2013

A ROBUST LINEAR DYNAMIC POSITIONING CONTROLLER FOR A MARINE SURFACE VEHICLE

Thomas P. DeRensis
University of Rhode Island, tderensis@gmail.com

A ROBUST LINEAR DYNAMIC POSITIONING CONTROLLER FOR A MARINE SURFACE VEHICLE

BY

THOMAS P. DERENSIS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

UNIVERSITY OF RHODE ISLAND

2013

MASTER OF SCIENCE THESIS

OF

THOMAS P. DERENSIS

APPROVED:

Thesis Committee:

Major Professor Richard Vaccaro

Peter Swaszek

Stephen Licht

Nasser H. Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2013

ABSTRACT

The autonomous surface vehicle (ASV) owned by the University of Rhode Island currently uses a PID controller that is insufficient for its maneuvering goals. The goal was to develop a robust linear controller based on state space dynamic positioning models that is able to follow a given path with minimal error in the presence of environmental disturbances and model perturbations. A controller was designed for the linear Cybership II model in MATLAB and then tested in Simulink with these disturbances on a nonlinear Cybership II model. The linear controller uses a feedfoward inverse filter to follow a desired elliptical path with high accuracy and uses new methods for choosing feedback gains to provide greater H_∞ robustness than MATLAB's `place` function. It was found that these new methods tracked the desired ellipse with lower error than existing pole placement methods. These methods show promise for real world implementation of linear controllers on URI's ASV and other surface vehicles.

ACKNOWLEDGMENTS

I would like to thank Dr. Richard Vaccaro, my major professor, for his guidance and support during my research. His input was always helpful and insightful when answering a multitude of control systems questions. This thesis would not be possible without him. I would like to thank Dr. Stephen Licht for pointing me in the right direction in the field of marine control. I wish to thank the ASV team for their advice and for motivating me to work on this problem.

I am greatly appreciative of the support from friends and families during this research. Especially for their patience and understanding during the writing of this thesis. Thanks to my long-time friend Tim Forbes for his persistent support and excellent memory of deadlines. I would also like to thank the digital signal processing group at the Naval Undersea Warfare Center for convincing me to pursue my Master's degree. Finally, I would like to thank the University of Rhode Island for supporting my graduate year through a Teaching Assistant position.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Overview

As linear control techniques have advanced, little research has been done on linear control of marine systems. This is due to the inherent nonlinear nature of marine systems that arises from rotation in the global reference frame, changes in hydrodynamic properties at different speeds, and nonlinear wave disturbances [1]. Great advancements have been made in nonlinear marine control and these techniques are still being researched. While these control methods are impressive, their implementations are often limited to academic settings due to their complexity. Therefore, a reassessment of linear state space controllers [2] with updated design techniques could widen the accessibility of marine control to those with knowledge of linear state space control.

The first types of control systems that were developed were heading controllers after the gyrocompass was invented. These heading autopilots use a constant speed, a compass, and heading controller with feedback to steer. Position controllers were made possible after the invention of local and global positioning systems (GPS). This allowed for more advanced maneuvers such as waypoint tracking, trajectory tracking, and path following. Which type of controller that is used depends on the application, but the more interesting control problem involves precisely controlling position and orientation in a global reference frame.

A common control system that is used frequently because of its simple implementation and lack of a need for a model is a proportional-integral-derivative (PID) controller. Gains for the controller can be adjusted until desired results are obtained. This non model-based approach is not very scientific and could take a while to tune gains. There exist methods [3] for choosing gains if various parame-

ters are known, which can sometimes be difficult to determine. However, it is still useful when a quick controller is needed under relatively small disturbances and no previous model to build on. PID heading controllers can also be used with a simple heading model that uses a single rudder input [4] [5]. This approach needs only a few model parameters and there are well established methods for designing this controller [6].

Heading controllers fall short when doing precise waypoint navigation or path following. This is where position controllers become useful, allowing for much more complex maneuvers. This project will focus on these position controllers. Due to the hydrodynamics of marine vessels, many different state space models of varying complexity exist for use with position controllers [7]. Models that assume low operating speed or position holding are called dynamic positioning (DP) models [8], while higher speeds require more complex maneuvering models. This project's focus will be on DP models because the target vehicle will need to move at low speeds in order to do other processing.

The linear DP model has been used successfully for marine control, but it has limited application because of a few assumptions. The DP model assumes low speed and low yaw rate in order for the state space model to be accurate. It is also important for the vehicle to be fully-actuated or over-actuated to generate forces in each direction. For example, in three degrees of freedom (x, y, heading), there needs to be at least three inputs that can apply forces in each direction. In state space control, the gain selection has a unique solution for single-input, single-output (SISO) systems using pole placement. The DP model in our case has three inputs for the forces in x and y directions, as well as about the z-axis. The model also has 3 outputs for the position, making the model multiple-input, multiple-output (MIMO). In this case there are an infinite number of gain matrices, all of

which yield the desired closed-loop pole locations. There are new techniques for selecting the gain matrix for maximum robustness and other techniques that improve tracking system accuracy. These new techniques have not been investigated in marine control literature [9].

1.2 Motivation

The ocean engineering department at the University of Rhode Island has an autonomous surface vehicle (ASV) that currently uses a primitive proportional-integral-derivative (PID) controller. The use of this controller limits the maneuvering capabilities [10] of the vehicle and negatively impacts the vehicle's performance in the annual RoboBoat competition. It has been decide by the ASV team that an improved control system is needed to stay competitive in the annual RoboBoat competition.

1.3 Objective and Scope

The objective of this thesis is to develop a robust state space linear dynamic position controller that will improve the maneuverability of URI's ASV. Robustness is important because the linear controller will have to cope with the inaccuracies of the linear model and disturbances. New linear state space methods will be used that have not previously tested on marine vehicles. This includes the development of an inverse filter that will allow the tracking system to follow the desired input with increased accuracy and a method for choosing robust feedback gains based on pole placement [9]. These results will be compared to those commonly used in linear control design, such as MATLAB's `place` function.

Due to lack of modeling parameters for URI's ASV, testing will be done on a model of the Cybership II in Simulink [11]. The methods used in this thesis can be generalized to function with any fully-actuated marine surface vehicle. Simulation

will be made realistic by including a nonlinear boat model, external disturbances, and measurement noise. The controller will be tested by following an elliptical path similar to [1]. Results will be analyzed based on how well the position and heading follow the desired track for each method under disturbances and model perturbations.

List of References

[1] R. Skjetne, Ø. N. Smogeli, and T. I. Fossen, "A Nonlinear Ship Manoeuvering Model: Identification and adaptive control with experiments for a model ship," *Modeling, Identification and Control*, vol. 25, no. 1, pp. 3–27, 2004.

[2] R. J. Vaccaro, *Digital Control: A State-Space Approach*. McGraw-Hill, Inc., 1995.

[3] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Ltd., 2011.

[4] F. Azarsina and C. D. Williams, "Nomoto Indices for Constant-Depth Zigzag Manoeuvres of an Autonomous Underwater Vehicle," *ISRN Oceanography*, vol. 2013, Article ID 219545, doi:10.5402/2013/219545, 2013.

[5] C.-Y. Tzang and J.-F. Chen, "Fundamental Properties of Linear Ship Steering Dynamic Models," *Journal of Marine Science and Technology*, vol. 7, no. 2, pp. 79–88, 1999.

[6] J. Journée, "A Simple Method for Determining the Manoeuvring Indices K and T from Zigzag Trial Data," Delft University of Technology, Ship Hydromechanics Laboratory, Report 267, 1970.

[7] R. Skjetne, "The Maneuvering Problem," Phd Dissertation, Norwegian University of Science and Technology, 2005.

[8] A. J. Sørensen, S. I. Sagatun, and T. I. Fossen, "Design of a Dynamic Positioning System Using Model-Based Control," *Control Engineering Practice*, vol. 4, no. 3, pp. 359–368, 1996.

[9] R. J. Vaccaro, "Personal communication," 2012.

[10] M. Committee of 23rd ITTC, *ITTC - Recommended Procedures*, 2002.

[11] Norwegian University of Science and Technology. "MSS. Marine Systems Simulator." 2010. [Online]. Available: http://www.marinecontrol.org

CHAPTER 2

Background and Literature Review

2.1 State Space Representation

In state space control, the time-domain system is modeled as a set of input, output, and state variables that are related by first-order differential equations. To use varying numbers of inputs, outputs, and states, it is useful to represent these as vectors. A linear time-invariant system can use matrices to express the differential and algebraic equations. This compact form (1) will be used throughout this thesis for continuous systems.

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \end{aligned} \tag{1}$$

The vectors are $\boldsymbol{u}$ for input, $\boldsymbol{y}$ for output, and $\boldsymbol{x}$ for the state, where there are p inputs, q outputs, and n state variables. The $\boldsymbol{A}$ matrix is the $n \times n$ state matrix, $\boldsymbol{B}$ is the $n \times p$ input matrix, $\boldsymbol{C}$ is the $q \times n$ output matrix, and $\boldsymbol{D}$ is the $q \times p$ feedthrough matrix. Lastly, $\dot{\boldsymbol{x}}(t)$ is the time derivative of the state, $\boldsymbol{x}(t)$. All vectors and matrices in this thesis will be bold.

When implementing a state space controller, it is common to convert a model to discrete time based on a sampling time T. Then the controller can be updated at a fixed interval by a digital computer.

$$\begin{aligned} \boldsymbol{x}[k+1] &= \boldsymbol{\Phi}\boldsymbol{x}[k] + \boldsymbol{\Gamma}\boldsymbol{u}[k] \\ \boldsymbol{y}[k] &= \boldsymbol{C}\boldsymbol{x}[k] + \boldsymbol{D}\boldsymbol{u}[k] \end{aligned} \tag{2}$$

The continuous state matrix $\boldsymbol{A}$ and input matrix $\boldsymbol{B}$ are transformed into the discrete matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Gamma}$, while $\boldsymbol{C}$ and $\boldsymbol{D}$ remain the same for the discrete form. The discrete state space equation (2) is obtained by calculating the zero-order hold

for the continuous system, which can be done using (3).

$$\begin{aligned} \boldsymbol{\Phi} &= e^{\boldsymbol{A}T} \\ \boldsymbol{\Gamma} &= \left(\int_0^T e^{\boldsymbol{A}\tau} \, d\tau \right) \boldsymbol{B} \end{aligned} \tag{3}$$

2.2 State Space Control

Before beginning control design on a system, it is important to test the system to see if it is controllable. This will determine if control techniques can be used to stabilize the system. A common way to test this is to form the controllability matrix and check its rank.

$$\boldsymbol{W}_c = \begin{bmatrix} \boldsymbol{\Gamma} & \boldsymbol{\Phi}\boldsymbol{\Gamma} & \boldsymbol{\Phi}^2\boldsymbol{\Gamma} & \dots & \boldsymbol{\Phi}^{n-1}\boldsymbol{\Gamma} \end{bmatrix} \tag{4}$$

If $\boldsymbol{W}_c$ has full rank, the system is controllable. Note that in equation 4 the discrete system was tested, however, the controllability of the continuous system could also be checked. Similarly, observability can be be checked by forming the observability matrix in (5). If $\boldsymbol{W}_o$ has full rank, then the system is observable.

$$\boldsymbol{W}_o = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C}\boldsymbol{\Phi} \\ \boldsymbol{C}\boldsymbol{\Phi}^2 \\ \vdots \\ \boldsymbol{C}\boldsymbol{\Phi}^{n-1} \end{bmatrix} \tag{5}$$

2.2.1 Stability

A stable system is one that will not give unbounded output when given a bounded input. It is possible for this system to oscillate forever, but it will be remain bounded between some values. This is also known as bounded-input, bounded-output (BIBO) stability. A stronger form of stability is asymptotic stability. This is when a system's state, $\boldsymbol{x}(t)$, approaches an equilibrium point $\boldsymbol{x}_e$ (usually $\mathbf{0}$) as $t \to \infty$.

A simple way to determine the stability of a linear system is to look at the eigenvalues, or poles, of the $\boldsymbol{A}$ matrix. For continuous systems, the system is stable if the eigenvalues of $\boldsymbol{A}$ do not have positive real parts. The system is asymptotically stable if the eigenvalues of $\boldsymbol{A}$ have strictly negative real parts. For discrete systems, the eigenvalues of $\boldsymbol{\Phi}$ (discrete form of $\boldsymbol{A}$) must be inside or on the unit circle for stability. Asymptotic stability requires the eigenvalues to be inside the unit circle. Note that stability has nothing to do with robustness to modeling errors (see section 2.2.5) or how quickly the system reaches equilibrium.

2.2.2 Poles and Settling Time

The settling time of a system is how quickly the system reaches its equilibrium to within some percentage, usually 1% or 2%, and is determined by the location of the poles. Assuming a continuous system is stable, the more negative the real part of the poles are, the faster the response due to that pole will settle. For example, if a system has two poles, one at -12 and another at -3, the response of the pole at -3 will settle more slowly, making it the dominant pole and determining the system's overall settling time. Settling time for the pole at -3 can be estimated by using equation 6 for a tolerance of 1%.

$$T_s \approx \frac{\ln(\text{tolerance})}{\Re(\text{slowest pole})} = \frac{\ln(0.01)}{-3} \approx 1.5sec \tag{6}$$

Special types of poles that are known to have nice properties [1] are called Bessel poles. They are the roots of the Bessel polynomials and can be generated using the formula in equation 7.

$$Bessel_n(x) = \sum_{k=0}^{n} \frac{(2n-k)!}{(n-k)!k!} \frac{x^k}{2^{n-k}} \tag{7}$$

The roots of each Bessel polynomial have different settling times. Normalizing these roots to one second settling time allows the roots be scaled to the desired

settling time.

$$s_n = \frac{roots(Bessel_n(x))}{T_s} \tag{8}$$

The normalized Bessel poles for polynomials $n = 1 \ldots 6$ are given in Table 1. These Bessel poles will be used when no other suitable poles can be chosen for controller design.

s_1	-4.6200		
s_2	-4.0530 $\pm$ j2.3400		
s_3	-5.0093	-3.9668 $\pm$ j3.7845	
s_4	-4.0156 $\pm$ j5.0723	-5.5281 $\pm$ j1.6553	
s_5	-6.4480	-4.1104 $\pm$ j6.3142	-5.9268 $\pm$ j3.0813
s_6	-4.2169 $\pm$ j7.5300	-6.2613 $\pm$ j4.4018	-7.1205 $\pm$ j1.4540

Table 1: The roots of normalized Bessel polynomials with $T_s = 1$ from [1]

2.2.3 Control System Structures

For the simplest case, a system can have no controller applied to it. This is called the open-loop system because it has no feedback for control. This system could be asymptotically stable, but we have no control over how quickly the system will settle. If the open-loop system is not stable, does not settle in the desired amount of time, or does not have desired robustness, then it is useful to introduce a feedback gain $\boldsymbol{L}$ to get desired pole locations, and therefore settling time. This is called a state-feedback regulator. A regulator will drive the system to an equilibrium point. This state-feedback regulator assumes that all states can be measured directly; i.e., $\boldsymbol{C} = \boldsymbol{I}$, and $\boldsymbol{D} = \boldsymbol{0}$. To design the regulator, we choose

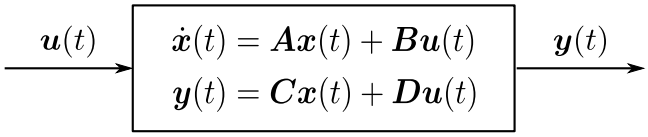


Figure 1: An open-loop state space system.

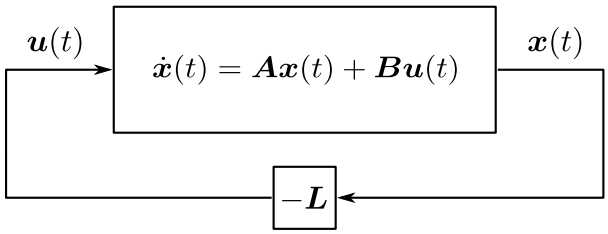


Figure 2: A state-feedback regulator

desired poles for the closed-loop system and calculate the gain matrix $\boldsymbol{L}$ to obtain a system that has those poles.

In many cases all of the states cannot be measured. To deal with this an observer needs to be designed that will use the input to the plant and its output to estimate the state. In addition to the state feedback gains, $\boldsymbol{L}$, the observer gains, $\boldsymbol{K}$, need to be calculated so that the observer settles quicker than the plant. This means that the plant states need to be estimated correctly before the system can be regulated to equilibrium. The architectures discussed so far have only been

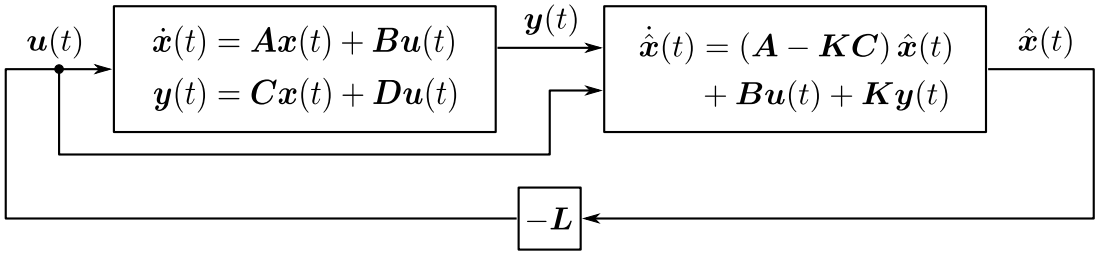


Figure 3: A state-feedback regulator with full order observer

designed to regulate to a fixed equilibrium point. A more useful control system would be one that could be take a desired output for the plant and drive the output of the system to that reference input. This is called a tracking system. In a tracking system the output of the plant is subtracted from the desired output to obtain an error term. This error term is fed into what is called the additional dynamics block [1] to generate the needed input to the plant. The $\boldsymbol{L}$ gain for the entire system is now split into two parts, where the gain matrix $\boldsymbol{L}_1$ is used to

regulate the plant and has columns equal to the number of plant states, and $\boldsymbol{L}_2$ is used for the additional dynamics and has columns equal to the number of plant outputs.

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{L}_2 \\ {\scriptstyle p \times n} & {\scriptstyle p \times q} \end{bmatrix} \tag{9}$$

The additional dynamics $\boldsymbol{A}_a$ and $\boldsymbol{B}_a$ implement an integrator similar to the I term in PID controllers and can also reject disturbances if designed to do so.

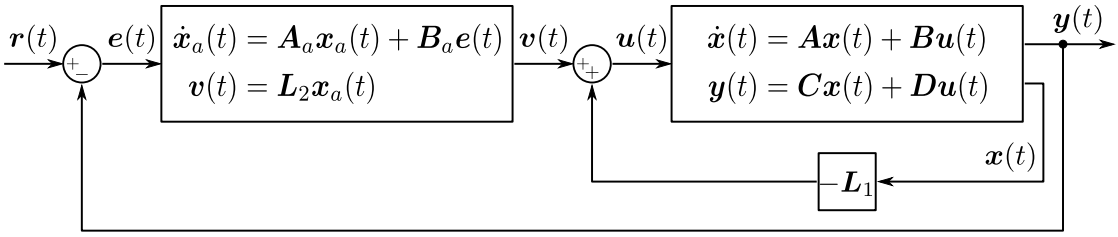


Figure 4: A tracking system

The next logical step would to combine the tracking system and observer to form an even more useful linear controller. A tracking system with an observer can follow any reference input with minimal measured states thanks to the work of the observer. Due to the separation principal in regards to control and estimation, the tracking system and observer can be designed separately. The closed-loop poles of the overall system are the tracking system poles $\boldsymbol{L}$ combined with the observer poles $\boldsymbol{K}$. However, to obtain an observer-based tracking system with acceptable stability margins, the tracking system gains must be taken into account when calculating the observer gains.

2.2.4 Norms

A norm of a vector is the size or length of a vector in a vector space. The norm of a vector $\boldsymbol{x}$ is represented as $\|\boldsymbol{x}\|$ and has the following properties.

1. $\|\boldsymbol{x}\| = 0$ iff $\boldsymbol{x} = \boldsymbol{0}$

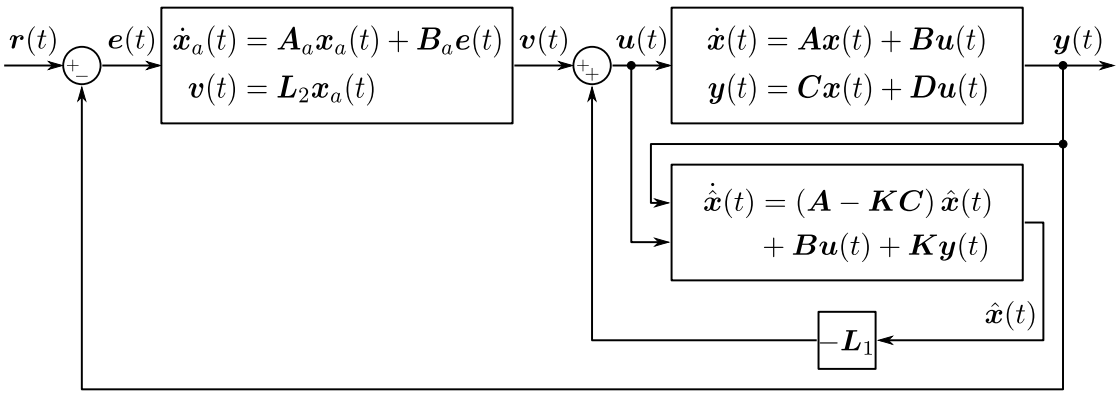


Figure 5: An observer-based tracking system

2. $\|\alpha\boldsymbol{x}\| = \alpha\|\boldsymbol{x}\|,\ \forall\,\alpha > 0$
3. $\|\boldsymbol{x}_1 + \boldsymbol{x}_2\| \leq \|\boldsymbol{x}_1\| + \|\boldsymbol{x}_2\|$

There are a few different types of norms. The L_2 norm is the most intuitive type of norm and is the distance from the origin to the vector $\boldsymbol{x}$ that arises from the Pythagorean theorem. The L_p norm is a generalized norm that is equal to the L_2 norm for $p = 2$ and is defined for $p \geq 1$. Lastly, there is the L_∞ norm, or maximum norm. For a vector $\boldsymbol{u}(t) = [u_1(t)...u_n(t)]^\intercal$ the signal norms are as defined in equations 10-12.

$$L_2\ norm:\ \|\boldsymbol{u}(t)\|_{L_2} = \sqrt{\int_0^\infty \boldsymbol{u}^\intercal(t)\boldsymbol{u}(t)\,\mathrm{d}t} \tag{10}$$

$$L_p\ norm: \|\boldsymbol{u}(t)\|_{L_p} = \left(\int_0^\infty \|\boldsymbol{u}(t)\|^p\,\mathrm{d}t\right)^{\frac{1}{p}} \tag{11}$$

$$L_\infty\ norm: \|\boldsymbol{u}(t)\|_{L_\infty} = \sup_t\left\{\max_i |u_i(t)|\right\} \tag{12}$$

2.2.5 Input-Output Stability and Robustness

A system H with input $\boldsymbol{u}(t)$ and output $\boldsymbol{y}(t)$ is said to be finite-gain L-stable if there exits a gain called $\delta < \infty$ and a constant positive bias β such that equation 13 is satisfied. The T subscript stands for the truncation operator, which sets the

continuous function $\boldsymbol{u}_T(t) = 0$ for $t > T$.

$$\|\boldsymbol{y}_T(t)\| \leq \delta \|\boldsymbol{u}_T(t)\| + \beta \tag{13}$$

The gain δ is an upper bound and can be calculated for the case of L_2 signal norms. This result is called the system infinity norm of the system. If the system is described by the transfer function matrix $G(j\omega)$, then the system infinity norm is

$$\delta = G_\infty \text{ norm} = \sup_\omega \{\sigma_1 (G(j\omega))\} \tag{14}$$

where σ_1 is the maximum singular value of a matrix. In other words, the G_∞ norm is the maximum singular value over all frequencies of the transfer function matrix $G(j\omega)$. For SISO systems, this is the maximum value of the Bode magnitude plot. For a state space model the transfer function matrix is shown in equation 15.

$$G(j\omega) = \boldsymbol{C}(j\omega \boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D} \tag{15}$$

The H_∞ norm, when combined with the Small Gain Theorem (SGT) [2], allows us to define a measure of robustness for a control system. The SGT states that for two systems H_1 and H_2 connected as shown in figure 6, if $\delta_1\delta_2 < 1$ then the system from $\boldsymbol{u}(t) = [\boldsymbol{u}_1(t)^\intercal, \boldsymbol{u}_2(t)^\intercal]^\intercal$ to $\boldsymbol{y}(t) = [\boldsymbol{y}_1(t)^\intercal, \boldsymbol{y}_2(t)^\intercal]^\intercal$ is finite gain stable.

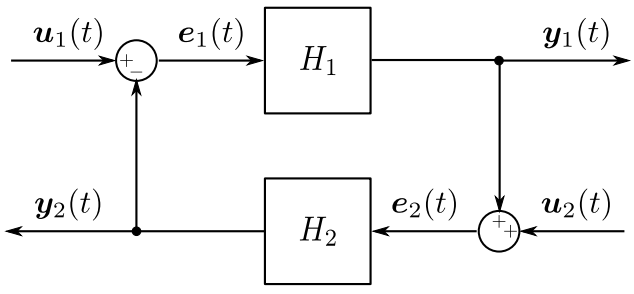


Figure 6: The Small Gain Theorem: Feedback interconnection of H_1 and H_2

For this thesis, only the special case when $\boldsymbol{u}_2 = 0$ is needed. The upper bound for δ_1 is maximized when δ_2 is minimum because the product of norms must be

less than 1. Therefore, it can be concluded from the SGT that minimizing the H_∞ norm of H_2 will result in the largest allowable H_∞ norm of H_1. If H_1 is an unknown perturbation of the plant model with norm of δ, then minimizing δ_2 results in the greatest stability robustness. This guarantees that the closed-loop system will be stable if

$$\delta < \delta_{max} = \frac{1}{\delta_2} \tag{16}$$

2.3 Marine Models

For marine vehicles, many models exist of varying complexity [3]. Marine systems are inherently nonlinear because of hydrodynamic forces being a function of velocity, the Coriolis effect at different velocities, rotation from body coordinates to global coordinates, and buoyancy forces. The most complex 6 degree of freedom (DOF) model for marine motion in body coordinates is expressed in equation 17.

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu}$$

$$\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{M}_A\dot{\boldsymbol{\nu}}_r + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{g}(\boldsymbol{\eta}) + \boldsymbol{g}_0 = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{disturb}} \tag{17}$$

All other models are justified simplifications of this motion model. It is rare that this model is used directly in control design because of its complexity. Instead, it is used for simulating the full dynamics of the system. The notation for marine vehicles and different model options will be discussed in the following sections.

2.3.1 Notation and Reference Frames

For a full 6 DOF system, the notation for positions, velocities, and forces that are defined by the Society of Naval Architects and Marine Engineers (SNAME) are presented in Table 2. For surface vehicles such as the CyberShipII and URI's ASV, the majority of motion occurs in the first, second, and sixth DOF. For this reason, surface vehicles are said to have 3 DOF and most controllers for surface vehicles

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

Table 2: The SNAME notation for marine vehicles

are designed with this in mind. In some cases roll and pitch are important motions in control, but this thesis does not focus on those situations. It is convenient to split the state variables into vectors for positions, velocities, and forces as shown for 6 DOF in equation 18.

$$\boldsymbol{\eta} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix}, \boldsymbol{\nu} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix}, \boldsymbol{\tau} = \begin{bmatrix} X \\ Y \\ Z \\ K \\ M \\ N \end{bmatrix} \tag{18}$$

The vehicle's position and orientation is represented by $\boldsymbol{\eta}$, the vehicle's velocity in each direction is $\boldsymbol{\nu}$, and the forces acting on the vehicle are represented by $\boldsymbol{\tau}$. These vectors allow the same equations to be used for different degrees of freedom. For example, 3 DOF only includes states representing positions, velocities, and forces in the xy-plane. The vector equations would be the same but the vectors would only include the first, second and sixth rows (19). From now on all vector representations will be in 3 DOF because this thesis is only investigating the control of surface vehicles.

$$\boldsymbol{\eta} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}, \boldsymbol{\nu} = \begin{bmatrix} u \\ v \\ r \end{bmatrix}, \boldsymbol{\tau} = \begin{bmatrix} X \\ Y \\ N \end{bmatrix} \tag{19}$$

Before jumping into all of the possible marine models, reference frames must be discussed. The two reference frames that are important for control on a small scale

are the North-East-Down (NED) and body-fixed (BODY) reference frames.

The NED reference frame $\{n\} = \{x_n, y_n, z_n\}$ has origin o_n and is defined as a tangent plane on the Earth's surface. The x_n-axis points towards true North, the y_n-axis points towards East, and the z_n-axis points into the Earth. This causes the heading to be defined as positive clockwise about the origin and 0 degrees at true North. This reference frame is used for marine vehicles that stay at approximately the same latitude and longitude. In this thesis the NED reference frame will used for global positions and all commanded positions.

The body-fixed reference frame $\{b\} = \{x_b, y_b, z_b\}$ has origin o_b which is fixed to a point on the vehicle close to the center of gravity and rotates with the marine vehicle. The x_b-axis is directed from aft to fore, the y_b-axis is directed from port to starboard, and the z_b axis is directed from top to bottom. Velocities $\boldsymbol{\nu}$ and forces $\boldsymbol{\tau}$ on the vehicle are expressed these BODY coordinates, while the position $\boldsymbol{\eta}$ of the vehicle must be expressed in the global NED reference frame.

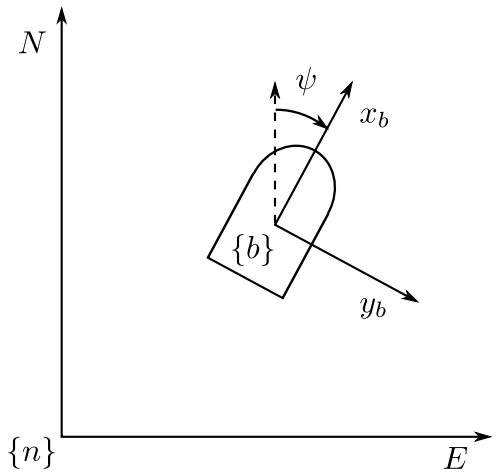


Figure 7: NED and Body reference frames

2.3.2 Rigid-Body Kinetics

The first part of equation 17 that will be discussed is the motions of rigid bodies derived from Newtonian physics [3]. These kinetics are represented by

$$\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau}_{RB} \tag{20}$$

The $\boldsymbol{M}_{RB}$ matrix is the rigid-body mass matrix and the $\boldsymbol{C}_{RB}(\boldsymbol{\nu})$ matrix is the velocity dependent Coriolis and centripetal matrix due to rotation of the BODY reference frame about the global NED reference frame. The velocity vector $\boldsymbol{\nu} = [u, v, r]^\intercal$ and force vector $\boldsymbol{\tau} = [X, Y, N]^\intercal$ are in reference to the BODY frame. The rigid-body mass matrix is unique and defined by equation 21.

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m & 0 & -my_g \\ 0 & m & mx_g \\ -my_g & mx_g & I_z \end{bmatrix} \tag{21}$$

In $\boldsymbol{M}_{RB}$, m is the mass of the vehicle, x_g and y_g are the distances to the center of gravity (CG) from body origin o_b, and I_z is the moment of inertia about the z_b-axis. In practice the CG is on the y_b-axis, making $y_g = 0$. These variables are among the easiest to obtain because they can be measured directly. Methods for estimating I_z can be found in [4].

Unlike the rigid-body mass matrix, the rigid-body Coriolis and centripetal matrix is not unique. In this paper the equation derived by Fossen in [3] will be used.

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & -m(x_g r + v) \\ 0 & 0 & -m(y_g r - u) \\ m(x_g r + v) & m(y_g r - u) & 0 \end{bmatrix} \tag{22}$$

The $\boldsymbol{C}_{RB}(\boldsymbol{\nu})$ matrix in equation 22 is nonlinear and depends on the velocity of the vehicle.

2.3.3 Hydrodynamic Forces

The hydrodynamic effects result from the added mass and Coriolis-centripetal forces due to the rotation of the vehicle relative to the NED frame as well as

viscous damping. Motion of a marine vehicle will induce some motion in the surrounding water, displacing it and then settling back after the vehicle has passed through. This gives the fluid kinetic energy, causing the added mass and Coriolis-centripetal terms. Linear damping comes from potential damping and skin friction and nonlinear damping is caused by quadratic damping and higher order terms.

$$-\boldsymbol{M}_A\dot{\boldsymbol{\nu}}_{\boldsymbol{r}} - \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r - \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r = \boldsymbol{\tau}_{\text{hyd}} \tag{23}$$

$$\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{\nu}_c \tag{24}$$

The hydrodynamic forces are combined in equation 23 where $\boldsymbol{\nu}_r$ is the relative velocity and $\boldsymbol{v}_c = [u_c, v_c, 0]^\intercal$ is an irrotational ocean current. The full added mass matrix is expressed in equation 25.

$$\boldsymbol{M}_A = -\begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{r}} \end{bmatrix} \tag{25}$$

The hydrodynamic derivatives in $\boldsymbol{M}_A$ are consistent with SNAME notation. For example, the hydrodynamic added mass force X along the x-axis due to sway acceleration $\dot{v}$ is represented by $X_{\dot{v}}$. The added mass matrix is positive semi-definite for a rigid body at rest or moving at positive forward speed. Methods for identifying these coefficients are presented in [5].

For the added Coriolis-centripetal matrix $\boldsymbol{C}_A(\boldsymbol{\nu})$, the full form is expressed in equation 26.

$$\boldsymbol{C}_A(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & Y_{\dot{u}}u + Y_{\dot{v}}v + Y_{\dot{r}}r \\ 0 & 0 & -X_{\dot{u}}u - X_{\dot{v}}v - X_{\dot{r}}r \\ -Y_{\dot{u}}u - Y_{\dot{v}}v - Y_{\dot{r}}r & X_{\dot{u}}u + X_{\dot{v}}v + X_{\dot{r}}r & 0 \end{bmatrix} \tag{26}$$

For most surface vehicles the surge and steering dynamics can be decoupled due to xz-plane symmetry. This reduces the added mass matrix to four unknowns shown in equation 27. Note that due to symmetry $N_{\dot{v}} = Y_{\dot{r}}$.

$$\boldsymbol{M}_A = -\begin{bmatrix} X_{\dot{u}} & 0 & 0 \\ 0 & Y_{\dot{v}} & Y_{\dot{r}} \\ 0 & Y_{\dot{r}} & N_{\dot{r}} \end{bmatrix} \tag{27}$$

This assumption also reduces the added Coriolis-centripetal matrix to equation 28.

$$\boldsymbol{C}_A(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & Y_{\dot{v}}v + Y_{\dot{r}}r \\ 0 & 0 & -X_{\dot{u}}u \\ -Y_{\dot{v}}v - Y_{\dot{r}}r & X_{\dot{u}}u & 0 \end{bmatrix} \tag{28}$$

The total damping on a marine vehicle can be split into linear damping and nonlinear quadratic damping. It is difficult to separate these effects in practice, but it is convenient for analysis and control.

$$\boldsymbol{D}(\boldsymbol{\nu}_r) = \boldsymbol{D} + \boldsymbol{D}_n(\boldsymbol{\nu}_r) \tag{29}$$

The linear damping matrix $\boldsymbol{D}$ can be written in full using SNAME notation similar to $\boldsymbol{M}_A$. The difference is that the force is caused by velocity instead of acceleration.

$$\boldsymbol{D} = -\begin{bmatrix} X_u & X_v & X_r \\ Y_u & Y_v & Y_r \\ N_u & N_v & N_r \end{bmatrix} \tag{30}$$

Under the same assumption of xz-plane symmetry the damping matrix reduces to equation 31.

$$\boldsymbol{D} = -\begin{bmatrix} X_u & 0 & 0 \\ 0 & Y_v & Y_r \\ 0 & N_v & N_r \end{bmatrix} \tag{31}$$

Nonlinear damping assuming xz-plane symmetry is shown in equation 32.

$$\boldsymbol{D}_n(\boldsymbol{\nu}) = -\begin{bmatrix} X_{|u|u}|u| & 0 & 0 \\ 0 & Y_{|v|v}|v| + Y_{|r|v}|r| & Y_{|v|r}|v| + Y_{|r|r}|r| \\ 0 & N_{|v|v}|v| + N_{|r|v}|r| & N_{|v|r}|v| + N_{|r|r}|r| \end{bmatrix} \tag{32}$$

All of the hydrodynamic derivatives in this section are dependent on the path of the vehicle and vehicle velocities. If these values are determined during a specific maneuver under certain conditions, they may not be valid for other speeds or paths. Therefore it is important to design a robust controller that can handle the natural uncertainty and nonlinearity of these marine systems.

2.3.4 Dynamic Positioning Models

Dynamic Positioning (DP) models assume zero or low speed. The models presented in this section will be valid for low speed maneuvering up to 2 m/s [3].

The DP model can be derived from the advanced model shown in equation 17. The $\boldsymbol{g}_0$ and $\boldsymbol{g}(\boldsymbol{\eta})$ terms that represent buoyancy and restoring forces can be dropped because they are not relevant for 3 DOF models. In addition, the rigid-body mass and added mass can be combined into one $\boldsymbol{M}$ matrix (33).

$$\boldsymbol{M}_{RB} + \boldsymbol{M}_A = \boldsymbol{M} \tag{33}$$

The rotation matrix can be simplified from the general rotation matrix $\boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})$ to the rotation matrix $\boldsymbol{R}(\psi)$ about the z-axis due to the reduction to 3 DOF.

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{34}$$

This gives the nonlinear result in equation 35 which is suitable for simulation, but difficult to work with for control design.

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{R}(\psi)\boldsymbol{\nu} \\ \boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r &= \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{disturb}} \end{aligned} \tag{35}$$

The nonlinear DP equation can be linearized by making some assumptions. It is known that at zero speed, nonlinear damping can be ignored and control design can use linear damping D [3]. The relative velocity $\boldsymbol{\nu}_r$ can be ignored if ocean currents are treated as a slowly varying bias vector $\boldsymbol{b}$ which can be compensated by the integral action of the controller. To remove the rotation matrix the position $\boldsymbol{\eta}$ is typically replaced by vessel parallel coordinates $\boldsymbol{\eta}_p$.

$$\boldsymbol{\eta}_p = \boldsymbol{R}^{\intercal}(\psi_d)\boldsymbol{\eta} \tag{36}$$

$$\dot{\boldsymbol{\eta}}_p = r \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{\eta}_p + \boldsymbol{\nu} \approx \boldsymbol{\nu} \tag{37}$$

The vessel parallel coordinate system [6] is used for designing linear controllers. It defines a fixed coordinate system that is translated to the desired position (x_d, y_d)

and rotated to the desired heading ψ_d. The vessel parallel reference frame is temporary and changes based on the desired position input to the controller. Notice that for low yaw rates $r \approx 0$ the rate of change of the vessel parallel position is approximately equal to velocity.

This results in the following linear DP model which will be used for the design of a state space controller.

$$\begin{aligned} \dot{\boldsymbol{\eta}}_p &= \boldsymbol{\nu} \\ \boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} &= \boldsymbol{R}^{\intercal}(\psi)\boldsymbol{b} + \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{disturb}} \end{aligned} \tag{38}$$

2.3.5 Thrusters

A simple thruster model assumes there is a way to generate a force in each direction, also known as fully actuated. It takes into account the limitations of propellers by modeling the actuators using time constants for each direction in the diagonal matrix $\boldsymbol{A}_{thr}$.

$$\boldsymbol{A}_{thr} = -\begin{bmatrix} \frac{1}{T_{surge}} & 0 & 0 \\ 0 & \frac{1}{T_{sway}} & 0 \\ 0 & 0 & \frac{1}{T_{yaw}} \end{bmatrix} \tag{39}$$

$$\dot{\boldsymbol{\tau}} = \boldsymbol{A}_{thr}(\boldsymbol{\tau} - \boldsymbol{\tau}_{com}) \tag{40}$$

The time constants represent how quickly the thrusters reach their commanded input $\boldsymbol{\tau}_{com}$. The higher the time constant, the slower the propeller is.

To get the generalized forces in each direction the vehicle must be fully actuated and have a thruster configuration matrix $\boldsymbol{T}_c$. This process is called control allocation. The configuration matrix has a column for each actuator that splits the force into components of surge, sway, and yaw. Some examples are below.

$$\boldsymbol{t}_{main\_prop} = \begin{bmatrix} 1 \\ 0 \\ -l_{y_i} \end{bmatrix}, \quad \boldsymbol{t}_{azmuth\_thruster} = \begin{bmatrix} \cos(\alpha_i) \\ \sin(\alpha_i) \\ l_{x_i}\sin(\alpha_i) - l_{y_i}\cos(\alpha_i) \end{bmatrix} \tag{41}$$

The distance from o_b in the x and y direction for the i-th actuator is represented by l_{x_i} and l_{y_i}, respectively, and the angle of the i-th propeller is represented by

α_i. Each thruster is combined into $\boldsymbol{T}_c$ and multiplied by a force coefficient matrix $\boldsymbol{K}_f = \text{diag}(K_1, \ldots, K_n)$ and the control input $\boldsymbol{u}$ to get the desired thrust.

$$\boldsymbol{\tau} = \boldsymbol{T}_c(\boldsymbol{\alpha})\boldsymbol{K}_f\boldsymbol{u} \tag{42}$$

2.4 Current Marine Control Techniques

Industrial control systems tend to use linear control methods because of their simplicity and ease of implementation, while the majority of academic marine control techniques are nonlinear. A few reasons for this are discussed by Strand [7]. The inherent properties of the mechanical system are nonlinear and certain system properties can be exploited in nonlinear control design. Linearization is always an approximation, hence it is easier to design a robust controller with nonlinear design. Strand also argues that nonlinear control is simpler for marine systems because it requires no linearization or gain scheduling methods. Now that that has been said, this thesis will attempt to show that robust linear techniques will be sufficient for marine systems under certain operating conditions.

2.4.1 Nonlinear

A simple technique used in the MSS toolbox [8] is a nonlinear DP PID controller with a passive wave filter to prevent wear on thrusters. Adaptive control [9] has been implemented for the CyberShip II for a DP controller under wave disturbances. Fossen [3] presents many examples of DP control, including PID with acceleration feedback, linear quadradic optimal control, integrator backstepping , and sliding-mode control.

2.4.2 Linear

For reduced models, such as the linear Nomoto model [10], there has been significant research in the system identification of the Nomoto Indicies [11, 12]. These models are often good enough for quick implementations of heading autopi-

lots under constant forward velocity. This model is not sophisticated enough to achieve the maneuvering goals of URI's ASV and this thesis.

Linear DP controllers have been proposed using vessel parallel coordinates by Fossen [3] which will be investigated in this thesis.

List of References

[1] R. J. Vaccaro, *Digital Control: A State-Space Approach.* McGraw-Hill, Inc., 1995.

[2] H. J. Marquez, *Nonlinear Control Systems.* John Wiley & Sons, Inc., 2003.

[3] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control.* John Wiley & Sons, Ltd., 2011.

[4] R. Aasen and B. Hays, "Method for Finding Min and Max Values of Error Range for Calculation of Moment of Inertia," in *69th Annual Conference Of Society of Allied Weight Engineers, Inc*, no. 3504, May 2010.

[5] H. K. Yoon and K. P. Rhee, "Identification of hydrodynamic coefficients in ship maneuvering equations of motion by estimation-before-modeling technique," *Ocean Engineering*, vol. 30, p. 23792404, 2003.

[6] J. P. Strand, K. Ezal, T. I. Fossen, and P. V. Kokotovi¸, "Nonlinear Control of Ships: A Locally Optimal Design," in *4th IFAC nonlinear control systems design symposium*, 1998, pp. 732–737.

[7] J. P. Strand, "Nonlinear Position Control Systems Design for Marine Vessels," Ph.D. dissertation, Norwegian University of Science and Technology, 1999.

[8] Norwegian University of Science and Technology. "MSS. Marine Systems Simulator." 2010. [Online]. Available: http://www.marinecontrol.org

[9] R. Skjetne, Ø. N. Smogeli, and T. I. Fossen, "A Nonlinear Ship Manoeuvering Model: Identification and adaptive control with experiments for a model ship," *Modeling, Identification and Control*, vol. 25, no. 1, pp. 3–27, 2004.

[10] C.-Y. Tzang and J.-F. Chen, "Fundamental Properties of Linear Ship Steering Dynamic Models," *Journal of Marine Science and Technology*, vol. 7, no. 2, pp. 79–88, 1999.

[11] J. Journée, "A Simple Method for Determining the Manoeuvring Indices K and T from Zigzag Trial Data," Delft University of Technology, Ship Hydromechanics Laboratory, Report 267, 1970.

[12] F. Azarsina and C. D. Williams, "Nomoto Indices for Constant-Depth Zigzag Manoeuvres of an Autonomous Underwater Vehicle," *ISRN Oceanography*, vol. 2013, Article ID 219545, doi:10.5402/2013/219545, 2013.

CHAPTER 3

Materials and Methods

For this thesis, all design and experimentation was done in MATLAB and Simulink. MATLAB was used to calculate controller gains and Simulink was used to connect control blocks and run experiments. Controller design was aided by using the `digcontr` library [1] provided by Dr. Vaccaro, a professor at the University of Rhode Island. This library contains functions for designing regulators, tracking systems, and observers using robust pole placement to calculate controller gains. These functions were designed to provide improved H_∞ robustness for MIMO systems compared with the `place` function provided by MATLAB.

The Marine Systems Simulator (MSS) library developed at the Norwegian University of Science and Technology (NTNU) was used for marine models and Simulink examples of guidance, navigation, and control (GNC). The GNC section of this library contains models for the CyberShip II, a dynamic positioning (DP) control example for the CyberShip II, and many other useful marine control blocks. These Simulink blocks aided in the development of realistic marine simulations for testing state space control systems.

3.1 The Model

The maneuvering objectives of URI's ASV dictate that a DP model in 3 DOF should be used to design the control system. If the state of the model is assumed to be $\boldsymbol{x} = [\boldsymbol{\eta}_p^\intercal, \boldsymbol{\nu}^\intercal]^\intercal$, the input is $\boldsymbol{u} = \boldsymbol{\tau}$, and the position $\boldsymbol{\eta}_p$ is measured, then the linear time-invariant state space model can be derived using the DP model in equation 38,

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{E}\boldsymbol{\tau}_{\text{disturb}} \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x} \end{aligned} \tag{43}$$

L	1.255	m	$X_{\dot{u}}$	-2.0	kg	X_u	-0.72253	kg/s
B	0.29	m	$Y_{\dot{v}}$	-10.0	kg	Y_v	-0.88965	kg/s
m	23.8	kg	$Y_{\dot{r}}$	-0.0	kgm	Y_r	-7.250	kgm/s
I_z	1.76	kgm^2	$N_{\dot{v}}$	-0.0	kgm	N_v	0.0313	kgm/s
x_g	0.046	m	$N_{\dot{r}}$	-1.0	kgm^2/s	N_r	-1.900	kgm^2/s
$X_{\|u\|u}$	-1.3274	kg/m	X_{uuu}	-5.8664	kgs/m^2	T_{surge}	0.1	
$Y_{\|v\|v}$	-36.4729	kg/m	$N_{\|v\|v}$	3.9565	kg	T_{sway}	0.11	
$Y_{\|r\|v}$	-0.805	km	$N_{\|r\|v}$	0.130	kgm	T_{yaw}	0.12	
$Y_{\|v\|r}$	-0.845	km	$N_{\|v\|r}$	0.080	kgm			
$Y_{\|r\|r}$	-3.450	kgm	$N_{\|r\|r}$	-0.750	kgm			

Table 3: CyberShip II experimentally identified parameters

where the matrices A, B, C, and E are expressed in (44).

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & -\boldsymbol{M}^{-1}\boldsymbol{D} \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{M}^{-1} \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{E} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{M}^{-1} \end{bmatrix} \tag{44}$$

An alternative representation includes the thrusters $\boldsymbol{\tau}$ in the state $\boldsymbol{x}$ and uses the commanded thrust $\boldsymbol{\tau}_{\text{com}}$ (see section 2.3.5) as the input $\boldsymbol{u}$. The state becomes $\boldsymbol{x} = [\boldsymbol{\eta}_p^\intercal, \boldsymbol{\nu}^\intercal, \boldsymbol{\tau}^\intercal]^\intercal$ and $\boldsymbol{u} = \boldsymbol{\tau}_{\text{com}}$, making the model in (45).

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{M}^{-1}\boldsymbol{D} & \boldsymbol{M}^{-1} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{thr} \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{A}_{thr} \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{E} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{M}^{-1} \\ \boldsymbol{0} \end{bmatrix} \tag{45}$$

Adding thruster dynamics to the model is a good approximation for the unknown behavior of the nonlinear thrusters. A controller designed with this model should be robust to unmodeled dynamics.

To fill in these models and design the controller, parameters need to be obtained for $\boldsymbol{M}$ and $\boldsymbol{D}$. Thankfully, system identification has been done on the CyberShip II which is similar in size to the target ASV. A controller designed using these parameters should be able to serve as a guide for the URI ASV when its parameters are determined. The parameters for the CyberShip II [2] are found in Table 3. All damping parameters except those requiring yaw motion (Y_r, N_r, $Y_{|r|v}$, ...) were identified experimentally by Skjetne [3] during tow tests. The thruster

X_u	-2.0	kg/s
Y_v	-7.0	kg/s
Y_r	-0.1	kgm/s
N_v	-0.1	kgm/s
N_r	-0.5	kgm^2/s

Table 4: CyberShip II estimated parameters from MSS

dynamics were estimated using engineering judgment by the author. The other parameters were identified by Skjetne using an adaptive maneuvering controller while performing an ellipse maneuver. Before this system identification was done, these parameters were estimated and used for control in a CyberShip II DP example from MSS [4]. These parameters are shown in table 4. Now that a suitable model has been filled, it is possible design the linear state space control system.

3.2 Controller Design

The goal of the controller is to take a reference input such as a track of an ellipse and follow it with minimal error. As a test, a full state space regulator can be designed as shown in figure 2 to hold the vehicle at the origin. For a regulator, it is assumed that all states $\boldsymbol{x} = [\boldsymbol{\eta}_p^\intercal, \boldsymbol{\nu}^\intercal]^\intercal$ are measured and the sampling rate is T. Poles were selected based on the rules in [5] to achieve the settling time T_s.

Closed-Loop Regulator Pole Selection Methods

1. Use sufficiently damped plant poles: $\Re(pole) < \frac{s_1}{T_s}$
2. Use added damping poles if the pole is not sufficiently damped and has complex parts: $\frac{s_1}{T_s} < pole < 0$. Replace the pole's real part with $\frac{s_1}{T_s}$ to damp the pole while keeping the imaginary part the same: $pole = \frac{s_1}{T_s} + \Im(pole)$.
3. For unstable poles of $\boldsymbol{A}$, if $-|\Re(pole)| < \frac{s_1}{T_s}$, then it can be used as a closed-loop pole.
4. Use normalized Bessel poles scaled by settling time: $\frac{s_n}{T_s}$

These continuous s-plane poles must be converted to the z-plane before being used

to calculate gains for a digital system. This is done using

$$\text{zpole} = e^{T \cdot \text{spole}} \tag{46}$$

Once the closed-loop pole locations have been selected to obtained the desired settling time, the $\boldsymbol{L}$ gains can be calculated. For SISO systems, any method can be used to calculate these gains because there is a single solution. In the case of this MIMO systems there are infinite solutions to the pole placement problem. This means that there needs to be a method for assessing the quality of the gains chosen. Classical stability margins don't take into account perturbations in multiple inputs simultaneously. Instead of using these classical methods the H_∞ norm is used to test the robustness of the gains chosen from pole placement. This theory is based on input-output stability of control systems as shown in section 2.2.5.

The `rfbg` function, an experimental function under development for inclusion in the `digcontr` library, optimizes the robustness norm of the system and places the desired closed-loop poles. The resulting robustness norm δ for the discrete feedback system with the gains $\boldsymbol{L}$ can then be compared with other methods. A good measure as to what qualifies as a acceptable robustness is that δ should be ≥ 0.5. Also, tenths matter. For example, a δ of 0.7 is noticeably better than 0.6.

A tracking system will allow the system to follow a given input such as a desired path. For a tracking system, the additional dynamics must first be chosen. The main roll of the additional dynamics is to integrate the error signal to track the desired input and reject constant or slowly varying error. To accomplish this, a SISO discrete additional dynamics can simply be $\boldsymbol{\Phi}_a = 1$, $\boldsymbol{\Gamma}_a = 1$.The additional dynamics can also be selected to reject a specific disturbance such as waves and then replicated for each input. For example a wave spectrum transfer function of the form in (47) can be canceled by choosing the roots of the denominator of $G(s)$

to be the eigenvalues of $\boldsymbol{A}_a$.

$$G(s) = \frac{2\lambda\omega_0\sigma}{s^2 + 2\lambda\omega_0 s + \omega_0^2} \tag{47}$$

After the additional dynamics have been chosen, the open-loop design model for the system can be calculated by forming the combined state $\boldsymbol{x}_d = [\boldsymbol{x}^\intercal, \boldsymbol{x}_a^\intercal]^\intercal$.

$$\boldsymbol{A}_{d\_o} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{B}_a\boldsymbol{C} & \boldsymbol{A}_a \end{bmatrix}, \quad \boldsymbol{B}_{d\_o} = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{C}_{d\_o} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \tag{48}$$

Poles must be selected for design model using the pole selection method below [5].

Closed-Loop Tracking System Pole Selection Methods

1. If the plant has slow stable zeros: $\Re(zero) > 4\frac{s_1}{T_s}$, then choose this as a pole. The pole won't affect the settling time of the tracking system because it will be canceled by the plant zero.
2. Use any of the rules for choosing regulator poles.

The poles that were chosen for the design model can be fed with $\boldsymbol{A}_{d\_o}$ and $\boldsymbol{B}_{d\_o}$ into the `rfbg` function to obtain gains $\boldsymbol{L}_d$.

$$\boldsymbol{L}_d = \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{L}_2 \end{bmatrix} \tag{49}$$

Robustness of this closed loop system can be calculated by using the input perturbation model [6] for error. This is done by introducing a system Δ with input f and output w at the input of the design model and calculating the system from w to f.

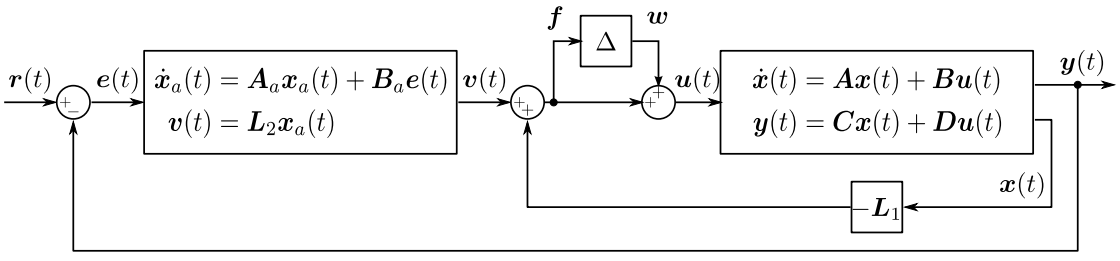


Figure 8: A tracking system with input perturbation Δ

To do this, start with the formulas for the plant and additional dynamics and reduce the equations until only the states ($\boldsymbol{x}$, $\boldsymbol{x}_a$) and the input $\boldsymbol{w}$ and output $\boldsymbol{f}$ remain. The reference input is set to 0 for this derivation. This means that robustness is not affected by any other blocks attached to the reference input, such as the FIF. An example derivation for the tracking system is given below.

$$\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} & \dot{\boldsymbol{x}}_a &= \boldsymbol{A}_a\boldsymbol{x}_a + \boldsymbol{B}_a\boldsymbol{e} \\
\dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}(\boldsymbol{w} + \boldsymbol{f}) & \dot{\boldsymbol{x}}_a &= \boldsymbol{A}_a\boldsymbol{x}_a - \boldsymbol{B}_a\boldsymbol{C}\boldsymbol{x} \\
\dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}(\boldsymbol{w} + \boldsymbol{v} - \boldsymbol{L}_1\boldsymbol{x}) & & \\
\dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}(\boldsymbol{w} + \boldsymbol{L}_2\boldsymbol{x}_a - \boldsymbol{L}_1\boldsymbol{x}) & \boldsymbol{f} &= \boldsymbol{v} - \boldsymbol{L}_1\boldsymbol{x} \\
\dot{\boldsymbol{x}} &= (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{L}_1)\boldsymbol{x} + \boldsymbol{B}\boldsymbol{L}_2\boldsymbol{x}_a + \boldsymbol{B}\boldsymbol{w} & \boldsymbol{f} &= \boldsymbol{L}_2\boldsymbol{x}_a - \boldsymbol{L}_1\boldsymbol{x}
\end{aligned} \tag{50}$$

This results in the matrix from of the design model from $\boldsymbol{w}$ to $\boldsymbol{f}$ shown in (51). Now the results of section 2.2.5 can be applied with the system Δ representing H_1. Minimizing the H_∞ norm of the system from $\boldsymbol{w}$ to $\boldsymbol{f}$ results in the greatest stability robustness.

$$\boldsymbol{A}_{d\_r} = \begin{bmatrix} \boldsymbol{A} - \boldsymbol{B}\boldsymbol{L}_1 & \boldsymbol{B}\boldsymbol{L}_2 \\ -\boldsymbol{B}_a\boldsymbol{C} & \boldsymbol{A}_a \end{bmatrix}, \quad \boldsymbol{B}_{d\_r} = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{C}_{d\_r} = \begin{bmatrix} -\boldsymbol{L}_1 & \boldsymbol{L}_2 \end{bmatrix} \tag{51}$$

The experimental `rfgb` function from the `digcontr` library directly minimizes the H_∞ norm of the robustness design model from $\boldsymbol{w}$ to $\boldsymbol{f}$, such as the one in equation 51. This function optimizes the gain matrices $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ until maximum robustness is achieved. MATLAB's `place` function uses a different method to attempt to choose acceptable gains. The `place` function tries to make the closed loop eigenvectors as orthogonal as possible. This is only indirectly related to robustness [7]. Therefore, `rfbg` produces gains that have higher robustness than `place`.

The final resulting closed-loop design from $\boldsymbol{r}$ to $\boldsymbol{y}$ for the tracking system is

$$\boldsymbol{A}_{d\_c} = \begin{bmatrix} \boldsymbol{A} - \boldsymbol{B}\boldsymbol{L}_1 & \boldsymbol{B}\boldsymbol{L}_2 \\ -\boldsymbol{B}_a\boldsymbol{C} & \boldsymbol{A}_a \end{bmatrix}, \quad \boldsymbol{B}_{d\_c} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{B}_a \end{bmatrix}, \quad \boldsymbol{C}_{d\_c} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \tag{52}$$

3.3 Observer Design

Up to now it has be assumed that all plant states are measured. However, this is rarely the case in real systems. When only some states can be measured the rest of the states must be estimated using an observer. Designing the observer is similar to regulator and tracking system design. The rules for choosing pole locations of the observer are below [5].

Observer Pole Selection Methods

1. If the plant has zeros with $\Re(zero) < 0$, then use these for observer poles.
2. If the plant has zeros with $\Re(zero) > 0$, then use the reflection of the zero about the imaginary axis: $pole = -\Re(zero) + \Im(zero)$.
3. Use normalized Bessel poles that are three to five time faster than the desired settling time of the control system.

Use `obg_ts`, another experimental function under development for inclusion in the `digcontr` library, to design the observer with these desired closed-loop poles for the tracking system. The robustness of this system can again be determined by forming the model from $\boldsymbol{w}$ to $\boldsymbol{f}$ with states $\boldsymbol{x}_d = [\boldsymbol{x}^\intercal, \hat{\boldsymbol{x}}^\intercal, \boldsymbol{x}_a^\intercal]^\intercal$ by inserting the unknown model Δ at the input $\boldsymbol{u}$. The resulting model is

$$\boldsymbol{A}_{d\_r} = \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{B}\boldsymbol{L}_1 & \boldsymbol{B}\boldsymbol{L}_2 \\ \boldsymbol{K}\boldsymbol{C} & \boldsymbol{A} - \boldsymbol{K}\boldsymbol{C} - \boldsymbol{B}\boldsymbol{L}_1 & \boldsymbol{B}\boldsymbol{L}_2 \\ -\boldsymbol{B}_a\boldsymbol{C} & \boldsymbol{0} & \boldsymbol{A}_a \end{bmatrix}, \quad \boldsymbol{B}_{d\_r} = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \tag{53}$$

$$\boldsymbol{C}_{d\_r} = \begin{bmatrix} \boldsymbol{0} & -\boldsymbol{L}_1 & \boldsymbol{L}_2 \end{bmatrix}, \qquad \boldsymbol{D}_{d\_r} = \boldsymbol{D}$$

The closed-loop design model from $\boldsymbol{r}$ to $\boldsymbol{y}$ which will be used to calculate the

inverse filter is

$$\boldsymbol{A}_{d\_c} = \begin{bmatrix} \boldsymbol{A} & -\boldsymbol{B}\boldsymbol{L}_1 & \boldsymbol{B}\boldsymbol{L}_2 \\ \boldsymbol{K}\boldsymbol{C} & \boldsymbol{A} - \boldsymbol{K}\boldsymbol{C} - \boldsymbol{B}\boldsymbol{L}_1 & \boldsymbol{B}\boldsymbol{L}_2 \\ -\boldsymbol{B}_a\boldsymbol{C} & \boldsymbol{0} & \boldsymbol{A}_a \end{bmatrix}, \quad \boldsymbol{B}_{d\_c} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{B}_a \end{bmatrix}, \tag{54}$$

$$\boldsymbol{C}_{d\_c} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \qquad \boldsymbol{D}_{d\_c} = \boldsymbol{D}$$

3.4 Feedforward Inverse Filter

To ensure that the designed observer-based tracking system tracks the desired input exactly, a feedforward inverse filter (FIF) [8] can be designed. The inverse filter is an exact inverse of the design model and generates an input to the controller that will produce the desired position with a delay of d samples, where d is calculated in (55). Using the discrete design model ($\boldsymbol{\Phi}_d$, $\boldsymbol{\Gamma}_d$, $\boldsymbol{C}_d$, $\boldsymbol{D}_d$) the exact inverse filter ($\boldsymbol{\Phi}_f$, $\boldsymbol{\Gamma}_f$, $\boldsymbol{C}_f$, $\boldsymbol{D}_f$) can be calculated using (55).

$$\begin{aligned} d &= \min_d(\boldsymbol{C}_d\boldsymbol{\Phi}_d^{d-1}\boldsymbol{\Gamma}_d \neq \boldsymbol{0}) \\ \boldsymbol{D}_f &= (\boldsymbol{C}_d\boldsymbol{\Phi}_d^{d-1}\boldsymbol{\Gamma}_d)^{-1} \\ \boldsymbol{C}_f &= -\boldsymbol{D}_f\boldsymbol{C}_d\boldsymbol{\Phi}_d^d \\ \boldsymbol{\Gamma}_f &= \boldsymbol{\Gamma}_d\boldsymbol{D}_f \\ \boldsymbol{\Phi}_f &= \boldsymbol{\Phi}_d + \boldsymbol{\Gamma}\boldsymbol{C}_f \end{aligned} \tag{55}$$

When creating the inverse filter, the zeros and poles of the design model will become the poles and zeros of the filter respectively to create an exact inverse. If there are high frequency zeros in the design model caused by sampling, then the inverse filter will attempt to cancel these out resulting in high frequency oscillations. This can be remedied by choosing appropriate initial conditions, as outlined in section 3.6. These oscillations die out after a while and therefore can be canceled out completely by choosing good initial conditions. A simple way to do this is to simulate the commanded input backwards through the inverse filter until the oscillations are no longer present in the filter output. Then take the final state of the inverse filter,

now the initial state, and negate it to get the new initial state. Simulating the filter forward with the reference input and the new initial state should result in no unwanted oscillations.

If the design model has eigenvalues outside of the unit circle, then the resulting inverse filter will be unstable. This can be fixed by incrementing d by one. This creates a band-limited inverse filter that is valid for low frequencies. To prevent high frequencies from being input into the FIF, care should be taken to smooth the input to the filter. Notice that the FIF is not in the feedback loop and does not affect the robustness calculations.

3.5 Track Generation

The goal maneuver was to perform an ellipse similar to Fossen in [3]. The desired path can generated using the parametric equation of the ellipse centered at the origin with semi-major axis a and semi-minor axis b. Recall that $\boldsymbol{x}$ is north and $\boldsymbol{y}$ is east and notice that $\boldsymbol{y}_d$ is inverted so that the ellipse turns clockwise.

$$\begin{aligned} \boldsymbol{x}_d(t) &= b\sin(t) \\ \boldsymbol{y}_d(t) &= -a\cos(t) \\ \boldsymbol{\psi}_d(t) &= \tan^{-1}(\frac{a}{b}\tan(t)) \\ \boldsymbol{\eta}_d(t) &= \begin{bmatrix} \boldsymbol{x}_d(t) & \boldsymbol{y}_d(t) & \boldsymbol{\psi}_d(t) \end{bmatrix}^\intercal \end{aligned} \tag{56}$$

To generate a single rotation about the ellipse the parametric variable t can be replaced by a vector from 0 to 2π that has number of elements equal to TS where T is the sampling interval and S is the amount of time to travel along the ellipse.

3.6 Initial Conditions

The initial conditions for $\boldsymbol{\eta}$ are simply $\boldsymbol{\eta}_d(0)$ and the initial velocities and accelerations can be calculated by taking derivatives of the parametric equations.

$$\begin{aligned} \dot{\boldsymbol{x}}_d(t) &= b\cos(t) \\ \dot{\boldsymbol{y}}_d(t) &= a\sin(t) \\ \boldsymbol{r}_d(t) &= \frac{ab}{a^2\sin(t)^2 + b^2\cos(t)^2} \\ \boldsymbol{\nu}_0 &= \begin{bmatrix} \dot{\boldsymbol{x}}_d(0) & \dot{\boldsymbol{y}}_d(0) & \boldsymbol{r}_d(0) \end{bmatrix}^\intercal \end{aligned} \tag{57}$$

$$\begin{aligned} \ddot{\boldsymbol{x}}_d(t) &= -b\sin(t) \\ \ddot{\boldsymbol{y}}_d(t) &= a\cos(t) \\ \dot{\boldsymbol{r}}_d(t) &= \frac{2ab(b^2 - a^2)\sin(2t)}{(a^2\sin(t)^2 + b^2\cos(t)^2)^2} \\ \dot{\boldsymbol{\nu}}_0 &= \begin{bmatrix} \ddot{\boldsymbol{x}}_d(0) & \ddot{\boldsymbol{y}}_d(0) & \dot{\boldsymbol{r}}_d(0) \end{bmatrix}^\intercal \end{aligned} \tag{58}$$

The initial thrust can be calculated using the linear DP model and the initial acceleration and velocities.

$$\boldsymbol{u}_0 = \boldsymbol{\tau}_0 = M\dot{\boldsymbol{\nu}}_0 + D\boldsymbol{\nu} \tag{59}$$

Now the initial state $\boldsymbol{x}_0$ can be used to calculate the initial state of the additional dynamics.

$$\boldsymbol{x}_{a0} = \boldsymbol{L}_2^\intercal(\boldsymbol{L}_2\boldsymbol{L}_2^\intercal)^{-1}(\boldsymbol{L}_1\boldsymbol{x}_0 + \boldsymbol{u}_0) \tag{60}$$

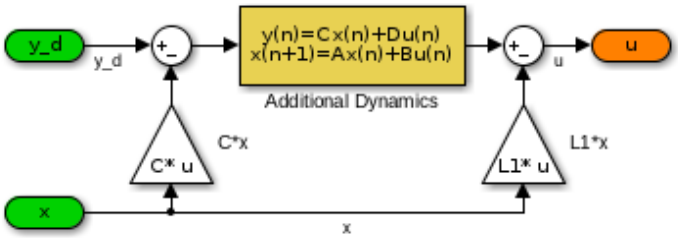
The initial observer state $\hat{\boldsymbol{x}}_0$ can be initialized to match the initial plant state $\boldsymbol{x}_0$.

3.7 Implementation in Simulink

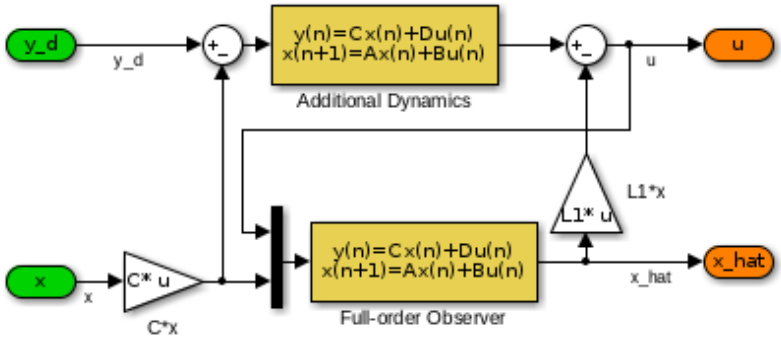
Simulink was used to simulate the designed controller and boat model. Disturbances were added to simulate waves, wind, and measurement noise to make the simulation more realistic. Many debugging tools such as scopes to view signals, outputs to the MATLAB workspace, and switches to toggle elements were used.

3.7.1 The Controller

To build the tracking system controller, two Simulink blocks were made without model connections so that models could be swapped later. The first control block uses full state feedback while the other uses a full-order observer. In each case discrete state space blocks were used for the designed additional dynamics, gain blocks were used for the $\boldsymbol{L}_1$ gains and the output selection matrix $\boldsymbol{C}$, and initial conditions were set based on section 3.5. When the observer was used, a discrete state space block was implemented with input $\boldsymbol{u}_o = [\boldsymbol{u}, \boldsymbol{C}\boldsymbol{x}]^\intercal$ by the use of a multiplexer, and $\boldsymbol{B} = [\boldsymbol{K}, \boldsymbol{\Gamma}]$. The full state feedback tracking system and observer-based tracking system were connected as in (4) and (5) respectively.



(a) The full state feedback tracking system



(b) The observer-based tracking system

Figure 9: The two controller blocks without plant models

3.7.2 Models

First a linear boat model based on vessel parallel coordinates was created as a continuous state space block. This is the same model that the controller was

designed for and was used to verify that the controller was functioning as expected. It is a rough approximation of the actual real world vehicle.

For the nonlinear CyberShip II model, a previous Simulink block from the MSS library was modified to add nonlinearities and thruster dynamics. Orange blocks are modifications to the MSS model and include the thruster machineries for each DOF and the nonlinear matrices $\boldsymbol{D}_n(\boldsymbol{\nu})$, $\boldsymbol{C}_{RB}(\boldsymbol{\nu})$, and $\boldsymbol{C}_A(\boldsymbol{\nu})$. An input for external disturbances and an output for total force were added to the model as well. The simulation model was based on equation 35 and includes switches to toggle all nonlinear elements for testing purposes. Care was taken to initialize the integrators with appropriate initial velocity and acceleration.

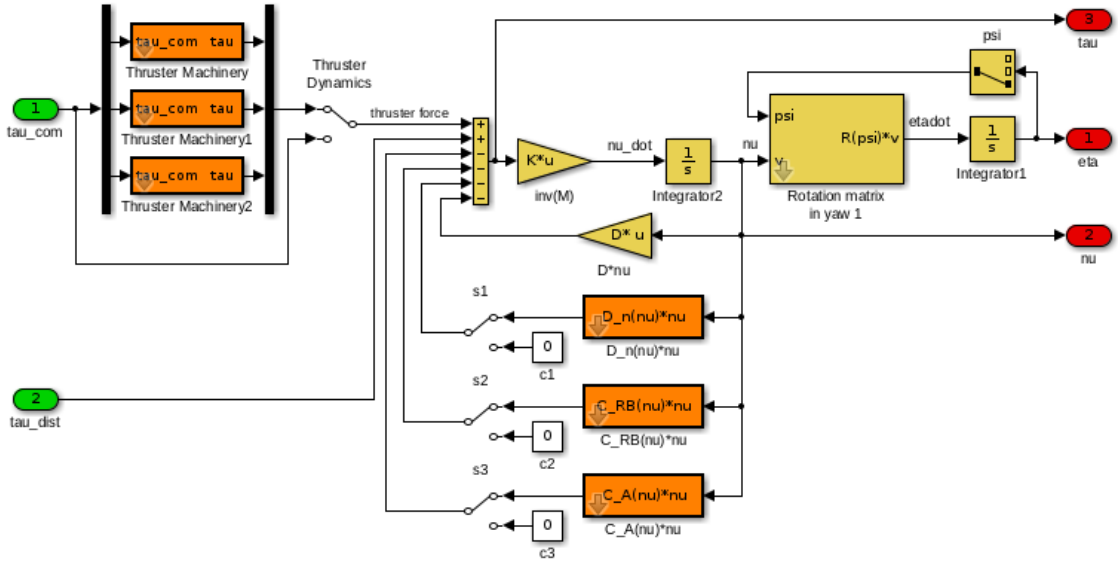


Figure 10: The nonlinear model used in simulation

The thrusters were modeled as time constants in surge, sway, and yaw (see equation 40). A thrust configuration matrix was not used because it is assumed that the necessary thrust can be generated. This means that the model can be the same for any fully actuated vehicle.

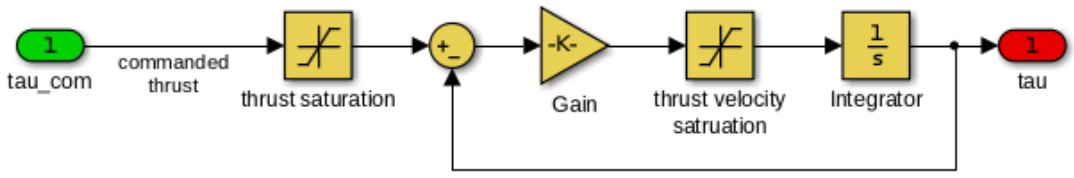


Figure 11: The thruster model

3.7.3 Feedforward Inverse Filter

The FIF was added between the desired input and the control system to enhance its performance. Its implementation was a simple discrete state space block. A switch was added to enable and disable the FIF to observe its effects on the control system. The desired path was input into the FIF from the MATLAB workspace, transformed to $\boldsymbol{\eta}_p$ coordinates, and then sent into the controller.

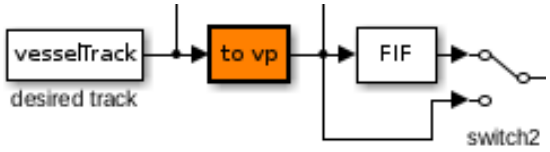


Figure 12: The Feedforward Inverse Filter with transformation of $\boldsymbol{\eta}_d$ to $\boldsymbol{\eta}_p$

3.7.4 Visualizations and Disturbances

Scopes were added for viewing signals such as position, velocity, and error in real time. An xy-plot of the vehicle position displays the marine vehicle as it travels along the desired path.

Disturbances were added for waves, wind, and measurement noise, with switches to toggle each on and off. Waves were implemented as band-limited white noise passed through a second-order wave spectrum transfer function as described by equation 47. Wind was implemented as a constant direction force acting on the model. To do this, the force had to be rotated from the global frame so that it always acted on the boat in the same direction. For example, if the wind direction is from south to north and the boat is facing north, then the boat is going to be

pushed in the positive surge direction. If the boat turns to the east, then the boat will be pushed in the negative sway direction.

The measurement noise was added to the output of the model before the transformation to vessel parallel coordinates was performed to simulate how the controller would actually be implemented. Separate white noise blocks were used to simulate noise for the GPS position and orientation from the compass so that they could be adjusted individually.

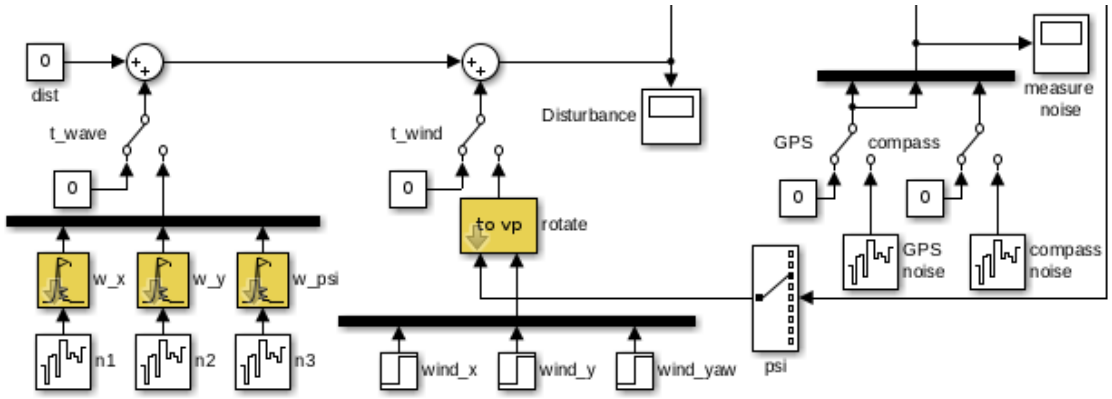


Figure 13: The disturbances available

3.7.5 The Full Simulation

The controller, model, FIF, disturbances, and visualizations were connected to form a versatile testing platform for any surface vehicle. The output of the non-linear model is $\boldsymbol{\eta}$, so it needed to be transformed back to vessel parallel coordinates $\boldsymbol{\eta}_p$ using equation 36 before being sent back into the controller. A three sample delay was needed for calculating error and converting to vessel parallel coordinates due to the effects of the feedforward inverse filter described in 3.4. Unneeded modules can be easily disabled via switches and others can be added if needed. There are a total of four different simulations using the ellipse path. Two models for a full state feedback tracking system with and without the thrusters in the state and two for an observer-based tracking system with and without thrusters.

List of References

[1] R. J. Vaccaro, "Digital control library," 2012.

[2] R. Skjetne, "The Maneuvering Problem," Phd Dissertation, Norwegian University of Science and Technology, 2005.

[3] R. Skjetne, Ø. N. Smogeli, and T. I. Fossen, "A Nonlinear Ship Manoeuvering Model: Identification and adaptive control with experiments for a model ship," *Modeling, Identification and Control*, vol. 25, no. 1, pp. 3–27, 2004.

[4] Norwegian University of Science and Technology. "MSS. Marine Systems Simulator." 2010. [Online]. Available: http://www.marinecontrol.org

[5] R. J. Vaccaro, "Rules for selecting poles," 2012, personal communication.

[6] J. B. Burl, *Linear Optimal Control*. Addison Wesley Longman, Inc., 1999.

[7] R. J. Vaccaro, *Digital Control: A State-Space Approach*. McGraw-Hill, Inc., 1995.

[8] R. J. Vaccaro, "Personal communication," 2012.

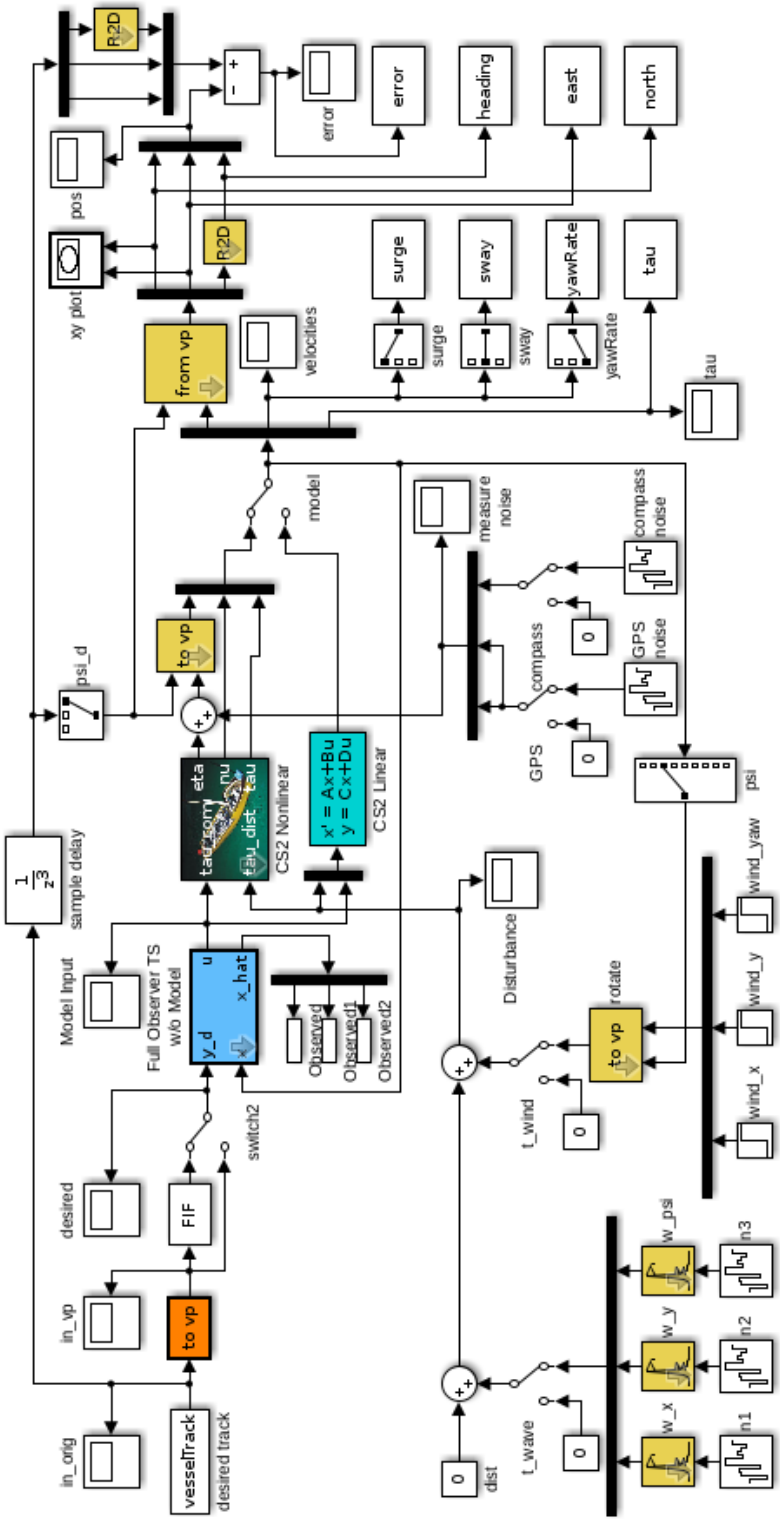


Figure 14: The full observer-based tracking system with thruster model

CHAPTER 4

Results and Discussion

4.1 Calculating the Controller

The plant model that includes the thrusters in the state (45) was chosen because it was most accurate description of the physical system. The model parameters used were those from table 3 and damping parameters were chosen to be the estimated parameters in table 4. This way the real parameters can be used during simulation to test robustness. The resulting continuous matrices for the CyberShip II become

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & -\boldsymbol{M}^{-1}\boldsymbol{D} & \boldsymbol{M}^{-1} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{A}_{thr} \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{A}_{thr} \end{bmatrix}, \quad \boldsymbol{C} = \begin{bmatrix} \boldsymbol{I} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \boldsymbol{E} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{M}^{-1} \\ \mathbf{0} \end{bmatrix} \tag{61}$$

The continuous model can then be converted to discrete time using the `zohe` function from the `digcontr` library or MATLAB's `c2d` function.

The sampling time used in the controller design was $T = 0.1$ seconds and the settling time was chosen as $T_s = 5$ seconds. The discrete additional dynamics matrices were chosen to act as a simple integrator replicated three times, one for each tracked output signal which results in an identity matrix for $\boldsymbol{\Phi}_a$ and $\boldsymbol{\Gamma}_a$.

$$\boldsymbol{\Phi}_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{\Gamma}_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{62}$$

The closed-loop poles for the additional dynamics can be chosen to be a repeated pole at s_1/T_s. The $\boldsymbol{A}$ matrix for the CyberShip II has three sufficiently damped poles caused by the thrusters, which settle in about 0.5 seconds. In this case, we choose these as closed-loop poles and resort to the second and fourth normalized Bessel poles scaled by the settling time for the rest. Combining these,

the twelve continuous closed-loop poles end up being

$$\texttt{spoles} = \begin{bmatrix} -0.9240 & -0.9240 & -0.9240 & -10.0 & -9.0909 & -8.3333 & \dots \\ & & & -0.8106 \pm 0.4680i & -0.8031 \pm 1.0145i & -1.1056 \pm 0.3311i \end{bmatrix} \tag{63}$$

The continuous poles can be converted to discrete time using equation 46 and used to calculate the digital L gains can be calculated. This is achieved using the experimental `rfbg` function from Vaccaro [1]. This function optimizes the robustness norm of the system and places the desired closed-loop poles. The gains for this discrete system end up being

$$\boldsymbol{L_1} = \begin{bmatrix} 57.7263 & 4.5533 & -0.9538 & 62.2768 & 4.6527 & -0.5890 \\ 6.8580 & 89.2700 & 1.4620 & 2.6220 & 82.8885 & 1.9821 \\ -4.5006 & 4.0262 & 10.3562 & -0.5318 & 6.4055 & 8.5309 \\ & & & 0.1650 & -0.0213 & -0.1775 \\ & & & 0.0030 & 0.1323 & -0.2589 \\ & & & 0.0062 & 0.0751 & 0.5068 \end{bmatrix} \tag{64}$$

$$\boldsymbol{L_2} = \begin{bmatrix} 1.8549 & -0.0331 & -0.0676 \\ 0.3968 & 3.0575 & -0.0025 \\ -0.3427 & 0.0833 & 0.4511 \end{bmatrix} \tag{65}$$

The resulting maximum robustness norm δ for the designed discrete tracking system using `rfbg` which stands for robust feedback gains, is 0.7287.

A four times faster full-order observer with settling time $T_s/4$ was designed to estimate the state variables from the three measured positions. The plant has no special properties to aid in selecting pole locations, so the Bessel poles are used. The second, third, and fourth Bessel poles are scaled to the observer settling time.

$$\texttt{opoles} = \begin{bmatrix} -3.2424 \pm 1.8720i & -3.1734 \pm 3.0276i & -4.0074 \\ & -3.2124 \pm 4.0578i & -4.4225 \pm 1.3243i \end{bmatrix} \tag{66}$$

Next, the plant model, plant gains, and observer poles are used to calculate the observer gains $\boldsymbol{K}$ using the `obg_ts` function. Similar to `rfbg`, the `obg_ts`

function optimizes the robustness of the controller.

$$\boldsymbol{K} = \begin{bmatrix} -0.55535 & -0.42743 & -0.55101 \\ -0.22118 & 0.04152 & -0.20381 \\ 1.77202 & 1.23628 & 1.58549 \\ 4.74671 & 1.39377 & 1.81438 \\ 0.51974 & 1.77561 & 0.46327 \\ -1.58401 & -0.88016 & 0.07099 \\ -1307.78 & -527.104 & -657.944 \\ -187.864 & -438.014 & -180.830 \\ 91.0399 & 53.2962 & 49.8671 \end{bmatrix} \tag{67}$$

This results in a final robustness of 0.5160.

Now the design model for the observer-based tracking system can be formed using equation 54 and the inverse filter can be calculated using equation 55. The resulting FIF has a delay of three samples and is a band-limited inverse that is valid for low frequencies.

The above procedure can be repeated using the same poles and MATLAB's `place` function. The results for the tracking system gains are

$$\boldsymbol{L_1} = \begin{bmatrix} 68.9017 & -5.4295 & 3.6896 & 69.9811 & -2.0964 & 1.8248 \\ 4.9876 & 80.6543 & 2.7687 & 1.8553 & 82.5026 & 2.7194 \\ -16.4466 & 7.0581 & 8.8568 & -9.7805 & 5.9246 & 7.6167 \\ & & & 0.2090 & -0.0035 & 0.0515 \\ & & & 0.0143 & 0.2439 & 0.0327 \\ & & & -0.0468 & 0.0430 & 0.3410 \end{bmatrix} \tag{68}$$

$$\boldsymbol{L_2} = \begin{bmatrix} 2.2919 & -0.3114 & 0.1852 \\ 0.3145 & 2.5721 & 0.1101 \\ -0.7378 & 0.3139 & 0.3630 \end{bmatrix} \tag{69}$$

The resulting robustness norm of the tracking system is 0.5927. The observer gains were then calculated to be

$$\boldsymbol{K} = \begin{bmatrix} 0.3290 & -0.1215 & 0.0283 \\ 0.0922 & 0.3668 & 0.2368 \\ 0.0255 & -0.1650 & 0.3759 \\ 1.8069 & 0.0964 & -0.1352 \\ -0.1153 & 1.2876 & -0.1502 \\ -0.0780 & 0.0742 & 1.3270 \\ -271.2921 & -85.6188 & 40.7033 \\ 77.2658 & -187.3975 & 162.9843 \\ 4.5341 & -13.0532 & -8.5007 \end{bmatrix} \tag{70}$$

This resulted in a final robustness of 0.2483. Again, the FIF has a delay of three samples.

4.2 Simulations

Once the two controllers were designed using the `place` and `rfgb` methods, simulations were run to test robustness. The desired path was an ellipse with semi-major axis $a = 20$ and semi-minor axis $b = 10$ (56) that takes 200 seconds to complete a single revolution. This ellipse is translated to the east by a meters so that the initial position is the origin and the boat travels clockwise around the ellipse.

$$\begin{aligned} t &= 0 : T : 200 \\ \boldsymbol{x}_d(t) &= 10 \sin(t\frac{2\pi}{200}) \\ \boldsymbol{y}_d(t) &= 20 - 20 \cos(t\frac{2\pi}{200}) \\ \boldsymbol{\psi}_d(t) &= \tan^{-1}(2 \tan(t\frac{2\pi}{200})) \\ \boldsymbol{\eta}_d(t) &= \begin{bmatrix} \boldsymbol{x}_d(t) & \boldsymbol{y}_d(t) & \boldsymbol{\psi}_d(t) \end{bmatrix}^\intercal \end{aligned} \tag{71}$$

The average forward velocity U was calculated to be 0.48 m/s, with a maximum of 0.62 m/s around the top and bottom of the ellipse. The average yaw rate around the ellipse is 1.8 deg/s, with a maximum of 3.6 deg/s. These were calculated directly from the derivatives of the parametric equations derived in equation 57. The linear model is linearized about $U = 0$, $r = 0$. Plots of these are shown in figure 15.

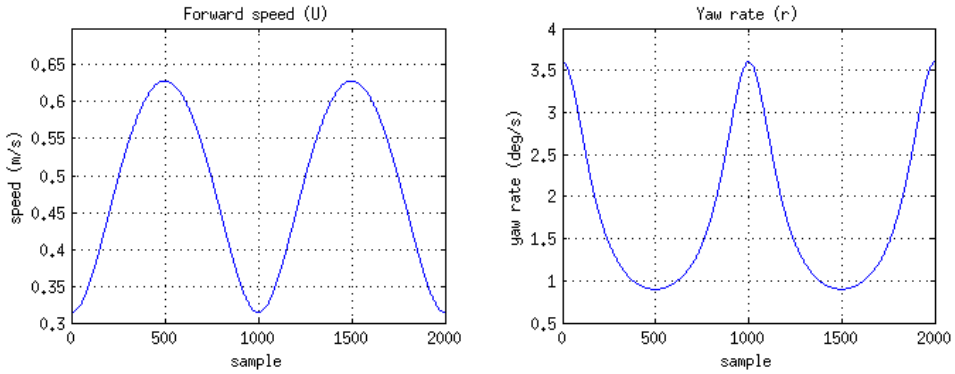


Figure 15: The forward speed and yaw rate around an ellipse in 200 seconds

The first set of simulations were run on the nonlinear model with no disturbances or perturbations of the plant model. Without the feedforward inverse filter, the controller goes unstable for both pole placement methods at this relatively high speed. The results can be found in figure 16.

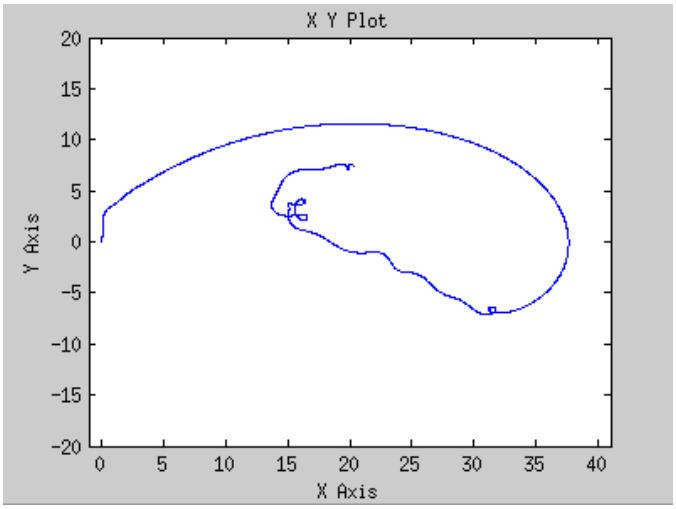


Figure 16: An example of when the FIF is not used

When the FIF is enabled the boat is able to travel the entire perimeter of the ellipse even when disturbances and model perturbations are introduced. For this testing, the wave spectrum had parameters $\sigma = 0.5m$, $\omega_0 = 0.8\ rad/s$, and $\lambda = 0.1$. The resulting wave disturbance had an amplitude of 0.1 Newton. The wind was chosen as a force constantly pushing the boat to the Northeast with magnitude

1 Newton. Measurement noise was modeled as white noise which was kept very low due to its high frequency component. Compass noise had a peak amplitude of about about 0.001 radians and GPS noise had a peak amplitude of 0.003 meters.

Model perturbations were chosen to be realistic measurement errors and roughly correspond to how difficult the parameters are to measure. The hydrodynamic derivatives adaptability and experimentally identified previously by Skjetne were used as perturbations because the controller was designed with the estimated parameters by Fossen in the Cybership II example from the MSS. Table 5 shows the perturbations of the model parameters.

param	**model**	**simulation**	**difference**
m	23.800	27.800	4.0
I_z	1.760	2.160	0.4
x_g	0.046	0.246	0.2
X_u	-2.0	-0.72253	1.2775
Y_v	-7.0	-0.88965	6.1104
Y_r	-0.1	-7.250	-7.1500
N_v	-0.1	0.03130	0.1313
N_r	-0.5	-1.900	-1.4000
$X_{\dot{u}}$	-2.0	-1.0	1.0
$Y_{\dot{v}}$	-10.0	7.0	3.0
$Y_{\dot{r}}$	0.0	-1.0	-1.0
$N_{\dot{v}}$	0.0	-1.2	-1.2
$N_{\dot{r}}$	-1.0	-1.8	-0.8
T_{surge}	0.1	0.3	0.2
T_{sway}	0.11	0.41	0.3
T_{yaw}	0.12	0.32	0.2

Table 5: CyberShip II model perturbations

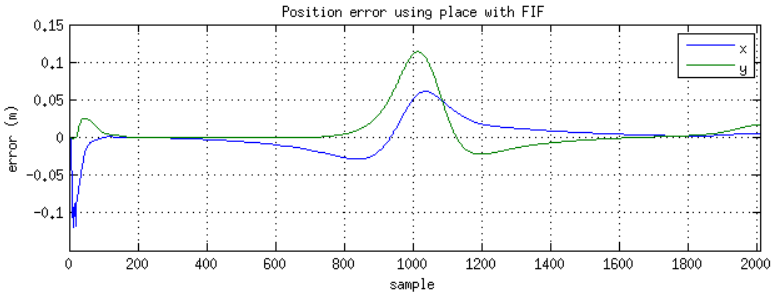
The results for the poles placed by `rfbg` and `place` are given in table 6 when simulated on a nonlinear model with no disturbances, wind and wave disturbances, and disturbances with model perturbations from table 5. Error is calculated as the difference between $\boldsymbol{\eta}_d$ and $\boldsymbol{\eta}$ with a three sample delay to account for the FIF. Standard deviations were calculated over the entire simulation for x, y, and ψ.

The maximum error is also displayed for each signal.

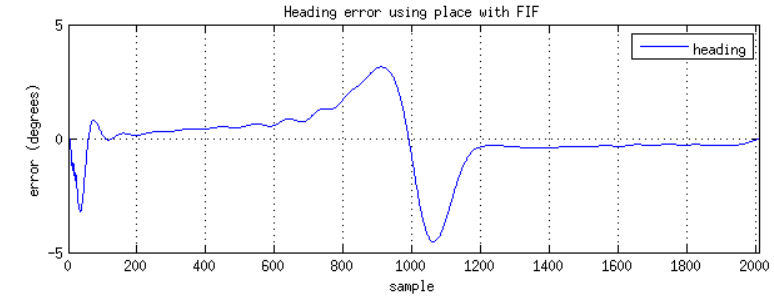
Standard Deviations				
method	**signal**	**no disturb**	**disturb**	**disturb, perturb**
place	x (m)	0.0204	0.0228	0.0823
	y (m)	0.0216	0.0275	0.0937
	heading (deg)	1.3135	3.1442	8.3100
rfbg	x (m)	0.0234	0.0239	0.0443
	y (m)	0.0240	0.0247	0.0342
	heading (deg)	1.1399	2.0326	6.7626
Absolute Maximum Error				
method	**signal**	**no disturb**	**disturb**	**disturb, perturb**
place	x (m)	0.1190	0.1209	0.2450
	y (m)	0.1138	0.1200	0.5008
	heading (deg)	4.5330	11.5593	19.2981
rfbg	x (m)	0.1134	0.1111	0.1188
	y (m)	0.1011	0.1078	0.1683
	heading (deg)	4.5620	5.3811	14.2147

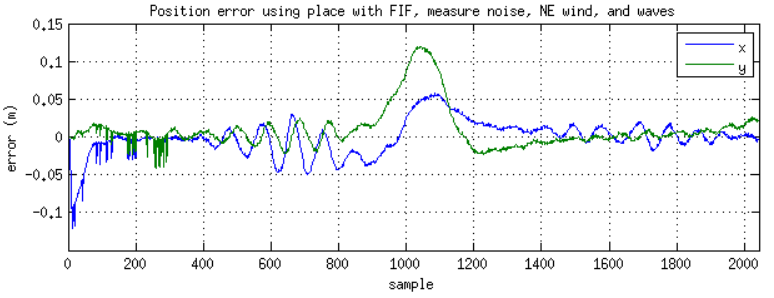
Table 6: CyberShip II standard deviations and maximum errors with different disturbances

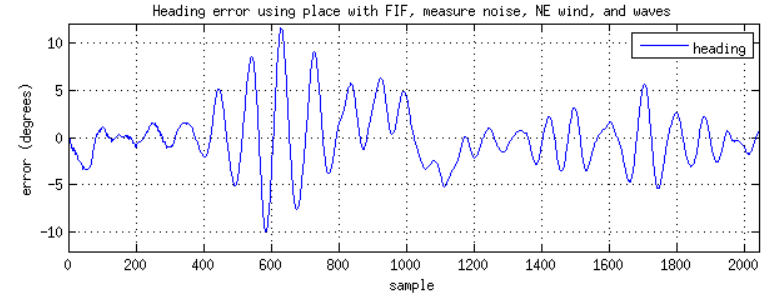
Plots of the error over time for each simulation using MATLAB's `place` function are displayed in the figures below. It is important to note the the thrust input was limited by the saturation of 3 Newtons in the sway direction when all disturbances and perturbations were used. Results are split into position error and heading error.

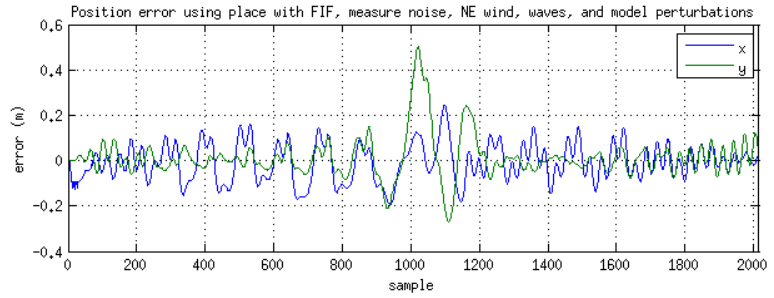


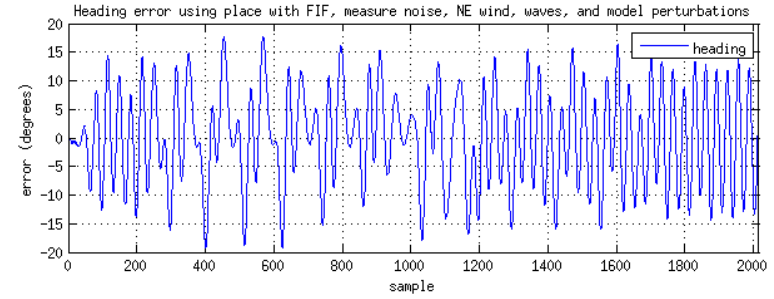
A comparison of both methods under all disturbances and perturbations is best viewed in the xy plots in Figure 17. The actual path of the simulated vehicle is shown.

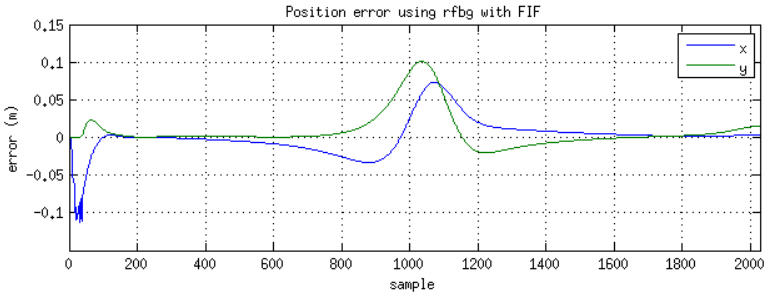

Heading error using place with FIF
heading
error (degrees)
sample

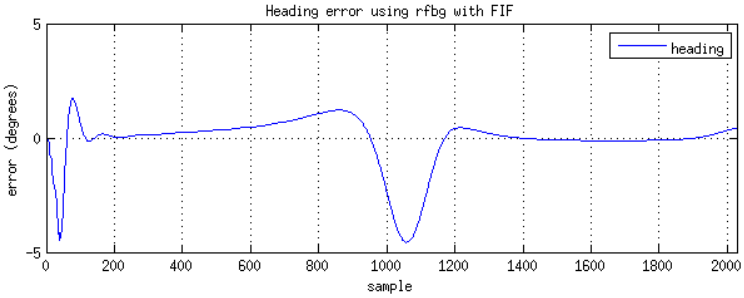

Position error using place with FIF, measure noise, NE wind, and waves
x
y
error (m)
sample

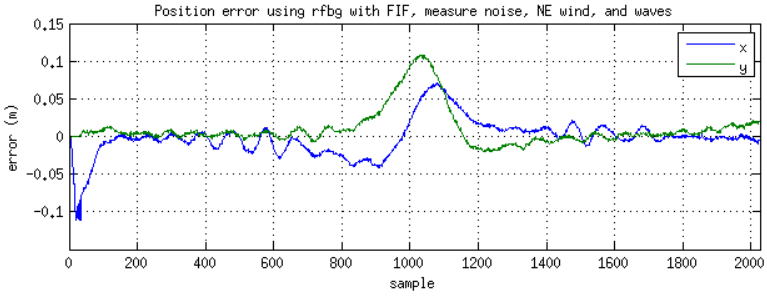

Heading error using place with FIF, measure noise, NE wind, and waves
heading
error (degrees)
sample

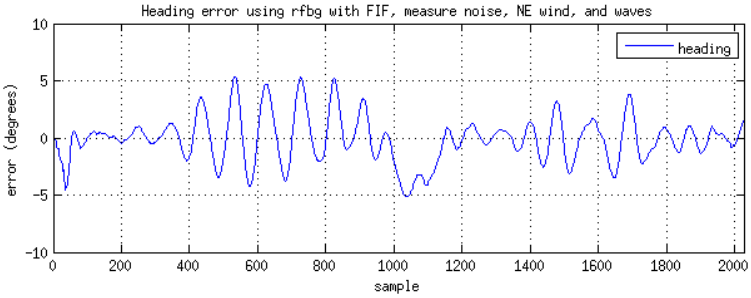

Position error using place with FIF, measure noise, NE wind, waves, and model perturbations
x
y
error (m)
sample

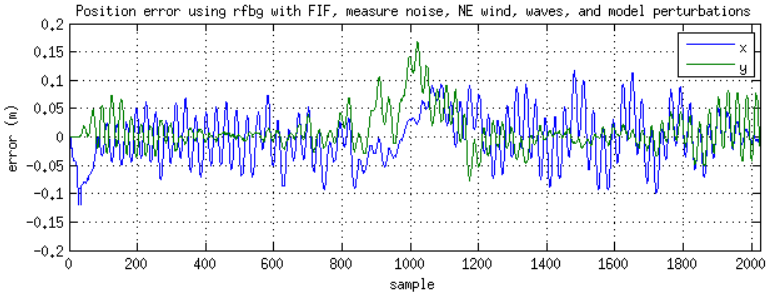

Heading error using place with FIF, measure noise, NE wind, waves, and model perturbations
heading
error (degrees)
sample

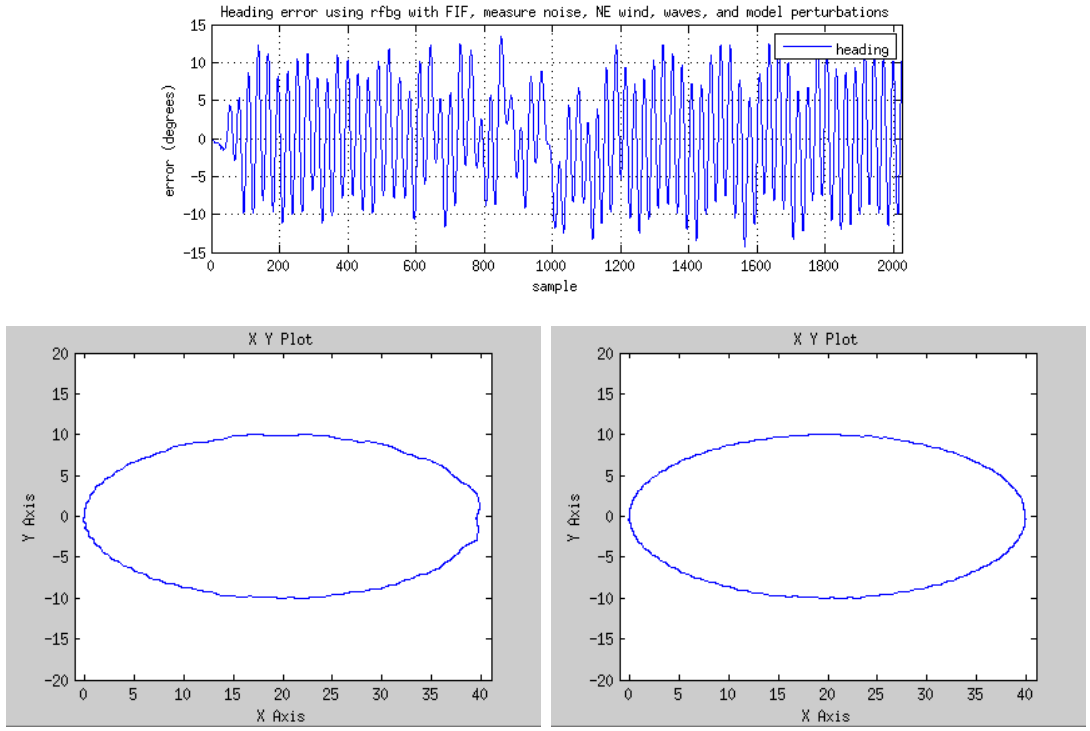

Position error using rfbg with FIF
error (m)
sample
x
y


Heading error using rfbg with FIF
error (degrees)
sample
heading


Position error using rfbg with FIF, measure noise, NE wind, and waves
error (m)
sample
x
y


Heading error using rfbg with FIF, measure noise, NE wind, and waves
error (degrees)
sample
heading


Position error using rfbg with FIF, measure noise, NE wind, waves, and model perturbations
error (m)
sample
x
y



(a) `place` (b) `rfbg`

Figure 17: xy plots of the ellipse maneuver under all disturbances and perturbations

A second set of simulations were done with different model perturbations shown in table 7. The parameters were altered more modestly and the zero values in the mass and damping matrices were altered to be small, but nonzero.

$$\boldsymbol{M}_{RB} = \begin{bmatrix} m & -0.003 & 0.001 \\ 0.002 & m & mx_g \\ -0.005 & mx_g & I_z \end{bmatrix}, \quad \boldsymbol{M}_A = -\begin{bmatrix} X_{\dot{u}} & -0.002 & 0.003 \\ -0.003 & Y_{\dot{v}} & Y_{\dot{r}} \\ -0.001 & N_{\dot{v}} & N_{\dot{r}} \end{bmatrix},$$
$$\boldsymbol{D} = -\begin{bmatrix} X_u & -0.004 & -0.002 \\ -0.003 & Y_v & Y_r \\ -0.001 & N_v & N_r \end{bmatrix} \tag{72}$$

The values for $\boldsymbol{M}_A$ and $\boldsymbol{D}$ were chosen to be positive so that the resulting matrix was positive definite. These nonzero values account for errors in the symmetry assumptions that were made previously when deriving the model in section 2.3.4.

The results with only model perturbations are displayed in the plots below in

the same fashion as before. Error plots are split into position and heading error and table 8 shows the standard deviation and absolute maximum errors for the `place` and `rfbg` gain methods.

param	**model**	**simulation**	**difference**
m	23.800	25.800	2.0
I_z	1.760	2.060	0.3
x_g	0.046	0.096	0.05
X_u	-2.0	-1.0	1.0
Y_v	-7.0	-5.0	2.0
Y_r	-0.1	-0.13	-0.03
N_v	-0.1	-0.12	-0.02
N_r	-0.5	-0.7	-0.2
$X_{\dot{u}}$	-2.0	-1.0	1.0
$Y_{\dot{v}}$	-10.0	-7.0	3.0
$Y_{\dot{r}}$	0.0	-0.2	-0.2
$N_{\dot{v}}$	0.0	-0.3	-0.3
$N_{\dot{r}}$	-1.0	-1.8	-0.8
T_{surge}	0.1	0.3	0.2
T_{sway}	0.11	0.41	0.3
T_{yaw}	0.12	0.32	0.2

Table 7: CyberShip II model perturbations (second)

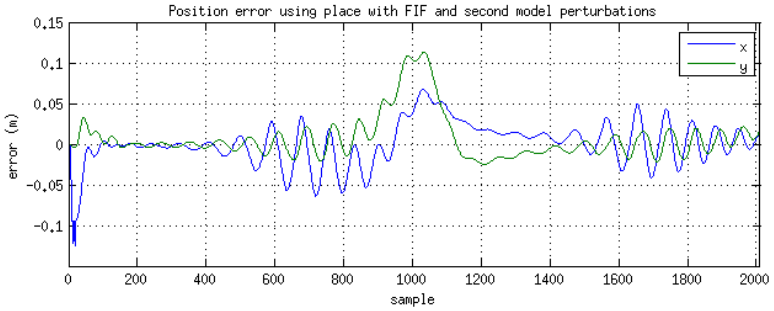
Error with second set of perturbations

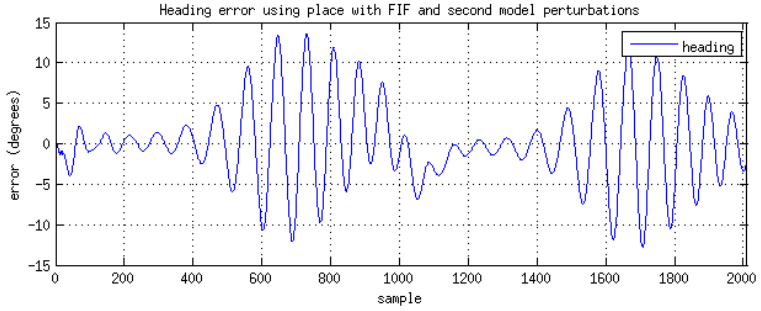
method	**signal**	**std deviation**	**max error**
place	x (m)	0.0266	0.1253
	y (m)	0.0276	0.1146
	heading (deg)	4.9671	13.6058
rfbg	x (m)	0.0232	0.1257
	y (m)	0.0240	0.1005
	heading (deg)	1.0628	4.5007

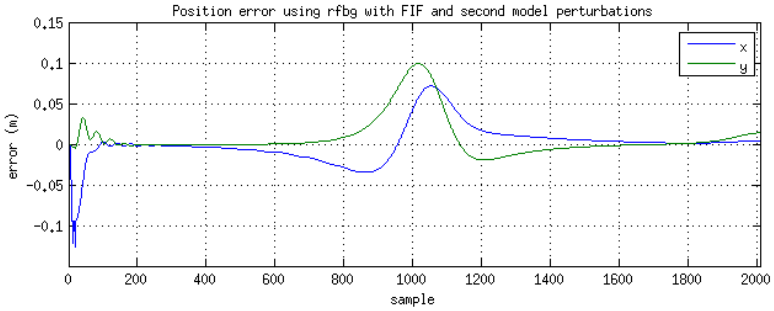
Table 8: CyberShip II errors with second model perturbations

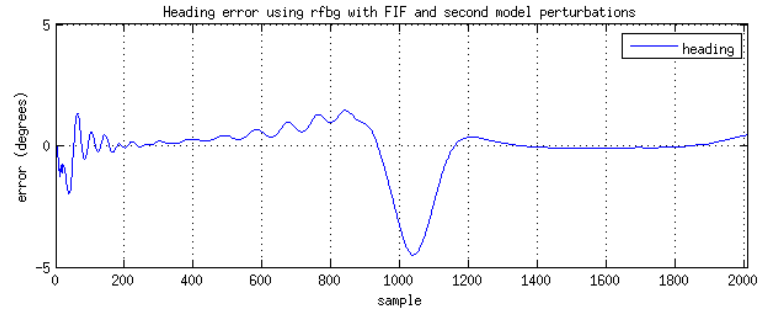
4.3 Analysis

Right away it easy to see that feedforward inverse filter is an absolute necessity when using this DP model and speed. Without the FIF, the controller goes off









path when going around the tighter arc of the ellipse. This is where the yaw rate is at its maximum and the furthest point from the origin. The linear approximation made by converting to vessel parallel coordinates assumes that the boat is close to the origin and has a yaw rate close to 0. The FIF is able to adjust the desired input to the controller so that the marine vehicle never strays too far from the desired

ellipse. With the FIF enabled, the error is still at its maximum when the boat is farthest away from the origin and turning the quickest. Previous state space controllers would have to constrain their speed to where the linearization is more accurate, while the FIF allows for greater maneuvering speed.

Both pole placement methods were able to make it around the entire ellipse without becoming unstable when using the FIF. However, `rfbg` produced gains that had a measurably greater robustness than `place`. The robustness norm was 0.5160 using `rfbg` and 0.2483 using `place`, which is a difference of 0.2677. This difference is much greater than the 0.1 difference that is required to view a measurable difference. This was apparent from the standard deviation results in tables 6 and 8. The controller using robust feedback gains consistently had lower standard deviations and lower maximum errors. When tested under the first set of model perturbations and all disturbances, the maximum position error only rose to 0.1683 meters in the y direction when using `rfbg` and using `place` resulted in more than double that, with 0.5008 meters. With the second set of perturbations, the gains produced by `place` resulted in heading standard deviation of over 4 times greater than `rfbg`.

The most noticeable result from each error plot is the oscillations in each state caused by disturbances and model perturbations. Without any disturbances, the boat will follow the desired trajectory with minimal error as expected. The addition of the second set model perturbations causes oscillations as shown in the second set of plots. The lower the robustness of the designed controller is, the more pronounced the oscillations are. This is likely due to the controller overcompensating for unmodeled perturbations when trying to get back on track. The heading was affected the most by these oscillations, while the position error is relatively small. If the goals for the application vehicle depend highly on heading,

then more care must be taken to estimate parameters accurately. However, for path following, position is more robust to modeling errors and disturbance. Using `rfbg` the error never exceeded 0.17 meters for position, while the heading error never exceeded 14.25 degrees.

The improved robustness of `rfbg` over `place`, when combined with the feed-foward inverse filter, produces acceptable results when simulated on the nonlinear CyberShip II model under disturbances, measurement noise, and significant model perturbations. Robustness can be improved further if maneuvering speed and yaw rate is reduced to where linearizations are more accurate. All theory is applicable to other marine surface vehicles, including URI's ASV.

4.3.1 Improvements and Application to URI's ASV

From these simulations we see that a few things can be done to improve the performance of the linear controller. It is possible to gain schedule with respect to forward speed or yaw rate if higher speeds are desired. A design consideration when path planning could be cap the maximum speed and yaw rate that is sent into the controller to prevent the controller from entering highly nonlinear regions.

As the marine vehicle operates for an extended period of time, it may venture far from the origin where the coordinate change to vessel coordinates is less accurate. To prevent issues, the origin of the controller can be translated to a new point close to the vehicle. When doing this, the states $\boldsymbol{x}_a$ and $\hat{\boldsymbol{x}}$ of the controller would need to be calculated to fit the new origin to avoid large jumps from the controller adjusting. This includes integrators accumulating and the observer settling to the correct state.

The simulation could be improved slightly by adjusting the measurement noise models so that high frequencies are removed by use of a high pass filter. This is more realistic because most sensors have some filters built. Wind noise could also

be made sightly more realistic by adding a noise element or gusts.

URI's ASV currently has two trolling motors, which means that it is under actuated. The controller designed assumes that the vehicle is fully-actuated or over-actuated, so one or more motors would need to be added that produce forces in a different direction that the first two. The current motor controller takes forward thrust and turning commands and mixes them internally using an unknown method. In other words, it is not certain what input commands produces a certain output force. This means the motors would need to be modeled or a new motor controller would be needed.

The current computing power and code structure will allow for the computations needed to implement the modest requirements of this controller. The matrix multiplications are small enough that they will be able to be completed in less than a sampling period $T = 0.1$.

System identification for the ASV will need to be done to obtain approximate parameters for the DP model. This thesis has shown that even if the estimated parameters are not 100% correct, the linear controller is robust enough to handle these perturbations.

List of References

[1] R. J. Vaccaro, "Personal communication," 2012.

CHAPTER 5

Conclusion

Methods for designing a robust linear DP controller for a marine surface vehicle were presented. This included a feedforward inverse filter (FIF) to better track the desired input and a robust feedback gain calculation. Simulation on the nonlinear model of the CyberShip II around an ellipse showed that the FIF was necessary to provide stability at higher speeds. Robust feedback gain calculation improved path tracking in the presence of wind and wave disturbances, measurement noise, and model perturbations when compared to MATLAB's pole placement function. Further improvements can be made by using gain scheduling and translating the control origin so that the boat model remains close to the origin where the linearized transformation to vessel parallel coordinates is valid.

All work done in this thesis can be applied to URI's ASV. The developed controller would be an improvement over the current controller and allow for greater maneuverability. To implement the controller on the ASV, a rough system identification is needed to estimate parameters of the DP model. The ASV must also be modified so that it can produce forces and moments in each direction by adding actuators.

The MATLAB code and Simulink model that was developed during this thesis is applicable to a variety of marine surface vehicles. The code can be modified for other models so that others can test their marine system. The Simulink models were constructed with a modular element so that parts can be enabled and disabled using switches.

To summarize, these results show that linear controllers are sufficient for DP controllers under higher speeds provided that an inverse filter and robust controller

is designed. This widens the accessibility of marine control to those that have knowledge of linear theory, but may have limited knowledge of nonlinear systems.

APPENDIX

Code

A.1 Control Design

```
% Calculates feedback gains for cybership ii using a variety of
    methods
% and models.
%
% author: Thomas DeRensis

addpath('digcontr')
addpath('ele503')
load('sroots.mat');

ul = 0;
%ul = mean(U);
vl = 0;
rl = 0;
%rl = mean(r);
[A, B, C, D, E] = ...
    chooseModel(BoatID.CybershipII, ModelID.
        LinearizedDPThrusters, ul, vl, rl);

T = 0.1;                    % Sampling interval (s)
sampleTime = T;
Ts = 5.0;                     % Time to settle (s)
nStates = length(A);        % num plant states
mStates = 3;                % num measured states

fprintf('Plant has %d states and %d measured states.\n',
    nStates, mStates);
fprintf('Using %.2f sec sampling interval and %.1f sec settling
    time.\n', T, Ts);

% Compute zero order hold equivalent model (similar to c2d)
[phi, gamma] = zohe(A, B, T);

fprintf('Checking Controllability... ')
Wc = ctrb(phi, gamma);
if (rank(Wc) == length(phi))
    fprintf('The system is controllable\n');
else
    fprintf('ERROR: Not controllable!\n');
end
```

```
37
38 fprintf('Checking Observability... ')
39 Wo = obsv(phi, C);
40 if (rank(Wo) == length(phi))
41     fprintf('The system is observable\n');
42 else
43     fprintf('ERROR: Not observable!\n');
44 end
45
46 [spoles, plantMultiplicity] = ...
47     choosePlantPoles(A, Ts, PoleMethod.AddedDamping);
48
49 [phia, gammaa, spoles_ad, addMultiplicity] = ...
50     chooseAdditionalDynamicsPoles(false, true, Ts, T);
51
52 spoles = [spoles_ad spoles]; % Combine desired closed-loop
       poles
53 %spoles = [ms1/Ts ms2/Ts ms4/Ts]%ms3/Ts];
54 zpoles = exp(T*spoles);
55 multiplicity = [addMultiplicity plantMultiplicity];
56
57 if T == 0
58     [L1, L2, delta] = ...
59         choosePlantGains(A, B, C, phia, gammaa, spoles, T, ...
60             GainMethod.RFBG2, multiplicity);
61 else
62     [L1, L2, delta] = ...
63         choosePlantGains(phi, gamma, C, phia, gammaa, zpoles, T
               , ...
64             GainMethod.Place, multiplicity);
65 end
66
67 %% Observer
68 %%%%%%%%%%
69 useObserver = true;
70 K = 0;
71 if useObserver
72     observerSpeed = 4.0;
73     Tso = Ts/observerSpeed;
74     fullOrder = true;
75     opoles = chooseObserverPoles(A, B, C, Tso, ...
76         PoleMethod.PlantZeros, fullOrder, mStates);
77
78     if fullOrder
79         zopoles = exp(T*opoles);
80         if T == 0
```

```
81              K = obg_ts(A, B, C, phia, gammaa, L1, L2, opoles, T
                    );
82              %K = place(A', C', opoles)';
83              delta = robustTSOB(A, B, C, phia, gammaa, L1, L2, K
                    , T);
84          else
85              %K = obg_ts(phi, gamma, C, phia, gammaa, L1, L2,
                    zopoles, T);
86              K = place(phi', C', zopoles)';
87              delta = robustTSOB(phi, gamma, C, phia, gammaa, L1,
                     L2, K, T);
88          end
89      else
90          C1 = eye(mStates);
91          Cm = [C1 zeros(mStates,n-mStates)];
92          zopoles = exp(T*opoles);
93          [F,G,H,K,P]=roo(phi,gamma,Cm,zopoles);
94          L11 = L1(:,1:size(Cm));
95          L12 = L1(:,size(Cm)+1:end);
96          delta = robustTS_RDOB(phi, gamma, Cm, C, phia, gammaa,
                F, G, H, L11, L12, L2, K, T);
97      end
98      fprintf('Robustness with observer: %f\n', delta);
99  end
100
101 if useObserver
102     if T == 0
103         sys = designTSOB(A, B, C, phia, gammaa, L1, L2, K, T);
104     else
105         sys = designTSOB(phi, gamma, C, phia, gammaa, L1, L2, K
                , T);
106     end
107 else
108     if T == 0
109         sys = designTS(A, B, C, phia, gammaa, L1, L2, T);
110     else
111         sys = designTS(phi, gamma, C, phia, gammaa, L1, L2, T);
112     end
113 end
114 S = stepinfo(sys, 'SettlingTimeThreshold', 0.01);
115 Ts0 = S.SettlingTime;
116 fprintf('Final settling time: %f\n', Ts0);
117
118 %%
119 % Feedthrough Inverse Filter
120 if T ~= 0
```

```
121     [A_f, B_f, C_f, D_f] = ...
122         createInverseFeedthrough(phi, gamma, C, phia, gammaa,
                ...
123             L1, L2, K, T, useObserver);
124 end
125
126 fprintf('Done designing control system.\n');
127
128 genVesselTrack
129 ic
```

A.2 Create the Model

```
1  function [A, B, C, D, E] = chooseModel(boatID, modelID, u, v, r
       )
2
3  switch (boatID)
4      case BoatID.CybershipII
5          % Cybership II parameters
6          initCybership
7
8      case BoatID.URI_ASV
9          % TODO: get URI ASV parameters
10         L = 1.8288; % length (m) - 6 feet
11         B = 0.9144;  % width (m) - 3 feet
12         mass = 43.0913; % weigth (kg) - 95 lbs
13
14     otherwise
15         err = MException('chooseModel:InvalidBoatID', 'Invalid
               boat ID %d', boatID);
16         throw(err)
17 end
18
19 switch(modelID)
20     case ModelID.LinearizedDP
21         %%%
22         % Linearized Dynamic Positioning (DP) Model (p157)
23         % Assumptions:
24         %   - low speed
25         %   - bias vector b used to represent quadratic
               velocity terms
26         %   - position represented in vessel parallel
               coordinates
27
28         %   eta_p = transpose(R(phi))*eta
29         %   eta_p_dot = v
30         %   M*v_dot + D*v = transpose(R(phi))*b + tau
31         %   B_dot = 0
```

```
32          %   tau = B*u
33          %%%
34
35          % Form model
36          A = [ zeros(3)     eye(3)    ;
37                zeros(3)  -inv(M)*Damp ];
38          B = [ zeros(3)   ;
39                  inv(M)    ];
40          C = [ eye(3) zeros(3) ]; % output is position
41
42          D = zeros(3);
43
44          E = B;
45      case ModelID.LinearizedDPYaw
46          S = [ 0 1 0;
47               -1 0 0;
48                0 0 0];
49
50          % Form model
51          A = [ r*S    eye(3)    ;
52                zeros(3)  -inv(M)*Damp ];
53          B = [ zeros(3)   ;
54                  inv(M)    ];
55          C = [ eye(3) zeros(3) ]; % output is position
56
57          D = zeros(3);
58
59          E = B;
60      case ModelID.LinearizedManeuvering
61          %%%
62          % Linearized Maneuvering Equation (p131)
63          % Assuptions:
64          %   - restoring forces neglected
65          %   - nonlinear Coriolis and centripetal forces are
                linearized about the
66          %       cruise speed U
67          %   - nonlinear damping is approximated by a linear
                damping matrix D
68          %   - ocean currents neglected
69
70          %   (M_RB + M_A)*v_dot + (C*_RB + C*_A + D)*v = tau
71          %   M*v_dot + N*v = tau
72          %%%
73
74          % Rigid body Coriolis (linearized about U)
75          C_RB = [ 0   0      0      ;
```

```
76                  0   0     mass*U     ;
77                  0   0   mass*x_g*U ];
78          % Added Coriolis (linearized about U)
79          C_A = -[ 0   0      0      ;
80                   0   0   Y_vdot*U   ;
81                   0   0   Y_rdot*U ];
82
83          N = C_RB + C_A + Damp;
84
85          % Form model
86          A = [ zeros(3)     eye(3)    ;
87                zeros(3)   -inv(M)*N ];
88          B = [ zeros(3)   ;
89                  inv(M)     ];
90          C = [ eye(3) zeros(3) ]; % output is position
91
92          D = zeros(3);
93
94          E = B;
95      case ModelID.LinearizedDPThrusters
96          %%%
97          % Linearized Dynamic Positioning (DP) Model with
                thruster (p157)
98          % Assumptions:
99          %   - low speed
100         %   - bias vector b used to represent quadratic
                velocity terms
101         %   - position represented in vessel parallel
                coordinates
102
103         %   eta_p = transpose(R(phi))*eta
104         %   eta_p_dot = v
105         %   M*v_dot + D*v = transpose(R(phi))*b + tau
106         %   B_dot = 0
107         %   tau = B*u
108         %%%
109         fprintf('Linearizing about u = %f, v = %f, r = %f\n', u
                , v, r);
110         S = [ 0 1 0;
111              -1 0 0;
112               0 0 0];
113         % Rigid body Coriolis (linearized about U)
114         %C_RB = [ 0   0      0      ;
115         %         0   0     mass*U     ;
116         %         0   0   mass*x_g*U ];
117         % Added Coriolis (linearized about U)
```

```
118         %C_A = -[ 0   0      0     ;
119         %         0   0   Y_vdot*U   ;
120         %         0   0   Y_rdot*U ];
121         C_RB = [ 0 0 -mass*(x_g*r + v);
122                  0 0 mass*u;
123                  mass*(x_g*r+v) -mass*u 0];
124         C_A = [0 0 Y_vdot*v+Y_rdot*r;
125                0 0 -X_udot*u;
126                -Y_vdot*v-Y_rdot*r X_udot*u 0];
127         N = C_RB + C_A + Damp;
128         % Form model
129         A = [ r*S           eye(3)      zeros(3) ;
130         %A = [ zeros(3) eye(3) zeros(3);
131               zeros(3)  -inv(M)*N      inv(M)    ;
132               zeros(3)    zeros(3)      A_thr    ];
133         B = [ zeros(3)   ;
134               zeros(3)   ;
135                 -A_thr   ];
136         C = [ eye(3) zeros(3) zeros(3) ]; % output is position
137
138         D = zeros(3);
139
140         E = [ zeros(3) ;
141               inv(M)    ;
142               zeros(3) ];
143     otherwise
144         err = MException('InvalidParam', 'Invalid model ID %d',
                modelID);
145         throw(err)
146 end
```

A.3 CyberShip II Parameters

```
1  % Cybership II parameters
2  len = 1.255;      % m (length)
3  width = 0.29;     % m (width)
4  mass = 23.800;    % kg (mass)
5  I_z = 1.760;      % kgm^2
6  x_g = 0.046;      % m
7
8  X_udot = -2.0;    % kg
9  Y_vdot = -10.0;   % kg
10 Y_rdot = 0.0;     % kgm
11 N_vdot = 0.0;     % kgm
12 N_rdot = -1.0;    % kgm^2
13
14 % Use these for simulation
15 X_u = -0.72253;  % kg/s
```

```
Y_v = -0.88965;  % kg/s
Y_r = -7.250;    % kgm/s
N_v = 0.03130;   % kgm/s
N_r = -1.900;    % kgm/s

% Use these for design
X_u = -2; % from Fossen simulink
Y_v = -7; % from Fossen simulink
Y_r = -0.1; % from Fossen simulink
N_v = -0.1; % from Fossen simulink
N_r = -0.5; % from Fossen simulink

X_uu = -1.32742; % kg/m
X_uuu = -5.8664; % kgs/m^2
Y_vv = -36.4729; % kg/m
Y_rv = -0.805;   % kg
Y_vr = -0.845;   % kg
Y_rr = -3.450;   % kgm
N_vv = 3.95645;  % kg
N_rv = 0.130;    % kgm
N_vr = 0.080;    % kgm
N_rr = -0.750;   % kgm



% Rigid body mass
M_RB = [ mass     0       0     ;
         0      mass     mass*x_g   ;
         0   mass*x_g     I_z   ];
% Added mass - strictly positive
M_A = -[ X_udot      0       0      ;
           0       Y_vdot   Y_rdot   ;
           0       Y_rdot   N_rdot ];

M = M_RB + M_A;

% Damping (linear) - strictly positive
Damp = -[ X_u    0     0    ;
            0    Y_v   Y_r   ;
            0    N_v   N_r ];
% Time constants (about 0.5 second settling time)
T_surge = 0.1;
T_sway = 0.11;
T_yaw = 0.12;

A_thr = -[  1/T_surge      0            0     ;
```

```
                0          1/T_sway          0      ;
                0             0          1/T_yaw ];
```

A.4 CyberShip II Perturbation

```
% Cybership II parameters
len = 1.255;      % m (length)
width = 0.29;     % m (width)
mass = 23.800 + 4;   % kg (mass)
I_z = 1.760 + 0.4;      % kgm^2
x_g = 0.046 + 0.2;      % m

X_udot = -2.0 + 1;   % kg
Y_vdot = -10.0 + 3;  % kg
Y_rdot = 0.0 - 1;    % kgm
N_vdot = 0.0 - 1.2;     % kgm
N_rdot = -1.0 - 0.8;   % kgm^2

% Use for simulation
X_u = -0.72253;  % kg/s
Y_v = -0.88965;  % kg/s
Y_r = -7.250;    % kgm/s
N_v = 0.03130;   % kgm/s
N_r = -1.900;    % kgm/s

X_uu = -1.32742; % kg/m
X_uuu = -5.8664; % kgs/m^2
Y_vv = -36.4729; % kg/m
Y_rv = -0.805;   % kg
Y_vr = -0.845;   % kg
Y_rr = -3.450;   % kgm
N_vv = 3.95645;  % kg
N_rv = 0.130;    % kgm
N_vr = 0.080;    % kgm
N_rr = -0.750;   % kgm

% Rigid body mass
M_RB = [ mass     0       0     ;
           0     mass    mass*x_g   ;
           0   mass*x_g    I_z   ];
% Added mass - strictly positive
M_A = -[ X_udot      0       0      ;
            0      Y_vdot   Y_rdot   ;
            0      N_vdot   N_rdot ];

M = M_RB + M_A;

% Damping (linear) - strictly positive
```

```
Damp = -[ X_u   0     0    ;
          0     Y_v   Y_r  ;
          0     N_v   N_r ];
% Time constants
T_surge = 0.1+0.2;
T_sway = 0.11+0.3;
T_yaw = 0.12+0.2;

A_thr = -[  1/T_surge      0            0     ;
            0          1/T_sway         0     ;
            0              0        1/T_yaw ];
```

A.5 CyberShip II Perturbation 2

```
% Cybership II parameters
len = 1.255;      % m (length)
width = 0.29;     % m (width)
mass = 23.800 + 2;   % kg (mass)
I_z = 1.760 + 0.3;     % kgm^2
x_g = 0.046 + 0.05;     % m

X_udot = -2.0 + 1;   % kg
Y_vdot = -10.0 + 3;  % kg
Y_rdot = 0.0 - 0.2;     % kgm
N_vdot = 0.0 - 0.3;     % kgm
N_rdot = -1.0 - 0.8;    % kgm^2

% Use for simulation
%X_u = -0.72253;  % kg/s
%Y_v = -0.88965;  % kg/s
%Y_r = -7.250;    % kgm/s
%N_v = 0.03130;   % kgm/s
%N_r = -1.900;    % kgm/s

% Use these for design
X_u = -2 +1; % from Fossen simulink
Y_v = -7+2; % from Fossen simulink
Y_r = -0.1-0.03; % from Fossen simulink
N_v = -0.1-0.02; % from Fossen simulink
N_r = -0.5-0.2; % from Fossen simulink

X_uu = -1.32742; % kg/m
X_uuu = -5.8664; % kgs/m^2
Y_vv = -36.4729; % kg/m
Y_rv = -0.805;   % kg
Y_vr = -0.845;   % kg
Y_rr = -3.450;   % kgm
N_vv = 3.95645;  % kg
```

```
N_rv = 0.130;     % kgm
N_vr = 0.080;     % kgm
N_rr = -0.750;    % kgm

% Rigid body mass
M_RB = [ mass      -0.003       0.001     ;
         0.002      mass      mass*x_g    ;
         -0.005   mass*x_g      I_z     ];
% Added mass - strictly positive
M_A = -[ X_udot      -0.002       -0.003      ;
           -0.003      Y_vdot   Y_rdot    ;
           -0.001      N_vdot   N_rdot  ];

M = M_RB + M_A;

% Damping (linear) - strictly positive
Damp = -[ X_u    -0.004      -0.002     ;
            -0.003    Y_v   Y_r    ;
            -0.001    N_v   N_r  ];
% Time constants
T_surge = 0.1+0.2;
T_sway = 0.11+0.3;
T_yaw = 0.12+0.2;

A_thr = -[  1/T_surge      0            0      ;
                0        1/T_sway       0      ;
                0           0        1/T_yaw  ];
```

A.6 Choose Plant Poles

```
function [spoles, multiplicity] = choosePlantPoles(A, Ts,
    poleMethod)

addpath('digcontr');
addpath('ele503');
load('sroots.mat');

n = length(A);

fprintf('Choosing plant poles... ');

switch poleMethod;
    case PoleMethod.Bessel
        fprintf('Using Bessel poles s%d\n', n)
        spoles = get_spoles(n)/Ts;
    case PoleMethod.BesselOptimal
        fprintf('Using optimal Bessel poles\n')
        spoles = getOptimalBesselPoles(n, Ts);
```

```
18      case PoleMethod.SufficientlyDamped
19          fprintf('Using sufficiently damped poles\n')
20          sdp = getSufficientlyDampedPoles(A, Ts);
21          num_sdp = length(sdp);
22          fprintf('Found %d sufficiently damped poles\n', num_sdp
                );
23          spoles = [sdp getOptimalBesselPoles(n - num_sdp)];
24      case PoleMethod.AddedDamping
25          fprintf('Using added damping poles\n')
26          add = getAddedDampingPoles(A, Ts);
27          num_add = length(add);
28          fprintf('Found %d added damping poles\n', num_add);
29          sdp = getSufficientlyDampedPoles(A, Ts);
30          num_sdp = length(sdp);
31          [u,~,idx] = unique(sdp);
32          multiplicity = accumarray(idx(:),1,[],@sum)';
33          sdp = u;
34          fprintf('Found %d sufficiently damped poles\n', num_sdp
                );
35          spoles = [add sdp getOptimalBesselPoles(n - num_add -
                num_sdp, Ts)];
36      otherwise
37          fprintf('ERROR: Unimplemented pole selection method: %d
                \n', ...
38              poleSelectionMethod);
39          return
40  end
41
42  multiplicity = [multiplicity ones(1,n-sum(multiplicity))];
```

A.7 Choose Additional Dynamics Poles

```
1   function [Aa, Ba, spoles, multiplicity] =
        chooseAdditionalDynamicsPoles(waves, repeat, Ts, T)
2
3   load('sroots.mat');
4
5   fprintf('Choosing additional dynamics poles... ');
6
7   if waves
8       fprintf('Canceling waves... ');
9       % wave disturbance
10      lambda = 0.1;
11      w_0 = 0.8;
12      spoles_w1 = roots([1 2*lambda*w_0 w_0*w_0]);
13      spoles_w2 = roots([1 2*lambda*w_0 w_0*w_0]) - 0.0001;
14      spoles_w3 = roots([1 2*lambda*w_0 w_0*w_0]) + 0.0001;
15      zpoles_w1 = exp(T*spoles_w1);
```

```
16
17      alphad1 = real(zpoles_w1(1));
18      betad1 = imag(zpoles_w1(1));
19      phia1 = [ 1 0 0; 0 alphad1 betad1; 0 -betad1 alphad1];
20      gammaa1 = [1 1 1]';
21      Aa = kron(eye(3), phia1);
22      Ba = kron(eye(3), gammaa1);
23
24      if real(spoles_w1(1)) > s1/Ts
25          fprintf('Added damping poles... ')
26          spoles_w1 = s1/Ts + 1j*imag(spoles_w1);
27          spoles_w2 = s1/Ts + 1j*imag(spoles_w2) - 0.0001;
28          spoles_w3 = s1/Ts + 1j*imag(spoles_w3) + 0.0001;
29      else
30          disp('Sufficiently damped poles... ')
31      end
32
33      % Use wave poles and bessel
34      if repeat
35          fprintf('Repeated poles\n');
36          spoles = [s1/Ts spoles_w1'];
37          multiplicity = [3 3 3];
38      else
39          fprintf('Unique poles\n');
40          spoles = [spoles_w1' spoles_w2' spoles_w3' s2/Ts s1/Ts
                ];
41      end
42  else
43      if T == 0
44          Aa = zeros(3);
45          Ba = eye(3);
46      else
47          Aa = eye(3);
48          Ba = eye(3);
49      end
50      if repeat
51          fprintf('Using repeated Bessel s1/Ts pole\n');
52          spoles = s1/Ts;
53          multiplicity = 3;
54      else
55          fprintf('Using Bessel s1 s2 poles\n');
56          spoles = [s1/Ts s2/Ts];
57      end
58  end
59
60  sv = svd([Ba Aa*Ba Aa*Aa*Ba]);
```

```
61 fprintf('Ratio of singular values (lower is better): %f\n', sv
   (1)/sv(end));
```

A.8 Calculate Tracking System Gains

```
1  function [L1, L2, delta] = ...
2      choosePlantGains(A, B, C, Aa, Ba, poles, T, gainMethod,
          multiplicity)
3
4  fprintf('Choosing feedback gains... ');
5
6  switch gainMethod
7      case GainMethod.Place
8          fprintf('Using place\n');
9          [L1, L2, delta] = ...
10             mts(A, B, C, Aa, Ba, poles, multiplicity, T, 'place
                   ');
11     case GainMethod.FBG
12         fprintf('Using fbg\n');
13         [L1, L2, delta] = ...
14             mts(A, B, C, Aa, Ba, poles, multiplicity, T, 'fbg')
                   ;
15     case GainMethod.RFBG
16         fprintf('Using rfbg\n');
17         [L1, L2, delta] = ...
18             mts(A, B, C, Aa, Ba, poles, multiplicity, T, 'rfbg'
                   );
19     case GainMethod.RFBG2
20         fprintf('Using rfbg2\n');
21         [L1, L2, delta] = ...
22             mts(A, B, C, Aa, Ba, poles, multiplicity, T, 'rfbg2
                   ');
23     otherwise
24         fprintf('ERROR: Unimplemented feedback gain method: %d\
              n', ...
25             gainMethod);
26         return
27 end
28
29 fprintf('Full-state feedback robustness norm: %f\n', delta);
```

A.9 Create Feedforward Inverse Filter

```
1  function [A_f, B_f, C_f, D_f] = ...
2      createInverseFeedthrough(A, B, C, Aa, Ba, L1, L2, K, T, ...
3          useObservers)
4
5  % Full design model
6  if useObservers
```

```
7       sys = designTSOB(A, B, C, Aa, Ba, L1, L2, K, T);
8   else
9       sys = designTS(A, B, C, Aa, Ba, L1, L2, T);
10  end
11  A_d = sys.A;
12  B_d = sys.B;
13  C_d = sys.C;
14  D_d = sys.D;
15
16  fprintf('Checking zeros of the design model\n');
17  tzeros = tzero(A_d, B_d, C_d, D_d);
18  for tz = tzeros'
19      if abs(tz) > 1
20          fprintf('WARNING: %.4f + %.4fi is outside the unit
                circle\n', ...
21              real(tz), imag(tz));
22      end
23  end
24
25  fprintf('Calculating inverse feedfoward filter\n');
26
27  if T == 0
28      % Continuous
29      for d = 1:10
30          condition = cond(C_d*A_d^(d-1)*B_d);
31          if (condition >0 && condition ~= Inf)
32              fprintf('delay = %d cond = %f\n',d,condition);
33              %break;
34          end
35          % Calculate inverse feedthrough
36          D_f = inv(C_d*A_d^(d-1)*B_d);
37          C_f = -D_f*C_d*A_d^d;
38          B_f = B_d*D_f;
39          A_f = A_d + B_d*C_f;
40          stable = 1;
41          for la = eig(A_f)'
42              if abs(la) > 1
43                  stable = 0;
44              end
45          end
46          if stable == 1
47              break;
48          end
49      end
50  else
51      for d = 2:100
```

```
52          condition = cond(C_d*A_d^(d-1)*B_d);
53          if (condition >0 && condition ~= Inf)
54              fprintf('delay = %d cond = %f\n',d,condition);
55          end
56          % Calculate inverse feedthrough
57          D_f = inv(C_d*A_d^(d-1)*B_d);
58          C_f = -D_f*C_d*A_d^d;
59          B_f = B_d*D_f;
60          A_f = A_d + B_d*C_f;
61          stable = true;
62          eig(A_f);
63          for e = eig(A_f)'
64              if abs(e) >= 1
65                  stable = false;
66              end
67          end
68          if stable == true
69              break;
70          end
71      end
72  end
```

A.10 FIF Initial Conditions

```
1  %load inverse_filter
2  %load ellipse
3  psiIn(end)=2*pi;
4  ref=[northIn eastIn psiIn];
5  ref1=[ref ;ref(end:-1:1,:);ref];
6  t1=[t;t+t(end)+T];
7  t1=[t1;t+t1(end)+T];
8  insys=ss(A_f,B_f,C_f,D_f,T);
9  [y1,t2,x]=lsim(insys,ref1,t1,zeros(length(A_f),1));
10 xf0=-x(2*length(psiIn),:)';
11 [y,t3,x3]=lsim(insys,ref,t,xf0);
12 sig1=[t ref(:,1)];
13 sig2=[t ref(:,2)];
14 sig3=[t ref(:,3)];
```

A.11 TSOB Design Model

```
1  function sys = designTSOB(A, B, C, Aa, Ba, L1, L2, K, T)
2  % sys = designTSOB(A, B, C, Aa, Ba, L1, L2, K, T)
3  %
4  % Calculates the design model for the tracking system with full
       order
5  % observer.
6  %
7  % INPUT
```

```
%   A, B, C      Plant Model
%   Aa, Ba       Additional Dynamics model
%   L1           Plant gains
%   L2           Additional dynamics gains
%   K            Observer gains
%   T            Sampling time
% OUPUT
%   sys          The design model
%
% AUTHOR: tderensis

[n, m] = size(B);
q = length(Aa);

A_d = [  A          -B*L1          B*L2;
         K*C    (A-K*C-B*L1)      B*L2;
         -Ba*C   zeros(q,n)        Aa   ];
B_d = [ zeros(2*n, m);
         Ba ];
C_d = [C zeros(m,n+q)];
D_d = zeros(m);
sys = ss(A_d, B_d, C_d, D_d, T);
```

A.12 TSOB Robust Model

```
function [delta] = robustTSOB(A, B, C, Aa, Ba, L1, L2, K, T)
% [delta] = robustTS(A, B, C, Aa, Ba, L1, L2, T)
% INPUT
%   A, B, C - Plant matrices
%   Aa, Ba  - Additional dynamics matrices
%   L1      - Plant gains
%   L2      - Additional dynamics gains
%   K       - Observer gain
%   T       - Sampling Time
% OUTPUT
%   delta   - Multivariable robustness norm
%
% Calculates the multivariable robustness norm of the system
    using the
% input pertubation model.
%
% author: tderensis

[n,p]=size(B);
[q,m]=size(Ba);

AA = [  A          -B*L1              B*L2  ;
        K*C      (A-K*C-B*L1)         B*L2  ;
```

```
        -Ba*C     zeros(q,n)        Aa    ];

% Check for stability
for i = eig(AA)'
    if abs(i) > 1
        delta = 0;
        return
    end
end

BB = [  B ;
        zeros(n,p);
        zeros(q,m);];

CC = [zeros(p,n) -L1 L2];

sys=ss(AA,BB,CC,zeros(p,p),T);
delta=1/norm(sys,inf);

return
```

A.13 Generate Ellipse

```
% Create a vessel track

simTime = 200;
t = [0:T:simTime]';
scale = (2.0*pi/simTime);

% Elipse
a = 20.0;
b = 10.0;
eastIn = a-a*cos(t*scale);
dEastIn = a*scale*sin(t*scale);
ddEastIn = a*scale^2*cos(t*scale);
northIn = b*sin(t*scale);
dNorthIn = b*scale*cos(t*scale);
ddNorthIn = -b*scale^2*sin(t*scale);
U = sqrt(dEastIn.^2 + dNorthIn.^2);
dU = (scale^2*(a^2-b^2)*sin(2*t*scale))./(2*sqrt((a*sin(t*scale
    )).^2 + (b*cos(t*scale)).^2));
%dU = sqrt(ddEastIn.^2 + ddNorthIn.^2);

psiIn = atan2(-dEastIn, -dNorthIn)+pi;
psiIn(end) = 0;
%psiIn = atan2(a*tan(t*(2*pi/simTime)),b)*180/pi;
%psiIn(floor(end/4+1):floor(3*end/4+1)) = psiIn(floor(end/4+1):
    floor(3*end/4+1)) + 180;
```

```
%psiIn = mod(psiIn, 360);

% yawRate
r = (a*b*scale)./(a^2*sin(scale*t).^2+b^2*cos(scale*t).^2);

ra = (2*a*b*scale^2*(b^2-a^2)*sin(2*scale*t))./(a^2*sin(scale*t
    ).^2+b^2*cos(scale*t).^2).^2;

vesselTrack.time = t;
vesselTrack.signals.values = [northIn, eastIn, psiIn];
vesselTrack.signals.dimensions = 3;
```

A.14 Initial Conditions

```
% initial conditions
ang = psiIn(1);
eta0 = [ northIn(1) eastIn(1) psiIn(1) ]';
nu0 = [ dNorthIn(1) dEastIn(1) r(1) ]';
nu_dot0 = [ddNorthIn(1) ddEastIn(1) ra(1)]';
u0 = M*nu_dot0+Damp*nu0;
tau0 = -M*nu_dot0 + Damp*nu0;
dtau0 = A_thr*(tau0-u0);
x0 = [ eta0' nu0' tau0']';
%x0 = [ northIn(1) eastIn(1) ang 0 0 0]';
rot = [ cos(ang) -sin(ang) 0;
        sin(ang) cos(ang) 0;
        0 0 1];
m = zeros(3, length(psiIn));
for i = 1:length(psiIn)
    rot = [ cos(psiIn(i)) -sin(psiIn(i)) 0;
            sin(psiIn(i)) cos(psiIn(i)) 0;
            0 0 1];
    m(:,i) = rot*[northIn(i); eastIn(i); psiIn(i)];
end
% rotate to vessel parallel coordinates
x0(1:3) = rot'*x0(1:3);

% initial additional dynamics (controller must be designed
    first)
xa0 = L2'*inv(L2*L2')*(L1*x0+u0);
xo0 = x0;
inverse_IC
```

A.15 Boat ID

```
classdef BoatID
   enumeration
      CybershipII, URI_ASV
   end
```

```
end
```

A.16 Model ID

```
classdef ModelID
    enumeration
        LinearizedDP, LinearizedDPYaw, LinearizedManeuvering,
        LinearizedDPThrusters
    end
end
```

A.17 Gain Selection Method

```
classdef GainMethod
    enumeration
        Place, FBG, RFBG, RFBG2
    end
end
```

A.18 Pole Selection Method

```
classdef PoleMethod
    enumeration
        Bessel, BesselOptimal, SufficientlyDamped, AddedDamping,
            PlantZeros
    end
end
```

BIBLIOGRAPHY

Aasen, R. and Hays, B., "Method for Finding Min and Max Values of Error Range for Calculation of Moment of Inertia," in *69th Annual Conference Of Society of Allied Weight Engineers, Inc*, no. 3504, May 2010.

Azarsina, F. and Williams, C. D., "Nomoto Indices for Constant-Depth Zigzag Manoeuvres of an Autonomous Underwater Vehicle," *ISRN Oceanography*, vol. 2013, Article ID 219545, doi:10.5402/2013/219545, 2013.

Burl, J. B., *Linear Optimal Control*. Addison Wesley Longman, Inc., 1999.

Committee of 23rd ITTC, M., *ITTC - Recommended Procedures*, 2002.

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

Journée, J., "A Simple Method for Determining the Manoeuvring Indices K and T from Zigzag Trial Data," Delft University of Technology, Ship Hydromechanics Laboratory, Report 267, 1970.

Marquez, H. J., *Nonlinear Control Systems*. John Wiley & Sons, Inc., 2003.

Norwegian University of Science and Technology. "MSS. Marine Systems Simulator." 2010. [Online]. Available: http://www.marinecontrol.org

Skjetne, R., "The Maneuvering Problem," Phd Dissertation, Norwegian University of Science and Technology, 2005.

Skjetne, R., Smogeli, Ø. N., and Fossen, T. I., "A Nonlinear Ship Manoeuvering Model: Identification and adaptive control with experiments for a model ship," *Modeling, Identification and Control*, vol. 25, no. 1, pp. 3–27, 2004.

Sørensen, A. J., Sagatun, S. I., and Fossen, T. I., "Design of a Dynamic Positioning System Using Model-Based Control," *Control Engineering Practice*, vol. 4, no. 3, pp. 359–368, 1996.

Strand, J. P., "Nonlinear Position Control Systems Design for Marine Vessels," Ph.D. dissertation, Norwegian University of Science and Technology, 1999.

Strand, J. P., Ezal, K., Fossen, T. I., and Kokotović, P. V., "Nonlinear Control of Ships: A Locally Optimal Design," in *4th IFAC nonlinear control systems design symposium*, 1998, pp. 732–737.

Tzang, C.-Y. and Chen, J.-F., "Fundamental Properties of Linear Ship Steering Dynamic Models," *Journal of Marine Science and Technology*, vol. 7, no. 2, pp. 79–88, 1999.

Vaccaro, R. J., *Digital Control: A State-Space Approach*. McGraw-Hill, Inc., 1995.

Vaccaro, R. J., "Digital control library," 2012.

Vaccaro, R. J., "Personal communication," 2012.

Vaccaro, R. J., "Rules for selecting poles," 2012, personal communication.

Yoon, H. K. and Rhee, K. P., "Identification of hydrodynamic coefficients in ship maneuvering equations of motion by estimation-before-modeling technique," *Ocean Engineering*, vol. 30, p. 23792404, 2003.