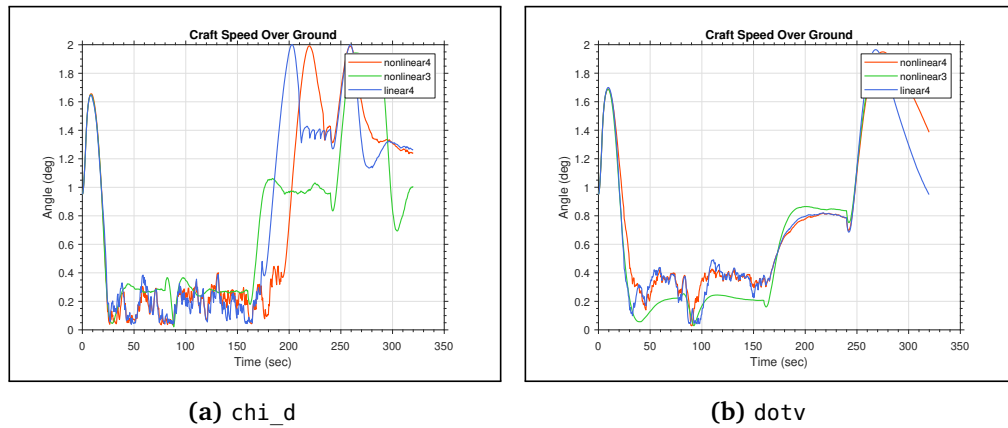


Figure 6.18: Speed Over Ground in adverse conditions

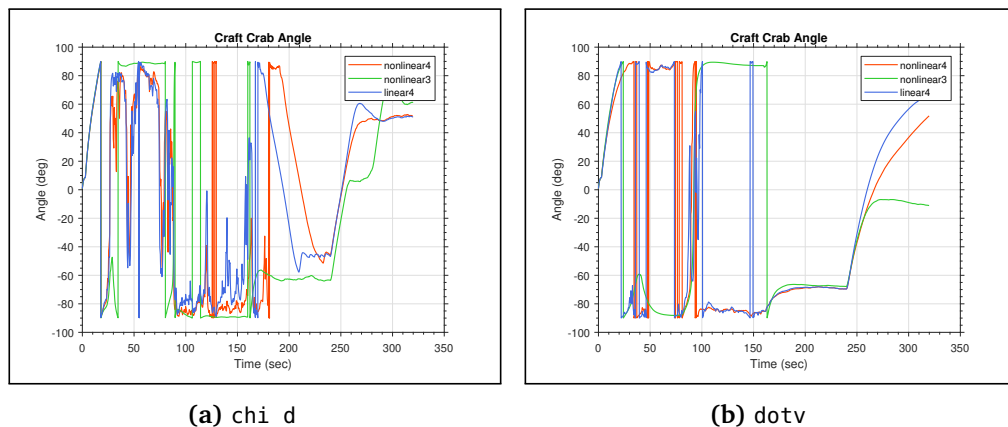


Figure 6.19: Crab angle in adverse conditions

6.3 Results from HIL Simulation

In our HIL simulations, we focused on simulating nominal conditions and evaluating the performance of on selected promising solvers. Unfortunately, we did not have sufficient time to conduct a comprehensive analysis of adverse conditions. Therefore, our results and findings primarily reflect the performance under expected operating conditions.

6.3.1 Nominal Conditions with Noisy Observer

For this simulation, we use the same parameters as described in tbl. 6.1 to plot performance of the course keeping controller using three solvers that are – `nonlinear4` with `chi_d`, `nonlinear4` with `dotv` and `linear4` with `dotv`.

The solver computation time is plotted as a histogram where the Solver Rate is set to 2Hz as before. The solver time for the results shown in fig. 6.20, 6.21, and 6.22 remains below the threshold for 99.9430%, 98.7365%, and 100% of the time, respectively.

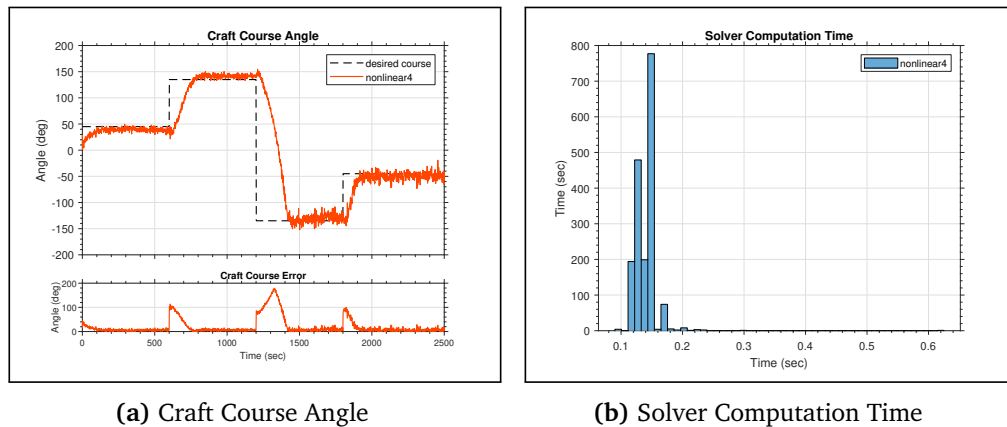


Figure 6.20: `nonlinear4` with `chi_d`

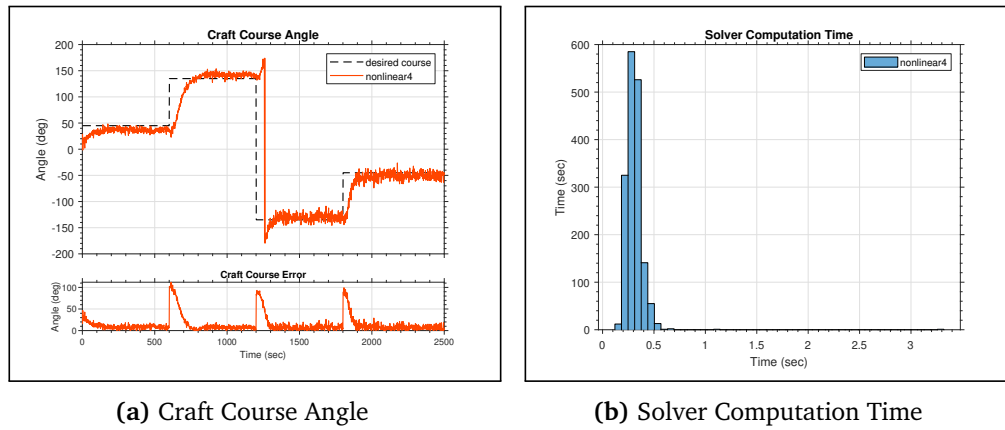


Figure 6.21: nonlinear4 with dotv

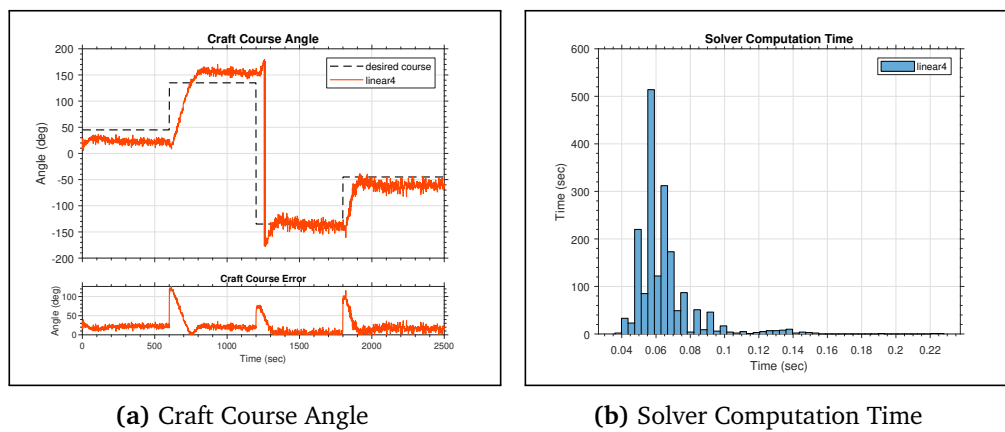


Figure 6.22: linear4 with dotv

Chapter 7

Discussions

In this section, we interpret and describe the significance of the results obtained in Sec. 6.

7.1 MIL Simulation

MIL simulations provide a rapid and efficient means of evaluating the feasibility of a controller within a realistic yet controlled environment. MIL simulation allows us to recreate and analyze controller performance in near-critical scenarios which is crucial in evaluating the controller's performance and robustness. To assess our desired behavior, the key aspects we focused on were

1. Low oscillations in course and rudder: Reducing excessive back-and-forth movements or instability in the course and rudder yields a smoother trajectory and more stable behavior.
2. Low RMSE: Minimizing the overall deviation between the desired and actual values results in more accurate system performance.
3. Low rudder effort: Minimizing the required effort to achieve the desired course angle maximizes vehicle endurance.
4. Fast settling time and low overshoot: A faster response time without overshooting or excessive corrections improves path-following behavior.
5. Low solver time: Reducing the time taken by the solver to converge to the optimal solution improves the real-time performance of the system. We set the solver frequency threshold at 2Hz which restricts us to a maximum solve time of 0.5 seconds.
6. High solver success rate: A high success rate of the solver ensures robust performance and consistently accurate and reliable solutions within the desired constraints.

7.1.1 Ideal Conditions

We focus on examining potential modeling errors, performing basic validation tests of the controller, and addressing implementation issues such as angle wrap and numerical stability. This discussion corresponds to the plots depicted in figs. 6.1, 6.3, 6.4 and 6.5.

The path traveled by the vehicle is depicted in fig. 6.2 where we see the effect of the proposed reference model. At each fixed interval, the bearing is constant resulting in straight-line paths. At the end of the simulation, the vehicle approaches the region of origin.

Course keeping with `chi_d`

In fig. 6.3a, we observe that the deviation in course angle is comparable across the three models – `nonlinear4`, `nonlinear3` and `linear4`, indicating similar performance. All models achieve zero steady-state error, validating the proposed controllers. As `nonlinear4` model has the highest fidelity among the three, it exhibits the shortest settling time and virtually no overshoot, while `linear4` and `nonlinear3` models exhibit comparable overshoot. All three models successfully identify the shortest path when the course changes from -135 degrees to 135 degrees. Furthermore, the absence of disturbances results in similar course-keeping behavior for all four reference angles. We note that the reference model neglects the vehicle's dynamics, leading to an initially infeasible course and a significant error in the course angle which rapidly reduces to zero.

In fig. 6.4a, we observe a large initial input rudder angle which is physically unrealistic. However, once the initial phase concludes, a smooth trajectory is observed, albeit with increasing overshoot as the model fidelity decreases. Outliers are observed in rudder input for all plots except `nonlinear3` after $T > 200$ which corresponds to the discontinuity in the heading. All models eventually settle to a constant rudder angle without any oscillations, indicating a stable and consistent behavior.

In fig. 6.5a, we see that the maximum solve time for all models is less than 0.32 seconds, while the mean solve time is less than 0.046 seconds. These results are promising for the proposed online implementation. From $t > 200$ secs, we notice that the outliers correspond to irregular rudder angles, which both linear and nonlinear models struggle with. The effects of warm start can be observed as well as the computation time increases during significant angle changes but rapidly decreases to around 0.01 seconds otherwise.

As anticipated, the nonlinear models exhibit higher computation times compared to the linear models but with only marginal improvement in performance. Therefore, further investigation into the impact of disturbances is necessary to enhance our understanding of the different controller's performance.

Course keeping with `dotv`

Compared to the objective function `chi_d`, several observations can be made in the fig. 6.3b. There is a marginal decrease in course-keeping performance across all models with a greater settling time compared to the `chi_d` cost function. However, aside from these differences, the objective function displays similar behavior as its counterpart.

As seen previously, all models exhibit large initial changes in the rudder with small overshoots for models with lower fidelity (`nonlinear3` and `linear4`). However, unlike its counterpart, the rudder does not take outliers indicating a more robust performance. It is worth noting that the solver still exhibits a higher solver time at $t > 200$ i.e. when facing the discontinuity in the heading. The high maximum time of the `dotv` solver could potentially pose challenges during practice. However, leveraging the sub-optimal solution in such cases may prove beneficial. Despite the exceptionally high maximum solver time, the mean solver time remains below 0.025 seconds, indicating overall efficient performance.

All proposed solvers exhibit comparable RMSE values. Although the error is lowest for the `nonlinear4` model with the `chi_d` cost function, the difference is not necessarily significant. It is worth mentioning that both cost functions have outliers in solver time; however, these outliers remain within acceptable limits. For 3DOFs models, the presence of outliers could be attributed to numerical instability caused by the arctan function in Eq. 3.16, as the solver aims to converge optimally in a specific direction. In the case of 2DOFs models, the omission of surge dynamics might contribute to the outliers. Since surge dynamics are predicted and may not accurately reflect reality, conducting field trials becomes crucial to validate this approach thoroughly.

7.1.2 Nominal Conditions with Noisy Sensors

The primary objective of this test is to evaluate the controller's performance under realistic conditions by subjecting the controller to noise and disturbances. This discussion corresponds to the plots depicted in figs. 6.6, 6.7, 6.8, 6.9, 6.10 and 6.11.

In fig. 6.7, we immediately see significant drift compared to the path in fig. 6.2b. The success of the model is measured by a constant straight-line motion during the fixed time interval. Since we only observe drift and oscillations while turning, the steady-state course-keeping behavior is effective.

Course keeping with `chi_d`

For fig. 6.8a, it is immediately apparent that all the models exhibit larger overshoot and settling time due to the action of disturbances. However, the controller quickly converges to a steady state with a significant quasi-constant error (at $\chi_d = 45$ and $\chi_d = 135$) in the course angle. Tests indicate that the steady-state error can be

reduced by increasing the prediction horizon of the controller or by increasing the cost on the rudder, albeit at the expense of rudder oscillations.

In the rudder plot (fig. 6.9a), we observe that large initial jumps occur with significant angle changes, which is expected but may not be realistic. The introduction of noise leads to high oscillations, although the trends in rudder angle displacement remain slowly varying. During steady state, the rudder angle has a variance of ± 5 degrees. Notably, the `nonlinear3` model performs poorly for $\chi_d = 135$ degrees. Further investigation is required to ensure that these numerical instabilities do not cause issues in deployment.

In fig. 6.10a, outliers are observed at $t > 200$ in time but not in the rudder angle (`nonlinear4` and `linear4`). The mean solver time is higher than the ideal scenario but remains under the threshold, suggesting that warm start, even with noise, is beneficial.

Course keeping with `dotv`

As with `chi_d`, we observe a significant steady state error in fig. 6.8b which can be improved with higher costs in state deviation or a longer prediction horizon. The values chosen here can be attributed to the trade-off between minimizing the error in course and the associated oscillations in rudder motion. Interestingly, this cost function demonstrates slightly faster convergence for all three models compared to `chi_d`.

Moving to fig. 6.9b, a wider range of rudder angle values are observed, indicating greater variability. In terms of oscillations in steady state, `dotv` exhibits slightly higher level of oscillations compared to `chi_d`.

In the time plot in fig. 6.10b, there is a consistent trend of higher solve time across all models for `dotv`. However, these solve times remain under the defined threshold except for a few outliers. It is worth mentioning that these outliers are again observed at $t > 200$. Additionally, it is observed that the solve time increases as the model fidelity increases.

Overall, the `dotv` cost function for `linear4` and `nonlinear3` models has the best performance, as shown in fig. 6.6. Computation time for `chi_d` with `linear4` model is the lowest with `linear4 dotv` close second. All proposed solvers are robust under limits and withstand noise and disturbances. Mean solver time remains under the threshold implying that online implementation under normal circumstances is feasible. While there are areas that require further investigation and improvements, the course-keeping performance is generally satisfactory and aligns with expectations.

7.1.3 Adverse Conditions with Soft Constraints

The primary objective of this test is to evaluate the controller's performance as the vessel's SOG approaches zero. This discussion corresponds to the plots depicted in figs. 6.13, 6.14, 6.15, 6.16, 6.17, 6.18 and 6.19.

The path plots in fig. 6.14 are quite telling as we can clearly see large oscillations in path for `chi_d` with `nonlinear4` and `linear4` models. `dotv` on the other hand produces paths with large offset but stable straight lines. We also observe that when the vehicle travels with the currents, larger SOG results in longer distances traveled by the vehicle and vice versa.

Course keeping with `chi_d`

The results of the analysis appear challenging to interpret, primarily due to two reasons (1) fluctuations in the heading angle result in corresponding fluctuations in the course angle, and (2) as the surge goes below zero, there is a "switch" in the course angle due to the negative value of the arctan function. Both of these factors are undesirable as they prevent the achievement of a stable course and may subject the vehicle to increased physical strain, particularly under adverse conditions. This issue affects both the `nonlinear4` and `linear4` models. On the other hand, the `nonlinear3` model manages to maintain a stable course. However, due to the poor accuracy of the speed prediction model at lower speeds, it is difficult to conclusively determine whether `nonlinear3` is a suitable choice.

Another significant finding is the presence of increased oscillations and a large steady-state error, leading to potential angle wrap issues as shown in the fig. 6.15a. Furthermore, the solver success rate for `nonlinear4` is approximately 69%, which raises concerns about the robustness of the controller. It is likely that the failure of the solver is due to the arctan function becoming undefined as the surge approaches zero. Despite this, the solver's failure is indicated by the message `Maximum_Iterations_Exceeded`, suggesting that further iterations may improve the convergence of the solver.

In fig. 6.15a, there are prominent and unstable oscillations in the rudder when the vehicle moves against the disturbances, which adversely affects the stability of the vehicle. However, as the vehicle moves alongside the currents, the performance gradually improves. The large oscillations indicate that the solver employed may not be suitable for this particular scenario.

Among the models examined, the `nonlinear3` model demonstrates particularly promising results. However, it should be acknowledged that the accuracy of the prediction model may pose limitations on the real-world applicability of the observed performance. Therefore, further assessment and validation are necessary to ascertain whether similar performance can be achieved in practical situations.

In fig. 6.17a, we see that when the motion is against the currents, the time to solve for the `nonlinear4` model exceeds the set threshold. This indicates that online implementation may not be feasible in this particular scenario using the `nonlinear4` model. In contrast, both the `linear4` and `nonlinear3` models demonstrate performance within the desired limits, suggesting their suitability for online implementation.

Course keeping with dotv

The dotv cost function produces a stable course but with a large steady-state error. Oscillations are seen in the course but the heading remains smooth, indicating that while the surge drops below zero, the vehicle does not experience strong changes in motion. The large steady state error further indicates that the controller is able to operate at the boundary of the inadmissible zone as the disturbances increase.

In fig. 6.16b, oscillations increase compared to the fig. 6.9b and are much less pronounced than its counterpart. Further, nonlinear models perform much better than linear models validating that the disturbances increase the nonlinearity present in the dynamics making the use of NMPC controller more relevant.

The solver time for nonlinear4 model exceeds the threshold at various instants, however, the mean solver time remains under the threshold. The performance can further be improved by increasing the discretization period (T_s), limiting the solver rate and/or using suboptimal NMPC when necessary. From fig. 6.13, we note that RMSE of dotv is higher than chi_d however, the former is still preferable to the latter because of high course and rudder oscillations, low solver success rate and high solver time of chi_d objective function.

The dotv cost function, implemented with both the nonlinear4 and linear4 models, emerges as our preferred choice. It consistently demonstrates strong performance in both nominal and adverse operating conditions. The linear model is always able to converge to the solution within the acceptable threshold whereas the maximum time taken by the nonlinear model only occasionally exceeds the limit. Despite this difference, the comparable RMSE performance suggests that the use of nonlinear dynamics is primarily necessary for handling adverse conditions, simplifying the problem during nominal operating conditions. Additionally, the solver displays robustness, with a success rate exceeding 99%.

7.2 HIL Simulation

The use of HIL simulations presents an opportunity to evaluate the proposed controller by integrating it into the existing software stack onboard the vehicle. Our objective is to (1) test the interoperability of the controller with the existing software system; (2) evaluate the real-time performance of the controller; and (3) analyze the course-keeping performance of the controller on an embedded platform. This discussion corresponds to the plots depicted in figs. 6.20, 6.21 and 6.22.

7.2.1 Nominal Conditions with Noisy Sensors

The integration of the controller with the DUNE and Neptus systems has been successfully achieved, allowing for seamless communication and control. In the real-time implementation, the controller demonstrates stable performance, converging to the desired course references. The presence of oscillations in the course can be attributed to sensor noise, which affects the accuracy of the controller's response. While some steady-state error is observed, it is consistent with the results obtained from MIL simulations. In cases where outliers occur, the use of sub-optimal or previously optimal trajectories was considered. It is anticipated that performance can be further enhanced through fine-tuning of the controller's parameters.

The central question of this research was to design and develop a course controller that is robust to noise, disturbances, and singularities. This is required as a PID-based controller such as the one explored in [9] and [17] often fail to generate a stable and controlled behavior as the SOG approaches zero. The NMPC controller, on the other hand, is able to optimize the system behavior and find the course angle that minimizes deviation from the desired course while still generating a stable course over ground, as evident in Fig. 6.15b.

The findings and conclusions of this study contribute to the broader understanding of the research problem and highlight the importance of the controller's performance in real-world scenarios. The obtained results demonstrate the robustness of the controller to both noise and disturbances, which is crucial for practical implementation. By successfully avoiding regions of singularities and optimizing the stability of the course angle being followed, the controller proves its effectiveness in challenging conditions. Although there may be a larger steady-state error, the risk of losing controllability is low, ensuring safe operation. Furthermore, the controller exhibits better performance than pid under nominal disturbances and maintains stability even under adverse conditions. Despite the higher computational cost associated with the IPOPT solver, its robustness proves to be advantageous. As the solver performs well within our benchmark, these findings provide a basis for proceeding with field trials to validate the controller's performance in real-world environments.

Nevertheless, it is important to take the findings of this study in context. During the development process, we noticed that tuning is very important as poor tuning can not only result in poor performance but also lead to numerical instability. Further, even with good tuning, establishing an upper limit is important as edge cases may lead to unexpected output. The different solvers shown in the results have different levels of success under different conditions. For example, the `nonlinear3` model with `dotv` cost function performs the best in adverse conditions but has the worst RMSE under nominal conditions. This indicates that choosing the best solver across all conditions might not be straightforward.

The guidance model explored in this study is a course-keeping behavior where we evaluated the controller's rise time and overshoot to assess its performance. However, in field trials, we expect to deploy the vehicle with a way-point reference model. While a good better course-keeping behavior should translate to good course-following behavior, it should be kept in mind that tuning parameters might differ.

The HIL implementation of the controller sufficiently proves the feasibility of the online NMPC algorithm in nominal conditions however, it is beneficial to continue to explore alternatives that may yield faster solver time than IPOPT. Hence, successful implementation of SQP-based solver is desirable, along with real-world field trials.

7.3 Future Work

By analyzing the nominal course plot in fig. 6.8, we observe a consistent steady-state error. To address this issue, we identified two potential solutions: (1) increasing the cost on the state and (2) extending the time prediction horizon. However, both approaches have their drawbacks, as the former results in increased rudder oscillations and the latter leads to longer solver computation times. An alternative and more elegant approach to mitigate this error might be to introduce integral action within the NMPC framework, as discussed in [64]. By incorporating integral action, we aim to minimize rudder oscillations while improving the overall performance of the controller.

The surge dynamics of the vehicle utilized in this study rely on a prediction model trained using a large dataset obtained at sea. While this approach enables the emulation of dynamics, its accuracy is limited, particularly when extrapolating beyond the range of the available data. Various factors, such as limited training data at the extremes and local geographical variations, contribute to this limitation. To enhance the controller's performance in this context, we propose employing an extremum-seeking guidance model that aims to minimize the steady-state error by identifying the boundary of the admissible zone closest to the desired course angle. Additionally, integrating Nonlinear Model Horizon Estimation algorithms into the system can facilitate online estimation of the noise in the speed model, further improving the controller's performance in scenarios involving uncertain or unknown surge dynamics.

These enhancements have the potential to enhance the robustness and accuracy of the controller in real-world operating conditions and are form topics of further research.

Chapter 8

Conclusion

In this project, we successfully designed and implemented a Nonlinear Model Predictive Control based course controller for the AutoNaut vehicle. Throughout the work, we developed three models with varying levels of fidelity and incorporated two different cost functions to achieve our objective.

The controllers were tested under different conditions categorized as ideal, nominal, and adverse scenarios. Under ideal conditions, all three models demonstrated excellent performance, closely following the reference path with zero steady-state error. The introduction of noise and disturbances in the nominal conditions allowed us to evaluate the expected performance of the controller and test its robustness in a realistic setting. The results also highlighted the shortcomings of the developed controller and pointed us toward potential research directions aimed at enhancing performance. Finally, during adverse conditions characterized by very low SOG, the controllers exhibited pronounced oscillations and higher solver time. Among the two proposed cost functions, it was evident that $\text{dot}v$ yielded trajectories with fewer oscillations and a more stable course. While all controllers demonstrated effectiveness in minimizing errors in the course angle, we identified two controllers that show promising potential for online implementation and resilience.

Moving to hardware implementation, we encountered challenges in achieving successful online implementation using the SQP-based solver software *acados*. Our analysis highlighted potential errors in the implementation process, which were thoroughly examined and discussed. As an alternative, we chose the more robust but computationally expensive solver IPOPT. Our testing showed that the solver produced trajectories on an embedded platform for nominal operating conditions under acceptable time. While the work was sufficiently validated through MIL and HIL simulations, further validation through in-water tests remains.

Bibliography

- [1] R. Costanza, 'The ecological, economic, and social importance of the oceans,' *Ecological Economics*, vol. 31, pp. 199–213, 2 Nov. 1999, ISSN: 0921-8009. DOI: 10.1016/S0921-8009(99)00079-8.
- [2] 'Geostationary satellite-based observations for ocean applications,' *Current Science*, vol. 117, pp. 506–515, 3 Aug. 2019, ISSN: 00113891. DOI: 10.18520/CS/V117/I3/506-515.
- [3] 'Integrating autonomous underwater vessels, surface vessels and aircraft as persistent surveillance components of ocean observing studies,' *2012 IEEE/OES Autonomous Underwater Vehicles, AUV 2012*, 2012. DOI: 10.1109/AUV.2012.6380734.
- [4] 'Ship-based contributions to global ocean, weather, and climate observing systems,' *Frontiers in Marine Science*, vol. 6, p. 434, JUL Aug. 2019, ISSN: 22967745. DOI: 10.3389/FMARS.2019.00434/BIBTEX.
- [5] 'Challenges and future trends in marine robotics,' *Annual Reviews in Control*, vol. 46, pp. 350–368, 2018, ISSN: 1367-5788. DOI: <https://doi.org/10.1016/j.arcontrol.2018.10.002>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1367578818300038>.
- [6] L. Camus, H. Andrade, A. S. Aniceto, M. Aune, K. Bandara, S. L. Basedow, K. H. Christensen, J. Cook, M. Daase, K. Dunlop, S. Falk-Petersen, P. Fietzek, G. Fonnes, P. Ghaffari, G. Gramvik, I. Graves, D. Hayes, T. Langeland, H. Lura, T. Kristiansen, O. A. Nøst, D. Peddie, J. Pederick, G. Pedersen, A. K. Sperrevik, K. Sørensen, L. Tassara, S. Tjøstheim, V. Tverberg and S. Dahle, 'Autonomous surface and underwater vehicles as effective ecosystem monitoring and research platforms in the arctic—the glider project,' *Sensors*, vol. 21, no. 20, 2021, ISSN: 1424-8220. DOI: 10.3390/s21206752. [Online]. Available: <https://www.mdpi.com/1424-8220/21/20/6752>.
- [7] P. Johnston and M. Poole, 'Marine surveillance capabilities of the autonomous wave-propelled unmanned surface vessel (usv),' in *OCEANS 2017 - Aberdeen*, 2017, pp. 1–46. DOI: 10.1109/OCEANSE.2017.8084782.
- [8] A. Dallolio, B. Agdal, A. Zolich, J. A. Alfredsen and T. A. Johansen, 'Long-endurance green energy autonomous surface vehicle control architecture,' 2019. DOI: 10.23919/OCEANS40490.2019.8962768.

- [9] A. Dallolio, H. Øveraas, J. A. Alfredsen, T. Fossen and T. Johansen, 'Design and validation of a course control system for a wave-propelled unmanned surface vehicle,' *Field Robotics*, vol. 2, pp. 748–773, Mar. 2022. DOI: 10.55417/fr.2022025.
- [10] J. Rawlings, 'Tutorial: Model predictive control technology,' in *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, vol. 1, 1999, 662–676 vol.1. DOI: 10.1109/ACC.1999.782911.
- [11] M. Fink, *Implementation of linear model predictive control – tutorial*, 2021. DOI: 10.48550/ARXIV.2109.11986. [Online]. Available: <https://arxiv.org/abs/2109.11986>.
- [12] A. () Bemporad and M. Morari, 'Robust model predictive control: A survey,' *Robustness in identification and control*, pp. 207–226, Oct. 1999. DOI: 10.1007/BFB0109870. [Online]. Available: <https://link.springer.com/chapter/10.1007/BFb0109870>.
- [13] M. Morari and J. H. Lee, 'Model predictive control: Past, present and future,' *Computers Chemical Engineering*, vol. 23, pp. 667–682, 4-5 May 1999, ISSN: 0098-1354. DOI: 10.1016/S0098-1354(98)00301-9.
- [14] L. Grüne and J. Pannek, 'Nonlinear model predictive control,' 2011. DOI: 10.1007/978-0-85729-501-9. [Online]. Available: <http://link.springer.com/10.1007/978-0-85729-501-9>.
- [15] T. A. Johansen, 'Chapter 1 introduction to nonlinear model predictive control and moving horizon estimation,' 2011.
- [16] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings and M. Diehl, 'CasADi – A software framework for nonlinear optimization and optimal control,' *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019. DOI: 10.1007/s12532-018-0139-4.
- [17] A. Dallolio, H. Øveraas and T. Johansen, 'Gain-scheduled steering control for a wave-propelled unmanned surface vehicle,' *Ocean Engineering*, vol. 257, p. 111 618, Aug. 2022. DOI: 10.1016/j.oceaneng.2022.111618.
- [18] 'Master Mariner - AutoNaut's wave-propelled USV,' *U. S. Technology*, Mar. 2017.
- [19] E. Bøckmann and S. Steen, 'Experiments with actively pitch-controlled and spring-loaded oscillating foils,' *Applied Ocean Research*, vol. 48, pp. 227–235, 2014, ISSN: 0141-1187. DOI: <https://doi.org/10.1016/j.apor.2014.09.004>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0141118714000923>.
- [20] D. J. Leith and W. E. Leithead, 'Survey of gain-scheduling analysis and design,' *International Journal of Control*, vol. 73, no. 11, pp. 1001–1025, 2000. DOI: 10.1080/002071700411304. eprint: <https://doi.org/10.1080/002071700411304>. [Online]. Available: <https://doi.org/10.1080/002071700411304>.

- [21] P Ngo, J. Das, J. Ogle, J. Thomas, W. Anderson and R. Smith, 'Predicting the speed of a wave glider autonomous surface vehicle from wave model data,' *IEEE International Conference on Intelligent Robots and Systems*, pp. 2250–2256, Oct. 2014. DOI: 10.1109/IR05.2014.6942866.
- [22] R. Smith, J. Das, G. Hine, W. Anderson and G. Sukhatme, 'Predicting wave glider speed from environmental measurements,' Sep. 2011. DOI: 10.23919/OCEANS.2011.6106989.
- [23] P Ngo, W. Al-Sabban, J. Thomas, W. Anderson, J. Das and R. Smith, 'An analysis of regression models for predicting the speed of a wave glider autonomous surface vehicle,' *Australasian Conference on Robotics and Automation, ACRA*, Jan. 2013.
- [24] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. 2021. DOI: 10.1002/9781119575016.
- [25] H. Øveraas, A. Heggernes, A. Dallolio, T. H. Bryne and T. Arne Johansen, 'Predicting the speed of a wave-propelled autonomous surface vehicle using metocean forecasts,' in *OCEANS 2022 - Chennai, 2022*, pp. 1–6. DOI: 10.1109/OCEANSCennai45887.2022.9775485.
- [26] C. E. Rohrs, J. L. Melsa and D. G. Schultz, *Linear Control Systems*. 1993.
- [27] M. G. Forbes, R. S. Patwardhan, H. Hamadah and R. B. Gopaluni, 'Model predictive control in industry: Challenges and opportunities,' *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 531–538, 2015, 9th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2015, ISSN: 2405-8963. DOI: <https://doi.org/10.1016/j.ifacol.2015.09.022>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2405896315011039>.
- [28] M. Arnold and G. Andersson, 'Model predictive control of energy storage including uncertain forecasts,' Jan. 2011.
- [29] N. Scianca, D. D. Simone, L. Lanari and G. Oriolo, 'MPC for humanoid gait generation: Stability and feasibility,' *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1171–1188, Aug. 2020. DOI: 10.1109/tro.2019.2958483. [Online]. Available: <https://doi.org/10.1109%5C%2Ftro.2019.2958483>.
- [30] P Ru and K. Subbarao, 'Nonlinear model predictive control for unmanned aerial vehicles,' *Aerospace*, vol. 4, p. 31, Jun. 2017. DOI: 10.3390/aerospace4020031.
- [31] G. Sanchez, M. Murillo, L. Genzelis, N. Deniz and L. Giovanini, 'Mpc for nonlinear systems: A comparative review of discretization methods,' Sep. 2017, pp. 1–6. DOI: 10.23919/RPIC.2017.8214333.
- [32] A. Rao, 'A survey of numerical methods for optimal control,' *Advances in the Astronautical Sciences*, vol. 135, Jan. 2010.

- [33] H. Bock and K. Plitt, 'A multiple shooting algorithm for direct solution of optimal control problems*,' *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984, 9th IFAC World Congress: A Bridge Between Control Science and Technology, Budapest, Hungary, 2-6 July 1984, ISSN: 1474-6670. DOI: [https://doi.org/10.1016/S1474-6670\(17\)61205-9](https://doi.org/10.1016/S1474-6670(17)61205-9). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1474667017612059>.
- [34] F. Clarke, 'Necessary conditions in optimal control and in the calculus of variations,' 2007.
- [35] M. Athans and P. Falb, *Optimal Control: An Introduction to the Theory and Its Applications* (Dover Books on Engineering). Dover Publications, 2007, ISBN: 9780486453286. [Online]. Available: <https://books.google.no/books?id=XJJDSZ2HEEC>.
- [36] J. H. Lee, 'Model predictive control and dynamic programming,' in *2011 11th International Conference on Control, Automation and Systems*, 2011, pp. 1807–1809.
- [37] D. Mayne, J. Rawlings, C. Rao and P. Scokaert, 'Constrained model predictive control: Stability and optimality,' *Automatica*, vol. 36, no. 6, pp. 789–814, 2000, ISSN: 0005-1098. DOI: [https://doi.org/10.1016/S0005-1098\(99\)00214-9](https://doi.org/10.1016/S0005-1098(99)00214-9). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0005109899002149>.
- [38] S. Dughman and J. Rossiter, 'A survey of guaranteeing feasibility and stability in mpc during target changes,' *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 813–818, 2015, 9th IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2015, ISSN: 2405-8963. DOI: <https://doi.org/10.1016/j.ifacol.2015.09.069>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2405896315011507>.
- [39] A. Wächter and L. Biegler, 'On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,' *Mathematical programming*, vol. 106, pp. 25–57, Mar. 2006. DOI: [10.1007/s10107-004-0559-y](https://doi.org/10.1007/s10107-004-0559-y).
- [40] F. E. Curtis, O. Schenk and A. Wächter, 'An interior-point algorithm for large-scale nonlinear optimization with inexact step computations,' *SIAM Journal on Scientific Computing*, vol. 32, pp. 3447–3475, Jan. 2010. DOI: [10.1137/090747634](https://doi.org/10.1137/090747634).
- [41] L. Biegler, 'On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming,' *Mathematical Programming*, vol. 106, Jan. 2004.
- [42] A. Wächter and L. Biegler, 'Line search filter methods for nonlinear programming: Motivation and global convergence,' *SIAM Journal on Optimization*, vol. 16, pp. 1–31, Jan. 2005. DOI: [10.1137/S1052623403426556](https://doi.org/10.1137/S1052623403426556).

- [43] B. Aleksandrov, C. Acad, B. Rumenin, C. Magele, Stoyanov, B. Sotirova, Ritchie, Toepfer, H. Brauer, M. Hristov, Repetto, B. Antchev, B. Mihailov, B. Romansky, B. Vasilev, J. Tanaka, V. Valchev, V. Shelyagin, U. Acad and A. Stoyanova, 'Review of hardware-in-the-loop -a hundred years progress in the pseudo-real testing,' vol. 54, pp. 70–84, Dec. 2019.
- [44] *Software engineering | function oriented design - javatpoint*, www.javatpoint.com. [Online]. Available: <https://www.javatpoint.com/software-engineering-function-oriented-design>.
- [45] *Course - Guidance, Navigation and Control of Vehicles - TTK4190 - NTNU*. [Online]. Available: <https://www.ntnu.edu/studies/courses/TTK4190#tab=omEmnet>.
- [46] J. Qin and T. Badgwell, 'A survey of industrial model predictive control technology,' *Control engineering practice*, vol. 11, pp. 733–764, Jul. 2003. DOI: 10.1016/S0967-0661(02)00186-7.
- [47] D. Mayne, J. Rawlings, C. Rao and P. Sokaert, 'Constrained model predictive control: Stability and optimality,' *Automatica*, vol. 36, pp. 789–814, Jun. 2000. DOI: 10.1016/S0005-1098(99)00214-9.
- [48] M. Diehl, H. Bock, J. Schlöder, R. Findeisen, Z. Nagy and F. Allgöwer, 'Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations,' *Journal of Process Control*, vol. 12, pp. 577–585, Jun. 2002. DOI: 10.1016/S0959-1524(01)00023-3.
- [49] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen and M. Diehl, *Acados: A modular open-source framework for fast embedded optimal control*, 2020. arXiv: 1910.13753 [math.OC].
- [50] I. Collado-Gonzalez, A. Gonzalez-Garcia, C. Sotelo, D. Sotelo and H. Castañeda, 'A real-time nmpc guidance law and robust control for an autonomous surface vehicle,' *IFAC-PapersOnLine*, vol. 54, pp. 252–257, Nov. 2021. DOI: 10.1016/j.ifacol.2021.10.101.
- [51] A. Gonzalez-Garcia, I. Collado-Gonzalez, R. Cuan Urquizo, C. Sotelo, D. Sotelo and H. Castañeda, 'Path-following and lidar-based obstacle avoidance via nmpc for an autonomous surface vehicle,' *Ocean Engineering*, vol. 266, p. 112900, Dec. 2022. DOI: 10.1016/j.oceaneng.2022.112900.
- [52] M. Saljanin, S. Müller, J. Kiebler, J. Neubeck and A. Wagner, 'A model predictive control approach for highly automated vehicles in urban environments,' *Automotive and Engine Technology*, vol. 7, Jun. 2022. DOI: 10.1007/s41104-022-00103-x.
- [53] G. Frison, D. Kouzoupis, A. Zanelli and M. Diehl, 'Blasfeo: Basic linear algebra subroutines for embedded optimization,' *ACM Transactions on Mathematical Software*, vol. 44, Apr. 2017. DOI: 10.1145/3210754.

- [54] G. Frison and M. Diehl, *Hpipm: A high-performance quadratic programming framework for model predictive control*, 2020. arXiv: 2003.02547 [math.OC].
- [55] J. Ferreau, C. Kirches, A. Potschka, H. Bock and M. Diehl, 'Qpoases: A parametric active-set algorithm for quadratic programming,' *Mathematical Programming Computation*, vol. 6, Dec. 2014. DOI: 10.1007/s12532-014-0071-1.
- [56] J. Pinto, P. Calado, J. Braga, P. Dias, R. Martins, E. Marques and J. Sousa, 'Implementation of a control architecture for networked vehicle systems,' *Proceedings of the IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles*, vol. 3, Jan. 2012.
- [57] P. Dias, R. Gomes, J. Pinto, G. Gonçalves, J. Sousa and F. Pereira, 'Mission planning and specification in the neptus framework.,' Jan. 2006, pp. 3220–3225. DOI: 10.1109/ROBOT.2006.1642192.
- [58] S. D. Sæter, 'Colregs compliant collision avoidance system for a wave and solar powered usv,' 2018.
- [59] R. Martins, P. S. Dias, E. R. B. Marques, J. Pinto, J. B. Sousa and F. L. Pereira, 'Imc: A communication protocol for networked vehicles and sensors,' in *OCEANS 2009-EUROPE*, 2009, pp. 1–6. DOI: 10.1109/OCEANSE.2009.5278245.
- [60] A. S. Ferreira, J. Pinto, P. Dias and J. B. de Sousa, 'The lsts software tool-chain for persistent maritime operations applied through vehicular ad-hoc networks,' in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, pp. 609–616. DOI: 10.1109/ICUAS.2017.7991471.
- [61] B. O. Agdal, 'Design and implementation of control system for green unmanned surface vehicle,' 2018.
- [62] *Ts-7800-v2 industrial single board computer 1.3 ghz dual core arm-based cpu*, www.embeddedts.com. [Online]. Available: <https://www.embeddedts.com/products/TS-7800-V2> (visited on 03/06/2023).
- [63] H. Ghael, 'A review paper on raspberry pi and its applications,' Jan. 2020. DOI: 10.35629/5252-0212225227.
- [64] D. Ruscio, 'Model predictive control with integral action: A simple mpc algorithm,' *Modeling, Identification and Control: A Norwegian Research Bulletin*, vol. 34, pp. 119–129, Jan. 2013. DOI: 10.4173/mic.2013.3.2.
- [65] T. Perez and T. I. Fossen, 'A Matlab Toolbox for Parametric Identification of Radiation-Force Models of Ships and Offshore Structures,' *Modeling, Identification and Control*, vol. 30, no. 1, pp. 1–15, 2009. DOI: 10.4173/mic.2009.1.1.
- [66] a. aalok atharva, *Professional plots*, MATLAB Central File Exchange, 2023. [Online]. Available: <https://www.mathworks.com/matlabcentral/fileexchange/100766-professional-plots> (visited on 2023).

- [67] K. Worthmann, M. W. Mehrez, M. Zanon, G. K. I. Mann, R. G. Gosine and M. Diehl, 'Model predictive control of nonholonomic mobile robots without stabilizing constraints and costs,' *IEEE Transactions on Control Systems Technology*, vol. 24, no. 4, pp. 1394–1406, 2016. DOI: 10.1109/TCST.2015.2488589.
- [68] Y. Altman, *Export_fig*, Github, 2023. [Online]. Available: https://github.com/altmany/export_fig/releases/tag/v3.39 (visited on 2023).

Appendix A

AutoNaut USV model parameters

The system parameters are listed in tbl. A.1.

Table A.1: Vehicle Parameters

Parameter	Symbol	Value (Unit)
Mass	m	280 (Kg)
Length	L	4.8 (m)
Beam	B	0.78 (m)
Draft	T	0.24 (m)
Center of Gravity	CG	$[0 \ 0 \ 0]$ (m)
Moment of Inertia	I_z	551.7960 (Kg-m ²)

The hydrodynamic parameters used to calculate the added-mass matrices of the system are listed in tbl. A.2.

Table A.2: Hydrodynamic Parameters

Parameter	Value (Unit)
$X_{\dot{u}}$	6.72
$Y_{\dot{v}}$	109.11
$Y_{\dot{r}}$	74.978
$N_{\dot{v}}$	109.11
$N_{\dot{r}}$	546.85

The linear damping matrix is given in Eq. A.1.

$$\mathbf{D} = - \begin{bmatrix} 286.72 & 0 & 0 \\ 0 & 194.56 & 0 \\ 0 & 0 & 1098.6 \end{bmatrix}. \quad (\text{A.1})$$

The rudder model parameters are given in the tbl. A.3.

Table A.3: Rudder Parameters

Parameter	Symbol	Value (Unit)
Height	b	0.42
Width	w	0.25
Aspect Ratio	Λ	1.68
Coefficient	C_N	1.5598
Additional Drag Coefficient	t_R	0.366
Force Factor	a_H	0.2
Interaction Coefficient	x'_H	-1.8
Lateral Force Coefficient	x_H	-0.375
Longitudinal Rudder Position Coordinate	x_R	-2.4
Block Coefficient	C_B	0.3

The wind model parameters are given in the tbl. A.4.

Table A.4: Wind Model Parameters

Parameter	Symbol	Value (Unit)
Length Overall	L_{oa}	5
Frontal Projected Area	A_{F_w}	0.168
Lateral Projected Area	A_{L_w}	1.2
Wind Coefficient for x -axis	c_x	0.5
Wind Coefficient for y -axis	c_y	0.7
Wind Coefficient for ψ -axis	c_n	0.05

The weights for the speed model are given in the Eq. A.2.

$$w = [0.11639 \quad 0.21449 \quad 0.088068 \quad -0.006355 \quad 0.093746 \quad 0.23836] \quad (\text{A.2})$$

Appendix B

Code

The MIL simulator developed in MATLAB™ is attached with this report and will be available here until 10th June 2024. Thereon, the reader may email the author at gshubham96@gmail.com for the same. The link also contains all the raw results from the simulator. The simulator makes use of several third-party toolboxes. These are listed below

- The MSS Toolbox[65]
- Professional Plots[66]
- CasADi Tutorial by [67]
- export_fig[68]

The modified software stack is attached with this report and can also be downloaded from the public github repo.

Appendix C

Additional Plots

This section contains additional plots from the result shown in chapter 6.

C.1 Results from MIL Simulation

Results under Ideal Conditions

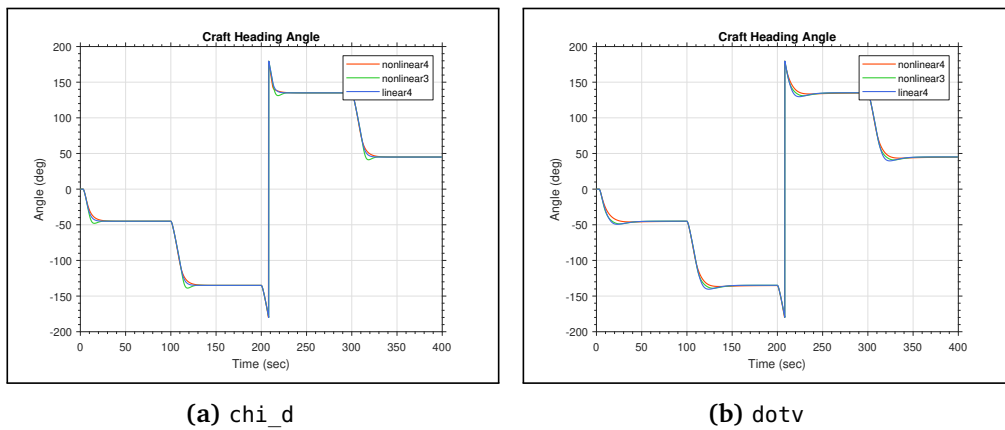


Figure C.1: Heading angle in ideal conditions

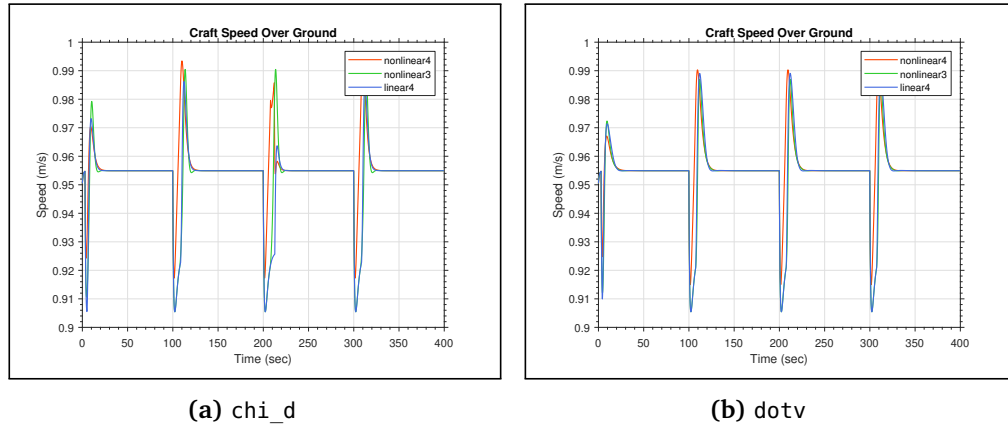


Figure C.2: SOG in ideal conditions

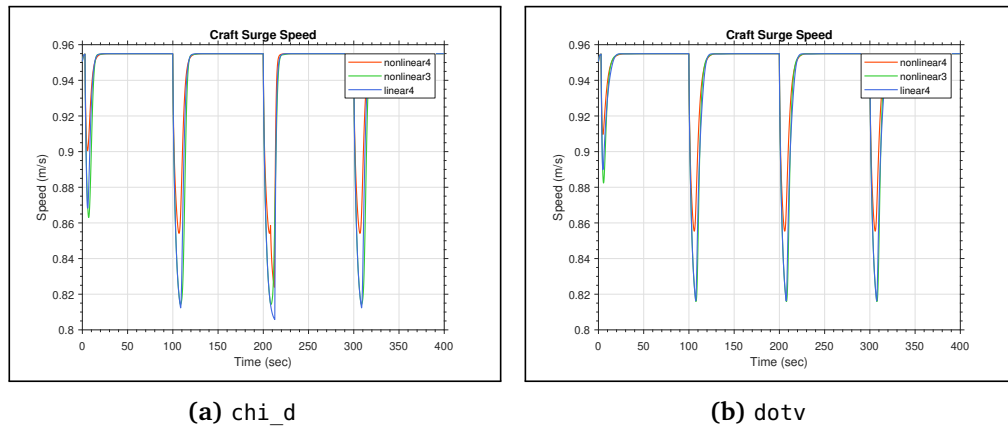


Figure C.3: Surge speed in ideal conditions

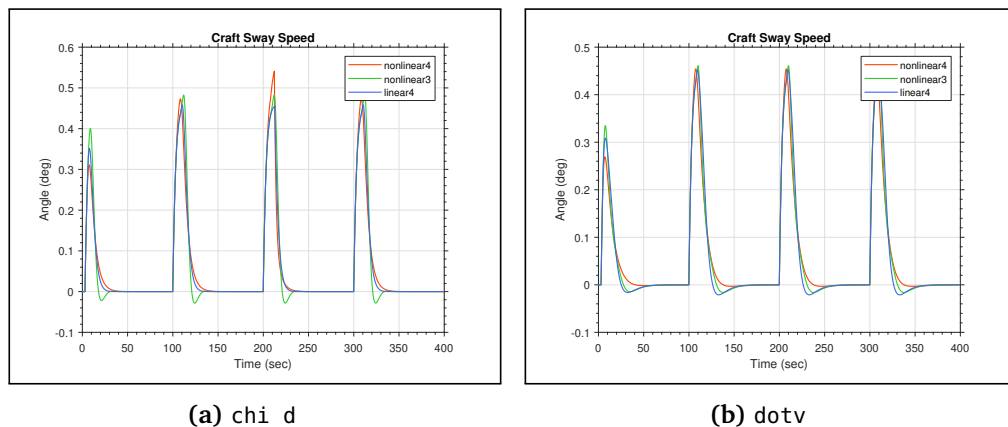


Figure C.4: Sway speed in ideal conditions

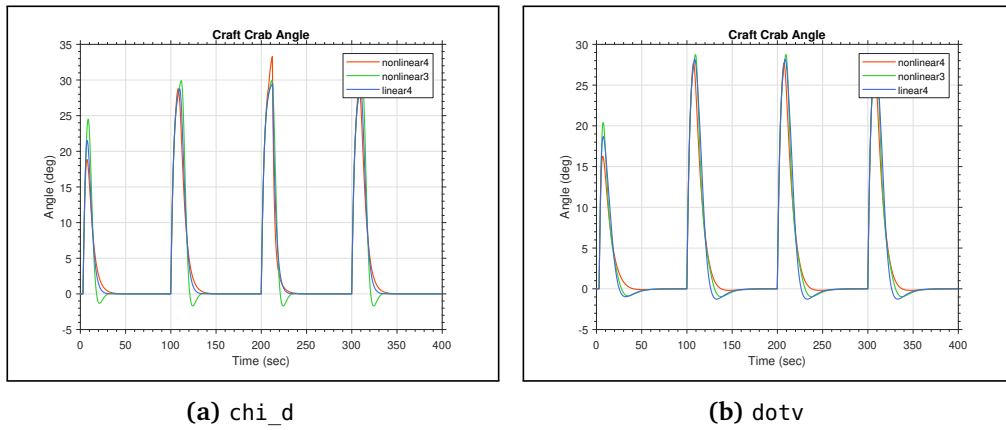


Figure C.5: Sideslip angle in ideal conditions

Results under Nominal Conditions

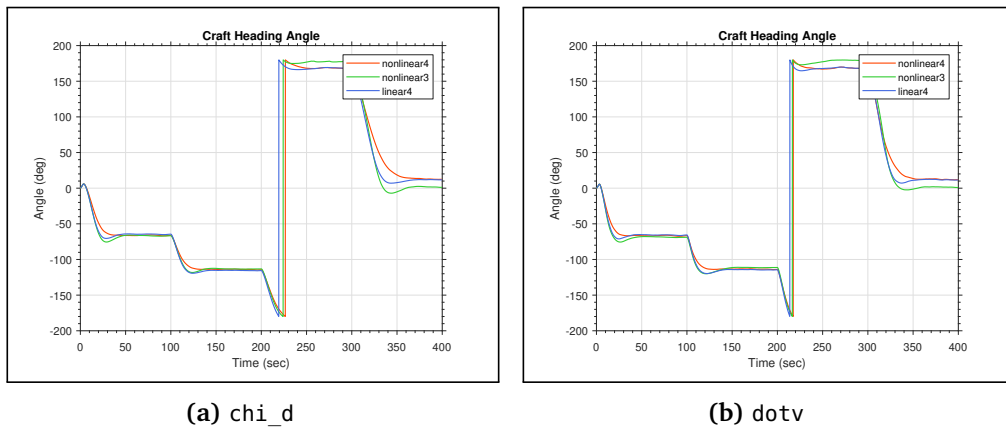


Figure C.6: Heading angle in nominal conditions

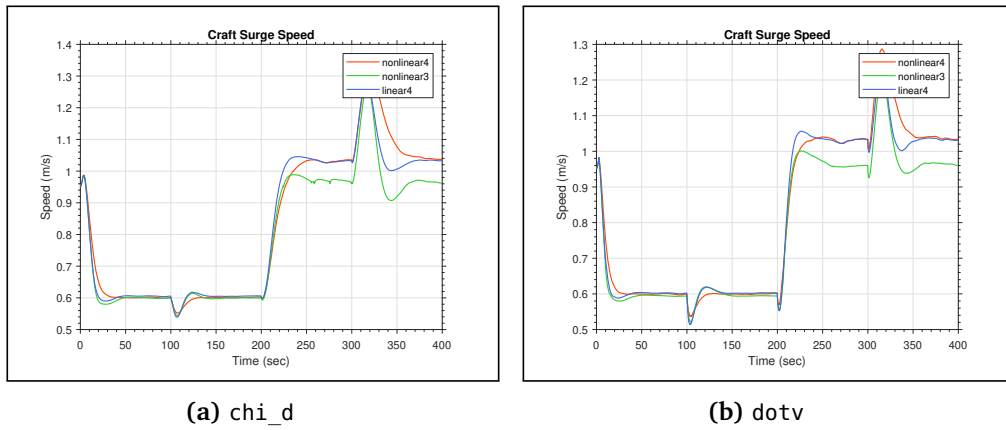


Figure C.7: Surge speed in nominal conditions

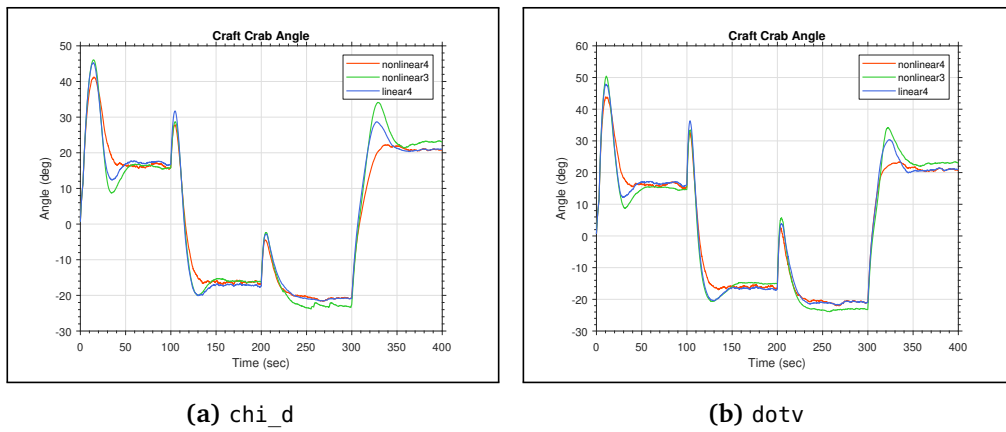


Figure C.8: Sway speed in nominal conditions

Results under Adverse Conditions

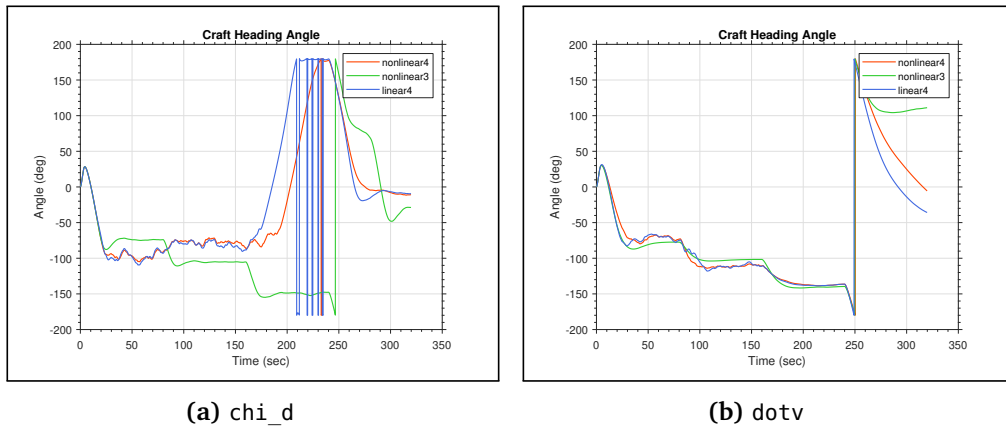


Figure C.9: Heading angle in adverse conditions

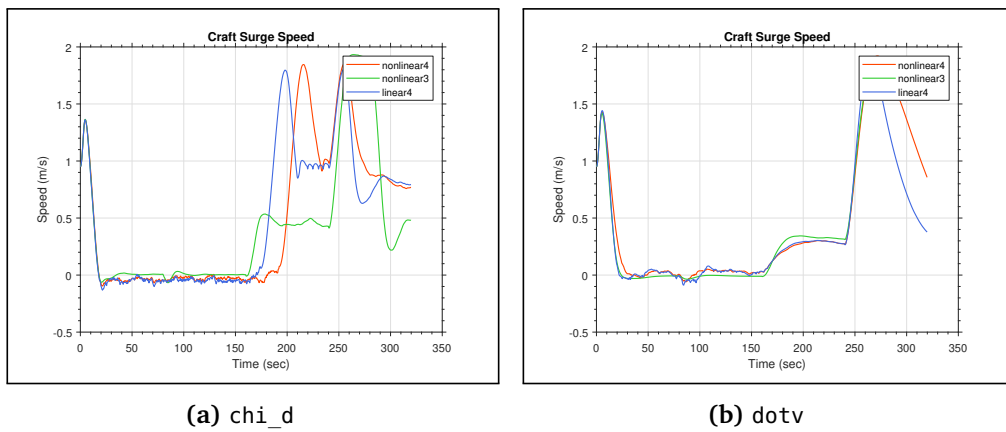


Figure C.10: Surge speed in adverse conditions

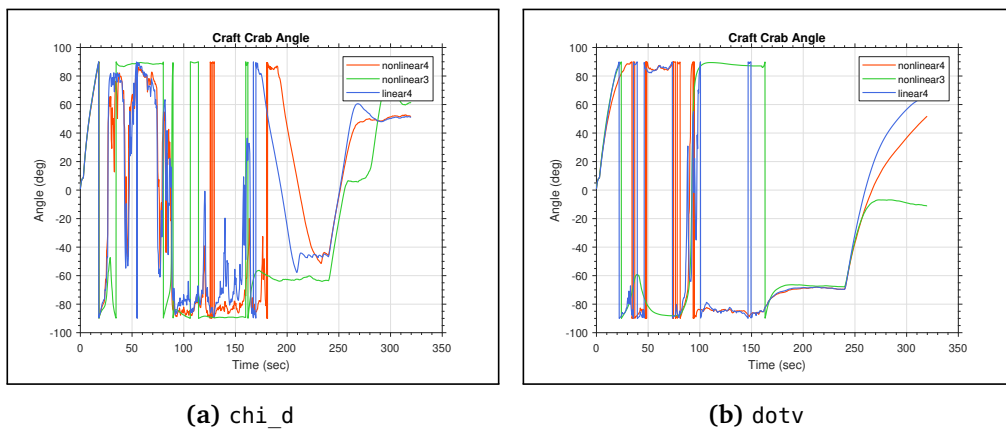


Figure C.11: Sway speed in adverse conditions

Masteravtale/hovedoppgaveavtale

Sist oppdatert 11. november 2020

Fakultet	Fakultet for ingeniørvitenskap
Institutt	Institutt for marin teknikk
Studieprogram	TMR4930 Marine Technology
Emnekode	TMR4930

Studenten	
Etternavn, fornavn	Garg, Shubham
Fødselsdato	28.03.1996
E-postadresse ved NTNU	shubhag@stud.ntnu.no

Tilknyttede ressurser	
Veileder	Asgeir Johan Sørensen
Eventuelle medveiledere	Tor Arne Johansen
Eventuelle medstudenter	

Oppgaven	
Oppstartsdato	15.01.2023
Leveringsfrist	11.06.2023
Oppgavens arbeidstittel	Design and Validation of a Course Controller for a Wave-powered Vehicle Using Model Predictive Control
Problembeskrivelse	<p>A new class of long-endurance green-energy vehicles are being developed that can be used for extended spatial-temporal studies of Oceans while having minimum environmental impact. These vehicles can play a crucial role in observing rapid environmental changes by allowing persistent and sustainable ocean monitoring. AutoNaut, developed by AutoNaut Ltd is one such vehicle being powered by the motion of the waves allowing it to operate for long duration without the need for physical human intervention. Unlike common marine platforms, the speed of the wave-powered vehicle can not be directly controlled and depends solely on the state of interaction of the vehicle and the environment. In case of AutoNaut, low controllability and maneuverability is observed when the resistive and dissipative forces offered by the environment outweigh the net propulsion forces acting on the vehicle. During this state, the vehicle can not maintain the desired course angle which can compromise the safety and efficiency of the vehicle. To ensure safe operations on both the open sea and near shores, additional constraints are introduced on the Guidance-Navigation-Control (GNC) system of the vehicle that can not be addressed by conventional control techniques. In [1], the authors analyzed the system nonlinearities when loss of controllability is observed, and proposed and validated a control design when USV's ground speed is not</p>

	<p>close to zero. This work aims to extend the control design to include cases when ground speed of the vehicle approaches zero by utilizing the mathematical knowledge of the vehicle's dynamics, the real-time state of the environment and tools of optimal control. [1] Dallolio, Alberto; Øveraas, Henning; Alfredsen, Jo Arve; Fossen, Thor Inge; Johansen, Tor Arne. (2022) Design and Validation of a Course Control System for a Wave-Propelled Unmanned Surface Vehicle. <i>Field Robotics</i>. volume 2. Academic article</p>
--	--

Risikovurdering og datahåndtering	
Skal det gjennomføres risikovurdering?	Nei
Dersom «ja», har det blitt gjennomført?	Nei
Skal det søkes om godkjenninger? (REK*, NSD**)	Nei
Skal det skrives en konfidensialitetsavtale i forbindelse med oppgaven?	Nei
Hvis «ja», har det blitt gjort?	Nei

* Regionale komiteer for medisinsk og helsefaglig forskningsetikk (<https://rekportalen.no>)

** Norsk senter for forskningsdata (<https://nsd.no/>)

Eventuelle emner som skal inngå i mastergraden
Formulation of MPC Problem Development of MPC Controller for LSTS Toolchain Validation of MPC Controller Implementation in Vehicle Robustness with Uncertain parameters Suboptimal MPC: Loss of Convergence

Retningslinjer - rettigheter og plikter

Formål

Avtale om veiledning av masteroppgaven/hovedoppgaven er en samarbeidsavtale mellom student, veileder og institutt. Avtalen regulerer veiledningsforholdet, omfang, art og ansvarsfordeling.

Studieprogrammet og arbeidet med oppgaven er regulert av Universitets- og høyskoleloven, NTNUs studieforskrift og gjeldende studieplan. Informasjon om emnet, som oppgaven inngår i, finner du i emnebeskrivelsen.

Veiledning

Studenten har ansvar for å

- Avtale veiledningstimer med veileder innenfor rammene master-/hovedoppgaveavtalen gir.
- Utarbeide framdriftsplan for arbeidet i samråd med veileder, inkludert veiledningsplan.
- Holde oversikt over antall brukte veiledningstimer sammen med veileder.
- Gi veileder nødvendig skriftlig materiale i rimelig tid før veiledning.
- Holde instituttet og veileder orientert om eventuelle forsinkelser.
- Inkludere eventuell(e) medstudent(er) i avtalen.

Veileder har ansvar for å

- Avklare forventninger om veiledningsforholdet.
- Sørge for at det søkes om eventuelle nødvendige godkjenninger (etikk, personvern hensyn).
- Gi råd om formulering og avgrensning av tema og problemstilling, slik at arbeidet er gjennomførbart innenfor normert eller avtalt studietid.
- Drøfte og vurdere hypoteser og metoder.
- Gi råd vedrørende faglitteratur, kildemateriale, datagrunnlag, dokumentasjon og eventuelt ressursbehov.
- Drøfte framstillingsform (eksempelvis disposisjon og språklig form).
- Drøfte resultater og tolkninger.
- Holde seg orientert om progresjonen i studentens arbeid i henhold til avtalt tids- og arbeidsplan, og følge opp studenten ved behov.
- Sammen med studenten holde oversikt over antall brukte veiledningstimer.

Instituttet har ansvar for å

- Sørge for at avtalen blir inngått.
- Finne og oppnevne veileder(e).
- Inngå avtale med annet institutt/ fakultet/institusjon dersom det er oppnevnt ekstern medveileder.
- I samarbeid med veileder holde oversikt over studentens framdrift, antall brukte veiledningstimer, og følge opp dersom studenten er forsinket i henhold til avtalen.
- Oppnevne ny veileder og sørge for inngåelse av ny avtale dersom:
 - Veileder blir fraværende på grunn av eksempelvis forskningstermin, sykdom, eller reiser.
 - Student eller veileder ber om å få avslutte avtalen fordi en av partene ikke følger den.
 - Andre forhold gjør at partene finner det hensiktsmessig med ny veileder.
- Gi studenten beskjed når veiledningsforholdet opphører.
- Informere veileder(e) om ansvaret for å ivareta forskningsetiske forhold, personvern hensyn og veiledningsetiske forhold.
- Ønsker student, eller veileder, å bli løst fra avtalen må det søkes til instituttet. Instituttet må i et slikt tilfelle oppnevne ny veileder.

Avtaleskjemaet skal godkjennes når retningslinjene er gjennomgått.

Godkjent av

Shubham Garg
Student

31.01.2023
Digitalt godkjent

Asgeir Johan Sørensen
Veileder

31.01.2023
Digitalt godkjent

Kristin J. Mørkve
Institutt

21.02.2023
Digitalt godkjent

Master`s Agreement / Main Thesis Agreement

Faculty	Faculty of Engineering
Institute	Department of Marine Technology
Programme Code	TMR4930 Marine Technology
Course Code	TMR4930

Personal Information	
Surname, First Name	Garg, Shubham
Date of Birth	28.03.1996
Email	shubhag@stud.ntnu.no

Supervision and Co-authors	
Supervisor	Asgeir Johan Sørensen
Co-supervisors (if applicable)	Tor Arne Johansen
Co-authors (if applicable)	

The Master`s thesis	
Starting Date	15.01.2023
Submission Deadline	11.06.2023
Thesis Working Title	Design and Validation of a Course Controller for a Wave-powered Vehicle Using Model Predictive Control
Problem Description	<p>A new class of long-endurance green-energy vehicles are being developed that can be used for extended spatial-temporal studies of Oceans while having minimum environmental impact. These vehicles can play a crucial role in observing rapid environmental changes by allowing persistent and sustainable ocean monitoring. AutoNaut, developed by AutoNaut Ltd is one such vehicle being powered by the motion of the waves allowing it to operate for long duration without the need for physical human intervention. Unlike common marine platforms, the speed of the wave-powered vehicle can not be directly controlled and depends solely on the state of interaction of the vehicle and the environment. In case of AutoNaut, low controllability and maneuverability is observed when the resistive and dissipative forces offered by the environment outweigh the net propulsion forces acting on the vehicle. During this state, the vehicle can not maintain the desired course angle which can compromise the safety and efficiency of the vehicle. To ensure safe operations on</p>

both the open sea and near shores, additional constraints are introduced on the Guidance-Navigation-Control (GNC) system of the vehicle that can not be addressed by conventional control techniques. In [1], the authors analyzed the system nonlinearities when loss of controllability is observed, and proposed and validated a control design when USV's ground speed is not close to zero. This work aims to extend the control design to include cases when ground speed of the vehicle approaches zero by utilizing the mathematical knowledge of the vehicle's dynamics, the real-time state of the environment and tools of optimal control. [1] Dallolio, Alberto; Øveraas, Henning; Alfredsen, Jo Arve; Fossen, Thor Inge; Johansen, Tor Arne. (2022) Design and Validation of a Course Control System for a Wave-Propelled Unmanned Surface Vehicle. Field Robotics. volume 2. Academic article

Risk Assessment and Data Management	
Will you conduct a Risk Assessment?	No
If “Yes”, Is the Risk Assessment Conducted?	No
Will you Apply for Data Management? (REK*, NSD**)	No
Will You Write a Confidentiality Agreement?	No
If “Yes”, Is the Confidentiality Agreement Conducted?	No

* REK -- <https://rekportalen.no/>

** Norwegian Centre for Research Data (<https://nsd.no/nsd/english/index.html>)

Topics to be included in the Master`s Degree (if applicable)
Formulation of MPC Problem Development of MPC Controller for LSTS Toolchain Validation of MPC Controller Implementation in Vehicle Robustness with Uncertain parameters Suboptimal MPC: Loss of Convergence

Guidelines – Rights and Obligations

Purpose

The Master's Agreement/ Main Thesis Agreement is an agreement between the student, supervisor, and department. The agreement regulates supervision conditions, scope, nature, and responsibilities concerning the thesis.

The study programme and the thesis are regulated by the Universities and University Colleges Act, NTNU's study regulations, and the current curriculum for the study programme.

Supervision

The student is responsible for

- Arranging the supervision within the framework provided by the agreement.
- Preparing a plan of progress in cooperation with the supervisor, including a supervision schedule.
- Keeping track of the counselling hours.
- Providing the supervisor with the necessary written material in a timely manner before the supervision.
- Keeping the institute and supervisor informed of any delays.
- Adding fellow student(s) to the agreement, if the thesis has more than one author.

The supervisor is responsible for

- Clarifying expectations and how the supervision should take place.
- Ensuring that any necessary approvals are acquired (REC, ethics, privacy).
- Advising on the demarcation of the topic and the thesis statement to ensure that the work is feasible within agreed upon time frame.
- Discussing and evaluating hypotheses and methods.
- Advising on literature, source material, data, documentation, and resource requirements.
- Discussing the layout of the thesis with the student (disposition, linguistic form, etcetera).
- Discussing the results and the interpretation of them.
- Staying informed about the work progress and assist the student if necessary.
- Together with the student, keeping track of supervision hours spent.

The institute is responsible for

- Ensuring that the agreement is entered into.
- Find and appoint supervisor(s).
- Enter into an agreement with another department / faculty / institution if there is an external co-supervisor.
- In cooperation with the supervisor, keep an overview of the student's progress, the number of supervision hours spent, and assist if the student is delayed by appointment.
- Appoint a new supervisor and arrange for a new agreement if:
 - The supervisor will be absent due to research term, illness, travel, etcetera.
 - The student or supervisor requests to terminate the agreement due to lack of adherence from either party.
 - Other circumstances where it is appropriate with a new supervisor.
- Notify the student when the agreement terminates.
- Inform supervisors about the responsibility for safeguarding ethical issues, privacy and guidance ethics
- Should the cooperation between student and supervisor become problematic, either party may apply to the department to be freed from the agreement. In such occurrence, the department must appoint a new supervisor

This Master`s agreement must be signed when the guidelines have been reviewed.

Signatures

Shubham Garg
Student

31.01.2023
Digitally approved

Asgeir Johan Sørensen
Supervisor

31.01.2023
Digitally approved

Kristin J. Mørkve
Department

21.02.2023
Digitally approved

