Trajectory-based Controller-Geometry-Codesign of Underwater Robots

Scientific Thesis for the procurance of the degree M.Sc.
from the Department of Electrical and Computer Engineering at the Technical University of Munich.

Supervised by Univ.-Prof. Dr.-Ing. Sandra Hirche
Dr.-Ing. Stefan Sosnowski
Chair of Information-Oriented Control

Submitted by B.Sc. Yongyu Chen
Schröfelhofstr. 22/0906
81375 München
0176 82559612

Submitted on Munich, 27.10.2017

Abstract

Conventionally, designing an underwater robot requires experienced experts to adapt the robot structure to a series of designing requirements: buoyancy neutral, high surge velocity, low power consumption, low construction cost and small compact volume, etc. This is usually an iterative process, since most of the robot geometric variables are tightly coupled and immediately affect the vehicle dynamics and performance of the robot. Furthermore, due to the nonlinear and coupled nature of the robot dynamics, changes in the vehicle geometric variables strongly influence the control design. In this work, we propose a novel a system-engineering approach to the controller-geometry-codesign: kinematic and dynamic requirements are derived from a set of desired trajectories the robot should perform. Together with the conventional design requirements we formulate the robot designing procedure as multi-stage iterative optimizations.

Due to the customizable robot geometric structure, we build the robot in a modular way in this work. That is, the robot constituent parts are modeled as simple shapes parametrized by several geometric decision variables. The inertia matrices, Coriolis matrices, restoring matrix and the damping matrix can be calculated from them. As a result, a parametrized vehicle dynamic model indicating the intrinsic couplings among these decision variables is derived.

Trim trajectories are chosen in this thesis since along them the vehicle moves stably. It implies that all velocities, roll and pitch angles as well as the hydrodynamic coefficients are constant. Based on these characteristics, we can linearize the dynamics of the error between the real robot' states and desired trim states uniquely. Each specified trim trajectory corresponds to a linear multi-input multi-output error dynamic system for a given robot geometry. Thus, the nonlinear robot dynamics is formulated as a linear switched system in the thesis and we utilize switched linear quadratic regulator to control the robot's moving trajectories.

Simulations for optimizing the actuator placements including six thrusters and four fins combined with two thrusters are performed, respectively. According to the simulation results, the proposed optimization algorithm is able to find a locally optimal and controllable robot geometry which can track the desired trim trajectories stably with a unconstrained switched LQR controller.

Contents

Chapter 1

Introduction and Related Work

1.1 Introduction

Two-thirds of the earth is covered by the ocean and the underwater robots can help us to explore and perform various tasks in this hazardous and uneasily attainable environment. The autonomous underwater vehicles (AUVs) have various application areas. For scientific purpose, they can be used for seafloor mapping or geological sampling. In the environmental area, people utilize them to inspect the underwater structures like pipelines, dams, etc. or to monitor the pollution. Besides, underwater robots also bring convenience to the ocean survey and undersea structures' construction and maintenance [Yuh00].
A good manoeuvring performance of underwater robots requires an appropriate robot structure adapting to the specific mission. This is a tough and time-consuming task involving many iterations of design and testing, even though for skilled expert. A simple underwater robot prototype includes CPU, the inertial navigation system, the sensory and communication system, and the power supply system (batteries). All of them are enclosed in a protecting hull. Besides, thruster and control fins are the main actuators. Their properties, numbers and locations will influence the robot's working performance heavily. Hence, a good designer should be able to find a suitable configuration of these components.
Traditionally, after the robot was built by the designer, the users will analyse the dynamics based on the current robot structure and actuator configuration. Then they will plan the trajectory and design the controller based on this dynamic model and the task requirements.
In this work, we separate the hull design and the actuator placement optimization. Designing the hull should satisfy the conventional specifications: buoyancy neutral, maximal surge velocity, maximal working time and low construction cost and a minimal volume to contain the inertial measurement unit, CPU and communication system. In terms of the actuator design we start from an innovative designing direction. That is, a set of requirements for kinematics and dynamics is directly derived from a set of trajectories the vehicle should perform. A suitable trajectory representation should be chosen and key requirements should be identified. In the design phase, an automated co-design of the optimal

kinematics and dynamics of the robot together with the synthesis of a stable controller for tracking is aimed for.

In Chapter 2, we propose a modular modeling method for the underwater robot prototype. Each component can be modeled as sample geometric shape determined by several geometric variables and then we can parametrize the robot dynamics with these variables.

From deep-sea exploration to underwater rescue, underwater vehicles require efficient path planning algorithms. An relative simple but representative trajectory is expected for designing the robot. So we use the trim trajectory since robot's motion is stable along them. Partial kinematic states (ϕ, θ), the dynamic states (linear and angular velocities) that influence the robot dynamics and therefore the hydrodynamic coefficients depending on velocities stay constant along the trim trajectory. A set of desired trim trajectories give us firstly information about the desired position for the robots. By adopting the Frenet-Serret frame, the desired orientation and the desired velocity along the trajectory will be derived in further. With the full information about the system states, we are able to formulate the error dynamics. These will be introduced in Chapter 3.

For analysis and controller design of nonlinear systems the typical approach is to linearize the nonlinear system. In this work we adopt a trajectory-based linearization way for the error dynamics usd in [SPK02a] for the following reasons. Firstly, for a specific trim trajectory segment the linearized error dynamics is unique. Hence, we can implement a number of analysing approaches for linear systems for the error dynamics. Secondly, tracking a desired trajectory is a concern for practical manoeuvring tasks. The error dynamic formulation along trim trajectories will be illustrated in Chapter 4. Moreover, task-specific trajectories for underwater robots can be planned as a combination of a set of trim trajectories, which means that we also have a set of linearized MIMO error dynamics. Thus, we design a switched LQR controller based on these linear systems. This will also be discussed in the same chapter.

The main contribution of this work is to find an optimal geometric configuration of the robot constituent parts. Based on the parametrized underwater vehicle model built in Chapter 2, we are able to identify the relationship between the geometric decision variables and each term in the dynamic model (the mass matrix, the Coriolis matrix, the damping matrix and the restoring matrix). After that, we propose two optimization algorithms which find a locally optimal hull size and a local optimum for actuator placement, respectively.

In Chapter 6, we utilize the geometry from the optimization results and calculate the corresponding optimal dynamics. Regular underwater robots are normally equipped with actuators symmetrically in pairs. These new built robots are visualized to provide us a more intuitive impression about the irregular robots. The irregularity is caused by the randomly generated initial geometric decision variables and the nonlinear optimization algorithm that is only able to find a local minimum depending on the initial values. The controllability of the current designed robot system is verified by means of checking the tracking performance along a set of trim trajectories.

Underwater dynamics are nonlinear, highly coupled, MIMO systems with six degrees of freedom (DOFs). When we change the geometric parameters of hull and actuators or the number of actuators, the geometric center, the center of buoyancy, the inertia matrix, the Coriolis matrix and the hydrodynamic damping will also change consequently. In addition, the modeling of thrusters and control fins is not accurate enough to indicate their real physical characteristics. To make the optimization solvable, we make a list of assumptions. These inaccuracies can be rectified and the optimization performance can be enhanced for future works. These limitations and the possible improvements will be discussed in the last chapter.

The complete underwater robot designing and verification processes proposed in this thesis are concluded in Figures 1.1 and 1.2.

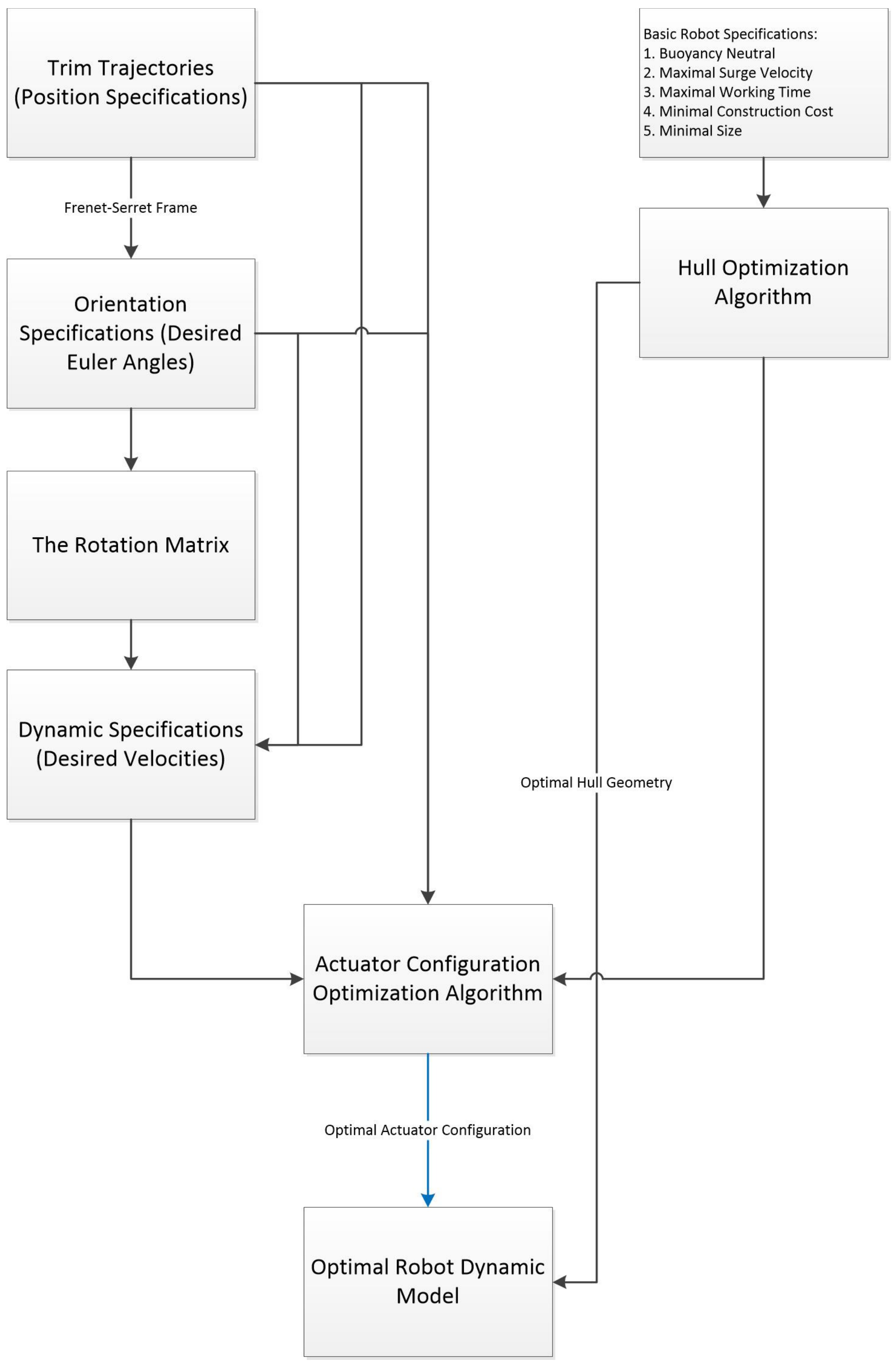

Trim Trajectories (Position Specifications)
Frenet-Serret Frame
Orientation Specifications (Desired Euler Angles)
The Rotation Matrix
Dynamic Specifications (Desired Velocities)
Basic Robot Specifications:
1. Buoyancy Neutral
2. Maximal Surge Velocity
3. Maximal Working Time
4. Minimal Construction Cost
5. Minimal Size
Hull Optimization Algorithm
Optimal Hull Geometry
Actuator Configuration Optimization Algorithm
Optimal Actuator Configuration
Optimal Robot Dynamic Model

Figure 1.1: Underwater robot geometric design procedure

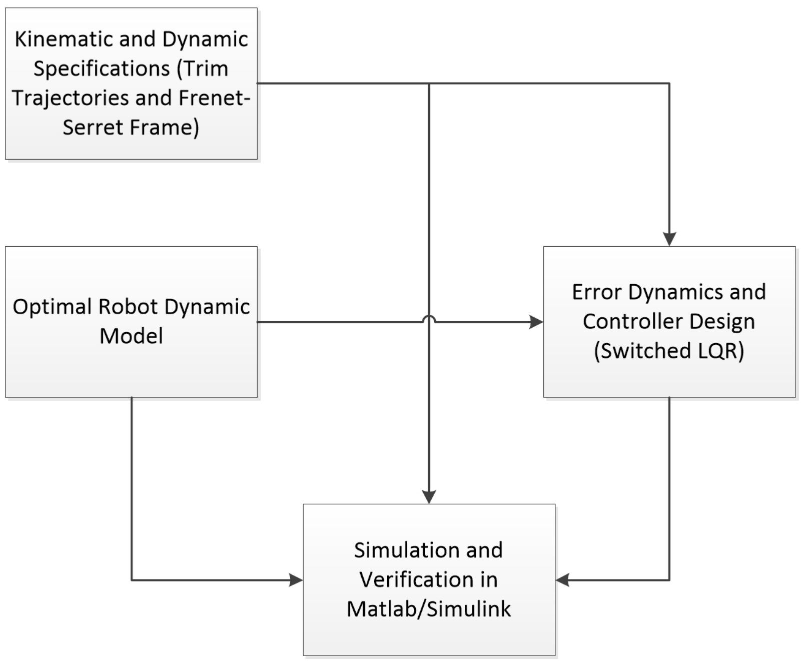

Kinematic and Dynamic Specifications (Trim Trajectories and Frenet-Serret Frame)
Optimal Robot Dynamic Model
Error Dynamics and Controller Design (Switched LQR)
Simulation and Verification in Matlab/Simulink

Figure 1.2: Verification of geometric design

1.2 Related Work

Our work draws from a number of ideas and methods from underwater robot modelling, error dynamics and controller design and computational robot design.

1.2.1 Underwater Robot Modular Dynamic Modeling

The chapter 7 of [Fos11] introduces 1 DOF heading autopilot model, 3DOF manoeuvring and dynamic positioning (DP) models, 4 DOF manoeuvring and 6 DOF coupled system for advanced 6DOF manoeuvring model. Especially for underwater robots with actuation in all DOFs, a nonlinear, strongly coupled, MIMO dynamic 6DOF model is required for the model-based controller and observer design. The detailed derivation can be found in the appendix A summarized from [Fos11, Fos94, FF95, Ant14].

Our goal is to design algorithms finding an optimal configuration for AUVs. It requires the robots to have a flexible modularized structure so that we can reconfigure the robot prototype easily. In [VVKR05], a novel modular underwater robot which can self-reconfigure by stacking and unstacking its component modules is designed. A modular dynamic modeling is desired for this flexible robot structure. With help of this method, the dynamic equation can be parametrized with geometric variables (location, size etc.) and especially the hydrodynamic coefficients can be estimated from them. The modular modeling methods are discussed in [Che07] and [Tan99].

1.2.2 Controller Design and Path Planning

Due to the requirement of full-DOF control, underwater robot control is a very challenge task. Diverse control methods can be implemented for the robot, e.g., adaptive control ([Yuh90, FS91]), sliding model control ([YS85, NN06]), feedback linearization ([WK03]), backstepping ([WYJ15, WK03]).

In [MSDE16], the 6 degrees of freedom (6DOF) dynamics and kinematics model of the underwater robots is linearized about the given desired trajectory. A linear time variant (LTV) state space model for underwater robots is obtained. And then a linear quadratic controller is designed according the linear models and applied to the nonlinear robot dynamics to track the desired trajectory. Our idea is very similar to this, we also want to linearize the nonlinear model along the trajectory. However, since we should utilize the linear models for the following optimization algorithm, a relative small number of linearized models is expected for the given trajectory. By special nonlinear transformation [SPK02b] along trim trajectory, the error dynamics between the real robot states and the trajectory-determined states can be uniquely linearized, which satisfies our trajectory design aim. The trim trajectory is widely used for unmanned aerial vehicles (UAVs) because of its simplicity for controling and analysing. Its benefits and practical usages can be found in [BA05, BLS08, Seb15].

1.2.3 Computational Design

The basic idea of this work was inspired from [DSZ+16]. In this paper, Du et al. proposed an interactive procedure to computationally design, optimize, and fabricate multicopters. The multicopter can be constructed from a collection of components including propellers, motors, and carbon fiber rods. They proposed an algorithm optimizing shape and controller parameters of the current design to ensure the multicopter's proper operation. Other merics including payload, battery usage, size, and cost are integrated into the optimization algorithm.

Generally speaking, both underwater robot and multicopter has the 6DOF dynamics. In spite of this, the underwater robots usually have full 6DOF actuation and thus contain a number of equilibrium points, while the multicopter stays at the equilibrium when the total resultant thrust equal to the gravity and the total result moment is equal to zero. In addition, beside the thrusters the underwater robots have also fins as control surfaces and their hydrodynamic characteristics are nonlinear and of high-order. The underwater robots possess a more complicated structure, making the geometric optimization a challenging task.

There are still a lot other works concerning the computational design of robots. In [SSL+14], a data-driven method for designing 3D models that can be fabricated was proposed. Schulz et al. propose the end-to-end system for design of robots with ground locomotion [SSS+17]. The biggest advantage of the computational design is that it allows people of all skill level to design and fabricate robots.

1.2.4 Geometric Optimization

[JSHL12] present a methodology to optimize the AUV profile in order to reduce the total resistance with Computational Fluid Dynamics (CFD). The majority of works in terms of actuator optimization discusses the control allocation problem, e.g [FJP08, JFB04, JFT05].
In [XWW15], Xu et al. propose a novel local optimization of thruster configuration based on a synthesized positioning capability criterion.
In our work, we optimize not only the hull but also the actuator configuration and our actuator includes thrusters and control fins. The control location problem normally formulated as optimization problems whose objective is to produce the specified generalized forces to minimize the control effort (power consumption), the position of thrusters is already fixed after the robot design is finished. In this case, the decision variables are normally the azimuth angle of the thrusters and the forces generated by them. Our optimizations are performed during the design phase. Hence, our decision variables contain not only the thrust forces, more importantly, an optimal thruster location should be found. Similarly, the deflection angle and the fin location should be optimized at the same time.

1.3 Contributions

To summarize, our main contribution in this work include:

- Deriving the design requirements for robot kinematics and dynamics from a set of trim trajectories.
- Approximating each constituent module of the underwater robot as a simple shape determined by several geometric decision variables and building the robot dynamics parametrized with them.
- Identifying the couplings among all geometric decision variables.
- Formulating an optimization problem that can optimize the size of the robot hull enclosure according to different design specifications.
- Formulating an optimization problem that can jointly optimize positions and orientations, spin directions and control parameters satisfying the trajectory-based design requirements.
- Linearizing and transforming the nonlinear underwater robot dynamic system into a linear switched system, designing the switched Linear Quadratic Regulator (LQR) for the system.
- Finding the state cost matrices and the input cost matrices for the switched LQR based on the stability.

- Providing an efficient numerical approach to solve the actuator placement optimization by using the multi-stage iterative optimization method.

Chapter 2

Modular Modeling of Underwater Robot Prototype

Based on the ideas of the computational underwater robot design, the designer should be able to assemble a underwater robot from a collection of components (hull enclosure, batteries, fins, thrusters, CPU, IMU, sensor and so on). The customizable robot structure implies the robot dynamics is determined by number and type of constituent components and their geometric variables. Unlike the conventional modeling method for which all components are fixed, in this chapter, we model the robot modularly. More precisely, each robot component is approximated as simple geometric shape determined by several size parameters. Then we can parametrize the mass, the moment of inertia and the hydrodynamic coefficients with these parameters. In addition, the position and orientation parameters also contribute the dynamics of the whole robot. By means of this modeling approach, a customizable dynamic model is obtained.

The underwater vehicles are mainly composed of hull and actuators and actuators can be further divided into thrusters and control fins. The modular modeling method is mainly adapted from [Tan99]. We only consider the hydrodynamic effects of hull, while the added mass and hydrodynamic damping of fins and thrusters are also calculated. Also, we ignore the moment of inertia of fins and thrusters and assume that it is merely determined by hull components. Besides, we take the electronic devices into account and model them as cuboid. They have contribution to the total mass and moment of inertia of the robot and specify the minimal size of the hull volume.

The inertia tensor for each modular component s with respect to its center of mass CG_s is given by

$$\boldsymbol{I}_{g,s} = \begin{pmatrix} I_{xx,s} & 0 & 0 \\ 0 & I_{yy,s} & 0 \\ 0 & 0 & I_{zz,s} \end{pmatrix}. \tag{2.1}$$

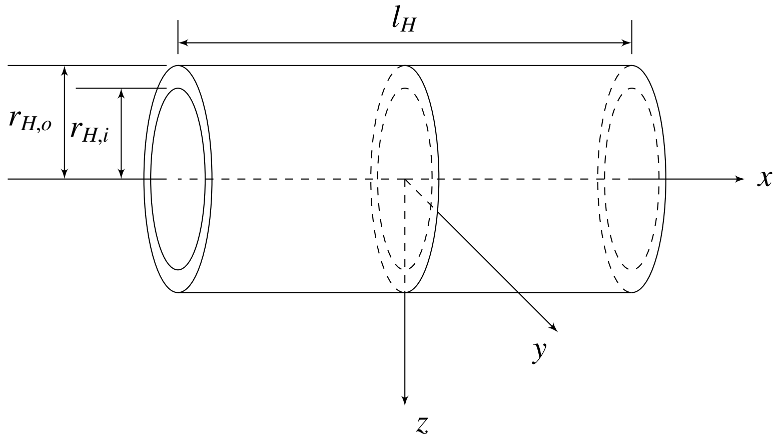


Figure 2.1: Modeling of hull as hollow cylinder

2.1 Modeling of Hull Components

As the main body of the underwater robot, the navigation system, sensory system and the power supply system are embedded in the hull.
The hull shell is modelled as hollow cylinder, as illustrated in Figure 2.1. We use three parameters to characterize it including the inner radius $r_{H,i}$, the outer radius $r_{H,o}$ and the hull length l_H. Then its mass and moment of inertia with respect to its center of gravity can be calculated as follows:

$$m_H = \rho_H \pi (r_{H,o}^2 - r_{H,i}^2) l_H, \tag{2.2}$$

$$I_{xx,H} = \frac{1}{2} m_H (r_{H,o}^2 + r_{H,i}^2), \tag{2.3}$$

and

$$I_{yy,H} = I_{zz,H} = \frac{1}{4} m_H (r_{H,o}^2 + r_{H,i}^2 + \frac{l_H^2}{3}). \tag{2.4}$$

In the following part, we simplify the notation of $r_{H,o}$ as r_H. Within the hull, we model the battery as solid cylinder, based on the defined coordinate in 2.2, the mass and moment of inertia are given as

$$m_{bat} = \rho \pi r_{bat}^2 l_{bat}, \tag{2.5}$$

$$I_{xx,bat} = I_{yy,bat} = \frac{1}{12} m_{bat} (3 r_{bat}^2 + l_{bat}^2), \tag{2.6}$$

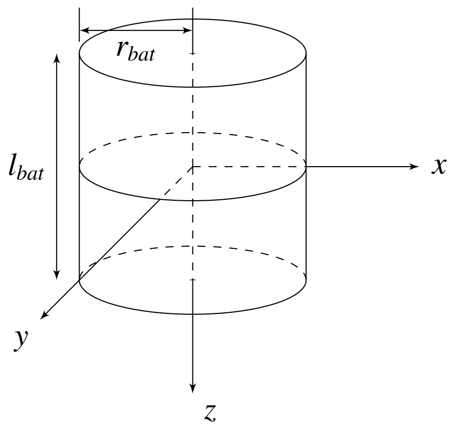


Figure 2.2: Modeling of battery as solid cylinder

and

$$I_{zz,bat} = \frac{1}{2} m_{bat} r_{bat}^2 \tag{2.7}$$

All other devices are assumed to have a constant mass m_d and a constant moment of inertia $\boldsymbol{I}_d$ with respect to the body frame.

2.2 Modeling of Control Actuators

Usually, fins and thrusters are used as actuators. In the second optimization phase, we want to optimize the geometric parameters of all actuators. For the design of robots, geometric placement of all actuators plays an decisive role for controlling performance. To find an optimal geometric properties of actuators in the designing phase, appropriate models for actuators are required to estimate the generated control efforts under different actuator configurations, and the lift of fin is treated as the control input.

2.2.1 Modeling of Thrusters

For modeling of thrusters, we adopt the modeling methods from [DSZ+16]. Based on this modeling, each thruster is parameterized by the motor orientaion $\mathbf{d}_T$ (unit vector), the motor position $\mathbf{r}_T$ in the body frame $\{b\}$ and the motor spin direction $b_T \in \{-1, 1\}$. We use u_T and $m_{T,r}$ to denote the magnitude of the thrust force and torque generated by the propeller rotation, respectively.
Yoerger et al. ([YCS90]) proposed the one-state model for the propeller. The torque Q_m and thrust T has the following relationship:

$$\dot{\omega}_p = \beta Q_m - \alpha \omega_p |\omega_p| \tag{2.8}$$

$$T = C_t \omega_p |\omega_p| \tag{2.9}$$

where ω_p is the propeller angular velocity, α and β are constant model parameters and C_t is a proportionality constant. The three parameters α, β and C_t require to be identified.
If the propeller rotates at steady state, i.e. $\dot{\omega}_p = 0$, then according to (2.8) and (2.9), the relationship between the thrust generated by the propeller and the rotation torque can be described with the following equation:

$$Q_m = \frac{\alpha}{C_t \beta} T, \tag{2.10}$$

which means that the torque generated due to the rotation of propellers is proportional to the thrust force of the propeller. Thus, we can assume the torque $m_{T,r}$ is proportional to the thrust force u_T

$$m_{T,r} = \lambda_T u_T, \tag{2.11}$$

where λ_T is the torque-force ratio. Taking the thruster motor orientation and the spin direction into consideration, we can have the rotation torque vector as

$$\mathbf{m}_{T,r} = b_T \lambda_T u_T \mathbf{d}_T. \tag{2.12}$$

Additionally, the thrust force will produce the thrust torque $\mathbf{m}_{T,t}$. For calculation of thrust torque, the center of gravity is assumed to be located in the body frame origin CO. Then the thrust torque can be expressed as

$$\mathbf{m}_{T,t} = \mathbf{r}_T \times u_T \mathbf{d}_T, \tag{2.13}$$

where $\mathbf{r}_T = (x_T, y_T, z_T)$ is the position vector of the thruster in the body frame $\{b\}$. Combining the thrust torque and the rotation torque, we express the total torque produced by a thruster as

$$\mathbf{m}_T = \mathbf{m}_{T,r} + \mathbf{m}_{T,t} = b_T \lambda_T u_T \mathbf{d}_T + \mathbf{r}_T \times u_T \mathbf{d}_T \in \mathbb{R}^3 \tag{2.14}$$

The force direction is determined by the thruster motor orientation $\mathbf{d}_T$ and the force vector $\mathbf{f}_T$ can be represented as

$$\mathbf{f}_T = u_T \mathbf{d}_T \in \mathbb{R}^3 \tag{2.15}$$

The generalised force vector for thruster is defined as

$$\tau_T = \begin{pmatrix} \mathbf{f}_T \\ \mathbf{m}_T \end{pmatrix} \tag{2.16}$$

If we treat the thrust as the control input, we can define an unique mapping vector for a thruster based on its geometric parameters.

$$\boldsymbol{B}_T = \begin{pmatrix} \mathbf{d}_T \\ b_T \lambda_T \mathbf{d}_T + \mathbf{r}_T \times \mathbf{d}_T \end{pmatrix} \in \mathbb{R}^6. \tag{2.17}$$

Then the generalised force generated by thrust can be in the following form:

$$\tau_T = \boldsymbol{B}_T u_T \tag{2.18}$$

2.2.2 Modeling of Control Fins

An accurate hydrodynamic modeling of control fins in Appendix C introduces high order nonlinear terms into the input matrix $\boldsymbol{B}_a$. The velocity of fin middle point in the body frame $\{b\}$ is denoted by $\mathbf{v}_{fin} = (u_{fin},\ v_{fin},\ w_{fin})^T$ and its three components have the following relationship with the robot linear velocity $\mathbf{v} = (u,\ v,\ w)^T$:

$$u_{fin} = u + z_{fin} q - y_{fin} r, \tag{2.19}$$

$$v_{fin} = v + x_{fin} r - z_{fin} p, \tag{2.20}$$

$$w_{fin} = w + y_{fin} p - x_{fin} q. \tag{2.21}$$

Then the fin velocity $\mathbf{v}^f_{fin} = (u^f_{fin},\ v^f_{fin},\ w^f_{fin})^T$ in the fin frame $\{f\}$ is calculated as:

$$\mathbf{v}^f_{fin} = \boldsymbol{R}^f_b \mathbf{v}_{fin}, \tag{2.22}$$

where $\boldsymbol{R}^f_b$ is determined by the fin position vector $\mathbf{r}_F$ in the body frame $\{f\}$. Then the angle of attack α is calculated as

$$\alpha = atan2(w^f_{fin}, u^f_{fin}). \tag{2.23}$$

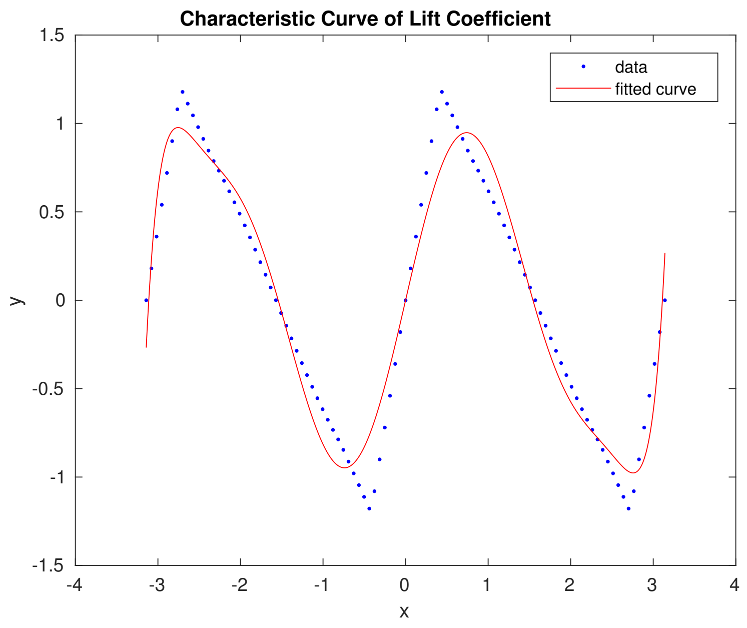


Figure 2.3: Approximation of relationship between lift coefficient C_L and attack angle α

The *atan*2 can not be approximated by a polynomial term numerically. Furthermore, the lift and drag coefficients C_L, C_D are piecewise functions of the angle of attack α. We are able to use high-order polynomials to approximate them. The lift coefficient characteristic curve is depicted in Figure C.2. The curve fitting result is shown in Figure 2.3 and the relationship between C_L and the attack angle α is

$$C_L = -0.1463\alpha^8 + 1.234\alpha^6 - 3.61\alpha^2 + 0.09403. \tag{2.24}$$

Similarly, we are able to approximate the drag coefficient characteristic curve in C.3 with high-order polynomials whose result is depicted in Figure 2.4. The fitting function is:

$$C_D = -0.4653\alpha^9 - 3.219\alpha^7 + 8.356\alpha^5 - 9.75\alpha^3 + 3.729\alpha. \tag{2.25}$$

These two high-order terms require considerably large computational capacity for the optimization. The aforementioned two problems from accurate hydrodynamic modeling make the optimization problem nearly unsolvable.
Strict restrictions on the fin geometric property, for instance the fins can only be mounted on the horizontal and vertical planes of the hull or they must be placed symmetrically in pairs , will bring the robot design back into the traditional way.
A compromise can be found by means of small angle of attack assumption which is inspired by modeling methods in [NM12]. Fins are modeled as rectangle with length a_F and width b_F. Assume the side edge b_F is directly attached on the hull surface. Then we represent the position and the orientation of the fins in the hull cylindrical coordinate system. As illustrated in Figure 2.5, x_F is defined as the x-coordinate of fin geometric center

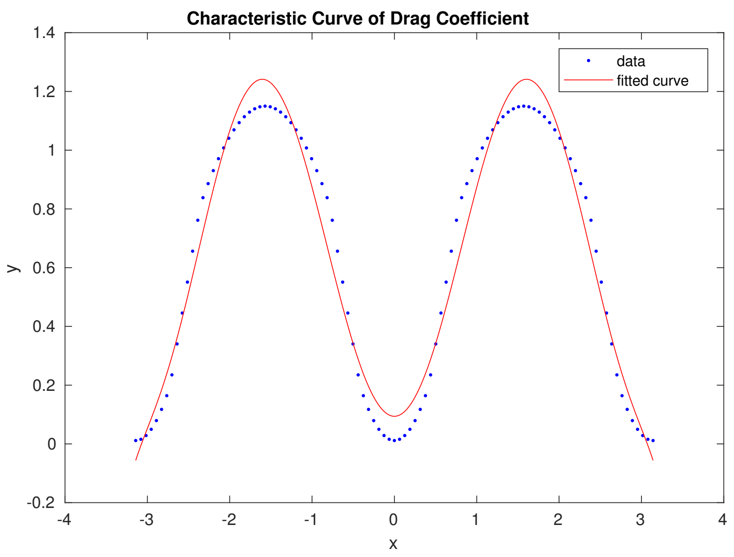


Figure 2.4: Approximation of relationship between drag coefficient C_D and attack angle α

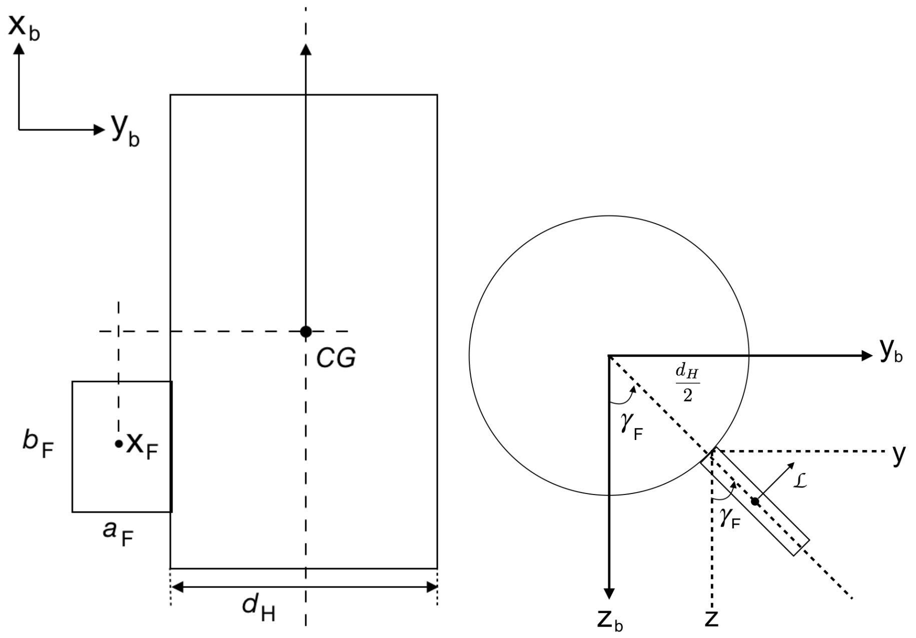


Figure 2.5: Definition of fin geometric parameters

in body frame $\{b\}$. We rotate the xz-plane of the robot body frame in the counterclockwise direction by angle γ_F until the fin geometric center located in this plane. Consequently, the i-th fin geometric center vector $\mathbf{r}_{F,i}$ can be written as

$$\mathbf{r}_{F,i} = \begin{pmatrix} x_{F,i} \\ 0.5(d_H + a_{F,i})\sin(\gamma_{F,i}) \\ 0.5(d_H + a_{F,i})\cos(\gamma_{F,i}) \end{pmatrix} \tag{2.26}$$

in body frame $\{b\}$. Then we can calculate the center cross product vector as

$$\mathbf{r}_{F,i}\times = \begin{pmatrix} 0 & -\frac{1}{2}(d_H + a_{F,i})\cos(\gamma_{F,i}) & \frac{1}{2}(d_H + a_{F,i})\sin(\gamma_{F,i}) \\ \frac{1}{2}(d_H + a_{F,i})\cos(\gamma_{F,i}) & 0 & -x_{F,i} \\ -\frac{1}{2}(d_H + a_{F,i})\sin(\gamma_{F,i}) & x_{F,i} & 0 \end{pmatrix}. \tag{2.27}$$

In the following, we use $\mathbf{e}_1 = [0 \quad 0 \quad 1]^T$, $\mathbf{e}_2 = [0 \quad 1 \quad 0]^T$ and $\mathbf{e}_3 = [1 \quad 0 \quad 0]^T$ to represent unit vectors in the body frame $\{b\}$ and n_f to denote the number of control fins. For each fin, the lift L and drag D of fins can be calculated as

$$\mathbf{L}_i(\alpha_i) := a_{F,i} b_{F,i} C_L(\alpha_i) Q(U_{fin}) \begin{pmatrix} 0 \\ cos(\gamma_{F,i}) \\ -sin(\gamma_{F,i}) \end{pmatrix} \tag{2.28}$$

$$\mathbf{D}_i(\alpha_i) = S C_D(\alpha_i) Q(U_{fin}) \mathbf{e}_1, \tag{2.29}$$

where $a_{F,i}$ and $b_{F,i}$ are the length and width of the i-th fin, respectively. C_L is the lift coefficient of fin and C_D is the drag coefficient. $Q(U_{fin}) = 0.5\rho U_{fin}^2$ is a short notation for simplicity. U_{fin} is the magnitude of fin middle point velocity vector relative to the surrounding fluid in the body frame $\{b\}$ and $U_{fin} = \sqrt{u_{fin}^2 + v_{fin}^2 + w_{fin}^2}$.
If the surge velocity is dominant, that is, u is much bigger than v and w, the previous formulas for calculating the lift and drag can be rewritten as

$$\mathbf{L}_i(\alpha_i) := a_{F,i} b_{F,i} C_L(\alpha_i) Q(u_{fin}) \begin{pmatrix} 0 \\ cos(\gamma_{F,i}) \\ -sin(\gamma_{F,i}) \end{pmatrix} \tag{2.30}$$

$$\mathbf{D}_i(\alpha_i) = S C_D(\alpha_i) Q(u_{fin}) \mathbf{e}_1. \tag{2.31}$$

It turns out the resultant force vector produced by n_f fins can be calculated as follows:

$$\mathbf{f}_F(\alpha_i) = \begin{pmatrix} \sum_{i=1}^{n_f} \mathbf{L}_i(\alpha_i)^T \mathbf{e}_1 + \sum_{i=1}^{n_f} \mathbf{D}_i(\alpha_i)^T \mathbf{e}_1 \\ \sum_{i=1}^{n_f} \mathbf{L}_i(\alpha_i)^T \mathbf{e}_2 + \sum_{i=1}^{n_f} \mathbf{D}_i(\alpha_i)^T \mathbf{e}_2 \\ \sum_{i=1}^{n_f} \mathbf{L}_i(\alpha_i)^T \mathbf{e}_3 + \sum_{i=1}^{n_f} \mathbf{D}_i(\alpha_i)^T \mathbf{e}_3 \end{pmatrix}. \tag{2.32}$$

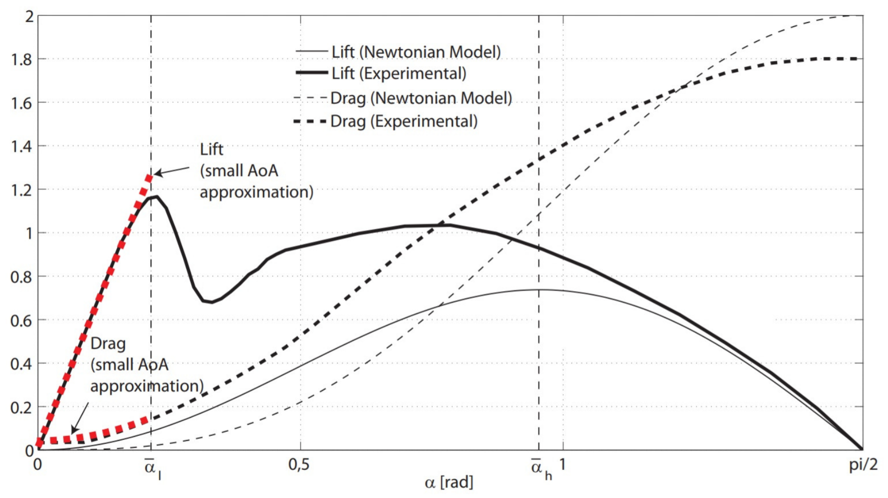


Figure 2.6: Small angle of attack assumption [NM12]

The resultant torque vector is calculated as:

$$\tau_F(\alpha_i) = \sum_{i=1}^{n_f} \mathbf{r}_{F,i} \times \mathbf{L}_i(\alpha_i) + \sum_{i=1}^{n_f} \mathbf{r}_{F,i} \times \mathbf{D}_i(\alpha_i). \tag{2.33}$$

For small angle of attack α, as depicted in Figure 2.6, it is reasonable to approximate the lift and drag coefficients as

$$C_L(\alpha_i) = c_L \alpha_i, \tag{2.34}$$

$$C_D(\alpha_i) = c_D \alpha_i^2, \tag{2.35}$$

where c_L, c_D are fin-specific parameters estimated from experiments. For the following calculations, we adopt this assumption. The drag moment of the i-th fin is calculated as

$$\mathbf{r}_{F,i} \times \boldsymbol{D}_i(\alpha_i) =$$

$$\begin{pmatrix} 0 & -\frac{1}{2}(d_H + a_{F,i})\cos(\gamma_{F,i}) & \frac{1}{2}(d_H + a_{F,i})\sin(\gamma_{F,i}) \\ \frac{1}{2}(d_H + a_{F,i})\cos(\gamma_{F,i}) & 0 & -x_{F,i} \\ -\frac{1}{2}(d_H + a_{F,i})\sin(\gamma_{F,i}) & x_{F,i} & 0 \end{pmatrix} \begin{pmatrix} c_D a_{F,i} b_{F,i} Q(u_{fin}) \alpha_i^2 \\ 0 \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ \frac{1}{2}(d_H + a_{F,i})\cos(\gamma_{F,i}) c_D a_{F,i} b_{F,i} Q(u_{fin}) \\ -\frac{1}{2}(d_H + a_{F,i})\sin(\gamma_{F,i}) c_D a_{F,i} b_{F,i} Q(u_{fin}) \end{pmatrix} \alpha_i^2. \tag{2.36}$$

The lift moment is given by

$$
\begin{aligned}
&\mathbf{m}_{F,i} = \mathbf{r}_{F,i} \times \boldsymbol{L}_i(\alpha_i) = \\
&\begin{pmatrix} 0 & -\frac{1}{2}(d_H + a_{F,i})\cos(\gamma_{F,i}) & \frac{1}{2}(d_H + a_{F,i})\sin(\gamma_{F,i}) \\ \frac{1}{2}(d_H + a_{F,i})\cos(\gamma_{F,i}) & 0 & -x_{F,i} \\ -\frac{1}{2}(d_H + a_{F,i})\sin(\gamma_{F,i}) & x_{F,i} & 0 \end{pmatrix}
\begin{pmatrix} 0 \\ c_L a_{F,i} b_{F,i} Q(u_{fin})\cos(\gamma_{F,i})\alpha_i \\ c_L a_{F,i} b_{F,i} Q(u_{fin})\sin(\gamma_{F,i})\alpha_i \end{pmatrix} \\
&= \begin{pmatrix} -c_L a_F b_F Q(u_{fin})\frac{1}{2}(d_H + a_F) \\ c_L a_F b_F Q(u_{fin}) x_F \sin(\gamma_F) \\ c_L a_F b_F Q(u_{fin}) x_F \cos(\gamma_F) \end{pmatrix} \alpha_i.
\end{aligned}
\tag{2.37}
$$

Note that the moment of fins should be calculated according the center of mass CG of the whole robot, that is, $\mathbf{r}_{F,i}$ here should be the vector connecting the center of fins and the center of mass CG. However, we assume that, for calculating the fin moments, the center of mass of the robot is the origin CO of the body frame $\{b\}$. It means that the vector $\mathbf{r}_{F,i}$ denotes the position of fin geometric center in the robot body frame.
We use the lift force generated by fins as the control input and treat the drag as disturbance. Similar to the defined mapping vector of thrusters, we also define the mapping vector for fins as follows:

$$
\boldsymbol{B}_F = \begin{pmatrix} 0 \\ c_L a_F b_F Q(u_{fin})\cos(\gamma_F) \\ c_L a_F b_F Q(u_{fin})\sin(\gamma_F) \\ -c_L a_F b_F Q(u_{fin})\frac{1}{2}(d_H + a_F) \\ c_L a_F b_F Q(u_{fin}) x_F \sin(\gamma_F) \\ c_L a_F b_F Q(u_{fin}) x_F \cos(\gamma_F) \end{pmatrix} \in \mathbb{R}^6. \tag{2.38}
$$

The fin mapping matrix is not only geometry-dependent but also state dependent, since it contains the surge velocity term $Q(u_{fin})$. For small drift angle, the angle of attack *alpha* of the hydrofoils can be approximated by the mechanic al angle of rotation of the hydrofoils [FJP08]. For our case, it means $\alpha = \delta_F$, where δ_F is the deflection angle of fins. Then the generalized force generated by fins can be written in the following form:

$$
\tau_F = \begin{pmatrix} \mathbf{f}_F \\ \mathbf{m}_F \end{pmatrix} = \boldsymbol{B}_F \delta_F. \tag{2.39}
$$

2.3 Dynamics Construction

The moments of inertia for each individual modules making up the robot is calculated with respect to the module's own frame, where the hull frame is usually chosen as the robot body frame $\{b\}$. It means, the elemental moments of inertia should be transformed to the body frame $\{b\}$. For each submodule whose center of gravity CG_s is $(x_s, y_s, z_s)^T$ in body frame $\{b\}$, we can compute their moments of inertia in $\{b\}$ as follows:

$$I^b_{xx,s} = I_{xx,s} + m_s(y_s^2 + z_s^2), \tag{2.40}$$

$$I^b_{yy,s} = I_{yy,s} + m_s(z_s^2 + x_s^2), \tag{2.41}$$

$$I^b_{zz,s} = I_{zz,s} + m_s(x_s^2 + y_s^2). \tag{2.42}$$

where $I^b_{xx,s}, I^b_{yy,s} and I^b_{zz,s}$ are moments of inertia with respect to the body axes $\{b\}$ of the component s. In this work, we only consider the moment of inertia of batteries, hull enclosure and the constant moment of inertia of all devices within the hull, the moments of inertia of fins, thrusters and other components are neglected.
Suppose we have totally N components, then the resultant center of mass $\mathbf{r}_G$ for the whole robot is

$$\mathbf{r}_G = \frac{\sum_{s=1}^N m_s \mathbf{r}_{G,s}}{\sum_{s=1}^N m_s}, \tag{2.43}$$

where m_s is the mass of the component s, and $\mathbf{r}_{G,s}$ is the position vector of the component s in the body frame $\{b\}$.

2.4 Estimation of the Hydrodynamic Effects

Hydrodynamic effects are the distinct feature for underwater robots. Strictly speaking, each component (hull, thrusters and fins) contribute to the hydrodynamic forces and moments. However, hydrodynamic effects depend not only on the geometric structure but also on the working condition (e.g. the inflow fluid velocity). Normally, the underwater robot dynamics is built in body frame (usually the geometric center of the hull). However, the hydrodynamic effect of each module is calculated with respect to its own geometric center. It means they should be transformed into body frame according to their positions. This transformation will make the nonlinear and strongly coupled hydrodynamic effects more complicated. Thus, the basic assumption for the estimation is that only hull contributes significantly to the hydrodynamic coefficients, while other contributions are negligible. Because we set the hull frame as the robot body frame, no transformation is needed. The hydrodynamic effects include mainly two parts: the added mass and the hydrodynamic damping.

2.4.1 Estimating the Added Mass Coefficients

For estimating the added mass, we assume that the added mass of thrusters and fins are negligible and only the added mass of the cylindrical hull is taken into consideration.

For a symmetrical cylindrical rigid body of mass m_H with circular section of radius r_H and length l_H, the added mass coefficients (definitions in Section A.2.1) can be derived theoretically by applying the strip theory as follows:

$$X_{\dot{u}} = -0.1 m_H, \tag{2.44}$$

$$Y_{\dot{v}} = -\pi \rho r_H^2 l_H, \tag{2.45}$$

$$Z_{\dot{w}} = -\pi \rho r_H^2 l_H, \tag{2.46}$$

$$K_{\dot{p}} = 0, \tag{2.47}$$

$$M_{\dot{q}} = -\frac{1}{12} \pi \rho r_H^2 l_H^3, \tag{2.48}$$

$$N_{\dot{r}} = -\frac{1}{12} \pi \rho r_H^2 l_H^3. \tag{2.49}$$

The cylindrical hull has three planes of symmetry. When the robot is completely submerged in the fluid, The added mass matrix $\boldsymbol{M}_A$ and the Coriolis matrix $\boldsymbol{C}_A$ can be calculated as follows:

$$\boldsymbol{M}_A = diag([-X_{\dot{u}} - Y_{\dot{v}} - Z_{\dot{w}} - K_{\dot{p}} - M_{\dot{q}} - N_{\dot{r}}]) \tag{2.50}$$

$$\boldsymbol{C}_A = \begin{pmatrix} 0 & 0 & 0 & 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v \\ 0 & 0 & 0 & Z_{\dot{w}}w & 0 & -X_{\dot{u}}u \\ 0 & 0 & 0 & -Y_{\dot{v}}v & X_{\dot{u}}u & 0 \\ 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v & 0 & -N_{\dot{r}}\dot{r} & M_{\dot{q}}q \\ Z_{\dot{w}}w & 0 & -X_{\dot{u}}u & N_{\dot{r}}r & 0 & -K_{\dot{p}}p \\ -Y_{\dot{v}}v & X_{\dot{u}}u & 0 & -M_{\dot{q}}q & K_{\dot{p}}p & 0 \end{pmatrix}. \tag{2.51}$$

The total mass matrix is

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_A. \tag{2.52}$$

Since $\boldsymbol{M}_{RB} \succ 0$ and $\boldsymbol{M}_A \succeq 0$, the complete robot inertia matrix is positive definite, i.e., $\boldsymbol{M} \succ 0$, thus $\boldsymbol{M}$ is invertible.

2.4.2 Estimating the Damping Coefficients

The hydrodynamic damping can be modeled as:

$$F_H = -\frac{1}{2}\rho C_D A_r |v| v, \tag{2.53}$$

where ρ is the fluid density, C_D is the drag coefficient, A_r the reference area, and v is an arbitrary linear or angular velocity. To estimate the drag coefficient of the robot hull, we make the following the assumptions:

1. The drag coefficient C_D is dependent on the Reynolds number Re, the Mach number Ma and the direction of the flow. Low Mach number is assumed, in this case the drag coefficient is independent of it. We also assume high speed manoeuvring where the incoming flow direction is more-or-less the same and the drag coefficient C_D can be treated as a constant [Dra17].
2. The skin friction is neglected.
3. The underwater robot is performing a non-coupled motion.
4. The quadratic damping terms are dominant.
5. The lift force of the hull is not considered.

The damping matrix is defined as

$$\boldsymbol{D} = diag([-X_{u|u|}|u|, -Y_{v|v|}|v|, -Z_{w|w|}|w|, -K_{p|p|}|p|, -M_{q|q|}|q|, -N_{r|r|}|r|]). \tag{2.54}$$

The damping coefficient in the x-direction due to the viscous force acting on the frontal area is given as

$$X_{|u|u} = -\frac{1}{2}\rho C_D A_{frontal} = -\frac{1}{2}\rho \pi C_D r_H^2, \tag{2.55}$$

The damping coefficients in the y and z axes due to viscous effects on the side projected areas of the hull is given as

$$Y_{|v|v} = -\frac{1}{2}\rho C_D 2 r_H l_H \tag{2.56}$$

and

$$Z_{w|w|} = -\frac{1}{2}\rho C_D 2 r_H l_H \tag{2.57}$$

respectively.
The hull has no contribution to the roll moment, hence

$$K_{p|p|} = 0. \tag{2.58}$$

The drag force due to pitch motion is given as

$$M_{q|q|}q|q| = -\frac{1}{2}\rho w|w| \int_{-l_H/2}^{l_H/2} C_D \, 2r \, dx \, x \tag{2.59}$$

For small angular motions, $w \approx qx$. Thus, the pitch damping coefficient is given by

$$M_{q|q|} = -\frac{1}{12}\rho C_D r_H l_H^4 \tag{2.60}$$

Similarly, the horizontal drag forces contribute the yaw drag, whose coefficient is same as the pitch drag:

$$N_{r|r|} = -\frac{1}{12}\rho C_D r_H l_H^4 \tag{2.61}$$

The details for derivation of these parameters can be found in [Tan99], it also includes the drag caused by thrusters and struct linking hull and thruster which could be for future work. With help of two geometric parameters l_H and r_H, the hydrodynamic terms in the robot dynamic equation including $\boldsymbol{M}_A$, $\boldsymbol{C}_A$ and $\boldsymbol{D}$ can be computed.
To sum up, in this chapter we model the robot hull, batteries, fins as simple geometric shapes determined by several geometric parameters. In addition, for thrusters we only pay attention on their positions, motor orientations and spin directions regardless of their shapes. By means of this modeling method, we are able to construct a customizable robot dynamics parametrized with these geometric parameters. They will be chosen as decision variables for the robot geometry optimization so that the current dynamics can be computed at each optimization iteration.

Chapter 3

Path Planning for Underwater Robots

Underwater robots are used in practice to perform rescue tasks, to monitor the environment or to explore mysterious deep sea world. The success of these jobs requires an efficient trajectory planning. One important novelty of this work is that we propose kinematic and dynamic specifications from the trajectories. Due to the convenience for analysis and control design, trim trajectories will be chosen for our problem.

3.1 Trim Trajectory

The trim trajectory comes originally from path planning for aircraft. It facilitates the planning and control problem since it corresponds to a stable states. The vehicle motion is uniform in the body frame under trim condition, i.e., the surge, sway, heave, roll, pitch and yaw velocities stay constant within one trim segment. For our case, the linearized error dynamics between the desired trim trajectory segment and the real one is unique after a specific nonlinear transformation (discussed in Chapter 4) so that we are able to use the analysis and design methods for LTI system to design the optimization algorithm. This will be discussed in the next chapter.
A trim trajectory can be parameterized by three parameters, which can be denoted as

$$\eta_{\mathcal{T}} : \left(\|\mathbf{v}_{\mathcal{T}}\|, \dot{\psi}_{\mathcal{T}}, \gamma_{\mathcal{T}} \right)^T \tag{3.1}$$

where $\mathbf{v}_{\mathcal{T}}$ is the trim speed. For Frenet-Serret frames $\{FS\}$, the trim speed is equal to the surge velocity u. The angle $\gamma_{\mathcal{T}}$ is the motion path angle, for aircraft it is the so-called flight path angle. The trim yaw angle is given by $\psi_{\mathcal{T}}(t) = \dot{\psi}_{\mathcal{T}} t + \psi_0$ where $\dot{\psi}_{\mathcal{T}} \in \mathbb{R}$ is the constant yaw rate and $\psi_0 \in [0, 2\pi)$ is the initial yaw angle. Along the trim trajectory, the roll angle and pitch angle remain unchanged, i.e., $\dot{\phi}_{\mathcal{T}} = 0$ and $\dot{\theta}_{\mathcal{T}} = 0$.

$$\dot{\eta}_{\mathcal{T}} = \|\mathbf{v}_{\mathcal{T}}\| \begin{pmatrix} \cos(\gamma)_{\mathcal{T}} \cos(\psi_{\mathcal{T}}(t) - \psi_v) \\ \sin(\gamma)_{\mathcal{T}} \cos(\psi_{\mathcal{T}}(t) - \psi_v) \\ -\sin(\gamma_{\mathcal{T}}) \end{pmatrix}. \tag{3.2}$$

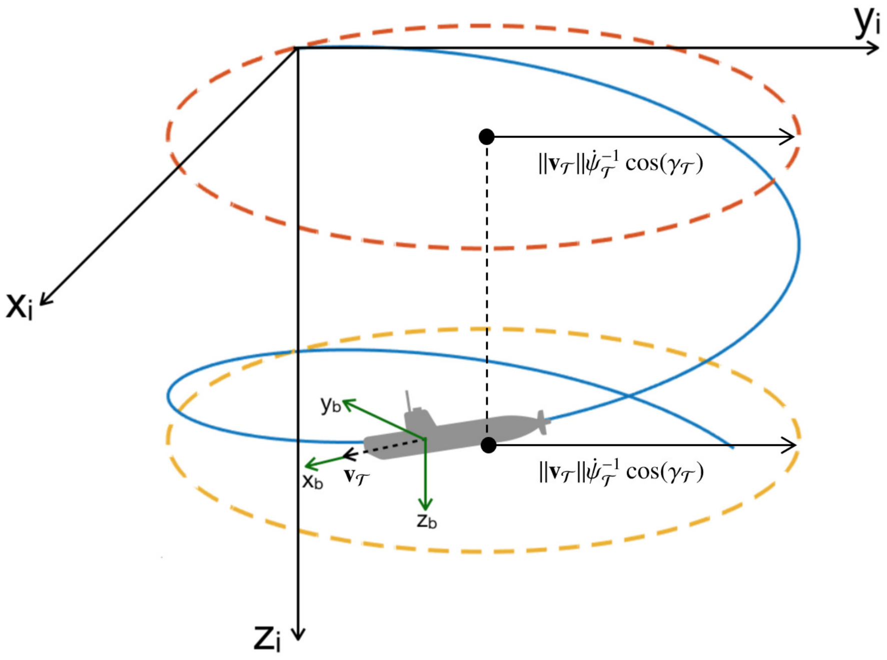

y_i
x_i
z_i
$||\mathbf{v}_\mathcal{T}||\dot{\psi}_\mathcal{T}^{-1}\cos(\gamma_\mathcal{T})$
$||\mathbf{v}_\mathcal{T}||\dot{\psi}_\mathcal{T}^{-1}\cos(\gamma_\mathcal{T})$
y_b
x_b
$\mathbf{v}_\mathcal{T}$
z_b

Figure 3.1: Trim trajectory

ψ_v is the angle between the robot's heading and the velocity vector. Because we use Frenet-Serret frame discussed in the next section, the vehicle's heading is coincided with the velocity vector, the angle ψ_v should be equal to zero. Integrating the above equation shows that a trim trajectory segment corresponds to a helix with radius $||\mathbf{v}_\mathcal{T}||\dot{\psi}_\mathcal{T}^{-1}\cos(\gamma_\mathcal{T})$ parameterized by

$$\eta_\mathcal{T}(t) = ||\mathbf{v}_\mathcal{T}||\dot{\psi}_\mathcal{T}^{-1}\cos(\gamma)\begin{pmatrix}\sin(\psi_\mathcal{T}(t)-\psi_v)-\sin(\psi_0-\psi_v)\\ \cos(\psi_\mathcal{T}(t)-\psi_v)+\cos(\psi_0-\psi_v)\\ -\dot{\psi}_\mathcal{T}\tan(\gamma_\mathcal{T})t\end{pmatrix}+\eta_0 \tag{3.3}$$

where $\eta_0 = [x_0, y_0, z_0]^T \in \mathbb{R}^3$ is the initial trim trajectory position. As illustrated in figure 3.2, the motion path angle $\gamma_\mathcal{T}$ is equal to the sum of the angle of attack α and the trim pitch angle $\theta_\mathcal{T}$, i.e., $\gamma_\mathcal{T} = \alpha + \theta_\mathcal{T}$. After implementing the Frenet-Serret frame for the trajectory frame, the robot velocity vector should be coincided with the x-axis of the robot body frame $\{b\}$, meaning that the angle of attack α is equal to zero, and $\gamma_\mathcal{T} = \theta_\mathcal{T}$.

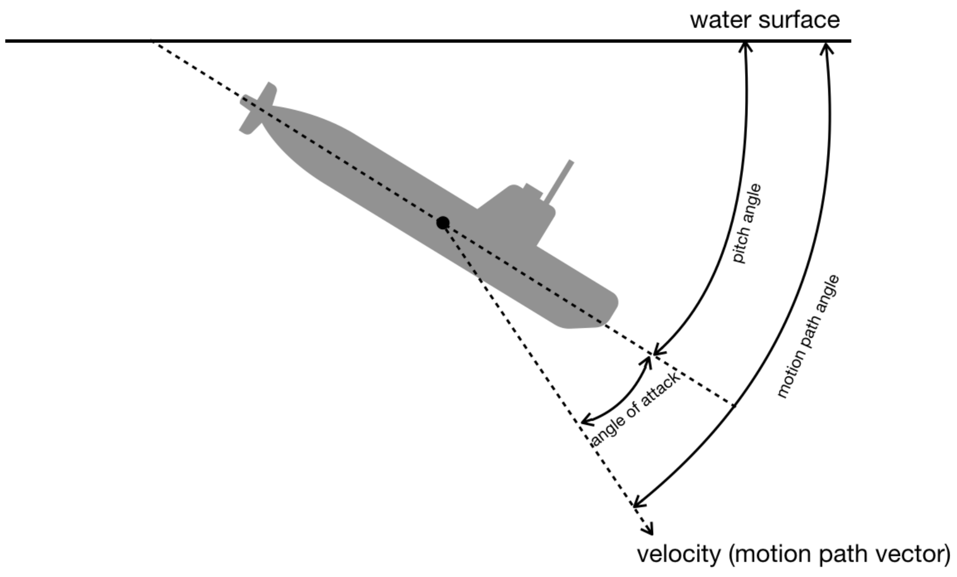


Figure 3.2: Trajectory angles

3.2 Frenet-Serret Frame

The underwater robots have six dynamic variables (3 linear velocities and 3 angular velocities) and six kinematic variables (positions and Euler angles in inertial frame $\{i\}$), all of which fully describes the robot's behaviour together. The trim trajectory merely offer us the information about the desired kinematic position, we still lack the data about the desired orientation (Euler angles) and the desired velocities along these trim trajectories. The Frenet-Serret model has the advantage of being a well-established means of modeling continuous spatial curves. The definition of the Frenet-Serret frame is elaborated in follows:
When a particle moves along a continuous, differentiable curve in three-dimensional Euclidean space $\mathbb{R}^3$, its motion can be characterized by three unit vectors: $\mathbf{T}^i(t)$, $\mathbf{N}^i(t)$ and $\mathbf{B}^i(t)$. $\mathbf{T}^i(t)$ is a unit vector tangent to the curve, denoting the motion direction.

$$\mathbf{T}^i(t) = \frac{\eta'_{\mathcal{T}}(t)}{||\eta'_{\mathcal{T}}(t)||}. \tag{3.4}$$

The normal vector $\mathbf{N}^i(t)$ is equal to the derivative with respect to the arclength parameter of the curve divided by its length which can be calculated as follows:

$$\mathbf{N}^i(t) = \frac{\mathbf{T}'(t)}{||\mathbf{T}'(t)||} = \frac{\eta'_{\mathcal{T}}(t) \times (\eta''_{\mathcal{T}}(t) \times \eta'_{\mathcal{T}}(t))}{||\eta'_{\mathcal{T}}(t)|| \cdot ||\eta''_{\mathcal{T}}(t) \times \eta'_{\mathcal{T}}(t))||}. \tag{3.5}$$

The binomal vector $\mathbf{B}^i(t)$ is the cross product of the normal vector $\mathbf{N}^i(t)$ and the tangent vector $\mathbf{T}^i(t)$ and is calculated as

$$\mathbf{B}^i(t) = \mathbf{T}^i(t) \times \mathbf{N}^i(t) = \frac{\eta_{\mathcal{T}}^{''}(t) \times \eta_{\mathcal{T}}^{'}(t)}{\|\eta_{\mathcal{T}}^{''}(t) \times \eta_{\mathcal{T}}^{'}(t))\|}. \tag{3.6}$$

These three vectors are expressed in inertial world frame $\{i\}$ because they describe the kinematic property of the robot. $\mathbf{T}^i(t)$, $\mathbf{N}^i(t)$ and $\mathbf{B}^i(t)$ constitute an orthonormal basis spanning in $\mathbb{R}^3$ and we use $\{FS\}$ to denote the Frenet-Serret frame.
The transformation matrix from the Frenet-Serret frame to the inertial world frame $\{i\}$ is defined as

$$\begin{aligned} \boldsymbol{R}_{FS}^i &= (\mathbf{T}^i(t) \quad \mathbf{N}^i(t) \quad \mathbf{B}^i(t)) \\ &= \begin{pmatrix} c\psi_{\mathcal{T}} c\theta_{\mathcal{T}} & -s\psi_{\mathcal{T}} c\phi_d + c\psi_{\mathcal{T}} s\theta_{\mathcal{T}} s\phi_{\mathcal{T}} & s\psi_{\mathcal{T}} s\phi_{\mathcal{T}} + c\psi_{\mathcal{T}} c\phi_{\mathcal{T}} s\theta_{\mathcal{T}} \\ s\psi_{\mathcal{T}} c\theta_{\mathcal{T}} & c\psi_{\mathcal{T}} c\phi_{\mathcal{T}} + s\phi_{\mathcal{T}} s\theta_{\mathcal{T}} s\psi_{\mathcal{T}} & -c\psi_{\mathcal{T}} c\phi_{\mathcal{T}} + s\theta_{\mathcal{T}} s\psi_{\mathcal{T}} c\phi_{\mathcal{T}} \\ -s\theta_{\mathcal{T}} & c\theta_{\mathcal{T}} s\phi_{\mathcal{T}} & c\theta_{\mathcal{T}} c\phi_{\mathcal{T}} \end{pmatrix} \end{aligned} \tag{3.7}$$

Thus, we can calculate the desired orientation (Euler angles) $\lambda_{\mathcal{T}} = (\phi_{\mathcal{T}},\ \theta_{\mathcal{T}},\ \psi_{\mathcal{T}})^T$ along the trim trajectory as follows:

$$\theta_{\mathcal{T}} = -\sin^{-1}(\mathbf{T}^i(t)), \tag{3.8}$$

$$\phi_{\mathcal{T}} = -\sin^{-1}(\sec(\theta_{\mathcal{T}}) \cdot \mathbf{N}^i(t)) \tag{3.9}$$

The desired yaw angle comes from the definition of the trim trajectory:

$$\psi_{\mathcal{T}} = \dot{\psi}_{\mathcal{T}} t + \psi_0. \tag{3.10}$$

In the previous discussions, we have derived the complete desired kinematic states $\eta_{\mathcal{T}} = (\mathbf{p}_{\mathcal{T}},\ \lambda_{\mathcal{T}})^T$ for underwater robots for the trim trajectory. The desired dynamic states can be determined from the kinematics with help of the rotation matrix $\boldsymbol{R}_{FS}^i$.
For a given trim trajectory, we can calculate the desired velocities as follows:

$$\upsilon_{\mathcal{T}} = \begin{pmatrix} \mathbf{v}_{\mathcal{T}} \\ \omega_{\mathcal{T}} \end{pmatrix} = (\boldsymbol{R}_{FS}^i)^{-1}. \tag{3.11}$$

In conclusion, the 12 state variables $\mathbf{x}_{\mathcal{T}} = (\eta_{\mathcal{T}},\ \upsilon_{\mathcal{T}})^T$ fully specify the kinematics and dynamics the robot should perform along the trim trajectory. We will use the desired states to derive the error dynamics in the next chapter.

Chapter 4

Trajectory Tracking Error Dynamics

The underwater robot system is a strongly coupled nonlinear system. Traditionally, the underwater robot dynamics is lineazied about constant states with specific values. For instance, in [Fos11] (pp.174), low-speed application is assumed for neutrally buoyant submersible. Thus, the nonlinear Coriolis, centripetal, damping, restoring and buoyancy forces and moments can be linearized about $\upsilon = \mathbf{0}$ and $\phi = \theta = 0$. The linearized manoeuvring model ([Fos11], pp.131) for marine craft is based on the assumption of constant cruise speed U. Also, the nonlinear damping is approximated by a linear damping matrix.

In this work, we present a path-based linearization method for the error dynamics between robot's real states and the path-determined states. Our method is in essence also linearization about constant states. Nevertheless, our linearization is based on all dynamic (velocities) and kinematic (position and Euler angles) states except for the yaw angle ψ instead several of them. Therefore, we bring nearly the full information of the trajectory into the linearized model. Furthermore, we do not require explicit values for these states. In other words, our linearized model is a function of the states. For a specific trim trajectory, we evaluate the function with explicit values from the trajectory specifications. The state variables of underwater robots can be divided into two groups, i.e., kinematic states and dynamic states. Dynamic states include three linear and three angular velocities represented in the robot frame $\{b\}$:

$$\mathbf{x}_{dyn} = \upsilon = \begin{pmatrix} \mathbf{v} \\ \omega \end{pmatrix} = \begin{pmatrix} u \\ v \\ w \\ p \\ q \\ r \end{pmatrix} \in \mathbb{R}^6. \tag{4.1}$$

Kinematic states consist of the position and orientation of the underwater robot in the inertial world frame $\{i\}$:

$$\mathbf{x}_{kin} = \eta = \begin{pmatrix} \mathbf{p} \\ \lambda \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{pmatrix} \in \mathbb{R}^6. \tag{4.2}$$

The whole states are defined as:

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_{dyn} \\ \mathbf{x}_{kin} \end{pmatrix} \in \mathbb{R}^{12}. \tag{4.3}$$

The state space representation $\mathcal{P}$ of nonlinear underwater robot system can be written in the following form:

$$\mathcal{P} := \frac{d}{dt}\begin{pmatrix} \mathbf{x}_{dyn} \\ \mathbf{x}_{kin} \end{pmatrix} = \begin{pmatrix} \mathfrak{F}_{dyn}(\mathbf{x}_{dyn}, \mathbf{x}_{kin}) \\ \mathfrak{F}_{kin}(\mathbf{x}_{dyn}, \mathbf{x}_{kin}) \end{pmatrix} + \begin{pmatrix} \mathfrak{B}(\mathbf{x}_{dyn})\mathfrak{H}(\mathbf{x}_{dyn}, \mathbf{u}) \\ 0 \end{pmatrix} \tag{4.4}$$

Now, we want to reformulate the standard underwater robot equations of motion

$$\boldsymbol{M}\dot{\upsilon} + \boldsymbol{C}(\upsilon)\upsilon + \boldsymbol{D}(\upsilon)\upsilon + \mathbf{g}(\eta) = \tau \tag{4.5}$$

into the aforementioned form.
More precisely, we want to identify all the terms in the following equations from the robot dynamics and kinematics.

$$\mathcal{P} : \begin{cases} \dfrac{d}{dt}\mathbf{v} = \mathfrak{F}_{\mathbf{v}}(\mathbf{v}, \omega) + \mathfrak{F}_{\mathbf{v}}^{\lambda}(\Pi_i\lambda) + \mathfrak{B}_{\mathbf{v}}(\mathbf{v}, \omega)\mathfrak{H}(\mathbf{v}, \mathbf{u}) & (4.6) \\ \dfrac{d}{dt}\omega = \mathfrak{F}_{\omega}(\mathbf{v}, \omega) + \mathfrak{F}_{\omega}^{\lambda}(\Pi_i\lambda) + \mathfrak{B}_{\omega}(\mathbf{v}, \omega)\mathfrak{H}(\mathbf{v}, \mathbf{u}) & (4.7) \\ \dfrac{d}{dt}\mathbf{p} = \boldsymbol{R}(\lambda)\mathbf{v} & (4.8) \\ \dfrac{d}{dt}\lambda = \boldsymbol{Q}(\Pi_i\lambda)\omega. & (4.9) \end{cases}$$

where λ represents the Euler angles (ϕ, θ, ψ) and $\Pi_i\lambda = (\phi, \theta)^T$ denotes the two kinematic states appearing in the underwater robot dynamics since the restoring term $\mathbf{g}(\eta)$ includes the two angles. $\boldsymbol{R}(\lambda)$ is the linear velocity transformation matrix and $\boldsymbol{Q}(\Pi_i\lambda)$ is the angular velocity transformation matrix. The dynamic equation 4.5 can be rewritten as

$$\dot{\upsilon} = -\boldsymbol{M}^{-1}\boldsymbol{C}(\upsilon)\upsilon - \boldsymbol{M}^{-1}\boldsymbol{D}(\upsilon)\upsilon - \boldsymbol{M}^{-1}\boldsymbol{g}(\eta) + \boldsymbol{M}^{-1}\tau. \tag{4.10}$$

Note that the inertia matrix $\boldsymbol{M}$ is the sum of the rigid body inertia matrix $\boldsymbol{M}_{RB}$ and the added mass inertia matrix $\boldsymbol{M}_A$, the Coriolis matrix $\boldsymbol{C}(\upsilon)$ includes the rigid body Coriolis matrix $\boldsymbol{C}_{RB}(\upsilon)$ and the added mass Coriolis matrix $\boldsymbol{C}_A$. We assign the first three rows of the rigid body added mass $\boldsymbol{C}_{RB}(\mathbf{v},\omega)$ matrix into sub-matrices $\boldsymbol{C}_{RB,\mathbf{v}}(\mathbf{v},\omega)$ and the last three rows into sub-matrices $\boldsymbol{C}_{RB,\omega}(\mathbf{v},\omega)$:

$$\boldsymbol{C}_{RB,\mathbf{v}}(\mathbf{v},\omega) = \begin{pmatrix} 0 & 0 & 0 & m(y_g q + z_g r) & -m(x_g q - w) & -m(x_g r + v) \\ 0 & 0 & 0 & -m(y_g p + w) & m(z_g r + x_g p) & -m(y_g r - u) \\ 0 & 0 & 0 & -m(z_g p - v) & -m(z_g q + u) & m(x_g p + y_g q) \end{pmatrix} \tag{4.11}$$

and

$$\boldsymbol{C}_{RB,\omega}(\mathbf{v},\omega) = \begin{pmatrix} -m(y_g + z_g r) & m(y_g p + w) & m(z_g p - v) & 0 & -I_{yz}q - I_{xz}p + I_z r & I_{yz}r + I_{xy}p - I_y q \\ m(x_g q - w) & -m(z_g r + x_g p) & m(z_g q + u) & I_{yz}q + I_{xz}p - I_z r & 0 & -I_{xz}r - I_{xy}q + I_x p \\ m(x_g r + v) & m(y_g r - u) & -m(x_g p + y_g q) & -I_{yz}r - I_{xy}p + I_y q & I_{xz}r + I_{xy}q - I_x p & 0 \end{pmatrix}. \tag{4.12}$$

Similarly, the added mass Coriolis matrix can be also divided into two parts $\boldsymbol{C}_{A,\mathbf{v}}(\mathbf{v},\omega)$ and $\boldsymbol{C}_{A,\omega}(\mathbf{v},\omega)$, where

$$\boldsymbol{C}_{A,\mathbf{v}}(\mathbf{v},\omega) = \begin{pmatrix} 0 & 0 & 0 & 0 & -Z_{\dot w}w & Y_{\dot v}v \\ 0 & 0 & 0 & Z_{\dot w}w & 0 & -X_{\dot u}u \\ 0 & 0 & 0 & -Y_{\dot v}v & X_{\dot u}u & 0 \end{pmatrix} \tag{4.13}$$

and

$$\boldsymbol{C}_{A,\omega}(\mathbf{v},\omega) = \begin{pmatrix} 0 & -Z_{\dot w}w & Y_{\dot v}v & 0 & -N_{\dot r}\dot r & M_{\dot q}q \\ Z_{\dot w}w & 0 & -X_{\dot u}u & N_{\dot r}r & 0 & -K_{\dot p}p \\ -Y_{\dot v}v & X_{\dot u}u & 0 & -M_{\dot q}q & K_{\dot p}p & 0 \end{pmatrix}. \tag{4.14}$$

For the drag matrix, we perform the same separation:

$$\boldsymbol{D}_{\mathbf{v}} = \begin{pmatrix} -X_{u|u|}|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & -Y_{v|v|}|v| & 0 & 0 & 0 & 0 \\ 0 & 0 & -Z_{w|w|}|w| & 0 & 0 & 0 \end{pmatrix} \tag{4.15}$$

$$\boldsymbol{D}_{\omega} = \begin{pmatrix} 0 & 0 & 0 & -K_{p|p|}|p| & & \\ 0 & 0 & 0 & 0 & -M_{q|q|}|q| & \\ 0 & 0 & 0 & 0 & 0 & -N_{r|r|}|r| \end{pmatrix}. \tag{4.16}$$

The restoring term

$$\mathbf{g}(\eta) = \begin{pmatrix} (W - B)\sin(\theta) \\ -(W - B)\cos(\theta)\sin(\phi) \\ -(W - B)\cos(\theta)\cos(\phi) \\ -(y_g W - y_b B)\cos(\theta)\cos(\phi) + (z_g W - z_b B)\cos(\theta)\sin(\phi) \\ (z_g W - z_b B)\sin(\theta) + (x_g W - x_b B)\cos(\theta)\cos(\phi) \\ -(x_g W - x_b B)\sin(\theta)\cos(\phi) - (y_g W - y_b B)\sin(\theta) \end{pmatrix} \tag{4.17}$$

can also be separated into the following two terms

$$\mathbf{g}_{\mathbf{v}}(\eta) = \begin{pmatrix} (W - B)\sin(\theta) \\ -(W - B)\cos(\theta)\sin(\phi) \\ -(W - B)\cos(\theta)\cos(\phi) \end{pmatrix} \tag{4.18}$$

$$\mathbf{g}_{\omega}(\eta) = \begin{pmatrix} -(y_g W - y_b B)\cos(\theta)\cos(\phi) + (z_g W - z_b B)\cos(\theta)\sin(\phi) \\ (z_g W - z_b B)\sin(\theta) + (x_g W - x_b B)\cos(\theta)\cos(\phi) \\ -(x_g W - x_b B)\sin(\theta)\cos(\phi) - (y_g W - y_b B)\sin(\theta) \end{pmatrix}, \tag{4.19}$$

where W is the total weight of the robot, and B denotes the total buoyancy. $\mathbf{r}_g = (x_g, y_g, z_g)^T$ is the center of gravity vector and $\mathbf{r}_b = (x_b, y_b, z_b)^T$ is the center of buoyancy vector. After reordering the Coriolis matrix, the damping matrix and the restoring matrix and comparing them with 4.6 and 4.7, we can obtain the following equations:

$$\mathfrak{F}_{\mathbf{v}}(\mathbf{v}, \omega) = -\boldsymbol{M}^{-1}(\boldsymbol{C}_{RB,\mathbf{v}} + \boldsymbol{C}_{A,\mathbf{v}} + \boldsymbol{D}_{\mathbf{v}}) \tag{4.20}$$

$$\mathfrak{F}_{\omega}(\mathbf{v}, \omega) = -\boldsymbol{M}^{-1}(\boldsymbol{C}_{RB,\omega} + \boldsymbol{C}_{A,\omega} + \boldsymbol{D}_{\omega}) \tag{4.21}$$

$$\mathfrak{F}_{\omega}^{\lambda}(\Pi_i \lambda) = -\boldsymbol{M}^{-1}\mathbf{g}_{\omega}(\eta) \tag{4.22}$$

$$\mathfrak{F}_{\mathbf{v}}^{\lambda}(\Pi_i \lambda) = -\boldsymbol{M}^{-1}\mathbf{g}_{\mathbf{v}}(\eta). \tag{4.23}$$

Now, let us come to the input part, $\mathbf{u}$ is the control input generated by fins and thrusters. Suppose we have n_t thrusters and n_f fins, then it can be written as

$$\mathbf{u} = (u_{T,1} \cdots u_{T,n_t}\ \delta_{F,1} \cdots \delta_{F,n_f})^T \in \mathbb{R}^{(n_t+n_f)\times 1}. \tag{4.24}$$

For each input there is a corresponding input mapping vector and we can formulate the input matrix as $\boldsymbol{B}_a = (\boldsymbol{B}_{T,1} \cdots \boldsymbol{B}_{T,n_t}\ \boldsymbol{B}_{F,1} \cdots \boldsymbol{B}_{F,n_f}) \in \mathbb{R}^{6\times(n_t+n_f)}$. The first three rows of the matrix $\boldsymbol{B}_a$ are denoted by $\boldsymbol{B}_{a,f} \in \mathbb{R}^{3\times(n_t+n_f)}$ and the last three rows of the matrix are represented by $\boldsymbol{B}_{a,m} \in \mathbb{R}^{3\times(n_t+n_f)}$. The generalized force generated by the actuators can be represented as $\tau = \boldsymbol{B}_a\mathbf{u}$. Separately written, $\mathbf{f} = \boldsymbol{B}_{a,f}\mathbf{u}$ and $\mathbf{m} = \boldsymbol{B}_{a,m}\mathbf{u}$, where $\mathbf{f}$ is the force vector along x-, y- and z- axes and $\mathbf{m}$ is the moment vector about x-, y- and z-axes.
Hence, for our modeling methods,

$$\mathfrak{B}_{\mathbf{v}}(\mathbf{v}, \omega)\mathfrak{H}(\mathbf{v}, \mathbf{u}) = \boldsymbol{M}^{-1}\boldsymbol{B}_{a,f}\mathbf{u}, \tag{4.25}$$

$$\mathfrak{B}_{\omega}(\mathbf{v}, \omega)\mathfrak{H}(\mathbf{v}, \mathbf{u}) = \boldsymbol{M}^{-1}\boldsymbol{B}_{a,m}\mathbf{u}. \tag{4.26}$$

This finishes the transformation of the robot dynamics into standard state space form.

$$\begin{tikzcd}
& \{b\} \arrow[dl, "\boldsymbol{R}"'] \arrow[dr, "\boldsymbol{R}_E"] & \\
\{i\} & & \{\mathcal{T}\} \arrow[ll, "\boldsymbol{R}_{\mathcal{T}}"']
\end{tikzcd}$$

Figure 4.1: Frame relationship

4.1 Nonlinear Transformation and Error Dynamics

Figure 4.1 shows the relationship between different frames, the $\{i\}$ denotes the inertial world frame, and $\boldsymbol{R}$ represents the coordinate transformation from $\{i\}$ to the real robot frame $\{b\}$. $\{\mathcal{T}\}$ denotes the Frenet-Serret frame along the trim trajectory, ideally the robot body frame $\{b\}$ should coincide with the the frame $\{\mathcal{T}\}$ for a perfect tracking, but it is not necessarily the case in reality. Thus, we use $\boldsymbol{R}_E$ to denote the transformation matrix from the body frame $\{b\}$ to the trim frame $\{\mathcal{T}\}$ and $\boldsymbol{R}_{\mathcal{T}}$ to denote the transformation matrix from the trim frame $\{\mathcal{T}\}$ to the inertial world frame $\{i\}$. Based on these definitions, we have the following relationship:

$$\boldsymbol{R}_E = \boldsymbol{R}_{\mathcal{T}}^{-1}\boldsymbol{R}. \tag{4.27}$$

The desired states can be calculated from the predefined reference trim trajectories. Now we want to formulate the error dynamics. The velocities errors are defined intuitively as the difference between the desired velocity and the real velocity:

$$\mathbf{v}_E = \mathbf{v} - \mathbf{v}_{\mathcal{T}}, \tag{4.28}$$

$$\omega_E = \omega - \omega_{\mathcal{T}}. \tag{4.29}$$

The position and orientation are defined in the inertial world frame $\{i\}$. We want to observe all the errors in the body frame $\{b\}$

$$\mathbf{p}_E = \boldsymbol{R}^{-1}(\mathbf{p} - \mathbf{p}_{\mathcal{T}}) \tag{4.30}$$

$$\lambda_E = \boldsymbol{Q}^{-1}(\lambda - \lambda_{\mathcal{T}}) \tag{4.31}$$

Since the desired velocities $\upsilon_{\mathcal{T}}$ and $\omega_{\mathcal{T}}$ are constant along the trim trajectory, then we can obtain the following relationships for the derivative of velocity errors:

$$\frac{d}{dt}\mathbf{v}_E = \frac{d}{dt}\mathbf{v} - \frac{d}{dt}\mathbf{v}_{\mathcal{T}} = \frac{d}{dt}\mathbf{v} \tag{4.32}$$

$$\frac{d}{dt}\omega_E = \frac{d}{dt}\omega - \frac{d}{dt}\omega_{\mathcal{T}} = \frac{d}{dt}\omega. \tag{4.33}$$

In the following, we study the error kinematics. We start from the nonlinear position transformation equation.

$$\begin{aligned}\mathbf{p}_E &= \boldsymbol{R}^{-1}(\mathbf{p} - \mathbf{p}_{\mathcal{T}}) \Rightarrow \\ \mathbf{p}_{\mathcal{T}} &= \mathbf{p} - \boldsymbol{R}\mathbf{p}_E.\end{aligned} \tag{4.34}$$

Taking derivative of both sides, we obtain

$$\frac{d}{dt}\mathbf{p}_{\mathcal{T}} = \frac{d}{dt}\mathbf{p} - (\frac{d}{dt}\boldsymbol{R})\mathbf{p}_E - \boldsymbol{R}(\frac{d}{dt}\mathbf{p}_E). \tag{4.35}$$

Since $\frac{d}{dt}\boldsymbol{R}(\lambda) = \boldsymbol{R}(\lambda)\boldsymbol{S}(\omega)$, we have the following equation:

$$\frac{d}{dt}\mathbf{p}_{\mathcal{T}} = \frac{d}{dt}\mathbf{p} - \boldsymbol{R}\boldsymbol{S}(\omega)\mathbf{p}_E - \boldsymbol{R}(\frac{d}{dt}\mathbf{p}_E). \tag{4.36}$$

Reforming it, we get

$$\boldsymbol{R}_{\mathcal{T}}\mathbf{v}_{\mathcal{T}} = \boldsymbol{R}\mathbf{v} - \boldsymbol{R}\boldsymbol{S}(\omega)\mathbf{p}_E - \boldsymbol{R}\frac{d}{dt}\mathbf{p}_E. \tag{4.37}$$

Premultiplying both sides by $\boldsymbol{R}^{-1}$ and using $\boldsymbol{R}_E^{-1} = \boldsymbol{R}^{-1}\boldsymbol{R}_{\mathcal{T}}$

$$\frac{d}{dt}\mathbf{p}_E = \mathbf{v} - \boldsymbol{R}_E^{-1}\mathbf{v}_{\mathcal{T}} - \boldsymbol{S}(\omega)\mathbf{p}_E \tag{4.38}$$

For the orientation error, we take the first derivative of the orientation transformation equation

$$\begin{aligned}&\lambda_E = \boldsymbol{Q}^{-1}(\lambda - \lambda_{\mathcal{T}}) \\ \Rightarrow &\frac{d}{dt}\lambda_E = \boldsymbol{Q}^{-1}(\frac{d}{dt}\lambda - \frac{d}{dt}\lambda_{\mathcal{T}}) + (\frac{d}{dt}\boldsymbol{Q}^{-1})(\lambda - \lambda_{\mathcal{T}}).\end{aligned} \tag{4.39}$$

Since

$$\frac{d}{dt}\lambda = \boldsymbol{Q}\omega \tag{4.40}$$

and

$$\frac{d}{dt}\lambda_{\mathcal{T}} = \boldsymbol{Q}_{\mathcal{T}}\omega_{\mathcal{T}}, \tag{4.41}$$

we obtain

$$\begin{aligned}\frac{d}{dt}\lambda_E &= \boldsymbol{Q}^{-1}\boldsymbol{Q}\omega - \boldsymbol{Q}^{-1}\boldsymbol{Q}_{\mathcal{T}}\omega_{\mathcal{T}} + (\frac{d}{dt}\boldsymbol{Q}^{-1})(\lambda - \lambda_{\mathcal{T}}) \Rightarrow \\ \frac{d}{dt}\lambda_E &= \omega - \boldsymbol{Q}^{-1}\boldsymbol{Q}_{\mathcal{T}}\omega_{\mathcal{T}} + (\frac{d}{dt}\boldsymbol{Q}^{-1})\boldsymbol{Q}\lambda_E\end{aligned} \tag{4.42}$$

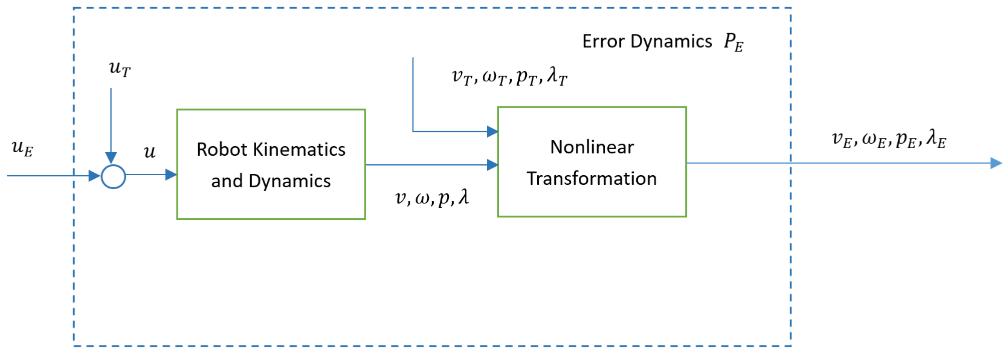

Error Dynamics P_E
u_T
$v_T, \omega_T, p_T, \lambda_T$
u_E
u
Robot Kinematics and Dynamics
Nonlinear Transformation
v, ω, p, λ
$v_E, \omega_E, p_E, \lambda_E$

Figure 4.2: Nonliear transformation and error dynamics formulation

To sum up, the nonlinear error dynamic model $\mathcal{P}_E$ for a given trim trajectory set is formulated as follows:

$$\mathcal{P}_E : \begin{cases} \frac{d}{dt}\mathbf{v}_E = \mathfrak{F}_{\mathbf{v}}(\mathbf{v}, \omega) + \mathfrak{F}^{\lambda}_{\mathbf{v}}(\Pi_i \lambda) + \mathfrak{B}_{\mathbf{v}}(\mathbf{v}, \omega)\mathfrak{H}(\mathbf{v}, \mathbf{u}) & (4.43) \\ \frac{d}{dt}\omega_E = \mathfrak{F}_{\omega}(\mathbf{v}, \omega) + \mathfrak{F}^{\lambda}_{\omega}(\Pi_i \lambda) + \mathfrak{B}_{\omega}(\mathbf{v}, \omega)\mathfrak{H}(\mathbf{v}, \mathbf{u}) & (4.44) \\ \frac{d}{dt}\mathbf{p}_E = \mathbf{v} - \boldsymbol{R}_E^{-1}\mathbf{v}_{\mathcal{T}} - \boldsymbol{S}(\omega_{\mathcal{T}})\mathbf{p}_E & (4.45) \\ \frac{d}{dt}\lambda_E = \omega - \boldsymbol{Q}^{-1}\boldsymbol{Q}_{\mathcal{T}}\omega_{\mathcal{T}} + (\frac{d}{dt}\boldsymbol{Q}^{-1})\boldsymbol{Q}\lambda_E. & (4.46) \end{cases}$$

We define the the error dynamics input as

$$\mathbf{u}_E = \mathbf{u} - \mathbf{u}_{\mathcal{T}}, \tag{4.47}$$

where the $\mathbf{u}$ is the control input for the nonlinear robot dynamics, and $\mathbf{u}_{\mathcal{T}}$ is the desired trim input. Since for the trim trajectory, the desired trim velocities, the desired trim roll angle and the desired trim pitch angle are constant meaning that $\dot{\upsilon}_{\mathcal{T}} = 0$ and the restoring term $\mathbf{g}(\eta)$ depends only on $\Pi_i \lambda$, we obtain

$$\begin{aligned} \tau_{\mathcal{T}} &= \boldsymbol{M}\dot{\upsilon}_{\mathcal{T}} + \boldsymbol{C}(\upsilon_{\mathcal{T}})\upsilon_{\mathcal{T}} + \boldsymbol{D}(\upsilon_{\mathcal{T}})\upsilon_{\mathcal{T}} + \mathbf{g}(\eta_{\mathcal{T}}) \\ &= \boldsymbol{C}(\upsilon_{\mathcal{T}})\upsilon_{\mathcal{T}} + \boldsymbol{D}(\upsilon_{\mathcal{T}})\upsilon_{\mathcal{T}} + \mathbf{g}(\Pi_i \lambda_{\mathcal{T}}), \end{aligned} \tag{4.48}$$

and

$$\mathbf{u}_{\mathcal{T}} = \boldsymbol{B}_a^{-1}\tau_{\mathcal{T}} \tag{4.49}$$

The nonlinear transformation procedure is depicted in Figure 4.2.

4.2 Linearization of Error Dynamics

In the previous section, we have derived the error dynamic model for a given trim trajectory segment. In this section, we want to prove that the linearization for a specific trim trajectory segment $\eta_{\mathcal{T}}$ is unique.
Firstly, we linearize the velocities error and get the following equations:

$$\begin{cases} \frac{d}{dt}\delta\mathbf{v}_E = \boldsymbol{A}_{\mathbf{v}}^{\mathbf{v}}\delta\mathbf{v}_E + \boldsymbol{A}_{\mathbf{v}}^{\omega}\delta\omega_E + \boldsymbol{A}_{\mathbf{v}}^{\lambda}\delta\lambda_E + \boldsymbol{B}_{\mathbf{v}}\delta\mathbf{u} & (4.50) \\ \frac{d}{dt}\delta\omega_E = \boldsymbol{A}_{\omega}^{\mathbf{v}}\delta\mathbf{v}_E + \boldsymbol{A}_{\omega}^{\omega}\delta\omega_E + \boldsymbol{A}_{\omega}^{\lambda}\delta\lambda_E + \boldsymbol{B}_{\omega}\delta\mathbf{u} & (4.51) \end{cases}$$

where

$$\begin{cases} \boldsymbol{A}_{\mathbf{v}}^{\mathbf{v}} = \frac{\partial}{\partial\mathbf{v}}[\mathfrak{F}_{\mathbf{v}}(\mathbf{v},\omega) + \mathfrak{B}_{\mathbf{v}}(\mathbf{v},\omega)\mathfrak{H}(\mathbf{v},\mathbf{u})] & (4.52) \\ \boldsymbol{A}_{\mathbf{v}}^{\omega} = \frac{\partial}{\partial\omega}[\mathfrak{F}_{v}(\mathbf{v},\omega) + \mathfrak{B}_{\mathbf{v}}(\mathbf{v},\omega)\mathfrak{H}(\mathbf{v},\mathbf{u})] & (4.53) \\ \boldsymbol{A}_{\mathbf{v}}^{\lambda} = \frac{\partial}{\partial\lambda}\mathfrak{F}_{\mathbf{v}}^{\lambda}(\Pi_i\lambda) & (4.54) \\ \boldsymbol{B}_{v} = \frac{\partial}{\partial\mathbf{u}}\mathfrak{B}_{\mathbf{v}}(\mathbf{v},\omega)\mathfrak{H}(\mathbf{v},\mathbf{u}) & (4.55) \end{cases}$$

and

$$\begin{cases} \boldsymbol{A}_{\omega}^{\mathbf{v}} = \frac{\partial}{\partial\mathbf{v}}[\mathfrak{F}_{\omega}(\mathbf{v},\omega) + \mathfrak{B}_{\omega}(\mathbf{v},\omega)\mathfrak{H}(\mathbf{v},\mathbf{u})] & (4.56) \\ \boldsymbol{A}_{\omega}^{\omega} = \frac{\partial}{\partial\omega}[\mathfrak{F}_{\omega}(\mathbf{v},\omega) + \mathfrak{B}_{\omega}(\mathbf{v},\omega)\mathfrak{H}(\mathbf{v},\mathbf{u})] & (4.57) \\ \boldsymbol{A}_{\omega}^{\lambda} = \frac{\partial}{\partial\lambda}\mathfrak{F}_{\omega}^{\lambda}(\Pi_i\lambda) & (4.58) \\ \boldsymbol{B}_{\omega} = \frac{\partial}{\partial\mathbf{u}}\mathfrak{B}_{\omega}(\mathbf{v},\omega)\mathfrak{H}(\mathbf{v},\mathbf{u}). & (4.59) \end{cases}$$

Next, we want to show that the linearization of the position error $\mathbf{p}_E$ is time-invariant. We define

$$\mathbf{f}_{\mathbf{p}_E} = \mathbf{v} - \boldsymbol{R}_E^{-1}\mathbf{v}_{\mathcal{T}} - \boldsymbol{S}(\omega_{\mathcal{T}})\mathbf{p}_E \tag{4.60}$$

$$\frac{d}{dt}\delta\mathbf{p}_E = \underbrace{\frac{\partial\mathbf{f}_{\mathbf{p}_E}}{\partial\mathbf{v}_E}}_{(1)}\cdot\delta\mathbf{v}_E + \underbrace{\frac{\partial\mathbf{f}_{\mathbf{p}_E}}{\partial\omega_E}}_{(2)}\cdot\delta\omega_E + \underbrace{\frac{\partial\mathbf{f}_{\mathbf{p}_E}}{\partial\mathbf{p}_E}}_{(3)}\cdot\delta\mathbf{p}_E + \underbrace{\frac{\partial\mathbf{f}_{\mathbf{p}_E}}{\partial\lambda_E}}_{(4)}\cdot\delta\lambda_E \tag{4.61}$$

We calculate these four terms separately. Since $\mathbf{v}_E = \mathbf{v} - \mathbf{v}_{\mathcal{T}}$ and $\mathbf{v}_{\mathcal{T}}$ is a constant, we have

$$(1): \frac{\partial\mathbf{f}_{\mathbf{p}_E}}{\partial\mathbf{v}_E} = \boldsymbol{I}_{3\times1}. \tag{4.62}$$

$$(2): \frac{\partial \mathbf{f}_{\mathbf{p}_E}}{\partial \omega_E} = \boldsymbol{0}_{3\times1}. \tag{4.63}$$

$$(3): \frac{\partial \mathbf{f}_{\mathbf{p}_E}}{\partial \mathbf{p}_E} = -\frac{\partial}{\partial \mathbf{p}_E}(\boldsymbol{S}(\omega_{\mathcal{T}})\mathbf{p}_E)\Big|_{\eta=\eta_{\mathcal{T}}} = -\boldsymbol{S}(\omega_{\mathcal{T}}). \tag{4.64}$$

$$\begin{aligned}(4): \frac{\partial \mathbf{f}_{\mathbf{p}_E}}{\partial \lambda_E} &= -\frac{\partial}{\partial \lambda}(\boldsymbol{R}_E^{-1}\mathbf{v}_{\mathcal{T}})\frac{\partial \lambda}{\partial \lambda_E}\Big|_{\eta=\eta_{\mathcal{T}}} \\ &= -\frac{\partial}{\partial \lambda}(\boldsymbol{R}^{-1}\boldsymbol{R}_{\mathcal{T}}\mathbf{v}_{\mathcal{T}})\frac{\partial \lambda}{\partial \lambda_E}\Big|_{\eta=\eta_{\mathcal{T}}} \\ &= \boldsymbol{S}(\boldsymbol{R}^{-1}\boldsymbol{R}_{\mathcal{T}}\mathbf{v}_{\mathcal{T}})\boldsymbol{Q}^{-1}\frac{\partial \lambda}{\partial \lambda_E}\Big|_{\eta=\eta_{\mathcal{T}}}\end{aligned} \tag{4.65}$$

When the robot tracks the trim trajectory perfectly, we have the relationship:

$$\boldsymbol{R}_{\mathcal{T}} = \boldsymbol{R}. \tag{4.66}$$

Also we have

$$\frac{\partial \lambda}{\partial \lambda_E} = \frac{\partial(\boldsymbol{Q}\cdot\lambda_E + \lambda_C)}{\partial \lambda_E} = \boldsymbol{Q}. \tag{4.67}$$

Combining 4.65, 4.66 and 4.67, we obtain

$$(4): \frac{\partial \mathbf{f}_{\mathbf{p}_E}}{\partial \lambda_E} = \boldsymbol{S}(\mathbf{v}_{\mathcal{T}}). \tag{4.68}$$

Taking 4.62, 4.63, 4.64 and 4.68 into 4.61, we obtain

$$\frac{d}{dt}\delta\mathbf{p}_E = \delta\mathbf{v}_E - \boldsymbol{S}(\omega_{\mathcal{T}})\delta\mathbf{p}_E - \boldsymbol{S}(\mathbf{v}_{\mathcal{T}})\delta\lambda_E. \tag{4.69}$$

For the orientation dynamics, we linearize in the same way. We define

$$\mathbf{f}_{\lambda_E} = \omega - \boldsymbol{Q}^{-1}\boldsymbol{Q}_{\mathcal{T}}\omega_{\mathcal{T}} + (\frac{d}{dt}\boldsymbol{Q}^{-1})\boldsymbol{Q}\lambda_E \tag{4.70}$$

$$\frac{d}{dt}\delta\lambda_E = \underbrace{\frac{\partial \mathbf{f}_{\lambda_E}}{\partial \mathbf{v}_E}}_{(5)}\cdot\delta\mathbf{v}_E + \underbrace{\frac{\partial \mathbf{f}_{\lambda_E}}{\partial \omega_E}}_{(6)}\cdot\delta\omega_E + \underbrace{\frac{\partial \mathbf{f}_{\lambda_E}}{\partial \mathbf{p}_E}}_{(7)}\cdot\delta\mathbf{p}_E + \underbrace{\frac{\partial \mathbf{f}_{\lambda_E}}{\partial \lambda_E}}_{(8)}\cdot\delta\lambda_E \tag{4.71}$$

$\mathbf{f}_{\lambda_E}$ has nothing to do with $\mathbf{v}_E$ and ω_E, thus

$$(5): \frac{\partial \mathbf{f}_{\lambda_E}}{\partial \mathbf{v}_E} = \mathbf{0}_{3\times1}, \tag{4.72}$$

$$(7) : \frac{\partial \mathbf{f}_{\lambda_E}}{\partial \lambda_E} = \mathbf{0}_{3\times1}. \tag{4.73}$$

For angular velocity, we have

$$(6) : \frac{\partial \mathbf{f}_{\lambda_E}}{\partial \omega_E} = \frac{\partial \omega}{\partial \omega_E} = \frac{\partial(\omega_E + \omega_{\mathcal{T}})}{\partial \omega} = \boldsymbol{I}_{3\times3}. \tag{4.74}$$

Along the trim trajectory $\eta_{\mathcal{T}}$, the angular velocity transformation matrix $\boldsymbol{Q}^{-1} = \boldsymbol{Q}_{\mathcal{T}}^{-1}$, which is a constant matrix, thus

$$\frac{d}{dt}(\boldsymbol{Q}^{-1})\Big|_{\eta=\eta_{\mathcal{T}}} = \frac{d}{dt}(\boldsymbol{Q}_{\mathcal{T}}^{-1}) = \mathbf{0}_{3\times3} \tag{4.75}$$

and

$$(\frac{d}{dt}\boldsymbol{Q}^{-1})\boldsymbol{Q}\lambda_E\Big|_{\eta=\eta_{\mathcal{T}}} = \mathbf{0}_{3\times1} \tag{4.76}$$

Then, similar to the method for calculating the term (4) 4.65, we have the following transformations:

$$\begin{aligned}(8) : \frac{\partial \mathbf{f}_{\lambda_E}}{\partial \lambda_E} &= -\frac{\partial}{\partial \lambda}(\boldsymbol{Q}^{-1}\boldsymbol{Q}_{\mathcal{T}}\omega_{\mathcal{T}})\frac{\partial \lambda}{\partial \lambda_E}\Big|_{\eta=\eta_{\mathcal{T}}} \\ &= \boldsymbol{S}(\boldsymbol{Q}^{-1}\boldsymbol{Q}_{\mathcal{T}}\omega_{\mathcal{T}})\boldsymbol{Q}^{-1}\frac{\partial \lambda}{\partial \lambda_E}\Big|_{\eta=\eta_{\mathcal{T}}} \\ &= \boldsymbol{S}(\boldsymbol{Q}^{-1}\boldsymbol{Q}_{\mathcal{T}}\omega_{\mathcal{T}})\boldsymbol{Q}^{-1}\boldsymbol{Q}\Big|_{\eta=\eta_{\mathcal{T}}} \\ &= \boldsymbol{S}(\boldsymbol{Q}^{-1}\boldsymbol{Q}_{\mathcal{T}}\omega_{\mathcal{T}})\Big|_{\eta=\eta_{\mathcal{T}}} \\ &= \boldsymbol{S}(\omega_{\mathcal{T}}). \end{aligned} \tag{4.77}$$

The orientation error dynamics can be summarized from 4.72, 4.73, 4.74, 4.77 as:

$$\frac{d}{dt}\delta\lambda_E = \delta\omega_E - \boldsymbol{S}(\omega_{\mathcal{T}})\delta\lambda_E \tag{4.78}$$

To sum up, the linearized error dynamics is

$$\mathcal{P}_{E,L} : \begin{cases} \frac{d}{dt}\delta\mathbf{x}_{dyn_E} = \boldsymbol{A}_d^d\delta\mathbf{x}_{dyn_E} + \boldsymbol{A}_k^d\delta\mathbf{x}_{kin_E} + \boldsymbol{B}\mathbf{u} & (4.79) \\ \frac{d}{dt}\delta\mathbf{x}_{kin_E} = \delta\mathbf{x}_{dyn_E} + \boldsymbol{A}_k^k\delta\mathbf{x}_{kin_E} & (4.80) \end{cases}$$

where

$$\delta\mathbf{x}_{dyn_E} = \begin{pmatrix}\delta\mathbf{v}_E \\ \delta\omega_E\end{pmatrix} \in \mathbb{R}^6 \tag{4.81}$$

denotes the 6 dynamic error states (velocity errors) and

$$\delta \mathbf{x}_{kin_E} = \begin{pmatrix} \delta \mathbf{p}_E \\ \delta \lambda_E \end{pmatrix} \in \mathbb{R}^6 \tag{4.82}$$

The error dynamics system matrix $\boldsymbol{A}_d^d$ is defined as

$$\boldsymbol{A}_d^d = \begin{pmatrix} \boldsymbol{A}_{\mathbf{v}}^{\mathbf{v}} & \boldsymbol{A}_{\mathbf{v}}^{\omega} \\ \boldsymbol{A}_{\omega}^{\mathbf{v}} & \boldsymbol{A}_{\omega}^{\omega} \end{pmatrix} \in \mathbb{R}^{6\times 6} \tag{4.83}$$

indicating the mutual interaction between the linear velocities and angular velocities. The error dynamics system matrix $\boldsymbol{k}_d^d$ is defined as

$$\boldsymbol{A}_k^d = \begin{pmatrix} 0 & \boldsymbol{A}_{\mathbf{v}}^{\lambda} \\ 0 & \boldsymbol{A}_{\omega}^{\lambda} \end{pmatrix} \in \mathbb{R}^{6\times 6} \tag{4.84}$$

representing the influence of the kinematic states on the dynamics states. It is noticeable that only the orientation states (Euler angles) λ_E have impact on the velocity errors $\mathbf{v}_E$ and ω_E. The robot dynamics is not dependent on its position. It can be identified from the robot dynamic equation 5.1 and the restoring term definition 4.17 that the robot dynamics is influenced by the roll angle ϕ and pitch angle θ due to the restoring part. The kinematic error system matrix $\boldsymbol{A}_k^k$ is defined as

$$\boldsymbol{A}_k^k = \begin{pmatrix} -\boldsymbol{S}(\omega_{\mathcal{T}}) & -\boldsymbol{S}(\mathbf{v}_{\mathcal{T}}) \\ 0 & -\boldsymbol{S}(\omega_{\mathcal{T}}) \end{pmatrix} \in \mathbb{R}^{6\times 6}, \tag{4.85}$$

The input matrix $\boldsymbol{B}$ is

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{B}_{\mathbf{v}} \\ \boldsymbol{B}_{\omega} \end{pmatrix} \in \mathbb{R}^{(n_t+n_f)\times 6}. \tag{4.86}$$

From the previous analysis, we come to the conclusion that the linearized error dynamic system is a bijective function of trim trajectories $\eta_{\mathcal{T}} = (\|\mathbf{v}_{\mathcal{T}}\|, \dot{\psi}_{\mathcal{T}}, \gamma_{\mathcal{T}})^T$. This property is of significance since we can adopt the methods for MIMO linear systems to handle the originally nonlinear and strongly coupled system.

A system is said to be controllable at time t_0 if the system can be transferred from any initial state $\mathbf{x}(t_0)$ to any other state in a finite interval of time by means of an unconstrained control input. For the actuator optimization problem in the next chapter, the controllability is the primal requirement of a certain actuator configuration. The error dynamics is completely state controllable if and only if the vectors $\boldsymbol{B}_E, \boldsymbol{A}_E\boldsymbol{B}_E, \ldots, \boldsymbol{A}_E^{11}\boldsymbol{B}_E$ are linearly independent, which can be also checked with the controllability matrix which is defined as

$$\boldsymbol{C} = [\boldsymbol{B}_E \quad \boldsymbol{A}_E\boldsymbol{B}_E \quad \ldots \quad \boldsymbol{A}_E^{11}\boldsymbol{B}_E]. \tag{4.87}$$

An LTI system is controllable if the controllability matrix C has full row rank. In our case, we expect the rank of the controllability matrices all trim trajectory segments to be equal to 12. That is

$$\begin{aligned} rank(\boldsymbol{C}_{E,1}) &= 12 \\ &\vdots \\ rank(\boldsymbol{C}_{E,m}) &= 12 \end{aligned} \tag{4.88}$$

4.3 Switched LQR Controller Design for Trim Trajectories

In the previous two sections, we have proved that the linearized error dynamics is unique for each trim trajectory segment. For the whole trajectory consisting of a group of trim trajectores, the error dynamics can be formulated as a switched system. Suppose we have m linearized trim trajectory error dynamics,

$$\begin{aligned} \dot{\mathbf{x}}_{E,1} &= \boldsymbol{A}_{E,1}\mathbf{x}_{E,1} + \boldsymbol{B}_{E,1}\mathbf{u}_{E,1} \\ \dot{\mathbf{x}}_{E,2} &= \boldsymbol{A}_{E,2}\mathbf{x}_{E,2} + \boldsymbol{B}_{E,2}\mathbf{u}_{E,2} \\ &\vdots \\ \dot{\mathbf{x}}_{E,m} &= \boldsymbol{A}_{E,m}\mathbf{x}_{E,m} + \boldsymbol{B}_{E,m}\mathbf{u}_{E,m}, \end{aligned} \tag{4.89}$$

where $\mathbf{x}_{E,1} = (\mathbf{v}_{E,1}, \omega_{E,1}, \mathbf{p}_{E,1}, \lambda_{E,1})^T, \cdots, \mathbf{x}_{E,m} = (\mathbf{p}_{E,m}, \lambda_{E,m}, \mathbf{v}_{E,m}, \omega_{E,m})^T$ are error states using the nonlinear transformations 4.28, 4.29, 4.30 and 4.31. The error dynamics system matrix $\boldsymbol{A}_{E,j}$, $j = 1, \cdots, m$ is

$$\boldsymbol{A}_{E,j} = \begin{pmatrix} \boldsymbol{A}^d_{d,j} & \boldsymbol{A}^d_{k,j} \\ \boldsymbol{I}_{3\times3} & \boldsymbol{A}^d_{d,j} \end{pmatrix} \in \mathbb{R}^{12\times12} \tag{4.90}$$

and the error dynamics input matrix $\boldsymbol{B}_{E,j}$ is

$$\boldsymbol{B}_{E,j} = \begin{pmatrix} \boldsymbol{B}_{\mathbf{v},j} \\ \boldsymbol{B}_{\omega,j} \end{pmatrix} \in \mathbb{R}^{12\times(p+q)}. \tag{4.91}$$

Both of them are calculated with linearization methods in Section 4.2.
For each trim trajectory segment, we have a 12-dimensional MIMO linear system, thus we design a LQR controller with weighting matrix $Q_{E,j}$ and $\mathcal{R}_{E,j}$ to minimize the cost function for the j-th trim trajectory

$$J = \int_0^\infty [\mathbf{x}^T_{E,j} Q_{E,j} \mathbf{x}_{E,j} + \mathbf{u}^T_{E,j} \mathcal{R}_{E,j} \mathbf{u}_{E,j}] dt \tag{4.92}$$

subject to $\mathbf{x}_{E,j}(0) = \mathbf{x}_{E,j,0}$, $Q_{E,j} \geq 0$, $Q^T_{E,j} = Q_{E,j}$, $\mathcal{R}_{E,j} > 0$ and $\mathcal{R}^T_{E,j} = \mathcal{R}_{E,j}$. Assume that $(\boldsymbol{A}_{E,j}, \boldsymbol{B}_{E,j})$ is controllable, the optimal solution to minimize the cost function is $\mathbf{u}_{E,j} = -\mathcal{K}_j \mathbf{x}_{E,j}$, where $\mathcal{K}_j = \mathcal{R}^{-1}_{E,j} \boldsymbol{B}^T_{E,j} \boldsymbol{S}_{E,j}$ is a constant linear state feedbak gain.

The matrix $\boldsymbol{S}_E \geq 0$ which is the solution of the continuous Algebraic Riccati Equation (ARE):

$$\boldsymbol{Q}_{E,j} + \boldsymbol{A}_{E,j}^T \boldsymbol{S}_{E,j} + \boldsymbol{S}_{E,j}\boldsymbol{A}_{E,j} - \boldsymbol{S}_{E,j}\boldsymbol{B}_{E,j}\mathcal{R}_{E,j}^{-1}\boldsymbol{B}_{E,j}^T\boldsymbol{S}_{E,j} = 0. \tag{4.93}$$

4.4 Stability-based Selection of Weighting Matrices

After we implement the state feedback $\mathbf{u}_{E,j} = -\mathcal{K}_j\mathbf{x}_{E,j}$ to the error dynamics system $\dot{\mathbf{x}}_{E,j} = \boldsymbol{A}_{E,j}\mathbf{x}_{E,j} + \boldsymbol{B}_{E,j}\mathbf{u}_{E,j}$ for the j-th trim trajectory, we obtain the closed loop error dynamics system:

$$\begin{aligned}\dot{\mathbf{x}}_{E,j} &= \boldsymbol{A}_{E,j}\mathbf{x}_{E,j} + \boldsymbol{B}_{E,j}\mathbf{u}_{E,j}\\ &= (\boldsymbol{A}_{E,j} - \boldsymbol{B}_{E,j}\mathcal{K}_j)\cdot\mathbf{x}_{E,j}\end{aligned} \tag{4.94}$$

We want prove that each trim trajectory can be stabilized by the corresponding LQR controller. Define the Lyapunov function $V_{E,j}(\mathbf{x}_{E,j}) = \mathbf{x}_{E,j}^T\boldsymbol{S}_{E,j}\mathbf{x}_{E,j}$, $j = 1, \cdots, m$, where $\boldsymbol{S}_E$ is positive definite and symmetric, $\boldsymbol{S}_E \geq 0$, $\boldsymbol{S}_E^T = \boldsymbol{S}_E$. Differentiating it we obtain

$$\begin{aligned}\dot{V}_{E,j}(\mathbf{x}_{E,j}) &= \dot{\mathbf{x}}_{E,j}^T\boldsymbol{S}_{E,j}\mathbf{x}_{E,j} + \mathbf{x}_{E,j}^T\boldsymbol{S}_{E,j}\dot{\mathbf{x}}_{E,j}\\ &= \left[(\boldsymbol{A}_{E,j} - \boldsymbol{B}_{E,j}\mathcal{K}_{E,j})\mathbf{x}_{E,j}\right]^T\boldsymbol{S}_{E,j}\mathbf{x}_{E,j} + \mathbf{x}_{E,j}^T\boldsymbol{S}_{E,j}[(\boldsymbol{A}_{E,j} - \boldsymbol{B}_{E,j}\mathcal{K}_{E,j})\mathbf{x}_{E,j}]\\ &\overset{\mathcal{K}_{E,j}=\mathcal{R}_E^{-1}\boldsymbol{B}_{E,j}^T\boldsymbol{S}_{E,j}}{=} \mathbf{x}_{E,j}^T\left[\underbrace{(\boldsymbol{A}_{E,j} - \boldsymbol{B}_{E,j}\mathcal{R}_{E,j}^{-1}\boldsymbol{B}_{E,j}^T\boldsymbol{S}_{E,j})^T\boldsymbol{S}_{E,j}}_{(1)} + \underbrace{\boldsymbol{S}_{E,j}(\boldsymbol{A}_{E,j} - \boldsymbol{B}_{E,j}\mathcal{R}_{E,j}^{-1}\boldsymbol{B}_{E,j}^T\boldsymbol{S}_{E,j})}_{(2)}\right]\mathbf{x}_{E,j}\end{aligned} \tag{4.95}$$

$$\begin{aligned}(1) &= (\boldsymbol{A}_{E,j}^T - \boldsymbol{S}_{E,j}^T\boldsymbol{B}_{E,j}(\mathcal{R}_{E,j}^{-1})^T\boldsymbol{B}_{E,j}^T)\boldsymbol{S}_{E,j}\\ &= (\boldsymbol{A}_{E,j}^T - \boldsymbol{S}_{E,j}\boldsymbol{B}_{E,j}\mathcal{R}_{E,j}^{-1}\boldsymbol{B}_{E,j}^T)\boldsymbol{S}_{E,j}\\ &= \boldsymbol{A}_{E,j}^T\boldsymbol{S}_{E,j} - \boldsymbol{S}_{E,j}\boldsymbol{B}_{E,j}\mathcal{R}_{E,j}^{-1}\boldsymbol{B}_{E,j}^T\boldsymbol{S}_{E,j}\end{aligned} \tag{4.96}$$

$$\begin{aligned}(2) &= \boldsymbol{S}_{E,j}\boldsymbol{A}_{E,j} - \boldsymbol{S}_{E,j}\boldsymbol{B}_{E,j}\mathcal{R}_{E,j}^{-1}\boldsymbol{B}_{E,j}^T\boldsymbol{S}_{E,j}\\ &= \boldsymbol{S}_{E,j}\boldsymbol{A}_{E,j} - \boldsymbol{S}_{E,j}^T\boldsymbol{B}_{E,j}\mathcal{R}_{E,j}^{-1}\boldsymbol{B}_{E,j}^T\boldsymbol{S}_{E,j}\end{aligned} \tag{4.97}$$

$$\begin{aligned}(1)+(2) &= \boldsymbol{A}_{E,j}^T\boldsymbol{S}_{E,j} - \boldsymbol{S}_{E,j}\boldsymbol{B}_{E,j}\mathcal{R}_{E,j}^{-1}\boldsymbol{B}_{E,j}^T\boldsymbol{S}_{E,j} + \boldsymbol{S}_{E,j}\boldsymbol{A}_{E,j} - \boldsymbol{S}_{E,j}^T\boldsymbol{B}_{E,j}\mathcal{R}_{E,j}^{-1}\boldsymbol{B}_{E,j}^T\boldsymbol{S}_{E,j}\\ &\overset{ARE}{=} -\boldsymbol{Q}_{E,j} - \boldsymbol{S}_{E,j}^T\boldsymbol{B}_{E,j}\mathcal{R}_{E,j}^{-1}\boldsymbol{B}_{E,j}^T\boldsymbol{S}_{E,j}\end{aligned} \tag{4.98}$$

Since $\mathcal{R}_{E,j}$ is diagonal and positive definite, its inverse is also diagonal and positive definite, i.e., $\mathcal{R}_{E,j} \succ 0 \Rightarrow \mathcal{R}_{E,j}^{-1} \succ 0$, then $\boldsymbol{S}_{E,j}^T\boldsymbol{B}_{E,j}\mathcal{R}_{E,j}^{-1}\boldsymbol{B}_{E,j}^T\boldsymbol{S}_{E,j} \succ 0$, since $\boldsymbol{Q}_{E,j} \geq 0$, we have

$\dot{V}_{E,j}(\mathbf{x}_{E,j}) = (1) + (2) \prec 0$. The derivative of the Lyapunov function is negative definite, so the error dynamics system can be asymptotically stabilized by the LQR state feedback controller $\mathbf{u}_{E,j} = -\mathcal{K}\mathbf{x}_{E,j}$ at the equilibrium point $\mathbf{x}_{E,j} = 0$. Note that the error dynamics input $\mathbf{u}_{E,j} = \mathbf{u}_j - \mathbf{u}_{\mathcal{T}_j}$, then the input $\mathbf{u}_j$ given into the nonlinear underwater dynamic system for the j-th trim trajectory is

$$\begin{aligned} \mathbf{u}_j &= \mathbf{u}_{E,j} + \mathbf{u}_{\mathcal{T}_j} \\ &= \mathbf{u}_{\mathcal{T}_j} - \mathcal{K}\mathbf{x}_{E,j}, \end{aligned} \tag{4.99}$$

where $\mathbf{u}_{\mathcal{T}_j}$ is the constant desired trim input for the j-th trim trajectory segment. The input $\mathbf{u}_j$ is composed of a constant feedforward input $\mathbf{u}_{\mathcal{T}_j}$ and a state feedback $-\mathcal{K}\mathbf{x}_{E,j}$.
When we implement all the corresponding LQR state feedback controller to all trim trajectory segments, we obtain

$$\begin{aligned} \dot{\mathbf{x}}_{E,1} &= (\boldsymbol{A}_{E,1} - \mathcal{R}_{E,1}\boldsymbol{B}_{E,1}^T\boldsymbol{S}_{E,1})\mathbf{x}_{E,1} \\ \dot{\mathbf{x}}_{E,2} &= (\boldsymbol{A}_{E,2} - \mathcal{R}_{E,2}\boldsymbol{B}_{E,2}^T\boldsymbol{S}_{E,2})\mathbf{x}_{E,2} \\ &\vdots \\ \dot{\mathbf{x}}_{E,m} &= (\boldsymbol{A}_{E,m} - \mathcal{R}_{E,m}\boldsymbol{B}_{E,m}^T\boldsymbol{S}_{E,m})\mathbf{x}_{E,m} \end{aligned} \tag{4.100}$$

If we denote the closed system matrix as

$$\begin{aligned} \bar{\boldsymbol{A}}_{E,1} &= \boldsymbol{A}_{E,1} - \mathcal{R}_{E,1}\boldsymbol{B}_{E,1}^T\boldsymbol{S}_{E,1} \\ \bar{\boldsymbol{A}}_{E,2} &= \boldsymbol{A}_{E,2} - \mathcal{R}_{E,2}\boldsymbol{B}_{E,2}^T\boldsymbol{S}_{E,2} \\ &\vdots \\ \bar{\boldsymbol{A}}_{E,m} &= \boldsymbol{A}_{E,m} - \mathcal{R}_{E,m}\boldsymbol{B}_{E,m}^T\boldsymbol{S}_{E,m} \end{aligned} \tag{4.101}$$

Then we can represent the closed loop error dynamics as follows:

$$\begin{aligned} \dot{\mathbf{x}}_{E,1} &= \bar{\boldsymbol{A}}_{E,1}\mathbf{x}_{E,1} \\ \dot{\mathbf{x}}_{E,2} &= \bar{\boldsymbol{A}}_{E,2}\mathbf{x}_{E,2} \\ &\vdots \\ \dot{\mathbf{x}}_{E,m} &= \bar{\boldsymbol{A}}_{E,m}\mathbf{x}_{E,m} \end{aligned} \tag{4.102}$$

Thus, we can define our closed-loop error dynamics as a switched system

$$\Sigma_{S,\bar{A}_E} : \dot{\mathbf{x}}_E(t) = \bar{\boldsymbol{A}}_E(t)\mathbf{x}_E(t) \tag{4.103}$$

where $\bar{\boldsymbol{A}}_E(t) \in \{\bar{\boldsymbol{A}}_{E,1}, \ldots, \bar{\boldsymbol{A}}_{E,m}\}$.
LQR controller can ensure the stability of each trim trajectory segment, however it can not ensure the stability of the whole trajectory. The stability for arbitrary switching can be proven by means of common quadratic lyapunov functions (CQLFs) ([SWM+07]). Recall that $V_{E,j}(\mathbf{x}_{E,j}) = \mathbf{x}_{E,j}^T\boldsymbol{S}_{E,j}\mathbf{x}_{E,j}$ is a quadratic Lyapunov function (QLF) for the LTI

system: $\mathbf{x}_{E,j} = \bar{\boldsymbol{A}}_{E,j}(t)\mathbf{x}_{E,j}(t)$, where $\boldsymbol{S}_{E,j}$ is symmetric and positive definite satisfying $\boldsymbol{S}_{E,j}\bar{\boldsymbol{A}}_{E,j} + \bar{\boldsymbol{A}}_{E,j}^T\boldsymbol{S}_{E,j}$ is negative definite. The existence of CQLF $V(\mathbf{x}_E) = \mathbf{x}_E^T\boldsymbol{S}_E\mathbf{x}_E$ is equivalent to determine whether such a matrix $\boldsymbol{S}_E$ exists for a given group of Hurwitz matrices $\{\boldsymbol{A}_{E,1}, \boldsymbol{A}_{E,2}, \ldots, \boldsymbol{A}_{E,m}\}$ such that

$$\begin{aligned}
\bar{\boldsymbol{A}}_{E,1}^T\boldsymbol{S}_E + \boldsymbol{S}_E\bar{\boldsymbol{A}}_{E,1} &\prec 0 \\
\bar{\boldsymbol{A}}_{E,2}^T\boldsymbol{S}_E + \boldsymbol{S}_E\bar{\boldsymbol{A}}_{E,1} &\prec 0 \\
&\vdots \\
\bar{\boldsymbol{A}}_{E,m}^T\boldsymbol{S}_E + \boldsymbol{S}_E\bar{\boldsymbol{A}}_{E,m} &\prec 0,
\end{aligned} \tag{4.104}$$

This is a system of linear matrix inequalities (LMIs), it is said to be feasible if a solution $\boldsymbol{S}_E$ exists. Otherwise, the LMIs are infeasible. Thus, determining whether or not the switched system $\Sigma_{S,\bar{A}_E}$ has a CQLF amounts to checking the feasibility of a system of LMIs [SWM+07]. Converting these to the nonstrict LMIs

$$\begin{aligned}
\bar{\boldsymbol{A}}_{E,1}^T\boldsymbol{S}_E + \boldsymbol{S}_E\bar{\boldsymbol{A}}_{E,1} &\leq \boldsymbol{I} \\
\bar{\boldsymbol{A}}_{E,2}^T\boldsymbol{S}_E + \boldsymbol{S}_E\bar{\boldsymbol{A}}_{E,2} &\leq \boldsymbol{I} \\
&\vdots \\
\bar{\boldsymbol{A}}_{E,m}^T\boldsymbol{S}_E + \boldsymbol{S}_E\bar{\boldsymbol{A}}_{E,m} &\leq \boldsymbol{I}
\end{aligned} \tag{4.105}$$

$$\boldsymbol{S}_E \geq \boldsymbol{I}. \tag{4.106}$$

This can be formulated as semidefinite programming so that we can utilize the CVX [GB14, GB08] to check whether a matrix $\boldsymbol{S}_E$ satisfying the aforementioned LMIs can be found. Recall that $\bar{\boldsymbol{A}}_{E,j}, j = 1, \cdots, m$ is the closed loop system matrix after we implement the state feedback gain, $\mathbf{u}_E = -\mathcal{R}_E^{-1}\boldsymbol{B}_{E,j}\boldsymbol{S}_E$ where specified weighting matrices $\mathcal{Q}_{E,j}$ and $\mathcal{R}_{E,j}$ are user-defined and $\boldsymbol{S}_E$ is calulated from AREs. The LMIs 4.104 can be rewritten as:

$$\begin{aligned}
(\boldsymbol{A}_{E,1} - \mathcal{R}_{E,1}\boldsymbol{B}_{E,1}^T\boldsymbol{S}_E)\boldsymbol{S}_E + \boldsymbol{S}_E(\boldsymbol{A}_{E,1} - \mathcal{R}_{E,1}\boldsymbol{B}_{E,1}^T\boldsymbol{S}_E) &\prec 0 \\
(\boldsymbol{A}_{E,2} - \mathcal{R}_{E,2}\boldsymbol{B}_{E,2}^T\boldsymbol{S}_E)\boldsymbol{S}_E + \boldsymbol{S}_E(\boldsymbol{A}_{E,2} - \mathcal{R}_{E,2}\boldsymbol{B}_{E,1}^T\boldsymbol{S}_E) &\prec 0 \\
&\vdots \\
(\boldsymbol{A}_{E,m} - \mathcal{R}_{E,m}\boldsymbol{B}_{E,m}^T\boldsymbol{S}_E)\boldsymbol{S}_E + \boldsymbol{S}_E(\boldsymbol{A}_{E,m} - \mathcal{R}_{E,m}\boldsymbol{B}_{E,m}^T\boldsymbol{S}_E) &\prec 0.
\end{aligned} \tag{4.107}$$

Besides, the existence of a common $\boldsymbol{S}_E$ means all AREs of all trim trajectories possess the same solution:

$$\begin{aligned}
\mathcal{Q}_{E,1} + \boldsymbol{A}_{E,1}^T\boldsymbol{S}_E + \boldsymbol{S}_E\boldsymbol{A}_{E,1} - \boldsymbol{S}_E\boldsymbol{B}_{E,1}\mathcal{R}_{E,1}^{-1}\boldsymbol{B}_E^T\boldsymbol{S}_E &= 0 \\
&\vdots \\
\mathcal{Q}_{E,m} + \boldsymbol{A}_{E,m}^T\boldsymbol{S}_E + \boldsymbol{S}_E\boldsymbol{A}_{E,m} - \boldsymbol{S}_E\boldsymbol{B}_{E,m}\mathcal{R}_{E,m}^{-1}\boldsymbol{B}_E^T\boldsymbol{S}_E &= 0
\end{aligned} \tag{4.108}$$

Thus, there are coupling between the selections of the weighting matrices $Q_{E,1}, \cdots, Q_{E,m}$ and $\mathcal{R}_{E,1}, \cdots, \mathcal{R}_{E,m}$ and their values play an essential role for the global stability of the switched system.
Thus, the first step to design the switched LQR controller is to select the values of $\mathcal{R}_{E,1}, \cdots, \mathcal{R}_{E,m}$. Because tracking the specified trajectories is the most significant, the weighting coefficients for positions should be bigger than those of the Euler angles and velocities weighting coefficients should be smaller than both of them. Then, using m state costing matrices we could solve the LMIs 4.107 to find a common matrix $\boldsymbol{S}_E$. With the values of $\mathcal{R}_{E,1}, \cdots, \mathcal{R}_{E,m}$ and the matrix $\boldsymbol{S}_E$, the another corresponding weighting matrix $Q_{E,j}$, $j = 1, \cdots, m$ penalizing the error states $\mathbf{x}_{E,j}$ can be calculated from the corresponding ARE 4.108.

Chapter 5

Formulation of Optimization Problem

To determine the geometric configuration is the most significant part in this work. In the Chapter 2, we have established the modular dynamic model for the robot determined by several geometric decision variables. Additionally, the Coriolis terms, the drag term in the dynamics depend also on the velocities. The restoring term is influenced by the kinematics. The kinematic and dynamic requirements come from the desired trim trajectories. In conclusion, planned trim trajectory combinations $\mathcal{T}_1, \cdots, \mathcal{T}_m$ and decision variables for hull $\mathfrak{d}_H$, thrusters $\mathfrak{d}_P$ and fins $\mathfrak{d}_F$ specify the current robot dynamics. These geometric decision variables are defined as follows:

Definition 5.1 *The decision variables for the hull* $\mathfrak{d}_H$ *are defined as* $\mathfrak{d}_H = \{\, l_H, r_H \mid l_H \in \mathbb{R}^+, r_H \in \mathbb{R}^+ \,\}$*, where* l_H *denotes the length of the hull cylinder and* r_H *is the radius of the hull cylinder.*

Definition 5.2 *The decision variables for the thrusters* $\mathfrak{d}_P$ *are defined as* $\mathfrak{d}_P = \{\, b_T, \mathbf{d}_T, \mathbf{r}_T \mid b_T \in \{-1, 1\}, \mathbf{d}_T \in \mathbb{R}^3, \mathbf{r}_T \in \mathbb{R}^3 \,\}$*, where* b_T *denotes the motor spin direction,* $\mathbf{d}_T$ *is the motor orientation and* $\mathbf{r}_T$ *represents the motor position.*

Definition 5.3 *The decision variables for the fins* $\mathfrak{d}_F$ *are defined as* $\mathfrak{d}_F = \{\, x_T, \gamma_T \mid x_T \in \mathbb{R}, \gamma_T \in \{-\pi, \pi\} \,\}$*, where* x_T *denotes the x-coordinate of the fin geometric center in the body frame* $\{b\}$*,* γ_T *is the fin rotation angle.*

Let us firstly write the dynamics equation of underwater robots into the form that the inertia matrix, the Coriolis matrix, the damping matrix and the input matrix are parametrized with dynamic and kinematic state variables and the decision variables:

$$
\begin{aligned}
&\boldsymbol{M}_{RB}(\mathbf{r}_T, \mathfrak{d}_H, \mathfrak{d}_F)\dot{\upsilon} + \boldsymbol{M}_A(\mathbf{x}_{dyn,\mathcal{T}}, \mathfrak{d}_H)\dot{\upsilon} \\
&+ \boldsymbol{C}_{RB}(\mathbf{x}_{dyn,\mathcal{T}}, \mathbf{r}_T, \mathfrak{d}_H, \mathfrak{d}_F)\upsilon + \boldsymbol{C}_A(\mathbf{x}_{dyn,\mathcal{T}}, \mathfrak{d}_H)\upsilon + \boldsymbol{D}(\mathbf{x}_{dyn,\mathcal{T}}, \mathfrak{d}_H)\upsilon \\
&+ \mathbf{g}(\mathbf{x}_{kin,\mathcal{T}}, \mathfrak{d}_F, \mathbf{r}_T, \mathfrak{d}_H) \\
&= \boldsymbol{B}_T(\mathbf{d}_T, \mathbf{r}_T, b_T)\begin{pmatrix} u_{T,1} \\ \vdots \\ u_{T,n_t} \end{pmatrix} + \boldsymbol{B}_F(\mathbf{x}_{dyn,\mathcal{T}}, \mathfrak{d}_F)\begin{pmatrix} \delta_{F,1} \\ \vdots \\ \delta_{F,n_f} \end{pmatrix}.
\end{aligned}
\tag{5.1}
$$

We have already assumed that the moment of inertia is only determined by the hull, thus $\mathfrak{d}_H$ determines the $\boldsymbol{I}_g$. All components contribute to the total mass of the robot. The positions of all modules will influence the center of mass $\mathbf{r}_g^b$, see equation 2.43. Hence, according to the A.71, all geometric decision variables will influence the robot rigid body matrix.

$$\left.\begin{array}{l}\mathfrak{d}_P\\ \mathfrak{d}_H\\ \mathfrak{d}_F\end{array}\right\} \Rightarrow \boldsymbol{M}_{RB}, \tag{5.2}$$

The added mass inertia matrix is a hydrodynamic term, based on the assumptions that all the hydrodynamic coefficient are only calculated from the hull geometry and the equation 2.50. Since this term is velocity dependent, we have different desired velocities for different trim trajectory segments. It means that for each segment we have a corresponding desired added mass inertia matrix, i.e., $\boldsymbol{M}_{A,1}, \cdots, \boldsymbol{M}_{A,m}$. To sum up, we have

$$\left.\begin{array}{l}\mathfrak{d}_H\\ \mathbf{x}_{dyn,\mathcal{T}}\end{array}\right\} \Rightarrow \boldsymbol{M}_A. \tag{5.3}$$

According to the equation A.72, the rigid body Coriolis term is calculated from the mass m, the angular velocity ω and the center of mass $\mathbf{r}_g^b$. All robot components have contribution to the total mass and the center of gravity. The m trim trajectory segments give m trim angular velocities $\omega_{\mathcal{T},1}, \cdots, \omega_{\mathcal{T},m}$, and thus we also have m desired rigid body Coriolis matrices $\boldsymbol{C}_{RB,1}, \cdots, \boldsymbol{C}_{RB,m}$.

$$\left.\begin{array}{l}\mathfrak{d}_P\\ \mathfrak{d}_H\\ \mathfrak{d}_F\\ \mathbf{x}_{dyn,\mathcal{T}}\end{array}\right\} \Rightarrow \boldsymbol{C}_{RB}(\mathbf{x}_{dyn}) \tag{5.4}$$

For the hydrodynamics added mass Coriolis matrix, see 2.51, only the hull hydrodynamics and the trim trajectory segments influence it, thus we obtain

$$\left.\begin{array}{l}\mathfrak{d}_H\\ \mathbf{x}_{dyn,\mathcal{T}}\end{array}\right\} \Rightarrow \boldsymbol{C}_A(\mathbf{x}_{dyn}). \tag{5.5}$$

We have m added mass Coriolis matrices $\boldsymbol{C}_{A,1}, \cdots, \boldsymbol{C}_{A,m}$. As a hydrodynamic term, the damping is determined by the same group of variables:

$$\left.\begin{array}{l}\mathfrak{d}_H\\ \mathbf{x}_{dyn,\mathcal{T}}\end{array}\right\} \Rightarrow \boldsymbol{D}(\mathbf{x}_{dyn}). \tag{5.6}$$

The m trim trajectories give us m damping matrices $\boldsymbol{D}_1, \cdots, \boldsymbol{D}_m$ The restoring term 4.17 depends on the kinematics $\Pi_i\lambda$ from trim trajectories and the center of gravity $\mathbf{r}_g^b$, so we obtain

$$\left.\begin{array}{l}\mathfrak{d}_P\\ \mathfrak{d}_H\\ \mathfrak{d}_F\\ \mathbf{x}_{kin,\mathcal{T}}\end{array}\right\} \Rightarrow \mathbf{g}(\eta). \tag{5.7}$$

We have m restoring terms as well, that is, $\mathbf{g}_1(\eta), \cdots, \mathbf{g}_m(\eta)$. Recall that in Chapter 4, we discussed how to calculate the desired generalised force $\tau_{\mathcal{T}}$ (4.48) and the desired control inputs of actuators $\mathbf{u}_{\mathcal{T}}$ (4.49) along the trim trajectory segment $\mathcal{T}$. Besides, from the aforementioned relationships, we know for each trim trajectory segment, we have a corresponding $\boldsymbol{C}_{RB}$, $\boldsymbol{C}_A$, $\boldsymbol{D}$ and $\mathbf{g}(\eta)$. Thus, we can parametrize the desired inputs with the decision variables and the trim specifications as follows:

$$\begin{aligned}\tau_{d,\mathcal{T}} =& \boldsymbol{C}_{RB}(\mathbf{x}_{dyn,\mathcal{T}}, \mathbf{r}_T, \mathfrak{d}_H, \mathfrak{d}_F)\upsilon_{\mathcal{T}} + \boldsymbol{C}_A(\mathbf{x}_{dyn,\mathcal{T}}, \mathfrak{d}_H)\upsilon_{\mathcal{T}} + \boldsymbol{D}(\mathbf{x}_{dyn,\mathcal{T}}, \mathfrak{d}_H)\upsilon_{\mathcal{T}} \\ &+ \mathbf{g}(\mathbf{x}_{kin,\mathcal{T}}, \mathfrak{d}_F, \mathbf{r}_T, \mathfrak{d}_H), \end{aligned} \tag{5.8}$$

and

$$\mathbf{u}_{d,\mathcal{T}} = \boldsymbol{B}_a^{-1}(\mathbf{x}_{dyn,\mathcal{T}})\tau_{d,\mathcal{T}}. \tag{5.9}$$

Based on the previous analysis of the relationship between the geometric decision variables, the trim specifications and the robot dynamics, we can see that, the hull geometric variables $\mathfrak{d}_H$ have impact on all the dynamic terms. In order to make the optimization feasible, we decouple the optimization of $\mathfrak{d}_H$ from other decision variables. It means that, we perform a two-stage optimization. The hull volume is determined with its optimization algorithm at the first stage and the hull geometry is set as fixed for optimization of actuator configuration in the second optimization phase.

5.1 Optimization of Hull Size

The first stage of geometric optimization is to find optimal hull size. We use a cuboid to approximately model the navigation and sensory system, communication devices and CPU. The size and weight of the cuboid will be treated as constant for the hull optimal design and this raise the requirements for the minimal volume of the robot hull. Suppose the minimal radius and minimal length of the robot hull are denoted by $r_{H,min}$ and $l_{H,min}$ respectively. These two variables come from the requirement that the robot hull should at least enclose all the necessary devices (CPU, sensor and IMU and so on). The number of batteries is an integer decision variable denoted by $n_{bat} \in \mathbb{N}^+$. Buoyancy neutral is an important property for submersibles which means its mass is equal to the equivalent volume of water. More precisely, the total weight of the robot W should be equal to the total buoyancy B. We want to keep this property for our robot design, however, sometimes it is difficult to make the robot buoyancy neutral by merely changing the size of the robot. So we also need ballast material to help us to build the robot. Suppose we use $m_{ballast}$-kg ballast material with density $\rho_{ballast}$. The volume of the ballast material can be computed as $V_{ballast} = m_{ballast}/\rho_{ballast}$. The hull optimization problem is formulated as follows:

$$\begin{aligned}
\min_{l_H, r_H, n_{bat}, m_{ballast}} \quad & M(l_H, r_H, n_{bat}, m_{ballast}) + \mu_{BN}(B - W)^2 \\
\text{s.t.} \quad & l_{H,min} \leq l_H, \\
& r_{H,min} \leq r_H, \\
& E_{min} \leq n_{bat} E_{bat}, \\
& c(l_H, r_H, n_{bat}, m_{ballast}) + c_d \leq c_{max}, \\
& V_{bat}(n_{bat}) + V_{ballast}(m_{ballast}) + V_d \leq V(l_H, r_H)
\end{aligned} \tag{5.10}$$

where M denotes the optimization metrics from the robot design requirements. We will elaborate the objective function and the constraints in the following section. The volume V_d, the mass m_d and the cost c_d of all the devices are assumed to be known. c includes the cost for batteries, robot hull and the ballast. c_{max} is the budget for constructing the robot. V denotes the volume of the robot, V_{bat} is the volume of all batteries and the $V_{ballast}$ is the volume of the ballast material. E_{min} is the requirement for minimal energy storage, $V, V_{bat}, V_d, m_d, c_d, c, c_{max}, E_{min} \in \mathbb{R}^+$.

5.1.1 Buoyancy Neutral Optimization

Buoyancy neutral is the primal design goal, the buoyancy B can be calculated as

$$B = \rho_f g \, \pi r_H^2 l_H, \tag{5.11}$$

where ρ_f is the fluid density, r_H and l_H are the radius and length of the cylindrical hull, respectively.
Suppose the thickness of the hull is t_H and the density of the hull material is ρ_H, the density of the air is ρ_{air}. The weight of each battery is m_{bat}. The battery weight is calculate as

$$W_{bat} = n_{bat} m_{bat} g, \tag{5.12}$$

The hull profile surface weight is

$$W_{pfl} = \pi(r_H^2 - (r_H - t_H)^2) l_H \rho_H g, \tag{5.13}$$

The robot hull end caps weight is

$$W_{caps} = 2\pi r_H^2 t_H \rho_H g, \tag{5.14}$$

The air weight is

$$W_{air} = ((\pi(r_H - t_H)^2) l_H - V_{bat} - V_d) \rho_{air}, \tag{5.15}$$

The ballast weight is

$$W_{ballast} = m_{ballast} g, \tag{5.16}$$

The total weight is

$$W = W_{bat} + W_{pfl} + W_{caps} + W_{air} + W_{ballast}. \tag{5.17}$$

We want to make the total weight W equal the buoyancy B, thus we minimize their squared difference $(B-W)^2$ with large weighting coefficient μ_{BN}.
The ideal result for the buoyancy neural optimization is that there is no difference between the buoyancy and the gravity, i.e., $B = W$. However, it does not mean the restoring term disappears and the kinematic states have no influence on the dynamics states. When the buoyancy B equals the gravity W, the term $\mathbf{g}_{\mathbf{v}}(\eta)$ (4.18) will equal zero. It implies the restoring forces do not influence the surge, sway and heave velocity. However, based on 4.19, the angular velocities will be influenced by the buoyancy and the gravity, as long as the center of gravity $\mathbf{r}_g$ is not coincided with the center of buoyancy $\mathbf{r}_b$.

5.1.2 Metrics for Hull Design

In addition to the buoyancy neutral property, we also have other specifications which should be written into the optimization objective. Firstly, we want to maximize the energy storage of the robot, thus we want to increase the number of batteries,

$$M_{energy} = -\mu_{energy} n_{bat}, \tag{5.18}$$

where μ_{energy} is the weighting coefficient for energy storage.
The robot mainly moves along the surge direction, to extend the working time, less effort should be used to overcome the drag from the surge. It can also save the power consumption. Thus, we want to minimize the surge drag, and the surge drag is proportional to the frontal area of the hull. We define the surge minimizing term as

$$M_{drag} = \mu_{surge} r_H^2, \tag{5.19}$$

where the μ_{surge} is the weighting coefficient for minimizing the surge drag. At last, we want to construct the robot with less cost, thus we want to minimize all the decision variables.

$$M_{cost} = \mu_{cost,l} l_H + \mu_{cost,r} r_H + \mu_{cost,bat} n_{bat} + \mu_{cost,ballast} m_{ballast}, \tag{5.20}$$

where $\mu_{cost,l}$, $\mu_{cost,r}$, $\mu_{cost,bat}$ and $\mu_{cost,ballast}$ are weighting coefficients for minimizing the cost.

5.1.3 Hull Optimization Constraints

As mentioned above, the first two constraints are linear due to the requirement that the robot hull should contain the electronic devices. The third constraint means that the total energy storage $n_{bat}E_{bat}$ should at least exceed the minimal energy storage requirement E_{min} which is also linear. The fourth constraint indicates that the construction cost of robot hull enclosure, the cost for buying batteries, ballast and devices should be kept within the budget. Assume the cost for unit volume hull enclosure is constant, then this term is nonlinear due to the existence of decision variable product $r_H^2 l_H$ for calculating the volume. It is the same case for the nonlinearity in the last constraint. It means that the hull enclosure should be able to contain all the components except for the actuators.

5.1.4 Solving the Optimization Problem

Based on the formulation on the previous sections, we can see that in our hull optimization objective function, there exists products of two decision variables. Hence the objective is nonlinear. As analysed in the previous section, the last two constrains are nonlinear, too. Moreover, we have continuous decision variables l_H, r_H, $m_{ballast}$ and the discrete decision variable n_{bat}.
To sum up, in order to get a feasible solution for our hull optimization, we should formulate it as Mixed Integer Nonlinear Optimization (MINLP) which can be solved by the free MATLAB Toolbox for Optimization called OPTI Toolbox [CW12].
Note that for MINLP, we can only find the local minimum by means of the solver, thus the initial value is of importance. By testing the optimization codes, we found that, especially the discrete decision variable n_{bat} influences the optimization result obviously. So we choose the initial value of n_{bat} as follows:

$$n_{bat,init} = \lceil \frac{E_{min}}{E_{bat}} \rceil. \tag{5.21}$$

The other initial values of continuous decision variables can be set to zero, as they will not influence the optimization result.
After we get the optimal value $\mathfrak{d}_H^{opt}$ of the hull components, we know the size of the hull. Then, we can calculate all the hydrodynamic coefficients, the mass of the hull m_H, the moment of inertia of the hull $\boldsymbol{I}_{g,H}$, the center of gravity of hull in the body frame $\mathbf{r}_{g,H}$. All of these are kept constant in the next optimization stage.

5.2 Optimization of Actuator Configuration

The main concern of this work is to derive an optimal configuration of actuators by means of an optimization algorithm. Suppose we have m desired trim trajectories and n actuators, the basic idea behind the optimization is always to place all actuators in such a way that all trim trajectories can be perfectly tracked with the least control effort. Suppose there are n_t thrusters and n_f fins, the input vector $\mathbf{u}$ can be written as

$$\mathbf{u} = \begin{pmatrix} u_{T,1} \\ \vdots \\ u_{T,n_t} \\ \delta_{F,1} \\ \vdots \\ \delta_{F,n_f} \end{pmatrix}, \tag{5.22}$$

where $u_{T,1}, \ldots, u_{T,n_t}$ are thrusts generated by n_t thrusters, and $\delta_{F,1}, \ldots, \delta_{F,n_f}$ are deflection angles of n_f fins. The input matrix $\boldsymbol{B}_a$ maps the input vector into generalized force in 6-degrees of freedom, which is concatenated by the corresponding geometry-determined

```
Input : Initial geometric variables $\mathbf{d}_{T,1}, \cdots, \mathbf{d}_{T,n_t}, \mathbf{r}_{T,1}, \cdots, \mathbf{r}_{T,n_t}, x_{F,1}, \cdots, x_{F,n_f}$,
        $\gamma_{F,1}, \cdots, \gamma_{F,n_f}$, user-defined breaking constant $\epsilon$, configuration selection
        range $k_{final}$, maximal allowed iterations $k_{max}$
Output: Optimized actuator configuration geometry variables:
        $\mathbf{d}^{opt}_{T,1}, \cdots, \mathbf{d}^{opt}_{T,n_t}, \mathbf{r}^{opt}_{T,1}, \cdots, \mathbf{r}^{opt}_{T,n_t}, \mathbf{b}^{opt}_T = (b^{opt}_{T,1}, \cdots, b^{opt}_{T,n_t})^T, x^{opt}_{F,1}, \cdots, x^{opt}_{F,n_f}$,
        $\gamma^{opt}_{F,1}, \cdots, \gamma^{opt}_{F,n_f}$,
        Optimized error dynamics $\boldsymbol{A}^{opt}_{E,1}, \ldots, \boldsymbol{A}^{opt}_{E,m}$ and $\boldsymbol{B}^{opt}_{E,1}, \ldots, \boldsymbol{B}^{opt}_{E,m}$ for each trim
        trajectory,
        Control gains $\mathcal{K}_1, \cdots, \mathcal{K}_m$

// Calculate the initial parameters;
Calculate the initial geometric center $\mathbf{r}_G$;
Calculate the initial rigid body inertia matrix $\mathbf{M}_{RB}$;
Calculate the desired generalized forces $\tau_{d,\mathcal{T}_j}, j = 1, \ldots, m$ for each trim trajectory

// Determine the spin direction $b_{T,i}$;
bestConditionNumber $\longleftarrow +\infty$;
foreach $\mathbf{b}_T \in$ all $2^{n_t}$ assignments to $b_i$ do
    Compute error dynamics system and input matrices $\mathbf{A}_{E,1}, \ldots, \mathbf{A}_{E,m}$, $\mathbf{B}_{E,1}, \ldots, \mathbf{B}_{E,m}$
    and the controllability matrix $\mathbf{C}_{E,1}, \ldots, \mathbf{C}_{E,m}$ for each trim trajectory;
    Compute the condition number $C_s(\mathbf{C}_{E,j}), s = 1, \cdots, 2^{n_t}, j = 1, \cdots, m$ for each
    trim trajectory under the s-th spin configuration;
    Compute the average condition number $C_{av,s} = \frac{1}{m} \sum_{j=1}^{m} C_s(\mathbf{C}_{E,j})$;
end
Choose the optimal spin configuration $\mathbf{b}_T(s_{min})$ corresponding to the smallest average
condition number, where $s_{min} = \underset{s}{\operatorname{argmin}} C_{av,s}, s = 1, \cdots, 2^{n_t}$;
// Main Loop for optimization;
while at the k-th iteration, the total number of iterations $k_{total}$ is smaller than $k_{max}$ do
    Optimize $\mathbf{u}^*_1, \cdots, \mathbf{u}^*_m$ by solving a convex optimization problem;
    Optimize $\mathbf{r}_{T,1}, \cdots, \mathbf{r}_{T,n_t}$ and $x_{F,1}, \cdots, x_{F,n_f}$ by solving a nonlinear optimization
    problem;
    Optimize $\mathbf{d}_{T,1}, \cdots, \mathbf{d}_{T,n_t}$ sequentially from QCQP relaxation;
    Optimize $\gamma_{F,1}, \cdots, \gamma_{F,n_f}$ sequentially by solving a nonlinear optimization
    problem;
    Compute error dynamics system and input matrices $\mathbf{A}_{E,1}, \ldots, \mathbf{A}_{E,m}$, $\mathbf{B}_{E,1}, \ldots, \mathbf{B}_{E,m}$
    and the controllability matrix $\mathbf{C}_{E,1}, \ldots, \mathbf{C}_{E,m}$ for each trim trajectory;
    if $\mathbf{C}_{E,j}, j = 1, \ldots, m$ is singular or $\frac{1}{m} \sum_{j=1}^{m} \|\mathbf{C}^k_{E,j} - \mathbf{C}^{k-1}_{E,j}\| < \epsilon$ then
        Optimization terminates;
    end
    Update the center of gravity $\mathbf{r}_G$;
    Update the rigid body inertia matrix $\mathbf{M}_{RB}$;
    Update the desired input $\tau_{d,\mathcal{T}_j}, j = 1, \ldots, m$ for each trim trajectory;
end
```

```
// Postprocessing;
Calculate the optimal configuration index $k_{opt} = \operatorname{argmin}_k C_{av,k}$, where
$C_{av,k} = \frac{1}{m}\sum_{j=1}^{m} C_k(\mathbf{C}_{E,j}), k = k_{total} - k_{last}, \cdots, k_{total} - 1$;
Choose the optimal directions of the thrusters:
$\mathbf{d}_{T,1}^{opt} = \mathbf{d}_{T,1}(k_{opt}), \cdots, \mathbf{d}_{T,n_t}^{opt} = \mathbf{d}_{T,n_t}(k_{opt})$;
Choose the optimal positions of the thrusters: $\mathbf{r}_{T,1}^{opt} = \mathbf{r}_{T,1}(k_{opt}), \cdots, \mathbf{r}_{T,n_t}^{opt} = \mathbf{r}_{T,n_t}(k_{opt})$;
Choose the optimal fin geometric centers: $x_{T,1}^{opt} = x_{T,1}(k_{opt}), \cdots, x_{T,n_f}^{opt} = x_{T,n_f}(k_{opt})$;
Choose the optimal fin rotation angles: $\gamma_{T,1}^{opt} = \gamma_{T,1}(k_{opt}), \cdots, \gamma_{T,n_f}^{opt} = \gamma_{T,n_f}(k_{opt})$;
Update the the desired generalized forces $\tau_{d,\mathcal{T}_j}^{opt}, j = 1, \ldots, m$;
Update the the desired control inputs $\mathbf{u}_{d,j}^{opt} = (\mathbf{B}_a^{opt})^{-1}\tau_{d,j}, j = 1, \ldots, m$;
Calculate the optimal linearized error dynamics $\mathbf{A}_{E,1}^{opt}, \ldots, \mathbf{A}_{E,m}^{opt}$ and $\mathbf{B}_{E,1}^{opt}, \ldots, \mathbf{B}_{E,m}^{opt}$;
Choose $\mathcal{Q}_{E,1}, \cdots, \mathcal{Q}_{E,m}$ and $\mathcal{R}_{E,1}, \cdots, \mathcal{R}_{E,n}$ ;
Compute $\mathcal{K}_1, \cdots, \mathcal{K}_m$.
```

Algorithm 1: Actuator Configuration Optimization Algorithm

mapping vector of each fin and thruster. It is defined as

$$\boldsymbol{B}_a = \begin{pmatrix} \boldsymbol{B}_{T,1} & \ldots & \boldsymbol{B}_{T,n_t} & \boldsymbol{B}_{F,1} & \ldots \boldsymbol{B}_{F,n_f} \end{pmatrix}. \tag{5.23}$$

Mathematically, the optimization problem can be written as

$$\begin{array}{ll} \min\limits_{\mathbf{u}_1^*,\cdots,\mathbf{u}_m^*,\mathfrak{d}_P,\mathfrak{d}_F} \sum\limits_{j=1}^{m} \|\boldsymbol{B}_a\mathbf{u}_j^* - \tau_{d,j}\|_2^2 & \\ \text{s.t.} \quad \mathbf{d}_{T,i}^T\mathbf{d}_{T,i} = 1, & i = 1, \cdots, n_t \\ b_{T,i} \in \{-1, 1\}, & i = 1, \cdots, n_t \\ -l_H/2 \le \mathbf{r}_{T,i}(1) \le l_H/2, & i = 1, \cdots, n_t \\ \mathbf{r}_{T,i}(2)^2 + \mathbf{r}_{T,i}(3)^2 = r_H^2, & i = 1, \cdots, n_t \\ -(l_H - b_{F,i})/2 \le x_{F,i} \le (l_H - b_{F,i})/2, & i = 1, \cdots, n_f \\ -\pi/2 \le \gamma_{F,i} \le \pi/2, & i = 1, \cdots, n_f \\ \mathbf{u}_{min} \le \mathbf{u}_j^* \le \mathbf{u}_{max}, & j = 1, \cdots, m \end{array} \tag{5.24}$$

The product of $\boldsymbol{B}_a$ and $\mathbf{u}_j^*$ causes coupling between the control inputs and the geometric decision variables. Moreover, there exists both convex and non-convex constraints. Directly solving this optimization problem is impossible. Based on the modeling methods of thrusters and fins in chapter 2, optimizing all geometric parameters of actuators at the same time are nearly unsolvable. According to the idea from [DSZ+16], we adopt an iterative method to optimize all geometric parameters and the control input vector in sequence. That is, we divide the decision variables into four groups: the spin direction of all thrusters $b_{T,1}, \cdots, b_{T,n}$, the control inputs $\mathbf{u}_1^*, \cdots, \mathbf{u}_m^*$, the position of thrusters and fins including $\mathbf{r}_{T,1}, \cdots, \mathbf{r}_{T,n_t}$ and $x_{F,1}, \cdots, x_{F,n_f}$ and the direction (orientation) of thrusters and fins $\mathbf{d}_{T,1}, \cdots, \mathbf{d}_{T,n_t}$ and $\gamma_{F,1}, \cdots, \gamma_{F,n_f}$. The spin direction of all thrusters should be determined at first according to the average condition numbers of all trim trajectories. Once they are determined, for other optimization phases, we keep them constant. The other three decision variables will be optimized in sequence. In one optimization iteration, we set all other groups of decision variables as fixed and just optimize the control inputs at first. At the second optimization stage, the positions of all thrusters and fins will be optimized, while we treat control inputs and directions as constant. As in the previous two optimization phases, control inputs and positions will be set unchanged for the third-stage optimization. Unlike the previous two optimization stages where all control inputs and all positions are optimized simultaneously, we are only able to optimize the direction (orientation) vector of one thruster (fin), while the directions (orientations) of other thrusters (fins) are fixed. After one direction (orientation) vector is optimized, we proceed until all directions (orientations) are optimized. One iteration step ends after all these sub-optimization problems are finished. In the following sections, the details of this algorithms will be discussed.

5.2.1 Initialization

We apply nonlinear programming to solve the position and direction optimization. The nonlinear optimization algorithms can only find the local minimum which is strongly dependent on the initial value. The initial guess will influence the result considerably. One option is to set the initial values according to an expert's pre-knowledge of robot design. Our motivation for this work is to find a computational approach allowing non-experts to design, explore, and evaluate a wide range of different underwater robots. From their perspective, we randomly generate the thruster motor direction $\mathbf{d}_T$, the thruster position $\mathbf{r}_T$, the fin geometric center position x_F and the fin orientation γ_F.
Note that the direction vector $\mathbf{d}_T$ is constrained as unit vector which breaks the convexity. To generate a unit direction vector, we use the following steps:

- Choose a random value of θ_T between $-\pi$ and π;
- Choose a random value of z between -1 and 1;
- Compute the resulting point : $(d_x, d_y, d_z) = (\sqrt{1-z^2}\cos(\theta_T),\ \sqrt{1-z^2}\sin(\theta_T), z)$

The thruster position $\mathbf{r}_T = (x_T, y_T, z_T)^T$ is supposed to be located on the hull cylinder surface. Thus, the x-component should not exceed the hull length, and x_T should be randomly generated within the range $[-l_H/2, l_H/2]$. Since the thrusters are attached on the hull surface, the y-component and z-component are constrained by $y_T^2 + z_T^2 = r_H^2$. Thus, we can randomly generate the angle ρ_T within the range $[-\pi, \pi]$, then $y_T = r_H \cos(\rho_T)$ and $z_T = r_H \sin(\rho_T)$.
Besides, we should choose x_F and γ_F from $[-(l_H - b_F)/2, (l_H - b_F)/2]$ at random, respectively.

5.2.2 Spin Direction Optimization

Unlike the other continuous geometric decision variables, the spin directions $b_{T,1}, \cdots, b_{T,n_t}$ of thrusters are discrete decision variables which can be chosen from $\{-1, 1\}$. Thus we want to choose the most appropriate spin direction configuration before the main optimization loops. For n_t thrusters, we have 2^{n_t} spin configurations to select. The condition number can be used as an input-output controllability measure. A small condition number is aimed for, because the multivariable effects of uncertainty are not likely to be serious [SP05]. Thus we choose the condition number of the error dynamics controllability matrix as the measure for selecting spin directions.
Since we have m different trim trajectories, one spin direction configuration may be very suitable for one trim trajectory segment but work badly for the others. To solve this problem, for each spin direction configuration, we calculate the average condition number of all trim trajectories for each spin configuration. Then we compare them and choose the spin directions corresponding to the smallest average condition number as the optimal ones. They will be set as fixed for further optimizations.

$$\mathbf{b}_T^{opt} = \mathbf{b}_T(s_{min}), \tag{5.25}$$

and

$$s_{min} = \underset{s}{\operatorname{argmin}}\, C_{av,s}, \tag{5.26}$$

where $C_{av,s} = \dfrac{1}{m} \sum_{j=1}^{m} C_s(\boldsymbol{C}_{E,j})$ and s is index for the spin direction ($s = 1, \cdots, 2^{n_t}$).

5.2.3 Control Input Optimization

The control input optimization is the first part in the main loop. During this phase, we want to choose the optimal inputs of all actuators by minimizing the difference between the desired generalized force and the generalized force generated by the current actuator configuration for each trim trajectory segment. Since in this situation, we only consider $\mathbf{u}_1^*, \cdots, \mathbf{u}_m^*$, the optimization problem 5.24 can be simplified as

$$\begin{aligned} \min_{\mathbf{u}_1^*, \cdots, \mathbf{u}_m^*} \quad & \textstyle\sum_{j=1}^{m} ||\boldsymbol{B}_a \mathbf{u}_j^* - \tau_{d,j}||_2^2 \\ \text{s.t.} \quad & \mathbf{u}_{min} \leq \mathbf{u}_j^* \leq \mathbf{u}_{max}, \quad j = 1, \cdots, m \end{aligned} \tag{5.27}$$

The objective function is the sum of least squares which is convex, and the linear constrains for control input vectors are also convex. Thus, it is a convex optimization problem which can be solved by CVX, a package for specifying and solving convex programs [GB14, GB08].

5.2.4 Position Optimization

For position optimization of the actuators we refer to finding the optimal position vector $\mathbf{r}_T$ for thrusters and the optimal geometric center x_F for fins in the body frame $\{b\}$. Recall

the defined thruster mapping matrix 2.17 and the fin mapping vector 2.38 in Chapter 2. By observing their structures, we infer that the positions only have impact on the generated moments. Thus, we abbreviate the desired trim generalized forces $\tau_{d,1}, \cdots, \tau_{d,m}$ to the desired trim moments $\mathbf{m}_{d,1}, \cdots, \mathbf{m}_{d,m}$.
From equation 2.17 we know that the moments generated by thrusters consist of two parts, i.e., the moment produced by a thruster's rotation $\mathbf{m}_{T,r} = b_T \lambda_T u_T \mathbf{d}_T$ and the moment due to thrust force $\mathbf{r}_T \times u_T \mathbf{d}_T$ relating to our decision variable $\mathbf{r}_T$ in this optimization phase. As mentioned before, due to the implementing of the iterative optimization method, b_T and $\mathbf{d}_T$ are constant for position optimization. Thus, we can extract the rotation moment from the desired trim moment,

$$\mathbf{m}'_{d,j} = \mathbf{m}_{d,j} - \sum_{i=1}^{n_t} b_{T,i} \lambda_T u_{T,i} \mathbf{d}_{T,i}, \ j = 1, \cdots, m \tag{5.28}$$

Then we obtain a series of quasi-desired moments $\mathbf{m}'_{d,1}, \cdots, \mathbf{m}'_{d,m}$ for m trim trajectories. The term $\mathbf{r}_{T,i} \times u_{T,i} \mathbf{d}_{T,i}, i = 1, \cdots, n_t$ for the i-th thruster can be reformulated as $-u_{T,i}\mathbf{d}_{T,i} \times \mathbf{r}_{T,i}, i = 1, \cdots, n_f$. We define the position mapping matrix for the i-th thruster as

$$\boldsymbol{B}^T_{pos,i} = -u_{T,i}\mathbf{d}_{T,i} \times = \begin{pmatrix} 0 & u_{T,i}d_{T,z,i} & -u_{T,i}d_{T,y,i} \\ -u_{T,i}d_{T,z,i} & 0 & u_{T,i}d_{T,x,i} \\ u_{T,i}d_{T,y,i} & -u_{T,i}d_{T,x,i} & 0 \end{pmatrix} \in \mathbb{R}^{3\times3}, \tag{5.29}$$

where the direction vector $\mathbf{d}_{T,i} = (x_{T,i}, y_{T,i}, z_{T,i})^T$ corresponds to the i-th thruster and the $u_{T,i}$ is its generated thrust force. Then its thrust moment can be calculated as

$$\mathbf{m}_{T,i,t} = \boldsymbol{B}^T_{pos,i} \mathbf{r}_{T,i}, \tag{5.30}$$

where $\mathbf{r}_{T,i} = (r_{T,x,i}, r_{T,y,i}, r_{T,z,i})$ represents the position vector of i-th thruster in the body frame $\{b\}$.
Similarly, from 2.38, we extract the last three rows for moment mapping to obtain the moments generated by i-th fin for m-th trim trajectory segment as:

$$\mathbf{m}_{F,i} = \begin{pmatrix} -\frac{1}{2} c_L a_{F,i} b_{F,i} Q_j(u)(d_H + a_{F,i}) \\ c_L a_{F,i} b_{F,i} Q_j(u) x_{F,i} \sin(\gamma_{F,i}) \\ c_L a_{F,i} b_{F,i} Q_j(u) x_{F,i} \cos(\gamma_{F,i}), \end{pmatrix} \delta_{F,i}, \tag{5.31}$$

where $a_{F,i}$ and $b_{F,i}$ are the length and width of the i-th fin, respectively. $x_{F,i}$ is the x-coordinate of the i-th fin in the body frame $\{b\}$ and $\gamma_{F,i}$ is the rotation angle of the i-th fin. $Q_j(u) = 0.5\rho u^2_{T,j}$, where the $u_{T,j}$ is the desired trim surge velocity for the j-th trim trajectory segment. Reforming it, we obtain

$$\mathbf{m}_{F,i} = \begin{pmatrix} -\frac{1}{2} c_L a_{F,i} b_{F,i} Q_j(u)(d_H + a_{F,i})\delta_{F,i} & 0 & 0 \\ 0 & c_L a_{F,i} b_{F,i} Q_j(u) \sin(\gamma_{F,i})\delta_{F,i} & 0 \\ 0 & 0 & c_L a_{F,i} b_{F,i} Q_j(u) \cos(\gamma_{F,i})\delta_{F,i} \end{pmatrix} \begin{pmatrix} 1 \\ x_{F,i} \\ x_{F,i} \end{pmatrix}. \tag{5.32}$$

We define the three-dimensional quasi-position vector $\mathbf{r}_{F,i}$ for the i-th fin as:

$$\mathbf{r}_{F,i} = \begin{pmatrix} 1 \\ x_{F,i} \\ x_{F,i} \end{pmatrix}, \tag{5.33}$$

where $-\frac{1}{2}(l_H - b_{F,i}) \leq x_F \leq \frac{1}{2}(l_H - b_{F,i})$. Then, we define the position mapping matrix for the i-th fin of j-th trim trajectory segment as

$$\boldsymbol{B}^F_{pos,i,\mathcal{T}_j} = \begin{pmatrix} -\frac{1}{2}c_L a_{F,i} b_{F,i} Q_j(u)(d_H + a_{F,i})\delta_{F,i} & 0 & 0 \\ 0 & c_L a_{F,i} b_{F,i} Q_j(u)\sin(\gamma_{F,i})\delta_{F,i} & 0 \\ 0 & 0 & c_L a_{F,i} b_{F,i} Q_j(u)\cos(\gamma_{F,i})\delta_{F,i} \end{pmatrix}. \tag{5.34}$$

We also have the similar relationship:

$$\mathbf{m}_{F,i} = \boldsymbol{B}^F_{pos,i,\mathcal{T}_j}\mathbf{r}_{F,i}. \tag{5.35}$$

Recall the constraints concerning the positions: $-l_H/2 \leq \mathbf{r}_{T,i}(1) \leq l_H/2, i = 1, \cdots, n_t$, $\mathbf{r}_{T,i}(2)^2 + \mathbf{r}_{T,i}(3)^2 = r_H^2, i = 1, \cdots, n_t$, $-(l_H - b_{F,i})/2 \leq x_{F,i} \leq (l_H - b_{F,i})/2, i = 1, \cdots, n_f$. The first and third constraint are convex, while the second unit vector constraint breaks the convexity. We want to transform all constraints into convex constraints by means of redefining the position vector of thrusters in the cylindrical frame,

$$\mathbf{r}_{T,i,cyl} = \begin{pmatrix} x_{T,i} \\ r_H\cos(\rho_{T,i}) \\ r_H\sin(\rho_{T,i}) \end{pmatrix} \in \mathbb{R}^{3\times1}. \tag{5.36}$$

Based on the previous definitions, we can simplify and reformulate the basic optimization problem as

$$\begin{array}{lllll} \min\limits_{x_{T,1},\cdots,x_{T,n_t};x_{F,1},\cdots,x_{F,n_f};\rho_{T,1},\cdots,\rho_{T,n_t}} & \multicolumn{4}{l}{\sum_{j=1}^m \sum_{i=1}^n \|\boldsymbol{B}^a_{pos,i}\mathbf{r}_{a,i} - \mathbf{m}'_{d,j}\|_2^2} \\ \text{s.t.} & l_H/2 & \leq x_{T,i} & \leq l_H/2, & i = 1, \cdots, n_t \\ & -\pi & \leq \rho_{T,i} & \leq \pi, & i = 1, \cdots, n_t \\ & -(l_H - b_{F,i}) & \leq x_{F,i} & \leq (l_H - b_{F,i}), & i = 1, \cdots, n_f \end{array}, \tag{5.37}$$

where the $\mathbf{r}_{a,i}$ denotes the position vector for the i-th actuator. Explicitly, for thrusters $\mathbf{r}_{a,i} = \mathbf{r}_{T,cyl,i}$ and for fins $\mathbf{r}_{a,i} = \mathbf{r}_{F,i}$. $\boldsymbol{B}^a_{pos,i}$ represents the corresponding position mapping matrix (for the i-th fin along j-th trim trajectory: $\boldsymbol{B}^F_{pos,i,\mathcal{T}_j}$, for the i-th thruster: $\boldsymbol{B}^T_{pos,i}$). We can stack them together and define the complete position mapping matrix for totally m trim trajectories and n actuators (n_t thrusters and n_f fins) as

$$\boldsymbol{B}_{pos} = \begin{pmatrix} \boldsymbol{B}^T_{pos,1} & \cdots & \boldsymbol{B}^T_{pos,n_t} & \boldsymbol{B}^F_{pos,1,\mathcal{T}_1} & \cdots & \boldsymbol{B}^F_{pos,n_f,\mathcal{T}_1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{B}^T_{pos,1} & \cdots & \boldsymbol{B}^T_{pos,n_t} & \boldsymbol{B}^F_{pos,1,\mathcal{T}_1} & \cdots & \boldsymbol{B}^F_{pos,n_f,\mathcal{T}_1} \end{pmatrix} \in \mathbb{R}^{(m\times3)\times(n\times3)}. \tag{5.38}$$

Stacking all the position vectors and defining the position vector $\mathbf{r}_{pos,a}$ as

$$\mathbf{r}_{pos,a} = \begin{pmatrix} x_{T,1} \\ r_H \cos(\rho_{T,1}) \\ r_H \sin(\rho_{T,1}) \\ \vdots \\ x_{T,p} \\ r_H \cos(\rho_{T,n_t}) \\ r_H \sin(\rho_{T,n_t}) \\ 1 \\ x_{F,1} \\ x_{F,1} \\ \vdots \\ 1 \\ x_{F,n_f} \\ x_{F,n_f} \end{pmatrix} \in \mathbb{R}^{(n \times 3) \times 1}. \tag{5.39}$$

Besides, we also stack all the quasi desired moments into the vector

$$\mathbf{m}_{pos,a} = \begin{pmatrix} \mathbf{m}'_{d,1} \\ \vdots \\ \mathbf{m}'_{d,m} \end{pmatrix} \in \mathbb{R}^{(m \times 3) \times 1} \tag{5.40}$$

Our goal is to find an optimal position vector $\mathbf{r}_{pos,a}$ so that the generated generalized forces by all the actuators approach the desired quasi moments for all trim trajectories as close as possible. Mathematically, we expect

$$\begin{pmatrix} \boldsymbol{B}^T_{pos,1} & \cdots & \boldsymbol{B}^T_{pos,n_t} & \boldsymbol{B}^F_{pos,1,\mathcal{T}_1} & \cdots & \boldsymbol{B}^F_{pos,n_f} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{B}^T_{pos,1} & \cdots & \boldsymbol{B}^T_{pos,n_t} & \boldsymbol{B}^F_{pos,1,\mathcal{T}_m} & \cdots & \boldsymbol{B}^F_{pos,n_f,\mathcal{T}_m} \end{pmatrix} \begin{pmatrix} x_{T,1} \\ r_H \cos(\rho_{T,1}) \\ r_H \sin(\rho_{T,1}) \\ \vdots \\ x_{T,n_t} \\ r_H \cos(\rho_{T,n_t}) \\ r_H \sin(\rho_{T,n_f}) \\ 1 \\ x_{F,1} \\ x_{F,1} \\ \vdots \\ 1 \\ x_{F,n_f} \\ x_{F,n_f} \end{pmatrix} = \begin{pmatrix} \mathbf{m}'_{d,1} \\ \vdots \\ \mathbf{m}'_{d,m} \end{pmatrix}. \tag{5.41}$$

We can formulate the optimization problem as

$$\begin{array}{llllll}\min\limits_{x_{T,1},\cdots,x_{T,n_t};x_{F,1},\cdots,x_{F,n_f};\rho_{T,1},\cdots,\rho_{T,n_t}} & \|\boldsymbol{B}_{pos}\mathbf{r}_{pos,a}-\mathbf{m}_{pos,a}\|_2^2 & & & & \\ \text{s.t.} & l_H/2 & \leq x_{T,i} & \leq l_H/2, & i=1,\cdots,n_t & . \\ & -\pi & \leq \rho_{T,i} & \leq \pi, & i=1,\cdots,n_t & \\ & -(l_H-b_{F,i}) & \leq x_{F,i} & \leq (l_H-b_{F,i}), & i=1,\cdots,n_f & \end{array} \tag{5.42}$$

All constraints are linear convex, while the vector $\mathbf{r}_{pos,a}$ depends nonlinearly on all decision variables and the objective function is nonlinear least square. Thus this optimization problem should be solved by constrained nonlinear programming. In the first iteration, we choose the randomly generated initial $\mathbf{r}_{T,1},\cdots,\mathbf{r}_{T,n_t}$ and convert them into $x_{T,1},\cdots,x_{T,n_t}$ and $\rho_{T,1},\cdots,\rho_{T,n_t}$. Combining them with the randomly generated $x_{F,1},\cdots,x_{F,n_f}$, we can construct our initial guess for $\mathbf{r}_{pos,a}$. For the other iterations, we utilize the optimization results from the $k-1$ iteration at iteration k.

5.2.5 Orientation Optimization

Assume now we optimize the c-th thruster orientation $\mathbf{d}_{T,c}=[d_{x,c}\quad d_{y,c}\quad d_{z,c}]^T$

$$\begin{pmatrix} u_{T,c} & 0 & 0 \\ 0 & u_{T,c} & 0 \\ 0 & 0 & u_{T,c} \\ b_{T,c}\lambda_{T,c}u_{T,c} & -z_{T,c}u_{T,c} & y_{T,c}u_{T,c} \\ z_{T,c}u_{T,c} & b_{T,c}\lambda_{T,c}u_{T,c} & -x_{T,c}u_{T,c} \\ -y_{T,c}u_{T,c} & x_{T,c}u_{T,c} & b_{T,c}\lambda_{T,c}u_{T,c}\end{pmatrix}\begin{pmatrix} d_{x,c} \\ d_{y,c} \\ d_{z,c}\end{pmatrix}=\tau'_{d,j}, \tag{5.43}$$

where $\tau'_{d,j}=\tau_{d,j}-\sum_{i=1,i\neq c}^{n}\tau_{a,i,\mathcal{T}_j}$, $j=1,\cdots,m$, i.e., we subtract the sum of generated forces and moments of the other $n-1$ actuators under the current geometric configuration from the desired generalised force τ for the m-th trim trajectory. We define the motor orientation optimization matrix $\boldsymbol{B}^T_{ort,c,\mathcal{T}_j}$ for the c-th thruster along j-th trim trajectory as

$$\boldsymbol{B}^T_{ort,c,\mathcal{T}_j}=u_{T,c}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ b_{T,c}\lambda_{T,c} & -z_{T,c} & y_{T,c}u_{T,c} \\ z_{T,c}u_{T,c} & b_{T,c}\lambda_{T,c}u_{T,c} & -x_{T,c}u_{T,c} \\ -y_{T,c}u_{T,c} & x_{T,c}u_{T,c} & b_{T,c}\lambda_{T,c}u_{T,c}\end{pmatrix}. \tag{5.44}$$

From this matrix, we can see that the direction optimization matrix is dependent on the thruster input force u_T and its geometric parameters b_T and $\mathbf{r}_T$. u_T is calculated in the first input optimization phase and trajectory-dependent. $\mathbf{r}_T=(x_T,y_T,z_T)^T$ is optimized in the second phase. In the current direction optimization phase, we take the optimal values of them from the previous two optimization phases. Then our general optimization in this case can be simplified as

$$\begin{array}{ll}\min\limits_{\mathbf{d}_{T,c}} & \sum_{i=1}^{m} \|\boldsymbol{B}^T_{ort,c,\mathcal{T}_j}\mathbf{d}_{T,c} - \tau'_{d,m}\|_2^2 \\ \text{s.t.} & \mathbf{d}^T_{T,c}\mathbf{d}_{T,c} \quad = 1\end{array}. \tag{5.45}$$

The objective function is a sum of least squares and convex, however the constraint $\mathbf{d}^T_{T,c}\mathbf{d}_{T,c} = 1$ breaks the convexity. This optimization problem should also be solved by nonlinear programming. Therefore, the initial value plays a decisive role for the solution. A reasonable relaxation is to convert the equality constraint $\mathbf{d}^T_{T,c}\mathbf{d}_{T,c} = 1$ into an inequality constraint $\mathbf{d}^T_{T,c}\mathbf{d}_{T,c} \leq 1$. Then the optimization problem becomes a quadratically constrained quadratic program (QCQP):

$$\begin{array}{ll}\min\limits_{\mathbf{d}_{T,c}} & \sum_{j=1}^{m} \|\boldsymbol{B}^T_{ort,c,\mathcal{T}_j}\mathbf{d}_{T,c} - \tau'_{d,m}\|_2^2 \\ \text{s.t.} & \mathbf{d}^T_{T,c}\mathbf{d}_{T,c} \quad \leq 1\end{array}. \tag{5.46}$$

It is a convex programming problem having a global minimum that can be solved by CVX [GB14, GB08]. This global minimum will be chosen as the initial value for the original nonlinear optimization problem 5.45.

In terms of optimizing the orientation of fins, we adopt the same approach. Firstly, we reformulate equation 2.38 and extract a vector including only the orientation of the current optimized fin.

$$\tau_{F,c,\mathcal{T}_j} = c_L a_{F,c} b_{F,c} q_j(u)\delta_{F,c}\begin{pmatrix}0\\ \cos(\gamma_{F,c})\\ -\sin(\gamma_{F,c})\\ -\frac{1}{2}(d_H + a_{F,c})\\ x_{F,c}\sin(\gamma_{F,c})\\ x_{F,c}\cos(\gamma_{F,c})\end{pmatrix}. \tag{5.47}$$

Note that the fin deflection angle $\delta_{F,c}$ is from the result in the control input optimization phase.

Then we can formulate the optimization problem

$$\begin{array}{ll}\min\limits_{\gamma_{F,c}} & \sum_{j=1}^{m} \|\tau_{F,c,\mathcal{T}_j} - \tau'_{d,j}\|_2^2 \\ \text{s.t.} & -\pi \quad \leq \gamma_{F,c} \quad \leq \pi\end{array}. \tag{5.48}$$

This is sum of nonlinear squares with linear constraints which is not convex. We should use nonlinear optimization to solve this problem. For the initial value, we utilize the same method as in the position optimization. For the first iteration, we take the randomly generated fin orientation within $[-\pi, \pi]$ as the initial guess, whereas for other iterations we use the result from the last iteration as the initial value.

5.2.6 Termination Criteria and Convergence Discussion

The basic criteria for breaking the iterative loop is the loss of controllability for a certain configuration. By testing the real optimization codes with 6 thruster and with 2 thrusters and 4 fins, we found that the controllability will not be broken within a large number of iterations. However, all geometric parameters of actuators have already stayed nearly constant. Therefore, we propose another criteria to break the optimization loop. Because the controllability matrix $\boldsymbol{C}_E$ is uniquely related to the system matrix $\boldsymbol{A}_E$ and the input matrix $\boldsymbol{B}_E$ and these two matrices are bijective functions of geometric decision variables $\mathfrak{d}_P$ and $\mathfrak{d}_F$, we can use the increase of l_2 norm of error dynamics controllability matrices $||\boldsymbol{C}_E^k - \boldsymbol{C}_E^{k-1}||$ to indicate the convergence of actuator geometric decision variables. Since we have m trim trajectories, we take the average of all norm incremental amount $\frac{1}{m}\sum_{j=1}^m ||\boldsymbol{C}_{E,j}^k - \boldsymbol{C}_{E,j}^{k-1}||$, if it is smaller than a user-predefined small constant ϵ, the optimization will end. The choice of a suitable value ϵ is not simple. For some randomly generated initial geometric variables, the average norm difference can decrease within the value in tens of iterations, while for some other configuration it can not fall down into the ϵ even though in hundreds of iterations which takes long time. However, in these cases, the average norm difference actually already stay stable (oscillating within a quite small range) for many iterations. It means that ϵ is set too small that the average norm difference can never reach. Hence, we set the maximal iterations to avoid redundant iterations. A better alternative to solve this problem is that we choose ϵ according to the initial average norm of all trim trajectory error dynamics controllability matrices $\boldsymbol{C}_{E,av,init} = \frac{1}{m}\sum_{j=1}^m ||\boldsymbol{C}_{E,j}^1||$. For instance, if $\boldsymbol{C}_{E,av,init}$ is in the order of 10^5, we can set ϵ as 100.

5.2.7 Postprocessing

Once one of the breaking criteria described in the last subsection is satisfied, the main loop searching for a local optimal actuator configuration will be terminated. Suppose we have totally k_{total} optimization iterations, selecting the most optimal one is of significance. As in the spin direction optimization phase, the spin direction configuration with smallest average condition number of the error dynamics controllability matrices along all trim trajectories will be treated as the best one. Smaller condition number means better ability against uncertainties. The kinematic tracking is the first concern for the robot working performance. By observing the implementation results, we find that there is no obvious difference between the tracking error in the very last steps. Hence, we can combine these two points. For the last k_{last} iterations (iterations $k_{total} - k_{last}, \cdots, k_{total} - 1$), we choose the geometric variables at iteration k_{opt} with the smallest average condition number of m trim trajectories error dynamics controllability matrices, i.e.,

$$k_{opt} = \underset{k}{\operatorname{argmin}}\, C_{av,k}, \tag{5.49}$$

where $C_{av,k} = \frac{1}{m}\sum_{j=1}^m C_k(\boldsymbol{C}_{E,j}), k = k_{total} - k_{last}, \cdots, k_{total} - 1$.

Then we choose the geometric variables at iteration k_{opt} as the optimal result. Using these geometric parameters and the modular modeling method elaborated in Chapter 2, we can build the optimal robot dynamics as

$$\begin{aligned}
&\boldsymbol{M}_{RB}^{opt}(\mathbf{r}_T^{opt}, \mathfrak{d}_H^{opt}, \mathfrak{d}_F^{opt})\dot{\upsilon} + \boldsymbol{M}_A^{opt}(\mathbf{x}_{dyn,\mathcal{T}}, \mathfrak{d}_H^{opt})\dot{\upsilon} \\
&+ \boldsymbol{C}_{RB}^{opt}(\mathbf{x}_{dyn,\mathcal{T}}, \mathbf{r}_T^{opt}, \mathfrak{d}_H^{opt}, \mathfrak{d}_F^{opt})\upsilon + \boldsymbol{C}_A^{opt}(\mathbf{x}_{dyn,\mathcal{T}}, \mathfrak{d}_H^{opt})\upsilon + \boldsymbol{D}(\mathbf{x}_{dyn,\mathcal{T}}, \mathfrak{d}_H^{opt})\upsilon \\
&+ \mathbf{g}^{opt}(\mathbf{x}_{kin,\mathcal{T}}, \mathfrak{d}_F^{opt}, \mathbf{r}_T^{opt}, \mathfrak{d}_H^{opt}) \\
&= \boldsymbol{B}_T^{opt}(\mathbf{d}_T^{opt}, \mathbf{r}_T^{opt}, b_T^{opt})\begin{pmatrix} u_{T,1} \\ \vdots \\ u_{T,n_t} \end{pmatrix} + \boldsymbol{B}_F^{opt}(\mathbf{x}_{dyn,\mathcal{T}}, \mathfrak{d}_F^{opt})\begin{pmatrix} \delta_{F,1} \\ \vdots \\ \delta_{F,n_f} \end{pmatrix}.
\end{aligned} \tag{5.50}$$

The optimal desired trim input is calculated as

$$\begin{aligned}
\tau_{d,\mathcal{T}}^{opt} =& \boldsymbol{C}_{RB}^{opt}(\mathbf{x}_{dyn,\mathcal{T}}, \mathbf{r}_T^{opt}, \mathfrak{d}_H^{opt}, \mathfrak{d}_F)\upsilon_{\mathcal{T}} + \boldsymbol{C}_A^{opt}(\mathbf{x}_{dyn,\mathcal{T}}, \mathfrak{d}_H^{opt})\upsilon_{\mathcal{T}} + \boldsymbol{D}^{opt}(\mathbf{x}_{dyn,\mathcal{T}}, \mathfrak{d}_H^{opt})\upsilon_{\mathcal{T}} \\
&+ \mathbf{g}^{opt}(\mathbf{x}_{kin,\mathcal{T}}, \mathfrak{d}_F, \mathbf{r}_T^{opt}, \mathfrak{d}_H^{opt}).
\end{aligned} \tag{5.51}$$

Based on this dynamics, we use the reformulation approach and the nonlinear transformation in chapter 3 to get m linearized error dynamics system matrices $\boldsymbol{A}_{E,1}^{opt}, \cdots, \boldsymbol{A}_{E,m}^{opt}$ and input matrices $\boldsymbol{B}_{E,1}^{opt}, \cdots, \boldsymbol{B}_{E,m}^{opt}$ corresponding to m trim trajectories specifications $\mathcal{T}_1, \cdots, \mathcal{T}_m$. The feedback gain $\mathcal{K}_1^{opt}, \cdots, \mathcal{K}_m^{opt}$ of the error dynamics will be calculated from these system and input matrices. The error dynamics inputs are:

$$\begin{aligned}
\mathbf{u}_{E,1} &= -\mathcal{K}_1^{opt} x_{E,1} \\
&\vdots \\
\mathbf{u}_{E,m} &= -\mathcal{K}_m^{opt} x_{E,m}
\end{aligned} \tag{5.52}$$

Recall the definition of the error dynamics input $\mathbf{u}_E = \mathbf{u} - \mathbf{u}_{\mathcal{T}}$, then we obtain the control inputs giving into the robot system for m trim trajectories $\mathbf{u}_1, \cdots, \mathbf{u}_m$:

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{u}_{E,1} + \mathbf{u}_{\mathcal{T}_1}^{opt} \\
&\vdots \\
\mathbf{u}_m &= \mathbf{u}_{E,m} + \mathbf{u}_{\mathcal{T}_m}^{opt}
\end{aligned} \tag{5.53}$$

where $\mathbf{u}_{\mathcal{T}_1}^{opt}, \cdots, \mathbf{u}_{\mathcal{T}_m}^{opt}$ are desired trim inputs from trim trajectory specification which is calculated using 5.9, i.e., $\mathbf{u}_{\mathcal{T}_j}^{opt} = (\boldsymbol{B}_a^{opt})^{-1}\tau_{d,\mathcal{T}_j}^{opt}, j = 1, \cdots, m$. The tracking performance will be verified in Matlab/Simulink abd discussed in the next chapter.

5.2.8 Summary

To sum up, the biggest challenge in this work is the coupling between the three different groups of geometric variables. Each term in the robot dynamics (the inertia matrices, the

Coriolis matrices, the damping matrix, the restoring matrix, the input mapping matrix) is determined by several of them. By observing the relationship between each term and these decision variable, see 5.2, 5.3, 5.4, 5.5, 5.6 and 5.7, we find out that the decision variables of the robot hull exist in all terms. Thus, we extract $\mathfrak{d}_H$ from the decision variables and optimize them firstly. This is what we do in the hull optimization part. For the rest of geometric decision variables, i.e., the control inputs, the positions and orientations of all actuators, we adopt an iterative way to decouple three different groups of them. Each time we only optimize one of the these three variable clusters, while the rest two clusters of decision variables are set to be constant. Note that, the spin directions of all thrusters do not take apart in the main iterative optimization loop since they are discrete decision variables. They will be determined separately before the main loop according to the average condition number. This step is called actuator configuration optimization.
Three criteria are selected to break the actuator optimization loop:

- The controllability of arbitrary trim trajectory error dynamics is not satisfied.
- The specified maximal iteration number is exceeded.
- $\frac{1}{m}\sum_{i=1}^{m}\|\boldsymbol{C}_{E,j}^{k}-\boldsymbol{C}_{E,j}^{k-1}\|<\epsilon.$

Very small average norm means the controllability matrix update themselves within a negligible range and the controllability matrix is only determined by the geometric variables, indicating that the geometric variables can not be optimized anymore.
We use the condition number to pick out the most optimal geometric configuration from the results in the very last k_{last} iterations, because a smaller condition number indicates a better ability to handle uncertainties. Meanwhile, the tracking errors show no obvious difference for the very last k_{last} geometric configurations.

Chapter 6

Verification and Simulation

In this chapter, we will implement the designed optimization algorithms from Chapter 5 in MATLAB and verify its correctness by checking whether the robot with optimized geometry can track the specified trim trajectories. The first stage hull optimization algorithm provides us a local optimum of the hull configuration. The conventional robot design specifications will be satisfied in the first design phase. Then we execute the actuator optimization codes on basis of this hull configuration and it will give us a locally optimal configuration of actuators. Two groups of actuator configuration will be exemplarily tested including 6 thrusters and 4 fins with 2 thrusters. We choose these two configurations because the number of actuators is equal to the degrees of freedom. We will visualize the robot hull and the actuator placement in MATLAB to give us an intuitive view of the unconventional actuator placement. The trim kinematic specifications are given in Table 6.1. We expect the robot to track these specified seven trim trajectories in 200 seconds.

It is necessary to check whether the robot can realize the tracking goal of the specified trim trajectories (kinematic specifications) under the current geometric structure. Based on the optimal actuator configuration, we build the robot dynamics and then linearize it in terms of the the seven trim trajectories in the Table 6.1. Then we design the switched LQR control law using the method in Chapter 4 for the seven linearized systems. The trajectory tracking will be simulated with and without actuator saturation in MATLAB/Simulink,

Table 6.1: Trim trajectories specifications

Time(s)	$\|\mathbf{v}_{\mathcal{T}}\|$ (m/s)	$\dot{\psi}_{\mathcal{T}}$ (rad/s)	$\gamma_{\mathcal{T}}$ (rad)
0-40	2	0	0
40-60	2	0.0803	0
60-80	2	0.0803	19.5
80-120	2	0	19.5
120-140	2	-0.0803	19.5
140-160	2	-0.0803	0
160-200	2	0	0

respectively.
Our idea to design the robot based on the trim kinematic specifications can be interpreted as training the robot structure using the trajectory datasets. Using this concept from statistics and machine learning, we perform the third group of simulation where we use one steeper trim trajectory to train the 6 thruster configuration and then test generalization with an untrained trajectory set, i.e. one for which the configuration was not optimized.

6.1 Simulation for 6 Thrusters

For the first simulation testing, we specify that the robot only has 6 thrusters as actuators. We choose the pure thruster configuration mainly due to the following three reasons:

- There exists a special discrete decision variable b_T in the thruster decision variables group $\mathfrak{d}_P$. The spin direction b_T does not take apart in the main iterative optimization loop;
- The input mapping matrix $\boldsymbol{B}_a$ consisting merely of thrusters does not depend on the robot's dynamics states, i.e., it is not trim trajectory-determined;
- The underwater structure with pure thrusters configuration is comparable to the multicopters discussed in [DSZ+16].

In this case, the condition number corresponding to the geometric configuration at the last iteration (68) is the smallest. Thus, we take the geometric values from the last iteration. Figures 6.1 and 6.2 visualize the final result of the thruster locations from the anterior view and posterior view, respectively. The hull is drawn as a hollow cylinder with length l_H and radius r_H which come from the results of the hull optimization. In the hull, the body frame $\{b\}$ is plotted where the red arrow represents its x-axis, the green arrow represents the y-axis and the blue arrow denotes the z-axis. We use arrow line segment to represent the thrusters' geometric information. All of them are unit vectors whose starting point is attached on the hull surface representing their positions $\mathbf{r}_{T,1}, \cdots, \mathbf{r}_{T,6}$ and the arrows denote their orientations $\mathbf{d}_{T,1}, \cdots, \mathbf{d}_{T,6}$.
We set the value of ϵ according to the initial average norm of all trim trajectory error dynamics controllability matrices $\boldsymbol{C}_{E,av,init} = \frac{1}{7}\sum_{j=1}^{7} ||\boldsymbol{C}_{E,j}^{1}||$, here we choose $\epsilon = 0.001\boldsymbol{C}_{E,av,init}$. By using this ϵ value for breaking, the optimization algorithm will break only for 68 iterations, only one third of the specified maximal iterations ($k_{max} = 200$). The norms of the error dynamics controllability matrices are in the order of 10^9, as illustrated in Figure 6.3. They increase roughly in the first four iterations and fall down until the 18-th iteration. After that, the norm values do not change heavily and begin to converge slowly from the 50-th iteration. The positions and orientations of the 6 thrusters are depicted in 6.34. The positions and orientations of all 6 thrusters except for the thruster 2 direction vector $\mathbf{d}_{T,2}$ change obviously in the first 50 iterations and tend to converge from the 50-th iteration

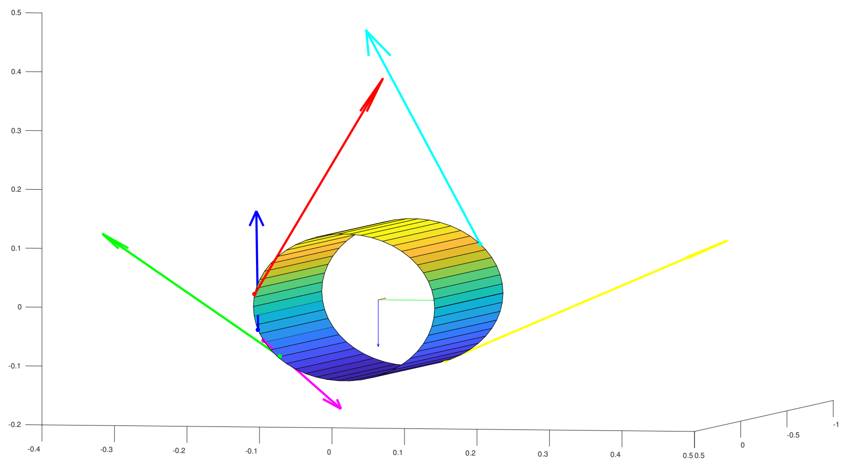


Figure 6.1: Visualization of optimization result (6 thrusters) perspective 1

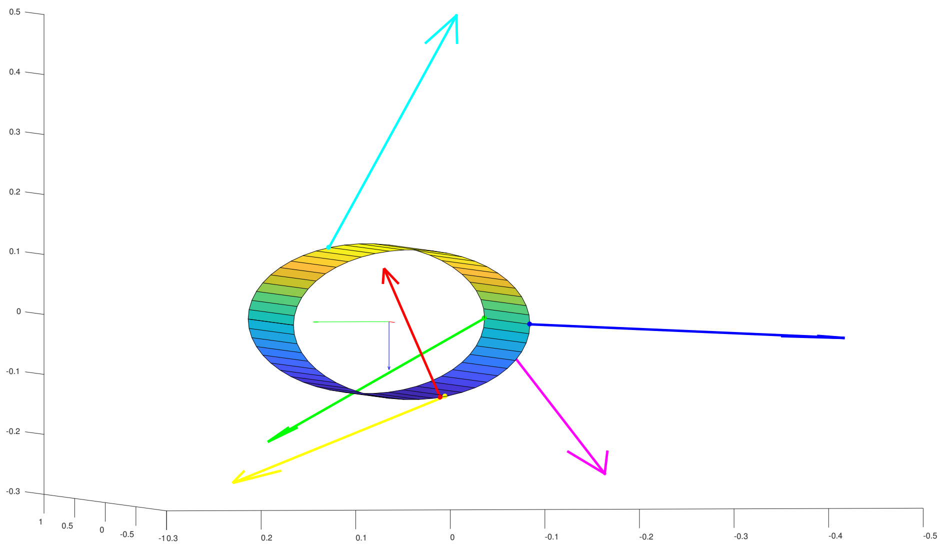


Figure 6.2: Visualization of optimization result (6 thrusters) perspective 2

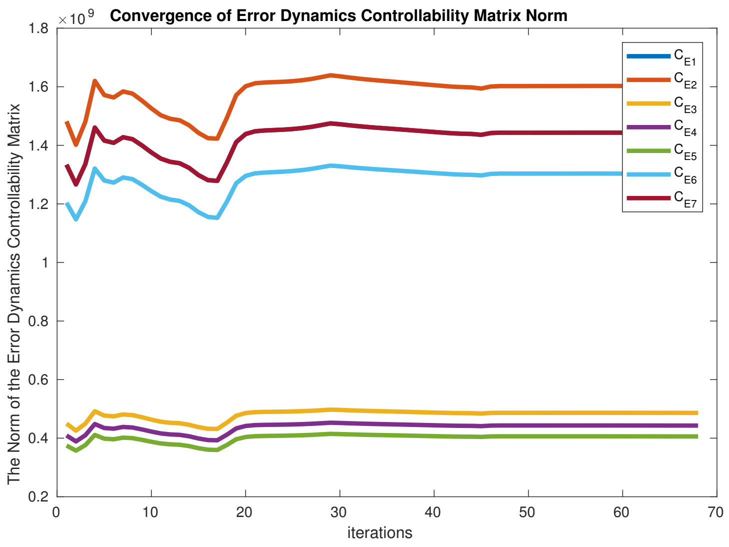


Figure 6.3: Convergence of error dynamics controllability matrix norm (6 thrusters)

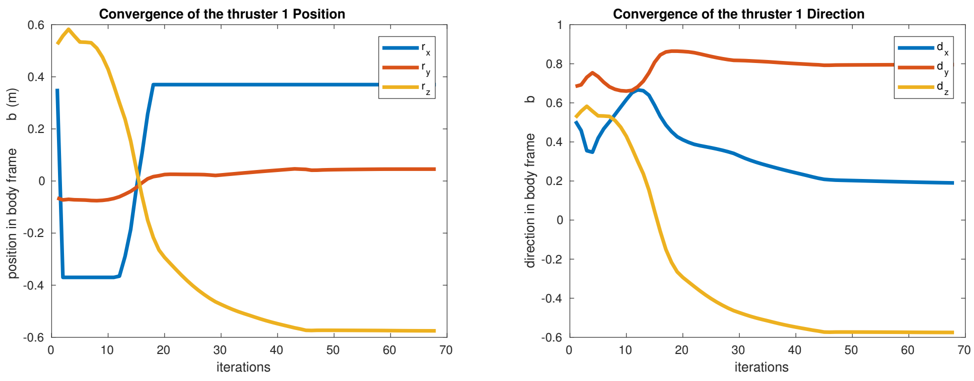


Figure 6.4: Convergence of position and direction of thruster 1 (6 thrusters)

corresponding to the convergence of the controllability matrices norms. Only one direction vector's changing contributes to the norm negligibly, the optimization loop can still be broken.

Recall that we have three optimization steps: control input optimization, position optimization and orientation optimization. As depicted in Figure 6.10, the control input objective function (5.27) value $fval_{input}$ increases from 210 to 380 and then decreases significantly to 50 at the iteration 30. After that, it falls down slightly until the last iteration. The convergence value of $fval_{input}$ is merely one fifths of the initial objective value, namely 40. The position optimization objective value $fval_{pos}$ starts also approximately from 110 and decreases to 10 in 68 optimization loops.

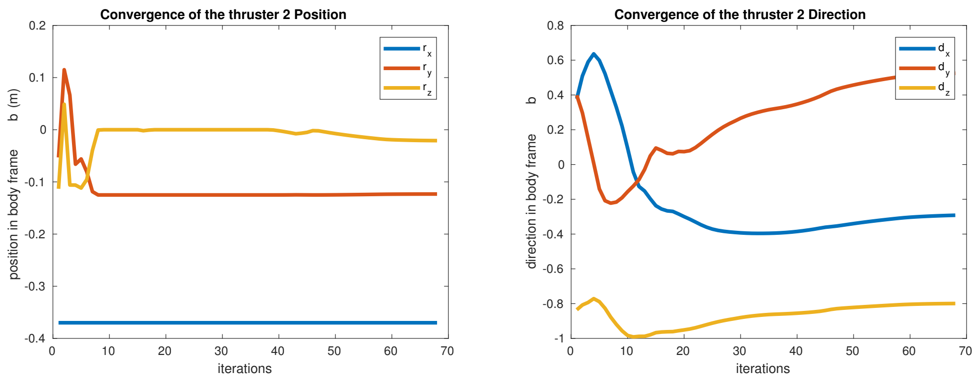


Figure 6.5: Convergence of position and direction of thruster 2 (6 thrusters)

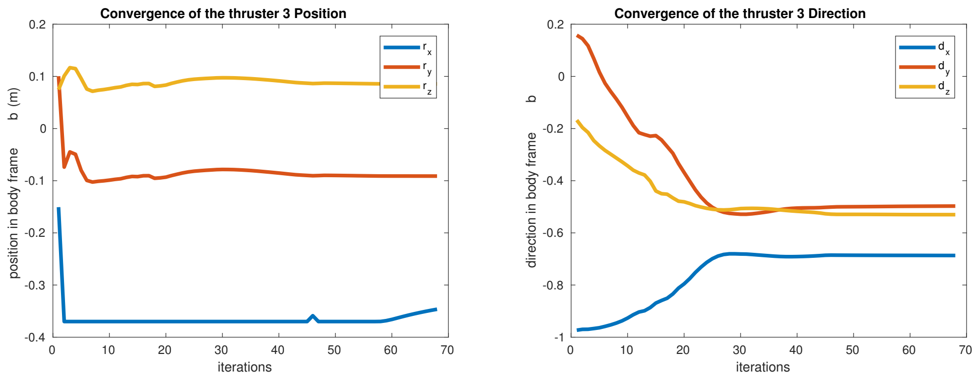


Figure 6.6: Convergence of position and direction of thruster 3 (6 thrusters)

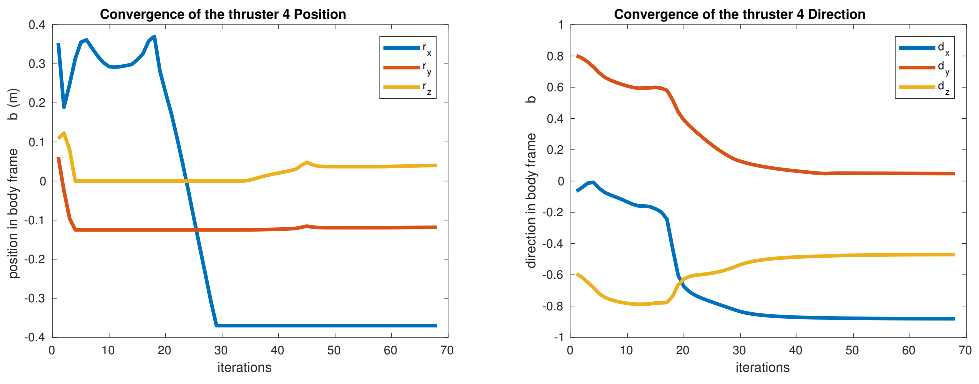


Figure 6.7: Convergence of position and direction of thruster 4 (6 thrusters)

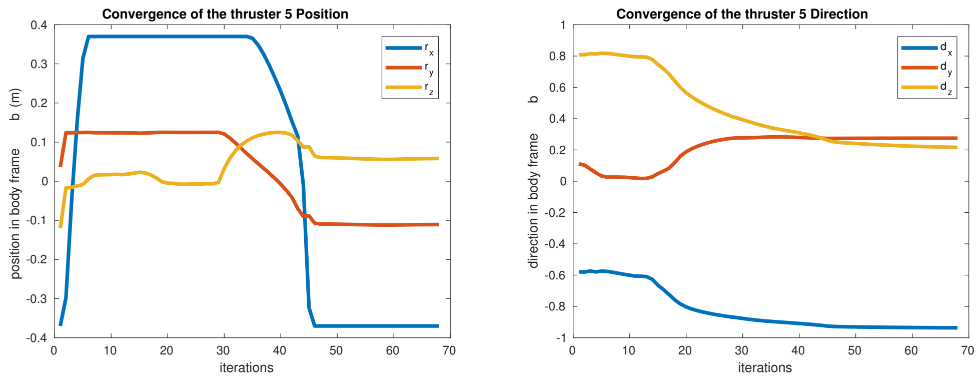


Figure 6.8: Convergence of position and direction of thruster 5 (6 thrusters)

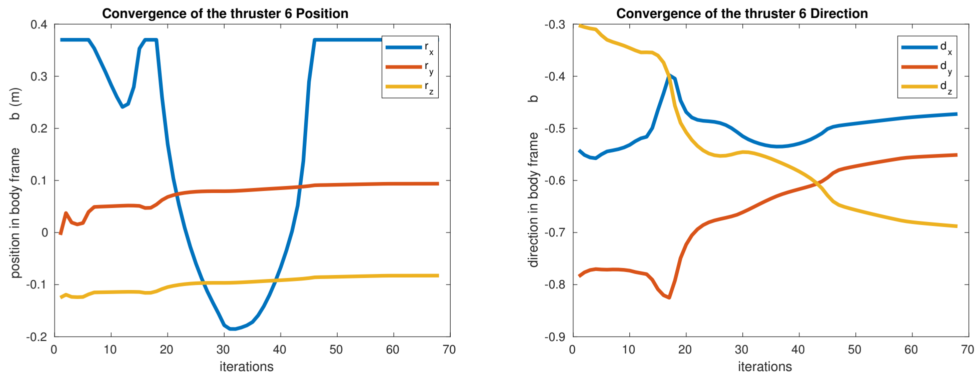


Figure 6.9: Convergence of position and direction of thruster 6 (6 thrusters)

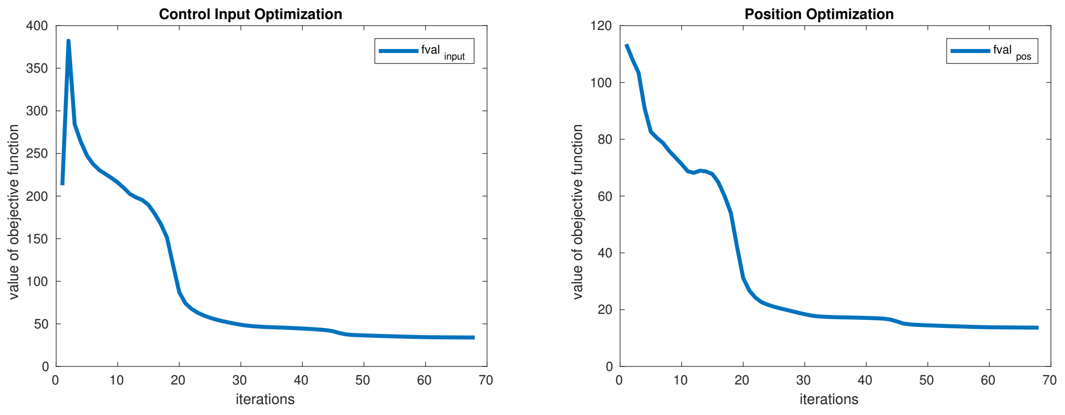


Figure 6.10: Value of objective functions (6 thrusters)

Table 6.2: State cost matrices

Index	Q_E
1	diag([1000,1000,1000,300,300,300,1,1,1,1,1,1])
2	diag([1000,1000,1000,300,300,300,1,1,1,1,1,1])
3	diag([1000,1000,1000,300,300,300,1,1,1,1,1,1])
4	diag([1000,1000,1000,1,1,1,1,1,1,1,1,1])
5	diag([1000,1000,1000,300,300,300,1,1,1,1,1,1])
6	diag([1000,1000,1000,300,300,300,1,1,1,1,1,1])
7	diag([1000,1000,1000,300,300,300,1,1,1,1,1,1])

Table 6.3: Input cost matrices for 6 thrusters

Index	$\mathcal{R}_E$
1	diag([0.001,0.001,0.001,0.001,0.001,0.001])
2	diag[0.001,0.001,0.001,0.001,0.001,0.001]
3	diag([0.0001,0.0001,0.0001,0.0001,0.0001,0.0001])
4	diag([0.0001,0.0001,0.0001,0.0001,0.0001,0.0001])
5	diag([0.0001,0.0001,0.0001,0.0001,0.0001,0.0001])
6	diag([0.0001,0.0001,0.0001,0.0001,0.0001,0.0001])
7	diag([0.001,0.001,0.001,0.001,0.001,0.001])

We build the switched LQR controller using the state cost and input cost matrices from Table 6.2 and 6.3. Under the current configuration, we want the robot to track the specified 7 trim trajectories in 6.1. As illustrated in 6.11, the robot can realize perfect tracking without actuator constraint. When we add the actuator saturation, as depicted in Figure 6.12, the robot is able to keep the trim paths roughly.

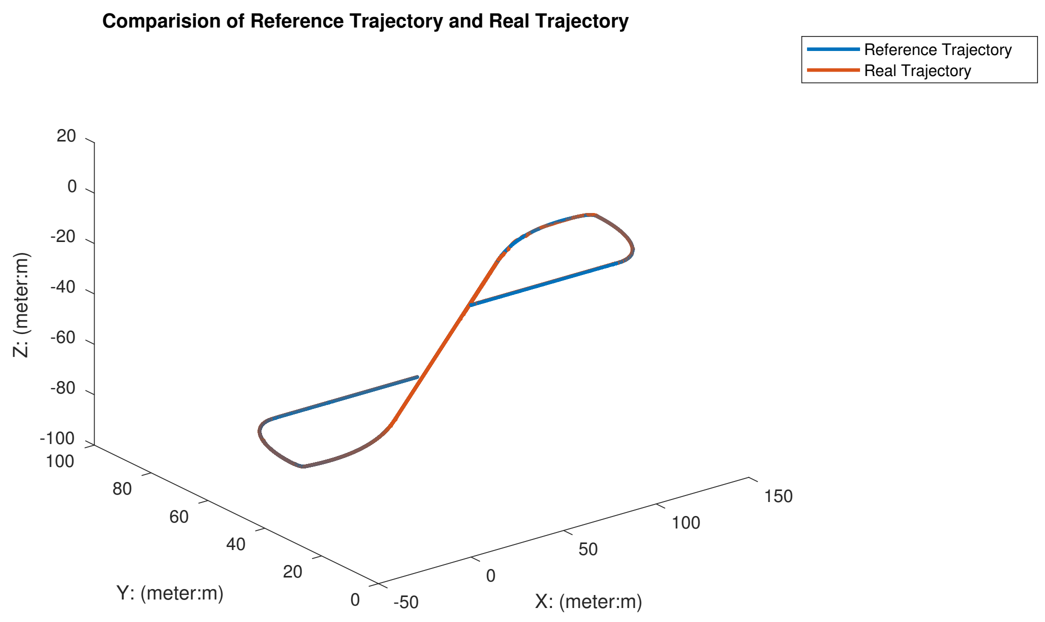


Figure 6.11: Comparison of desired trajectory and real trajectory (6 thrusters, no actuator saturation)

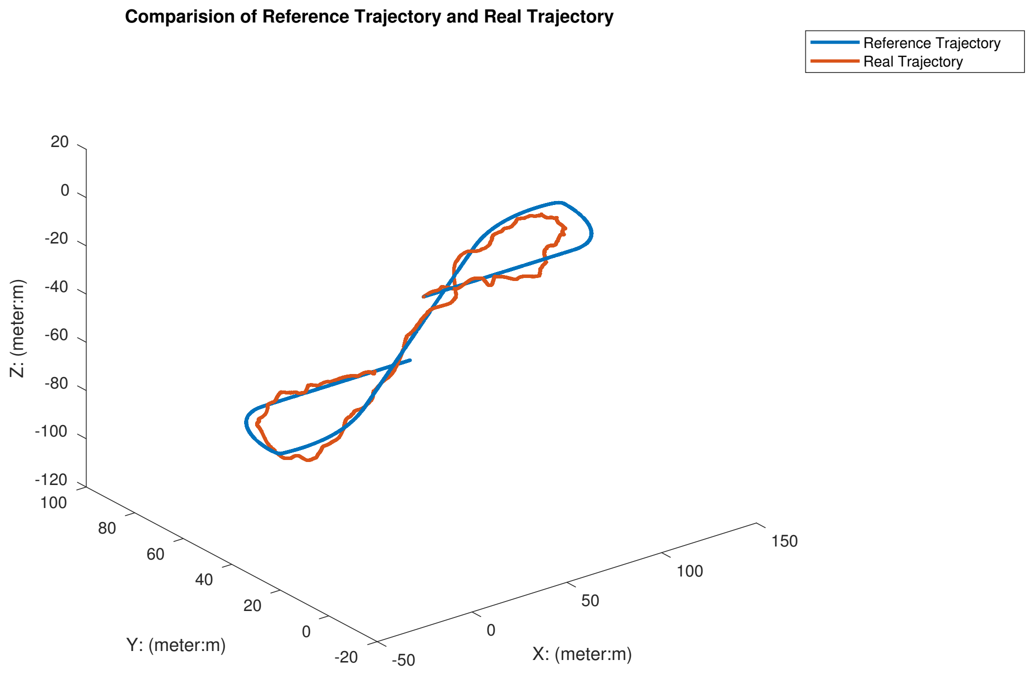


Figure 6.12: Comparison of desired trajectory and real trajectory (6 thrusters, with actuator saturation)

6.2 Training and Test for Thrusters

As mentioned before, in this session, we use a trim trajectory segment as training data for the optimal geometric parameters for the 6 thrusters and another trim trajectory segment for validating. The trim specifications are listed in Table 6.4. The train trim trajectory possess a double trim speed ($||\mathbf{v}_{\mathcal{T}}|| = 2$ m/s) than the test trim segment ($||\mathbf{v}_{\mathcal{T}}|| = 1$ m/s). The yaw rate direction of the train trim is opposed to that of the test trajectory with nearly triple magnitude. The motion path angle $\gamma_{\mathcal{T}}$ of the train trim segment is a little smaller than that of the validation trim segment.

Table 6.4: Trim Trajectory Specifications

Trim Trajectory Property	Time(s)	$\|\|\mathbf{v}_{\mathcal{T}}\|\|$ (m/s)	$\dot{\psi}_{\mathcal{T}}$ (rad/s)	$\gamma_{\mathcal{T}}$ (rad)
Train Trajectory	0-100	2	0.1	0.5
Test Trajectory	100-125	1	-0.35	0.6

General speaking, the train trim trajectory segment is more difficult to be tracked by the robot than the validation trim trajectory segment.

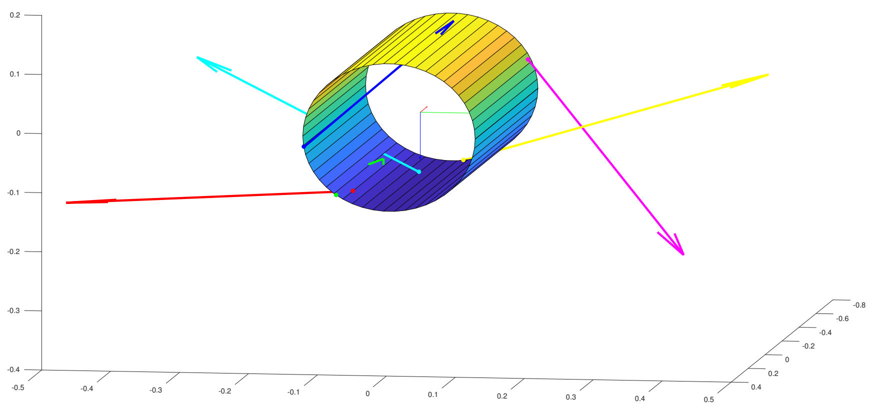


Figure 6.13: Visualization of optimization result (6 thrusters, train and test) perspective 1

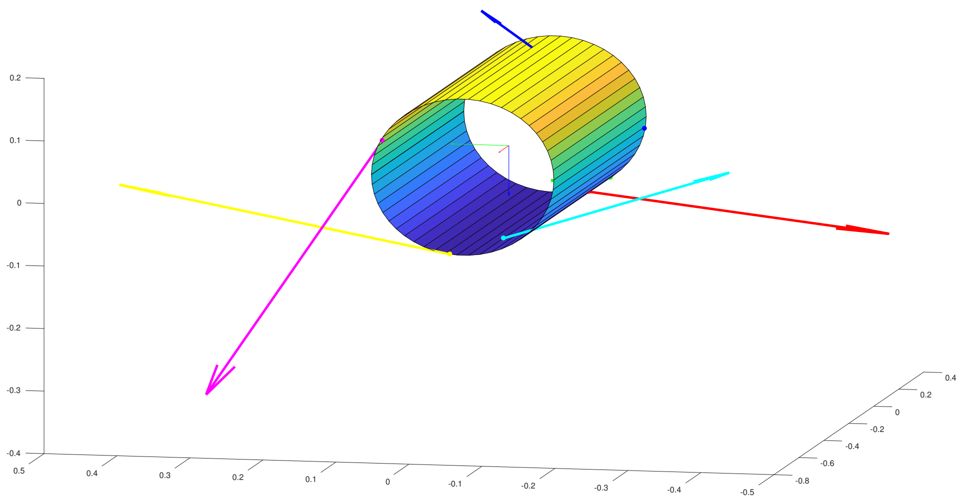


Figure 6.14: Visualization of optimization result (6 thrusters, train and test) perspective 2

Since only one trim trajectory segment need to be tracked, the calculation amount is much less than the case of seven trim trajectories in the previous section. From Figures 6.16, 6.17, 6.18, 6.18, 6.19, 6.20, 6.21, we can see that the positions and orientations of the 6 thrusters converge within only 4 iterations, which corresponds to the variations of the controllability matrix norm $\boldsymbol{C}_{E,1}$ illustrated in Figure 6.15.

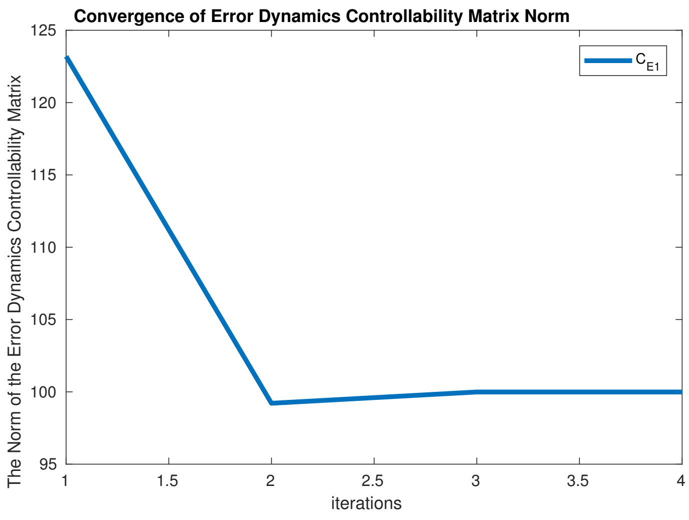


Figure 6.15: Convergence of error dynamics controllability matrix norm (6 thrusters, train and test)

It is notable that the control input optimization error $fval_{input}$ and the position optimization error $fval_{pos}$ can be minimized to zero, which is not achievable for optimization in terms of seven trim trajectories. It is easy to find a actuator geometric configuration so that they can generate the desired trim input $\tau_{\mathcal{T}}$ under saturation constraints for just one trim trajectory. However, when the desired trim trajectory segments increase to seven, a geometric configuration that is able to generate the desired seven trim generalized forces $\tau_{d,\mathcal{T}_1},\cdots,\tau_{d,\mathcal{T}_7}$ under consideration of the actuator capability could not be found by our optimization algorithm.

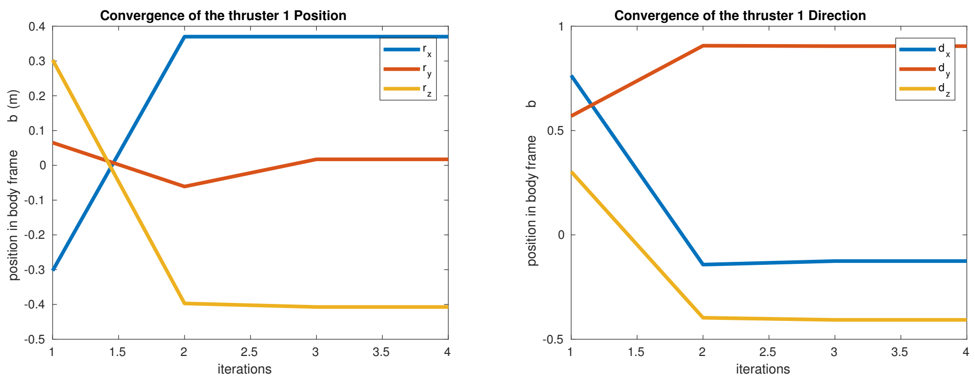


Figure 6.16: Convergence of position and direction of thruster 1 (6 thrusters, train and test)

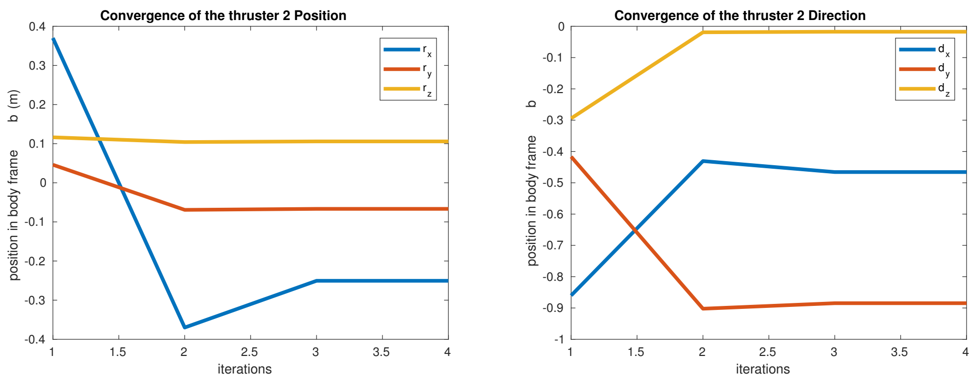


Figure 6.17: Convergence of position and direction of thruster 2 (6 thrusters, train and test)

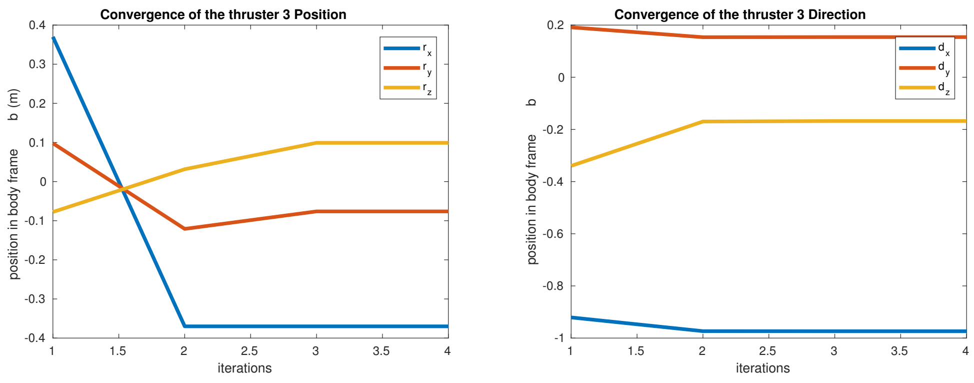


Figure 6.18: Convergence of position and direction of thruster 3 (6 thrusters, train and test)

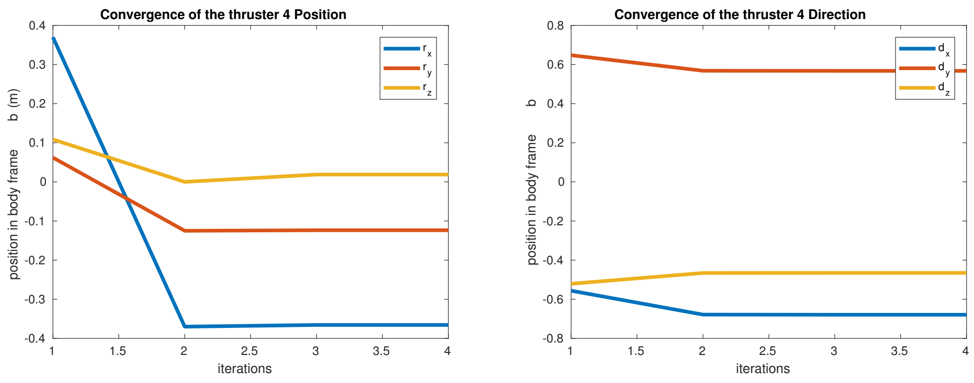


Figure 6.19: Convergence of position and direction of thruster 4 (6 thrusters, train and test)

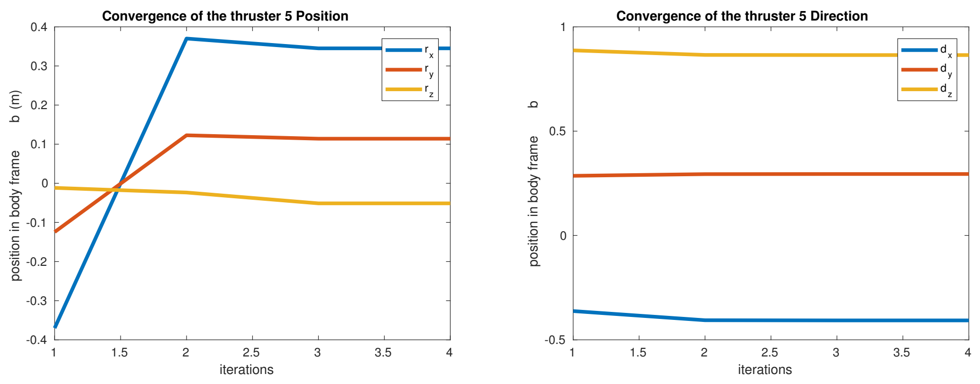


Figure 6.20: Convergence of position and direction of thruster 5 (6 thrusters, train and test)

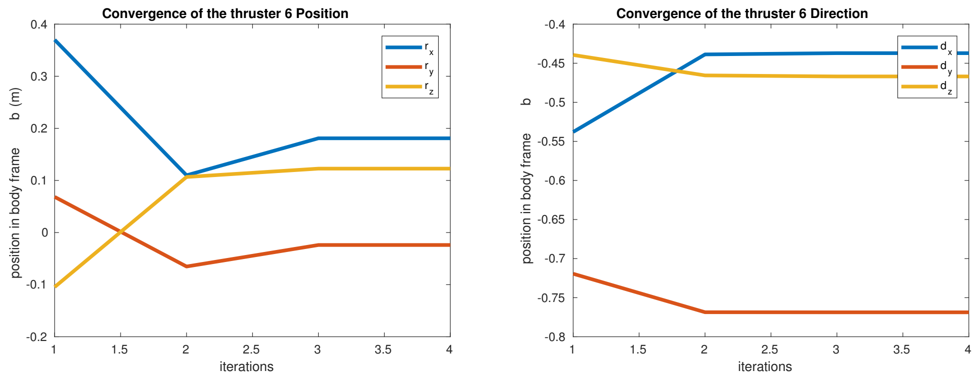


Figure 6.21: Convergence of position and direction of thruster 6 (6 thrusters, train and test)

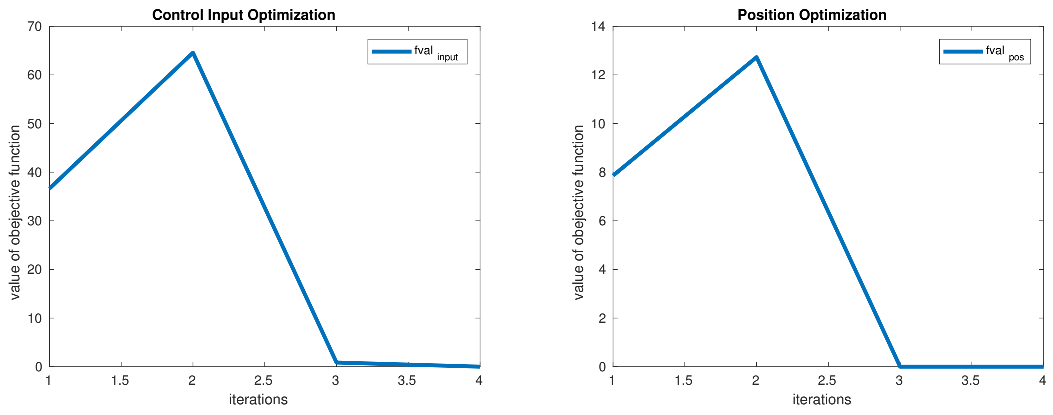


Figure 6.22: Value of objective function (6 thrusters, train and test)

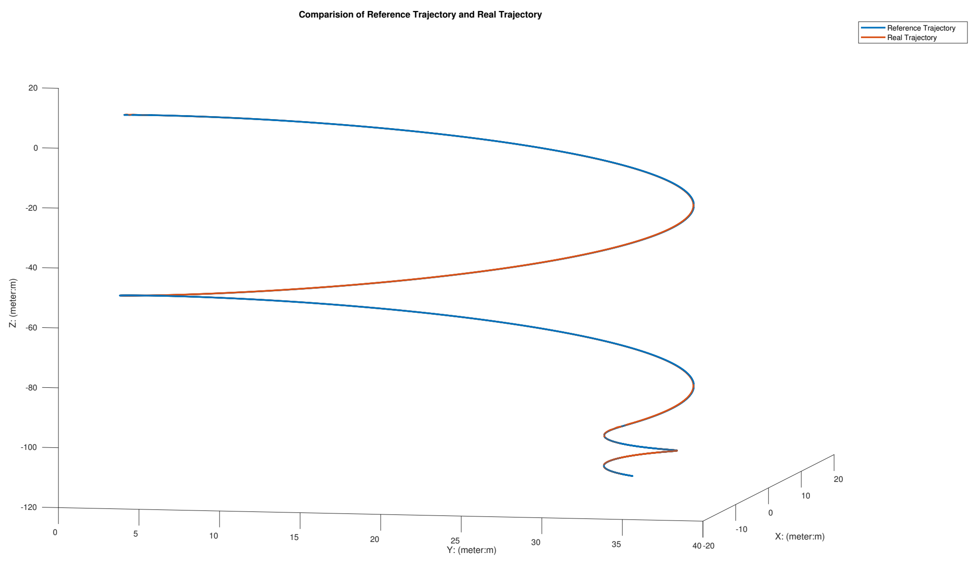


Figure 6.23: Comparison of desired trajectory and real trajectory (6 thrusters, no actuator saturation, train and test)

As illustrated in 6.23, the trained robot from the first trim trajectory segment track both trim paths perfectly when the capability of the thrusters are not taken into consideration. The Figure 6.24 depicts the tracking performance when we specify the maximal and minimal thrust of the thrusters as $[-35, 40]$ N. The tracking deviation from the desired train trim trajectory is bigger than that from the test trim trajectory due to its higher trim speed and yaw rate.

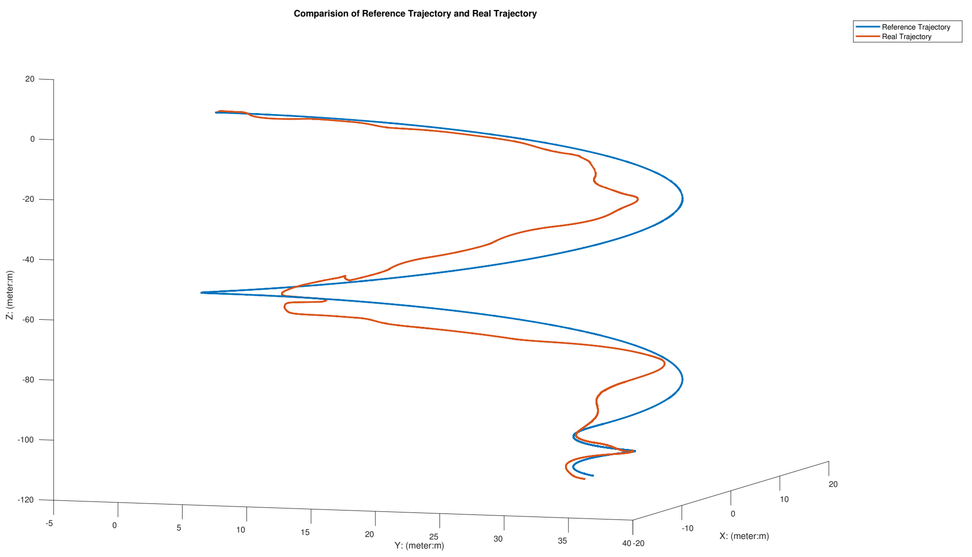


Figure 6.24: Comparison of desired trajectory and real trajectory (6 thrusters, with actuator saturation, train and test)

6.3 Simulation for 4 Fins and 2 Thrusters

Compared with pure thrusters configuration, the placement of fins deserves further study since they are trajectory dependent. Not only the geometric variables concerning fins but also the trim specifications will contribute to the generated generalized force. Assume that the robot now has 4 fins and 2 thrusters to control its motion. In this case, we also set $\epsilon = 0.001 \times \frac{1}{7} \sum_{j=1}^{7} ||\boldsymbol{C}_{E,j}^{1}||$.
As depicted in Figures 6.25 and 6.26, the fins allocation in this case is quasi-symmetrical, two fins are located on the right front hull and other two are located on the left tail part of the hull. One thruster points to the right front side while other points to the left front side. From Figures 6.27, 6.28, 6.29 and 6.30, we can see that the positions $x_{F,1}$, $x_{F,2}$, $x_{F,3}$ and $x_{F,4}$ stay constant after about 38 iterations, while the orientations keep on changing. It can be seen from Figures 6.31 and 6.32 that the positions and orientations of thruster 1 and thruster 2 converge after about 50 iterations.

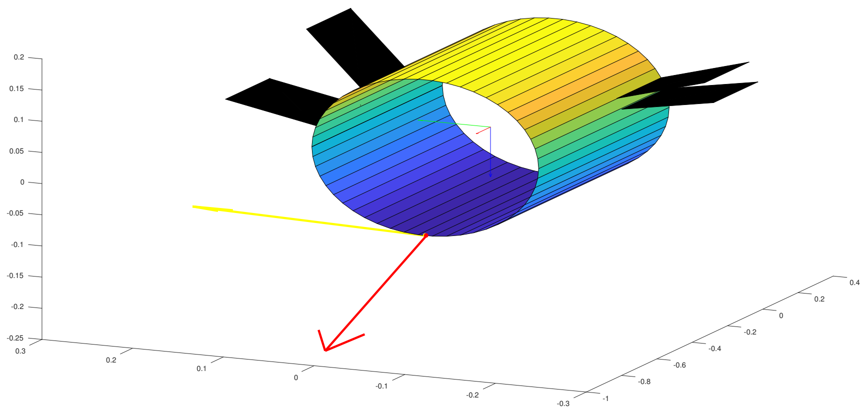


Figure 6.25: Visualization of optimization result (4 fins and 2 thrusters) perspective 1

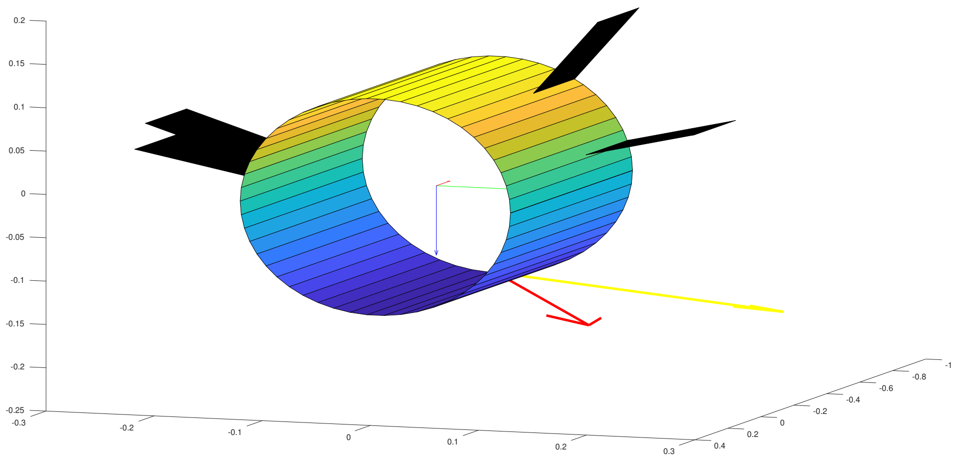


Figure 6.26: Visualization of optimization result (4 fins and 2 thrusters) perspective 2

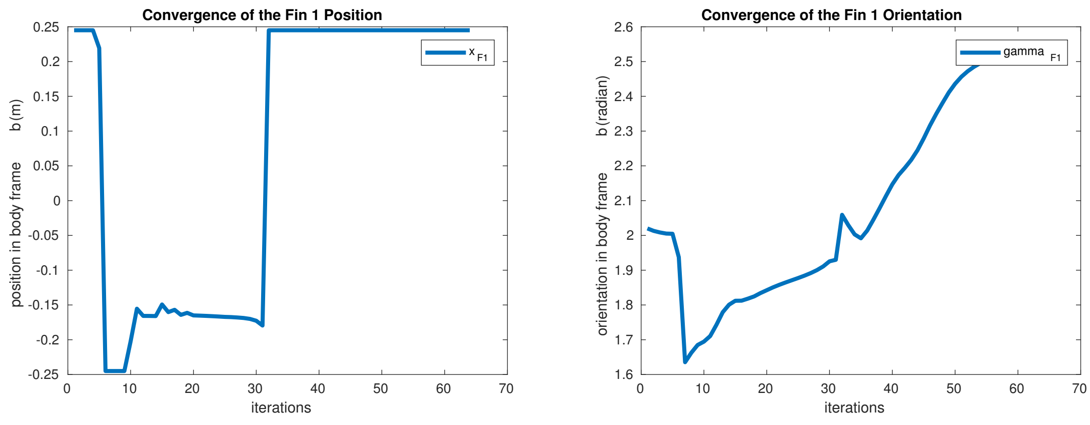


Figure 6.27: Convergence of position and orientation of fin 1 (4 fins and 2 thrusters)

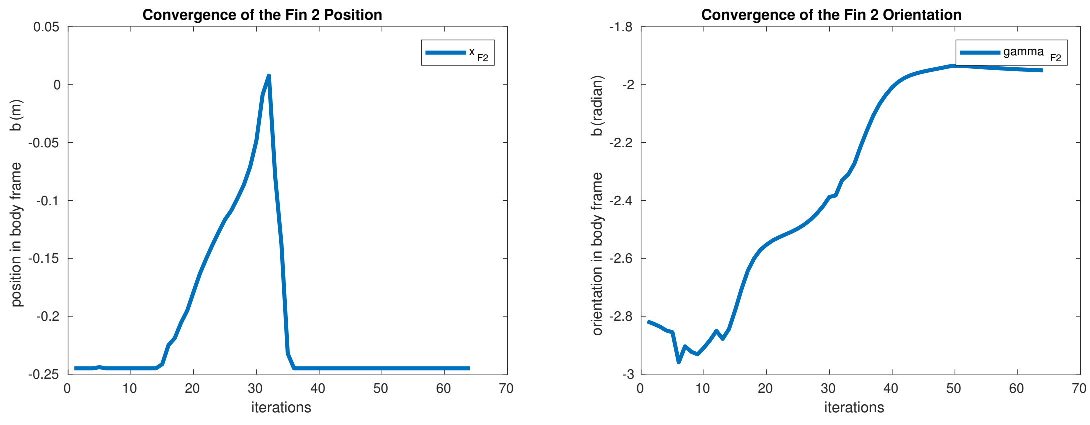


Figure 6.28: Convergence of position and orientation of fin 2 (4 fins and 2 thrusters)

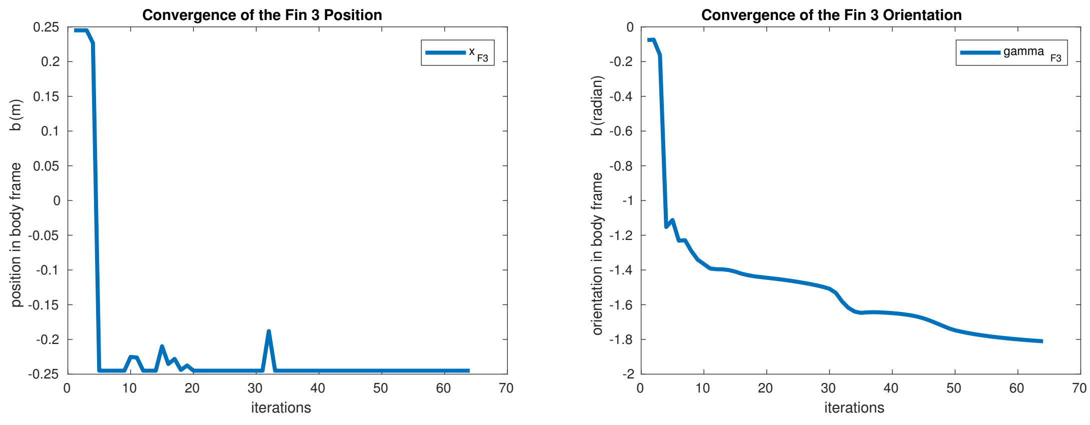
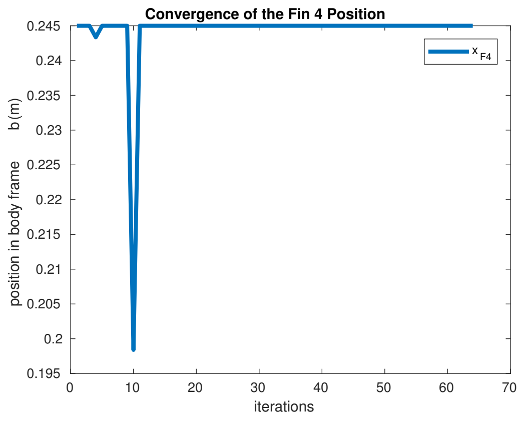


Figure 6.29: Convergence of position and orientation of fin 3 (4 fins and 2 thrusters)



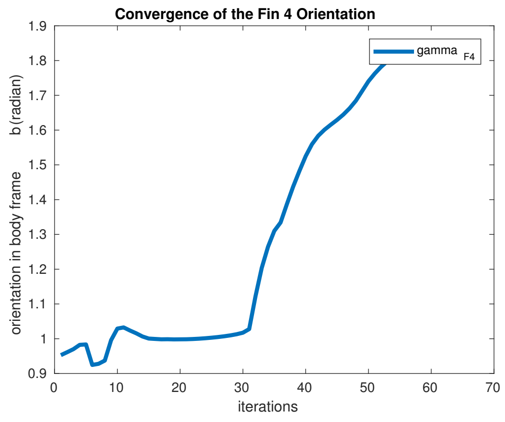
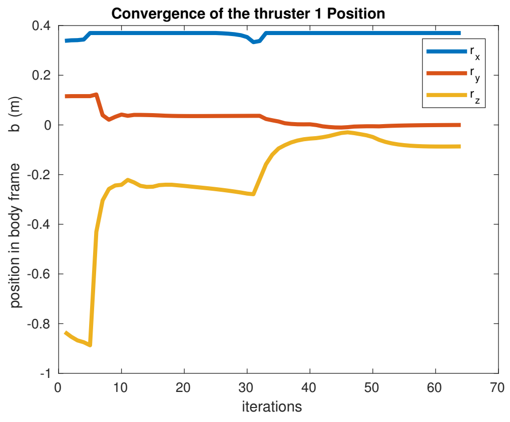


Figure 6.30: Convergence of position and orientation of fin 4 (4 fins and 2 thrusters)



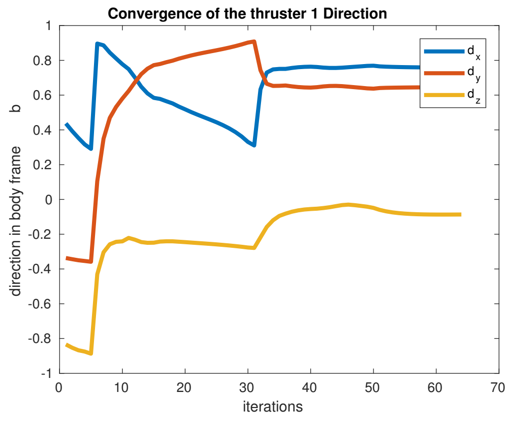
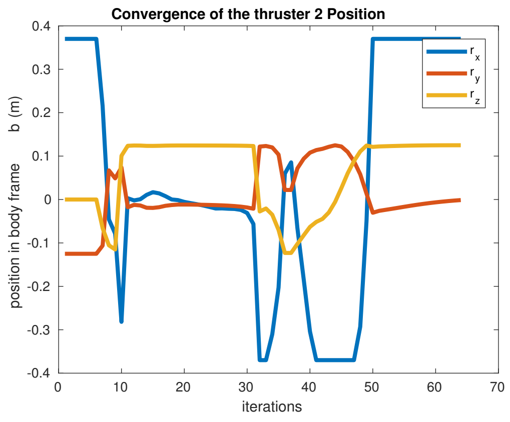


Figure 6.31: Convergence of position and direction of thruster 1 (4 fins and 2 thrusters)



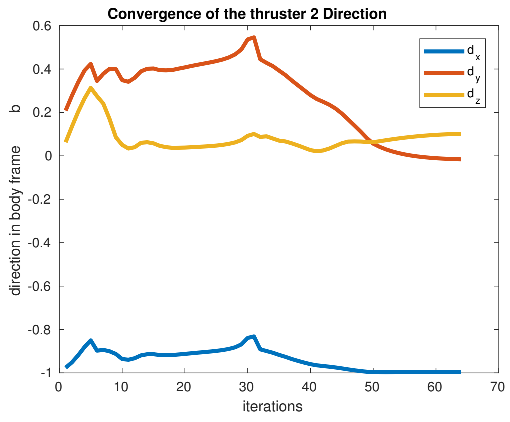


Figure 6.32: Convergence of position and direction of thruster 2

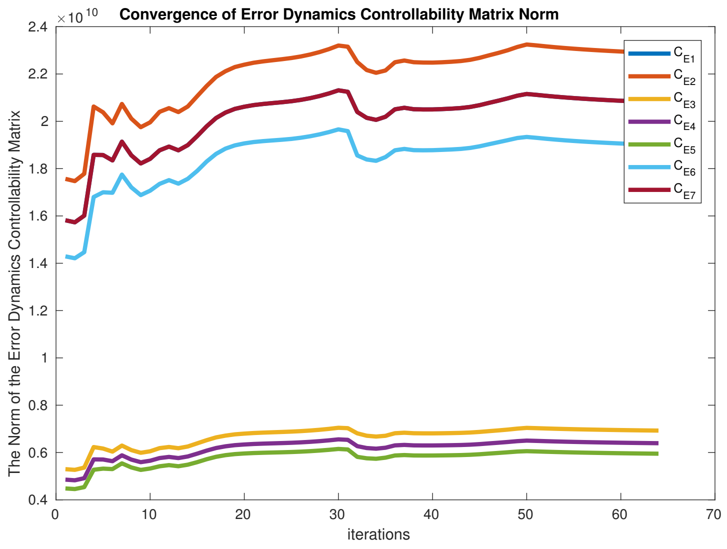


Figure 6.33: Convergence of error dynamics controllability matrix norm (4 fins and 2 thrusters)

The value of the control input objective function $fval_{input}$ increases at first and then falls down sharply from the 5-th iteration to the 10-th iteration. Afterwards, it restarts to increase until the 30-th iteration and then falls down again. The position optimization objective function $fval_{pos}$ rises sharply in the beginning but decreases quickly to about 50.

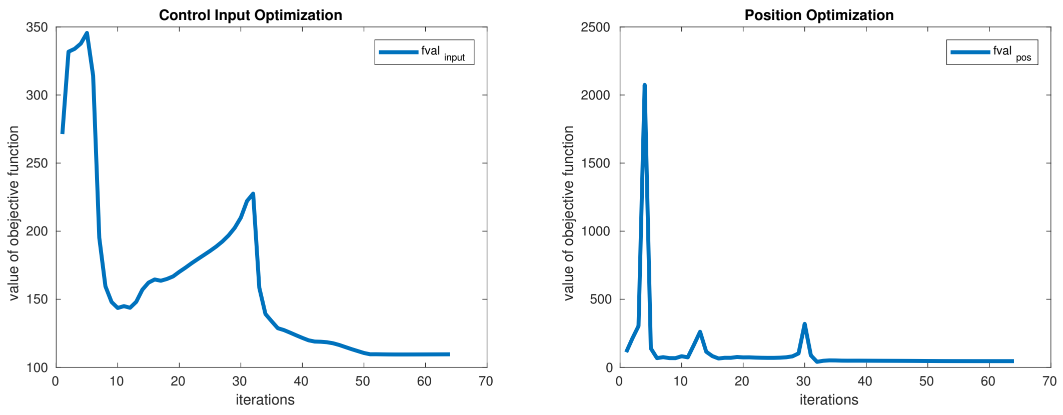


Figure 6.34: Value of objective function (4 fins and 2 thrusters)

Table 6.5: Input cost matrices for 4 fins and 2 thrusters

Index	$\mathcal{R}_E$
1	diag([0.0001,0.0001,0.0001,0.0001,0.0001,0.0001])
2	diag[0.00001,0.00001,0.00001,0.00001,0.00001,0.00001]
3	diag([0.00001,0.00001,0.00001,0.00001,0.00001,0.00001])
4	diag([0.00001,0.00001,0.00001,0.00001,0.00001,0.00001])
5	diag([0.00001,0.00001,0.00001,0.00001,0.00001,0.00001])
6	diag([0.00001,0.00001,0.00001,0.00001,0.00001,0.00001])
7	diag([0.001,0.001,0.001,0.001,0.001,0.001])

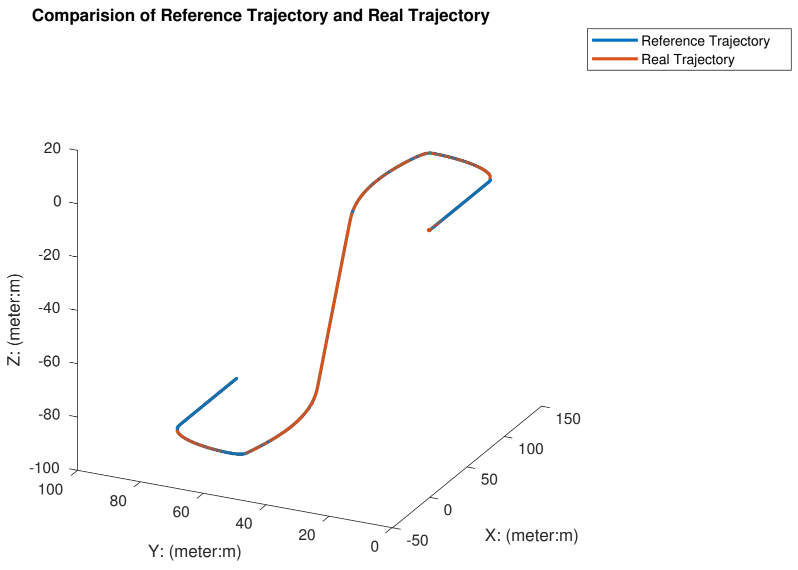


Figure 6.35: Comparison of desired trajectory and real trajectory (4 fins and 2 thrusters, no actuator saturation)

We use the state and input cost matrices in Table 6.2 and Table 6.5 to design the switched LQR controller. Note that in this case smaller input weighting matrices are chosen meaning larger control inputs are needed to track the desired trajectory. As depicted in Figure 6.35, the robot with current optimized realize perfect tracking without actuator capability constraints with same weighting matrices. When the thrust is restricted to $[-35, 40]$ N and the deflection is constrained in $[-\pi, \pi]$ rad, as depicted in Figure 6.36, the real trajectory preserves the fundamental characteristic of the reference trajectory roughly, although the deviation is still large.

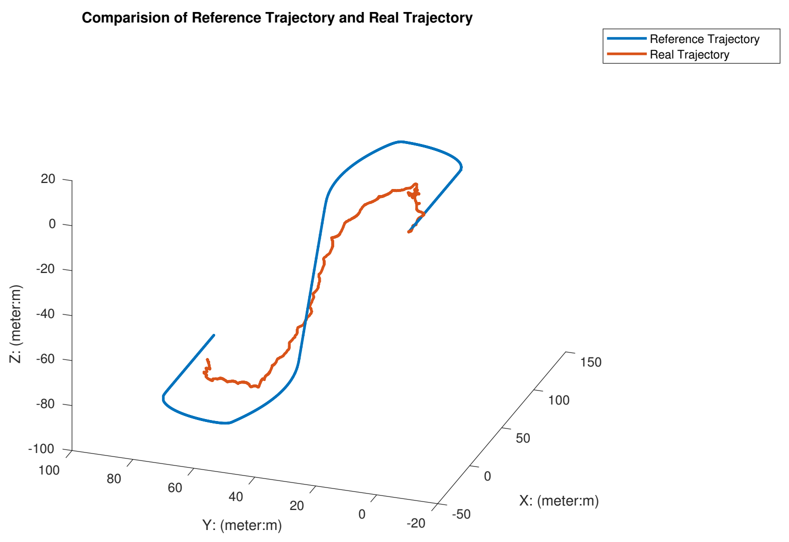


Figure 6.36: Comparison of desired trajectory and real trajectory (4 fins and 2 thrusters, with actuator saturation)

6.4 Summary

It can be concluded from these simulation results that our optimization algorithm is able to find locally optimal geometric configurations depending on the randomly generated initial values. By means of the multi-stage iterative optimization in the algorithm, the control input objective function and the position objective function can also be minimized to a small value but not zero. Normally, the geometric configuration at the breaking iteration corresponds to the least average condition number and thus is chosen as optimal value.
Without consideration of actuator's capability, a nearly perfect tracking is realizable for different actuator configurations by tuning input and state weighting matrices appropriately. However, in realistic situations, the robot with only thrusters tracks the desired trim trajectories unstably while the robot using fins with thrusters as actuators perform worse with considerably large tracking error. For LQR controller, the limitations of actuators are not taken into consideration which influence the tracking performance of the underwater robots heavily. Thus, it is recommendable to choose switched Model Predictive Controller (MPC) to control the robot. The main differences between MPC and LQR are that LQR optimizes in a fixed horizon whereas MPC in a receding horizon and that the optimal control input is calculated at each step where as LQR uses the single optimal value for the whole horizon. Hence, MPC allows real-time optimization against hard constraints [Wan09] which can enhance the tracking performance of the underwater robots.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, we introduce an approach for designing underwater robots computationally by formulating the design procedure as multi-stage iterative optimizations according to design specifications. The distinguishing novelty of our work is that we obtain design requirements from the desired trajectories. In addition, Frenet-Serret frame is selected such that the surge velocity is tangent to the trim trajectory curve and direction of other velocities are consequently specified.
We use trim trajectories to represent the robot's motion paths since the robot moves stably, i.e., all 12 states except for the yaw angle ψ are constant. Another decisive reason for selecting trim trajectories is that we can build the error dynamics using nonlinear tranformations 4.28, 4.29, 4.30 and 4.31 whose linearization is unique. The nonlinear underwater dynamics can be transformed into MIMO linear system parametrized by trim trajectory specifications. In this way, a number of analysis and design methods for linear systems can be implemented. More importantly, trajectories' information is brought into the robot dynamics. Therefore, it is reasonable to state that the underwater robot geometry is defined by the trim trajectories.
Besides, the conventional underwater robot design requirements (buoyancy neutral, surge velocity maximization, energy storage maximization, cost minimization and compact structure) must also be taken into consideration.
Because we deign the underwater robot in an iterative way, the robot geometry will be updated continuously. Therefore, an customizable robot dynamics determined by these geometric parameters is aimed for. Based on this requirement, we propose a modular modeling method. All consisting modules of the robot will be represented by simple geometric shapes determined with several geometric variables so that we can construct and reconfigure the robot easily. These geometric variables will be identified as decision variables and consequently the robot dynamics can be parametrized by them.
By studying the relationship between the robot dynamics and the geometric decision variables we find that there are strong couplings among these geometric parameters.
The main approach to obtaining a feasible solution of an optimal robot geometry is to

decouple these decision variables. Since the hull parameters couple with all other decision variables, the first step is to decouple the hull decision variables. This motivates us to separate the hull optimization and the actuator placement optimization. The hull parameters will be determined in the first phase and set as constant for further optimizations steps. The spin directions are discrete decision variables, thus they are determined in the beginning of the actuator optimization according to the average error dynamics condition number and separated from positions and orientations parameters. The positions and orientations of fins and thrusters as well as the control inputs (thrust forces and deflection angles) are decoupled in an iterative way. We optimize the control inputs at the first stage and keep the positions and orientations as unchanged. Similarly, the optimization of positions are based on the assumption that the control inputs and the orientations are constant. At the third optimization stage of orientation, the control inputs and positions are fixed. By means of these iterative multi-stage optimization approaches the decision variables are totally decoupled.
As we obtain a set of MIMO linearized error dynamic systems from the trim trajectories, we are able to formulate the total system as a linear switched system. Consequently, we utilize the switched LQR controller for our system. All input and state weighting matrices influence the stability of the system together. Thus, a method to select their values is proposed in this work based on the stability analysis by common Lyapunov function.
Finally, it can be concluded from the simulation results in Chapter 6 that the users are able to utilize our designing procedure to design an underwater robot prototype with locally optimized geometry adapting to the training trim trajectories. With the optimized actuator configuration the underwater robot system is controllable and can track the desired paths stably using the switched LQR controller. However, if the actuator capabilities are taken into account, the switched LQR controller works unsatisfactorily and the robot could not track the desired trajectories accurately.

7.2 Limitations and Future Work

This is a pioneer research work, thus we make a list of assumptions and there exists limitations which can be rectified and optimized for the future works.

7.2.1 Prototype Modeling

The accuracy of customizable modelling of underwater robots plays an essential role for the result of the optimization algorithms. However, due to the nature of nonlinearity in the robot dynamics, there exists couplings among the geometric decision variables. In our work, we adopt a series of assumptions to obtain a feasible solution.
The hydrodynamic effects are the main source for nonlinearities. We simplify the calculation by assuming only the robot hull contributes to the hydrodynamic effects. However, it is not true in reality.
The robot hull is also influenced by the lift force but we neglect it. Fins are modeled as

rectangular plate whose added mass should also be calculated and transformed into the robot dynamics. Also, fins affect the hydrodynamic damping coefficients. Furthermore, as a significant component of the robot prototype, in this thesis, we pay attention on thruster's functionality to generate forces and moments but ignore their contributions to hydrodynamic effects as a geometric shape. According to [Tan99], the thruster housings can also be modeled as hollow cylinder similar to hull enclosure and the water bodies within the thruster can be modeled as solid cylinders like the batteries. Both of them should be considered for added mass coefficients and damping coefficient estimation in future work.

By calculation of the robot's total moment of inertia, we merely take the hull components (hull enclosure, batteries and electronic devices) into consideration. For a more accurate modeling , the actuator moment of inertia should also be considered. Note that the moment of inertia for actuators is given with respect to their own center of mass. It should be transformed into the robot body frame $\{b\}$ using equations 2.40, 2.41, 2.42 whose values are determined by their position in the body frame. Thus, in the main optimization, not only the center of gravity $\mathbf{r}_G$ but also the robot moment of inertia $\boldsymbol{I}_g$ should be updated after the geometric decision variables $\mathbf{r}_T$, x_F and γ_F are optimized. Both $\mathbf{r}_G$ and $\boldsymbol{I}_g$ affect the rigid body inertia matrix $\boldsymbol{M}_{RB}$ (see A.71).

The hydrodynamic modeling for fins in our work is only viable for very small angle of attack α. The lift and drag coefficients C_L and C_D are assumed to be constant in this case. However, the angle of attack α affects them heavily. High order polynomial might be used to approximate the relationship between the attack angle α and the hydrodynamic coefficients C_L and C_D. In Appendix C, an accurate hydrodynamic modeling is discussed. Besides, in our actuator optimization algorithm, the drag of fins is not treated as control input like the lift but as disturbance. In order to enhance the optimization quality, a more suitable fin modeling is desired.

Recall the thruster modeling in the Section 2.2.1, the moments generated by thrusters is composed of two parts: the rotation moment vector $\mathbf{m}_{T,r} = b_T \lambda_T u_T \mathbf{d}_T$ and the thrust moment $\mathbf{m}_{T,t} = \mathbf{r}_T \times u_T \mathbf{d}_T$. Strictly speaking, both of them are modeled inaccurately. For the rotation moment, we ignore the dynamics of the propeller and assume the rotation moment is proportional to the thrust. Furthermore, the thrust moment $\mathbf{m}_{T,t}$ should be calculated with respect to the robot center of gravity CG instead of the body frame origin CO. Therefore, the thruster moment should be calculated in an accurate way as:

$$\mathbf{m}_{T,t} = (\mathbf{r}_T - \mathbf{r}_G) \times u_T \mathbf{d}_T, \tag{7.1}$$

where $\mathbf{r}_G$ denotes the robot center of mass and $\mathbf{r}_T$ is the position the thruster in the body frame $\{b\}$. However, r_G is updated at each optimization loop and computed from $\mathbf{r_T}$, x_F and γ_F. Consequently, it brings coupling between $\mathfrak{d}_P$ and $\mathfrak{d}_F$ which is not expected. Therefore, we assume the center of gravity CG is located in the body frame origin CO for calculating the thrust moment. Nevertheless, for the future work these two simplified modelings can be made accurately.

7.2.2 Optimization Algorithms

To make the optimization problems solvable, we perform a number of assumptions to decouple the geometric decision variables. To decouple the underwater robot dynamics, we designed the optimization for the hull and for the actuators separately. From the simulation results, we observe that, not only the thrusters but also the fins tend to be located at the ends of the hull, which inspire us that the increase of the hull length can minimize the objective function. Thus combining the optimization of hull size and the optimal actuator allocation will be the further step to enhance the optimal geometry.

Besides, for our case, the actuators combination is fixed for our optimization. It would also make sense to research different actuator combinations and choose the optimal one under fixed total actuators.

For our optimization, we just choose the important geometric variables. Actually, we can select more decision variables: the fin size a_F and b_F can be optimized.

The characteristics of the thruster (especially the maximal thrust force) also influence the designed robot. From the simulation of trajectory tracking with actuator saturation we know that the bigger the actuator's maximal capability is, the better the robot tracks the desired trajectories. Choosing thrusters with bigger maximal thrust is opposed to our power and cost consumption minimization goals. Therefore, we can also set the type of thrusters as a discrete decision variable.

Minimizing the power consumption is very significant for actuators. For the control allocation problem, there is always a quadratic term penalizing the power consumption. The power consumption could be formulated as a polynomial function of the control input. Thus, one possible extension of our work is that we add a term concerning the power consumption in the first control input optimization phase. Tracking the trim trajectory accurately with least control efforts is aimed for.

In our designed algorithm, all trim trajectory segments are equally important. From the simulation results we see that instabilities and oscillations occurs usually along trim trajectories with larger motion path angle $\gamma_{\mathcal{T}}$ and larger yaw rate $\dot{\psi}_{\mathcal{T}}$. Adding weighting for each trim trajectory might also make sense. In [DSZ+16], the desired force and the desired moments are weighted with different coefficients in the objective function. It can probably be used for our optimization objective functions as well.

Since the underwater robot geometry is defined from the trim trajectories, how to choose a reasonable set of trim trajectory should also be studied. If we have a large set of trim trajectories, statistical techniques, e.g., random split, k-fold crossvalidation, bootstrapping, might be used to process the trim trajectory data set.

In our work, we use the the condition number of the linearized error dynamics controllability matrix as a measure for the controllability. There are also other parameters measuring the controllability. According to [Moo81], the smallest singular value of the controllability gramian can be used as the measure of controllability. In [JK91], Kim et al. introduce new measure of controllability for linear time invariant dynamical systems, especially to guide the placement of actuators to control vibrating structures. The matlab function *gram* in the Control System Toolbox can be used to compute the controllability gramian for stable systems. However, the linearized error dynamic systems are always unstable in this work. The switched system need to be stabilized by the designed controller and this approach is not suitable. The method for calculating the controllability gramian for unstable system can be found in [NS04, ZSW99] as long as the system matrix $\boldsymbol{A}_E$ has no imaginary axis poles. The error dynamic system matrices $\boldsymbol{A}_{E,1}, \cdots, \boldsymbol{A}_{E,7}$ in Chapter 6 (6 thrusters and 4 fins with 2 thrusters) possess always imaginary poles. Thus, the aforementioned is not suitable for our systems, either. However, if we extract the kinematic error states (position and Euler angles errors) from the 12 dimensional error dynamics system, the subsystem concerning only kinematic errors does not possess imaginary poles. Hence, the method from [NS04, ZSW99] can be implemented. This deserves further study.

7.2.3 Controller Design

For our work, designing a suitable controller with consideration of strict constraints is still a challenge. Our robots possess generally more complicated dynamics than the standard underwater robots. We are able to derive m linear systems with 12 states from m trim trajectories. A linear MIMO switched system is obtained. This motivates us to simplify the controller design process by using switched LQR control and automatically solving its parameters from our optimization results. However, one obvious defect is of the LQR controller is that it does not take the actuator constraints into account. As a result, the robot can not realize stable tracking. It is recommendable to adopt switched MPC controller for our linear switched error dynamics system since it incorporates constraints explicitly.

Appendix A

Dynamic Model of Underwater Robots

An accurate model of the underwater robot dynamics plays a decisive role for designing controller for underwater robots. Especially for the computational design of the underwater robots, the vehicle dynamics is varying in each optimization loop, because the geometric variables are always updated and consequently the parameters in the robot dynamics are changed. Therefore, it is necessary to identify all robot dynamic parameters and derive them systematically. In this chapter, we firstly focus on the kinematics of underwater robots, i.e., on the translational and rotational motions of underwater robots without consideration of causes of motions. Then hydrodynamic effects (added mass and hydrodynamic damping) and the restoring forces (gravity and buoyancy) are taken into consideration to derive the dynamic model for underwater robots.
For all the variables relating to underwater robots (position, velocity, acceleration, forces, moments), we use the following standard representation as depicted in Table A.1.

Table A.1: Standard notation of underwater robot

translational motion	***position***	***linear velocity***	***force***
surge (motion in x-direction)	x	u	X
sway (motion in y-direction)	y	v	Y
heave (motion in z-direction)	z	w	Z
rotational motion	***Euler angle***	***angular velocity***	***moment***
roll (rotation about x-axis)	ϕ	p	K
pitch (rotation about y-axis)	θ	q	M
yaw (rotation about z-axis)	ψ	r	N

A.1 Kinematics of Underwater Robots

A.1.1 Reference Frames and Basic Notations

A world inertial frame $\{i\} = (x_i, y_i, z_i)$ describes the environment of the underwater robot which is fixed in space. Stationary objects like the walls of basin, which enter the fluid

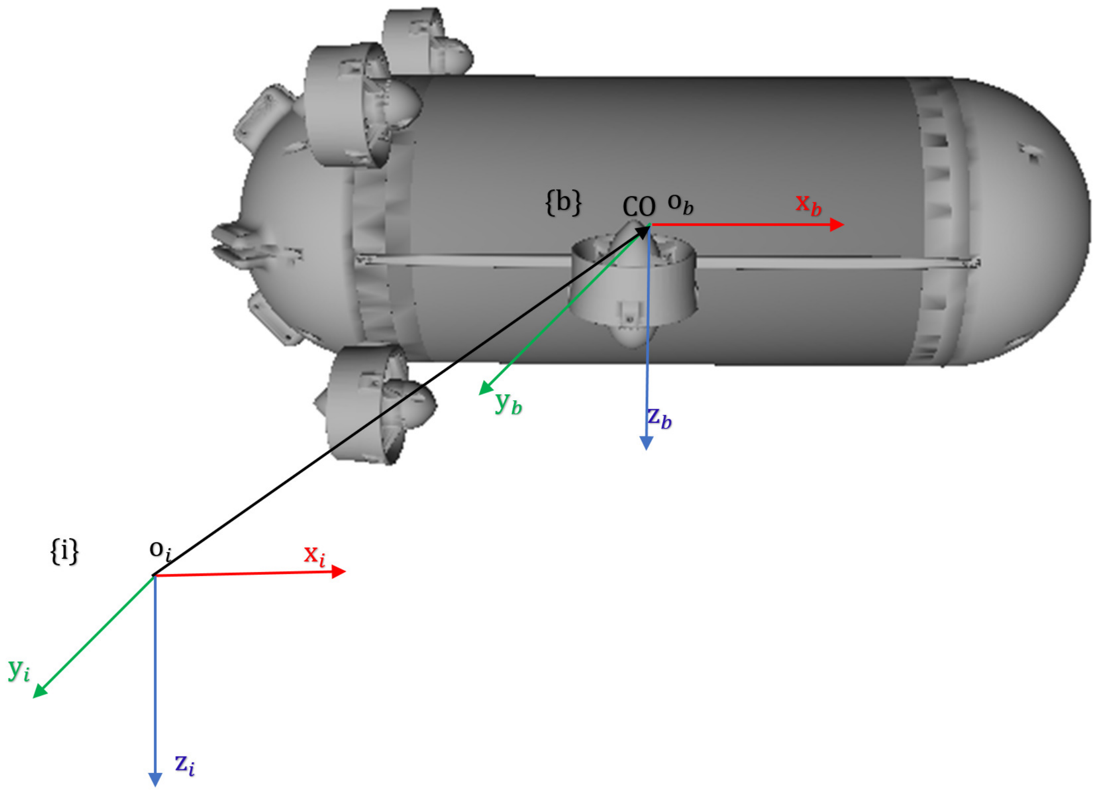


Figure A.1: Inertial world frame $\{i\}$ and body frame $\{b\}$

mechanics as boundary conditions, are fixed in space and therefore independent of time. The world coordinate frame is defined in such a way that the directions x_i and y_i are in the plane of the undisturbed water surface and z_i is in the positive direction pointing downwards into the fluid [VSHvH14].

The body-fixed reference frame $\{b\} = (x_b, y_b, z_b)$ is a moving coordinate frame fixed to the robot. The origin o_b is usually chosen to coincide with the geometric center of hull which will be referred to as CO. The longitudinal axis x_b is directed from aft to fore, the transversal axis y_b is directed to starboard and the normal axis z_b orientates from top to bottom.

The rotation matrix transforming quantities (linear velocities, forces and moments) in inertial frame $\{b\}$ to body frame $\{i\}$ are denoted as $\boldsymbol{R}$, and they are elements in *SO(3)* belonging to *special orthogonal group of order 3:*

$$SO(3) = \left\{\boldsymbol{R}|\boldsymbol{R} \in \mathbf{R}^{3\times3}, \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}, det(\boldsymbol{R}) = 1\right\}. \tag{A.1}$$

The vector cross-product $\times$ is defined by

$$\zeta \times \mathbf{a} := \boldsymbol{S}(\zeta)\mathbf{a}, \tag{A.2}$$

where $\boldsymbol{S}$ is a skew-symmetrical matrix satisfying $\boldsymbol{S} = \boldsymbol{S}^T$ and defined as

$$\boldsymbol{S}(\zeta) = -\boldsymbol{S}^T(\zeta) = \begin{pmatrix} 0 & -\zeta_3 & \zeta_2 \\ \zeta_3 & 0 & -\zeta_1 \\ -\zeta_2 & \zeta_1 & 0 \end{pmatrix}, \zeta = \begin{pmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{pmatrix}. \tag{A.3}$$

The rotation matrix $\boldsymbol{R}_{\zeta,\beta}$ corresponds to a rotation of angle β in the counter-clockwise direction about a rotation axis parallel to the unit vector $\zeta = [\zeta_1, \zeta_2, \zeta_3]^T, ||\zeta|| = 1$:

$$\boldsymbol{R}_{\zeta,\beta} = \boldsymbol{I}_{3\times3} + \sin(\beta)\boldsymbol{S}(\zeta) + [1 - \cos(\beta)]\boldsymbol{S}^2(\zeta), \tag{A.4}$$

where $\boldsymbol{I}_{3\times3}$ is the three-dimensional identity matrix. Because ζ is a unit vector, $\boldsymbol{S}^2(\zeta) = \zeta\zeta^T - \boldsymbol{I}_{3\times3}$.
While the position and orientation of underwater robots are expressed with respect to inertial frame $\{i\}$, the linear and angular velocities of underwater robots are conventionally expressed relative to body frame $\{b\}$. The position of underwater robots in the inertial world frame $\{i\}$ is denoted by the vector

$$\mathbf{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \in \mathbb{R}^3, \tag{A.5}$$

the orientation of underwater robots is described by Euler angles

$$\lambda = \begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix} \in S^3, \tag{A.6}$$

where $\mathbb{R}^3$ is the three dimensional Euclidean space and S^3 is a sphere. We can write them together into a six-dimensional generalized position vector $\eta \in \mathbf{R}^3 \times S^3$:

$$\eta = \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix} = \begin{pmatrix} \mathbf{p} \\ \lambda \end{pmatrix}. \tag{A.7}$$

The body-fixed linear velocity which is the linear velocity of o_b relative to world frame $\{i\}$ expressed in body frame $\{b\}$ is denoted by the vector

$$\mathbf{v}^b_{b/i} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} \in \mathbb{R}^3. \tag{A.8}$$

In other words, $\mathbf{v}^b_{b/i}$ denotes the changing of length of vector o_io_b observed in body frame $\{b\}$, where o_i and o_b are the origin of inertial world frame $\{i\}$ and body frame $\{b\}$, respectively.

The body-fixed angular velocity which is the angular velocity with respect to world axes $\{i\}$ expressed in $\{b\}$ is denoted by the vector

$$\omega_{b/i}^{b} = \begin{pmatrix} p \\ q \\ r \end{pmatrix} \in \mathbb{R}^3. \tag{A.9}$$

Define a new generalized velocity vector $\upsilon \in \mathbb{R}^6$:

$$\upsilon = \begin{pmatrix} \upsilon_1 \\ \upsilon_2 \end{pmatrix} = \begin{pmatrix} \mathbf{v}_{b/i}^{b} \\ \omega_{b/i}^{b} \end{pmatrix}. \tag{A.10}$$

The body-fixed forces and moments through o_b are expressed by

$$\mathbf{f}_b^b = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \in \mathbb{R}^3 \tag{A.11}$$

and

$$\mathbf{m}_b^b = \begin{pmatrix} K \\ M \\ N \end{pmatrix} \in \mathbb{R}^3, \tag{A.12}$$

respectively.
The generalized force vector $\tau \in \mathbb{R}^6$ is defined as

$$\tau = \begin{pmatrix} \mathbf{f}_b^b \\ \mathbf{m}_b^b \end{pmatrix}. \tag{A.13}$$

Principal Rotations are rotations about x, y and z axes by setting $\lambda = [1, 0, 0]^T$, $\lambda = [0, 1, 0]^T$, and $\lambda = [0, 0, 1]^T$ and $\beta = \phi, \beta = \theta$ and $\beta = \psi$, receptively. For simplicity, we use the following abbreviated notions for triangular functions:

$$\sin(\cdot) = s(\cdot), \tag{A.14}$$

$$\cos(\cdot) = c(\cdot), \tag{A.15}$$

$$\tan(\cdot) = t(\cdot). \tag{A.16}$$

Using formula A.4, we can get

$$\boldsymbol{R}_{x,\phi} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{pmatrix}, \tag{A.17}$$

$$\boldsymbol{R}_{y,\theta} = \begin{pmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{pmatrix}, \tag{A.18}$$

$$\boldsymbol{R}_{z,\psi} = \begin{pmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{A.19}$$

The Euler angles, roll(ϕ), pitch(θ) and yaw(ψ), is written as vector $\lambda = (\phi, \theta, \psi)^T$. For transformation from world frame $\{i\}$ to body frame $\{b\}$ it is customary to use the *zyx* convention based on the Euler angles ϕ, θ and ψ.

$$\begin{aligned} \boldsymbol{R}^{-1}(\lambda) &:= \boldsymbol{R}_{z,\psi}\boldsymbol{R}_{y,\theta}\boldsymbol{R}_{x,\phi} \\ &= \begin{pmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi c\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{pmatrix} \end{aligned} \tag{A.20}$$

and the inverse transformation from body frame $\{b\}$ to world frame $\{i\}$ is characterized by the rotation matrix:

$$\boldsymbol{R}(\lambda) = \boldsymbol{R}^{-T}(\lambda) = \boldsymbol{R}^T_{x,\phi}\boldsymbol{R}^T_{y,\theta}\boldsymbol{R}^T_{z,\psi} \tag{A.21}$$

The derivative of underwater robot position can be related to the body-fixed velocity vector $\mathbf{v}^b_{b/i}$

$$\dot{\mathbf{p}} = \boldsymbol{R}(\lambda)\mathbf{v}^b_{b/i}. \tag{A.22}$$

The inverse transformation is

$$\mathbf{v}^b_{b/i} = \boldsymbol{R}^{-1}(\lambda)\dot{\mathbf{p}}. \tag{A.23}$$

In terms of relationship between body-fixed angular velocity $\omega^b_{b/i}$ and the Euler rate $\dot{\lambda} = (\dot{\phi}, \dot{\theta}, \dot{\psi})^T$ another transformation matrix $\boldsymbol{Q}(\lambda)$ is used. By expanding the following relationship

$$\omega^b_{b/i} = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + \boldsymbol{R}^T_{x,\phi}\begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + \boldsymbol{R}^T_{x,\phi}\boldsymbol{R}^T_{y,\theta}\begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} := \boldsymbol{Q}^{-1}(\lambda)\dot{\lambda} \tag{A.24}$$

we can yield

$$= \boldsymbol{Q}^{-1}(\lambda)\begin{pmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & c\theta s\phi \\ 0 & -s\phi & c\theta c\phi \end{pmatrix}, \tag{A.25}$$

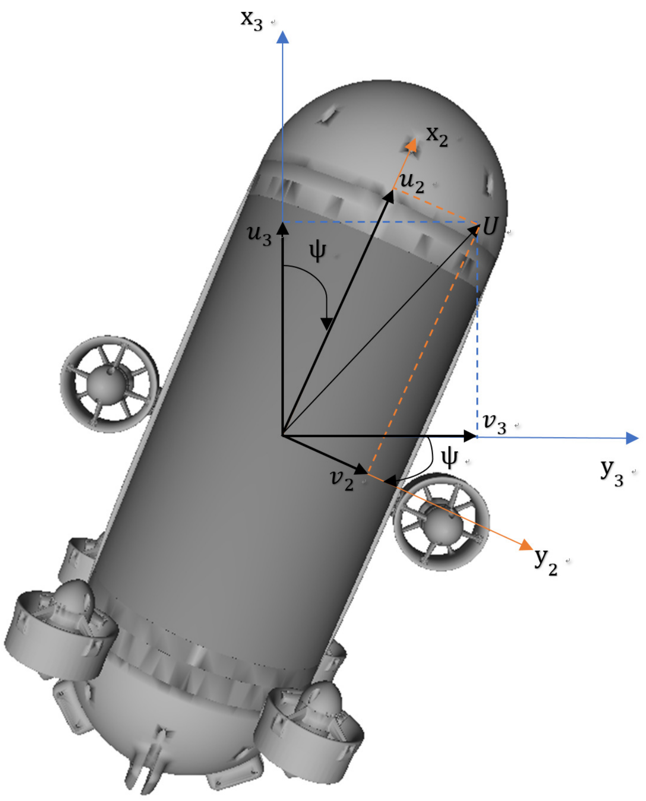


Figure A.2: Rotation over yaw angle ψ about z_3. Note that $w_3 = w_2$

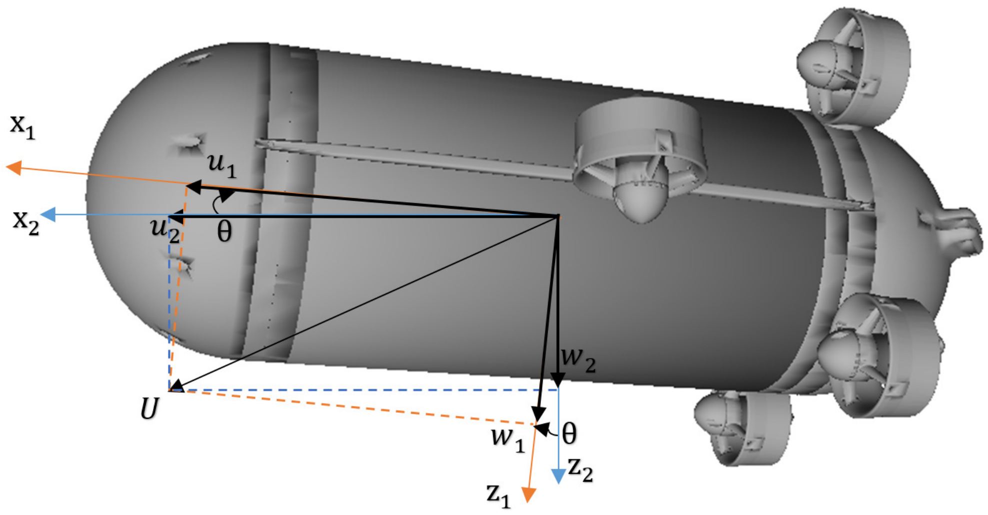


Figure A.3: Rotation over pitch angle θ about y_2. Note that $v_3 = v_2$

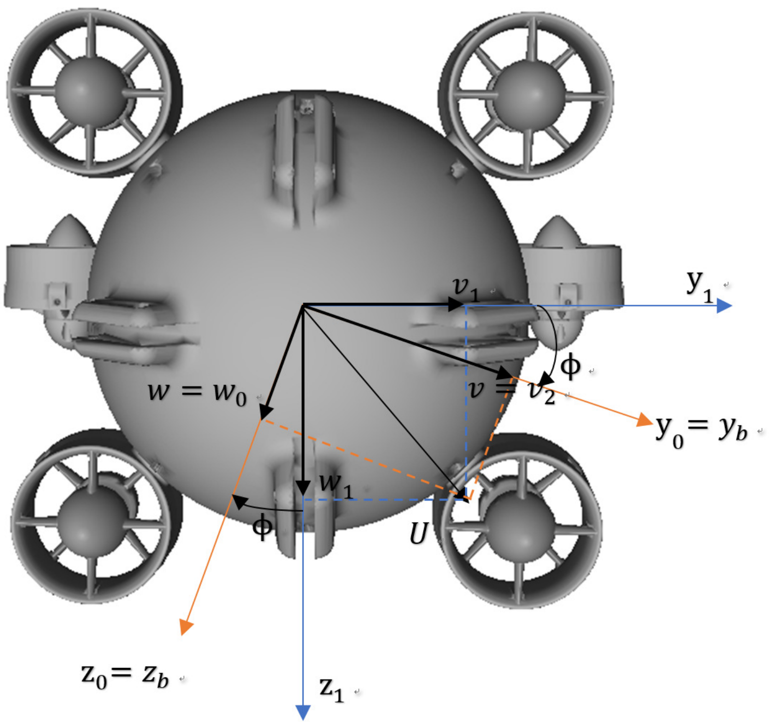


Figure A.4: Rotation over roll angle ϕ about x_1. Note that $u_1 = u$

and

$$\boldsymbol{Q}(\lambda) = \begin{pmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{pmatrix} \tag{A.26}$$

Notice that $\boldsymbol{Q}(\lambda)$ is undefined for pitch angle of $\theta \neq \pm 90°$. With help of A.26 we can obtain

$$\dot{\lambda} = \boldsymbol{Q}(\lambda)\omega^b_{b/i}. \tag{A.27}$$

The inertial world frame $\{i\}$ is a frame of reference with homogeneous and isotropic description of time and space in a time-dependent manner. Newton's second law and Euler's law are formulated in inertial frame $\{i\}$.
Note that vectors are always coordinate-free which means vectors exist without any defined frame and their length and direction stay invariant for all frames. When a vector is observed in a arbitrary frame, it should be decoupled as a linear combination of unit basis vectors of this frame. Thus, a vector is represented differently in terms of coordinates in different frames.
The unit vectors $\mathbf{i}_b$, $\mathbf{j}_b$ and $\mathbf{k}_b$ denote unit basis vectors of body-fixed frame $\{b\}$. The unit vectors $\mathbf{i}_i$, $\mathbf{j}_i$ and $\mathbf{k}_i$ denote unit basis vectors of the inertial world frame $\{i\}$. For rotation of a body frame relative to an inertial frame, we can derive the relationship of coordinates of an arbitrary vector in $\{i\}$ and $\{b\}$.

Firstly, we study this problem in a simplified two-dimensional case. Assume the body-fixed frame $\{b\}$ rotates about the z axis of the inertial frame $\{i\}$ with a constant angular velocity $\omega_{z,\theta}$, $\theta = \omega_{z,\theta} t$ and these two frames coincide at time $t = 0$. Conversion between coordinates (x_i, y_i) and (x_b, y_b) can be performed as follows:

$$x_i = x_b \cos\theta - y_b \sin\theta, \tag{A.28}$$

$$y_i = x_b \sin\theta + y_b \cos\theta, \tag{A.29}$$

$$x_b = x_i \cos(-\theta) - y_i \sin(-\theta), \tag{A.30}$$

$$y_b = x_i \sin(-\theta) + y_i \cos(-\theta). \tag{A.31}$$

The unit vectors $\mathbf{i}_b$ and $\mathbf{j}_b$ can be represented by coordinates in body frame $\{b\}$:

$$\mathbf{i}_b^b = (1, 0), \tag{A.32}$$

$$\mathbf{j}_b^b = (0, 1). \tag{A.33}$$

If we represent them with coordinates in inertial frame $\{i\}$, we can obtain

$$\mathbf{i}_b^i = (\cos\theta, \sin\theta), \tag{A.34}$$

and

$$\mathbf{j}_b^i = (-\sin\theta, \cos\theta). \tag{A.35}$$

Take derivative of $\mathbf{i}_b$ and $\mathbf{j}_b$ in the inertial frame $\{i\}$:

$$\frac{{}^i d}{dt}\mathbf{i}_b = \frac{d}{dt}(\cos\theta \mathbf{i}_i + \sin\theta \mathbf{j}_i) = \frac{d\theta}{dt}(-\sin\theta \mathbf{i}_i + \cos\theta \mathbf{j}_i) = \omega_{z,\theta}(-\sin\theta, \cos\theta) = \omega_{z,\theta}\mathbf{j}_b, \tag{A.36}$$

$$\frac{{}^i d}{dt}\mathbf{j}_b = \frac{d}{dt}(\cos\theta \mathbf{i}_i + \sin\theta \mathbf{j}_i) = \frac{d\theta}{dt}(-\sin\theta \mathbf{i}_i + \cos\theta \mathbf{j}_i) = \omega_{z,\theta}(-\sin\theta, \cos\theta) = -\omega_{z,\theta}\mathbf{i}_b. \tag{A.37}$$

Define the rotation vector $\omega_{b/i} = (0, 0, \omega_{z,\theta})$,

$$\frac{{}^i d}{dt}\mathbf{u} = \omega_{b/i} \times \mathbf{u}, \tag{A.38}$$

where $\mathbf{u}$ is either $\mathbf{i}_b$ or $\mathbf{j}_b$. This relationship can be extended to the three-dimensional rotation in space, that is, $\mathbf{u}$ can be $\mathbf{i}_b$, $\mathbf{j}_b$ or $\mathbf{k}_b$ and $\omega_{b/i}$ is the angular velocity vector for rotation from $\{i\}$ to $\{b\}$ in space.

Assume an arbitrary vector **a**, if we observe it in body frame $\{b\}$, we can write it as linear combination of unit vectors of $\{b\}$,

$$\mathbf{a}(t) = a_x(t)\mathbf{i}_b + a_y(t)\mathbf{j}_b + a_z(t)\mathbf{k}_b. \tag{A.39}$$

Differentiate of **a** with respect to time in the world frame $\{i\}$:

$$\begin{aligned}\frac{{}^i d}{dt}\mathbf{a} &= \frac{da_x}{dt}\mathbf{i}_b + \frac{{}^i d\mathbf{i}_b}{dt}a_x + \frac{da_y}{dt}\mathbf{j}_b + \frac{{}^i d\mathbf{j}_b}{dt}a_y + \frac{da_z}{dt}\mathbf{k}_b + \frac{{}^i d\mathbf{k}_b}{dt}a_z \\ &= \frac{da_x}{dt}\mathbf{i}_b + \frac{da_y}{dt}\mathbf{j}_b + \frac{da_z}{dt}\mathbf{k}_b + [\omega_{ib} \times (a_x\mathbf{i}_b + a_y\mathbf{j}_b + a_z\mathbf{k}_b)] \\ &= \frac{{}^b d}{dt}\mathbf{a} + \omega_{b/i} \times \mathbf{a}.\end{aligned} \tag{A.40}$$

The following formula can be obtained:

$$\frac{{}^i d}{dt}\mathbf{a} = \frac{{}^b d}{dt}\mathbf{a} + \omega_{b/i} \times \mathbf{a}, \tag{A.41}$$

and this formula will be repeatedly used in following sections for derivation of the underwater robot dynamics.
Assume point A is an arbitrary point in the rigid robot body. From Figure A.5 we obtain

$$\mathbf{r}_{A/i} = \mathbf{r}_{b/i} + \mathbf{r}_A, \tag{A.42}$$

where $\mathbf{r}_A$ is the distance from CO to A which denotes the location of the point A in body frame $\{b\}$. Time differentiation of $\mathbf{r}_{A/i}$ in the inertial world frame $\{i\}$ using A.41 gives

$$\begin{aligned}\mathbf{v}_{A/i} = \frac{{}^i d}{dt}(\mathbf{r}_{A/i}) &= \frac{{}^i d}{dt}(\mathbf{r}_{b/i} + \mathbf{r}_A) \\ &= \frac{{}^i d}{dt}\mathbf{r}_{b/i} + \frac{{}^i d}{dt}\mathbf{r}_A \\ &= \mathbf{v}_{b/i} + (\frac{{}^b d}{dt}\mathbf{r}_A + \omega_{b/i} \times \mathbf{r}_A).\end{aligned} \tag{A.43}$$

A fixed point A in the rigid robot body satisfies

$$\frac{{}^b d}{dt}\mathbf{r}_A = \mathbf{0}. \tag{A.44}$$

Thus, the equation A.43 can be written as

$$\mathbf{v}_{A/i} = \mathbf{v}_{b/i} + \omega_{b/i} \times \mathbf{r}_A. \tag{A.45}$$

This relationship is of great significance for further work, since it can be used to calculate the velocity at an arbitrary point A in the robot body from the translational velocities vector $\mathbf{v}_{b/i}$, the angular velocities vector $\omega_{b/i}$ and the position vector r_A denoting the position of point A in body frame $\{b\}$.

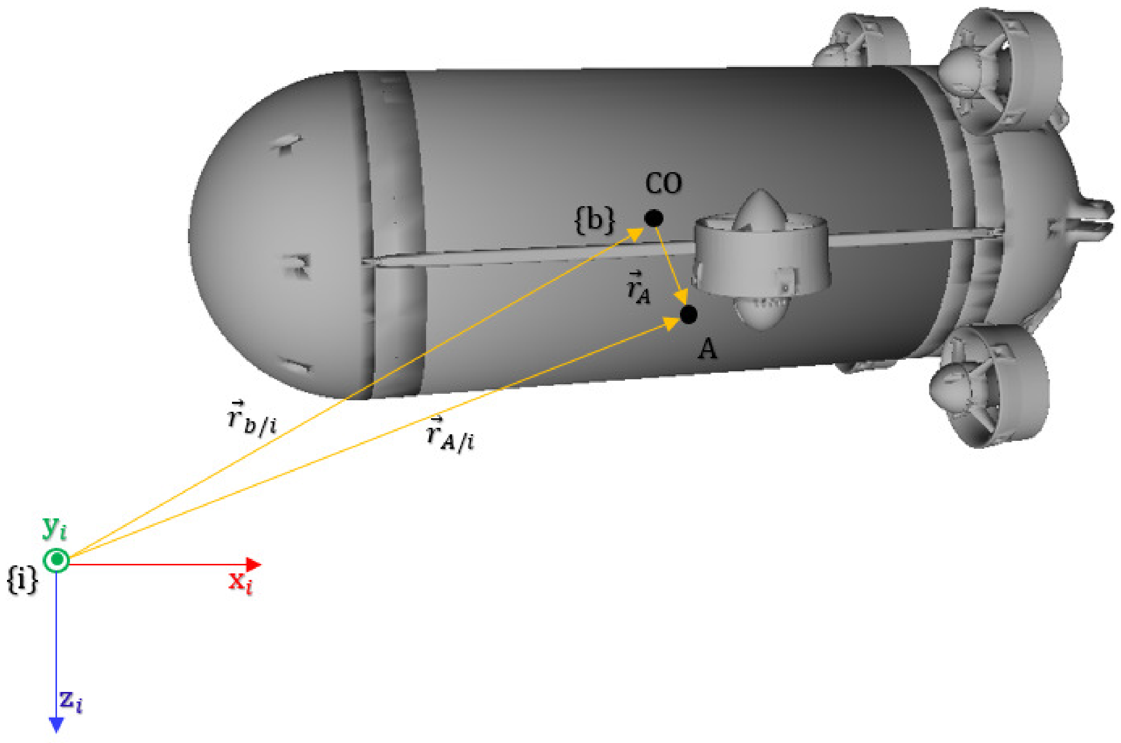


Figure A.5: Definition of vectors $\mathbf{r}_{A/i}$, $\mathbf{r}_{b/i}$ and $\mathbf{r}_A$

A.1.2 Newton-Euler Equations of Motion about CG

Newton's second law describes the relationship between the force exerted on the body $\mathbf{f}_g$ with the generated linear acceleration $\dot{\mathbf{v}}_{g/i}$. The resultant force acts on the center of gravity CG. Having implemented Newton's second law for a robot in body frame $\{b\}$, the following relationship is derived:

$$\frac{{}^i d}{dt}(m\mathbf{v}_{g/i}) = \mathbf{f}_g. \tag{A.46}$$

where $\mathbf{v}_{g/i}$ is the velocity of the CG with respect to the inertial frame $\{i\}$. Using A.41

$$\begin{aligned} \mathbf{f}_g &= \frac{{}^i d}{dt}(m\mathbf{v}_{g/i}) \\ &= \frac{{}^b d}{dt}(m\mathbf{v}_{g/i}) + \omega_{b/i} \times \mathbf{v}_{g/i} \\ &= m(\dot{\mathbf{v}}_{g/i} + \omega_{b/i} \times \mathbf{v}_{g/i}). \end{aligned} \tag{A.47}$$

Expressing all vectors in $\{b\}$

$$\mathbf{f}_g^b = m(\dot{\mathbf{v}}_{g/i}^b + \boldsymbol{S}(\omega_{b/i}^b)\mathbf{v}_{g/i}^b), \tag{A.48}$$

where

$$\boldsymbol{S}(\omega_{b/i}^b\mathbf{v}_{g/i}^b) = \omega_{b/i} \times \mathbf{v}_{g/i}. \tag{A.49}$$

We derive the rotational dynamics in a similar way. From Euler's second law,

$$\begin{aligned} \mathbf{m}_g &= \frac{{}^i d}{dt}\mathbf{h}_g \\ &= \frac{{}^i d}{dt}(\boldsymbol{I}_g\omega_{b/i}) \\ &= \frac{{}^b d}{dt}(\boldsymbol{I}_g\omega_{b/i}) + \omega_{b/i} \times (\boldsymbol{I}_g\omega_{b/i}) \\ &= \boldsymbol{I}_g\dot{\omega}_{b/i} - (\boldsymbol{I}_g\omega_{b/i}) \times \omega_{b/i}. \end{aligned} \tag{A.50}$$

Representing all vectors in body frame $\{b\}$

$$\boldsymbol{I}_g\dot{\omega}_{b/n}^b - \boldsymbol{S}(\boldsymbol{I}_g\omega_{b/i}^b)\omega_{b/i}^b = \mathbf{m}_g^b, \tag{A.51}$$

where

$$\boldsymbol{S}(\boldsymbol{I}_g\omega_{b/i}^b)\omega_{b/i}^b = (\boldsymbol{I}_g\omega_{b/i}^b) \times \omega_{b/i}^b, \tag{A.52}$$

and

$$\boldsymbol{I}_g := \begin{pmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{yx} & I_y & -I_{yz} \\ -I_{zx} & -I_{zy} & I_z \end{pmatrix}. \tag{A.53}$$

I_x, I_y and I_z are moments of inertia about the x_b, y_b and z_b axes respectively, and $I_{xy} = I_{yx}$, $I_{xz} = I_{zx}$, $I_{yz} = I_{zy}$ are the products of inertia defined as

$$I_x = \int_V (y^2 + z^2)\rho_m dV, \tag{A.54}$$

$$I_y = \int_V (x^2 + z^2)\rho_m dV, \tag{A.55}$$

$$I_z = \int_V (x^2 + y^2)\rho_m dV, \tag{A.56}$$

$$I_{xy} = \int_V xy\rho_m dV = \int_V yx\rho_m dV = I_{yx}, \tag{A.57}$$

$$I_{xz} = \int_V xz\rho_m dV = \int_V zx\rho_m dV = I_{zx}, \tag{A.58}$$

$$I_{yz} = \int_V yz\rho_m dV = \int_V zy\rho_m dV = I_{zy}. \tag{A.59}$$

Note that

$$\omega^b_{g/i} = \omega^b_{b/i}. \tag{A.60}$$

Combining A.48 and A.51, we can obtain the equations of motion for underwater robots about CG

$$\begin{pmatrix} \mathbf{f}^b_g \\ \mathbf{m}^b_g \end{pmatrix} = \boldsymbol{M}^{CG}_{RB} \begin{pmatrix} \dot{\mathbf{v}}^b_{g/i} \\ \dot{\omega}^b_{g/i} \end{pmatrix} + \boldsymbol{C}^{CG}_{RB}(\mathbf{v}) \begin{pmatrix} \mathbf{v}^b_{g/i} \\ \omega^b_{g/i} \end{pmatrix}, \tag{A.61}$$

where

$$\boldsymbol{M}^{CG}_{RB} = \begin{pmatrix} m\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_g \end{pmatrix} \tag{A.62}$$

and

$$\boldsymbol{C}^{CG}_{RB} = \begin{pmatrix} m\boldsymbol{S}(\omega^b_{b/n}) & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & -\boldsymbol{S}(\boldsymbol{I}_g\omega^b_{b/n}) \end{pmatrix}. \tag{A.63}$$

A.1.3 Newton-Euler Equations of Motion about CO

For underwater robots the center of gravity CG is varying with different moving directions due to different added masses. Since the hydrodynamic forces and moments are always calculated in CO as well, it is desirable to formulate Newton's law and Euler's law in CO which is the origin of the body frame $\{b\}$ to take the advantage of the robot's geometric properties.
The relationship between the velocity at CG and the velocity at CO in $\{b\}$ is expressed by

$$\begin{aligned} \mathbf{v}^b_{g/i} &= \mathbf{v}^b_{b/i} + \omega^b_{b/i} \times \mathbf{r}^b_g \\ &= \mathbf{v}^b_{b/i} - \mathbf{r}^b_g \times \omega^b_{b/i} = \mathbf{v}^b_{b/i} + \boldsymbol{S}^T(\mathbf{r}^b_g)\omega^b_{b/i}. \end{aligned} \tag{A.64}$$

We define

$$\boldsymbol{H}(\mathbf{r}^b_g) := \begin{pmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{S}^T(\mathbf{r}^b_g) \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{pmatrix} \tag{A.65}$$

and

$$\boldsymbol{H}^T(\mathbf{r}^b_g) := \begin{pmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{S}(\mathbf{r}^b_g) & \boldsymbol{I}_{3\times3} \end{pmatrix}. \tag{A.66}$$

Writing A.64 in vectorial form

$$\begin{pmatrix} \mathbf{v}^b_{g/i} \\ \omega^b_{b/i} \end{pmatrix} = \boldsymbol{H}(\mathbf{r}^b_g) \begin{pmatrix} \mathbf{v}^b_{b/i} \\ \omega^b_{b/i} \end{pmatrix}, \tag{A.67}$$

where $\mathbf{r}^b_g = (x_g, y_g, z_g)^T$ denotes the position of CG in body frame $\{b\}$ and $\boldsymbol{H}(\mathbf{r}^b_g) \in \mathbb{R}^{3\times3}$ is the transformation matrix from velocities at CG and velocities at o_b. It is noticeable that for a rigid body $\omega^b_{g/i} = \omega^b_{b/i}$. Therefore, A.67 can be rewritten as

$$\begin{pmatrix} \mathbf{v}^b_{g/i} \\ \omega^b_{g/i} \end{pmatrix} = \boldsymbol{H}(\mathbf{r}^b_g) \begin{pmatrix} \mathbf{v}^b_{b/i} \\ \omega^b_{b/i} \end{pmatrix}. \tag{A.68}$$

We can also get the transformation relationship for accelerations:

$$\begin{pmatrix} \dot{\mathbf{v}}^b_{g/i} \\ \dot{\omega}^b_{g/i} \end{pmatrix} = \boldsymbol{H}(\mathbf{r}^b_g) \begin{pmatrix} \dot{\mathbf{v}}^b_{b/i} \\ \dot{\omega}^b_{b/i} \end{pmatrix}. \tag{A.69}$$

Replacing the acceleration vector and velocity vector in A.61 with A.68 and A.69 and multiplying both sides with matrix $\boldsymbol{H}^T(\mathbf{r}^b_g)$, we obtain

$$\boldsymbol{H}^T(\mathbf{r}^b_g)\boldsymbol{M}^{CG}_{RB}\boldsymbol{H}(\mathbf{r}^b_g) \begin{pmatrix} \dot{\mathbf{v}}^b_{b/i} \\ \dot{\omega}^b_{b/i} \end{pmatrix} + \boldsymbol{H}^T(\mathbf{r}^b_g)\boldsymbol{C}^{CG}_{RB}\boldsymbol{H}(\mathbf{r}^b_g) \begin{pmatrix} \mathbf{v}^b_{b/i} \\ \omega^b_{b/i} \end{pmatrix} = \boldsymbol{H}^T(\mathbf{r}^b_g) \begin{pmatrix} \mathbf{f}^b_g \\ \mathbf{m}^b_g \end{pmatrix}. \tag{A.70}$$

We define

$$\begin{aligned}
\boldsymbol{M}_{RB}^{CO} &= \boldsymbol{H}^T(\mathbf{r}_g^b)\boldsymbol{M}_{RB}^{CG}\boldsymbol{H}(\mathbf{r}_g^b) \\
&= \begin{pmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{S}(\mathbf{r}_g^b) & \boldsymbol{I}_{3\times3} \end{pmatrix} \begin{pmatrix} m\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_g \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{S}^T(\mathbf{r}_g^b) \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{pmatrix} \\
&= \begin{pmatrix} m\boldsymbol{I}_{3\times3} & -m\boldsymbol{S}(\mathbf{r}_g^b) \\ m\boldsymbol{S}(\mathbf{r}_g^b) & \boldsymbol{I}_g - m\boldsymbol{S}^2(\mathbf{r}_g^b) \end{pmatrix}
\end{aligned} \tag{A.71}$$

and

$$\begin{aligned}
\boldsymbol{C}_{RB}^{CO} &= \boldsymbol{H}^T(\mathbf{r}_g^b)\boldsymbol{C}_{RB}^{CG}\boldsymbol{H}(\mathbf{r}_g^b) \\
&= \begin{pmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{S}(\mathbf{r}_g^b) & \boldsymbol{I}_{3\times3} \end{pmatrix} \begin{pmatrix} m\boldsymbol{S}(\omega_{b/n}^b) & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & -\boldsymbol{S}(\boldsymbol{I}_g\omega_{b/n}^b) \end{pmatrix} \begin{pmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{S}^T(\mathbf{r}_g^b) \\ \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{pmatrix} \\
&= \begin{pmatrix} m\boldsymbol{S}(\omega_{b/i}^b) & -m\boldsymbol{S}(\omega_{b/i}^b)\boldsymbol{S}(\mathbf{r}_g^b) \\ m\boldsymbol{S}(\mathbf{r}_g^b)\boldsymbol{S}(\omega_{b/i}^b) & m\boldsymbol{S}(\mathbf{r}_g^b)\boldsymbol{S}(\omega_{b/i}^b)\boldsymbol{S}^T(\mathbf{r}_g^b)\omega_{b/i}^b - \boldsymbol{S}(\boldsymbol{I}_g\omega_{b/i}^b)\omega_{b/i}^b \end{pmatrix} \\
&= \begin{pmatrix} m\boldsymbol{S}(\omega_{b/i}^b) & -m\boldsymbol{S}(\omega_{b/i}^b)\boldsymbol{S}(\mathbf{r}_g^b) \\ m\boldsymbol{S}(\mathbf{r}_g^b)\boldsymbol{S}(\omega_{b/i}^b) & -\boldsymbol{S}((\boldsymbol{I}_g - m\boldsymbol{S}^2(\mathbf{r}_g^b))\omega_{b/i}^b) \end{pmatrix}.
\end{aligned} \tag{A.72}$$

From the right side of A.70 we obtain:

$$\boldsymbol{H}^T(\mathbf{r}_g^b)\begin{pmatrix} \mathbf{f}_g^b \\ \mathbf{m}_g^b \end{pmatrix} = \begin{pmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{S}(\mathbf{r}_g^b) & \boldsymbol{I}_{3\times3} \end{pmatrix}\begin{pmatrix} \mathbf{f}_g^b \\ \mathbf{m}_g^b \end{pmatrix} = \begin{pmatrix} \mathbf{f}_g^b \\ \mathbf{m}_g^b + \mathbf{r}_g^b \times \mathbf{f}_g^b \end{pmatrix}. \tag{A.73}$$

For a rigid body, it is obviously correct:

$$\mathbf{f}_b^b = \mathbf{f}_g^b, \tag{A.74}$$

where $\mathbf{f}_b^b$ is the force exerted through the origin CO of the body frame $\{b\}$ observed in $\{b\}$. The moments about CO consist of two parts:

$$\mathbf{m}_b^b = \mathbf{m}_g^b + \mathbf{r}_g^b \times \mathbf{f}_g^b. \tag{A.75}$$

The first term $\mathbf{m}_g^b$ is the moment about CG and the second term results from the force exerting on CG, we just shift it to CO. The second part of moments about CO is the moment induced by the force exerting on the center of gravity CG. Taking A.74 and A.75 into A.73, we obtain:

$$\begin{pmatrix} \mathbf{f}_b^b \\ \mathbf{m}_b^b \end{pmatrix} = \boldsymbol{H}^T(\mathbf{r}_g^b)\begin{pmatrix} \mathbf{f}_g^b \\ \mathbf{m}_g^b \end{pmatrix}. \tag{A.76}$$

The first row of A.70 describes the translational motion about CG:

$$m[\dot{\mathbf{v}}_{b/i}^b + \boldsymbol{S}(\dot{\omega}_{b/i}^b)\mathbf{r}_g^b + \boldsymbol{S}(\omega_{b/i}^b)\mathbf{v}_{b/i}^b + \boldsymbol{S}^2(\omega_{b/i}^b)\mathbf{r}_g^b] = \mathbf{f}_b^b. \tag{A.77}$$

Using the vector cross-product:

$$m[\dot{\mathbf{v}}^b_{b/i} + \dot{\omega}^b_{b/i} \times \mathbf{r}^b_g + \omega^b_{b/i} \times \mathbf{v}^b_{b/i} + \omega^b_{b/i} \times (\omega^b_{b/i} \times \mathbf{r}^b_g)] = \mathbf{f}^b_b. \tag{A.78}$$

The inertia matrix $\boldsymbol{I}_b = \boldsymbol{I}_b^T \in \mathbb{R}^{3\times3}$ about an arbitrary origin o_b can be transformed from the inertia matrix $\boldsymbol{I}_g = \boldsymbol{I}_g^T \in \mathbb{R}^{3\times3}$about the body's center of gravity CG according to

$$\boldsymbol{I}_b = \boldsymbol{I}_g - m\boldsymbol{S}^2(\mathbf{r}^b_g) = \boldsymbol{I}_g - m(\mathbf{r}^b_g(\mathbf{r}^b_g)^T - (\mathbf{r}^b_g)^T\mathbf{r}^b_g\boldsymbol{I}_{3\times3}). \tag{A.79}$$

The second row of A.70 can be written as

$$\boldsymbol{I}_b\dot{\omega}^b_{b/i} + \boldsymbol{S}(\omega^b_{b/i})\boldsymbol{I}_b\omega^b_{b/i} + m\boldsymbol{S}(\mathbf{r}^b_g)\dot{\mathbf{v}}^b_{b/i} + m\boldsymbol{S}(\mathbf{r}^b_g)\boldsymbol{S}(\omega^b_{b/i})\mathbf{v}^b_{b/i} = \mathbf{m}^b_b. \tag{A.80}$$

Alternatively,

$$\boldsymbol{I}_b\dot{\omega}^b_{b/i} + \omega^b_{b/i} \times \boldsymbol{I}_b\omega^b_{b/i} + m\mathbf{r}^b_g \times (\dot{\mathbf{v}}^b_{b/i} + \omega^b_{b/i} \times \mathbf{v}^b_{b/i}) = \mathbf{m}^b_b. \tag{A.81}$$

A.1.4 Kinematic Equation of Underwater Robot

To sum up, the rigid-body kinetics can be expressed in vectorial way:

$$\begin{pmatrix} \mathbf{f}^b_b \\ \\ \mathbf{m}^b_b \end{pmatrix} = \mathbf{M}^{CO}_{RB} \begin{pmatrix} \dot{\mathbf{v}}^b_{b/i} \\ \\ \dot{\omega}^b_{b/i} \end{pmatrix} + \mathbf{C}^{CO}_{RB}(\upsilon) \begin{pmatrix} \mathbf{v}^b_{b/i} \\ \\ \mathbf{w}^b_{b/i} \end{pmatrix}. \tag{A.82}$$

Define $\upsilon = (u, v, w, p, q, r)^T$ representing the generalised velocity vector in $\{b\}$ and $\tau = (X, Y, Z, K, M, N)^T$ Simplifying the notation, we obtain

$$\boldsymbol{M}_{RB}\dot{\upsilon} + \boldsymbol{C}_{RB}(\upsilon)\upsilon = \tau_{RB}, \tag{A.83}$$

where $\boldsymbol{M}^{CO}_{RB} \equiv \boldsymbol{M}_{RB}$ and $\boldsymbol{C}^{CO}_{RB} \equiv \boldsymbol{C}_{RB}$. The rigid-body inertia matrix $\boldsymbol{M}_{RB}$ is symmetric and time-invariant satisfying

$$\boldsymbol{M}_{RB} = \boldsymbol{M}^T_{RB} \tag{A.84}$$

$$\dot{\boldsymbol{M}}_{RB} = \boldsymbol{0}_{6\times6} \tag{A.85}$$

A.1.5 Notation Simplifications

In the previous sections, we use superscripts and subscripts to indicate the reference frames or reference point of different quantities. In the following and in the thesis main body, we simplify the notations as follows:

$$\mathbf{v}^i_{b/i} \equiv \mathbf{v} \tag{A.86}$$

$$\omega^i_{b/i} \equiv \omega \tag{A.87}$$

A.2 Hydrodynamic Effects

In this section, the major hydrodynamic effects including on the rigid robot body will be discussed briefly. The theory of hydrodynamics is rather complex and it is difficult to develop a reliable model for most of the hydrodynamics effects [Ant14]. Thus, in this work, we model the hydrodynamic effects in the context of automatic control.

A.2.1 Added Mass and Inertia

When the robot is moving in the fluid, the surrounding fluid will be accelerated. A force is needed to achieve this acceleration and the surrounding fluid will exert a reaction force which posses same magnitude in the opposite direction. This reaction force is the the added mass contribution. The hydrodynamics force along x_b due to the linear acceleration in the x_b-direction is defined as:

$$X_A = -X_{\dot{u}}\dot{u} \tag{A.88}$$

and the added mass coefficient of surge acceleration $X_{\dot{u}}$ is defined as

$$X_{\dot{u}} = \left|\frac{\partial X_A}{\partial \dot{u}}\right|. \tag{A.89}$$

All the remaining 35 elements of $\boldsymbol{M}_A$ that relate the forces and moments components $(X, Y, Z, K, M, N)^T$ to the linear and angular accelerations $(\dot{u}, \dot{v}, \dot{w}, \dot{p}, \dot{q}, \dot{r})^T$ are defined in the same way. For underwater robots, we only consider the diagonal terms, the rest five diagonal terms are calculated as follows: The hydrodynamics force along y_b due to the linear acceleration in the y_b-direction is defined as:

$$Y_A = -Y_{\dot{v}}\dot{v}, \tag{A.90}$$

where the added mass coefficient of sway acceleration $Y_{\dot{v}}$ is defined as

$$Y_{\dot{v}} = \left|\frac{\partial Y_A}{\partial \dot{v}}\right|. \tag{A.91}$$

The hydrodynamics force along z_b due to the linear acceleration in the z_b-direction is defined as:

$$Z_A = -Z_{\dot{w}}\dot{w} \tag{A.92}$$

and the added mass coefficient of heave acceleration $Z_{\dot{w}}$ is defined as

$$Z_{\dot{w}} = \left|\frac{\partial Z_A}{\partial \dot{w}}\right|. \tag{A.93}$$

The hydrodynamics moment about x_b-axis due to the angular acceleration about the x_b-axis is defined as:

$$K_A = -K_{\dot{p}}\dot{p} \tag{A.94}$$

and the added mass coefficient of roll acceleration $K_{\dot{p}}$ is defined as

$$K_{\dot{p}} = \left| \frac{\partial K_A}{\partial \dot{p}} \right| . \tag{A.95}$$

The hydrodynamics moment about y_b-axis due to the angular acceleration about the y_b-axis is defined as:

$$M_A = -M_{\dot{p}} \dot{q} \tag{A.96}$$

and the added mass coefficient of pitch acceleration $M_{\dot{q}}$ is defined as

$$M_{\dot{q}} = \left| \frac{\partial M_A}{\partial \dot{q}} \right| . \tag{A.97}$$

The hydrodynamics moment about z_b-axis due to the angular acceleration about the z_b-axis is defined as:

$$N_A = -N_{\dot{r}} \dot{r} \tag{A.98}$$

and the added mass coefficient of yaw acceleration $N_{\dot{r}}$ is defined as

$$N_{\dot{r}} = \left| \frac{\partial N_A}{\partial \dot{r}} \right| . \tag{A.99}$$

For completely submerged bodies, we ignore all the non-diagonal terms and write $\boldsymbol{M}_A$:

$$\boldsymbol{M}_A = \begin{pmatrix} X_{\dot{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{\dot{v}} & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{\dot{w}} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{\dot{p}} & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{\dot{q}} & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{\dot{r}} \end{pmatrix}, \tag{A.100}$$

where $\boldsymbol{M}_A = \boldsymbol{M}_A^T \geq 0$. The added mass also has an added Coriolis and centripetal contribution. Since the underwater robot is completely submerged in the water and it has three planes of symmetry, the following structure of matrices of $\boldsymbol{C}_A(\upsilon)$ can be considered:

$$\boldsymbol{C}_A(\upsilon) = \begin{pmatrix} 0 & 0 & 0 & 0 & -Z_{\dot{w}} w & Y_{\dot{v}} v \\ 0 & 0 & 0 & Z_{\dot{w}} w & 0 & -X_{\dot{u}} u \\ 0 & 0 & 0 & -Y_{\dot{v}} v & X_{\dot{u}} u & 0 \\ 0 & -Z_{\dot{w}} w & Y_{\dot{v}} v & 0 & -N_{\dot{r}} \dot{r} & M_{\dot{q}} q \\ Z_{\dot{w}} w & 0 & -X_{\dot{u}} u & N_{\dot{r}} r & 0 & -K_{\dot{p}} p \\ -Y_{\dot{v}} v & X_{\dot{u}} u & 0 & -M_{\dot{q}} q & K_{\dot{p}} p & 0 \end{pmatrix}. \tag{A.101}$$

A.2.2 Damping Effects

The viscosity of the surrounding water causes dissipative drag and lift forces on the robot body. A common simplification for the underwater robots is to consider only linear and quadratic damping term and write them in a matrix $\boldsymbol{D}$ such that $\boldsymbol{D} \succ 0$. Assume, the coefficients of this matrix are constant. For a completely submerged robot body, the coupling dissipative terms are neglected, then we obtain

$$\begin{aligned}\boldsymbol{D} &= diag([X_u, Y_v, Z_w, K_p, M_q, N_r]) \\ &= diag([-X_{u|u|}|u|, -Y_{v|v|}|v|, -Z_{w|w|}|w|, -K_{p|p|}|p|, -M_{q|q|}|q|, -N_{r|r|}|r|]),\end{aligned} \tag{A.102}$$

where $X_{u|u|}$, $Y_{v|v|}$, $Z_{w|w|}$, $K_{p|p|}$, $M_{q|q|}$, $N_{r|r|}$ are linear quadratic damping coefficients due to surge, sway, heave, roll, pitch, yaw motion, respectively.

A.3 Restoring Force for Underwater Robots

Using m_{total} and V to denote the mass and the volume of the underwater robot, respectively. The submerged weight of the robot body and the buoyancy force are written as

$$W = m_{total}g, \tag{A.103}$$

$$B = \rho g V, \tag{A.104}$$

where g is the gravitational acceleration and ρ is the fluid velocity. Expressing them in the inertial world frame $\{i\}$

$$\mathbf{f}_W^i = \begin{pmatrix} 0 \\ 0 \\ W \end{pmatrix} \tag{A.105}$$

$$\mathbf{f}_B^i = -\begin{pmatrix} 0 \\ 0 \\ B \end{pmatrix} \tag{A.106}$$

Converting these two forces into the body frame $\{b\}$, we obtain

$$\mathbf{f}_W^b = \boldsymbol{R}(\lambda)\mathbf{f}_W^i, \tag{A.107}$$

$$\mathbf{f}_B^b = \boldsymbol{R}(\lambda)\mathbf{f}_B^i, \tag{A.108}$$

The moments generated by the gravity and the buoyancy are calculated as

$$\mathbf{m}_W^b = \mathbf{r}_g^b \times \mathbf{f}_W^b, \tag{A.109}$$

$$\mathbf{m}_B^b = \mathbf{r}_b^b \times \mathbf{f}_B^b, \tag{A.110}$$

where $\mathbf{r}_g^b = (x_g, y_g, z_g)$ is the position vector of the center of mass CG in the body frame $\{b\}$ and $\mathbf{r}_b^b = (x_b)$ denotes the center of buoyancy CO of the robot in the body frame $\{b$. Consequently, the restoring force and moment vector can be expressed in the body frame $\{b\}$ as

$$\begin{aligned}
\mathbf{g}(\eta) &= -\begin{pmatrix} \mathbf{f}_G^b + \mathbf{f}_B^b \\ \mathbf{r}_g^b \times \mathbf{f}_G^b + \mathbf{r}_b^b \times \mathbf{f}_B^b \end{pmatrix} \\
&= -\begin{pmatrix} \boldsymbol{R}(\lambda)\mathbf{f}_G^b + \mathbf{f}_B^b \\ \mathbf{r}_g^b \times \boldsymbol{R}(\lambda)\mathbf{f}_G^b + \mathbf{r}_b^b \times \boldsymbol{R}(\lambda)\mathbf{f}_B^b \end{pmatrix} \\
&= \begin{pmatrix} (W-B)\sin(\theta) \\ -(W-B)\cos(\theta)\sin(\phi) \\ -(W-B)\cos(\theta)\cos(\phi) \\ -(y_gW - y_bB)\cos(\theta)\cos(\phi) + (z_gW - z_b)\cos(\theta)\sin(\phi) \\ (z_gW - z_bB)\sin(\theta) + (x_gW - x_bB)\cos(\theta)\cos(\phi) \\ -(x_gW - x_bB)\cos(\theta)\sin(\phi) - (y_gW - y_bB)\sin(\theta) \end{pmatrix}.
\end{aligned} \tag{A.111}$$

A underwater vehicle is called buoyancy neutral if the its weight equals the buoyancy:

$$W = B. \tag{A.112}$$

In emergency situations, for instance power failure, it is better to design the underwater robot with $B > W$ (positive buoyancy) such that the vehicle will surface automatically. Note that the magnitude of B should only be slightly larger than W in this case, or too much energy is needed to keep the robot submerged, which contradict the power consumption minimizing goal.
Buoyancy neutral is one of the most important designing goals for our robot. For neutrally buoyant underwater robots $W = B$, equation A.111 therefore simplifies to

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ -(y_g - y_b)W\cos(\theta)\cos(\phi) + (z_g - z_b)W\cos(\theta)\sin(\phi) \\ (z_g - z_b)W\sin(\theta) + (x_g - x_b)W\cos(\theta)\cos(\phi) \\ -(x_g - x_b)W\cos(\theta)\sin(\phi) - (y_g - y_b)W\sin(\theta) \end{pmatrix}. \tag{A.113}$$

A.4 Dynamic Equation of Underwater Robots

Combining the kinematic equation A.82, added mass terms A.100, A.101, the damping term A.102 and the restoring term A.113, we obtain the complete underwater robot dynamic equation as follows:

$$\boldsymbol{M}_{RB}\dot{\upsilon} + \boldsymbol{M}_A\dot{\upsilon} + \boldsymbol{C}_{RB}(\upsilon)\upsilon + +\boldsymbol{C}_A(\upsilon)\upsilon + \boldsymbol{D}(\upsilon)\upsilon + \mathbf{g}(\eta) = \tau. \tag{A.114}$$

From this equation, we see that the restoring term $\mathbf{g}(\eta)$ introduces kinematic states into our robot dynamics. More precisely, for neutrally buoyant underwater robots, only the roll ϕ and pitch angle θ influence the dynamics.

Appendix B

Properties of Cross Product Operator

In A.3 we define the cross product operator, and in this chapter, some useful properties will be discussed. They will be used in Chapter 4 to prove the uniqueness of linearization of the error dynamics.

Identity B.1 *For cross product, we have the following property:*

$$\mu \times \nu = -\nu \times \mu, \tag{B.1}$$

and representing them with the cross product operator, we get

$$\mathbf{S}(\mu)\nu = -\mathbf{S}(\nu)\mu. \tag{B.2}$$

From [Fos11] (pp 25), the derivative of rotation matrix has the property:

$$\frac{d}{dt}\boldsymbol{R}(\lambda) = \boldsymbol{R}(\lambda)\boldsymbol{S}(\omega). \tag{B.3}$$

Differentiating the product of the rotation matrix $\boldsymbol{R}(\lambda)$ and an arbitrary vector $\mu \in \mathbb{R}^3$, we get

$$\begin{aligned}\frac{d}{dt}(\boldsymbol{R}(\lambda)\mu) &= \left(\frac{d}{dt}\boldsymbol{R}(\lambda)\right)\mu + \boldsymbol{R}(\lambda)\frac{d}{dt}\mu \\ &= \boldsymbol{R}\boldsymbol{S}(\omega)\mu + \boldsymbol{R}(\lambda)\frac{d}{dt}\mu \\ &= -\boldsymbol{R}\boldsymbol{S}(\mu)\omega + \boldsymbol{R}(\lambda)\frac{d}{dt}\mu.\end{aligned} \tag{B.4}$$

We can also calculate the derivative in an alternative way using the chain rule:

$$\begin{aligned}\frac{d}{dt}(\boldsymbol{R}(\lambda)\mu) &= \frac{d}{d\lambda}(\boldsymbol{R}(\lambda)\mu)\frac{d}{dt}\lambda + \boldsymbol{R}(\lambda)\frac{d}{dt}\mu \\ &= \frac{d}{d\lambda}(\boldsymbol{R}(\lambda)\mu)\boldsymbol{Q}(\lambda)\omega + \boldsymbol{R}^{-1}(\lambda)\frac{d}{dt}\mu.\end{aligned} \tag{B.5}$$

After comparison of B.4 and B.5, we have the following identity:

Identity B.2 *The derivative of the rotation matrix and an arbitrary vector $\mu \in \mathbb{R}^3$ with respect to the Euler angle* 1

$$\frac{d}{d\lambda}(\mathbf{R}(\lambda)\mu) = -\mathbf{R}(\lambda)\mathbf{S}(\mu)\mathbf{Q}^{-1}(\lambda). \tag{B.6}$$

Then, we want to derive another identity.

$$\begin{aligned} \frac{d}{dt}\boldsymbol{I}_{3\times3} &= \frac{d}{dt}(\boldsymbol{R}^{-1}(\lambda)\boldsymbol{R}(\lambda)) \\ &= \left(\frac{d}{dt}\boldsymbol{R}^{-1}(\lambda)\right)\boldsymbol{R}(\lambda) + \boldsymbol{R}^{-1}(\lambda)\left(\frac{d}{dt}\boldsymbol{R}(\lambda)\right) = \boldsymbol{0}_{3\times3}. \end{aligned} \tag{B.7}$$

Identity B.3 *Thus, for the time derivative of the inverse rotation matrix*

$$\begin{aligned} \frac{d}{dt}\left((\mathbf{R}^{-1}(\lambda)\right) &= -\mathbf{R}^{-1}(\lambda)\left(\frac{d}{dt}\mathbf{R}(\lambda)\right)\mathbf{R}^{-1}(\lambda) \\ &= -\mathbf{R}^{-1}(\lambda)\left(\mathbf{R}(\lambda)\mathbf{S}(\omega)\right)\mathbf{R}^{-1}(\lambda) \\ &= -\mathbf{S}(\omega)\mathbf{R}^{-1}(\lambda). \end{aligned} \tag{B.8}$$

Based on this identity, we derive our fourth identity in a similar way with B.2. Firstly, we take time derivative of the product of the product of inverse rotation matrix and arbitrary vector $\mu \in \mathbb{R}^3$:

$$\begin{aligned} \frac{d}{dt}\left(\boldsymbol{R}^{-1}(\lambda)\mu\right) &= \left(\frac{d}{dt}\boldsymbol{R}^{-1}(\lambda)\right)\mu + \boldsymbol{R}^{-1}(\lambda)\frac{d}{dt}\mu \\ &= \boldsymbol{S}(\boldsymbol{R}^{-1}(\lambda)\mu)\omega + \boldsymbol{R}^{-1}(\lambda)\frac{d}{dt}\mu. \end{aligned} \tag{B.9}$$

Alternatively, We can also calculate the derivative using the chain rule:

$$\begin{aligned} \frac{d}{dt}(\boldsymbol{R}^{-1}(\lambda)\mu) &= \frac{d}{d\lambda}(\boldsymbol{R}^{-1}(\lambda))\frac{d}{dt}\lambda + \boldsymbol{R}^{-1}(\lambda)\frac{d}{dt}\mu \\ &= \frac{d}{d\lambda}(\boldsymbol{R}^{-1}(\lambda)\mu)\boldsymbol{Q}(\lambda)\omega + \boldsymbol{R}^{-1}(\lambda)\frac{d}{dt}\mu. \end{aligned} \tag{B.10}$$

Comparing B.9 and B.10, we get the fourth identity:

Identity B.4 *The derivative of product of inverse rotation matrix and an arbitrary vector $\mu \in \mathbb{R}^3$ with respecto to the Euler angles λ satisfies:*

$$\frac{d}{d\lambda}(\mathbf{R}^{-1}(\lambda)\mu) = \mathbf{S}(\mathbf{R}^{-1}(\lambda)\mu)\mathbf{Q}^{-1}(\lambda). \tag{B.11}$$

Appendix C

Accurate Modeling of Hydrodynamic Damping

This chapter focuses on the computation of hydrodynamic damping of the hull and fins. Firstly, a calculation method of lift and drag is introduced for the general case. Afterwards, explicit calculation steps for fins and hull are discussed.

C.1 Basics of Hydrodynamics

In fluid dynamics, the angle of attack (alway denoted by the Greek letter α) is the angle between the reference line on a body (often the chord line of an airfoil) and the vector representing the relative motion between the body and the fluid through which it is moving [Hal15].

We have already specified a body frame $\{b\}$ whose origin coincides with the geometric center of the hull for describing the underwater robot dynamics. To describe the hydrodynamic damping on fins, we should define other body-fixed frames, so called fin frames $\{f\}$, with the origin approximately placed on the geometric center of each fin. The fin frame $\{f\}$ can be obtained from the body frame $\{b\}$ in the same way as the transformation from world frame $\{i\}$ to body frame $\{b\}$. First we translate the body frame $\{b\}$ until its origin is placed at the fin geometric center. Then, rotate the frame such that the xy-plane of the fin frame $\{f\}$ is exactly located in the fin reference plane and the y-axis points out from the robot body.

The flow frame $\{flow\}$ is also a body-fixed frame placed at a given point of the robot whose x axis is oriented as the velocity of the robot with respect to the fluid at this point and z-axis lying on the xz plane of the previous body frame. The transformation from the previous body fixed frame ($\{b\}$ for hull, $\{f\}$ for fins) to flow frame $\{flow\}$ is performed in two steps. First, rotating the body frame in the clockwise direction of an attack angle α about its y-axis which yields a new coordinate frame called stability axes. Then, rotate this stability frame about its z-axis by a sideslip angle β. The essential advantage of defining a flow frame is that the velocity of a robot body component relative to the fluid

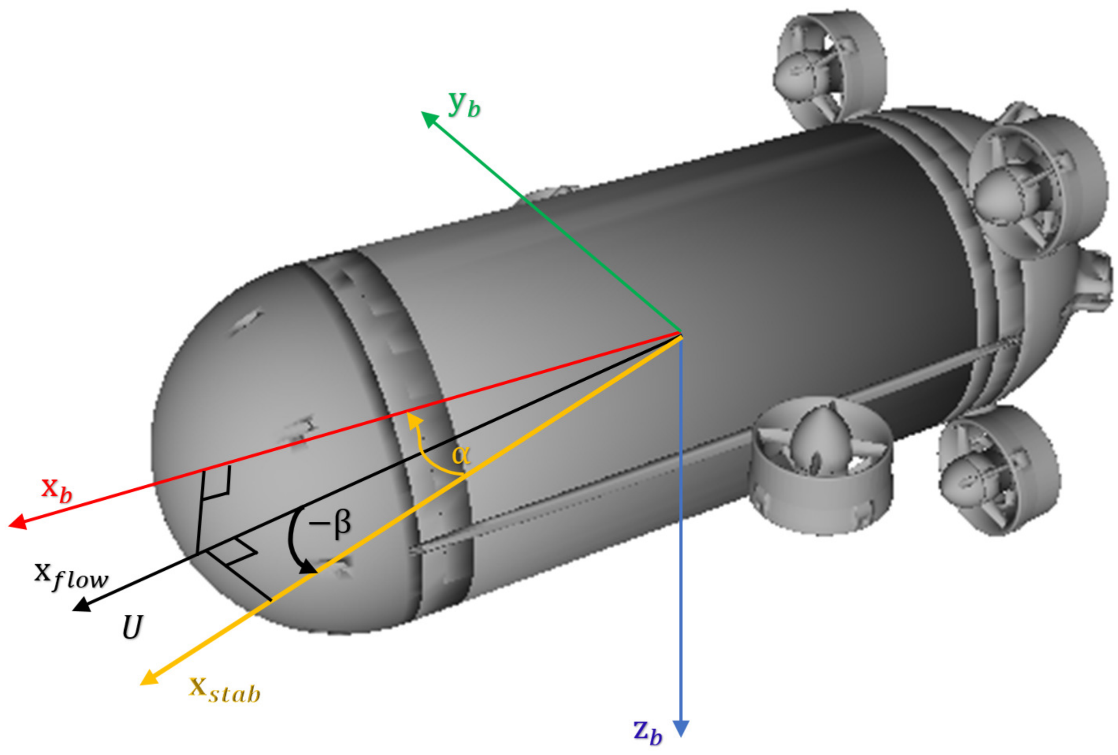


Figure C.1: Flow frame and stability frame

has only a x-component in the flow frame $\{flow\}$. In addition, in the flow frame $\{flow\}$, the drag and lift are totally decoupled. Drag will be always in the negative x-direction and lift is located in the z-axis.
The relationship between velocities in body, stability and flow frame can be expressed mathematically as follows:

$$\mathbf{v}^{stab} = \boldsymbol{R}_{y,\alpha}\mathbf{v}^{b}, \tag{C.1}$$

$$\mathbf{v}^{flow} = \boldsymbol{R}_{z,-\beta}\mathbf{v}^{stab}, \tag{C.2}$$

where

$$\boldsymbol{R}_{y,\alpha} = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \tag{C.3}$$

and

$$\boldsymbol{R}_{z,-\beta} = \begin{pmatrix} \cos(\beta) & \sin(\beta) & 0 \\ -\sin(\beta) & \cos(\beta) & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{C.4}$$

Then, we can obtain the transformation matrix from body frame to flow frame

$$\boldsymbol{R}_b^{flow} = \boldsymbol{R}_{z,-\beta}\boldsymbol{R}_{y,\alpha} = \begin{pmatrix} \cos(\beta)\cos(\alpha) & \sin(\beta) & \cos(\beta)\sin(\alpha) \\ -\sin(\beta)\cos(\alpha) & \cos(\beta) & -\sin(\beta)\sin(\alpha) \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix}. \tag{C.5}$$

The transformation matrix from flow frame to body frame is

$$\begin{aligned} \boldsymbol{R}_{flow}^b &= (\boldsymbol{R}_b^{flow})^T = (\boldsymbol{R}_{z,-\beta}\boldsymbol{R}_{y,\alpha})^T \\ &= \boldsymbol{R}_{y,\alpha}^T\boldsymbol{R}_{z,-\beta}^T = \boldsymbol{R}_{y,-\alpha}\boldsymbol{R}_{z,\beta} \\ &= \begin{pmatrix} \cos(\beta)\cos(\alpha) & -\sin(\beta)\cos(\alpha) & -\sin(\alpha) \\ \sin(\beta) & \cos(\beta) & 0 \\ \cos(\beta)\sin(\alpha) & -\sin(\beta)\sin(\alpha) & \cos(\alpha) \end{pmatrix}. \end{aligned} \tag{C.6}$$

The velocity transformation from body frame to flow frame can be written as

$$\mathbf{v}^{flow} = \boldsymbol{R}_b^{flow}\mathbf{v}^b \tag{C.7}$$

and

$$\mathbf{v}^b = \boldsymbol{R}_{flow}^b\mathbf{v}^{flow}. \tag{C.8}$$

Expressing the velocities in separate components:

$$\begin{pmatrix} U \\ 0 \\ 0 \end{pmatrix} = \boldsymbol{R}_b^{flow}\begin{pmatrix} u \\ v \\ w \end{pmatrix}, \tag{C.9}$$

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \boldsymbol{R}_{flow}^b\begin{pmatrix} U \\ 0 \\ 0 \end{pmatrix}. \tag{C.10}$$

From the upper equation we can see that only the x-component of the velocity remains after transformation from body frame to flow frame.
Combining C.10 and C.6, we obtain

$$u = U\cos(\alpha)\cos(\beta), \tag{C.11}$$

$$v = U\sin(\beta), \tag{C.12}$$

$$w = U\sin(\alpha)\cos(\beta), \tag{C.13}$$

and

$$U = \sqrt{u^2 + v^2 + w^2}. \tag{C.14}$$

The angle of attack α and sideslip angle β can be calculated from the velocities:

$$\alpha = atan2(w, u), \tag{C.15}$$

$$\beta = atan2(v, U). \tag{C.16}$$

Until now, we consider that a robot work in the ideal condition of still water. However, in the actual situation, underwater robots need to perform tasks in the ocean or in the river where moving water currents influence the robot dynamic behavior considerably. Thus, it is more realistic to extend the underwater robot model under consideration of current motion. Assume that the current velocities expressed in body frame $\{b\}$ are denoted by u_c, v_c, w_c, p_c, q_c and r_c. The velocities of the robot body relative to currents are

$$u_r = u - u_c, \tag{C.17}$$

$$v_r = v - v_c, \tag{C.18}$$

$$w_r = w - w_c, \tag{C.19}$$

$$p_r = p - p_c, \tag{C.20}$$

$$q_r = q - q_c, \tag{C.21}$$

$$r_r = r - r_c. \tag{C.22}$$

C.11, C.12, C.13 and C.14 can be rewritten as

$$u_r = U_r \cos(\alpha)\cos(\beta), \tag{C.23}$$

$$v_r = U_r \sin(\beta), \tag{C.24}$$

and

$$w_r = U_r \sin(\alpha)\cos(\beta), \tag{C.25}$$

and

$$U_r = \sqrt{u_r^2 + v_r^2 + w_r^2}. \tag{C.26}$$

Calculation of attack angle and sideslip angle can be also extended to

$$\alpha_r = atan2(w_r, u_r) \tag{C.27}$$

and

$$\beta_r = atan2(v_r, U_r). \tag{C.28}$$

The underwater robot dynamic model A.114 should also be modified after we consider the motion of the fluid:

$$\boldsymbol{M}_{RB}\dot{\upsilon} + \boldsymbol{M}_A\dot{\upsilon}_r + \boldsymbol{C}_{RB}(\upsilon)\upsilon + \boldsymbol{C}_A(\upsilon_r)\upsilon_r + \boldsymbol{D}(\upsilon_r)\upsilon_r + \mathbf{g}(\eta) = \tau, \tag{C.29}$$

where $\upsilon_r = (\mathbf{v}^b_{br/i}, \omega^b_{br/i})^T \in \mathbb{R}^6$ and

$$\mathbf{v}^b_{br/i} = \begin{pmatrix} u_r \\ v_r \\ w_r \end{pmatrix}, \tag{C.30}$$

$$\omega^b_{br/i} = \begin{pmatrix} p_r \\ q_r \\ r_r \end{pmatrix}. \tag{C.31}$$

Lift and drag force are expressed in flow frame $\{flow\}$ and written as follows:

$$L = \frac{1}{2} C_L S \rho U_r^2, \tag{C.32}$$

$$D = \frac{1}{2} C_D S \rho U_r^2. \tag{C.33}$$

where L represents the lift, D represents the drag, ρ is the fluid density, U is the speed of the underwater robot relative to the fluid in which it is moving, S is the effective surface area of the robot body component, and C_L and C_D are lift and drag coefficients, respectively.

The lift and drag coefficients relate the lift force and drag force to the fluid density around the robot, the fluid velocity and the reference area. They are not constant but depend on flow speed, flow direction, fluid density, fluid velocity and are shape-specific. They should be determined experimentally for each underwater robot. Speed, kinematic viscosity and a characteristic length scale of the object are incorporated into a dimensionless quantity called the Reynolds number Re [Dra17]. The lift and drag coefficients are functions of Reynolds number Re, Mach number Ma and angle of attack α.

Both coefficients are functions of angle of attack α, but their functional dependence on it is different for every robot and every robot component. Nevertheless, the lift coefficient characteristic and drag coefficient characteristic are structure-fixed.

Figures C.2 and C.3 show the typical structure of lift and drag coefficient curves depending on the angle of attack α. At a lower angle of attack, the lift coefficient shows an almost linear dependence on it as α increases. The stall angle is defined as the angle at which the lift coefficient reaches its maximum. Afterwards, the lift coefficient deceases with increasing angle of attack. The drag coefficient increments always with increasing angle of attack.

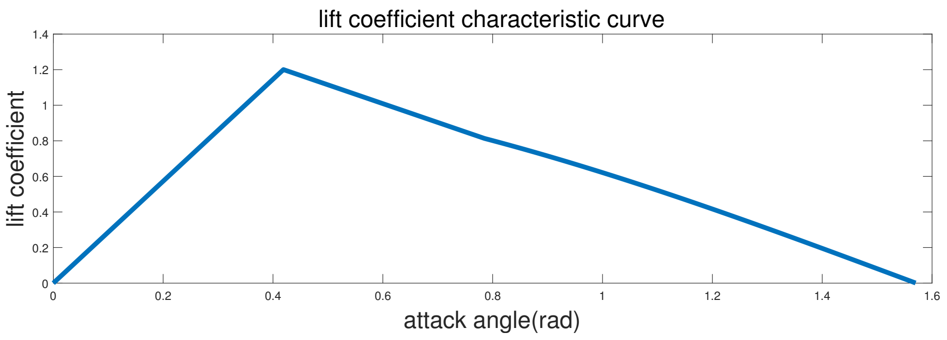


Figure C.2: Lift coefficient characteristic curve [CI07]

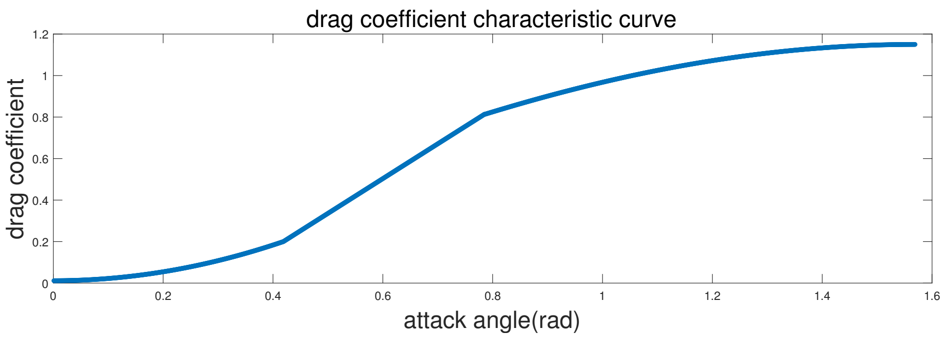


Figure C.3: Drag coefficient characteristic curve [CI07]

C.2 Hydrodynamic Damping on Fins

The vertical and horizontal fins are installed at the stern of *Snookie* to balance the pitch and yaw instability of forward motion. Regard the vertical fin as two fins on the upper and lower sides and horizontal fin as two fins on the left and right side, that is, four fins are placed symmetrically in difference of 90°. Assume the total hydrodynamic force exerts on the geometric centre of fins. Definitions of fin frames $\{f\}$ are illustrated in C.9, C.10.

The calculation and transformation of hydrodynamic damping of each fin are performed in the following steps:

1. The relative generalized velocity vector υ_r^b of our underwater robot is measurable, where

$$\upsilon_r^b = \begin{pmatrix} \mathbf{v}_{rb/i}^b \\ \omega_{rb/i}^b \end{pmatrix} = \begin{pmatrix} u_r \\ v_r \\ w_r \\ p_r \\ q_r \\ r_r \end{pmatrix}. \tag{C.34}$$

2. Calculate the velocity of the fin center point in body frame $\{b\}$ using A.45:

$$\mathbf{v}_{r,fin}^b = \mathbf{v}_r^b + \omega_r^b \times \mathbf{r}_{fin}^b, \tag{C.35}$$

   where $\mathbf{r}_{fin}^b$ is the position vector of a fin geometric center in body frame $\{b\}$.

3. Transform the relative velocity $\mathbf{v}_{r,fin}^b$ into the fin frame $\{f\}$ using

$$\mathbf{v}_{r,fin}^f = \boldsymbol{R}_b^f \mathbf{v}_{r,fin}^b \tag{C.36}$$

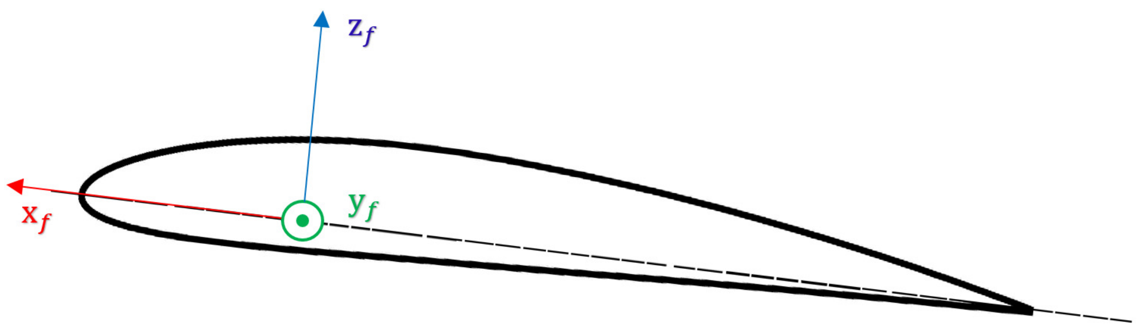


Figure C.4: Definition of fin frame $\{f\}$

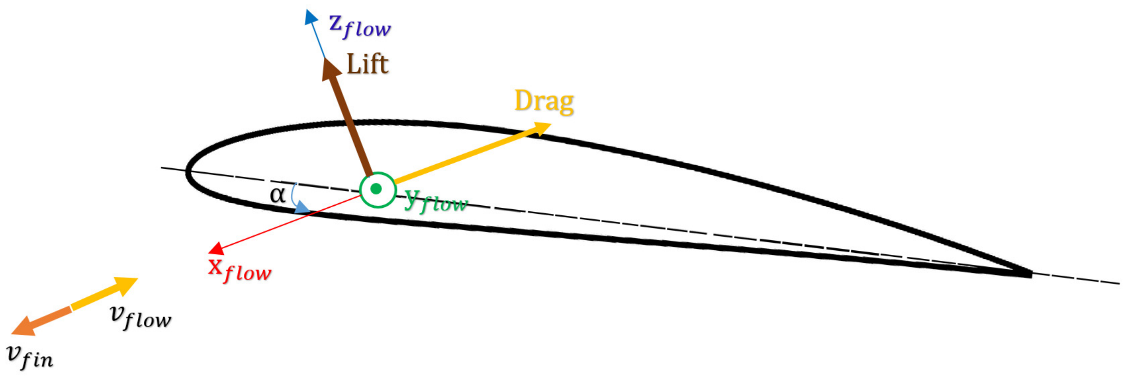


Figure C.5: Flow frame $\{flow\}$ and hydrodynamic damping for $\alpha \in [0, -\frac{\pi}{2}]$

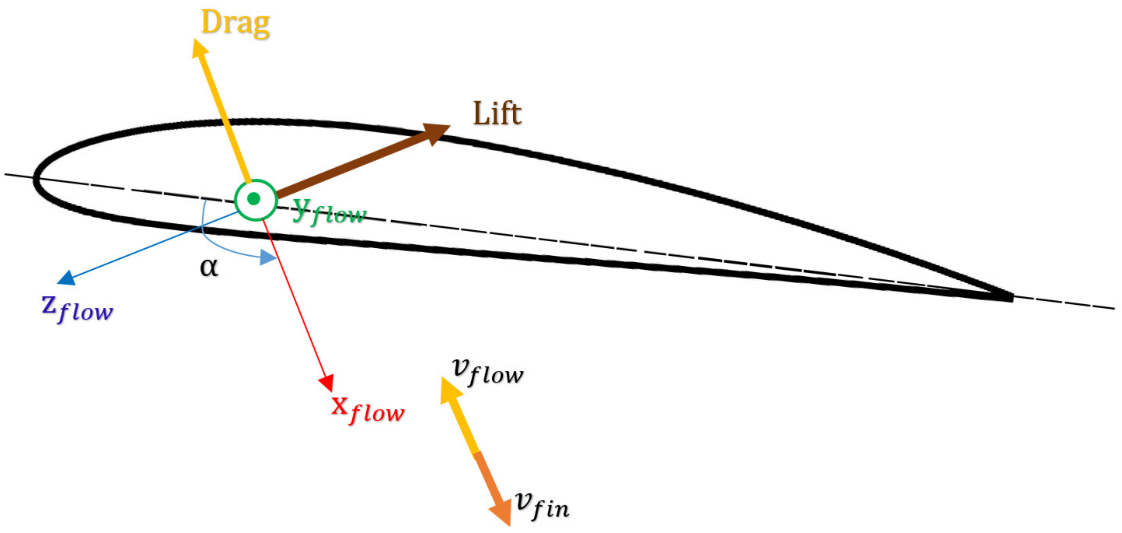


Figure C.6: Flow frame $\{flow\}$ and hydrodynamic damping for $\alpha \in [-\frac{\pi}{2}, -\pi]$

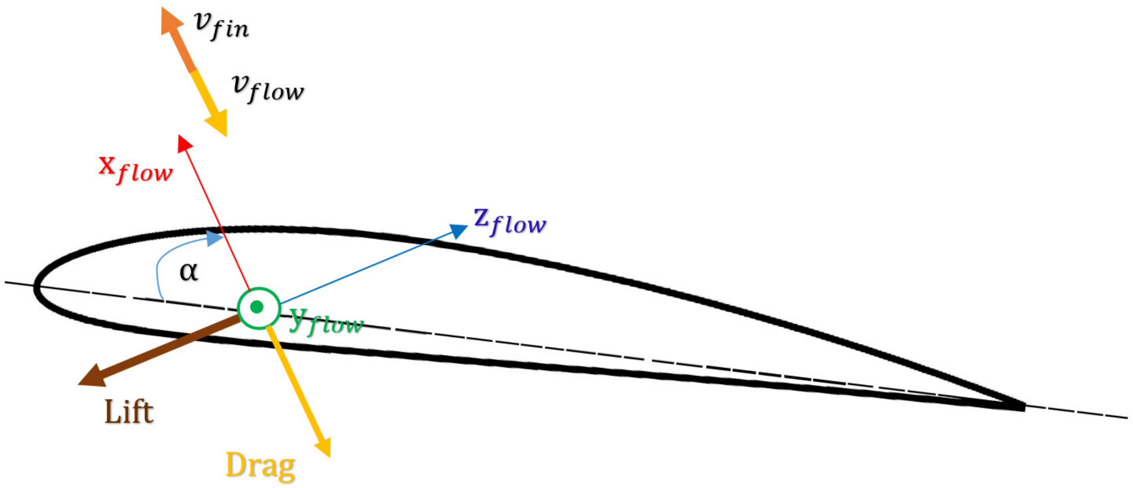


Figure C.7: Flow frame $\{flow\}$ and hydrodynamic damping for $\alpha \in [0, \frac{\pi}{2}]$

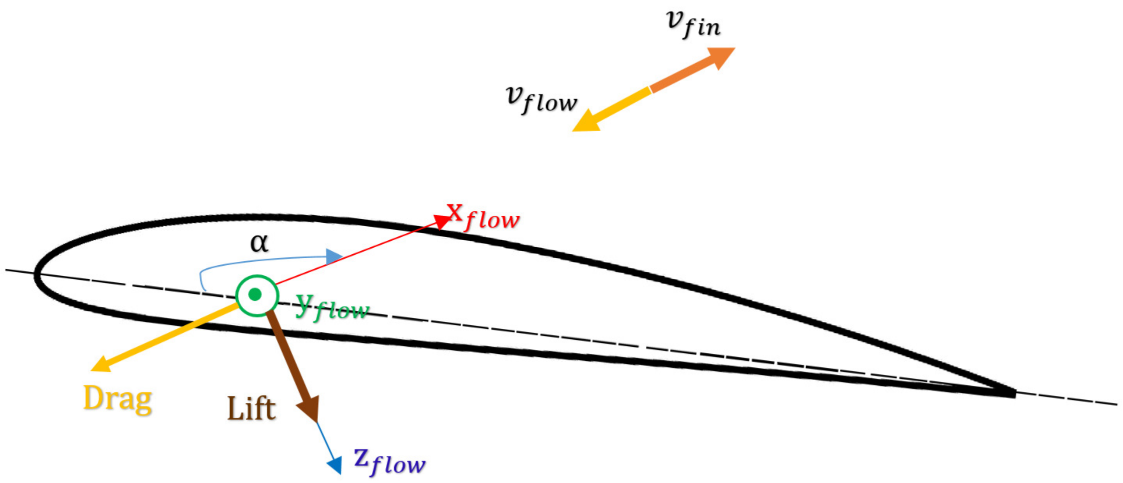


Figure C.8: Flow frame $\{flow\}$ and hydrodynamic damping for $\alpha \in [\frac{\pi}{2}, \pi]$

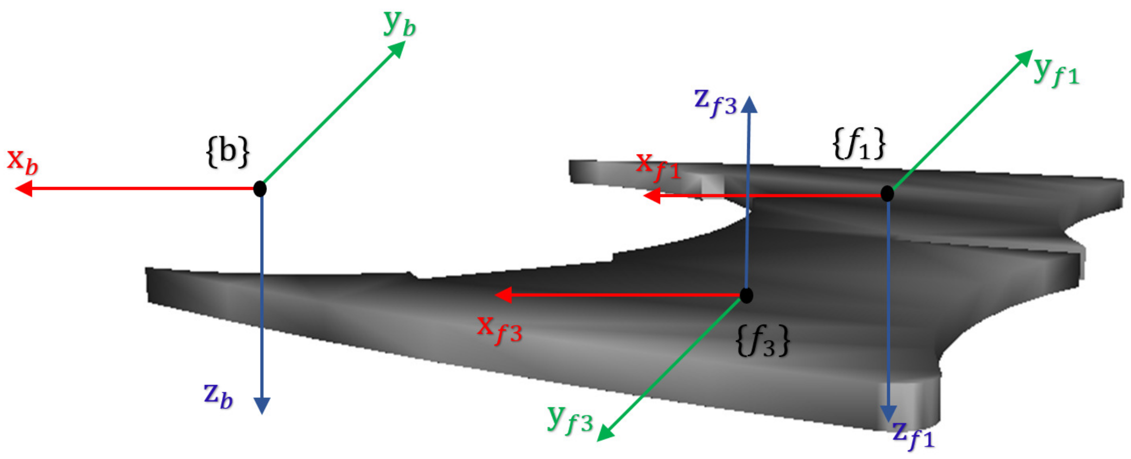


Figure C.9: Fin frames for horizontal fin of *Snoookie*

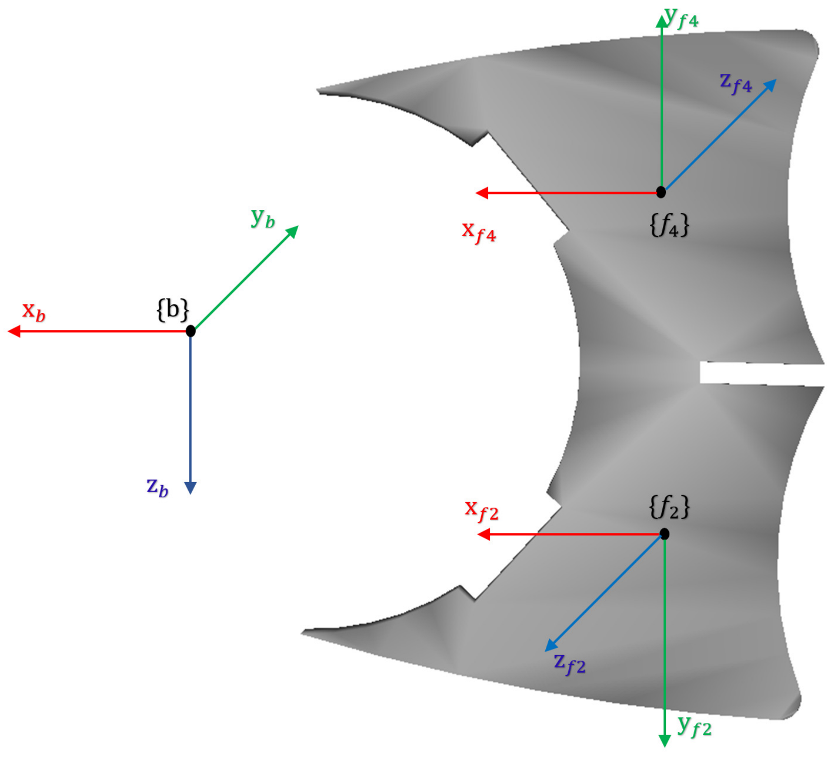


Figure C.10: Fin frames for vertical fin of *Snoookie*

4. Calculate the attack angle α in the fin frame $\{f\}$:

$$\alpha = atan2(\mathbf{v}^f_{r,fin}(3), \mathbf{v}^f_{r,fin}(1)). \tag{C.37}$$

5. Calculate the rotation matrix $\boldsymbol{R}^{flow}_f$ from fin frame $\{f\}$ to flow frame $\{flow\}$ based on the attack angle α according to C.5.

6. Transform the relative velocity vector of the fin with respect to the fluid $\mathbf{v}^f_{r,fin}$ to flow frame $\{flow\}$:

$$\mathbf{v}^{flow}_{r,fin} = \boldsymbol{R}^{flow}_f \mathbf{v}^f_{r,fin} = \begin{pmatrix} U_{r,fin} \\ 0 \\ 0 \end{pmatrix} \tag{C.38}$$

7. Determine the lift coefficient C_L and drag coefficient C_D based on the attack angle α according to the lift coefficient characteristic curve and the drag coefficient curve, respectively.

8. Calculate lift L_{fin} and drag D_{fin} in flow frame $\{flow\}$:

$$L_{fin} = \frac{1}{2} C_L S \rho U^2_{r,fin}, \tag{C.39}$$

$$D_{fin} = \frac{1}{2} C_D S \rho U^2_{r,fin}. \tag{C.40}$$

Define

$$\mathbf{f}^{flow}_f = \begin{pmatrix} D_{fin} \\ 0 \\ L_{fin} \end{pmatrix}. \tag{C.41}$$

9. Transfer the lift L_{fin} and drag D_{fin} into body frame $\{b\}$ and calculate the resulting moments:

$$\mathbf{f}^b_F = \boldsymbol{R}^f_{flow} \boldsymbol{R}^b_f \mathbf{f}^{flow}_f. \tag{C.42}$$

The moments of fin in body frame $\{b\}$ is calculated as

$$\mathbf{m}^b_F = \mathbf{r}^b_{fin} \times \mathbf{f}^b_F. \tag{C.43}$$

C.3 Hydrodynamic Damping on Hull

Calculation of hydrodynamic damping on the hull is much simpler since it shares the same body frame as the underwater robot dynamics. The steps for the calculation of hydrodynamic damping on the hull are as follows:

1. The relative generalized velocity vector υ_r^b of the underwater robot is measurable, where

$$\upsilon_{r,hull}^b = \upsilon_r^b = \begin{pmatrix} \mathbf{v}_{rb/i}^b \\ \omega_{rb/i}^b \end{pmatrix} = \begin{pmatrix} u_r \\ v_r \\ w_r \\ p_r \\ q_r \\ r_r \end{pmatrix}. \tag{C.44}$$

2. Calculate the attack angle α in body frame $\{b\}$:

$$\alpha = atan2(\mathbf{v}_{r,hull}^b(3), \mathbf{v}_{r,hull}^b(1)). \tag{C.45}$$

3. Calculate the rotation matrix $\boldsymbol{R}_b^{flow}$ from body frame $\{b\}$ to flow frame $\{flow\}$ based on attack angle α according to C.5.

4. Transform the relative velocity vector of the hull with respect to the fluid $\mathbf{v}_{r,hull}^b$ to flow frame $\{flow\}$:

$$\mathbf{v}_{r,hull}^{flow} = \boldsymbol{R}_b^{flow} \mathbf{v}_{r,hull}^f = \begin{pmatrix} U_{r,hull} \\ 0 \\ 0 \end{pmatrix} \tag{C.46}$$

5. Determine the lift coefficient C_L and drag coefficient C_D based on the attack angle α according to the lift coefficient characteristic curve and the drag coefficient curve, respectively.

6. Calculate lift L_{hull} and drag D_{hull} in flow frame $\{flow\}$:

$$L_{hull} = \frac{1}{2} C_L S \rho U_{r,hull}^2, \tag{C.47}$$

$$D_{hull} = \frac{1}{2} C_D S \rho U_{r,hull}^2. \tag{C.48}$$

Define

$$\mathbf{f}_H^{flow} = \begin{pmatrix} D_{hull} \\ 0 \\ L_{hull} \end{pmatrix}. \tag{C.49}$$

7. Transfer the lift L_{hull} and drag D_{hull} into body frame $\{b\}$:

$$\mathbf{f}_H^b = \boldsymbol{R}_{flow}^b \mathbf{f}_H^{flow}. \tag{C.50}$$

Since hull uses the same body-fixed frame with the underwater robot dynamics (the geometric center of hull), and the exerting point of hydrodynamic damping on hull is the geometric center , lift and drag on hull will not induce any moment.

List of Figures

Notations

$\{\mathcal{T}\}$ Trim frame

$\{FS\}$ Frenet-Serret frame

$\{i\}$ Inertial world frame

$\{b\}$ Body frame

$\{f\}$ Fin frame

$\{flow\}$ Flow frame

$\boldsymbol{Q}$ Linear velocity transform matrix from body frame $\{b\}$ to inertial world frame $\{i\}$

$\boldsymbol{R}$ Angular velocity transform matrix from body frame $\{b\}$ to inertial world frame $\{i\}$

$\mathcal{T}$ Trim trajectory

$\eta_{\mathcal{T}}$ Trim trajectory kinematic specification

m Number of trim trajectories

n Number of actuators

n_t Number of thrusters

n_f Number of fins

k Index for optimization iteration

k_{total} Total number of optimization iterations

s Index for spin direction configuration

$\mathbf{p}$ Position vector

λ Orientation vector

$\mathbf{v}$ Linear velocity vector

ω	angular velocity vector
$\boldsymbol{M}_{RB}$	Rigid body inertia matrix
$\boldsymbol{C}_{RB}$	Rigid body Coriolis matrix
$\boldsymbol{M}_{A}$	Added mass inertia matrix
$\boldsymbol{C}_{A}$	Added mass Coriolis matrix
$\boldsymbol{M}$	Complete inertia matrix
$\boldsymbol{C}$	Complete Coriolis matrix
$\boldsymbol{D}$	Damping matrix
$\mathbf{g}$	Restoring vector
τ	Generalized force vector
$\mathbf{f}$	Force vector
$\mathbf{m}$	Moment vector
υ	Generalized velocity vector
CO	Hull geometry center
CG	Center of gravity
CG	Center of buoyancy
B	Buoyancy
G	Gravity
L	Lift force
D	Drag force
C_L	Lift coefficient
C_L	Drag coefficient
c_L	Approximated constant lift coefficient
c_D	Approximated constant drag coefficient
u	Robot surge velocity in body frame $\{b\}$
v	Robot sway velocity in body frame $\{b\}$

w Robot heave velocity in body frame $\{b\}$

p Robot roll velocity in body frame $\{b\}$

q Robot pitch velocity in body frame $\{b\}$

r Robot yaw velocity in body frame $\{b\}$

x Robot position (x-component) in inertial world frame $\{i\}$

y Robot position (y-component) in inertial world frame $\{i\}$

z Robot position (z-component) in inertial world frame $\{i\}$

ϕ Robot roll angle

θ Robot pitch angle

ψ Robot yaw angle

α Angle of attack

ρ Fluid density

$\gamma_{\mathcal{T}}$ Motion path angle

δ_F Deflection angle of fin

$\mathbf{r}_G$ Robot center of mass in body frame $\{b\}$

$\mathbf{r}_F$ Position of fin geometric center in body frame $\{b\}$

$\mathbf{r}_T$ Position of thruster in body frame $\{b\}$

$\mathbf{d}_T$ Motion orientation of thruster in body frame $\{b\}$

λ_T Thrust-moment ratio of thruster

b_T Motion spin of thruster

l_H Hull length

r_H Hull radius

d_H Hull diameter

x_F Position fin of geometric center (x-component) in body frame $\{b\}$

a_F Length of fin

b_F Width of fin

γ_F Orientation angle of fin

m_{total} Total weight of robot

V Total volume of robot

$\boldsymbol{I}_g$ Moment of ineratia with respect to CG

$\boldsymbol{I}_b$ Moment of inertia with respect to CO

Bibliography

[Ant14] G. Antonelli. *Underwater Robots*. Springer International Publishing, 2014.

[BA05] L. Beji and A. Abichou. Tracking control of trim trajectories of a blimp for ascent and descent flight manoeuvres. *International Journal of Control*, 78(10):706–719, 2005. URL: http://dx.doi.org/10.1080/00207170500118643, arXiv:http://dx.doi.org/10.1080/00207170500118643, doi:10.1080/00207170500118643.

[BLS08] C. L. Bottasso, D. Leonello, and B. Savini. Path planning for autonomous vehicles by trajectory smoothing using motion primitives. *IEEE Transactions on Control Systems Technology*, 16(6):1152–1168, Nov 2008. doi:10.1109/TCST.2008.917870.

[Che07] Y. Chen. Modular modeling and control for autonomous underwater vehicle (auv). Master's thesis, National University of Singapore, Dept. of Mechanical Engineering, 2007.

[CI07] G. Campa and M. Innocenti. Model of an underwater vehicle, 2007. [Online; accessed 01-November-2016]. URL: https://de.mathworks.com/matlabcentral/fileexchange/1207-shark.

[CW12] Jonathan Currie and David I. Wilson. OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User. In N. Sahinidis and J. Pinto, editors, *Foundations of Computer-Aided Process Operations*, Savannah, Georgia, USA, 8–11 January 2012.

[Dra17] Drag Coefficient. Drag coefficient — Wikipedia, the free encyclopedia, 2017. [Online; accessed 01-March-2017]. URL: https://en.wikipedia.org/wiki/Drag_coefficient.

[DSZ+16] T. Du, A. Schulz, B. Zhu, B. Bickel, and W. Matusik. Computational multicopter design. *ACM Trans. Graph.*, 35(6):227:1–227:10, November 2016. URL: http://doi.acm.org/10.1145/2980179.2982427, doi:10.1145/2980179.2982427.

[FF95] T. I. Fossen and O. Fjellstad. Nonlinear modelling of marine vehicles in 6 degrees of freedom. *Journal of Mathematical Modelling of Systems*, 1(1), 1995.

[FJP08] T. I. Fossen, T. A. Johansen, and T. Perez. A Survey of Control Allocation Methods for Underwater Vehicles. *Book, Chapter*, (December):20, 2008. doi:10.1109/MED.2006.235844.

[Fos94] T. I. Fossen. *Guidance and Control of Ocean Vehicles*. John Wiley & Sons Ltd., 1994.

[Fos11] T. I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons Ltd., 2011.

[FS91] Thor I. Fossen and S. I. Sagatun. Adaptive control of nonlinear systems: A case study of underwater robotic systems. *Journal of Robotic Systems*, 8(3):393–412, 1991. URL: http://dx.doi.org/10.1002/rob.4620080307, doi:10.1002/rob.4620080307.

[GB08] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. http://stanford.edu/~boyd/graph_dcp.html.

[GB14] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[Hal15] N. Hall. Nasa aeronautics guided tour, 2015. URL: https://www.grc.nasa.gov/WWW/k-12/airplane/incline.html.

[JFB04] T. A. Johansen, T. I. Fossen, and S. P. Berge. Constrained nonlinear control allocation with singularity avoidance using sequential quadratic programming. *IEEE Transactions on Control Systems Technology*, 12(1):211–216, Jan 2004. doi:10.1109/TCST.2003.821952.

[JFT05] T. A. Johansen, T. I. Fossen, and P. Tondel. Efficient optimal constrained control allocation via multiparametric programming. *Journal of Guidance, Control and Dynamics*, 28(3), 2005.

[JK91] J. L. Junkins and Y. Kim. Measure of controllability for actuator placement. *Journal of Guidance, Control, and Dynamics*, 14(5):895–902, 1991. URL: http://arc.aiaa.org/doi/10.2514/3.20729, doi:10.2514/3.20729.

[JSHL12] T. Joung, K. Sammut, F. He, and S. Lee. Shape optimization of an autonomous underwater vehicle with a ducted propeller using computational fluid dynamics analysis. *International Journal of Naval Architecture and Ocean Engineering*, 4(1):44 – 56, 2012. URL: `http://www.sciencedirect.com/science/article/pii/S2092678216301881`, `doi:https://doi.org/10.2478/IJNAOE-2013-0077`.

[Moo81] B. Moore. Principal component analysis in linear systems: Controllability, observability, and model reduction. *IEEE Transactions on Automatic Control*, 26(1):17–32, 1981. `doi:10.1109/TAC.1981.1102568`.

[MSDE16] A. A. A. Makdah, E. Shammas, N. Daher, and I. ElHajj. Modeling and optimal three-dimensional trajectory tracking for an autonomous underwater vehicle. In *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 172–177, July 2016. `doi:10.1109/AIM.2016.7576762`.

[NM12] R. Naldi and L. Marconi. Control allocation for a ducted-fan aerial robot employing both lift and drag forces. In *2012 American Control Conference (ACC)*, pages 1726–1731, June 2012. `doi:10.1109/ACC.2012.6314880`.

[NN06] M. Narimani and M. Narimani. Design of adaptive-sliding mode controller for positioning control of underwater robotics. In *2006 Canadian Conference on Electrical and Computer Engineering*, pages 414–417, May 2006. `doi:10.1109/CCECE.2006.277781`.

[NS04] S. K. Nagar and S. K. Singh. An algorithmic approach for system decomposition and balanced realized model reduction. *Journal of the Franklin Institute*, 341(7):615–630, 2004. `doi:10.1016/j.jfranklin.2004.07.005`.

[Seb15] Y. B. Sebbane. *Lighter than Air Robots*, volume 1542. 2015. `arXiv:arXiv:1011.1669v3`, `doi:10.1017/CBO9781107415324.004`.

[SP05] S. Skogestad and I. Postlethwaite. *Multivariable feedback control: analysis and design*. 2005.

[SPK02a] C. Silvestre, A. Pascoal, and I. Kaminer. On the design of gain-scheduled trajectory tracking controllers. *International Journal of Robust and Nonlinear Control*, 12(9):797–839, 2002. `doi:10.1002/rnc.705`.

[SPK02b] C. Silvestre, A. Pascoal, and I. Kaminer. On the design of gain-scheduled trajectory tracking controllers. *International Journal of Robust and Nonlinear Control*, 12(9):797–839, 2002.

[SSL$^+$14] A. Schulz, A. Shamir, D. I. W. Levin, P. Sitthi-amorn, and W. Matusik. Design and fabrication by example. *ACM Trans. Graph.*, 33(4):62:1–62:11, July 2014. URL: http://doi.acm.org/10.1145/2601097.2601127, doi:10.1145/2601097.2601127.

[SSS+17] A. Schulz, C. Sung, A. Spielberg, W. Zhao, R. Cheng, E. Grinspun, D. Rus, and W. Matusik. Interactive robogami: An end-to-end system for design of robots with ground locomotion. *The International Journal of Robotics Research*, 36(10):1131–1147, 2017. URL: https://doi.org/10.1177/0278364917723465, arXiv:https://doi.org/10.1177/0278364917723465, doi:10.1177/0278364917723465.

[SWM+07] R. Shorten, F. Wirth, O. Mason, K. Wulff, and C. King. Stability Criteria for Switched and Hybrid Systems. *SIAM Review*, 49(4):545–592, 2007. doi:10.1137/05063516X.

[Tan99] S. Tang. Modeling and simulation of the autonomous underwater vehicle, autolycus. Master's thesis, Massachusetts Institute of Technology, Dept. of Ocean Engineering, 1999.

[VSHvH14] A.N. Vollmayr, S. Sosnowski, S. Hirche, and J.L. van Hemmen. Snookie: an autonomous underwater vehicle with artificial lateral line system. In Horst Bleckmann, Joachim Mogdans, and Sheryl L. Coombs, editors, *Flow Sensing in Air and Water*, chapter 20, pages 521–562. Springer Berlin Heidelberg, Berlin, Heidelberg, viii edition, 2014. doi:10.1007/978-3-642-41446-6_20.

[VVKR05] I. Vasilescu, P. Varshavskaya, K. Kotay, and D. Rus. Autonomous modular optical underwater robot (amour) design, prototype and feasibility study. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1603–1609, April 2005. doi:10.1109/ROBOT.2005.1570343.

[Wan09] L. Wang. *Model Predictive Control System Design and Implementation Using MATLAB®*. 2009. URL: http://medcontent.metapress.com/index/A65RM03P4874243N.pdf{%}5Cnhttp://books.google.com/books?hl=en{&}lr={&}id=PphumLcKPi4C{&}oi=fnd{&}pg=PR26{&}dq=Model+Predictive+Control+System+Design+and+Implementation+Using+Matlab{&}ots=Aew69h39In{&}sig=7zpSD5E-riHQaX51CmJuO6sMdYI{%}5Cnhtt, doi:10.1007/978-1-84882-331-0.

[WK03] S. Wadoo and P. Kachroo. *Autonomous underwater vehicles*. 2003. URL: http://74.205.29.124/uploads/publication{_}pdf/AUVsWHOI2003.pdf, doi:10.1201/b10463.

[WYJ15] C. Wei, W. Yanhui, and Z. Jianhui. Back-stepping control of underactuated auv's depth based on nonlinear disturbance observer. In *2015 34th Chinese*

Control Conference (CCC), pages 6061–6065, July 2015. doi:10.1109/ChiCC.2015.7260587.

[XWW15] S. Xu, L. Wang, and X. Wang. Local optimization of thruster configuration based on a synthesized positioning capability criterion. *International Journal of Naval Architecture and Ocean Engineering*, 7(6):1044 – 1055, 2015. URL: http://www.sciencedirect.com/science/article/pii/S2092678216300152, doi:https://doi.org/10.1515/ijnaoe-2015-0073.

[YCS90] D. R. Yoerger, J. G. Cooke, and J. J. E. Slotine. The influence of thruster dynamics on underwater vehicle behavior and their incorporation into control system design. *IEEE Journal of Oceanic Engineering*, 15(3):167–178, Jul 1990. doi:10.1109/48.107145.

[YS85] D. Yoerger and J. Slotine. Robust trajectory control of underwater vehicles. *IEEE Journal of Oceanic Engineering*, 10(4):462–470, Oct 1985. doi:10.1109/JOE.1985.1145131.

[Yuh90] J. Yuh. Modeling and control of underwater robotic vehicles. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(6):1475–1483, Nov 1990. doi:10.1109/21.61218.

[Yuh00] J. Yuh. Design and control of autonomous underwater robots: A survey. *Autonomous Robots*, 8(1):7–24, Jan 2000. URL: https://doi.org/10.1023/A:1008984701078, doi:10.1023/A:1008984701078.

[ZSW99] K. Zhou, G. Salomon, and E. Wu. Balanced realization and model reduction for unstable systems. *International Journal of Robust and Nonlinear Control*, 9(3):183–198, 1999. URL: http://doi.wiley.com/10.1002/(SICI)1099-1239(199903)9:3{%}3C183::AID-RNC399{%}3E3.3.CO;2-5, doi:10.1002/(SICI)1099-1239(199903)9:3<183::AID-RNC399>3.3.CO;2-5.

License