Dynamics of an Underwater Drifting Instrument: System Modeling and Controller Design for the μFloat

Jessica Noe

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Mechanical Engineering

University of Washington

2020

Reading Committee:

Brian Polagye, Chair

Benjamin Maurer

Sawyer B. Fuller

Program Authorized to Offer Degree:
Department of Mechanical Engineering

University of Washington

Abstract

Dynamics of an Underwater Drifting Instrument:
System Modeling and Controller Design for the μFloat

Jessica Noe

Chair of the Supervisory Committee:
Associate Professor Brian Polagye
Department of Mechanical Engineering

This thesis explores optimal depth control for the μFloat, an affordable oceanographic float designed for swarm sensing in coastal waters. The current Proportional-Integral-Derivative (PID) control system combined with a Butterworth filter on the pressure sensor data enables constant depth and constant velocity deployments, but float trajectories could be improved and power use could be reduced. Ship autopilots and autonomous underwater vehicles have made use of the optimal, model-based Linear Quadratic Gaussian (LQG) control method, which combines an Extended Kalman filter (EKF) and a Linear Quadratic Regulator (LQR) to create a stable, optimal feedback control system. Nonlinear and linear models of the μFloat dynamic system were developed and used with an LQG controller that included a nonlinear EKF state estimator and the LQR feedback controller. The EKF estimator delivered smoothed depth and velocity signals without the delay of the Butterworth filter. The simulated LQG controller was able to perform constant depth and constant velocity deployments with more efficient actuation than PID control, but currently has greater overshoot. The dynamic model developed in this thesis and the LQG controller elements could assist with ongoing development of the μFloat, and other buoyancy-driven floats.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Brian Polagye and the members of the Marine Renewable Energy Lab at the University of Washington for all of their support, encouragement, mentoring, and friendship. In particular, I would to thank Trevor Harrison, Corey Crisp, Dr. James Joslin, and Cassie Riel for the opportunity to work with them on the μFloat drifting swarm project.

The μFloat project is supported by the US Department of Defense Naval Facilities Engineering Command under N0002410D6318 / N0002418F8702.

Trevor Harrison and Corey Crisp created the existing μFloat code, including the Butterworth filter, PD controller, onboard processing code, data import code, and numerical derivative code.

Experimental data was collected by Trevor Harrison and Jessica Noe, with support from Captain Andrew Reay-Ellers and David Nichols.

DEDICATION

To the people who have believed in and supported me, and thereby changed the trajectory of my life for the better.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Table 1: Symbols, Model Variables

Symbol	**Value**	**Units**	**Description**
z	variable	m	Float Depth
$\dot{z}$	variable	m/s	Float Velocity
$\ddot{z}$	variable	m/s^2	Float Acceleration
θ	variable	$rotos$	Piston Angular Displacement
$\dot{\theta}$	variable	$rotos/s$	Piston Angular Velocity
s_m	variable	$motos$	Motor Speed Command

Table 2: Symbols, Model Forces

Symbol	**Units**	**Description**
F_w	N	Force of Weight
F_b	N	Force of Buoyancy
F_d	N	Force of Drag
F_a	N	Force of Added Mass

Table 3: Symbols, Model Physical Properties

Symbol	**Value**	**Units**	**Description**
m_f	4.90	kg	Float Mass in Air
V_f	varies	m^3	Float Volume
V_N	m_f/ρ	m^3	Float Neutral Buoyancy Volume
ΔV	varies	m^3	Float Volume Change
θ_N	varies by float	$rotos$	Neutral Buoyancy Piston Position
A_{cs}	0.0115	m^2	Float Cross-sectional Area
r_p	0.0301	m	Radius of Piston
C_d	0.85	unitless	Coefficient of Drag
C_a	1.0	unitless	Coefficient of Added Mass
F_{conv}	4.43x10^{-5}	$cm/roto$	Conversion Factor, Piston Motion
q_1	2.0559	$rotos/(motos\ sec)$	Piston Speed Command to Rotation
q_2	82.657	$rotos/sec$	Piston Speed Command to Rotation
dt_θ	0.100	sec	μFloat Control Loop Time Step

Table 4: Symbols, Environmental Properties

Symbol	**Value**	**Units**	**Description**
P	varies	$dbar$, Pa	Water Pressure
ρ	$1,000$	kg/m^3	Water Density
g	9.81	m/s^2	Gravitational Acceleration

Table 5: Symbols, Extended Kalman Filter Estimator, Model and Variables

Symbol	**Size**	**Units**	**Description**
t		s	Time Vector
dt	1x1	s	Time Step for Looping
$\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$	3x1	$[m/s;\ m/s^2;\ rotos/s]$	Vector of NL Model Functions
$F(t)$	3x3	multiple	$F(t) \equiv \partial\mathbf{f}/\partial\mathbf{x}$ evaluated at $\hat{\mathbf{x}}(t), \mathbf{u}(t)$
$\mathbf{h}(\mathbf{x}_k)$	2x1	$[Pa;\ rotos]$	Vector of Output Functions
$H_k(\hat{\mathbf{x}}_k^-)$	2x3	multiple	$H_k(\hat{\mathbf{x}}_k^-) \equiv \partial\mathbf{h}/\partial\mathbf{x}$ evaluated at $\hat{\mathbf{x}}_k^-$
$\mathbf{x}(t)$	3x1	$[m;\ m/s;\ rotos]$	State Vector, True
$\dot{\mathbf{x}}(t)$	3x1	$[m/s;\ m/s^2;\ rotos/s]$	State Vector Deriv, True
$\hat{\mathbf{x}}(t)$	3x1	$[m;\ m/s;\ rotos]$	State Vector, Est'd
$\dot{\hat{\mathbf{x}}}(t)$	3x1	$[m/s;\ m/s^2;\ rotos/s]$	State Vector Deriv, Est'd
$\mathbf{x}_k$	3x1	$[m;\ m/s;\ rotos]$	State Vector, True, Discrete
$\hat{\mathbf{x}}_k^-$	3x1	$[m;\ m/s;\ rotos]$	State Vector, Est'd, Disc, Before Update
$\hat{\mathbf{x}}_k^+$	3x1	$[m;\ m/s;\ rotos]$	State Vector, Est'd, Disc, After Update
$\mathbf{u}(t)$	1x1	$motos$	Motor Speed Command

Table 6: Symbols, Extended Kalman Filter Estimator, Continued

Symbol	**Size**	**Units**	**Description**
$G(t)$	3x3	unitless	Identity Matrix
$\mathbf{w}(t)$	3x1	$[m; m/s; rotos]$	Process Noise Vector
$Q(t)$	3x3	$[m; m/s; rotos]$	EKF Tuning Knob, Diag Matrix
$\tilde{\mathbf{y}}_k$	2x1	$[Pa; rotos]$	Measurement Vector
$\mathbf{v}_k$	2x1	$[Pa; rotos]$	Measurement Noise Vector
R_k	2x2	$[Pa; rotos]$	Std Dev of Meas Noise, Diag Matrix
$P(t)$	3x3	$[m; m/s; rotos]$	Covariance Diag Matrix, True
$\dot{P}(t)$	3x3	$[m/s; m/s^2; rotos/s]$	Covariance Diag Matrix Deriv, True
P_k^-	3x3	$[m; m/s; rotos]$	Covariance Diag Matrix, Before Update
P_k^+	3x3	$[m; m/s; rotos]$	Covariance Diag Matrix, After Update
K_k	3x2	multiple	Gain Matrix

Table 7: Symbols, PID Control

Symbol	**Size**	**Units**	**Description**
K_{p1}	1x1	1/s	Proportional Gain for "Depth" loop
K_{d1}	1x1	unitless	Derivative Gain for "Depth" loop
P	1x1	dbar	Filtered Pressure Signal (Depth proxy)
P_{ref}	1x1	dbar	Desired Pressure (Depth proxy)
Err_1	1x1	dbar	Error in Pressure Signal (Depth Error Proxy)
K_{p2}	1x1	(s motos)/dbar	Proportional Gain for "Velocity" loop
K_{d2}	1x1	(s^2 motos)/dbar	Derivative Gain for "Velocity" loop
dP/dt	1x1	dbar/s	Pressure Deriv Signal (Float Velocity proxy)
dP/dt_{ref}	1x1	dbar/s	Desired Pressure Deriv Value (Vel proxy)
Err_2	1x1	dbar/s	Error in dP/dt (Velocity Error Proxy)
s_m	1x1	motos	Control Signal, Motor Speed

Table 8: Symbols, LQG Control

Symbol	**Size**	**Units**	**Description**
$\mathbf{x}$	3x1	$[m;\ m/s;\ rotos]$	State Vector, True
$\dot{\mathbf{x}}$	3x1	$[m/s;\ m/s^2;\ rotos/s]$	State Vector Deriv, True
A	3x3	$[1/s;\ 1/s;\ 1/s]$	State Space "A" matrix
B	3x1	$[1/(s\ motos);\ 1/(s\ motos);\ 1/(s\ motos)]$	State Space "B" matrix
$\hat{\mathbf{x}}$	3x1	$[m;\ m/s;\ rotos]$	State Vector, Est'd
$\hat{\mathbf{x}}_{ref}$	3x1	$[m;\ m/s;\ rotos]$	Desired State Vector
$\hat{\mathbf{x}}_{err}$	3x1	$[m;\ m/s;\ rotos]$	State Vector Error
K	1x3	$[motos/m;\ (s\ motos)/m;\ motos/rotos]$	Feedback Control Gain
u	1x1	$motos$	Control Signal, Motor Speed
$\tilde{\mathbf{y}}_k$	2x1	$[Pa; rotos]$	Measurement Vector
$\tilde{\mathbf{P}}_k$	1x1	Pa	Pressure Measurement
$\tilde{\theta}_k$	1x1	$rotos$	Rotary Encoder Meas.
J	1x1	unitless	Cost Function
Q	3x3	unitless	Weighting of State Error
R	2x2	unitless	Weighting of Actuation Cost

Chapter 1
INTRODUCTION

1.1 Background

Underwater, profiling, drifting instrument packages ("floats") have become an important tool for oceanographic measurements. The best known example is the ARGO program, which has vastly expanded our understanding of the ocean's physical properties [13]. These are, however, generally expensive ($\sim$ \$20k) instrument packages designed for measurements in the deep ocean (i.e., operating depths $>$ 1000 m). Fewer float options exist for shallower, coastal waters. Swarm sensing using relatively large numbers of instrument packages (e.g., $>$ 20) in a relatively small area (e.g., $<$ 1 km^2) is a rapidly growing sector of oceanographic measurements. Swarm sensing missions benefit from relatively inexpensive instrumentation, such that the current generation of expensive, deep water floats are not well-suited to such operations.

1.2 μFloat Specifications

The μFloat (pronounced "micro-float") was designed to meet the intersection of these two needs - an inexpensive ($\sim$ \$2k), shallow water ($<$ 100 m) drifter [9]. It has a single degree of freedom to control its vertical position in the water column, changing its buoyancy by retracting a solid, cylindrical piston into the float body to decrease the float buoyancy and allow the float to descend vertically through the water. This piston is connected by a lead screw with 4.72 threads per cm (12 threads per inch) to an 84 RPM DC motor with planetary gearbox and integrated rotary encoder. The rotary encoder is the first of two important sensors used for the μFloat controller. The rotary encoder measures piston retraction into

the μFloat in units of "rotos", which are a measurement of angular displacement at 4,776.4 counts per revolution, or 2.2566 x 10^6 rotos per linear m (57,316.6 rotos per linear inch) of piston retraction [14].

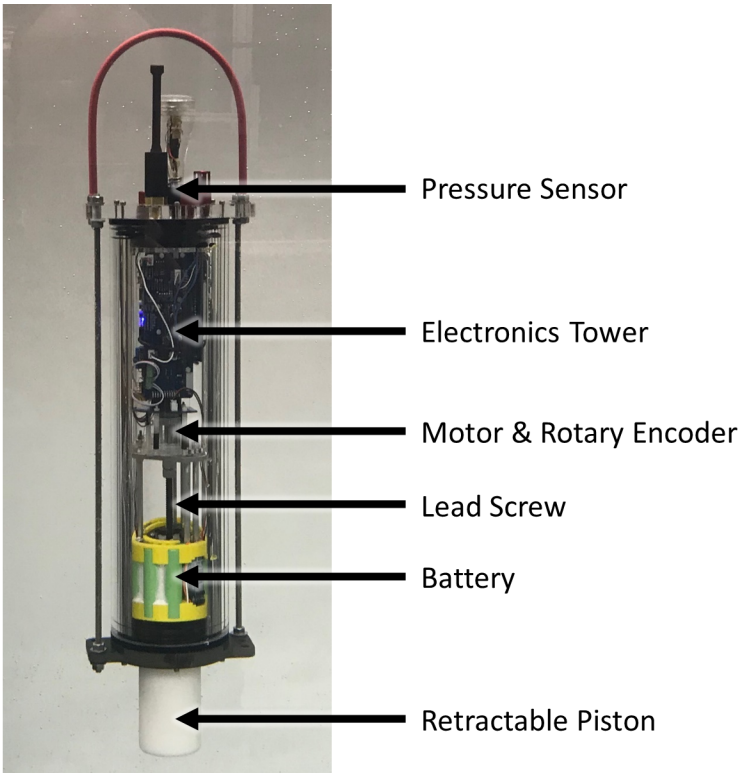

Figure 1.1: Components of the μFloat used for the control system and piston actuation.

The second important sensor for the μFloat control system is the external pressure sensor. The pressure sensor is a Honeywell MLH Series Heavy Duty Pressure Transducer with an operating range of 0-1,034 kPa (gauge) (0-150 psi) [10]. This is sufficient for the 0-100 m operating depth of the μFloat, as pressure at 100 m depth in saltwater with density of 1,030 kg/m^3 is $\sim$1000 kPa (146 psi). For control purposes, depth (z) is estimated from pressure as

$$z = \frac{P}{\rho g} \tag{1.1}$$

where P is the measured pressure, ρ is the estimated water density, and g is the acceleration due to gravity.

1.3 uFloat Existing Control Method

Currently, the μFloat instrument package utilizes a Proportional-Integral-Derivative (PID) controller. As discussed in detail in Chapter 5, the particular form of the existing μFloat controller is two Proportional-Derivative (PD) control loops cascaded in series. The first PD loop operates on the error between the target depth for the μFloat and the most recent measured depth. This first PD loop outputs a target velocity. The magnitude of this target velocity has a speed limit of 0.3 m/s. The second PD loop operates on the error between this target velocity and the most recent vertical velocity of the μFloat. The second PD loop outputs a motor speed command in units of "motos", a value representing the electrical power that will be sent to the motor to rotate the motor and lead screw assembly to retract or extend the piston.

1.4 Thesis Goals

Swarm sensing missions are likely to have different float control requirements. For example, one mission might require floats to maintain a constant depth for extended periods of time (Figure 1.2A), while another might require vertical profiling at a specified velocity, changing between positive and negative velocity at specified transition depths (Figure 1.2B). This requires a flexible control architecture.

There are also two performance improvement goals for the μFloat controller. The first is to minimize motor actuation to reduce power consumption and reduce self-noise from the motor. The second is to optimize the float trajectory to maximize time on target and minimize risk of overshooting the μFloat's operating limit of 100 m depth.

This thesis develops a Linear Quadratic Gaussian (LQG) control method for the μFloat that can provide both of the desired mission capabilities (constant depth, and constant velocity profiling) and achieve the performance improvements as compared to the μFloat's existing PD control system. To achieve the LQG control method, a dynamic model for μFloat motion in both nonlinear and linear form was developed. The nonlinear model is

compared to experimental field data to quantify the accuracy of the model prediction for float velocity and acceleration. The LQG method is a combination of a state estimator and a Linear Quadratic Regulator (LQR) control algorithm. A nonlinear, model-based Extended Kalman Filter (EKF) state estimator was chosen and implemented. The EKF estimator performance on field data is compared to the existing measurement smoothing method of a Butterworth filter. The LQR controller is developed using the linear form of the dynamic model. Finally, a simulation is developed using the nonlinear dynamic model equations, and the LQG and PD controllers are both tested in the simulation to compare their performance.

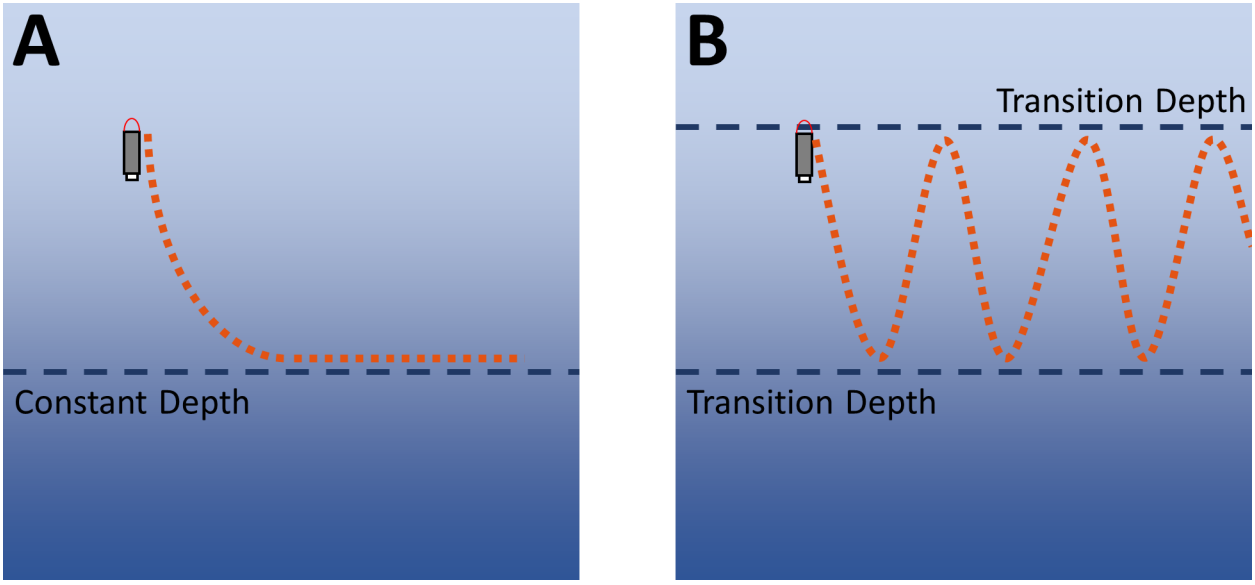

Figure 1.2: Examples of two deployment trajectories for the μFloat. (A) Descending to and maintaining a constant depth. (B) Maintaining a constant velocity while profiling between two transition depths.

Chapter 2
SYSTEM MODEL: NONLINEAR

A dynamic system model of the μFloat is required for the state estimator, the model-based controller, and the simulation developed in this thesis. In this chapter the model is developed as a set of nonlinear differential equations. In Chapter 3 these equations are linearized and represented in state space form.

2.1 Methods

A complete dynamic system model of the μFloat requires a hydrodynamic model for the interactions between the μFloat and the surrounding water, as well as an actuator model to describe how the system's control variable (motor speed command) relates to the movement of the piston.

2.1.1 Hydrodynamics Model

The hydrodynamics model begins with the forces acting on the μFloat as it moves vertically through the water, shown as a free body diagram in Figure 2.1. From Newton's Second Law, the governing equation for the μFloat behavior in the vertical direction can be expressed as

$$m_f \ddot{z} = F_w - F_b - F_d - F_a, \tag{2.1}$$

where m_f is the μFloat mass, $\ddot{z}$ is its vertical acceleration, F_w is the weight, F_b is the buoyant force, F_d is the drag force opposing motion, and F_a is the added mass force. The latter represents the force needed to accelerate the water surrounding the μFloat.

Each of these forces can be parameterized for the dynamic model. F_w is the product of the μFloat mass (m_f) and gravitational acceleration. F_b is equal to the weight of the volume

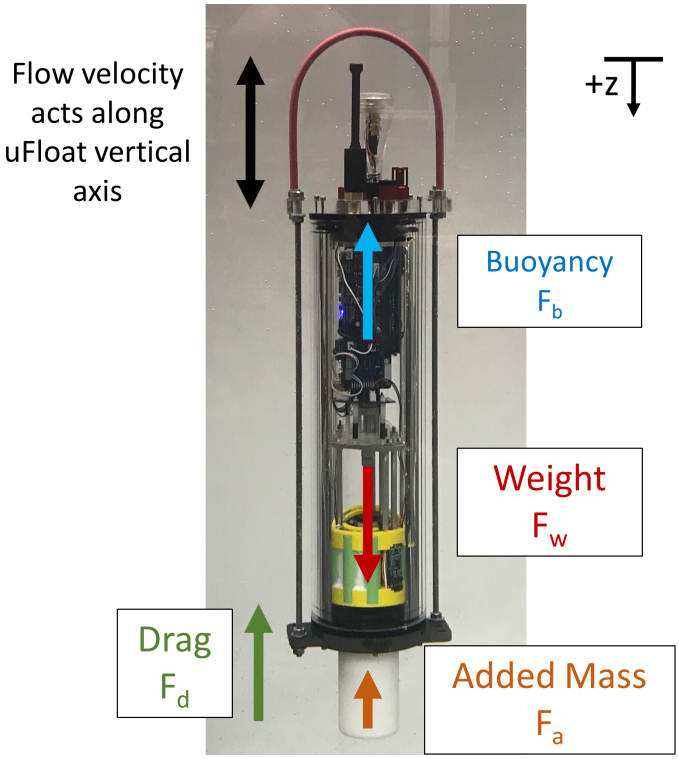

Figure 2.1: Free body diagram of μFloat for motion in the vertical direction. The directions of the drag and added mass force correspond to a descending (positive velocity), positively accelerating μFloat.

of water displaced by the μFloat and is given by

$$F_b = V_f \rho g, \tag{2.2}$$

where V_f is the μFloat volume and ρ is water density. F_d and F_a can be parameterized by Morison's equation for a solid body accelerating relative to the flow direction of a fluid [11]. These are given as

$$F_d + F_a = f(\dot{z}, \ddot{z}) = \frac{1}{2}\rho A_{cs} C_d \dot{z}|\dot{z}| + C_a V_f \rho \ddot{z}, \tag{2.3}$$

where A_{cs} is the cross-sectional area of the μFloat, C_d is the coefficient of drag, C_a is the coefficient of added mass, $\dot{z}$ is the μFloat velocity, and $\ddot{z}$ is the μFloat acceleration. The velocity and acceleration terms are approximations for the relative water velocity/acceleration

that drive drag and added mass. The relative motion is strictly equal to the μFloat motion when the water has no vertical motion. Substituting these terms into the force balance yields

$$m_f\ddot{z} = m_f g - V_f \rho g - \frac{1}{2}\rho A_{cs} C_d \dot{z}|\dot{z}| - C_a V_f \rho \ddot{z}. \tag{2.4}$$

Initial exploration with this form of the hydrodynamic model revealed that it was almost impossible to measure the mass and volume of the μFloat accurately and precisely enough for the model to accurately predict the piston position at which the μFloat would achieve neutral buoyancy. For example, the model might predict neutral buoyancy when the piston was retracted 2.3 cm from full extension, but in experiments the μFloat would achieve neutral buoyancy when retracted 2.7 cm. The model would therefore predict acceleration when it should predict zero acceleration. In addition to the obvious problem of model validation, this also highlighted a barrier for model-based control of μFloats with significant float-to-float build variability.

2.1.2 Hydrodynamics in Neutral Buoyancy Form

The solution to this modeling problem was to experimentally identify the neutral buoyancy position of the piston for each float, and make all changes to piston position and float volume relative to the neutral buoyancy values. This allows us to express the volume of the μFloat in terms of a neutral buoyancy volume,

$$V_f = V_N + \Delta V, \tag{2.5a}$$

$$V_N \approx \frac{m_f}{\rho} \tag{2.5b}$$

where V_N is the volume of the float when the piston is at its neutral buoyancy position, approximated as float mass (m_f) divided by water density (ρ), ΔV is the volume change caused by the piston moving away from its neutral buoyancy position, and V_f is therefore the total float volume. Additional details are available in appendix section A.2.

The volume change (ΔV) can be rewritten in terms of the angular piston position as measured by the rotary encoder sensor in units of "rotos" (θ), relative to the piston neu-

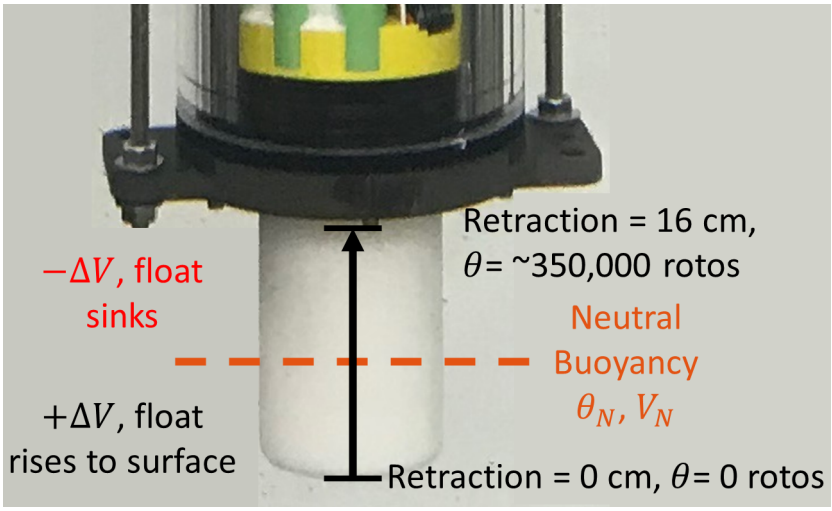

Figure 2.2: Physical relationships between the neutral buoyancy piston position, range of piston position, and change in volume.

tral buoyancy position (θ_N). Figure 2.2 shows the physical relations of the piston neutral buoyancy position to the range of piston position, and to the volume change.

The volume change, in m^3, is then given as

$$\Delta V \ = (-\theta + \theta_N) F_{conv} \pi r_p^2, \tag{2.6}$$

where $(-\theta + \theta_N)$ is the offset from neutral buoyancy position in rotos, F_{conv} is a conversion factor from rotos to piston extension length in meters, and r_p is the piston radius in meters.

Substituting these definitions for V_N and ΔV into Equation 2.5a gives a definition of float volume relative to the piston neutral buoyancy position,

$$V_f = \frac{m_f}{\rho} + (-\theta + \theta_N) \left(F_{conv} \pi r_p^2 \right). \tag{2.7}$$

Recalling the original hydrodynamic model,

$$m_f \ddot{z} = m_f g - V_f \rho g - \frac{1}{2} \rho A_{cs} C_d \dot{z} |\dot{z}| - C_a V_f \rho \ddot{z}, \tag{2.8}$$

Equation 2.7 can now be substituted for V_f in the buoyancy term, and because the volume change is small relative to the total float volume, V_f in the added mass term can be

approximated as $V_N = m_f/\rho$,

$$m_f\ddot{z} = m_f g - \left[\frac{m_f}{\rho} + (-\theta + \theta_N)\left(F_{conv}\pi r_p^2\right)\right]\rho g - \frac{1}{2}\rho A_{cs} C_d \dot{z}|\dot{z}| - C_a\frac{m_f}{\rho}\rho\ddot{z}. \tag{2.9}$$

Expanding the buoyancy term leads to a cancellation with the weight term, and the simplified form of the hydrodynamic model equation relative to neutral buoyancy is

$$m_f\ddot{z} = \left[(\theta - \theta_N)\left(F_{conv}\pi r_p^2\right)\right]\rho g - \frac{1}{2}\rho A_{cs} C_d \dot{z}|\dot{z}| - C_a m_f\ddot{z}. \tag{2.10}$$

2.1.3 Hydrodynamic Model Parameter Identification

Each parameter in Equation 2.10 must be defined to implement the model.

Physical Parameters

The three physical parameters of float mass (m_f), piston radius (r_p) and float cross-sectional area (A_{cs}) are measurable. A nominal mass of 4.90 kg was found by taking the average weight of multiple μFloats. The nominal cross-sectional area of the float endcap was taken from the CAD model as 0.0115 m^2. The piston radius was measured at a consistent 0.0301 m across multiple floats.

Piston Parameters

The conversion factor from angular rotation of the piston to piston linear extension (F_{conv}) was calculated from manufacturer specifications to be 4.43 x 10^{-7} linear m per roto of piston rotation. In operation, the piston angular position (θ) is continually measured by the rotary encoder at the motor. The neutral buoyancy piston position (θ_N) was found from field data. Specifically, time series were identified in which the float was holding constant depth in quiescent water and the mean piston position during those time series was calculated to determine a nominal, float-specific neutral buoyancy value.

Environmental Parameters

For this thesis, the water density (ρ) is taken to be 1,000 kg/m^3, consistent with experimental data collected in a freshwater lake.

Coefficients of Drag and Added Mass

The coefficients of drag (C_d) and added mass (C_a) are assigned nominal values for a smooth cylinder with approximately the same major dimensions as the μFloat. Specifically, $C_d = 0.85$, $C_a = 1.0$ [1].

State Values

The remaining values are the variables of the differential equations, float velocity ($\dot{z}$) and acceleration ($\ddot{z}$). These are related to the system's position state value, depth (z), as the first and second time derivative, respectively. In Chapter 3, the float depth and velocity are the state variables used to develop a linear, state space form of the dynamic model.

2.1.4 Actuator Model

The dynamic model for the μFloat system also requires a model of the actuator (motor and piston) behavior. The input to this system is the commanded motor speed in units of "motos", with a range of -3200 to 3200 motos. The output of the actuator system is measured by the motor's rotary encoder in units of "rotos", with 2.2566 x 10^6 rotos per linear m of piston movement ($1/F_{conv}$).

The relationship between the control input of commanded motor speed (s_m motos) to the μFloat system output of piston angular velocity ($\dot{\theta}$ rotos/sec) is approximated by a linear differential equation as

$$\dot{\theta} = q_1 s_m + q_2, \tag{2.11}$$

where $q_1 = 0.20559/dt_\theta$, $q_2 = 8.2657/dt_\theta$, and $dt_\theta = 0.100$ s. This equation was found experimentally using field data from several μFloats. Details are presented in Appendix C.

2.1.5 Complete Nonlinear Model

Differential Equations

The final nonlinear dynamic system model for the μFloat includes the hydrodynamic model (Equation 2.10) and the actuator model (Equation 2.11). For functional use, we isolate the state variables' derivatives ($\ddot{z}$, $\dot{\theta}$) on one side of the equation as

$$\ddot{z} = \left[(\theta - \theta_N) \left(F_{conv} \pi r_p^2 \right) \frac{\rho g}{(m_f + C_a m_f)} \right] - \frac{1}{2} \frac{\rho A_{cs} C_d}{(m_f + C_a m_f)} \dot{z} |\dot{z}|, \tag{2.12a}$$

$$\dot{\theta} = q_1(s_m) + q_2. \tag{2.12b}$$

Constraints and Initial Conditions

The μFloat has physical limitations that act as constraints on the dynamic model, and these must be included in any Matlab code that uses these differential equations. The constraint conditions for the state variable θ, representing piston position, are 0 to ~350,000 rotos. An additional constraint on the actuation system is the motor speed command range from -3,200 to +3,200 motos, with a "dead band" from -200 to +200 motos in which the power sent to the motor cannot overcome damping forces and no actuation occurs.

The constraint conditions for float depth (z) are the extents of the water depth, with the first constraint at the surface (0 m) and the second constraint at the maximum water depth. An additional behavior of the system that acts as a form of nonlinear constraint is the delay between initiation of the piston retraction and initiation of the μFloat descent. At the surface of the water, the piston will be fully extended, and must retract for a significant time period before crossing past the neutral buoyancy position, at which point the float will begin descending according to the model described here.

Using the model with an integrator, estimator, or control system also requires initial conditions for the states. The initial depth and velocity of the float are typically 0 m depth and 0 m/s velocity, as the deployment begins at the surface. The initial piston position is typically 0 rotos, as deployments begin with the piston at full extension.

2.2 Results

Figure 2.3 shows a comparison of the field data to the dynamic model predictions of float velocity and acceleration. The main feature of note is that while the velocity prediction is reasonably accurate, the acceleration magnitude is too high within time periods in which the piston has moved significantly away from the neutral buoyancy position, and float speed is above ∼0.2 m/s.

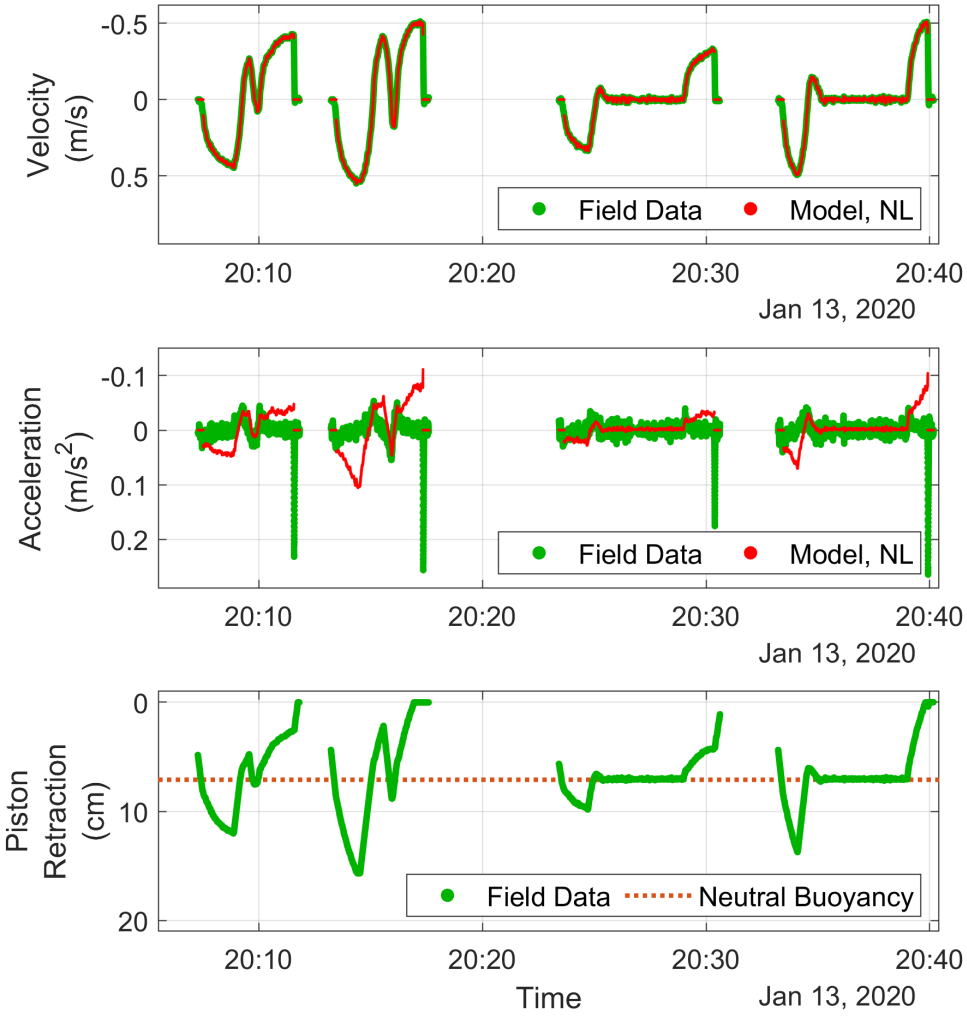

Figure 2.3: Comparison of field data to dynamic model predictions for float velocity and acceleration. The piston position and neutral buoyancy points are shown for context.

2.3 Discussion

The error in acceleration magnitude when the float is moving at higher velocities is significant. One of the first paths that could be explored to potentially address this is switching from nominal coefficients of drag and added mass to empirically determined coefficients from μFloat field data. However, initial attempts to estimate these parameters from experimental data were unsuccessful, potentially because the true values are a function of the Reynolds number and, therefore, change with float velocity. This could be implemented in the model, at the cost of greater parametric complexity.

It is also possible that the current low-order model is not capturing the effects of transient factors that can affect float volume, such as compressibility (depth dependency) and entrained air that escapes with time (time dependency). Compressibility could be accounted for with a linear relationship between the neutral buoyancy piston position and depth. Volume changes due to entrained air could be a time-dependent value added to the float volume that gradually reduces to zero with time underwater.

Because the μFloat is intended to be a multi-purpose platform, other instruments will likely be added for specific scientific missions. This will change V_f, A_{cs}, C_d, and C_a. Simulation and experiments would be required to understand the impact of these disturbances on model performance and stability.

Chapter 3
SYSTEM MODEL, LINEAR

3.1 Method

Because the LQG control method is linear, the nonlinear, second order hydrodynamic model is not directly compatible. First, the nonlinear term must be linearized using a Taylor Series approximation. Following this, the model must be converted to a set of coupled first order differential equations in state space form.

3.1.1 Taylor Series Approximation of Nonlinear Term

The only nonlinear term in the nonlinear μFloat hydrodynamic model (Equation 2.12a) is the velocity dependence of the drag term,

$$f(\dot{z}) = \frac{1}{2}\frac{\rho A_{cs} C_d}{(m_f + C_a m_f)}\dot{z}|\dot{z}|. \tag{3.1}$$

Using a Taylor Series, this can be approximated as

$$f(\dot{z}) \approx (\dot{z})c_2 a + \left[\frac{1}{2}c_2 sign(a) - c_2\right] a^2, \tag{3.2}$$

where a is the previous estimated value of the μFloat velocity ($\dot{z}$), and the coefficient c_2 is

$$c_2 = \left(\frac{\rho A_{cs} C_d}{m_f + C_a m_f}\right). \tag{3.3}$$

3.1.2 State Space Form of First Order Differential Equations

Converting the second order differential equation to coupled first order differential equations provides two state variables, $x_1 = z$ and $x_2 = \dot{z}$. The final state variable represents the state of the actuator, the piston's angular position, $x_3 = \theta$. The input to the actuator is the motor

speed command, this is the control variable $u_1 = s_m$. This gives the final form of the state vector ($\mathbf{x}$) and actuator input vector ($\mathbf{u}$) as:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} z \\ \dot{z} \\ \theta \end{bmatrix}, \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dot{\theta} \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u_1 \end{bmatrix} = \begin{bmatrix} s_m \end{bmatrix} \tag{3.4}$$

Manipulation yields a set of first-order linear differential equations

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} \dot{z} \\ \ddot{z} + \left(\theta_n c_1 + \left[\frac{1}{2} c_2 sign(a) - c_2\right] a^2\right) \\ \dot{\theta} - q_2 \end{bmatrix} = \begin{bmatrix} \dot{z} \\ -\dot{z} c_2 a + \theta c_1 \\ q_1 s_m \end{bmatrix}. \tag{3.5}$$

A detailed derivation is provided in Appendix B, starting with the nonlinear differential equations and ending with this final form of the linearized, first order differential equations.

The most recent, stored values of the physical states (z, $\dot{z}$, θ) are used as the values of x_1, x_2, and x_3, respectively, when this state space system is used within the LQG controller. The constant values on the left hand side of the equation can be neglected when the state space form is being used with the LQG controller to calculate the values of the derivatives ($\dot{z}$, $\ddot{z}$, $\dot{\theta}$), as these constant offsets only need to be considered if the state space form is being used to integrate the system forward in time. This gives the final form of the linearized, first order differential equations for the LQG controller:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_2 c_2 a + x_3 c_1 \\ q_1 u_1 \end{bmatrix}. \tag{3.6}$$

3.1.3 State Space Form of Linear Model

The LQG controller calculations uses these linear, first order differential equations in the canonical state space form,

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}. \tag{3.7}$$

Consequently, Equation 3.6 is formally recast as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -c_2 a & c_1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ q_1 \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix}. \tag{3.8}$$

3.1.4 Output Equation

The output equation of the canonical state space form is not necessary for the LQG controller, but is included here for reference. Before using the output equation, the state variables (x_1, x_2, x_3) would need to be converted back to the physical variables (z, $\dot{z}$, θ). The conversions can be found in appendix section B.5.

$$\mathbf{y} = C\mathbf{x} + D\mathbf{u}, \tag{3.9}$$

$$\begin{bmatrix} P \\ \theta \end{bmatrix} = \begin{bmatrix} \rho g & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \\ \theta \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} \begin{bmatrix} s_m \end{bmatrix}. \tag{3.10}$$

Chapter 4
FILTER METHODS

A filter or estimator is an important tool for improving control of dynamic systems. The primary control variable for the μFloat is pressure, which is a relatively noisy measurement. If the noise is passed into the control loop, the motor speed command will be determined based on the incorrect noisy value rather than the true pressure. Using a filter or estimator can reduce the amount of noise that passes into the system, improving the control outputs. This chapter describes and compares two filters: a Butterworth filter and Extended Kalman Filter.

4.1 Butterworth Filter

4.1.1 Methods

The μFloat onboard processing code that operates in real time during a deployment uses a low-pass Butterworth (BW) filter to smooth the noisy pressure signal. This is a common low-pass filter that allows a specific low-frequency range of the original signal to pass through without distortion [2]. Frequencies above the cutoff can be strongly suppressed with a high-order Butterworth filter [2]. The Butterworth filter's output is a smoothed pressure sensor signal used by the PD control system. The pressure time series can be numerically differentiated to produce a proxy signal for the float's vertical velocity through the water. This proxy signal is used with the second PD loop of the control system.

4.1.2 Results

The μFloat Butterworth filter reduces the high frequency noise of the pressure measurement, but has the disadvantage of delaying the signal by 1.6 seconds (Figure 4.1). This time delay

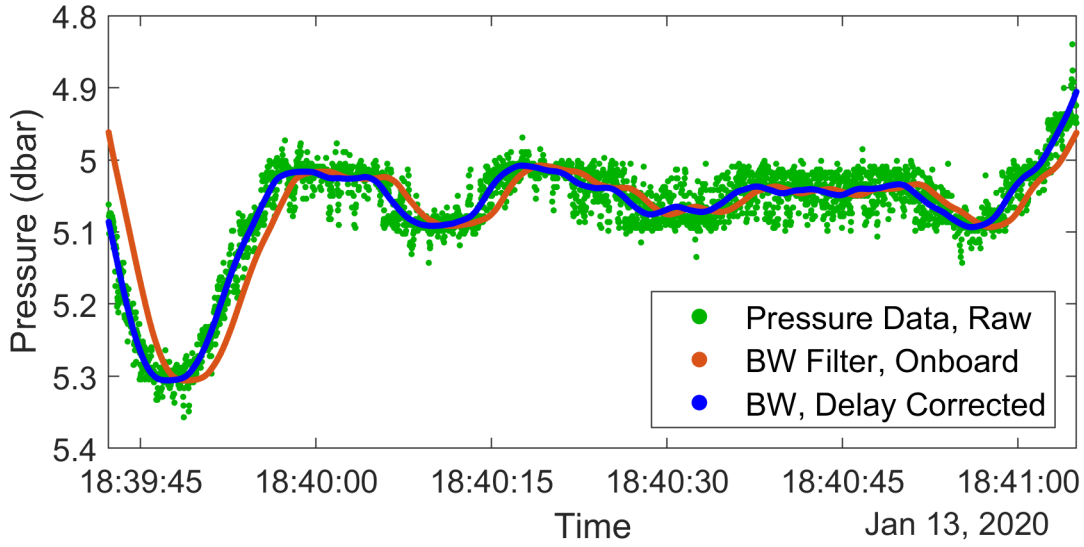

Figure 4.1: Butterworth filter performance. Green dots are the raw pressure signal with significant noise, the orange line is a smoothed, time-delayed (i.e., "phase shifted") output from the Butterworth filter that is available onboard the μFloat in real time during the deployment, and the blue line is the post-processed signal with the 1.6 sec time delay removed.

would lead to instability for a fast-moving system such as a small quadcopter, but the μFloat responds slowly enough that the time delay is negligible compared to the dominant time scale for the system dynamics. Overall, this has been an effective filter choice to use onboard the μFloat for real-time filtering.

4.2 Extended Kalman Filter

The Extended Kalman Filter (EKF) is a model-based sequential state estimation method for nonlinear dynamic systems. Here, we use a continuous dynamics, discrete measurement version of the EKF presented by Crassidis and Junkins [3]. Similar to the Butterworth filter, the EKF can reject high-frequency noise and produce a smoothed signal. A further advantage of the EKF estimator is that it can use the pressure and rotary encoder measurements to provide smoothed estimates of states not measured directly, such as depth and velocity.

4.2.1 Methods

The EKF estimator uses the μFloat nonlinear dynamic model. The EKF assumes that sensor noise is normally distributed (Gaussian white noise) and its standard deviation must be included as a parameter in the EKF, as a value in the the R_k matrix referenced in Table 4.2. The pressure sensor standard deviation is approximately 430 Pa, with slight differences based on depth and velocity of the float. This was found experimentally using the μFloat field data. The motor's rotary encoder is calibrated by the manufacturer, and a standard deviation is not available. Consequently, we assume that the standard deviation is 1 roto and note that varying this input by several orders of magnitude did not significantly affect model output. In the future, it would be possible to experimentally obtain an accurate estimate for this parameter if deemed of sufficient importance.

Model	$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) + G(t)\,\mathbf{w}(t), \mathbf{w}(t) \sim N(\mathbf{0}, Q(t))$ $\tilde{\mathbf{y}}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k, \mathbf{v}_k \sim N(\mathbf{0}, R_k)$
Initialize	$\hat{\mathbf{x}}(t_0) = \hat{\mathbf{x}}_0$ $P_0 = E\left\{\tilde{\mathbf{x}}(t_0)\,\tilde{\mathbf{x}}^T(t_0)\right\}$
Gain	$K_k = P_k^- H_k^T(\hat{\mathbf{x}}_k^-)[H_k(\hat{\mathbf{x}}_k^-)P_k^- H_k^T(\hat{\mathbf{x}}_k^-) + R_k]^{-1}$ $H_k(\hat{\mathbf{x}}_k^-) \equiv \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right\|_{\hat{\mathbf{x}}_k^-}$
Update	$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k[\tilde{\mathbf{y}}_k - \mathbf{h}(\hat{\mathbf{x}}_k^-)]$ $P_k^+ = [I - K_k H_k(\hat{\mathbf{x}}_k^-)]P_k^-$
Propagation	$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), \mathbf{u}(t), t)$ $\dot{P}(t) = F(t)\,P(t) + P(t)\,F^T(t) + G(t)\,Q(t)\,G^T(t)$ $F(t) \equiv \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right\|_{\hat{\mathbf{x}}(t), \mathbf{u}(t)}$

Figure 4.2: The EKF estimator algorithm in continuous dynamics, discrete measurement form. *Image Credit:* Crassidis and Junkins, Table 3.9, Extended Kalman Filter [3].

Table 4.2 shows the algorithm for the EKF estimator. The EKF requires a state vector,

$$\mathbf{x}(t) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} z \\ \dot{z} \\ \theta \end{bmatrix}, \tag{4.1}$$

and an input vector,

$$\mathbf{u}(t) = \begin{bmatrix} u_1 \end{bmatrix} = \begin{bmatrix} s_m \end{bmatrix}. \tag{4.2}$$

The EKF also requires a dynamic system model

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) + G(t)\mathbf{w}(t), \tag{4.3}$$

which will be the nonlinear model of the μFloat system (Equation 2.12), plus the process noise vector, $G(t)\mathbf{w}(t)$. The EKF method requires a system output model,

$$\tilde{\mathbf{y}}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k, \tag{4.4}$$

that describes the relationship between the output vector containing the measurements of pressure and piston position ($\tilde{\mathbf{y}}_k$) and the state vector at time step k ($\mathbf{x}_k$) and includes a measurement noise vector ($\mathbf{v}_k$). This output model will be

$$\tilde{\mathbf{y}}_k = \begin{bmatrix} \tilde{P}_k \\ \tilde{\theta}_k \end{bmatrix} = \begin{bmatrix} \rho g z_k \\ \theta_k \end{bmatrix} + \mathbf{v}_k = \begin{bmatrix} \rho g x_{1,k} \\ x_{3,k} \end{bmatrix} + \mathbf{v}_k. \tag{4.5}$$

The system requires an initialization step where the designer provides a best guess for the initial system states ($\hat{\mathbf{x}}_0$) and a best guess of the covariance (error) of the state knowledge (P_0). On each pass through the algorithm, an estimator gain (K_k) is calculated based on the current state estimate ($\hat{\mathbf{x}}_k^-$) the current covariance estimate (P_k^-) and the nonlinear system model, represented by H_k and R_k. In the update step, the EKF estimator uses the newly calculated gain (K_k) multiplied by the difference between the latest pressure and rotary encoder measurements ($\tilde{\mathbf{y}}_k$) and the prediction of the system output based on the most recent, stored state estimate ($\mathbf{h}(\hat{\mathbf{x}}_k^-)$) to produce an improved state estimate for this time

step ($\hat{\mathbf{x}}_k^+$). The covariance (P_k^-) is also updated using the gain (K_k) to obtain an improved covariance estimate for this time step (P_k^+). The final step is to propagate the estimate model forward in time to the start of the next EKF time step by integrating the nonlinear differential equations of the μFloat model, $\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)$, and the covariance differential equation $\dot{P}(t) = F(t)P(t) + P(t)F^T(t) + G(t)Q(t)G^T(t)$. This integration is done using Matlab's ode45 function [12]. The results of these integrations become the starting guess of the state vector and the covariance matrix for the next iteration of the algorithm.

The diagonal matrix $Q(t)$ represents the system disturbance noise. This is difficult to measure, and instead becomes the EKF tuning knob, allowing the designer to adjust the filter performance by adjusting each value within the $Q(t)$ matrix. Each q_n value within the $Q(t)$ matrix,

$$Q(t) = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix}, \tag{4.6}$$

is associated with a corresponding $x_n(t)$ value in the state vector $\mathbf{x}(t)$. The physical intuition for choosing the q_n values is that the value should be relatively large if the sensors are more reliable than the model prediction for the associated state, and the q_n value should be relatively small if the model prediction is likely to be more reliable than the sensor measurements [7]. For the μFloat EKF estimator, a reasonable $Q(t)$ matrix is:

$$Q(t) = \begin{bmatrix} 0.000005 & 0 & 0 \\ 0 & 0.00005 & 0 \\ 0 & 0 & 100000 \end{bmatrix}. \tag{4.7}$$

4.2.2 Results

As shown in Figure 4.3, the EKF estimator produces a satisfactory result. While the EKF estimate for depth is not quite as smooth as the Butterworth filtered estimate, it has virtually no time delay and is available in real-time for μFloat control.

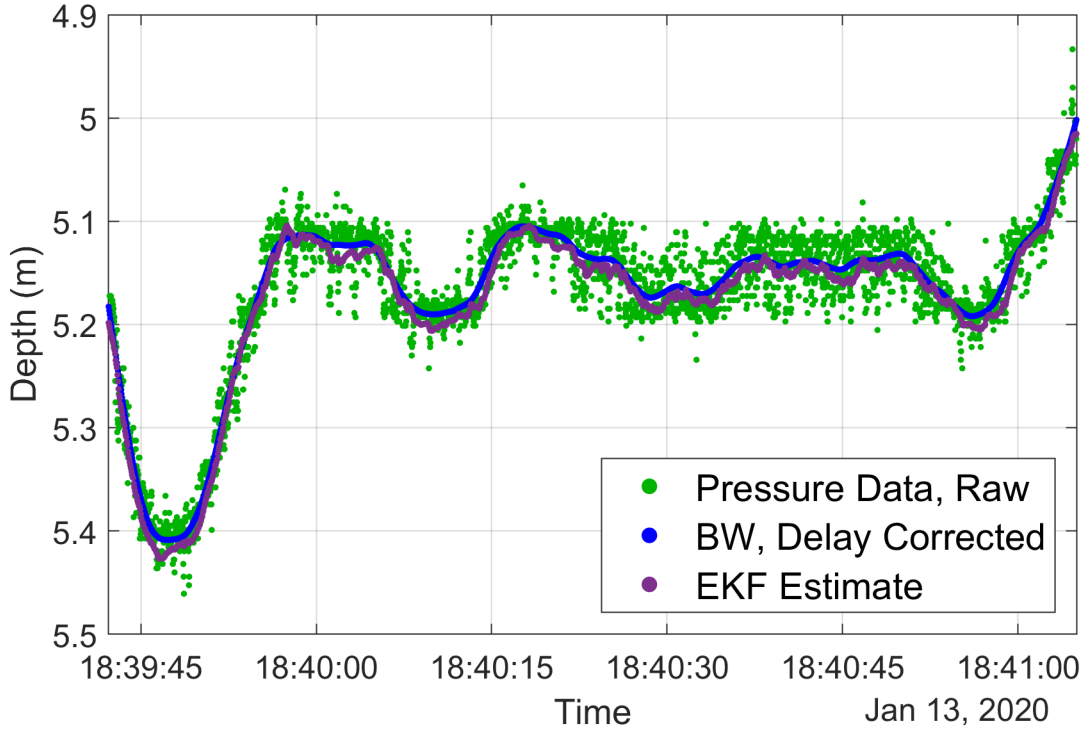

Figure 4.3: EKF estimator performance showing the noisy raw pressure signal (green), the Butterworth filter with the delay corrected in post-processing (blue), and the EKF estimate of the depth that would be available onboard the μFloat in real time during the deployment (purple).

4.3 Discussion: Filter Comparison

A final comparison of the Butterworth filter and Extended Kalman Filter estimator performance are shown in Figure 4.4. The EKF provides clearly superior performance in terms of time delay and provides a smooth representation of the pressure data. In addition to the qualitative comparison, we quantitatively evaluate performance by the root mean square error (RMSE) as

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (v_i - w_i)^2} \tag{4.8}$$

where **v** and **w** are any two vectors of length N for which an error value between the vectors is desired.

The EKF estimator has an RMSE of 0.022 when compared to the original noisy pressure data, and the Butterworth filter with the 1.6 sec delay that occurs during real-time performance produces a higher root mean square error of 0.039.

The real-time accuracy of the EKF represents an improvement over the Butterworth filter, but a potential disadvantage of the EKF is the higher computational cost. If the μFloat processor is able to complete the necessary calculations, switching to the EKF could provide the benefit of a smooth estimate of depth and velocity.

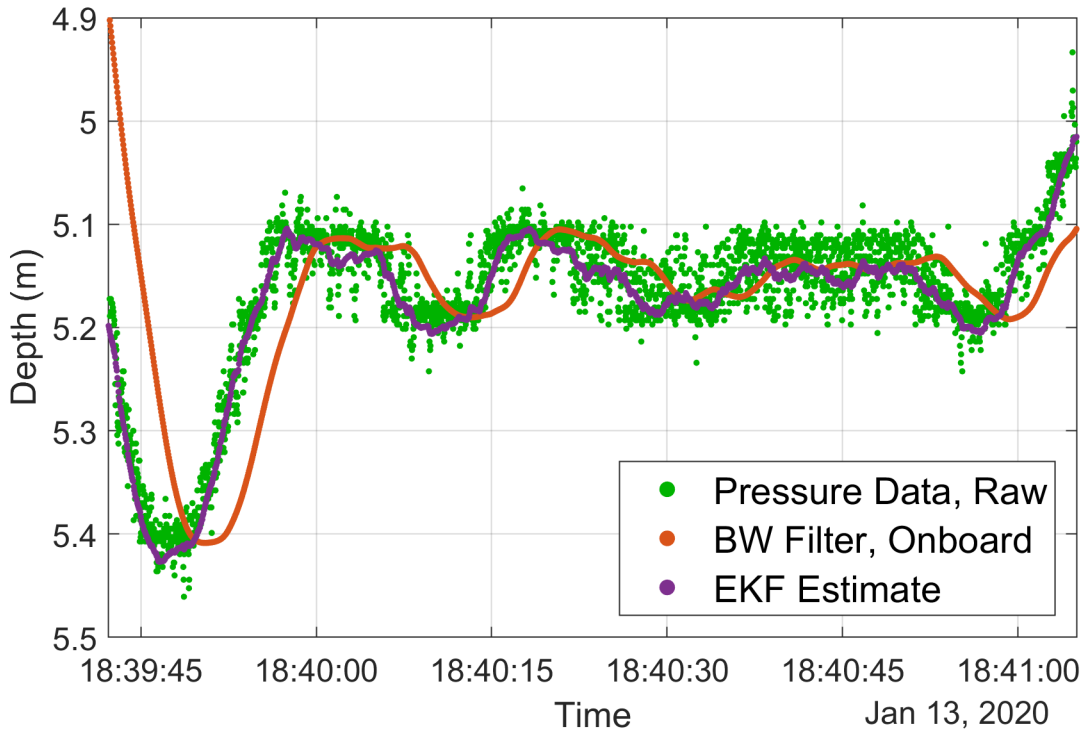

Figure 4.4: Comparison of the real time signals that would be available onboard the μFloat during a deployment for the Butterworth filter and the EKF estimator.

Chapter 5

CONTROL METHODS

The μFloat currently uses a classic PID controller, a common control method for automatic systems [12]. A potential alternate control method is the Linear Quadratic Gaussian (LQG) controller, commonly used for trajectory tracking, including applications such as determining optimal headings within autopilot and autonomous control systems for ships and underwater vehicles [5]. This chapter reviews the performance of each control method, evaluating them on the basis of minimizing actuation and following a prescribed trajectory, which can be described in terms of the dynamic system's transient response with metrics such as overshoot and settling time.

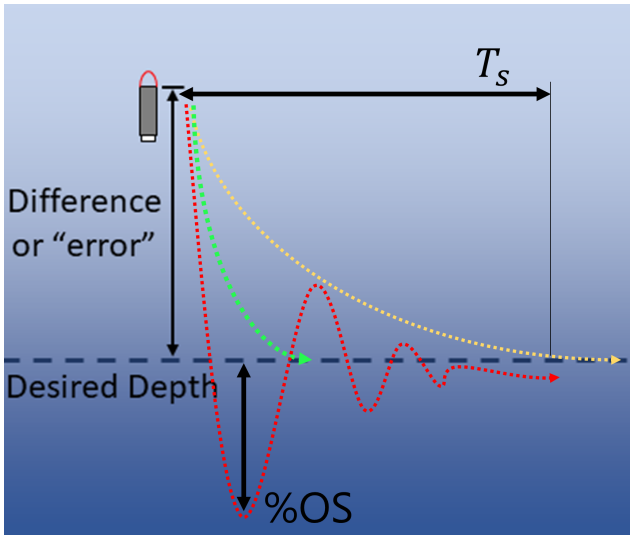

Figure 5.1: Transient response metrics of percent overshoot (%OS) and settling time (T_s). Green represents an ideal trajectory, with no OS and short T_s. Yellow represents the system actuating too slowly, with no OS, but excessively long T_s. Red is the worst possible trajectory for the μFloat, with long T_s and high OS causing risk of traveling past the 100 m depth operating limit.

As shown in Figure 5.1, overshoot (OS) is the maximum distance the float travels past the desired depth before settling to the desired depth. Settling time (T_s) is the length of time required for the float to settle and remain within 2% of the desired depth [12]. An ideal trajectory has a quick settling time and minimal overshoot. A dangerous trajectory for the μFloat is large overshoot, as the float could accidentally travel past the operational limit of 100 m depth, potentially causing pressure housing failure or damage to the float sensors. A second problematic trajectory is an unnecessarily slow descent, as the deployment could end before the float reaches its target depth.

5.1 Proportional Derivative Controller

A Proportional Derivative (PD) controller is a subset of the classic Proportional-Integral-Derivative (PID) control method. While more advanced control systems such as optimal and predictive control are available, PID continues to be a common control method for automated systems applications [12]. An advantage of PID control is that it does not require a dynamic model of the physical system; the control algorithm requires only an error value for the calculations. The error value can be as simple as the difference between a desired value for the system (commonly called a reference value or target value) and the measured value.

5.1.1 Method

The existing μFloat control utilizes two cascaded PD controllers, shown in Figure 5.2 in block diagram form. The discrete form of the PID feedback control equation is given in Appendix E. The operating signals for these two controllers are the pressure in decibars (P, dbar), and the first time derivative of the pressure (dP/dt, dbar/s). The relationship between pressure (P) and depth (z) is given by

$$z = \frac{P}{\rho g}, \tag{5.1}$$

where ρ is water density, and g is gravitational acceleration. In freshwater with density of 1,000 kg/m^3, 10 dbar is equal to 10.2 m of water depth. In saltwater with density of 1,030

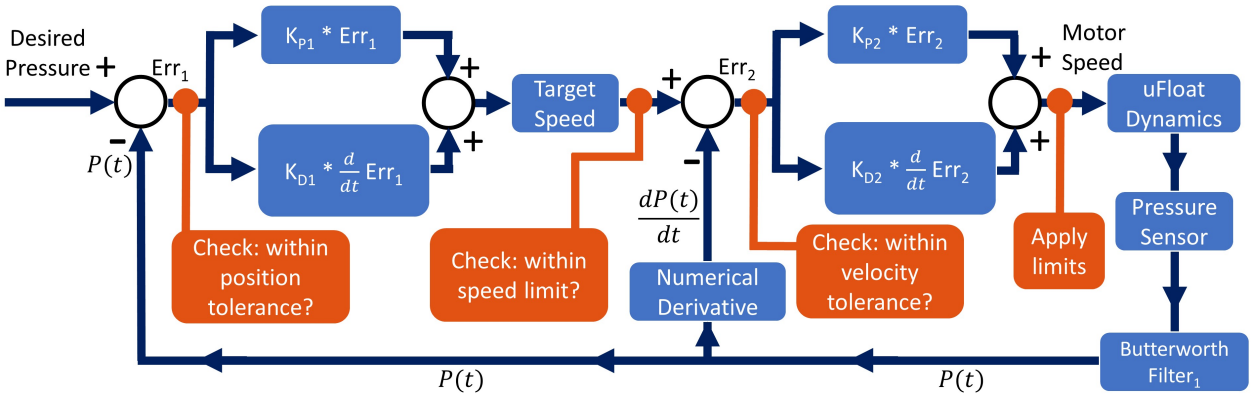

Figure 5.2: Block diagram showing the current μFloat control method of cascaded PD control with speed limit. The loop on the left is the "depth" PD controller, and the loop on the right is the "velocity" PD controller.

kg/m^3, 10 dbar is equal to 9.9 m depth. As these magnitudes are reasonably similar, the existing μFloat control system uses the pressure in dbar as a proxy for depth in m, and the pressure time derivative in dbar/s as a proxy for velocity in m/s.

To use the pressure signal as the operating signal for the PD controllers, the noisy pressure measurements are filtered with a high order Butterworth filter (Chapter 4.1). The filtered pressure signal has its first derivative calculated with a numerical method.

The first PD controller is the "depth" controller, calculating the error between the desired pressure and filtered pressure measurement, and outputting a target "velocity" for the float. The second PD controller is the "velocity" controller, calculating the error between the target velocity (after the speed limit was applied) and the derivative of the filtered pressure to output a motor speed command.

The proportional and derivative gains used on the μFloat were empirically chosen sweeping through a range of gain values (Trevor Harrison, pers. comm). However, even the best-performing gains resulted in significant depth overshoot. To compensate for this, a speed limit was added to the PD controller. Physically, the velocity restriction limits piston movement to less than its maximum range.

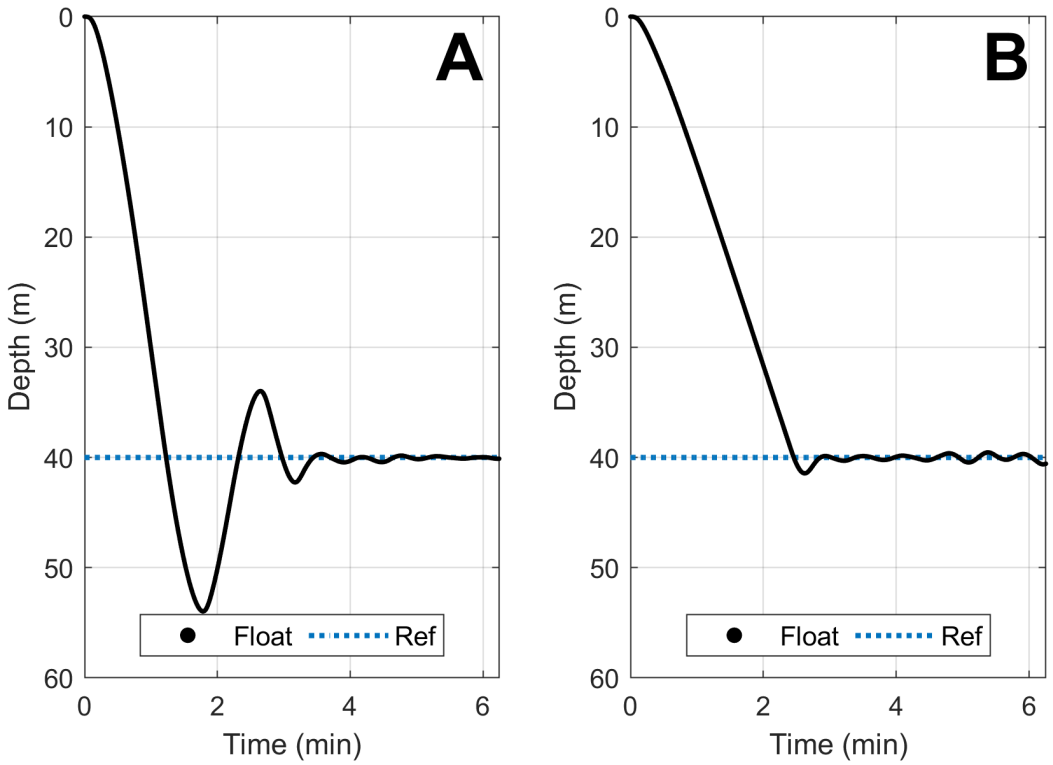

Figure 5.3: (A) PD controller with no speed limit. (B) PD controller with 0.3 m/s speed limit.

5.1.2 Results

Figure 5.3 shows that a speed limit improves the baseline PD control performance, reducing overshoot from 12 to 1.7 m for a 40 m target depth when starting from the surface. Settling time remains 3 minutes, as the float with no speed limit first reaches the reference depth at approximately 1 minute, but needs 2 additional minutes to recover from the overshoot and settle within 2% of the reference depth.

The existing PD controller with a speed limit meets the requirements of the μFloat. For example, in a constant depth deployment, as shown in Figure 5.4, the float settles to its target depth with limited overshoot.

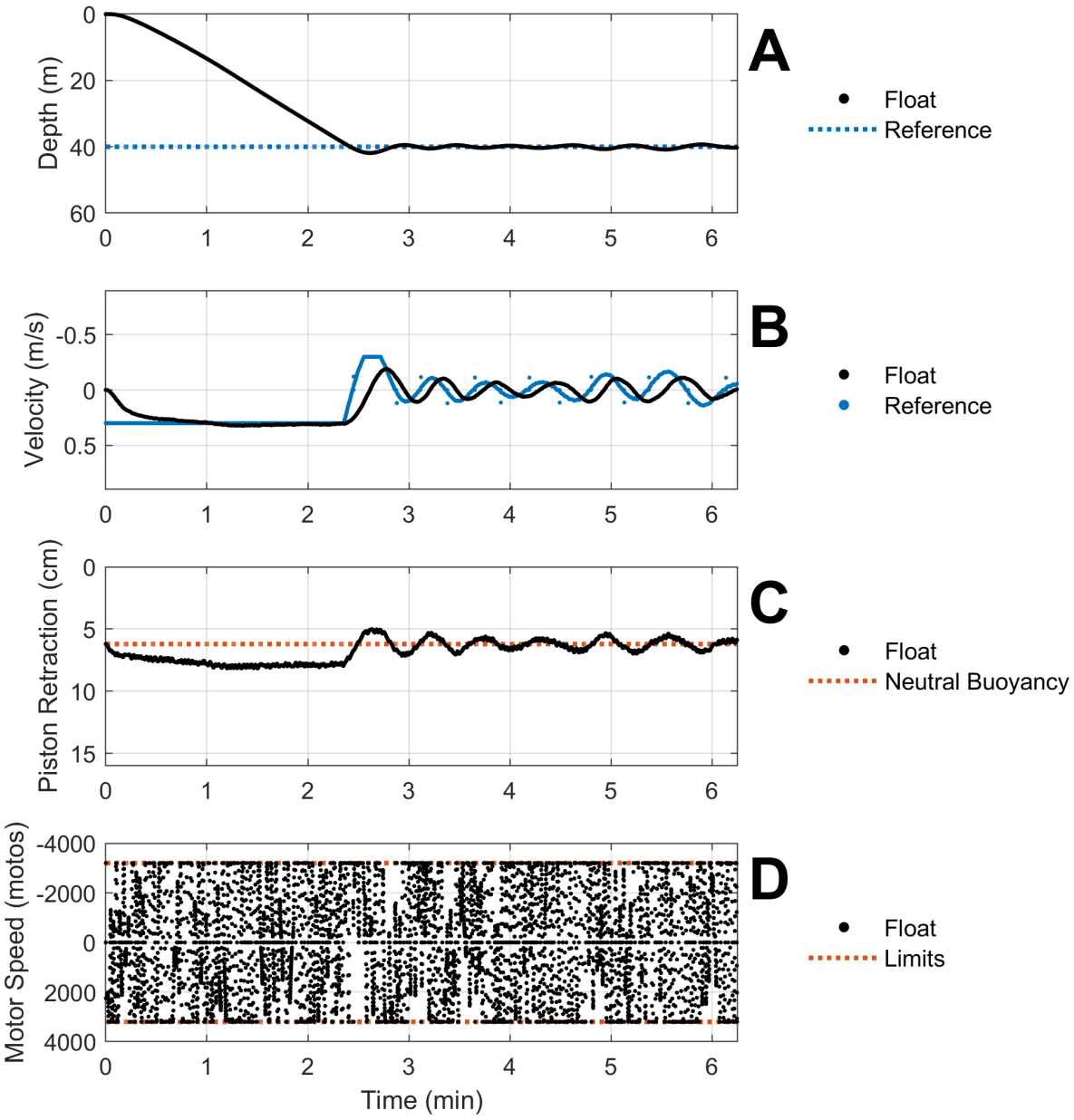

Figure 5.4: PD controller for a constant depth deployment.

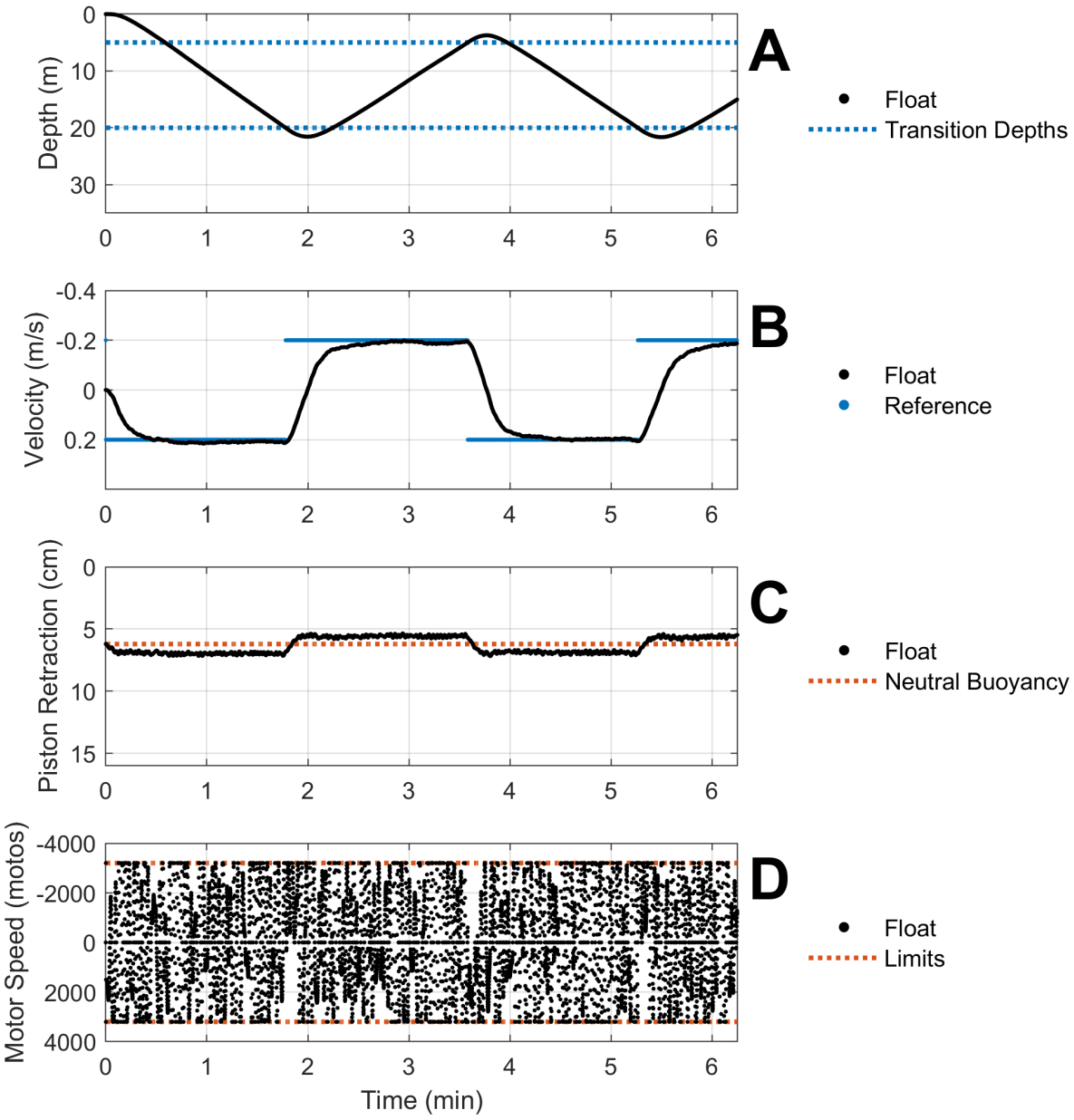

Figure 5.5: PD controller for a constant velocity profiling deployment.

Similarly, as shown in Figure 5.5, the PD controller allows the μFloat to perform profiling at a near-constant velocity, switching between positive and negative 0.2 m/s when the float arrives at the transition depths of 5 m and 20 m depth.

In both deployment modes, the motor does, however, continually actuate after approaching the reference depth, which could have implications for float endurance due to unnecessary power consumption. This could be reduced by introducing an actuation "dead band" around the target depth or velocity, a tolerance range in which actuation would be suppressed by sending a motor speed command of 0 motos.

5.2 Linear Quadratic Gaussian Controller

The Linear Quadratic Gaussian (LQG) controller implemented in this thesis combines the Extended Kalman Filter (EKF) state estimator from Chapter 4 with an optimal control method, a Linear Quadratic Regulator (LQR), shown as a block diagram in Figure 5.6. An advantage of the LQR control method is that if the system is controllable, observable, and modeled with sufficient accuracy, the resulting state feedback system will be both stable and optimal [4]. Further, the EKF estimator and LQR control algorithm are determined independently, which greatly simplifies the design process [4]. However, because LQG control requires a system model (unlike PD control), errors in the model or real-world disturbances outside the expected ranges may cause the feedback system to become unstable [4].

5.2.1 Method

The LQR control method is commonly used for trajectory tracking [4]. LQR control requires an ongoing estimate of all states, which is provided by the EKF estimator. The regulator acts to move a system from its current estimated state ($\hat{\mathbf{x}}$) to the desired (reference) state ($\hat{\mathbf{x}}_{ref}$) [6]. The LQR controller will act on the error between these values, defined as

$$\hat{\mathbf{x}}_{err} = (\hat{\mathbf{x}} - \hat{\mathbf{x}}_{ref}). \tag{5.2}$$

This error is the input to the LQR's proportional feedback control,

$$u = -K\hat{\mathbf{x}}_{err}. \tag{5.3}$$

where u is the input to the actuator system (motor speed command, s_m), and K is the control gain.

To determine the optimal control gain K, the LQR requires a linear system model in state space form,

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}, \tag{5.4}$$

where $\mathbf{x}$ and $\dot{\mathbf{x}}$ are respectively the state vector and its time derivative, and $\mathbf{u}$ is the control input (Equation 3.4). The μFloat linear state space model was developed in Chapter 3 and is presented in equation 3.8.

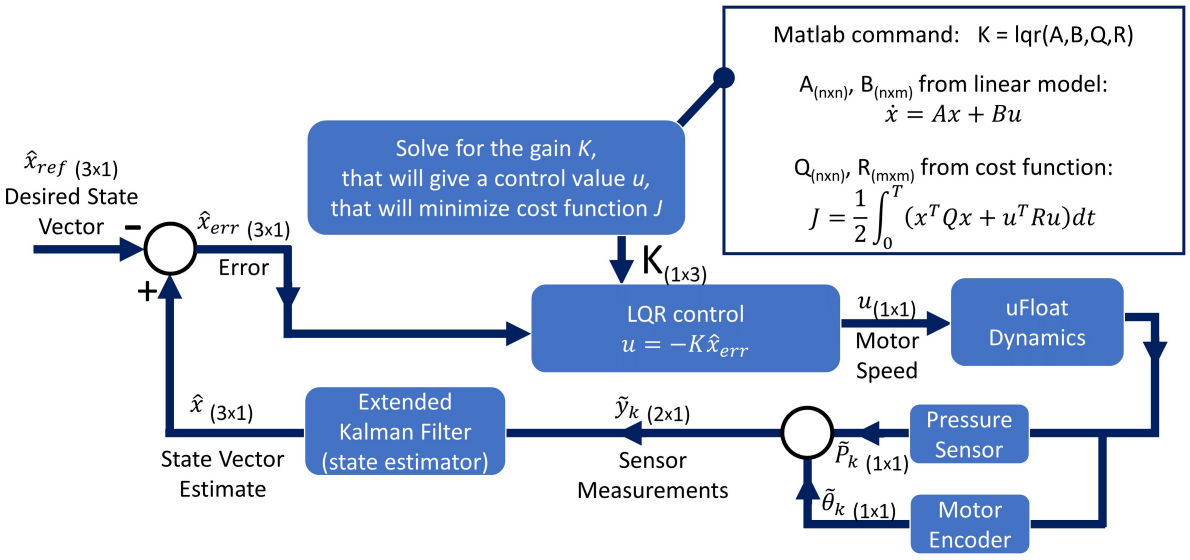

Figure 5.6: Block Diagram for LQG method, including EKF estimator and LQR proportional feedback controller.

The LQR method also requires a quadratic cost function (J) that accounts for both the cost of state errors, and the cost of actuation,

$$J(\mathbf{x}, u) = \int_0^\infty \mathbf{x}^T Q\mathbf{x} + u^T Ru \; dt, \tag{5.5}$$

where $\mathbf{x}$ is the state vector, matrix Q weights the cost of state errors, u is the control value of motor speed, and matrix R weights the cost of actuation [4]. In our application, the LQR controller uses the weighting matrix Q to prioritize whether the controller will attempt to minimize error in either the float depth or the float velocity. The LQR controller can therefore accommodate either constant depth or constant velocity profiling deployments. The weighting matrix R adjusts the magnitude of the command signal (motor speed) generated by the control algorithm. The R matrix was tuned to correspond to the desired range of motor speed commands, which must fall in the motor's range of -3,200 to 3,200 motos.

With all of these values determined, a series of calculations must be performed at each time step to solve for the gain (K) that will give a control value (u) that will minimize the cost function (J), thereby giving a trajectory that is optimal according to the designer's preferences. Packages to carry out these calculations are available for common programming languages (e.g., MATLAB, Python) [6].

5.2.2 Results

Figures 5.7 and 5.8 show the results of a basic LQG controller for a constant depth deployment and a profiling deployment, respectively. The constant depth deployment (Figure 5.7) has a large overshoot, approximately 15 m for a 40 m target depth, similar to the PD controller without speed limit (Figure 5.3A). The LQG controller performance would likely be improved by incorporating a similar non-linear speed limit.

For the profiling deployment, LQG control produces an unusual feature. As seen in Figure 5.8D, the motor speed command has significant variations only when the float is ascending in the water column. On periods of the deployment in which the float is descending, the float reaches the desired velocity of 0.2 m/s and then sends motor speed commands of zero. This is the expected behavior. The noisy motor speed commands when the float is ascending are an unexpected result that require further investigation. For both deployment types, motor actuation is relatively limited once the float approaches its target value (Figures 5.7D, 5.8D).

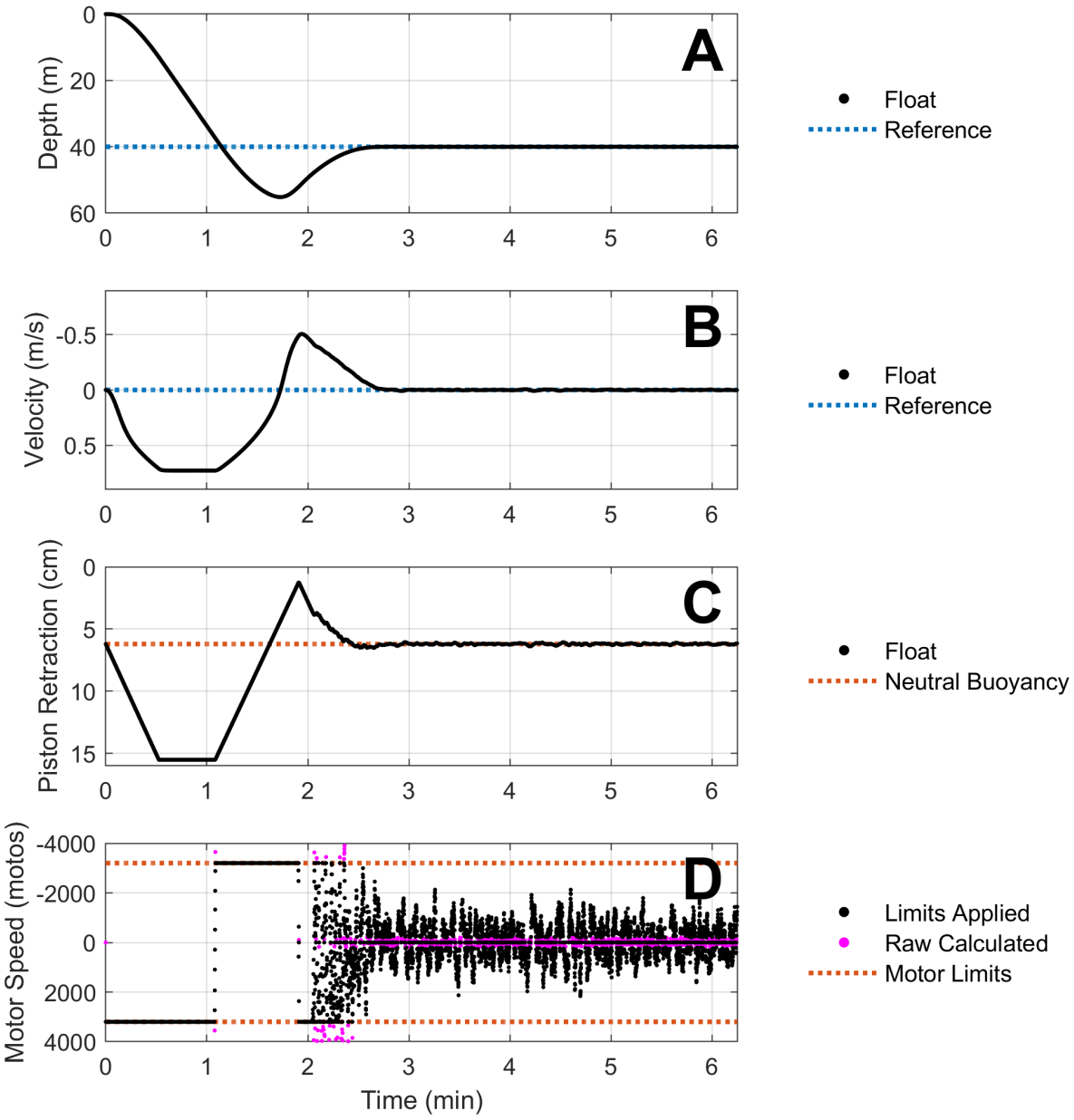

Figure 5.7: Proposed LQG controller for a constant depth deployment.

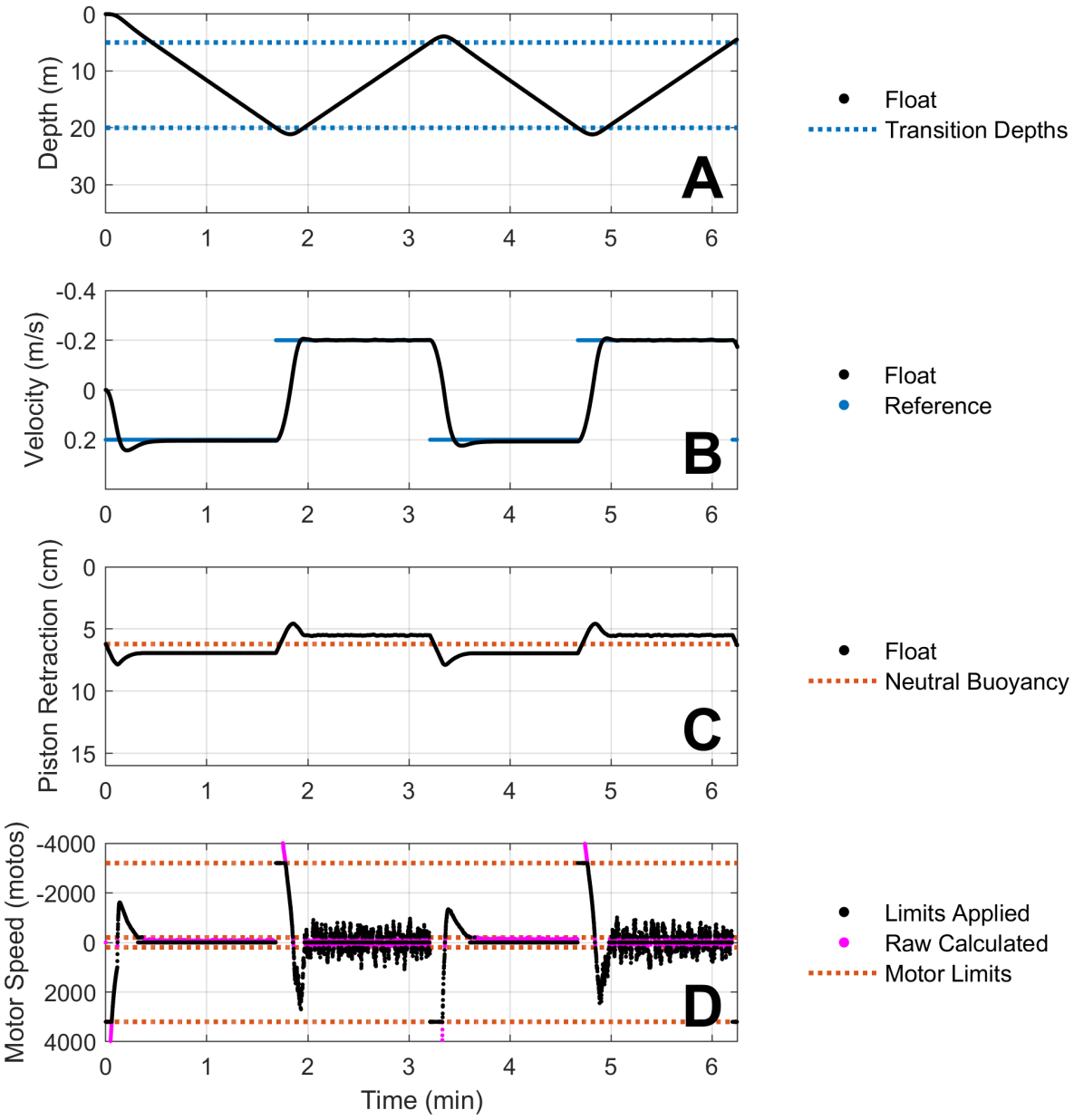

Figure 5.8: Proposed LQG controller for a constant velocity profiling deployment.

5.3 Discussion: Comparison of LQG and PD Controllers

5.3.1 Performance Characteristics

Figure 5.9 compares the PID and LQG control methods side by side for the same deployment mission (e.g., a constant depth of 40 meters). The two main observations are that the speed-limited PD control has significantly less overshoot (Figure 5.9A), and the LQG control produces significantly lower motor speed commands (Figure 5.9H). It is also notable that the LQG controller has lower magnitude oscillations about the piston neutral buoyancy point than the PD controller (Figure 5.9G). However, as the current μFloat PD system keeps the float within a reasonable distance of the reference depth (Figure 5.9A), this may not be a significant difference and this result requires verification on hardware.

The LQG controller's lower magnitude motor speed commands could be helpful for power conservation. This performance characteristic could also be helpful if the μFloat is equipped with a hydrophone, as this would minimize self-noise from the μFloat actuator.

5.3.2 Additional Work

Further analysis is needed on the LQG method's robustness to system variation and modeling error. This could be done with frequency analysis or simulation, and then confirmed with an experiment in a lab tank before attempting a field experiment.

A potential improvement for the LQG controller would be a speed limit, similar to the one created for the PD controller. This should help reduce overshoot. This could be implemented by starting the float in constant velocity control, and only switching control modes to constant depth control once the float is within a specified distance from the desired depth.

A potential improvement to the PD controller that could be explored is applying a constraint on actuation when the float is within a specified tolerance of the desired depth or velocity. Limiting or banning actuation signals within this tolerance band could reduce power use and self-noise, bringing the PD controller closer to the predicted LQG performance.

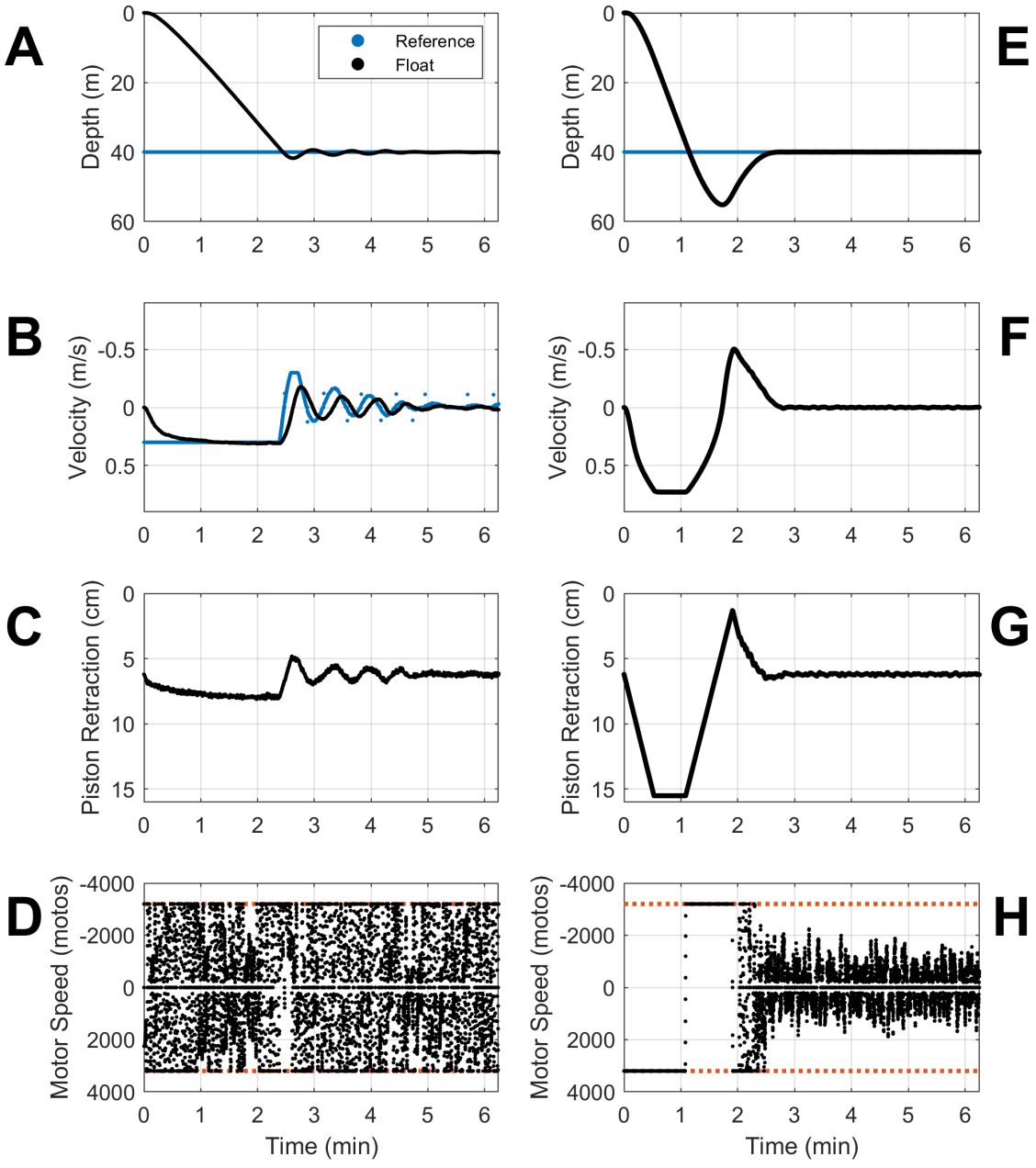

Figure 5.9: Comparison of PD and LQG control methods for a constant depth deployment. (A-D) PD controller with speed limit. (E-H) LQG controller, no speed limit.

Chapter 6

CONCLUSION

The nonlinear model for the μFloat dynamic system is a promising start but does not accurately predict acceleration. A path that could potentially address this is to switch from nominal coefficients of drag and added mass to empirically determined coefficients from μFloat field data. Similarly, one could treat the drag and added mass coefficients as free parameters and identify those that provide the best fit to multiple μFloat trajectories. Improving the acceleration prediction is an important first step in future work because it is likely to improve the results of both the EKF estimator and LQG control method. A more accurate nonlinear model would also improve the Matlab simulation of the μFloat that can be used to test different filters, estimators, and control methods.

This work has demonstrated that even without an accurate prediction of acceleration, the EKF estimator can produce a smoothed signal from the noisy pressure data to be used as an operating signal for control systems. The EKF estimator demonstrated a performance advantage over the current filtering method, as the EKF has negligible delay and the Butterworth filter has a 1.6 second delay. This could be helpful if the μFloat will be deployed into active tidal areas with faster-acting disturbances from down-welling or up-welling water.

This work has also demonstrated that the EKF method works well as the estimator portion of the LQG control method, and that LQG control can accommodate the two desired deployment types: constant depth and constant velocity profiling. The LQG simulation also suggests that this control architecture would require less overall actuation than the existing PD controller, which also meets the project goal of conserving power and minimizing self-noise from the motor. In this initial, simplest configuration, the LQG method does not prevent overshoot, and therefore does not meet the goal of optimizing the μFloat trajectory.

However, it does seem likely that an analogous implementation of the speed limit used with the PD control method would produce a similar improvement to the LQG performance, such that the LQG control could achieve comparable trajectory control to the existing PID control.

BIBLIOGRAPHY

[1] R.D. Blevins. *Applied Fluid Dynamics Handbook*. Van Nostrand Reinhold Company Inc, 1984.

[2] L.F. Chaparro. *Signals and Systems using MATLAB*. Elsevier Science & Technology, 2010.

[3] J.L. Crassidis and J.L. Junkins. *Optimal Estimation of Dynamic Systems*. Taylor & Francis Group, second edition, 2012.

[4] T. Duriez, et al. *Machine Learning Control: Taming Nonlinear Dynamics and Turbulence*. Cham: Springer International Publishing AG, 1st edition, 2016.

[5] T.I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, Inc., 1st edition, 2011.

[6] S.B. Fuller. Lecture notes, lqr control. University of Washington, Winter 2020.

[7] S.B. Fuller. Lecture notes, state estimation for control. University of Washington, Winter 2020.

[8] C.J. Geyer. Lecture notes, statistics 5101. University of Minnesota, Winter 2001.

[9] T.W. Harrison, et al. Adaptable swarm sensing in coastal waterways: Design and performance of the μfloat system. *In Preparation*, 2021.

[10] Honeywell International. Heavy duty pressure transducers.

[11] J.N. Newman. *Marine Hydrodynamics*. The MIT Press, 2017.

[12] N.S. Nise. *Control Systems Engineering*. John Wiley & Sons, Inc., 7th edition, 2015.

[13] D.H. Roemmich, et al. Integrating the ocean observing system: mobile platforms. *Proceedings of OceanObs'09: Sustained Ocean Observations and Information for Society*, 1:377–394, 2010.

[14] Servocity. 84 rpm hd premium planetary gear motor w/encoder.

Appendix A

NONLINEAR HYDRODYNAMIC MODEL DERIVATION

A.1 Newton's Second Law Equation

Our starting point is Newton's Second Law with parametric representations of drag and added mass

$$m_f \ddot{z} = F_w - F_b - F_d - F_a \tag{A.1.1}$$

$$m_f \ddot{z} = m_f g - V_f \rho g - \frac{1}{2}\rho A_{cs} C_d \dot{z}|\dot{z}| - C_a V_f \rho \ddot{z}. \tag{A.1.2}$$

A.2 Neutral Buoyancy Condition

When the float is at neutral buoyancy (piston position θ_N, volume V_N), we assume that velocity ($\dot{z}$) and acceleration ($\ddot{z}$) are approximately zero, and therefore the LHS, drag, and added mass terms are approximately zero as

$$0 = m_f g - V_N \rho g - 0 - 0 \tag{A.2.3}$$

$$m_f g \approx V_N \rho g. \tag{A.2.4}$$

We use this to determine a nominal neutral buoyancy volume (V_N) for the float,

$$\frac{m_f}{\rho} \approx V_N, \tag{A.2.5}$$

where float mass (m_f) is assumed to be the average weights of all uFloats (4.90 kg) and water density (ρ) is assumed to be 1,000 kg/m^3.

We now express the float volume in terms of the neutral buoyancy volume and the volume change (ΔV) relative to neutral buoyancy:

$$V_f = V_N + \Delta V \tag{A.2.6}$$

Volume change ΔV can be calculated as the volume of the piston offset relative to neutral buoyancy, with piston retraction measured by the rotary encoder count (θ) relative to the neutral buoyancy piston position (θ_N). Finding the conversion factor from angular position to volume uses values provided by the manufacturers of the motor and lead screw. The motor's rotary encoder counts 4,776.4 rotos per 1 revolution. The lead screw has a pitch of 12 revolutions per 1 inch of linear travel. The full conversion factor is given by

$$F_{conv} = \left[\left(\frac{1\ rev}{4,776.4\ rotos}\right)\left(\frac{1\ inch}{12\ rev}\right)\left(\frac{0.0254\ m}{1\ inch}\right)\right] = 4.4315 \text{ x } 10^{-7}\frac{m}{rotos}. \tag{A.2.7}$$

Substituting F_{conv} into the volume change equation yields

$$\Delta V = (-\theta + \theta_N)\ rotos\left[F_{conv}\ \frac{m}{rotos}\right](\pi r_p^2\ m^2). \tag{A.2.8}$$

Now, substituting equation A.2.8 for ΔV in equation A.2.6 and simplifying the notation, the new float volume definition is

$$V_f = V_N + (-\theta + \theta_n)\left(\pi r_p^2\right) F_{conv}. \tag{A.2.9}$$

Returning to the sum of forces acting on the float

$$m_f\ddot{z} = m_f g - V_f\rho g - \frac{1}{2}\rho A_{cs} C_d \dot{z}|\dot{z}| - C_m V_f \rho \ddot{z} \tag{A.2.10}$$

we substitute A.2.9 for V_f in the buoyancy term, and substitute the approximation $V_n \approx m_f/\rho$ in the added mass term as a reasonable approximation of the float volume as

$$m_f\ddot{z} = m_f g - \left[V_n + (-\theta + \theta_n)\left(\pi r_p^2\right) F_{conv}\right]\rho g - \frac{1}{2}\rho A_{cs} C_d \dot{z}|\dot{z}| - C_m\frac{m_f}{\rho}\rho\ddot{z}. \tag{A.2.11}$$

Moving the V_n out of the buoyancy term and substituting the V_N approximation yields

$$m_f\ddot{z} = \left(m_f g - \frac{m_f}{\rho}\rho g\right) - \left[(-\theta + \theta_n)\left(\pi r_p^2\right) F_{conv}\right]\rho g - \frac{1}{2}\rho A_{cs} C_d \dot{z}|\dot{z}| - C_m\frac{m_f}{\rho}\rho\ddot{z}. \tag{A.2.12}$$

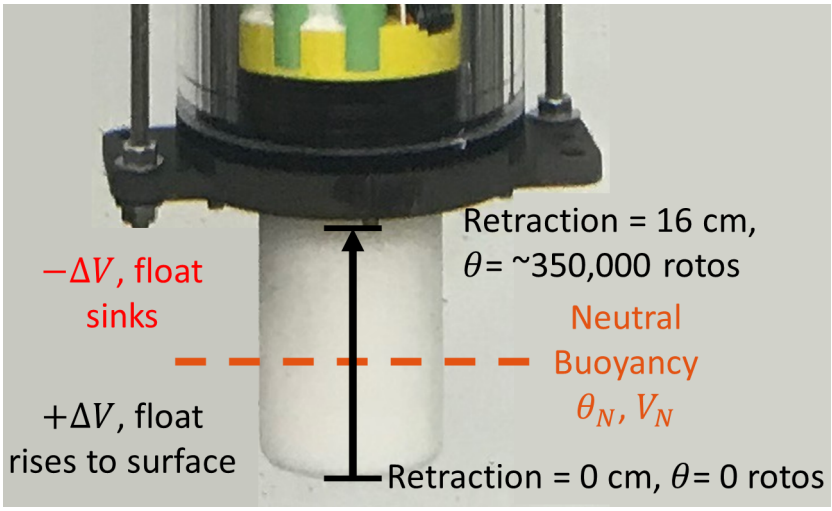

Figure A.1: Physical relationships between the neutral buoyancy piston position, piston position range, and change in volume.

Then, canceling terms, the first term reduces to zero

$$m_f \ddot{z} = -\left[(-\theta + \theta_n)\left(\pi r_p^2\right) F_{conv}\right] \rho g - \frac{1}{2}\rho A_{cs} C_d \dot{z}|\dot{z}| - C_m m_f \ddot{z} \tag{A.2.13}$$

and combining all $\ddot{z}$ terms on LHS yields

$$\left(m_f + C_m m_f\right) \ddot{z} = -\left[(-\theta + \theta_n)\left(\pi r_p^2\right) F_{conv}\right] \rho g - \frac{1}{2}\rho A_{cs} C_d \dot{z}|\dot{z}|. \tag{A.2.14}$$

Finally, distributing the negative sign on the buoyancy term yields the final nonlinear differential equation for the uFloat motion

$$\ddot{z} = \left[(\theta - \theta_n)\left(\pi r_p^2\right) F_{conv}\right] \frac{\rho g}{\left(m_f + C_m m_f\right)} - \frac{1}{2}\frac{\rho A_{cs} C_d}{\left(m_f + C_m m_f\right)} \dot{z}|\dot{z}|. \tag{A.2.15}$$

Appendix B

LINEARIZING THE HYDRODYNAMIC MODEL

For the LQG control method, the uFloat nonlinear hydrodynamic model must be converted to linear, first order differential equations. This requires linearizing the single nonlinear term and reducing the hydrodynamic differential equation A.2.15 from second order to a pair of coupled first order differential equations.

B.1 Taylor Series Approximation of Nonlinear Term

The only nonlinear portion of the uFloat hydrodynamic model is the last term of equation A.2.15

$$f(\dot{z}) = \frac{1}{2}\frac{\rho A_{cs} C_d}{(m_f + C_m m_f)}\dot{z}|.\dot{z}| \tag{B.1.1}$$

The state variable $\dot{z}$ can be rewritten as a squared term, if the sign of the velocity is included as a separate expression

$$f(\dot{z}) = \frac{1}{2}\left(\frac{\rho A_{cs} C_d}{m_f + C_m m_f}\right)\dot{z}^2 \; sign(\dot{z}). \tag{B.1.2}$$

For ease of manipulation, we introduce a constant, c_2, as

$$c_2 = \left(\frac{\rho A_{cs} C_d}{m_f + C_m m_f}\right) \tag{B.1.3}$$

such that,

$$f(\dot{z}) = \frac{1}{2}c_2\dot{z}^2 \; sign(\dot{z}). \tag{B.1.4}$$

We linearize this equation by a Taylor Series approximation with the form of

$$f(x) \approx f(a) + \frac{(x-a)}{1!}\frac{df}{dx}\bigg|_{x=a} + \frac{(x-a)^2}{2!}\frac{d^2 f}{dx^2}\bigg|_{x=a} + \frac{(x-a)^3}{3!}\frac{d^3 f}{dx^3}\bigg|_{x=a} + H.O.T... \tag{B.1.5}$$

To calculate the first three terms requires the first two derivatives of Equation B.1.4 with respect to $\dot{z}$

$$\frac{df}{d\dot{z}} = c_2\dot{z} \tag{B.1.6}$$

and

$$\frac{d^2f}{d\dot{z}^2} = c_2. \tag{B.1.7}$$

Substituting the function $f(\dot{z})$ and its derivatives into the Taylor Series approximation creates a linearized version of the term about a point a, where a is a the most recent estimate of velocity as

$$f(\dot{z}) \approx \frac{1}{2}c_2a^2\ sign(a) + (\dot{z} - a)\,(c_2a)\,. \tag{B.1.8}$$

Rearranging terms yields,

$$f(\dot{z}) \approx \frac{1}{2}c_2a^2\ sign(a) + (\dot{z})\,(c_2a) - \left(c_2a^2\right) \tag{B.1.9}$$

which can be further simplified as

$$f(\dot{z}) \approx (\dot{z})c_2a + \left[\frac{1}{2}c_2sign(a) - c_2\right]a^2. \tag{B.1.10}$$

B.2 Substituting the Taylor Series Approximation

Starting with the uFloat hydrodynamic nonlinear equation A.2.15

$$\ddot{z} = \left[(\theta - \theta_n)\left(\frac{0.0254\ \pi r_p^2}{57,316.6}\right)\left(\frac{\rho g}{m_f + C_mV_n\rho}\right)\right] - \frac{1}{2}\frac{\rho A_{cs}C_d}{(m_f + C_mV_n\rho)}\dot{z}|\dot{z}| \tag{B.2.11}$$

we substitute c_2 and define an additional constant c_1 as

$$c_1 = \left(\frac{0.0254\ \pi r_p^2}{57,316.6}\right)\left(\frac{\rho g}{m_f + C_mV_n\rho}\right). \tag{B.2.12}$$

This simplifies Equation A.2.15 to

$$\ddot{z} = [(\theta - \theta_n)c_1] - \left(\frac{1}{2}c_2\dot{z}|\dot{z}|\right) \tag{B.2.13}$$

The Taylor series approximation for the final term can now be substituted in for the last term to produce a μFloat hydrodynamic model in a linearized form,

$$\ddot{z} = [(\theta - \theta_n)c_1] - \left((\dot{z})c_2a + \left[\frac{1}{2}c_2sign(a) - c_2\right]a^2\right). \tag{B.2.14}$$

B.3 Coupled First Order Differential Equations

The linearized μFloat hydrodynamic model and the actuator model can now be written in matrix form as

$$\begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{z} \\ [(\theta - \theta_n)c_1] - \left(\dot{z}c_2 a + \left[\frac{1}{2}c_2 sign(a) - c_2\right] a^2\right) \\ q_1 s_m + q_2 \end{bmatrix} \tag{B.3.15}$$

Expanding the terms yields

$$\begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{z} \\ \theta c_1 - \theta_n c_1 - \dot{z}c_2 a - \left[\frac{1}{2}c_2 sign(a) - c_2\right] a^2 \\ q_1 s_m + q_2 \end{bmatrix} \tag{B.3.16}$$

which can then be rearranged to cluster state variable terms and constant terms as

$$\begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{z} \\ \theta c_1 - \dot{z}c_2 a - \left(\theta_n c_1 + \left[\frac{1}{2}c_2 sign(a) - c_2\right] a^2\right) \\ q_1 s_m + q_2 \end{bmatrix}. \tag{B.3.17}$$

Placing constants on LHS, still in terms of physical variables yields

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} \dot{z} \\ \ddot{z} + \left(\theta_n c_1 + \left[\frac{1}{2}c_2 sign(a) - c_2\right] a^2\right) \\ \dot{\theta} - q_2 \end{bmatrix} = \begin{bmatrix} \dot{z} \\ -\dot{z}c_2 a + \theta c_1 \\ q_1 s_m \end{bmatrix}. \tag{B.3.18}$$

Finally, we define state variables in terms of physical variables as

$$\dot{x}_1 = \dot{z} \tag{B.3.19a}$$

$$\dot{x}_2 = \ddot{z} + \left(\theta_n c_1 + \left[\frac{1}{2}c_2 sign(a) - c_2\right] a^2\right) \tag{B.3.19b}$$

$$\dot{x}_3 = \dot{\theta} - q_2. \tag{B.3.19c}$$

B.4 Converting from state variables to physical variables

The corresponding conversions from $\dot{x}_n$ back to physical variables are

$$\dot{z} = \dot{x}_1 \tag{B.4.20a}$$

$$\ddot{z} = \dot{x}_2 - \left(\theta_n c_2 + \left[\frac{1}{2}c_1 sign(a) - c_1\right] a^2\right) \tag{B.4.20b}$$

$$\dot{\theta} = \dot{x}_3 + q_2. \tag{B.4.20c}$$

These equations can be integrated to find the relationships between x_n and the physical variables $z,\ \dot{z},\ \theta$. These are definite integrals calculated over a time period starting at t_0 and ending at t_f.

The relation between x_1 and z is expressed as

$$\int_{t_0}^{t_f} \dot{z}\ dt = \int_{t_0}^{t_f} \dot{x}_1\ dt \tag{B.4.21a}$$

$$z\ \Big|_{\substack{t=t_f \\ t=t_0}} = x_1\ \Big|_{\substack{t=t_f \\ t=t_0}} \tag{B.4.21b}$$

$$z(t_f) - z(t_0) = x_1(t_f) - x_1(t_0) \tag{B.4.21c}$$

$$z(t_f) = z(t_0) + x_1(t_f) - x_1(t_0). \tag{B.4.21d}$$

Similarly, the relation between x_2 and $\dot{z}$ is expressed as

$$\int_{t_0}^{t_f} \ddot{z}\ dt = \int_{t_0}^{t_f} \dot{x}_2 - \left(\theta_n c_2 + \left[\frac{1}{2}c_1 sign(a) - c_1\right] a^2\right)\ dt \tag{B.4.22a}$$

$$\dot{z}\ \Big|_{\substack{t=t_f \\ t=t_0}} = x_2 - \left(\theta_n c_2 + \left[\frac{1}{2}c_1 sign(a) - c_1\right] a^2\right)\ t\ \Big|_{\substack{t=t_f \\ t=t_0}} \tag{B.4.22b}$$

$$\dot{z}(t_f) - \dot{z}(t_0) = x_2(t_f) - x_2(t_0) - \left(\theta_n c_2 + \left[\frac{1}{2}c_1 sign(a) - c_1\right] a^2\right)(t_f - t_0). \tag{B.4.22c}$$

Since $(t_f - t_0) = dt$, we obtain the final result:

$$\dot{z}(t_f) = \dot{z}(t_0) + x_2(t_f) - x_2(t_0) - \left(\theta_n c_2 + \left[\frac{1}{2}c_1 sign(a) - c_1\right] a^2\right) dt \tag{B.4.22d}$$

Finally, the relation between x_3 and θ is expressed as

$$\int_{t_0}^{t_f} \dot{\theta}\ dt = \int_{t_0}^{t_f} \dot{x}_3 + q_2\ dt \tag{B.4.23a}$$

$$\theta\ \Big|_{\substack{t=t_f \\ t=t_0}} = (x_3 + q_2 t)\ \Big|_{\substack{t=t_f \\ t=t_0}} \tag{B.4.23b}$$

$$\theta(t_f) - \theta(t_0) = x_3(t_f) - x_3(t_0) + q_2(t_f - t_0) \tag{B.4.23c}$$

Again, $(t_f - t_0) = dt$, so

$$\theta(t_f) = \theta(t_0) + x_3(t_f) - x_3(t_0) + q_2 dt. \tag{B.4.23d}$$

B.5 Summary of conversions between physical variables and state variables

The relations between x_1 and z are

$$z(t_f) = z(t_0) + x_1(t_f) - x_1(t_0) \tag{B.5.24a}$$

$$x_1(t_f) = x_1(t_0) + z(t_f) - z(t_0). \tag{B.5.24b}$$

The relations between x_2 and $\dot{z}$ are

$$\dot{z}(t_f) = \dot{z}(t_0) + x_2(t_f) - x_2(t_0) - \left(\theta_n c_2 + \left[\frac{1}{2} c_1 sign(a) - c_1\right] a^2\right) dt \tag{B.5.25a}$$

$$x_2(t_f) = x_2(t_0) + \dot{z}(t_f) - \dot{z}(t_0) + \left(\theta_n c_2 + \left[\frac{1}{2} c_1 sign(a) - c_1\right] a^2\right) dt \tag{B.5.25b}$$

The relations between x_3 and θ are

$$\theta(t_f) = \theta(t_0) + x_3(t_f) - x_3(t_0) + q_2 dt \tag{B.5.26a}$$

$$x_3(t_f) = x_3(t_0) + \theta(t_f) - \theta(t_0) - q_2 dt \tag{B.5.26b}$$

We note that conversion of the initial condition of the physical variables (z, $\dot{z}$, θ) to the initial condition of the state variable (x_1, x_2, x_3) would be a simple equality.

B.6 State Space Form of Linear Model

It is helpful to place the differential equations into the canonical state space form of matrices and vectors, with the state equation (Equation B.6.27a) and an output equation (B.6.27b) as

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{B.6.27a}$$

and

$$\mathbf{y} = C\mathbf{x} + D\mathbf{u}. \tag{B.6.27b}$$

B.6.1 State Equation

For the purposes of the LQG controller, the differential equations

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} \dot{z} \\ \ddot{z} + \left(\theta_n c_1 + \left[\frac{1}{2}c_2 sign(a) - c_2\right] a^2\right) \\ \dot{\theta} - q_2 \end{bmatrix} = \begin{bmatrix} \dot{z} \\ -\dot{z}c_2 a + \theta c_1 \\ q_1 s_m \end{bmatrix} \tag{B.6.28}$$

can be rewritten in the matrix and vector form

$$\begin{bmatrix} \dot{z} \\ \ddot{z} + \left(\theta_n c_1 + \left[\frac{1}{2}c_2 sign(a) - c_2\right] a^2\right) \\ \dot{\theta} - q_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -c_2 a & c_1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \\ \theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ q_1 \end{bmatrix} \begin{bmatrix} s_m \end{bmatrix}. \tag{B.6.29}$$

Substituting in the state variables (x_1, x_2, x_3), the derivatives ($\dot{x}_1$, $\dot{x}_2$, $\dot{x}_3$) and input variable (u_1) yields

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -c_2 a & c_1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ q_1 \end{bmatrix} \begin{bmatrix} u_1 \end{bmatrix}. \tag{B.6.30}$$

B.6.2 Output Equation

Before using the output equation, we convert the state variables x_1, x_2, x_3 back to the physical parameters z, $\dot{z}$, θ using the equations in Appendix B.5 as

$$\begin{bmatrix} P \\ \theta \end{bmatrix} = \begin{bmatrix} \rho g & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \\ \theta \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} \begin{bmatrix} s_m \end{bmatrix}. \tag{B.6.31}$$

Appendix C

ACTUATOR MODEL

The dynamic system requires a model of the actuator (motor, lead screw, piston) behavior. The important input to this system is the commanded motor speed in units of "motos", with a range of -3200 to 3200 motos. The output of the actuator system is measured by the motor encoder in units of "rotos", with 4,776.4 rotos per motor revolution.

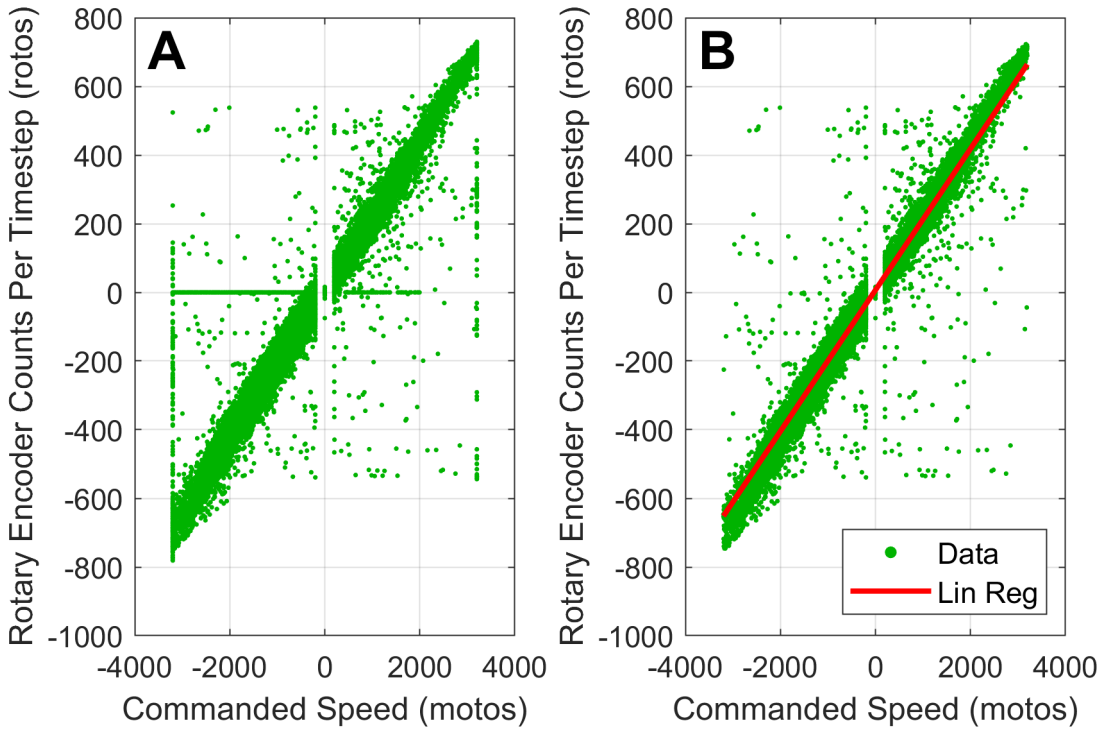

Figure C.1: (A) Field data (green) comparing commanded speed at start of time step to total motor rotation during the time step. (B) Data points along x = [-3200, 3200] and y = [0] removed from set, then linear regression value calculated (red).

C.1 Linear Equation from Field Data

Figure C.1 shows the relationship between the actuator input of commanded motor speed (s_m) in units of "motos" to the output of the angular rotation (θ) measured by the motor encoder in units of "rotos". The data has a few regions where measurements fall outside the linear trend. Removing these data points at the boundaries along the lines $x = [-3200, 3200]$ and $y = 0$ provides a reasonably linear relationship (Figure C.1B). A linear regression of the data points remaining after the removal of the outlying data points provides an equation of the form

$$y = mx + b. \tag{C.1.1}$$

Repeating this process for each μFloat used during the January 13, 2020 field testing on Lake Washington gives a mean linear approximation of

$$\theta \text{ rotos} = \frac{0.20559 \text{ rotos}}{1 \text{ motos}}(s_m \text{ motos}) + 8.2657 \text{ rotos}. \tag{C.1.2}$$

To be used as a differential equation for the state variable derivative $\dot{\theta}$, we can rewrite θ in terms of its velocity ($\dot{\theta}$) times the length of the time step between measurements (dt_θ),

$$(\dot{\theta})(dt_\theta) = 0.20559 s_m + 8.2657 \tag{C.1.3}$$

The dynamic model needs an equation for the state variable's first time derivative ($\dot{\theta}$), which can be found by dividing both sides of the equation by the measurement time step,

$$\dot{\theta} \ = \frac{0.20559}{dt_\theta} s_m + \frac{8.2657}{dt_\theta} \tag{C.1.4}$$

Giving the coefficients a simplified symbol to use in differential equations and state space form, we can express the motos-rotos relationship as

$$\dot{\theta} \ = q_1 s_m + q_2, \tag{C.1.5}$$

where $dt_\theta = 0.100$, $q_1 = 0.20559/dt_\theta$, and $q_2 = 8.2657/dt_\theta$.

Appendix D

ESTIMATED KALMAN FILTER DERIVATION

The Extended Kalman Filter is a nonlinear, sequential state estimation method that can use the nonlinear differential equations developed in Appendix A. The Extended Kalman Filter (EKF) method is detailed in Crassidis and Junkin's textbook Optimal Estimation of Dynamic Systems [3]. The method is summarized in their Table 3.9, reproduced here as Figure D.1.

Model	$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) + G(t)\,\mathbf{w}(t),\ \mathbf{w}(t) \sim N(\mathbf{0}, Q(t))$ $\tilde{\mathbf{y}}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k,\ \mathbf{v}_k \sim N(\mathbf{0}, R_k)$
Initialize	$\hat{\mathbf{x}}(t_0) = \hat{\mathbf{x}}_0$ $P_0 = E\left\{\tilde{\mathbf{x}}(t_0)\,\tilde{\mathbf{x}}^T(t_0)\right\}$
Gain	$K_k = P_k^- H_k^T(\hat{\mathbf{x}}_k^-)[H_k(\hat{\mathbf{x}}_k^-)P_k^- H_k^T(\hat{\mathbf{x}}_k^-) + R_k]^{-1}$ $H_k(\hat{\mathbf{x}}_k^-) \equiv \left.\dfrac{\partial \mathbf{h}}{\partial \mathbf{x}}\right\|_{\hat{\mathbf{x}}_k^-}$
Update	$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k[\tilde{\mathbf{y}}_k - \mathbf{h}(\hat{\mathbf{x}}_k^-)]$ $P_k^+ = [I - K_k H_k(\hat{\mathbf{x}}_k^-)]P_k^-$
Propagation	$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), \mathbf{u}(t), t)$ $\dot{P}(t) = F(t)P(t) + P(t)F^T(t) + G(t)Q(t)G^T(t)$ $F(t) \equiv \left.\dfrac{\partial \mathbf{f}}{\partial \mathbf{x}}\right\|_{\hat{\mathbf{x}}(t), \mathbf{u}(t)}$

Figure D.1: Crassidis Junkins Table 3.9 Extended Kalman Filter algorithm

D.1 Extended Kalman Filter Variables and Model

The Extended Kalman Filter for continuous dynamics and discrete measurements requires a model of the dynamic system ("plant") in the form of equations D.1.1, D.1.2:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), \mathbf{u}(t), t) + G(t)\mathbf{w}(t) \tag{D.1.1}$$

$$\tilde{\mathbf{y}}_k = \mathbf{h}(\hat{\mathbf{x}}_k) + \mathbf{v}_k \tag{D.1.2}$$

The state variables ($\hat{\mathbf{x}}$), output values ($\tilde{\mathbf{y}}_k$), and known input values ($\mathbf{u}$) for this method are:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \end{bmatrix} = \begin{bmatrix} \hat{z} \\ \dot{\hat{z}} \\ \hat{\theta} \end{bmatrix} \tag{D.1.3}$$

$$\hat{\tilde{y}} = \begin{bmatrix} \tilde{y}_1 \\ \tilde{y}_2 \end{bmatrix} = \begin{bmatrix} \tilde{P} \\ \tilde{\theta} \end{bmatrix} \tag{D.1.4}$$

$$\mathbf{u} = \begin{bmatrix} u_1 \end{bmatrix} = \begin{bmatrix} s_m \end{bmatrix} \tag{D.1.5}$$

A hat (ˆ) indicates that the variable is an estimated value. The tilde (˜) indicates a measured value. The state variable vector $\mathbf{x}(t)$ with size n x 1 represents the important states of the dynamic system. For the uFloat, these are depth in the z-direction (z), velocity in z-direction ($\dot{z}$), and position of the piston in terms of angular rotation (θ). The output vector $\tilde{\mathbf{y}}_k$ with size m x 1 represents the measurements that are taken: pressure (P) and piston angular position measured by the motor's rotary encoder (θ). The known control input $\mathbf{u}(t)$ is the motor speed command s_m.

The EKF system model term $\mathbf{f}(\hat{\mathbf{x}}(t), \mathbf{u}(t), t)$ is the uFLoat nonlinear dynamic model based on the neutral buoyancy position as represented by these differential equations:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{z} \\ \left[(\theta - \theta_n)\left(\frac{0.0254\ \pi r_p^2}{57{,}316.6}\right)\left(\frac{\rho g}{m_f + C_m V_n \rho}\right)\right] - \frac{1}{2}\left(\frac{\rho A_{cs} C_d}{m_f + C_m V_n \rho}\right)\dot{z}|\dot{z}| \\ q_1 s_m + q_2 \end{bmatrix} \tag{D.1.6}$$

Substitute in state variables:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} x_2 \\ \left[(x_3 - \theta_n)\left(\frac{0.0254\ \pi r_p^2}{57{,}316.6}\right)\left(\frac{\rho g}{m_f + C_m V_n \rho}\right)\right] - \frac{1}{2}\left(\frac{\rho A_{cs} C_d}{m_f + C_m V_n \rho}\right) x_2 |x_2| \\ q_1 u_1 + q_2 \end{bmatrix} \quad \text{(D.1.7)}$$

The equation's readability can be improved by substituting coefficients the c_1 and c_2 for these constant terms:

$$c_1 = \left(\frac{0.0254\ \pi r_p^2}{57{,}316.6}\right)\left(\frac{\rho g}{m_f + C_m V_n \rho}\right) \quad \text{(D.1.8)}$$

$$c_2 = \left(\frac{\rho A_{cs} C_d}{m_f + C_m V_n \rho}\right), \quad \text{(D.1.9)}$$

resulting in the system model form:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} x_2 \\ [(x_3 - \theta_n)c_1] - \frac{1}{2} c_2 x_2 |x_2| \\ q_1 u_1 + q_2 \end{bmatrix}. \quad \text{(D.1.10)}$$

To account for the physical constraint when the float is at the surface and the piston is extended past the neutral buoyancy point so that the float has positive buoyancy, and zero depth, velocity and acceleration, the system model changes to be

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} \dot{z} \\ \ddot{z} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ q_1 u_1 + q_2 \end{bmatrix}. \quad \text{(D.1.11)}$$

When the piston is retracted past the negative buoyancy position and the float can begin to descend through the water, the model in Equation D.1.10 becomes active again.

D.1.1 Process Noise Modeling

The EKF system model term $G(t)\mathbf{w}(t)$ is system process noise. The EKF method requires that noise must be Gaussian white noise that can be characterized by the standard deviation (σ). The process noise $\mathbf{w}(t)$ has two associated matrices, $G(t)$ and $Q(t)$, both with size n x n.

$G(t)$ is the identity matrix. $Q(t)$ is a diagonal matrix. On the diagonal is the square of the standard deviation of the process noise associated with each state variable. The process noise is usually impossible to measure or analytically characterize, so the Q matrix instead becomes the "tuning knob" of the Extended Kalman Filter estimator. The values on the diagonal of the $Q(t)$ matrix can be altered until reasonable performance is achieved. Later in this appendix there will be a discussion of how the Q matrix relates the covariance matrix $P(t)$ and the EKF gain value K, to help link the Q matrix values to the physical intuition associated with the gain K.

$$G(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{D.1.12}$$

$$Q(t) = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix} \tag{D.1.13}$$

An example of a $Q(t)$ matrix for the uFloat Extended Kalman Filter is:

$$Q(t) = \begin{bmatrix} 0.000005 & 0 & 0 \\ 0 & 0.00005 & 0 \\ 0 & 0 & 100000 \end{bmatrix} \tag{D.1.14}$$

Add the process noise term to get final form of EKF dynamics model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} x_2 \\ [(x_3 - \theta_n)c_1] - \frac{1}{2}c_2 x_2 |x_2| \\ q_1 u_1 + q_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{w}(t), \tag{D.1.15}$$

which matches the general form:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), \mathbf{u}(t), t) + G(t)\mathbf{w}(t). \tag{D.1.16}$$

D.1.2 EKF Output Model

The output equation D.1.2 includes the discrete measurement vector ($\tilde{\mathbf{y}}_k$), the function relating state variables ($\hat{\mathbf{x}}_k$) to measurements ($\tilde{\mathbf{y}}_k$), and the $\mathbf{v}_k$ term for sensor noise. The output functions are:

$$\tilde{P} = \rho g z + v_k, \tag{D.1.17}$$

$$\tilde{\theta} = \theta + v_k, \tag{D.1.18}$$

leading to a combined matrix and vector form of

$$\begin{bmatrix} \tilde{y}_{1,k} \\ \tilde{y}_{2,k} \end{bmatrix} = \begin{bmatrix} \rho g \hat{x}_{1,k} \\ \hat{x}_{3,k} \end{bmatrix} + \begin{bmatrix} v_{1,k} \\ v_{2,k} \end{bmatrix}, \tag{D.1.19}$$

which matches the general form:

$$\tilde{\mathbf{y}}_k = \mathbf{h}(\hat{\mathbf{x}}_k) + \mathbf{v}_k. \tag{D.1.20}$$

Again the noise must be Gaussian white noise, characterized by the standard deviation (σ). The measurement noise $\mathbf{v}(k)$ has an associated matrix $R(t)$ with size m x m. This is a diagonal matrix and the square of the standard deviation of each sensor's measurement noise would go on the diagonal in the appropriate row corresponding to the measurement in the $\tilde{\mathbf{y}}$ vector. The entries on the diagonal must be positive definite (real numbers greater than zero). The values for the μFloat system were determined to be:

$$R = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} 428^2 & 0 \\ 0 & 1^2 \end{bmatrix}, \tag{D.1.21a}$$

where:

$$\sigma_1 = \textit{standard deviation of pressure sensor measurements}, \tag{D.1.21b}$$

$$\sigma_2 = \textit{standard deviation of rotary encoder measurements}, \tag{D.1.21c}$$

and the standard deviation is defined as:

$$\sigma = \sqrt{\frac{1}{N-1}\sum (y_i - \tilde{y}_i)^2}. \tag{D.1.21d}$$

To develop a standard deviation for the pressure sensor measurements, we use a section of data where the floats are at the surface and the pressure is likely to be uniform for the time period. We calculate the mean of the pressure series as the likely true pressure, and then calculate the standard deviation of the measurements away from the mean. The rotary encoder is more difficult to characterize, and was assumed to be quite accurate, with a low standard deviation. The value of 1 roto was chosen, and experimenting with changing this value by multiple degrees of magnitude larger and smaller showed very little impact on the estimator performance. This leads to the assumption that the standard deviation value of 1 roto is reasonable, but if greater accuracy becomes necessary in the future, work could be done to obtain an experimentally determined standard deviation value for the rotary encoder.

D.2 Extended Kalman Filter Stages

D.2.1 Initialization of State Values and Error Covariance Matrix

The method requires an initial guess for the state variables. It is helpful if this is an accurate guess, but the EKF can recover if we make an inaccurate guess. I generally edit the data time series to start at a point where the float is on the surface with the piston at zero retraction, as this provides an easily known initial state: $z = 0$, $\dot{z} = 0$, $\theta = 0$,

$$\hat{\mathbf{x}}(t_0) = \hat{\mathbf{x}}_0 = [0; 0; 0]. \tag{D.2.22}$$

The method also requires an initial condition for the error covariance matrix P_0. This should be based on the confidence in the estimate of the initial state $\hat{\mathbf{x}}_0$. If we think the initial state estimate is good, P_0 should be small. If we think there is a lot of error in the initial state estimate, P_0 should be larger [3].

The formal definition of the covariance is the expected value of x_{err} squared [8],

$$P_0 = E\left(\mathbf{x}_{err}(t_0)\mathbf{x}_{err}^T(t_0)\right). \tag{D.2.23}$$

The matrix form of the error covariance initial value P_0 is

$$P_0 = Initial\ error\ covariance\ matrix = \begin{bmatrix} C_{p1} & 0 & 0 \\ 0 & C_{p2} & 0 \\ 0 & 0 & C_{p3} \end{bmatrix}. \tag{D.2.24}$$

The physical meaning of the P_0 matrix is the confidence in the initial guess:

- If the guess for x_0 (the initial estimate of the state variables at t_0) is good, the associated C_p should be small.
- If there is likely to be error in the guess for x_0, make the associated C_p larger.

An appropriate P_0 value for the uFloat EKF is:

$$P_0 = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \tag{D.2.25}$$

D.2.2 Gain Calculation

With the initialization steps completed, the method becomes a for loop that calculates the Gain, Update, and Propagation steps at each time step to generate an updated state estimate.

The gain calculation first needs the H_k matrix,

$$H_k(\hat{\mathbf{x}}_k^-) \equiv \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^-} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \frac{\partial h_1}{\partial x_3} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \frac{\partial h_2}{\partial x_3} \end{bmatrix}, \tag{D.2.26}$$

evaluated at the previous state estimate $\hat{\mathbf{x}}_k^-$ to use in calculating Kalman gain K_k. In the specific case of the uFloat, the H_k matrix is not dependent on any changing values, and could be calculated once in the initialization stage instead of at every time step. The definition of $\mathbf{h}(\mathbf{x}_k)$ comes from equation D.1.19:

$$\mathbf{h}(\mathbf{x}_k) = \begin{bmatrix} \rho g x_1 \\ x_3 \end{bmatrix}, \tag{D.2.27}$$

and therefore the H_K matrix is

$$H_k = \begin{bmatrix} \rho g & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{D.2.28}$$

The next step is to calculate the gain K matrix,

$$K_k = P^-_{cov,k} H_k^T [H_k P^-_{cov,k} H_k^T + R]^{-1} \tag{D.2.29}$$

.

In this system, H_k and R are both a constant matrix, so the gain K_k varies only with the covariance matrix $P^-_{cov,k}$. The explanation for what terms go into the calculation of the covariance matrix at each time step is below in section D.2.4. The physical intuition of the gain K is:

- very small K: measurements are ignored, heavy weighting on model.
  - This is a good K value if measurements are very noisy (inaccurate) and the model is very good.
  - This is a bad K value if measurements are reliable and model is not very accurate.
- large K: weights the measurements more than the model prediction.
- very large K: heavy weighting on measurement, model ignored.
  - This is a good K value if the measurements are good and the model is not very accurate.
  - This is a bad K value if the measurements are very noise or inaccurate and the model is pretty good, can result in too much noise entering estimate.

D.2.3 Update

For the Update stage, first calculate the output value $\mathbf{h}(\hat{\mathbf{x}}_k^-)$ using the previous state estimate $\hat{\mathbf{x}}_k^-$:

$$\mathbf{h}(\hat{\mathbf{x}}_k^-) = \begin{bmatrix} \rho g \hat{x}_{1,k}^- \\ \hat{x}_{3,k}^- \end{bmatrix}. \tag{D.2.30}$$

This value $\mathbf{h}(\hat{\mathbf{x}}_k^-)$ will be subtracted from the latest sensor measurements ($\tilde{\mathbf{y}}_k$) of pressure ($\tilde{P}$) and the piston's angular position ($\tilde{\theta}$) to find the error between the current sensor measurements and the previous estimate of the pressure and rotary encoder.

Next, an updated state estimate $\hat{\mathbf{x}}_k^+$ is calculated by summing the previous state estimate $\hat{\mathbf{x}}_k^-$ and the newly calculated gain K_k multiplied by the error:

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + K_k \left[\tilde{\mathbf{y}}_k - \mathbf{h}(\hat{\mathbf{x}}_\mathbf{k}^-)\right] \tag{D.2.31}$$

Finally, an updated covariance matrix P_k^+ is calculated using H_k (Equation D.2.28), gain K_k (Equation D.2.29), and the previous covariance matrix P_k^-,

$$P_k^+ = \left[I_{(nxn)} - K_k H_k\right] P_k^-. \tag{D.2.32}$$

D.2.4 Propagation

The final step in Table 3.9 is the propagation of the state estimate and covariance matrix forward to the next time step. This uses the differential equation representing the first derivatives of the state variables (eqns D.1.10, repeated below for ease of reading) and the derivative of the covariance matrix (Equation D.2.41). These derivatives are integrated forward in time by one time step (dt) with a numerical integrator such as ode45 in Matlab, or a similar numerical integration scheme like Runge-Kutta 4.

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), \mathbf{u}(t), t) \tag{D.2.33}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} x_2 \\ [(x_3 - \theta_n)c_1] - \frac{1}{2}c_2 x_2 |x_2| \\ q_1 u_1 + q_2 \end{bmatrix}. \tag{D.2.34}$$

Before integrating the covariance matrix P forward in time, first calculate a matrix $F(t)$ that represents a linearized first-order Taylor series expansion of the state derivatives evaluated at the updated state estimate $\hat{\mathbf{x}}_k^+$:

$$F(t) \equiv \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^+, \mathbf{u}_k} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \end{bmatrix} \tag{D.2.35}$$

The functions $\mathbf{f}$ are the derivatives of the state variables, Equation D.2.34. Taking the derivative of each f_n equation with respect to each state variable:

$$f_1 = \dot{x}_1 = x_2 \tag{D.2.36a}$$

$$\frac{\partial f_1}{\partial x_1} = 0, \quad \frac{\partial f_1}{\partial x_2} = 1, \quad \frac{\partial f_1}{\partial x_3} = 0 \tag{D.2.36b}$$

The second line in the system model is slightly more complex, and can be expanded and rewritten before taking partial derivatives:

$$f_2 = \dot{x}_2 = [(x_3 - \theta_n)c_1] - \frac{1}{2}c_2 x_2 |x_2| \tag{D.2.37a}$$

$$f_2 = \dot{x}_2 = x_3 c_1 - \theta_n c_1 - \frac{1}{2}c_2 x_2^2 \; sign(x_2) \tag{D.2.37b}$$

$$\frac{\partial f_2}{\partial x_1} = 0, \quad \frac{\partial f_2}{\partial x_2} = -c_2 x_2, \quad \frac{\partial f_2}{\partial x_3} = c_1 \tag{D.2.37c}$$

The third line in the system model

$$f_3 = \dot{x}_3 = q_1 u_1 + q_2 \tag{D.2.38a}$$

$$\frac{\partial f_3}{\partial x_1} = 0, \quad \frac{\partial f_3}{\partial x_2} = 0, \quad \frac{\partial f_3}{\partial x_3} = 0 \tag{D.2.38b}$$

This provides a final $F(t)$ matrix with the form:

$$F(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -c_2 x_2 & c_1 \\ 0 & 0 & 0 \end{bmatrix} \tag{D.2.39}$$

Substituting the values of c_1 and c_2 back into the matrix:

$$F(t) \equiv \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k^+, \mathbf{u}_k} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & \frac{-\rho A_{cs} C_d}{\left(m_f + C_m V_n \rho\right)} x_2 & \left(\frac{0.0254\ \pi r_p^2}{57{,}316.6}\right) \frac{\rho g}{\left(m_f + C_m V_n \rho\right)} \\ 0 & 0 & 0 \end{bmatrix} \tag{D.2.40}$$

The derivative function for the covariance matrix $P(t)$ is:

$$\dot{P}(t) = F(t)P(t) + P(t)F^T(t) + G(t)Q(t)G^T(t) \tag{D.2.41}$$

$F(t)$ was found above in Equation D.2.39, the initial value for P(t) is the calculated value P_k^+ from the Update step. The G(t) and Q(t) matrices are related to the process noise term in the original model (Equation D.1.1), repeated here:

$$\dot{\hat{\mathbf{x}}}(t) = f(\hat{\mathbf{x}}(t), \mathbf{u}(t), t) + G(t)\mathbf{w}(t)$$

$G(t)$ is an n x n diagonal matrix, and can be set to the identity matrix. This represents that every state variable has process noise associated with the dynamic process.

$$G(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{D.2.42}$$

It is usually not possible to know, measure, or estimate the process noise $\mathbf{w}(t)$. Therefore we cannot define a standard deviation of the process noise in the way we did for the measurement noise matrix R. Instead, the Q matrix related to the process noise is used as the "tuning knob" for the Extended Kalman Filter. The Q matrix is therefore a diagonal

matrix with different coefficient values for each state. These values must be greater than or equal to zero ($q_n \geq 0$).

$$Q(t) = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & q_2 & 0 \\ 0 & 0 & q_3 \end{bmatrix} \tag{D.2.43}$$

An example of an appropriate $Q(t)$ matrix for the uFloat EKF is:

$$Q(t) = \begin{bmatrix} 0.000005 & 0 & 0 \\ 0 & 0.00005 & 0 \\ 0 & 0 & 100000 \end{bmatrix} \tag{D.2.44}$$

The Q matrix is part of the calculation of the covariance matrix P, which is included in the calculation of the gain K and the Q matrix therefore impacts the gain K. The intuition for the gain K was that K determines whether the next state estimate weights more heavily the measurements, or the model.

Once the integration of the state and covariance forward in time is complete, the final step in the for-loop is to update the variables for the state ($\hat{\mathbf{x}}_k^-$)and covariance (P_k^-) to the new value generated by the integration process as the estimate for where the system will be at the next time step and the next iteration of the for loop.

Appendix E

PID CONTROL EQUATION

E.1 General form of PID

The discrete form of the Proportional-Integral-Derivative (PID) controller can be written as

$$u_{t_k} = P_{term} + I_{term} + D_{term}, \tag{E.1.1}$$

which expands to

$$u_{t_k} = k_p\left(e_{t_k}\right) + k_i\left(\sum_{n=1}^{k} e_{t_n}\Delta t\right) + k_d\left(\frac{e_{t_k} - e_{t_{k-1}}}{\Delta t}\right) \tag{E.1.2}$$

where u_{t_k} is the actuator command, k_p, k_i, and k_d are the gain values chosen by the designer, and Δt is the time between updates $(t_k - t_{k-1})$. The error at time t_k is calculated as

$$e_{t_k} = y_{ref} - y_{t_k}, \tag{E.1.3}$$

where y_{ref} is the reference or desired value of the operating signal and y_{t_k} is measured or estimated value of the operating signal. The integral term is the the accumulated error: the summation of the error at all previous time steps, multiplied by the Δt time interval.

VITA

Jessica Noe is a mechanical engineer in Seattle, WA. She has worked on marine engineering and dynamic control projects at the University of Washington since 2014 as an undergraduate student, graduate student, and as a staff engineer for the Marine Renewable Energy Laboratory in the Department of Mechanical Engineering.