The Pennsylvania State University
The Graduate School

MODEL-SCALE EVALUATION OF AUTONOMOUS SHIP LANDING GUIDANCE AND CONTROL MODES FOR ROTORCRAFT

A Dissertation in
Aerospace Engineering
by
Christopher M. Hendrick

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2024

The dissertation of Christopher M. Hendrick was reviewed and approved by the following:

Joseph F. Horn
Professor of Aerospace Engineering
Dissertation Advisor
Chair of Committee

Jack W. Langelaan
Professor of Aerospace Engineering
Director of Graduate Program

Eric N. Johnson
Professor of Aerospace Engineering

Christopher D. Rahn
J. Lee Everett Professor of Mechanical Engineering

Amy R. Pritchett
Professor of Aerospace Engineering
Department Head

Abstract

Ship landing in high sea states is a challenge for both manned and unmanned rotorcraft. A system that provides reliable autonomous recovery of ship-based rotorcraft could reduce mishap rates and reduce costs associated with training and certification testing. Such a system might also increase operational capability by allowing operations in more severe wave and wind conditions. These potential benefits have motivated a significant amount of public domain research on autonomous ship landing algorithms for rotorcraft. Existing works have primarily been simulation based, however, and the rigorous experimental evaluation of advanced landing algorithms is lacking in the public-domain literature. This is due in part to the inaccessibility of full-scale testing. Model-scale testing, on the other hand, offers a more accessible test bed for vetting autonomous landing solutions and has therefore been utilized in all existing openly available experimental studies. These studies have not considered the scaling of either the closed-loop aircraft dynamics or ship motions, however, meaning key aspects of the full-scale landing scenario may not be realistically represented at model-scale.

The objective of this research was to develop a methodology for performing dynamically scaled autonomous ship landing experiments, and then to use the proposed scaling method to perform a rigorous experimental analysis of advanced autonomous landing guidance and control modes. Toward this end, Froude scaling is proposed for relating aircraft closed-loop dynamics and ship motions across test scales and the validity of this method is then analyzed. Two representative landing guidance algorithms were then developed and experimentally evaluated using the proposed scaling methodology. The first is an advanced landing strategy that uses quadratic programming (QP) optimization to plan the landing path to a forecasted deck state. The second is a simpler "baseline" guidance method that tracks deck motions while closing the distance between the aircraft and deck at a constant rate. Both guidance algorithms command position and heading to an explicit model following (EMF) control law.

The guidance algorithms were first evaluated in experiments conducted in the Maneuvering and Seakeeping Basin (MASK) located at the Naval Surface Warfare Center Carderock Division. During these experiments, control law parameters were modified to impose artificial constraints on the maneuverability of the aircraft, providing insight into how well both guidance methods can cope with a less agile airframe. The results showed that the predictive landing strategy allowed for more direct landing paths to be planned when compared to the baseline algorithm and can also allow for landings to be performed with lower control bandwidth. The baseline guidance algorithm, on the other hand, proved to be both simple and reliable when the UAV was in high bandwidth

configurations, but may not be feasible for aircraft with limited control authority that must land in moderate to high sea states.

The experimental setup in the MASK did not include aerodynamic disturbances. Additional flight tests were therefore performed in the Penn State Indoor Flight Facility to determine if the QP guidance algorithm offers any advantage over the baseline method when operating in the presence of a significant aerodynamic disturbance. The experimental results obtained at Penn State did not indicate that the QP algorithm provides a definitive advantage over the baseline algorithm in terms of matching deck position, velocity, and attitude at landing while operating in a gusty environment. The QP algorithm did allow for landings with a shorter duration, however, resulting in lower total control usage due to less time operating in aerodynamic disturbances.

Table of Contents

List of Figures

List of Tables

Nomenclature

Symbols

g	=	gravitational acceleration
$\tilde{G}_{xx}$	=	rough autospectrum for signal x
$\hat{G}_{xx}$	=	smooth autospectrum for signal x
$\hat{G}_{xx,c}$	=	composite autospectrum for signal x
$\tilde{G}_{xy}$	=	rough cross spectrum for signals x and y
$\hat{G}_{xy}$	=	smooth cross spectrum for signals x and y
$\hat{G}_{xy,c}$	=	composite cross spectrum for signals x and y
J	=	system ID cost function, QP optimization cost function
j	=	jerk (acceleration time derivative)
K_{HPF}	=	deck filter high pass channel gain for baseline guidance algorithm
K_P	=	proportional control gain
K_I	=	proportional control gain
K_D	=	proportional control gain
M	=	aircraft mass
N_F	=	Froude scaling factor
N_{lag}	=	number of lagged outputs in AR model
p	=	roll rate
q	=	pitch rate, quaternion
Q	=	QP cost function output weighting matrix
Q_Δ	=	QP cost function jerk weighting matrix
r	=	yaw rate
R	=	QP cost function input weighting matrix
s	=	Laplace variable
S	=	QP cost function terminal output error weighting matrix

S_Δ	=	QP cost function terminal jerk weighting matrix
t	=	time
t_{land}	=	desired land time for QP guidance algorithm
T_g	=	duration of tau guidance trajectory
$T_{b/a}$	=	rotation matrix from a to b coordinates
$\vec{u}$	=	control vector
u_X	=	X position command filter input
u_Y	=	Y position command filter input
u_Z	=	Z position command filter input
$\vec{v}$	=	white noise vector
w	=	body axis vertical rate
$\vec{x}$	=	state vector
X^a	=	X position in frame a
$X(f)$	=	frequency domain representation of system inputs provided to CIFER®
$X^*(f)$	=	complex conjugate of $X(f)$
$\vec{y}$	=	output vector
$\vec{y}_{ref}$	=	reference outputs for QP cost function
Y^a	=	Y position in frame a
$Y(f)$	=	frequency domain representation of system outputs provided to CIFER®
Z^a	=	Z position in frame a
α	=	AR model coefficients
$\hat{\gamma}^2_{xy}$	=	coherence function
δ_{ail}	=	lateral input to UAV bare airframe
δ_{ele}	=	longitudinal input to UAV bare airframe
δ_{rud}	=	yaw input to UAV bare airframe
δ_{thr}	=	throttle input to UAV bare airframe
ζ	=	damping ratio
θ	=	pitch angle
τ	=	time delay
ϕ	=	roll angle
χ	=	tau guidance action gap
ψ	=	yaw angle
ω_{LPF}	=	deck filter low pass channel frequency for baseline guidance algorithm
ω_X	=	X position control command filter frequency
ω_Y	=	X position control command filter frequency

ω_Z	=	X position control command filter frequency
ω_θ	=	pitch command filter frequency
ω_ϕ	=	roll command filter frequency
ω_{psi}	=	yaw command filter frequency

Acronyms

AR	=	autoregressive
AFDD	=	Army Aeroflightdynamics Directorate
CIFER®	=	Comprehensive Identification from Frequency Responses software
DDP	=	differential dynamic programming
DRB	=	disturbance rejection bandwidth
DRP	=	disturbance rejection peak
GPU	=	graphics processing unit
ID	=	identification
iLQR	=	iterative LQR
LQR	=	linear quadratic regulator
MASK	=	Maneuvering and Seakeeping Basin
MCA	=	minor component analysis
MPC	=	model predictive control
MPPI	=	model predictive path integral control
NLP	=	nonlinear program
NSWCCD	=	Naval Surface Warfare Center Carderock Division
QP	=	quadratic program
SCONE	=	systematic characterization of the naval environment
UAV	=	unmanned aerial vehicle

Superscripts and Subscripts

0	=	initial condition
cmd	=	command
D	=	deck state
dlf	=	averaged deck level frame
dp	=	deck state prediction
f	=	value at final time

fs	=	full scale
lf	=	UAV level frame
I	=	inertial frame
k	=	value at current sample time
ms	=	model scale
UAV	=	UAV state

Acknowledgments

This work was partially funded by the United States Office of Naval Research (ONR) under grant N00014-20-1-2092. The views and conclusions contained herein are those of the authors only and should not be interpreted as representing those of the ONR, the U.S. Navy, or the U.S. Government.

This journey really started for me when I was an undergrad at the University at Buffalo. For that reason, I want to thank the many great friends I met at UB who helped me make it to grad school in the first place. This includes Zachary Perkins and William Gmoser, who both worked many late nights struggling through the UB undergrad engineering program with me.

When I first went to grad school I had no intention of pursuing a Ph.D., but the coursework and research I was exposed to at Penn State captured my interest and motivated me. The positive experience I have had at Penn State was largely made possible by the opportunity given to me by Dr. Joseph Horn, my advisor, when he took me on as a research assistant in his lab group. I would like to thank Dr. Horn for giving me this opportunity and for his mentorship throughout my graduate education.

The friends and fellow students I have met while working in the Vertical Lift Research Center of Excellence, including ZhouZhou Chen, Ashish Manjhi, Grant Li, Andrew Jue, and Joel Rachaprolu, have also made my time at Penn State more academically enriching and certainly more entertaining.

I would like to acknowledge my dissertation committee members Dr. Jack Langelaan, Dr. Eric Johnson, and Dr. Christopher Rahn for their comments and recommendations when preparing this work. I would also like to thank Dr. Langelaan for his help while testing and advice as co-principal investigator on the ONR project that funded much of this work.

I would also like to thank the other individuals who contributed to this work. This includes my (previously) fellow grad students Emma Jaques and Duncan Nicholson, as well as Anish Sydney, Eric Silberg, and Jared Soltis from the Naval Surface Warfare Center Carderock Division, all of whom were instrumental to the experiments performed for this dissertation. Duncan Nicholson also worked alongside me at the start of this project, and his help setting up UAV hardware and software was instrumental to getting this research off the ground.

Lastly, I would like to thank both of my Parents and my fiancé Lauren for their continued love and support. To my parents – you have been excellent examples of hard

work and have given me endless opportunity, making it possible for me to pursue my interests. Lauren – you moved to Penn State for me, you have had incredible patience while I have worked long and odd hours, and you have always believed in me. For this and so much more, I am grateful.

Chapter 1
Introduction

1.1 Background and Motivation

Rotorcraft have a long history of naval use, with the U.S. Navy accepting its first helicopter (a Sikorsky YR-4B) on October 16th, 1943 [1]. Initial interest in the use of helicopters for sea-based aviation was largely due to their vertical takeoff and landing capabilities eliminating the need for runways, as well as the potential use of helicopters as antisubmarine warfare aircraft. In the decades following the 1940s, the role of rotorcraft in naval aviation greatly expanded. Today manned and unmanned rotorcraft are vital assets used to perform a variety of maritime tasks, including search and rescue, reconnaissance, surveillance, cargo missions, and fire control support.

For all maritime tasks, two segments of the aircraft's mission are launch and recovery. Since the first use of rotorcraft from ships, these close-proximity operations have been identified as posing a significant risk, especially the landing phase. To safely land during moderate to high sea states, a pilot or autonomous control mode must approximately match the state of the deck at touchdown while coping with rough seas, a degraded visual environment, and high winds. The problem is further exacerbated by the turbulent airwake produced by flow over the ship's superstructure, which leads to difficult to predict aerodynamic disturbances. Due to these adverse conditions, the ability to perform safe landings is often the factor that limits the flight envelope. Furthermore, concerns related to landing during different wave and wind conditions contribute to the extensive testing required for both pilot training and certification of sea-based rotorcraft. Particularly troublesome is the certification process. For decades this has required full "dynamic interface" testing, where each rotorcraft-ship combination is flight tested at a full range of wind speeds and azimuths to establish safe launch and recovery envelopes [2]. These tests are expensive and logistically challenging, as tests depend on outdoor weather conditions

and ship and aircraft availability.

To help mitigate the aforementioned issues, various aspects of helicopter-shipboard operation have been actively researched by the rotorcraft community for decades. Over the past three decades, with large increases seen in onboard computational power, one point of interest has been the development of advanced technologies for performing autonomous ship deck landings. This has led to a sizeable body of research on deck state sensing and estimation, as well as autonomous landing guidance and control methods. Much progress has been made in these areas, as has been evidenced by autonomous unmanned aerial vehicles (UAVs) like Northrop Grumman's MQ-8C and Airbus' VSR700 (see Fig. 1.1). Still, there is interest in further developing autonomous landing systems to allow for landing in high sea states, for performing GPS denied landings, and for automating the landing task for piloted aircraft. Such a system could expand flight envelopes, reduce mishap rates, and reduce costs associated with pilot training and certification testing.

(a) Northrop Grumman MQ-8C performing an autonomous landing (from [3]).

(b) Airbus tests VSR700 autonomous landing system using optionally piloted aircraft (from [4]).

Figure 1.1: Example UAVs performing autonomous ship landings.

While there exists a large body of research on autonomous shipboard landing systems, rigorous experimental evaluation of these systems is lacking in the public domain. This is impacted by the fact that flight testing with a large rotorcraft operating from a ship is not feasible for most researchers. Hence, most studies either feature simulations or tests with small-scale UAVs. While testing at full scale is inevitably necessary for validation of a particular landing system, testing at model scale does have advantages that make it an appealing research tool. Key advantages are:

1. Model scale testing offers a low-risk, low-cost test bed for vetting autonomous landing solutions.

2. The effect of individual variables such as wind gusts, sea state, or aircraft handling qualities can be isolated.
3. It is easier to perform a high volume of tests at model scale.

Most currently published model scale tests, however, feature a small number of landings onto platforms with very limited amplitude motion. Furthermore, these tests typically utilize the high control bandwidth of small UAVs while landing on a platform moving with frequency content similar to that of a ship sized for full-scale manned rotorcraft operations. Landing small UAVs on a large ship does have practical use, but for using model scale experiments to gain knowledge applicable to larger aircraft, attention should be paid to the scaling of dynamics across test scales. This work presents a methodology for performing dynamically scaled autonomous ship landing experiments, where both ship motions and key aspects of aircraft closed-loop dynamics are systematically adjusted to model-scale. Model-scale experiments were then performed to analyze the performance of advanced autonomous landing guidance and control modes through high-volume flight testing, giving insight into the potential benefits and limitations of the autonomous landing guidance and control laws. Note that the emphasis here is on guidance and control only, and the associated deck state sensing and estimation problem will not be considered.

1.2 Existing Work

This section gives a review of the existing literature in four main areas relevant to the research discussed in this work: autonomous landing guidance and control laws, deck motion prediction algorithms, dynamic scaling, and existing autonomous landing experimental setups.

1.2.1 Guidance and Control for Autonomous Shipboard Landing

A variety of guidance and control methods have been proposed for autonomous shipboard landing of rotorcraft. The majority of these can either be categorized as model predictive control (MPC) methods or simple deck tracking methods. Additionally, algorithms based on tau guidance theory and optimization of polynomials have been used. This section gives a review of these landing methods. The discussion here will be contained to algorithms that do not require a tether, articulated landing gear, or other similar augmentations to the aircraft or ship. Note that a similar review was given by Pravitra in [5], but the

review given here is warranted for both completeness and tailoring conversation for this work.

1.2.1.1 Model Predictive Control

Many published shipboard landing algorithms can be classified as model predictive controllers. At its core, MPC is an optimal control methodology where the solution to an optimization problem of the form

$$\begin{aligned} \min \quad & J = h(\vec{x}_f) + \int_0^{t_f} g(\vec{x}, \vec{u}) dt \\ \text{s.t.} \quad & \dot{\vec{x}} = f(\vec{x}, \vec{u}) \\ & c(\vec{x}, \vec{u}) \leq \vec{0} \end{aligned} \tag{1.1}$$

is re-computed online. The goal of eq. (1.1) can be described as follows: determine the sequence of controls $\vec{u}$ that drives the states $\vec{x}$ to minimize the objective function J over the time horizon t_f, all while satisfying the system dynamics $\dot{\vec{x}} = f(\vec{x}, \vec{u})$ and constraints $c(\vec{x}, \vec{u}) \leq \vec{0}$. The solution yields an open loop control law that can be applied to the controlled system in one of two ways: by applying the optimal control input directly or by commanding the corresponding state trajectory to a lower-level trajectory following controller. Since open loop control is often not practical, the MPC framework introduces feedback through the process shown in Fig. 1.2: by applying only a small segment of the optimal solution before online re-optimization occurs. Re-planning in this manner provides some level of robustness to deviations from the expected path, changes in the desired final state, and other uncertainties.

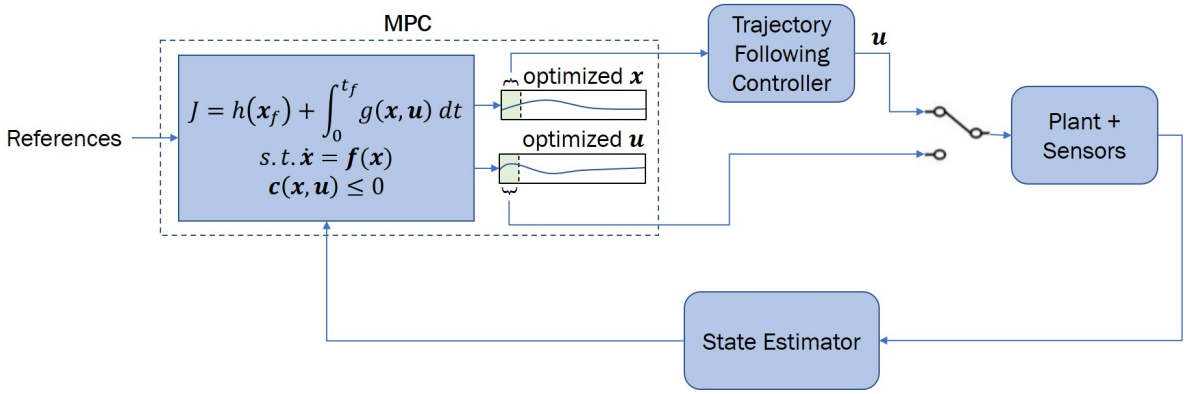


Figure 1.2: MPC architecture. Note that the switch denotes the choice to either command a trajectory tracking controller or to apply the optimal control to the system directly.

A major advantage of MPC is the ability to explicitly include a representation of the system dynamics in the solution, and therefore plan ahead for the expected dynamics. This comes at a cost, though, as optimal control problems are generally difficult to solve and rarely do analytic solutions exist. As a result, the optimal control problem is often discretized and cast to a static optimization problem that is solved online for re-planning. For real-time implementation, this requires the online optimization to be both fast and reliable.

One class of optimal control problems that can be solved analytically is the case of linear dynamics, a quadratic cost function, and no path constraints. With these assumptions, a solution can be found by applying Pontryagin's Minimum Principle and solving a two-point boundary value problem, similar to the Linear Quadratic Regulator (LQR) problem solution. The biggest difference is that, for path planning, the final state and/or time may by fixed, resulting in different boundary conditions than the classic LQR problem. This class of methods was applied to autonomous shipboard landing in [6], where a landing path was generated using second order integrator dynamics to produce position, velocity, and acceleration commands. The final state of the path was fixed to match a prediction of deck state at the specified land time and the optimal controls were re-calculated during landing to account for changes in the predicted deck state. The commands were sent to a dynamic inversion translation rate command - position hold controller, and the landing algorithm was tested with simulations of a helicopter similar to a UH-60 landing on a ship representative of a DDG-51 destroyer. Results showed the optimal control law to be sensitive to deck motion prediction errors.

A limited set of minimum time problems with both linear dynamics and state constraints have well known analytic solutions as well. One example is the minimum time problem with double integrator dynamics and limited accelerations. An approach similar to this was used to land a quadrotor on a moving platform in [7], though this work included nonlinear constraints and an iterative solution procedure was developed.

A method related to LQR that has been more commonly applied to shipboard landing is quadratic programming (QP) based linear MPC. From a high level, the usual process for writing a linear MPC problem as a QP begins by defining a discrete linear model of the controlled dynamics, as well as a quadratic cost function and linear constraints. The optimization problem is then generally posed in one of two ways. The first method, known as "direct transcription", is to include both the states and controls as decision variables and enforce the dynamics through the equality constraints. The second method, known as "direct shooting", is to eliminate the states from the optimization by forward

simulation, thereby reducing the number of decision variables and constraints. Direct shooting makes the optimization problem smaller, but can result in numerical issues for large horizon lengths [8]. In both cases the equations can be manipulated into the standard quadratic programming form, for which multiple fast and reliable optimization routines have been developed. This allows for an approximate solution to the continuous time linear-quadratic problem to be obtained, but with the benefit that path constraints can be included.

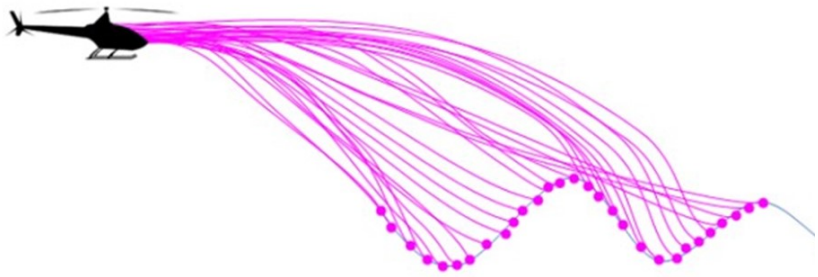
QP based linear MPC was used to formulate receding horizon ship landing controllers in [9–11]. A drawback to the MPC formulation in these works is that the horizon is simply receding and does not change size. Once the landing is nearly complete, the optimization will therefore take into account future states that would not occur until after landing. This can lead the optimization to take greater errors at landing in order to reduce errors that would have occurred after the fact. To address this, the group from [9] formulated a shrinking horizon MPC algorithm (see [12, 13]) that plans the trajectory to a future deck state at some pre-specified land time. The terminal deck state was assumed known a priori in this case, however. Improvements were made to this method in [14], with updates including a varying time horizon (the horizon can grow as well as shrink) and a deck motion prediction scheme to estimate deck state at touchdown. Simulations with these improvements were also conducted in [14], showing successful application of the method to a helicopter landing on a CG-26 destroyer subject to sea state 5 wave conditions. Note that references [9, 11–14] all verified performance via simulation. While these are all simulation based studies, QP based linear MPC has also been applied experimentally for landing small UAVs on translating ground vehicles in [15–17], demonstrating the ability to run reasonably sized QP optimizations in real time.

Two other optimal control methods related to LQR are the iterative linear quadratic regulator (iLQR) and differential dynamic programming (DDP). Both iLQR and DDP can handle nonlinear dynamics, though computational costs can make them more challenging to implement in real time than linear MPC, especially if constraints are included. The idea behind iLQR is as follows: given an initial guess for the optimal control sequence and corresponding state trajectory, the nonlinear dynamics and cost function can be approximated as linear and quadratic, respectively, about the state trajectory. An iterative procedure that includes so-called forward and backward passes is then used to refine the control sequence. The forward pass involves rolling out the state trajectory using the current estimate of the optimal control sequence, and the backward pass uses

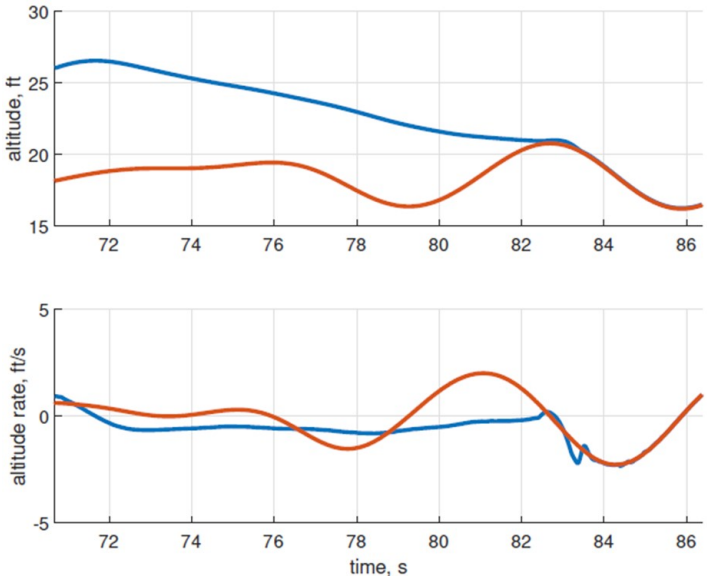
the rolled out states to determine a perturbation to the current control sequence that lowers the cost of the trajectory. The forward and backward passes may then be repeated until convergence. The DDP algorithm is nearly identical to iLQR, but the dynamics are approximated as quadratic about the state trajectory rather than linear. A good overview of unconstrained DDP and iLQR is given in [5].

A MPC algorithm for autonomous ship landing based on iLQR is also outlined in [5]. The algorithm, called ensemble iLQR (EiLQR), leverages parallelized general-purpose computing on graphics processing units (GPUs) to run a multitude of unconstrained iLQR optimizations simultaneously, with each iLQR instance planning for a different land time and hence different terminal deck state. A depiction of this concept is shown in Fig. 1.3a. The terminal deck states are forecast by autoregressive models and the time horizon of the trajectories shrinks as the aircraft approaches. The method was tested with GPU in the loop simulations and flight tests landing on static ground. A sample descent profile from the simulated landings is given in Fig. 1.3b, showing EiLQR is capable of planning a successful landing path with minimal additional maneuvering. Overall, the full results reported in [5] suggest that EiLQR can successfully be applied in moderate sea states and give improved attitude matching at touchdown due to the inclusion of nonlinear dynamics. Note that the work presented in [5] was not the first to apply DDP/iLQR to autonomous landing. Receding horizon DDP was also applied to landing a quadrotor on a moving platform in [18], but the authors focused on vision based estimation of the landing platform.

Very few other research campaigns have applied nonlinear MPC to ship landing. At the time of writing, the author of this paper is only aware of two other methods that have been explored. One is a minimum time algorithm that included nonlinear dynamics and path constraints (see [19]). The algorithm relied on the solution to a nonlinear programming (NLP) problem, which can be computationally expensive. To improve efficiency, the authors proposed a method of parameterizing the trajectory with polynomial basis functions in [20], significantly reducing the size of the NLP. Both formulations of the minimum time problem were tested with simulations of a UH-60A. The second is a model predictive path integral control (MPPI) algorithm discussed in [21]. MPPI is a stochastic trajectory optimization method that rolls out a large amount (potentially thousands) of state trajectories using monte carlo simulations. The control sequence is then updated based on a weighted average of the simulated trajectories (see [22] for a review and rigorous derivation of MPPI). Simulations of the MPPI ship landing controller from [21] showed large errors at landing, which is not acceptable in



(a) Illustration of EiLQR landing algorithm.



(b) Sample descent profile with EiLQR.

Figure 1.3: Schematic of EiLQR landing concept and sample descent profile (from [5]).

practice. The authors speculated that this was due to insufficient accuracy of the linear dynamic model used for planning.

1.2.1.2 Prescribed Form Trajectories

The MPC algorithms in the previous section do not prescribe any specific form that the trajectory must meet. This makes for a very general optimization problem, where the optimization space is all feasible controls. Trajectories with a prescribed form, on the other hand, assume that the trajectory must satisfy a certain structure (for example, the trajectory is defined by a polynomial). This allows the trajectory to be defined by a smaller number of parameters, making manual parameter selection or online optimization faster, but lacks the flexibility granted by optimal control. Additionally, prescribing the form of the trajectory does not guarantee that the planned path is dynamically feasible,

though this may not be an issue with reasonable commands and an effective trajectory following controller.

One prescribed form path planing method developed for autonomous ship landing is the polynomial path planner presented in [23]. The polynomial was fifth order with six coefficients. Four boundary conditions were imposed on the trajectory: the initial position and velocity were fixed to that of the helicopter, while the terminal position and velocity were fixed to values produced by a deck motion predictor. With six coefficients and four boundary conditions, free parameters are left for optimization. The free parameters were determined with a gradient descent optimization, where the cost function was designed to constrain the maximum velocity and acceleration of the approach. Altitude was also constrained to prevent premature contact with the deck. The method was tested with simulations of a light and medium class rotorcraft landing on a simulated ship representative of a DDG-51, as well as with simulations of a heavy class rotorcraft landing on a simulated ship representative of LHA-1. Impact velocities from the medium class simulation case are shown in Fig. 1.4, showing a low vertical impact velocity on all landing gear and a slightly larger lateral impact velocity. These results demonstrate that the polynomial path planner can produce a viable landing path for moderate sea states.

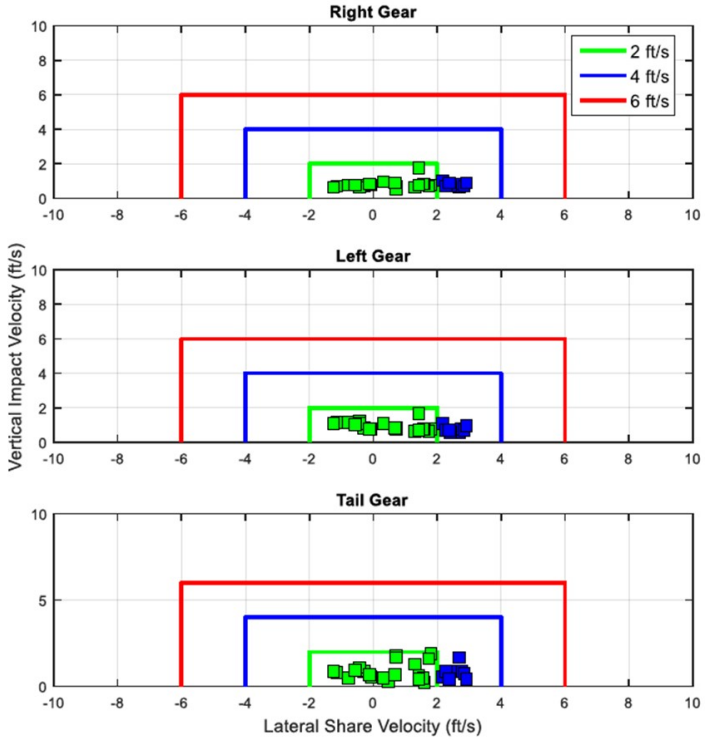


Figure 1.4: Simulated impact velocities with polynomial path planner from [23]

Prescribed form landing algorithms based on tau guidance have also been used. Tau theory is a nature inspired theory introduced by Lee [24], hypothesizing that animals plan guidance through the use of the variable τ where

$$\tau = \frac{\chi}{\dot{\chi}} \tag{1.2}$$

In eq. 1.2 the so called "action gap" χ represents the distance between the current state and a desired end state. Note this does not need to be a spatial distance, but can represent the distance between the current state and desired state for any degree of freedom. Lee also proposed a differential equation in terms of τ that smoothly regulates the action gap to zero in a specified time:

$$\frac{\chi}{\dot{\chi}} = \frac{k}{2}\left(t - \frac{T_g^2}{t}\right) \tag{1.3}$$

where k is a parameter used to tune the trajectory and T_g represents the total duration of the maneuver. This theory was used by researchers in [25] to create a feedback control law. The control law was tested with simulated ship landing and lateral reposition maneuvers. The focus was on proof of concept for the tau-based feedback control law, however, and shipboard landings were only simulated with a static deck and no disturbances. Another tau based method was proposed in [26], where a modified version of eq. (1.3) was used to produce position, velocity, and acceleration references. The algorithm was applied to landing a small UAV on a motion platform using vision-based deck state estimation.

1.2.1.3 Deck Tracking Methods

A simple method for performing ship landings is to match the oscillations of the deck while simultaneously closing the gap between the landing point and the aircraft. If the aircraft can adequately match the motion of the landing platform then this method is feasible and greatly simplifies trajectory generation. The drawback of deck tracking is that additional maneuvering may result when compared to MPC algorithms that plan to a predicted deck state. A schematic depicting this is shown in Fig. 1.5 : an MPC method that plans its landing path to a predicted deck state can plan a more direct landing path, provided the future deck state predictions are sufficiently accurate. This has the potential to alleviate maneuverability constraints, allowing aircraft that are not capable of tracking aggressive deck motions to land in rough seas.

In [6] a very simple implementation of deck tracking was tested with simulations of a

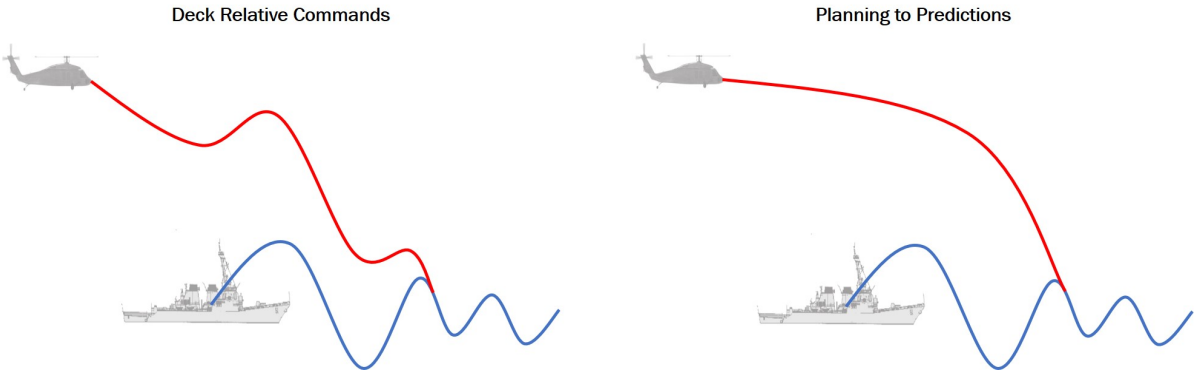


Figure 1.5: Illustration of deck tracking versus path planning to deck motion predictions.

helicopter similar to a UH-60 landing on a ship representative of a DDG-51 destroyer. The helicopter was commanded to match the surge and sway of the deck while the aircraft descended at a constant rate relative to the deck. Results of a sample landing with this method are shown in Fig. 1.6, where the expected oscillations are clearly visible. This paper also presented a linear-quadratic optimal control method discussed in section 1.2.1.1, which incorporated deck motion predictions. Simple deck tracking was found to be surprisingly reliable and actually outperformed the optimal control method in this case, due to the optimal control law's sensitivity to poor long term predictions. Results given by Pravitra in [5], though, showed that EiLQR (a more advanced MPC algorithm incorporating deck state predictions) could consistently outperform a deck tracking controller.

A number of other researches have landed UAVs on small platforms with deck tracking controllers. For example, in [27] a deck tracking controller was used to land a quadcopter on a wave glider at sea. In [28, 29] a 4 foot by 4 foot oscillating landing platform was towed behind a moving vehicle and landings were conducted using relative position estimates produced by a computer vision system. In [30–32] landing was performed to stationary oscillating motion platforms. A unique aspect of [32] is that only the state of the UAV relative to the deck was used for control, meaning that the inertial position and velocity of the UAV are unknown. To cope with this, deck motion was treated as an external disturbance to be estimated online. This approach allowed GPS denied landings, though the controller was only tested for small (magnitudes on the order of a few centimeters), low-frequency motions.

Others have utilized deck motion predictions to predict quiescent periods, and then initiate a landing with a deck tracking control law once a quiescent period is expected to occur. For example, this approach was taken in [33–35]. If quiescent periods are

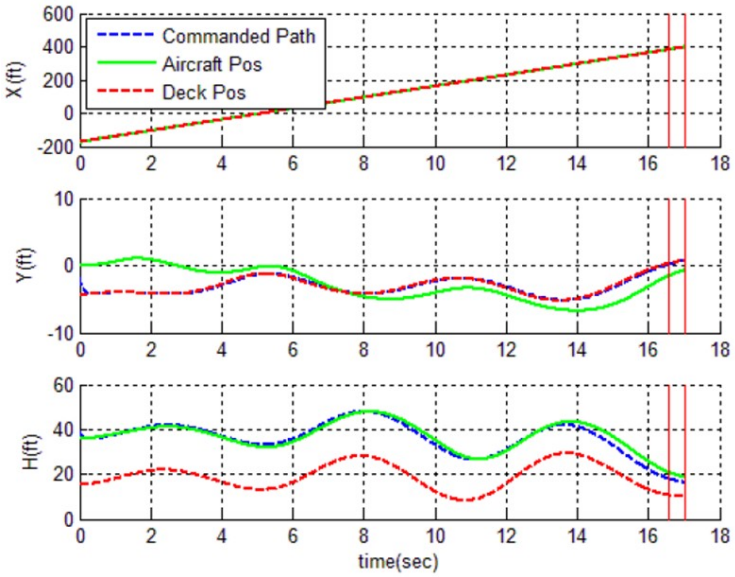


Figure 1.6: Sample landing with deck tracking controller [6]. The top plot shows longitudinal motion, the middle plot lateral motion, and the bottom plot altitude.

successfully identified then this facilitates landing, and it also does not introduce deck predictions into the feedback loop. If deck predictions are reliable enough, though, full use of the prediction in path planning could further reduce the required maneuvering.

Note that some of the methods discussed in the previous subsections can also be classified as deck tracking algorithms. Namely, the tau inspired approaches and any MPC algorithm that does not plan to a future deck state, as these methods must react to the deck at least during the final stages of landing.

1.2.2 Deck Motion Prediction

Some of the landing algorithms discussed in the previous subsection include predictions of future deck motions. As mentioned previously, there are several advantages to this if predictions are sufficiently accurate. Namely, predictions can allow an autonomous landing system to plan a more direct landing path and can also be used to identify calm periods for triggering the landing procedure. For piloted aircraft, deck motion predictions could aid in decision making in a similar way. For example, helicopter pilots make critical flight path decisions 5 to 10 seconds before touchdown [36]. The ability to predict deck motions 5 to 10 seconds into the future could therefore help the operator decide whether or not a landing should be waved off. These and other applications have motivated a number of deck motion prediction algorithms that can be found in the literature.

Deck motion prediction methods can generally be divided into two classes: those which incorporate measurements of the surrounding environment along with information on the ship model, and time series methods which use past measurements of ship state alone. Examples falling in the first class can be found in [37] and [38], where the ship is equipped with a Doppler radar for measuring the surrounding wave field. Inclusion of these measurements along with ship model parameters can increase forecasting accuracy, but requires the ship to be equipped with the prediction system. To date, such systems have not been incorporated into any published autonomous landing algorithms. Several landing algorithms have incorporated time series based predictions, however. A major advantage of time series methods is they require no model of the ship dynamics or special instrumentation onboard the ship, making the prediction method applicable to a wider variety of landing craft.

Time series based algorithms that have been applied to deck motion prediction include various neural network formulations (see [39–41]) and algorithms that represent ship motion as a sum of sines and perform adaptive spectral analysis (see [34,42,43]). The two prediction methods that have been most commonly applied to ship deck landing, however, are those based on autoregressive (AR) models and those based on minor component analysis (MCA). The AR model formulation assumes that the current output can be described by a linear combination of lagged outputs plus additive zero mean white noise. This is expressed as

$$\vec{y}_k = \alpha_1 \vec{y}_{k-1} + \alpha_2 \vec{y}_{k-2} \cdots + \alpha_{N_{lag}} \vec{y}_{k-N_{lag}} + \vec{v}_k \tag{1.4}$$

where $\vec{y}_k$ is the current output vector, $\vec{\alpha}$ represents the parameter matrices to be estimated, $\vec{v}_k$ is the noise vector, and N_{lag} represents the number of lagged outputs included in the model. Once the α matrices are estimated, the model is propagated forward in time to produce future predictions. This method was used for path planning in [5], and was also used for identifying quiescent periods in [33]. Many other studies have also investigated the use of different AR model variants for deck motion predictions (for example, [36, 42, 44–46]).

To give an idea of the prediction capabilities of a simple AR model, a comparison between predicted and actual deck heave is shown in Fig. 1.7. This plot was produced using ship motion time histories derived from simulation of a generic surface combatant representative of a DDG-51 type ship. This database was developed by the U.S. Navy Office of Naval Research and the Naval Surface Warfare Center under the Systematic

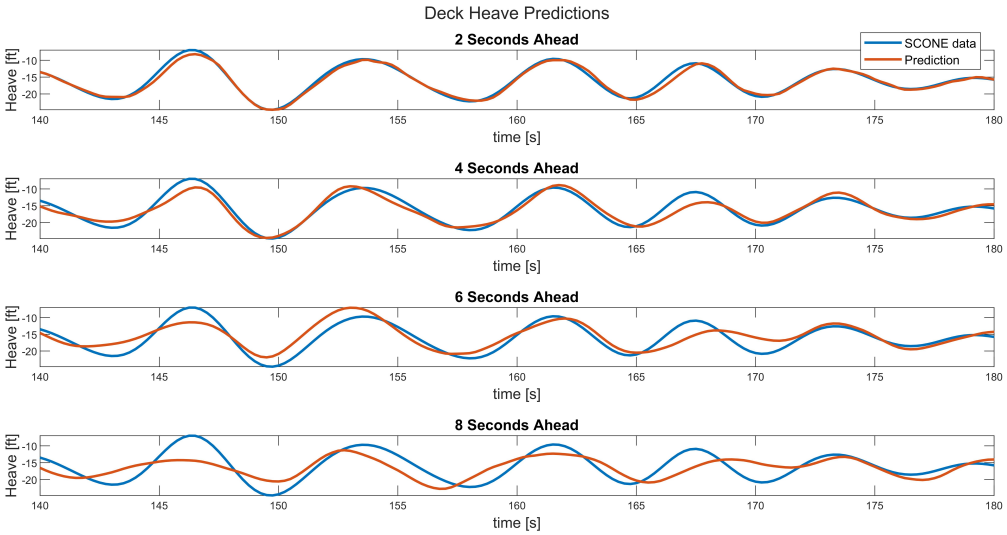


Figure 1.7: Ship heave predictions using an AR model and SCONE data.

Characterization of the Naval Environment (SCONE) program. More details on the SCONE data can be found in [47]. The AR model order was $N_{lag} = 30$ and the output vector used included deck heave, heave rate, and pitch motions, though just heave predictions are shown here. Looking at Fig. 1.7, it can be seen that deck predictions are very accurate at two seconds ahead and are reasonably accurate out to about five seconds. After five seconds accuracy begins to drop off.

The MCA time series prediction method uses the smallest eigenvalues and the associated eigenvectors of a signal's autocorrelation matrix to predict future values. A good description of the method is given by Monneau in [42]. MCA based predictions were incorporated in the ship landing guidance algorithms presented in [6, 14, 23]. MCA ship motion prediction algorithms were also compared to AR model formulations in [42] and [44]. Both studies showed the MCA method to give more accurate predictions, though the difference in prediction accuracy is not reported to be large and the MCA method was reported in [42] to be more computationally expensive. An additional factor to consider is that the AR methods used in both [42] and [44] consider a single measurement. For example, past measurements of deck heave are used to forecast future deck heave motions. It is possible, however, to include other measurements in the AR forecasting algorithm. This was done when producing Fig. 1.7, where deck heave, heave rate, and pitch angle are included in a single output vector to forecast future values of heave, heave rate, and pitch angle. Including multiple measurements in the output vector used in the prediction model allows correlations between measured signals to be

identified, which could potentially increase forecasting accuracy.

1.2.3 Dynamic Scaling

In the context of this paper, dynamic scaling refers to methods for relating the dynamics of a small scale aircraft to a larger scale model. This is relevant to this work, as model scale ship landing experiments should be designed so that key dynamics are roughly consistent with reduction in scale, allowing for more meaningful extrapolation of results to a target full scale use case. In order to do this, a dynamic scaling method is needed.

Two common scaling laws are Froude scaling and Mach scaling. Froude scaling assumes that two aircraft being compared are dynamically similar, meaning that the relative magnitude of the governing forces remains unchanged across scales [48]. Dynamic similarity is quantified through the parameter

$$F_r = \frac{V^2}{gL} \tag{1.5}$$

where V is a characteristic velocity, L is a characteristic length, and g is gravitational acceleration. With appropriate choice of characteristic length and velocity, F_r is representative of the ratio of inertial to gravitational forces. For example, as discussed by Mettler in [48, 49], for a conventional helicopter the characteristic velocity can be chosen as rotor tip speed and the characteristic length as rotor radius. F_r is then correlated with the ratio of blade lift to vehicle weight. Mach similarity, on the other hand, assumes that the characteristic velocity remains constant across test scales.

The Froude and Mach scaling assumptions can be used to derive useful scaling rules. For example, by using the Froude scaling assumption that eq. (1.5) remains constant across scales, it is shown in [48] that frequency must scale by the rule

$$\omega_{ms} = \sqrt{N_F}\,\omega_{fs} \quad , \quad N_F = \frac{L_{fs}}{L_{ms}} \tag{1.6}$$

where the Froude scaling factor N_F is defined by the characteristic length ratio and subscripts *fs* and *ms* signify full and model scale, respectively. It is also known that units of time scale by the inverse of the frequency scale factor:

$$t_{ms} = \frac{t_{fs}}{\sqrt{N_F}} \tag{1.7}$$

Using the scaling rules in eqs. (1.6) and (1.7), dimensional analysis can be applied to

derive other Froude scaling rules, some of which are shown in table 1.1. Note that we multiply by these values to go from full to model scale.

Table 1.1: Froude and Mach scaling factors.

Dimension	**Froude Scale Factor**	**Mach Scale Factor**
Time	$1/\sqrt{N_F}$	$1/N_F$
Frequency	$\sqrt{N_F}$	N_F
Position	$1/N_F$	$1/N_F$
Velocity	$1/\sqrt{N_F}$	1
Acceleration	1	N_F
Jerk	$\sqrt{N_F}$	N_F^2
Angles	1	1
Angular Rates	$\sqrt{N_F}$	N_F
Weight	$1/N_F^3$	$1/N_F^3$
Inertia	$1/N_F^5$	$1/N_F^5$

Mach scaling rules that are obtained by applying the speed similarity assumption are also shown in table 1.1. Clearly, the two laws are different and cannot be satisfied simultaneously. This is well documented in rotor aeroelasticity studies, for which Froude and Mach scaling have both commonly been employed. For example, in [50] and [51] discussions are given on when it is most appropriate to apply either Froude or Mach scaling for designing a model scale aeroelasticity experiment.

While the classic use of Froude and Mach scaling is for designing a model scale experiment based on some full scale prototype, these rules have also been used to relate the flight dynamics and control characteristics of differently sized aircraft that are not meant to be models of each other. Though this means that there may be significant deviations from similarity assumptions, these methods have still proven useful for roughly scaling dynamic behavior. An example of this is given in [48], where Froude and Mach scaling are used to study the flight dynamics of small-scale helicopters. The study used two small-scale helicopters: one which was capable of acrobatic maneuvers, and one that was designed for less aggressive operation. Froude scaling was applied to compare the small-scale aircraft to a Bell UH-1H and a Robinson R22, with the scale factor N_F taken as the ratio of full to model scale rotor radius. The results showed a reasonable relation between the non-acrobatic small-scale helicopter and the full-scale helicopters. The Froude scaled dynamics of the acrobatic helicopter did not compare well with larger aircraft, however. Instead, Mach scaling the acrobatic helicopter resulted

in scaled dynamics much more similar to the larger aircraft. Looking at the scaling parameters of table 1.1, this is intuitive as frequencies scale by N_F for Mach scaling, as opposed to scaling by $\sqrt{N_F}$ for Froude scaling, indicating faster dynamics and more control authority for a Mach scaled model (which are necessary properties for performing acrobatic maneuvers). based on this result, the author suggested that Froude and Mach scaling can be considered together to make predictions about the flight dynamics and maneuverability of a small-scale helicopter.

Other researchers have used Froude scaling to correlate the dynamics and handling qualities of small multi-rotor UAVs with larger aircraft. Some examples of this are given in [52–54]. In [52], state space models were identified for two different size hexacopter UAVs. Stability and control derivatives were related via Froude scaling, with the Froude number N_F taken as the ratio of full to small scale hub-to-hub distance. Results of the scaling comparison are shown in table 1.2, where the dynamics of the larger UAV prove to be well approximated by the Froude scaled dynamics of the smaller UAV (note, the smaller UAV is denoted as "UP hex" in table 1.2). In [53], a similar scaling comparison was given, where the lateral dynamics of an IRIS+ quadcopter were compared to those of an XV-15. The results of this are given in table 1.3, which show Froude scaling to give a much cruder relation between vehicle dynamics when comparing across such large scales. Still, a rough approximation is achieved despite the large differences between the two vehicles. In [54] Froude scaling was investigated as a method of translating the handling qualities requirements given for full scale manned rotorcraft in ADS-33E-PRF to small scale UAVs. The results showed Froude scaling based on hub-to-hub distance is a promising method for establishing predictive handling qualities metrics, such as disturbance rejection bandwidth (DRB), for UAVs. These studies demonstrate that Froude scaling can be effective for roughly correlating the dynamics of small multi-rotor UAVs to larger aircraft, even when the aircraft being compared are not perfectly dynamically similar.

Currently, no publicly available work has considered dynamically scaling autonomous ship landing experiments. The results discussed in the preceding paragraphs, however, indicate that Froude scaling is appropriate for roughly relating ship motion and aircraft dynamics across test scales. Froude scaling is also a logical method to choose for this purpose due to the force similarity assumption it is based on. Factors such as relative thrust to weight ratio have clear importance for maneuvering relative to the ship deck, and therefore similarity in this regard should be approximately maintained.

Table 1.2: Froude scaled hexacopter UAV dynamics with $N_F = 2.3$ [52].

Mode	Frequency (rad/s)		Damping	
	Scaled UP Hex	True ADD Hex	Scaled UP Hex	True ADD Hex
		Lateral-directional dynamics		
Unstable lateral oscillatory mode	2.22	2.08	−0.48	−0.41
Stable roll mode	2.3	2.44	n/a (first order)	
Yaw mode	0	0	n/a (first order)	
		Longitudinal-heave dynamics		
Unstable longitudinal oscillatory mode	2.22	2.08	−0.48	−0.41
Stable pitch mode	2.3	2.44	n/a (first order)	
Stable heave mode	0.224	0.291	n/a (first order)	
		Motor dynamics		
Stable motor lag modes (× 6 motors)	9.95	11.0	n/a (first order)	

Table 1.3: Froude scaled XV-15 and IRIS+ quadrotor lateral dynamics with $N_F = 35$ [53].

Mode	XV-15 Freq. [rad/sec]	Froude-Scaled Freq. [rad/sec]	IRIS+ Freq. [rad/sec]	Diff-erence
$\lambda_{\text{Roll}_{1,2}}$	0.4668	2.75	2.55	7.5%
λ_{Roll_3}	0.6458	3.80	2.65	35.6%

1.2.4 Ship Landing Experimental Setups

The majority of the ship landing guidance algorithms discussed in section 1.2.1 were evaluated through simulation, but this work is focused on the experimental evaluation of autonomous landing systems. A brief review of the types of autonomous ship landing experiments that can be found in the open literature is therefore pertinent.

(a) Test setup from [26]. (b) Test setup from [32].

Figure 1.8: Example experimental setups with small UAVs and motion platforms.

The most common test setup for experimentally evaluating autonomous landing systems involves small UAVs operating from motion platforms in a motion capture environment. Examples of such setups are shown in Fig 1.8, and similar setups can be found in many other works (for example, [7, 30, 31]). The advantage of this is cost and

easy to obtain hardware, but many small motion platforms can only achieve very limited translational motions. Ship motions may be large in heave and sway, which can make small motion platforms limiting even when scaling the magnitude of oscillations to be appropriate relative to the UAV.

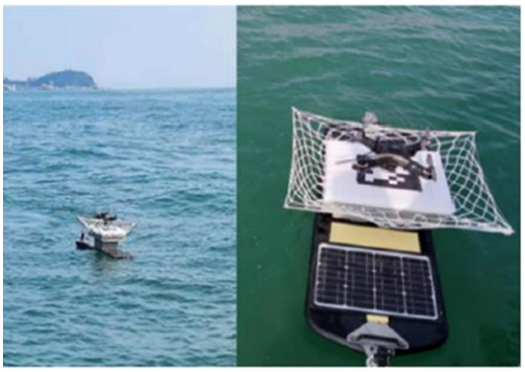
A small number of other researchers have performed landings on trailers towed by a ground vehicle. Examples of this are shown in Fig. 1.9. Both of these setups give added realism in that the landing platform can translate and change course, and there are natural aerodynamic disturbances. Emulating the oscillatory motion of the ship is difficult in such a setup, though, as a large motion platform must be attached to the trailer. At the time of writing, the setup shown in Fig. 1.9a is the only publicly available work that the author of this paper is aware of which includes a towed motion platform, and position oscillations are still limited to be on the order of 5 centimeters or less. Additionally, testing in an outdoor environment can impact the repeatability of test conditions. This gives some appeal to indoor flight testing for cases where a large amount of data at the same condition is desired.

(a) Towed motion platform for quadrotor vision based landings [29].

(b) UAV landing on a moving trailer [35].

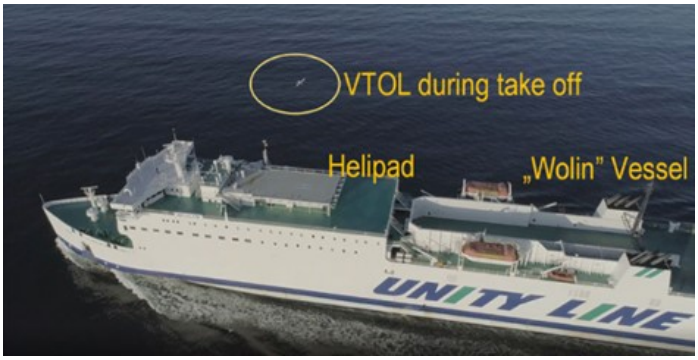
Figure 1.9: Example outdoor landings onto a towed trailer.

An even smaller amount of publicly available data can be found for autonomous ship landings performed at sea. Three examples of such cases are shown in Fig. 1.10. In [27] (see Fig. 1.10a), a small quadrotor was used to land on a stationary platform subject to waves. A total of 20 landings were performed with different controller configurations. In [10] (see Fig. 1.10b) a small quadrotor was used to perform cooperative landings with a 15-meter-long ship, but only results from one recorded landing were given. The only other published work that the author of this paper is aware of where autonomous landings of a rotorcraft were performed at sea is in [55], where two landings were performed to a 189-meter-long ferry (see Fig. 1.10c). While a test setup such as that of Fig. 1.10c can be used to gain valuable information on the viability of a complete autonomous landing solution, high volume testing in repeatable wave and wind conditions is difficult to achieve. In fact, in [55] it was cited that only two recorded tests were performed due factors including: unfavorable weather conditions and wind speed, position on international waters, time of day, and the presence of other vessels in the ferry's surroundings.



(a) Quadrotor landing on a wave glider [27].

(b) 15-meter-long landing craft used for cooperative landing with a quadrotor [10].



(c) Maritime operation of a UAV from the 189-meter-long Wolin ferry. [55].

Figure 1.10: Example outdoor landings at sea.

Overall, there is not currently a large amount of experimental data available on autonomous ship landing guidance and control systems. Most studies which do perform repetitive testing utilize small motion bases to emulate the ship, as this setup is more amenable to high volume testing than testing at sea. Studies using small UAVs and motion bases typically do not frequency scale ship motion, however. Often just the magnitudes of surge, sway, and heave are reduced until travel limits of the motion base are not encountered. Additionally, most of these studies utilize the full control bandwidth of the small-scale UAVs, which is typically much higher than the dominant frequency of full-scale ship motion. This may not be problematic for validating certain specific technologies or when testing close to a target use case. For using model-scale experiments to evaluate autonomous landing guidance and control laws intended for use at much larger scale, though, this is unrealistic as full-scale rotorcraft may have control bandwidths close to or less than the dominant frequency of ship motion. This work uses Froude scaling to better relate dynamics across test scales while utilizing a test setup that enables high volume testing.

1.3 Contributions

While there is a significant amount of published research on autonomous landing guidance and control laws, most existing studies are simulation based and there is a lack of openly available experimental analyses of these systems. This research worked toward filling this gap through the following contributions:

1. **The development of a methodology for performing dynamically scaled autonomous ship landing experiments**

   The existing experimental evaluations of autonomous landing systems that can be found in the public literature have been conducted at model-scale. These studies have not considered the scaling of dynamics across test scales, however. This work introduces the use of Froude scaling to consistently scale both aircraft closed-loop dynamics and ship motion, providing a more realistic model-scale representation of the full-scale landing scenario. The results demonstrate that model-scale tests can be performed in a more accessible test setup while retaining dynamic similarity. For example, even though the majority of the tests performed for this study are conducted in a wave basin, a Froude scaled UAV model operating from a motion platform can be used to provide a representative landing scenario provided the

control law, guidance algorithms, and ship motion time histories are all consistently Froude scaled.

2. **The extensive experimental evaluation of autonomous landing algorithms** Scaled autonomous ship landing experiments were performed (see Fig. 1.11), providing experimental data that clarifies the effectiveness and benefits of different landing methods in practice. More specifically, the following experimental studies were conducted:

   - *Comparison between simple deck tracking and advanced guidance algorithms*

     Two guidance algorithms were developed and evaluated as part of this work. The first is the so-called "baseline" algorithm that commands a deck-relative flight path, closing the distance between the aircraft and deck at a steady rate. The second is an advanced method representative of the MPC style guidance algorithms prevalent in the literature, for which few experimental validations have been published. More specifically, the advanced guidance algorithm utilizes quadratic programming (QP) optimization to plan a landing path in real-time, with the trajectory planned to deck state predictions produced by autoregressive time series models.

     In theory, the incorporation of landing path optimization and predicted deck motions should allow the QP algorithm to outperform the purely reactive deck tracking method. This claim is supported by simulations conducted in [5]. Performing such comparisons experimentally is of interest, though, because the robustness of advanced landing algorithms that use deck predictions directly in path planning is a point of concern. Time series based prediction algorithms that are commonly used for deck motion forecasting can be sensitive to noise in measured/estimated outputs, and even with high quality data deck motion predictions may be poor at times. A deck tracking guidance algorithm paired with high bandwidth control, on the other hand, may be comparatively more robust if control limits are not hit during landing, as path planning is simple and the main reliance is on proven flight control laws. The experimental results presented here give insight into the trade-offs between these two methods, as well as the necessary accuracy of deck motion predictions to perform safe landings.

   - *Studying the sensitivity of landing algorithms to degraded reference tracking bandwidth and maneuverability*

Using control laws implemented on small-scale UAVs, reference tracking bandwidths and maneuverability constraints (i.e., limits on acceleration or jerk) can be artificially degraded. This was used to study the sensitivity of the deck tracking and QP landing algorithms to limited control authority and speed of response. For a deck tracking method, degraded mobility will cause the aircraft to lag deck motion and eventually lead to hard landings. Scaled tests were performed with degraded reference tracking bandwidth to give insight into the bandwidth necessary to utilize a deck tracking guidance algorithm in high sea states at full-scale. A landing algorithm with predictive capabilities, on the other hand, may be able to adequately plan for reduced bandwidth and output constraints. Experiments were performed with the QP algorithm to demonstrate the effectiveness of this approach in practice.

- *Studying the robustness of advanced landing algorithms to disturbances*

  Additional experiments were performed where an industrial fan was used to apply an abrupt wind gust. These experiments evaluate the robustness of the QP algorithm (a representative shrinking-horizon guidance algorithm that re-plans the trajectory from the perturbed aircraft state) to sudden disturbances. The results are compared to those obtained with the baseline "deck tracking" method to determine if the advanced landing strategy offers any advantage over the deck tracking method when operating in the presence of a significant aerodynamic disturbance.

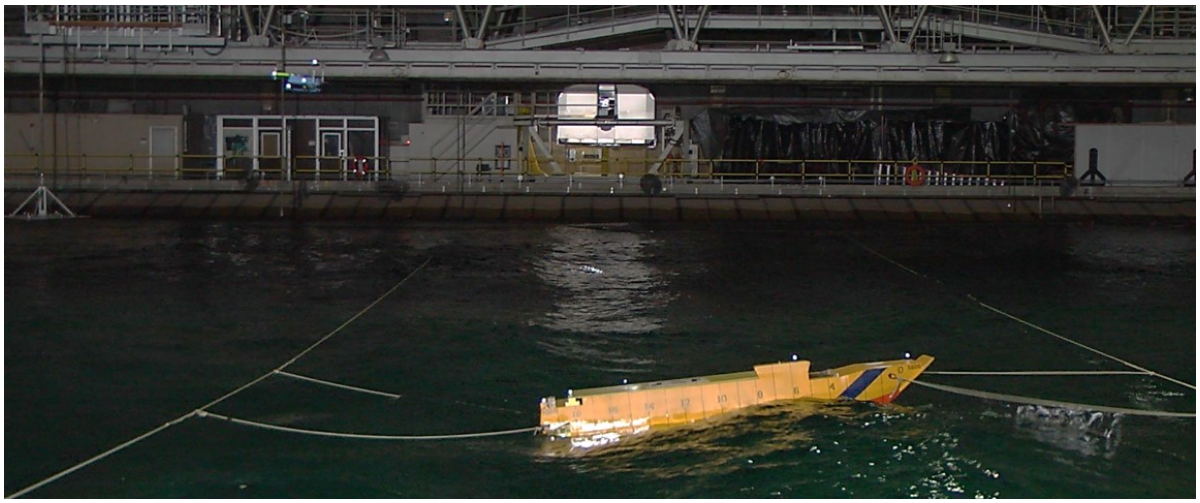


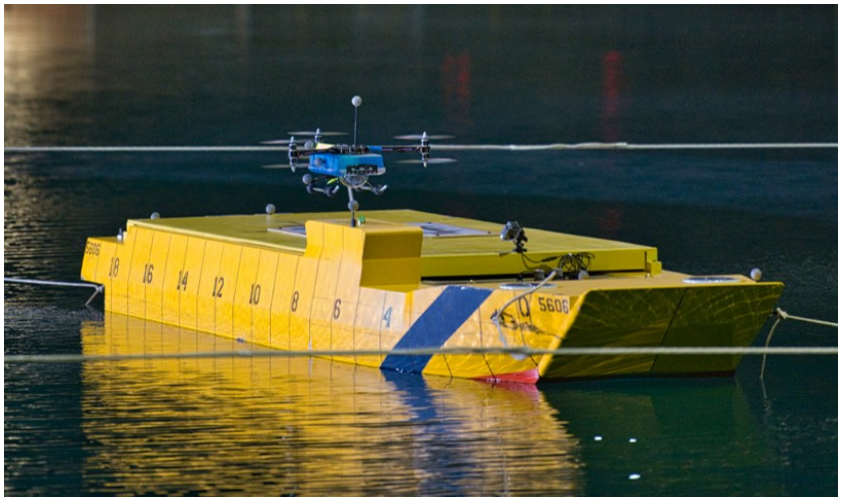
Figure 1.11: Autonomous landing experiment performed at the Maneuvering and Seakeeping Basin located at the Naval Surface Warfare Center Carderock Division.

Chapter 2
Model Following Control Laws

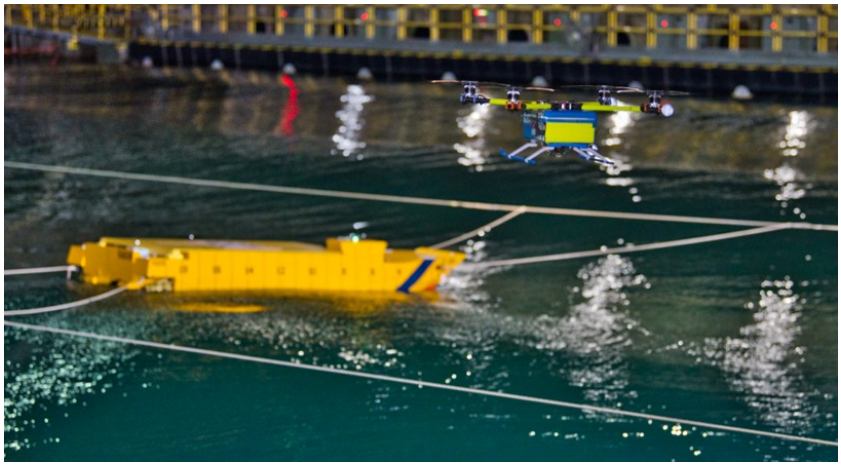
One of the goals of this work was the development of autonomous ship landing experiments where key aspects of the aircraft closed-loop dynamics are roughly consistent with reduction in scale. Toward this end, explicit model following (EMF) control laws were developed and implemented on two UAV platforms: a hexacopter and a quadcopter. The EMF control architecture was chosen as this allows for reference tracking dynamics and disturbance rejection bandwidths to be easily related across test scales. This chapter will describe the UAV platforms used in experimentation, as well as the system identification methods used to derive the aircraft models used in control design. The control architecture and design procedure will then be presented. Note that the scaling of the closed-loop system and the selection of controller parameters to match scaled reference tracking and disturbance rejection bandwidths will be discussed in detail in Chapter 4.

2.1 UAV Platforms

A coaxial hexacopter (Fig. 2.1a) and a quadcopter (Fig. 2.1b) were built for use in the model-scale autonomous landing experiments conducted for this research. Both vehicles are equipped with a Pixhawk Cube Orange flight controller running the PX4 flight control firmware, as well as an Odroid XU4 onboard computer. The Odroid uses the Robotic Operating System (ROS) to communicate with the flight controller over a serial link and is also fitted with a WIFI module, allowing for communication with a ground station computer. For both vehicles all electronics other than the motors are contained within a waterproof enclosure. To prevent the enclosure from overheating, the electronic speed controllers were fitted to aluminum heat sinks. The quadrotor is equipped with 12.5 inch diameter rotors, while the hexacopter uses 10 inch diameter rotors. Both UAVs have a hub-to-hub distance of about 1.75 feet and weigh about 6.6 pounds.



(a) Hexacopter UAV.



(b) Quadcopter UAV.

Figure 2.1: UAV platforms used in model-scale experiments.

2.2 UAV Model Identification

2.2.1 Overview of Transfer Function Modelling with CIFER®

UAV flight dynamics models were identified using the Comprehensive Identification from Frequency Responses (CIFER®) software package. CIFER® was developed by the U.S. Army Aeroflightdynamics Directorate (AFDD) at the NASA Ames Research Center and allows for frequency domain identification of generic state space models, as well as low order transfer functions. Here CIFER®'s transfer function identification capabilities were utilized, as multirotor UAVs like those used in this work can be well modelled through simple decoupled transfer functions. This section will give an overview of the methods

employed by CIFER® for transfer function modelling.

The first step in the system identification (ID) process is to record system inputs and outputs from flight tests where the aircraft dynamics are excited in the frequency range of interest. Typically, the aircraft dynamics are excited via frequency sweeps on each control axis. Once time histories are collected and provided to CIFER®, the average value and linear drift in each each data record is removed and the time history data is converted to the frequency domain using the chirp Z-transform. This results in frequency domain inputs $X(f)$ and outputs $Y(f)$, and allows for rough estimates of the input autospectrum $\tilde{G}_{xx}(f)$, output autospectrum $\tilde{G}_{yy}(f)$, and cross spectrum $\tilde{G}_{xy}(f)$ to be computed as

$$\begin{aligned} \tilde{G}_{xx}(f) &= \frac{2}{T}|X(f)|^2 \\ \tilde{G}_{yy}(f) &= \frac{2}{T}|Y(f)|^2 \\ \tilde{G}_{xy}(f) &= \frac{2}{T}\left[X^*(f)Y(f)\right] \end{aligned} \tag{2.1}$$

where T represents the duration of the time history data used in the chirp Z-transform and $X^*(f)$ represents the complex conjugate of $X(f)$ [56].

To reduce the effects of random errors in the extracted frequency response, CIFER® uses an overlapped windowing method. A graphical representation of this method is shown in Fig. 2.2 for a lateral stick frequency sweep: the original time history is broken into a number of smaller "windows" with a Hanning weighting function (represented by the bell shaped curves in Fig. 2.2) applied to each window. For each window, the weighted time domain data ($w(t)\delta_{lat}$ in Fig. 2.2) is transformed to the frequency domain using the chirp Z-transform and rough spectral estimates are calculated using eq. (2.1). A smooth spectral estimate is then obtained by averaging the rough spectral estimates across all windows with the equation

$$\hat{G}_{xx} = \frac{1}{Un_r}\sum_{k=1}^{n_r}\tilde{G}_{xx,k}(f) \tag{2.2}$$

where n_r is the number of windows used and U is a correction to the spectral density magnitude to account for the taper in the window weighting function. For a Hanning window this is calculated as $U = \sqrt{3/8}$. Equations analogous to eq. (2.2) are used to calculate the smooth output autospectrum $\hat{G}_{yy}$ and smooth cross spectrum $\hat{G}_{xy}$ [56].

The smooth autospectra and cross spectra can then be used to compute an estimate

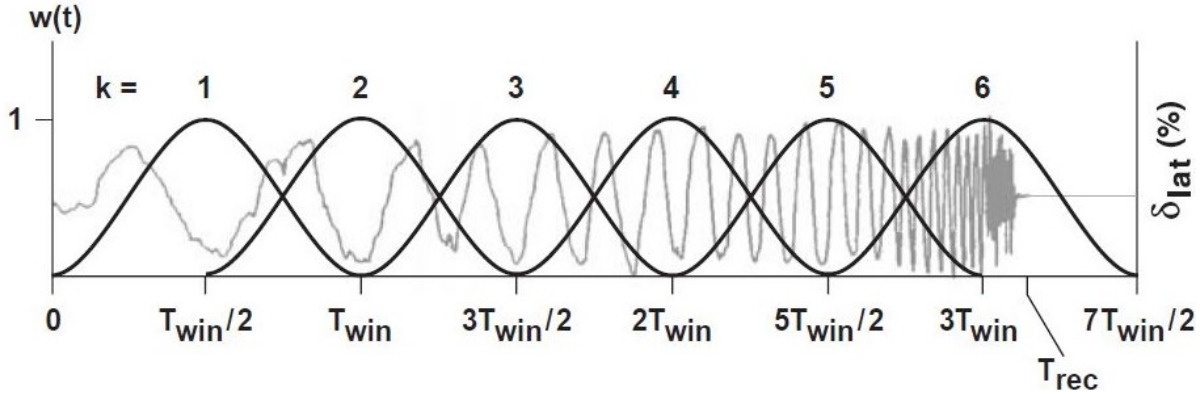


Figure 2.2: Example of overlapped windowing (adapted from [57])

of the coherence function with the equation

$$\hat{\gamma}_{xy}^2(f) = \frac{|\hat{G}_{xy}(f)|^2}{|\hat{G}_{xx}(f)|\,|\hat{G}_{yy}(f)|} \tag{2.3}$$

The coherence function varies from 0 to 1 and represents that portion of the output spectrum that is linearly attributable to the input spectrum at each frequency [57]. For a value of $\hat{\gamma}_{xy}^2 = 1$, the system is linear and all of the output spectrum is attributable to the input spectrum. In reality, however, the coherence function is never perfectly 1 due to system nonlinearities, noise, and unmeasured inputs like wind gusts. In practice, a coherence of $\hat{\gamma}_{xy}^2 \geq 0.6$ is sufficient for system ID purposes [56].

When computing the smooth autospectra there is a trade-off in the choice of window size: longer windows provide more low-frequency information, while smaller windows reduce the effects of random errors. CIFER® overcomes this trade-off by using a composite-windowing method, which automatically selects a range of window sizes to use then produces smooth spectral estimates $\hat{G}_{xx}$, $\hat{G}_{yy}$, and $\hat{G}_{xy}$ with each chosen window size. These estimates are then merged into composite spectral estimates $\hat{G}_{xx,c}$, $\hat{G}_{yy,c}$, and $\hat{G}_{xy,c}$ through an optimization procedure, automatically handling the trade-off in window size selection. The composite autospectra and cross spectra can then be used to calculate a composite coherence estimate. More details on this process can be found in [56].

Once the composite autospectra are obtained, the composite frequency response that captures the input-output dynamics can be calculated as

$$\hat{T}_c(f) = \frac{\hat{G}_{xy,c}(f)}{\hat{G}_{xx,c}(f)} \tag{2.4}$$

and CIFER® can then fit a transfer function model T to the composite frequency response. This is done by minimizing the cost function

$$J = \frac{20}{n_w} \sum_{\omega_1}^{\omega_{n\omega}} W_\gamma \left[W_g \left(|\hat{T}_c| - |T| \right)^2 + W_p \left(\angle \hat{T}_c - \angle T \right)^2 \right] \tag{2.5}$$

where $|\cdot|$ represents magnitude in dB, $\angle$ represents phase in degrees, n_ω is the number of frequency points to use in optimization, ω_1 and $\omega_{n\omega}$ are the lowest and highest frequency points used, and W_γ, W_g and W_p are weighting functions. W_γ is dependent on the coherence function at each frequency and is calculated as

$$W_\gamma(\omega) = \left[1.58 \left(1 - e^{-\gamma_{xy}^2} \right) \right]^2 \tag{2.6}$$

which emphasizes the frequency points with the highest coherence during the optimization. W_g and W_p are the relative weights for the magnitude and phase square errors and are set to $W_g = 1.0$ and $W_p = 0.01745$, which sets 1-dB of magnitude error comparable to 7.57 degrees of phase error [56].

The transfer function model T is a generic transfer function of up to the 4^{th} order and allows for the inclusion of input-output delays. That is,

$$T(s) = \frac{a_1 s^4 + a_2 s^3 + a_3 s^2 + a_4 s + a_5}{s^4 + b_1 s^3 + b_2 s^2 + b_3 s + b_4} e^{-\tau s} \tag{2.7}$$

where s represents the Laplace variable. To fit the transfer function model, the numerator and denominator order must be specified by the user, and the user must also make the decision on whether or not to include a time delay. The numerator coefficients, denominator coefficients, and time delay are then adjusted using Rosenbrock's multivariable search method to minimize eq. (2.5).

2.2.2 PX4 Stabilized Mode Controller

During the flight tests used for UAV system identification and model validation the PX4 stabilized mode controller was active. It is therefore necessary to understand the PX4 stabilized mode control structure, as model validation relied on reconstructing the closed loop system.

A block diagram of the PX4 stabilized mode controller is shown in Fig. 2.3. In stabilized mode the pitch and roll axes follow an attitude-command-attitude-hold response type, the yaw axis follows a rate-command-attitude-hold response type, and the heave

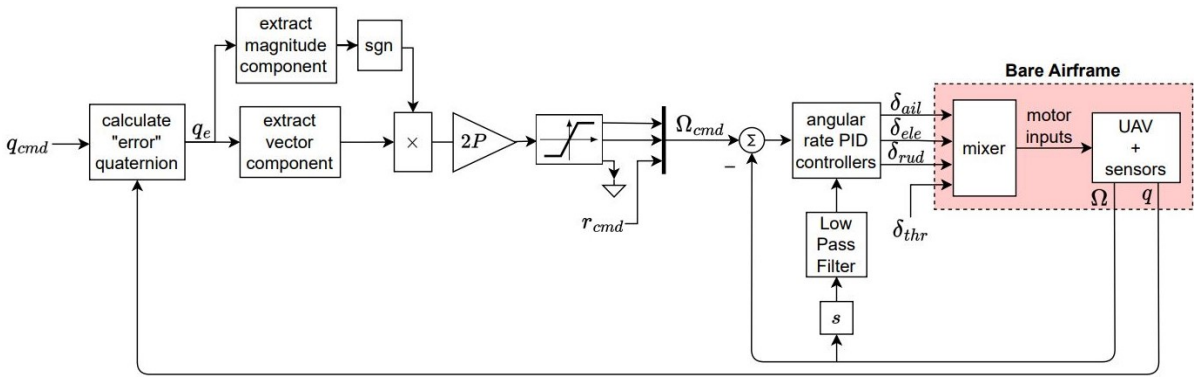


Figure 2.3: PX4 stabilized mode control architecture.

axis is open loop with throttle commands directly input to the mixer. Referring to Fig. 2.3, we can see that (for roll and pitch) the attitude controller is nonlinear and acts on a quaternion representation of vehicle attitude, where q_{cmd} represents the commanded quaternion and q represents the estimated vehicle attitude expressed as a quaternion. This notation is used in this section alone, and should not be confused with pitch rate command and pitch rate which are denoted by q_{cmd} and q, respectively, in all other sections of this work. Also note that Ω is used in this section to represent the body axes angular rate vector.

For the purpose of validating the identified transfer function models that will be discussed in Chapter 2.2.4, it is convenient to represent the attitude controller in a decoupled and linearized form with feedback acting on the Euler angles. To achieve this, an expression for the error quaternion q_e is needed. Noting that the q_e is defined as the quaternion that performs the rotation from q to q_{cmd} the following expression can be written:

$$q_{cmd} = q * q_e \tag{2.8}$$

where the operator $*$ denotes quaternion multiplication. Solving for the error quaternion gives

$$q_e = q^{-1} * q_{cmd} = \frac{q}{\|q\|} * q_{cmd} = \frac{1}{\|q\|} \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \begin{bmatrix} q_{0,cmd} \\ q_{1,cmd} \\ q_{2,cmd} \\ q_{3,cmd} \end{bmatrix} \tag{2.9}$$

where q_0 represents the scalar component of the quaternion and the quaternion norm is calculated as

$$\|q\| = q_0^2 + q_1^2 + q_2^2 + q_3^2 \tag{2.10}$$

To express q_e in terms of the measured and commanded Euler angles, q and q_{cmd} must be written in terms of Euler angles. For q the is written as

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\phi}{2})\cos(\frac{\theta}{2})\cos(\frac{\psi}{2}) + \sin(\frac{\phi}{2})\sin(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \sin(\frac{\phi}{2})\cos(\frac{\theta}{2})\cos(\frac{\psi}{2}) - \cos(\frac{\phi}{2})\sin(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \cos(\frac{\phi}{2})\sin(\frac{\theta}{2})\cos(\frac{\psi}{2}) + \sin(\frac{\phi}{2})\cos(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \cos(\frac{\phi}{2})\cos(\frac{\theta}{2})\sin(\frac{\psi}{2}) - \sin(\frac{\phi}{2})\sin(\frac{\theta}{2})\cos(\frac{\psi}{2}) \end{bmatrix} \tag{2.11}$$

and q_{cmd} can be expressed in the same manner using the commanded Euler angles. To develop a simplified decoupled model for model validation purposes, it is assumed that attitude commands are limited to a single axis and all other Euler angle commands and measurements remain close to 0. This assumption is valid for model validation as the system ID flight tests consisted of commands primarily on a single axis. For example, considering roll motion and letting $\theta = \psi = 0$, eq. (2.11) can be simplified to

$$q = \begin{bmatrix} \cos(\frac{\phi}{2}) & \sin(\frac{\phi}{2}) & 0 & 0 \end{bmatrix}^T \tag{2.12}$$

Similarly, q_{cmd} can be expressed as

$$q_{cmd} = \begin{bmatrix} \cos(\frac{\phi_{cmd}}{2}) & \sin(\frac{\phi_{cmd}}{2}) & 0 & 0 \end{bmatrix}^T \tag{2.13}$$

Substituting eqs. (2.12) and (2.13) into eq. (2.9) yields the following expression for q_e:

$$q_e = \begin{bmatrix} \cos(\frac{\phi}{2})\cos(\frac{\phi_{cmd}}{2}) + \sin(\frac{\phi}{2})\sin(\frac{\phi_{cmd}}{2}) \\ \cos(\frac{\phi}{2})\sin(\frac{\phi_{cmd}}{2}) - \sin(\frac{\phi}{2})\cos(\frac{\phi_{cmd}}{2}) \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\phi_{cmd}-\phi}{2}) \\ \sin(\frac{\phi_{cmd}-\phi}{2}) \\ 0 \\ 0 \end{bmatrix} \tag{2.14}$$

The scalar component of the error quaternion in Eq. (2.14) is always positive for $-90^\circ < \phi_{cmd} - \phi < 90^\circ$. Additionally, assuming small values of $(\phi_{cmd} - \phi)/2$, the first vector component of q_e is approximated as $(\phi_{cmd} - \phi)/2$. The commanded roll rate produced by the quaternion attitude controller from Fig. 2.3 can then be reduced to

$$p_{cmd} \approx sgn(q_{e,0})\frac{2P(\phi_{cmd} - \phi)}{2} = P(\phi_{cmd} - \phi) \tag{2.15}$$

Thus, the nonlinear attitude controller can be represented as a proportional gain controller acting on the Euler angle error for the de-coupled, linearized model assumptions. The

same results are obtained for pitch compensation.

A block diagram of the roll axis control loop with the linearized attitude controller is shown in Fig. 2.4. From Fig. 2.4, we can see that the inner rate loop consists of a PI controller with derivative (angular acceleration) feedback. The angular acceleration feedback is filtered by a second order low pass filter. The damping ratio for this filter is 0.707 and the break frequency ω_f was fixed at 20 Hz. Note that this same control architecture is applied to the pitch axis. Additionally, the same inner-loop rate controller is applied to the yaw rate response, but with the absence of the outer attitude loop as the pilot commands yaw rate directly. Using this control architecture, the closed loop system used in model validation can easily be constructed in the Laplace domain.

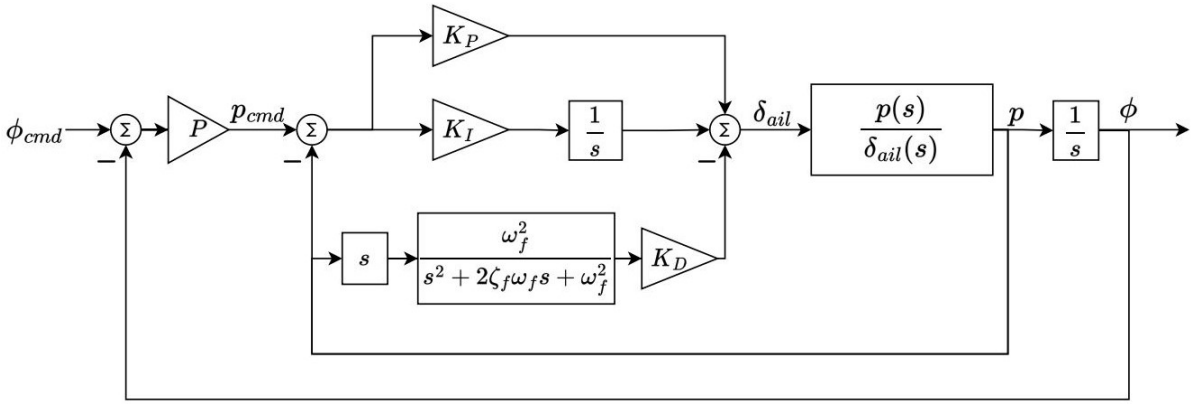


Figure 2.4: Block diagram of linearized PX4 stabilized mode roll axis controller.

2.2.3 Flight Test Procedure

Past work by Wei [58] demonstrated successful frequency domain system ID of a multirotor UAV from manual frequency sweep flight test data, where flight tests were conducted with an attitude command control law active. Due to the success of the identification in [58], a similar flight test procedure was used in this work, which consists of the following steps:

1. The pilot brings the UAV to the desired trim condition and holds this condition for at least 5 seconds.

2. The pilot begins manually sweeping the aircraft on a desired axis, beginning with slow, long-period commands and gradually increasing the frequency until manual limitations are met (the pilot cannot oscillate the stick any faster).

3. The aircraft is again held in a stable trim for at least 5 seconds.

Recall that the PX4 stabilized mode controller was active during flight test, as the bare airframe dynamics of the UAVs are highly unstable and impractical to fly open-loop. The swept pilot stick input therefore equates to swept roll angle, pitch angle, and yaw rate commands for the lateral, longitudinal, and directional axes. For heave, however, there is no feedback compensation and the stick inputs map directly to throttle inputs. A sample time history showing a roll command sweep performed with the quadcopter is shown in Fig. 2.5.

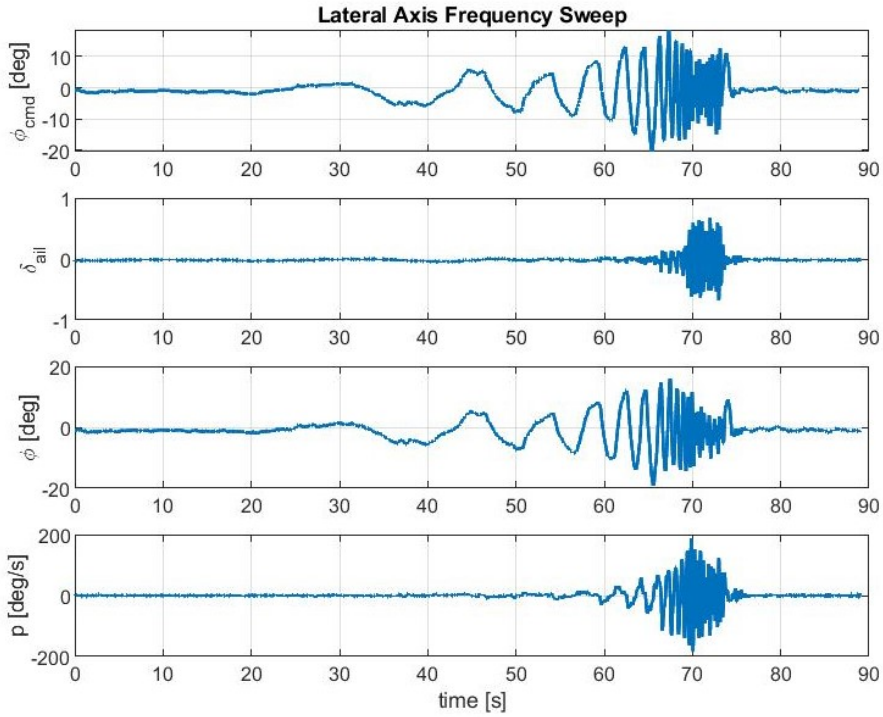


Figure 2.5: Lateral axis frequency sweep used for quadcopter model ID. Note that δ_{ail} is the PX4 bare airframe input shown in Fig. 2.3

All time history data used for system ID was recorded at a rate of 100 Hz. This includes pilot stick inputs, attitude commands, angular rate commands, and bare-airframe inputs, as well as the UAV attitude, body axis angular rate, and body axis velocity estimates produced by the PX4 extended Kalman filter. Note that the system ID flight tests were conducted in a motion capture facility, and motion capture position and heading measurements were streamed from the ground station computer to the UAV at a rate of 100 Hz. This allowed for accurate position, velocity, and heading estimates to be obtained.

2.2.4 Identified Models

The flight test procedure discussed in the previous section was used to identify models of the quadcopter and hexacopter UAVs operating around hover. These models were found sufficiently accurate in the low speed forward flight regime as well, allowing the controllers used for autonomous landing experiments to be designed around a single linear model.

For the quadcopter, bare airframe models were identified directly using the transfer function modelling capabilities of CIFER® (note that bare airframe here refers to the aircraft dynamics plus the mixer, as was shown in Fig. 2.3). Referring back to Eq. (2.7), CIFER® requires the user to provide the transfer function numerator and denominator order, as well as to specify whether or not to include a time delay. Based on other work on the system ID of multirotor UAVs (references [52, 58–60]) it was first assumed that the quadcopter dynamics could be captured by transfer functions of the form

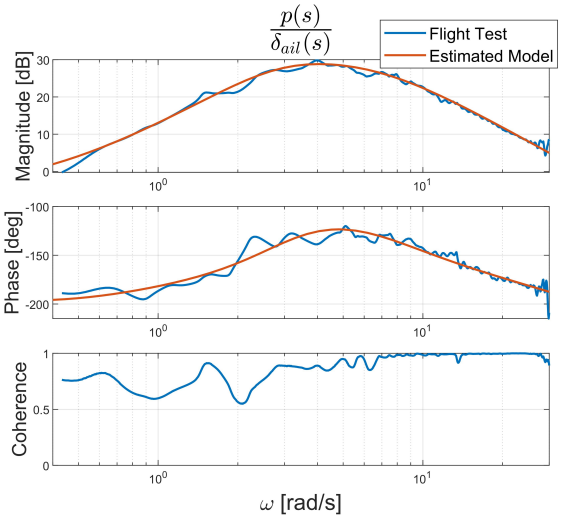
$$\begin{aligned}
\frac{p(s)}{\delta_{ail}(s)} &= \frac{L_{\delta_{ail}} s(s - Y_v)}{s^3 - Y_v s^2 - L_v g} e^{-\tau_{lat} s} \\
\frac{q(s)}{\delta_{ele}(s)} &= \frac{M_{\delta_{ele}} s(s - X_u)}{s^3 - X_u s^2 + M_u g} e^{-\tau_{lon} s} \\
\frac{r(s)}{\delta_{rud}(s)} &= \frac{N_{\delta_{rud}}}{s - N_r} e^{-\tau_{rud} s} \\
\frac{w(s)}{\delta_{thr(s)}} &= \frac{Z_{\delta_{thr}}}{s - Z_w} e^{-\tau_{thr} s}
\end{aligned} \tag{2.16}$$

where L_v, Y_v, M_u, X_u, N_r, and Z_w are the typical stability derivatives and $L_{\delta_{ail}}$, $M_{\delta_{ele}}$, $N_{\delta_{rud}}$, and $Z_{\delta_{thr}}$ are control derivatives. Since the stability and control derivatives are not identified directly by CIFER®'s transfer function modelling tools, just the order of the numerator and denominator from each transfer function shown in Eq. (2.16) was used to specify the model structure. It was found, however, that adding an additional pole to all four transfer functions and an additional zero to the yaw rate dynamics led to a better model fit. The quadcopter models identified with the updated structure are shown in table 2.1, and aircraft frequency response, model fit, and coherence are shown in Fig. 2.6.

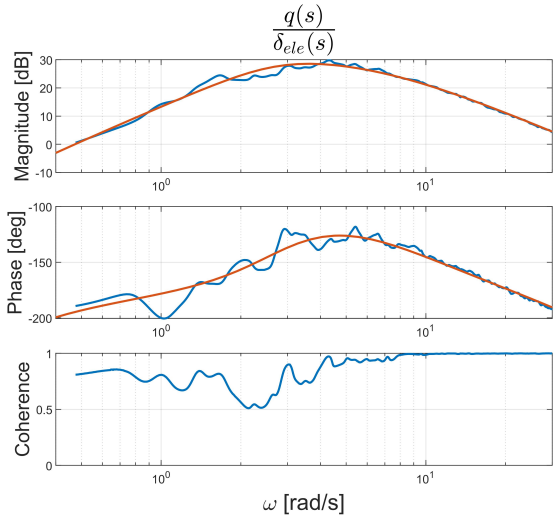
Looking at Fig. 2.6, excellent agreement is seen between the frequency response extracted from flight test and that of the estimated models. The yaw and vertical axes also show excellent coherence. On the pitch and roll axes the coherence does dip below the recommended limit of 0.6, but only slightly and for small frequency ranges. Overall, the

Table 2.1: Quadcopter Identified Models

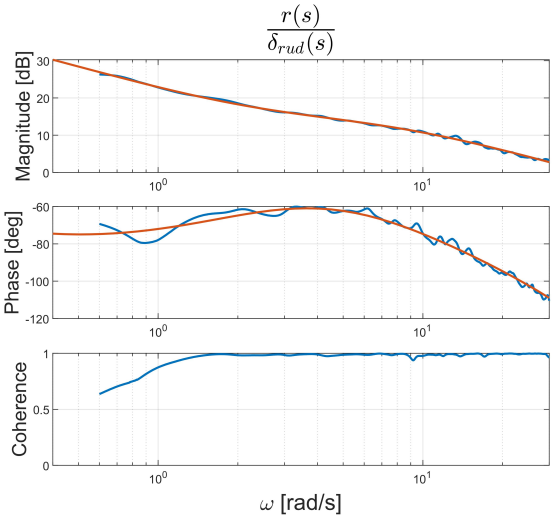
$\frac{p(s)}{\delta_{ail}(s)}e^{-\tau_{lat}s}$	$\frac{1624.8(s+0.5549)(s-0.2694)}{(s^2-3.823s+8.761)(s^2+11.38s+49.09)}e^{-0.0125s}$
$\frac{q(s)}{\delta_{ele}(s)}e^{-\tau_{lon}s}$	$\frac{1568.1(s^2+0.1455s+0.02292)}{(s^2-3.569s+7.731)(s^2+12.67s+43.07)}e^{-0.0159s}$
$\frac{r(s)}{\delta_{rud}(s)}e^{-\tau_{rud}s}$	$\frac{42.543(s+2.431)}{(s+7.906)(s+0.06621)}e^{-0.0170s}$
$\frac{w(s)}{\delta_{thr}(s)}e^{-\tau_{thr}s}$	$\frac{-518.22}{(s+8.008)(s+0.3203)}e^{-0.0100s}$



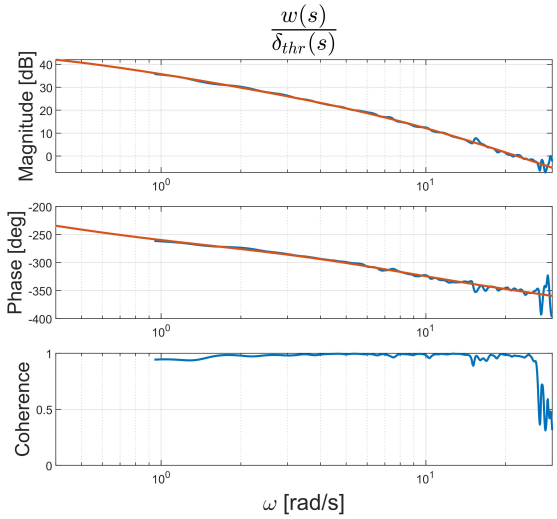
(a) δ_{ail} to roll rate.



(b) δ_{ele} to pitch rate.



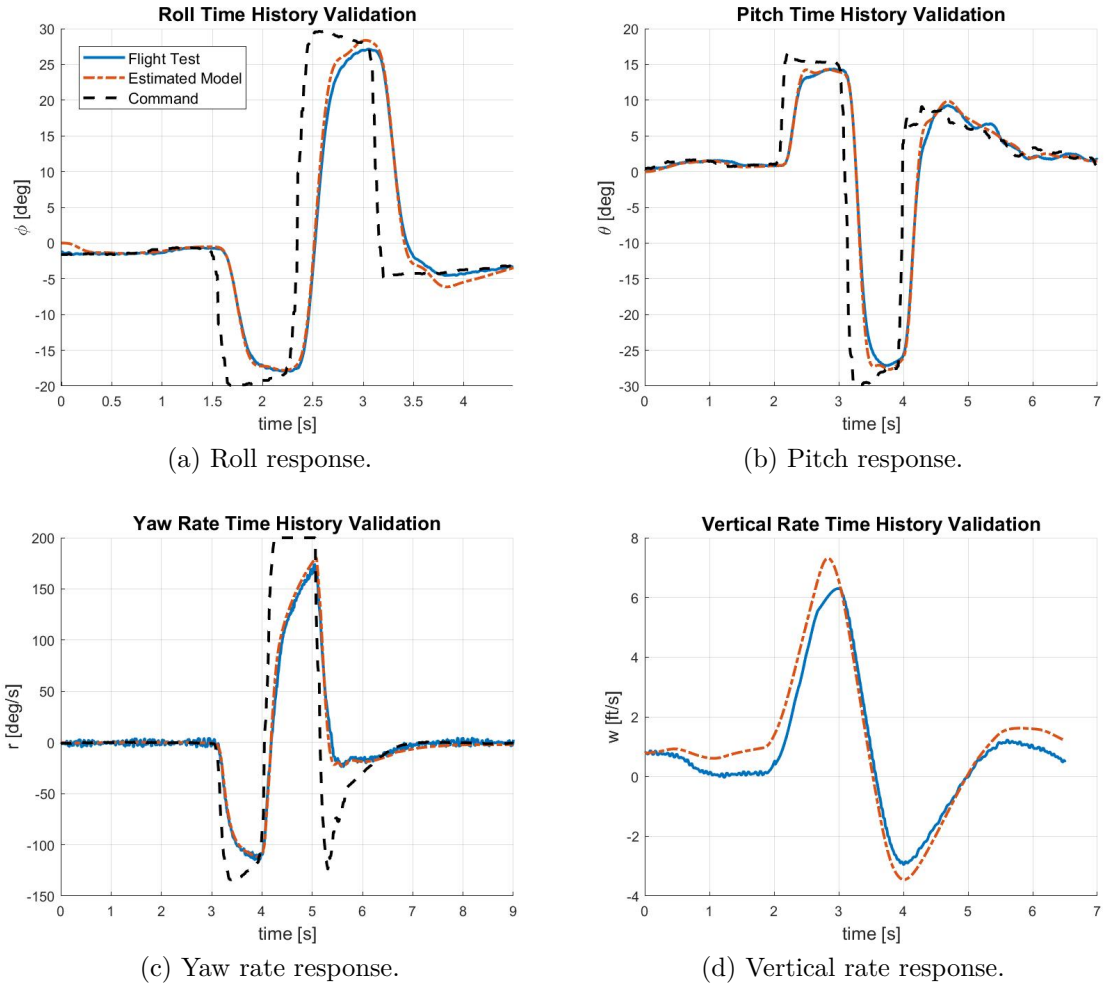
(c) δ_{rud} to yaw rate.



(d) δ_{thr} to vertical rate.

Figure 2.6: Quadcopter frequency response and transfer function model fits.

results shown in Fig. 2.6 indicate a successful identification of the quadcopter dynamics, which is confirmed by the time history validation plots shown in Fig. 2.7. To produce the time history validation, flight tests were conducted where doublet commands were input to the closed loop system on each axis. Closed-loop simulations were then run for comparison, where the closed loop models were formulated based on the linearized PX4 stabilized mode control laws discussed in section 2.2.2. The results show excellent agreement with flight test data for the roll, pitch, and yaw responses, and good agreement on the vertical axis as well.



(a) Roll response.

(b) Pitch response.

(c) Yaw rate response.

(d) Vertical rate response.

Figure 2.7: Quadcopter model time history validation.

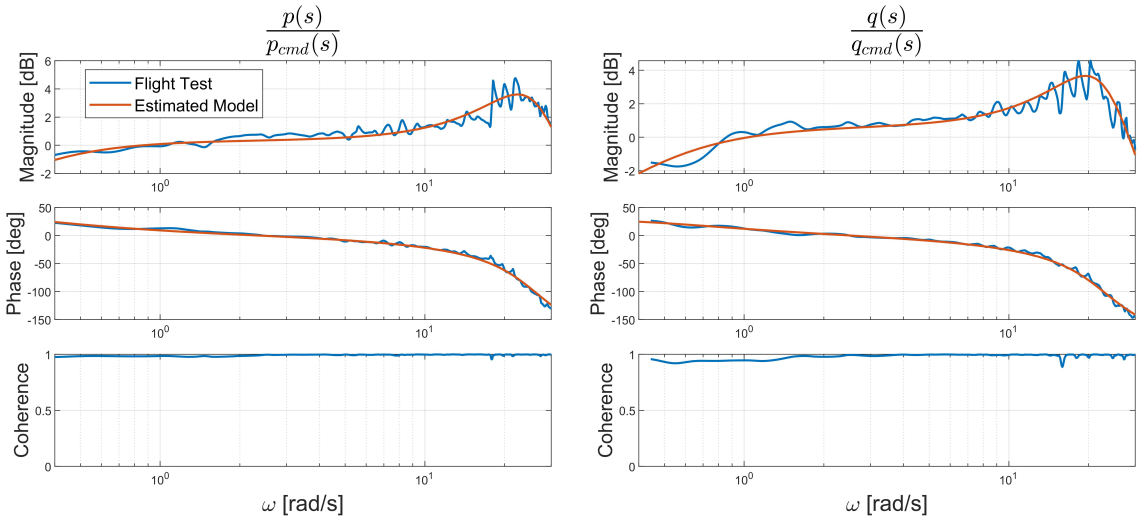
While the bare airframe model identification was successful for the quadcopter, accurate bare airframe frequency responses with sufficiently high coherence could only be extracted on the yaw and vertical axes of the hexacopter, and not on the pitch and roll axes. This likely could have been overcome by using the flight test procedures

presented by Ivler in [52], where the frequency sweeps were automated and injected into the closed loop system at the bare airframe input. For this work, however, it was deemed sufficient to simply identify models from roll rate command to aircraft roll rate (i.e. $\frac{p(s)}{p_{cmd}(s)}$) and pitch rate command to aircraft pitch rate (i.e. $\frac{q(s)}{q_{cmd}(s)}$) with the PX4 angular rate controller active. The closed loop roll and pitch rate dynamics were then considered as the plant when designing the roll and pitch attitude controllers. The main drawback to this is that it precludes the determination of stability margins by breaking the control loop at the bare airframe input. For these small UAVs, however, stability can be easily and safely verified through flight test.

Using this approach, the models estimated for the hexacopter are shown in table 2.2 and the hexacopter frequency response, model fit, and coherence are shown in Fig. 2.8. Looking at table 2.2 and Fig. 2.8, we see that yaw rate and vertical rate models were identified with the same structure as that used for the quadcopter and again agree well with flight test data in the frequency domain. Further, the yaw rate and vertical rate responses maintain high coherence over the frequency range of interest. The roll rate and pitch rate frequency response also show an excellent model fit and very high coherence. The high coherence is expected as the command roll and pitch rates are now considered as the inputs, meaning the filtering effects of the angular rate controller do not effect the extracted frequency response as they did for the quadcopter bare airframe identification. Overall, the results shown in Fig. 2.8 indicate successful fits to the chosen responses for the hexacopter UAV. This is confirmed by the time history comparisons shown in Fig. 2.9, where the estimated models provide an excellent prediction of the flight test data. In fact, the simulated pitch and roll responses shown in Fig. 2.9 are nearly indistinguishable from the flight test data.

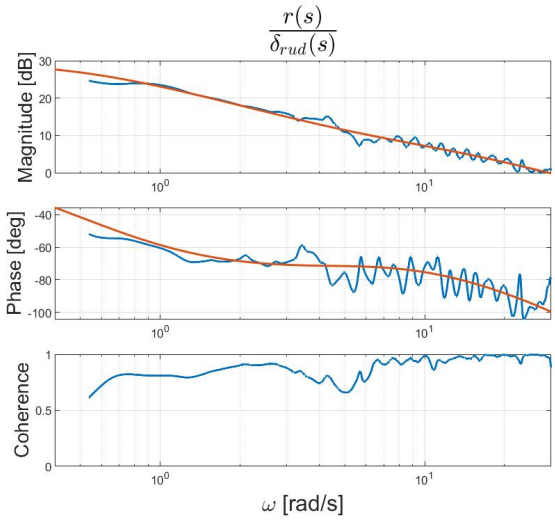
Table 2.2: Hexacopter Identified Models

$\frac{p(s)}{p_{cmd}(s)}e^{-\tau_{lat}s}$	$\frac{717.91(s+0.05038)}{(s+0.2513)(s^2+19.39s+691.1)}e^{-0.0084s}$
$\frac{q(s)}{q_{cmd}(s)}e^{-\tau_{lon}s}$	$\frac{568.01(s+0.1607)}{(s+0.4367)(s^2+17.46s+530.6)}e^{-0.0097s}$
$\frac{r(s)}{\delta_{rud}(s)}e^{-\tau_{rud}s}$	$\frac{31.259(s+6.609)}{(s+12.92)(s+0.5291)}e^{-0.0125s}$
$\frac{w(s)}{\delta_{thr}(s)}e^{-\tau_{thr}s}$	$\frac{-964.11}{(s+14.78)(s+0.3974)}e^{-0.0132s}$

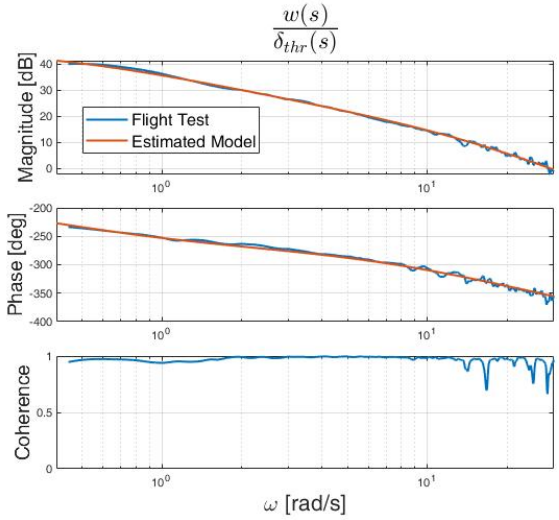


(a) Roll rate command to roll rate.

(b) Pitch rate command to pitch rate.

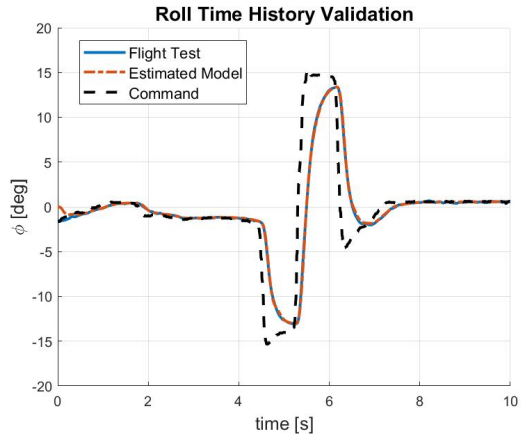


(c) δ_{rud} to yaw rate.

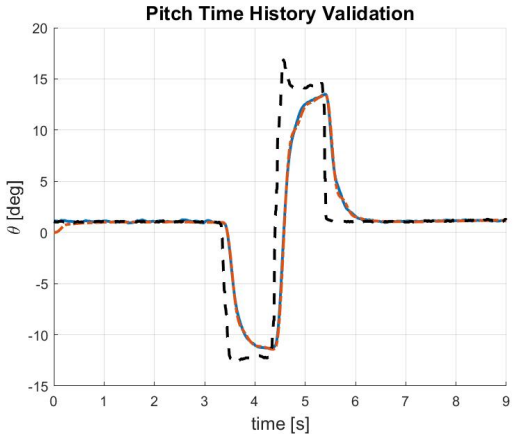


(d) δ_{thr} to vertical rate.

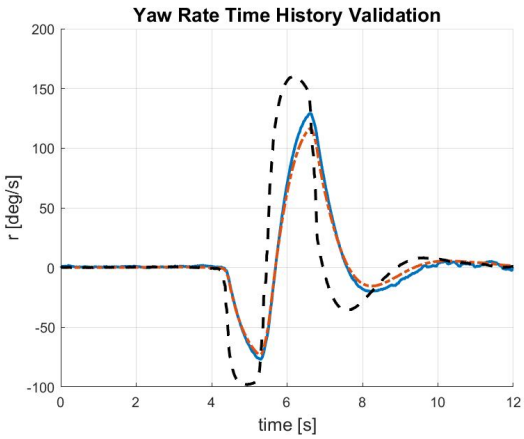
Figure 2.8: Hexacopter frequency response and transfer function model fits.



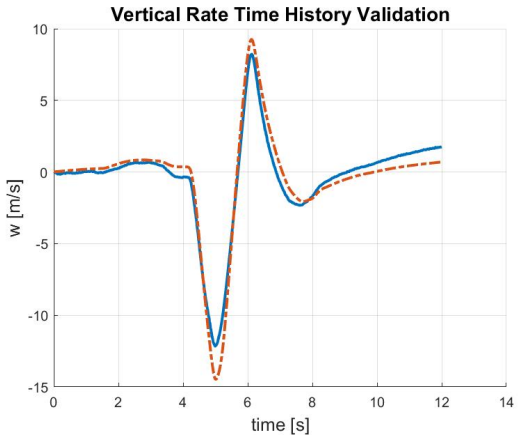
(a) Roll response.



(b) Pitch response.



(c) Yaw rate response.



(d) Vertical rate response.

Figure 2.9: Hexacopter model time history validation.

2.3 Control Architecture

Position and heading command control laws were developed using the nested control architecture shown from a high-level in Fig. 2.10. Note that the position and attitude control laws are run on the Odroid XU4 onboard computer and produce angular rate commands that are sent to the PX4 angular rate controller that runs on the Pixhawk autopilot.

The choice was made to send roll and pitch rate commands to PX4 because the bare airframe model ID was unsuccessful on the hexacopter roll and pitch axes, so models from roll and pitch rate commands to roll and pitch rate output were determined instead. While bare airframe models were determined for the quadcopter, formulating the attitude controllers to command roll and pitch rate allowed identical control structures to be used on both UAVs. The choice was made to also send yaw rate commands because it was desired to run the attitude controllers on the onboard computer carried by the UAV, and then send commands to the Pixhawk flight controller. This prevented the need to overwrite the default control modes available in the PX4 flight control firmware, keeping them immediately available as a fail-safe. In order to send commands from the onboard computer to PX4, though, the ROS package mavros was used. Mavros has built in functionality that enables sending a combination of angular rate commands and throttle inputs to PX4. Mavros does not, however, have the built in functionality to send a combination of pitch and roll rate commands and inputs δ_{rud} and δ_{thr}. Formulating the attitude control laws to send roll, pitch, and yaw rate commands therefore allowed for identical control structures to be used on both aircraft while also taking advantage of the convenient communication capabilities provided by mavros.

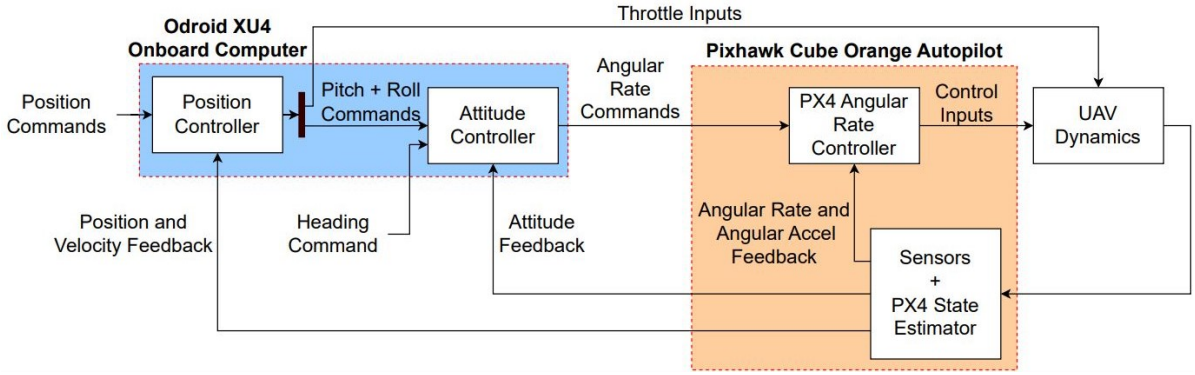


Figure 2.10: High-level control law block diagram.

2.4 Angular Rate Control

The PX4 angular rate controllers use the same control law as that shown in the inner loop of Fig. 2.4 and discussed in chapter 2.2.2. The main features of the control law are PI compensation on the angular rate error signal and filtered derivative (angular acceleration) feedback. For both the hexacopter and quadcopter UAVs, the angular rate control gains remained fixed at values which had previously been empirically tuned by a pilot. These control gains are tabulated for each control axis in table 2.3. Additionally, the filter on derivative feedback shown in Fig. 2.4 used a damping ratio of $\zeta_f = 0.707$ and frequency of $\omega_f = 20$ Hz on all axes.

Table 2.3: Angular rate control parameters for each axis.

	Quadcopter			**Hexacopter**		
	$\boldsymbol{K_P}$	$\boldsymbol{K_I}$	$\boldsymbol{K_D}$	$\boldsymbol{K_P}$	$\boldsymbol{K_I}$	$\boldsymbol{K_D}$
Roll Rate	0.1300	0.3800	0.0060	0.2250	0.8250	0.0045
Pitch Rate	0.1300	0.3800	0.0060	0.2250	0.8250	0.0045
Yaw Rate	0.2000	0.6000	0	0.6000	0.4800	0

2.5 Attitude Control

The explicit model following (EMF) control architecture [61] was used to create attitude-command-attitude-hold controllers for roll, pitch, and yaw. EMF features a command filter to shape reference commands, a feedforward inversion model to approximately cancel plant dynamics, and linear feedback compensation. A time delay on commands is also included to account for phase lag due to higher frequency dynamics not included in the inversion model. This architecture is shown for the pitch attitude controller in Fig. 2.11, where the feedback compensation is chosen as simple proportional-plus-integral (PI) control. An identical structure was used for the roll and yaw attitude controllers.

EMF was the control method of choice here because, with an accurate inversion model, EMF forces the reference tracking response to approximate the command filter with added input delay. For example, for pitch this allows the reference tracking response to be approximated as

$$\frac{\theta(s)}{\theta_{cmd}(s)} \approx G_{cf,\theta}(s)e^{-\tau_q s} \tag{2.17}$$

Additionally, an accurate inversion model effectively partitions the reference tracking and disturbance rejection responses of the closed loop system. These properties of EMF

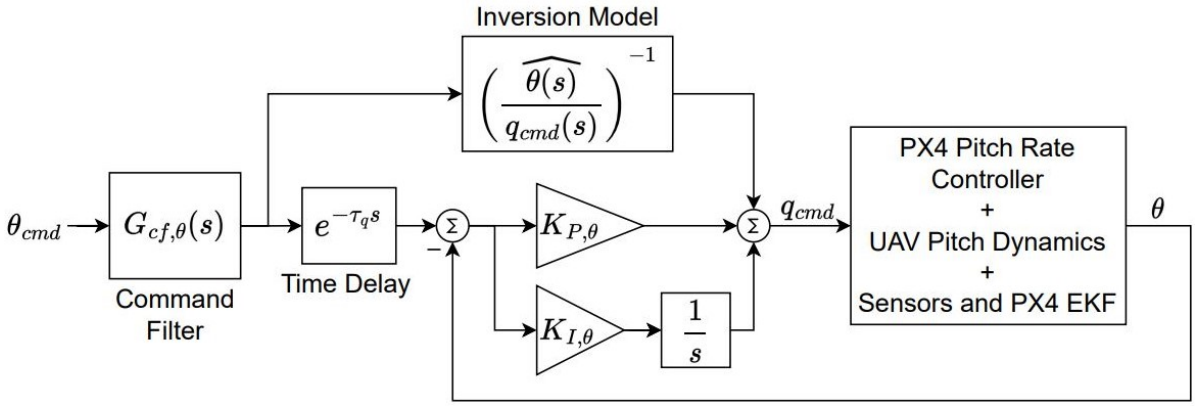


Figure 2.11: Pitch attitude controller.

make for easy tailoring of closed loop bandwidths and will be leveraged in Chapter 4.4 to systematically scale the UAV closed loop dynamics. Further, the EMF control architecture lends itself nicely to control design with transfer function models like those identified for the UAVs used in this work.

To design the attitude controller inversion models, the identified aircraft models are used. Since the attitude controller output consists of angular rate commands, the feedforward inversion model must be based on the attitude response due to a rate command. For the hexacopter pitch and roll axes, this is obtained by simply integrating the output of the identified models shown in table 2.2 and ignoring the time delay:

$$\begin{aligned}\left(\frac{\widehat{\phi(s)}}{p_{cmd}(s)}\right)_{hex} &= \frac{717.91(s+0.05038)}{s(s+0.2513)(s^2+19.39s+691.1)} \\ \left(\frac{\widehat{\theta(s)}}{q_{cmd}(s)}\right)_{hex} &= \frac{568.01(s+0.1607)}{s(s+0.4367)(s^2+17.46s+530.6)}\end{aligned} \tag{2.18}$$

where $(\cdot)_{hex}$ denotes that these inversion models are for the hexacopter. The identified time delay is then placed on the reference command as shown in Fig. 2.11:

$$\begin{aligned}\left(e^{-\tau_p s}\right)_{hex} &= e^{-0.0084s} \\ \left(e^{-\tau_q s}\right)_{hex} &= e^{-0.0097s}\end{aligned} \tag{2.19}$$

For the quadcopter attitude controllers and hexacopter yaw controller, however, the PX4 rate control loop must first be closed around the identified bare airframe models. The inversion model is then obtained by integrating the angular rate output to yield attitude. Using the quadcopter pitch axis as an example and referring back to the PX4 rate control

loop depicted in Fig. 2.4, this yields

$$\left(\widehat{\frac{\theta(s)}{q_{cmd}(s)}}\right)_{quad} = \frac{\frac{q(s)}{\delta_{ele}(s)}e^{-\tau_{lon}s}\left(K_{P,q}+\frac{K_{I,q}}{s}\right)}{s\left[1+s\frac{q(s)}{\delta_{ele}(s)}e^{-\tau_{lon}s}\left(K_{P,q}+\frac{K_{I,q}}{s}\right)\frac{\omega_f^2}{s^2+2\zeta_f\omega_f s+\omega_f^2}K_{D,q}\right]} \tag{2.20}$$

An issue that arises when evaluating Eq. (2.20) is that an internal delay appears due to the loop closure around the time delay $e^{-\tau_{lon}s}$. This is undesirable as the inversion model is most conveniently expressed as a transfer function. To circumvent this issue, the time delay can be approximated as a first order Padé approximation:

$$e^{-\tau_{lon}s} \approx \frac{-(s-\frac{2}{\tau_{lon}})}{(s+\frac{2}{\tau_{lon}})} \tag{2.21}$$

Eq. (2.21) can then be substituted into Eq. (2.20) along with the estimates of $q(s)/\delta_{ele}(s)$ and τ_{lon} from table 2.1 and the PX4 controller parameter values given in section 2.4. This results in the transfer function

$$\left(\widehat{\frac{\theta(s)}{q_{cmd}(s)}}\right)_{quad} = \frac{-203.85(s-125.8)(s+2.923)(s^2+0.1455s+0.02292)(s^2+177.7s+1.579e04)}{s(s+78.69)(s+2.244)(s+1.164)(s+0.0337)(s^2+16.88s+221.5)(s^2+213.6s+1.768e04)} \tag{2.22}$$

which no longer has an internal delay but does contain very high frequency poles and zeros, including a non-minimum phase zero due to the Padé approximation which would make the inverted model unstable. This can be corrected by neglecting the dynamics added by the high frequency poles and zeros (i.e. setting $s = 0$ for these cases) and accounting for the phase effects of neglected dynamics through an added time delay. Following this process, Eq. 2.22 reduces to

$$\left(\widehat{\frac{\theta(s)}{q_{cmd}(s)}}e^{-\tau_q s}\right)_{quad} = \frac{291.05(s+2.923)(s^2+0.1455s+0.02292)}{s(s+2.244)(s+1.164)(s+0.0337)(s^2+16.88s+221.5)}e^{-0.021s} \tag{2.23}$$

which consists of a valid inversion model and added time delay while still accurately representing the dynamics of the exact model with an internal delay derived in Eq. 2.20. This is confirmed by the frequency response comparison shown in Fig. 2.12, where the two different representations show a nearly identical frequency response out to at least 30 rad/s. This same process also proved to be effective for deriving inversion model and time delay parameters for the quadcopter pitch and yaw axes and hexacopter yaw axis.

The resulting expressions are as follows:

$$\begin{aligned}
\left(\widehat{\frac{\phi(s)}{p_{cmd}(s)}e^{-\tau_p s}}\right)_{quad} &= \frac{287.25(s+2.923)(s+0.5549)}{s(s^2+3.178s+3.676)(s^2+15.7s+245)}e^{-0.017s} \\
\left(\widehat{\frac{\psi(s)}{r_{cmd}(s)}e^{-\tau_r s}}\right)_{quad} &= \frac{10.166(s+3)(s+2.431)}{s(s+15.47)(s^2+3.211s+4.794)}e^{-0.0185s} \\
\left(\widehat{\frac{\psi(s)}{r_{cmd}(s)}e^{-\tau_r s}}\right)_{hex} &= \frac{26.914(s+6.609)(s+0.8)}{s(s+37.78)(s+4.552)(s+0.8272)}e^{-0.015s}
\end{aligned} \tag{2.24}$$

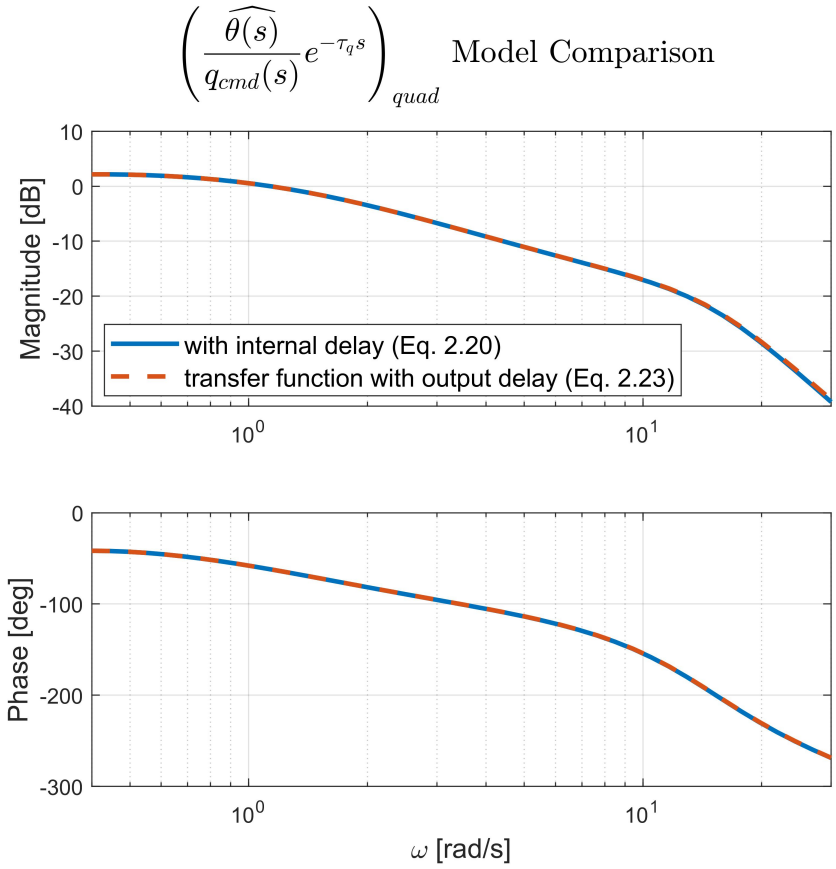


Figure 2.12: Comparison of quadcopter pitch axis inversion model with internal delay (derived from Eq. 2.20) and transfer function approximation with output delay (shown in Eq. 2.23)

With inversion models defined, the command filter structure can be chosen. Again using the quadcopter pitch axis as an example, note that the inversion model $\widehat{\frac{\theta(s)}{q_{cmd}(s)}}$ shown in Eq. (2.23) is of relative degree three. Inverting this will therefore result in an improper transfer function. This is avoided in practice by distributing the pitch

command filter shown in Fig. 2.11 to both the time delay and feedforward inverse plant. The command filter structure is then chosen such that the product of the command filter and inverted plant results in a proper transfer function. For the quadcopter pitch axis the command filter was therefore chosen to be a third order filter of the form

$$G_{\theta,cf} = \left(\frac{\omega_{\theta,cf}^2}{s^2 + 2\zeta\omega_{\theta,cf}s + \omega_{\theta,cf}^2}\right)\left(\frac{p_\theta}{s + p_\theta}\right) \tag{2.25}$$

where ω_θ is set to govern the reference tracking bandwidth and the pole at p_θ is placed at high frequency and is included to make the product $G_{\theta,cf}\left(\widehat{\frac{\theta(s)}{q_{cmd}(s)}}\right)^{-1}$ proper. Here it was found that simply fixing p_θ at 50 rad/s allowed the pitch response to be governed by the lower frequency ω_θ without causing any practical issues. The same command filter was used for the hexacopter pitch axis.

Following the same logic as the pitch attitude controller, a command filter structure identical to that of Eq. 2.25 was chosen for both the quadcopter and hexacopter roll attitude controllers:

$$G_{\phi,cf} = \left(\frac{\omega_{\phi,cf}^2}{s^2 + 2\zeta\omega_{\phi,cf}s + \omega_{\phi,cf}^2}\right)\left(\frac{p_\phi}{s + p_\phi}\right) \tag{2.26}$$

with p_ϕ also fixed at 50 rad/s. For both the hexacopter and quadcopter yaw axis, though, the inversion model $\widehat{\frac{\psi(s)}{r_{cmd}(s)}}$ shown in Eq. 2.24 is of relative degree two. A second order command filter of the form

$$G_{\psi,cf} = \frac{\omega_{\psi,cf}^2}{s^2 + 2\zeta\omega_{\psi,cf}s + \omega_{\psi,cf}^2} \tag{2.27}$$

was therefore used for yaw attitude control. Note that the process for choosing the value of the command filter frequencies ω_θ, ω_ϕ, and ω_ψ will be discussed in section 4.4.1. Additionally, the process for choosing the attitude control feedback gains will be discussed in section 4.4.3.

2.6 Position Control

The position control law is divided into two separate controllers: one for controlling inertial X^I and Y^I translations, and one for controlling altitude (i.e. inertial Z^I position). Similar to the attitude control laws, both the translational position and altitude controllers utilize the EMF control architecture.

2.6.1 X and Y Position Control

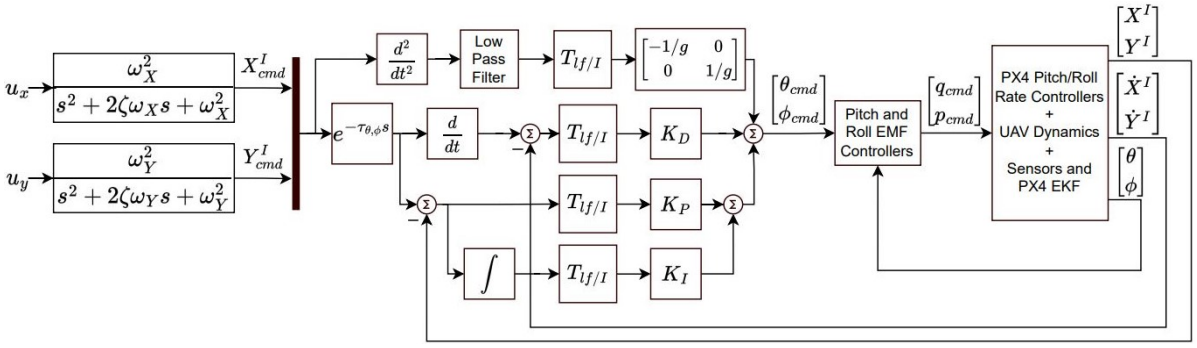


Figure 2.13: Inertial X and Y position controller.

A block diagram of the X^I and Y^I position control law can be seen in the outer loop of Fig. 2.13, which outputs pitch and roll commands to the lower level attitude controllers. The outer loop controller uses PID feedback compensation and second order command filters, where the command filter inputs u_x and u_y are produced by the trajectory generation algorithm (see Chapter 3). This same control law was used for both the hexacopter and quadcopter UAVs.

To determined the outer loop inversion model, the UAV level frame translational dynamics are modelled assuming that the aircraft will operate in hover and low speed flight during shipboard landing tasks. Note that the UAV level frame is defined by rotating the inertial frame X and Y axes to align with the UAV heading. The rotation matrix to transform from inertial frame to UAV level frame coordinates is therefore calculated as

$$T_{lf/I} = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.28}$$

Assuming small pitch and roll angles and low vertical accelerations, the UAV level frame translational accelerations can be approximated as

$$\begin{aligned} \ddot{X}^{lf} &= X_{v,lon}\dot{X}^{lf} - g\theta \\ \ddot{Y}^{lf} &= Y_{v,lat}\dot{Y}^{lf} + g\phi \end{aligned} \tag{2.29}$$

where coefficients $X_{v,lon}$ and $Y_{v,lat}$ characterize drag effects and $-g\theta$ and $g\phi$ account for thrust vectoring due to change in attitude. For low-speed flight, the translational dynamics are dominated by the thrust-vectoring terms and $X_{v,lon}$ and $Y_{v,lat}$ can be

dropped from Eq. 2.29 , resulting in the further simplified model

$$\begin{aligned} \ddot{X}^{lf} &= -g\theta \\ \ddot{Y}^{lf} &= g\phi \end{aligned} \tag{2.30}$$

Treating pitch and roll as the driving inputs for control design, assuming perfect pitch and roll command tracking, and taking the Laplace transform of Eq. 2.30 yields

$$\begin{aligned} \frac{X^{lf}(s)}{\theta_{cmd}(s)} &= \frac{-g}{s^2} \\ \frac{Y^{lf}(s)}{\phi_{cmd}(s)} &= \frac{g}{s^2} \end{aligned} \tag{2.31}$$

Inverting this leads to the level frame "inversion model" of acceleration feedforward with gains of $-1/g$ and $1/g$, as is shown in Fig. 2.13. The control law is designed to follow inertial position commands, however, so the desired inertial X and Y accelerations are first transformed to the level frame prior to applying the feedforward control gains. Note that the inertial acceleration feedforward command is passed through a low pass filter prior to applying the transformation. This is done because the quadratic programming based trajectory generation law (discussed in Chapter 3) outputs position commands u_x and u_y at 10 Hz, but the EMF control laws execute as a separate process at 100 Hz. The result is significant oscillation in the acceleration commands occurring at a frequency of 10 Hz. Including a second order low pass filter with a frequency of 25 rad/s and damping ratio of 0.707 was found to effectively attenuate these oscillations without introducing enough lag to harm performance.

The level frame model shown in Eq. 2.31 can also be used to design the PID control gains in the aircraft level frame. Similar to the feedforward compensation though, the inertial frame feedback errors must be transformed to the level frame prior to applying the feedback gains. The specific process used for determining PID control gains will be discussed in section 4.4.3. Additionally, the process for determining the command filter coefficients ω_X and ω_Y and time delay $\tau_{\theta,\phi}$ is discussed in section 4.4.1.

2.6.2 Altitude Control

A block diagram of the altitude control law is shown in Fig. 2.14. The control structure is similar to the that used for X and Y position control, with a second order command filter and PID feedback compensation (for discussion on the choice of command filter

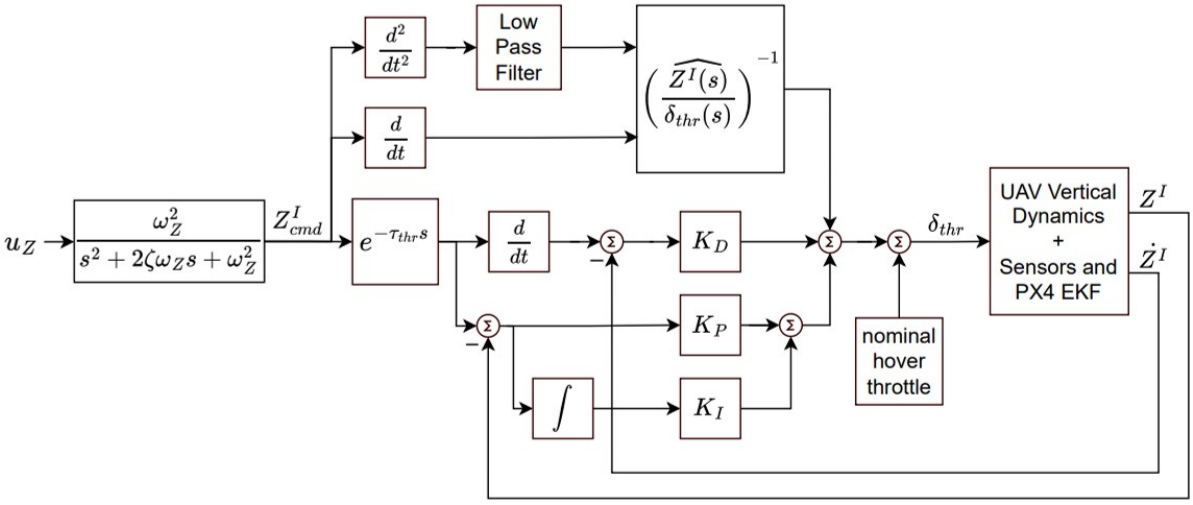


Figure 2.14: Altitude control law.

frequency and feedback gains see sections 4.4.1 and 4.4.3). The main differences in the altitude control structure are the absence of an inner control loop and the injection of the nominal hover throttle into the bare airframe input.

With the absence of an inner control loop, the feedforward inversion model is determined directly from the vertical axis system ID results. Again assuming small pitch and roll angles so that the body axis vertical rate w is approximately equivalent to the altitude rate $\dot{Z}^I$, the altitude inversion model is given as

$$\frac{\widehat{Z^I(s)}}{\delta_{thr}(s)} = \frac{1}{s}\frac{w(s)}{\delta_{thr}(s)} \tag{2.32}$$

where $w(s)/\delta_{thr}(s)$ is tabulated in tables 2.1 and 2.2 for the quadcopter and hexacopter, respectively. The altitude command delay τ_{thr} is set equal to the values given in tables 2.1 and 2.2 as well.

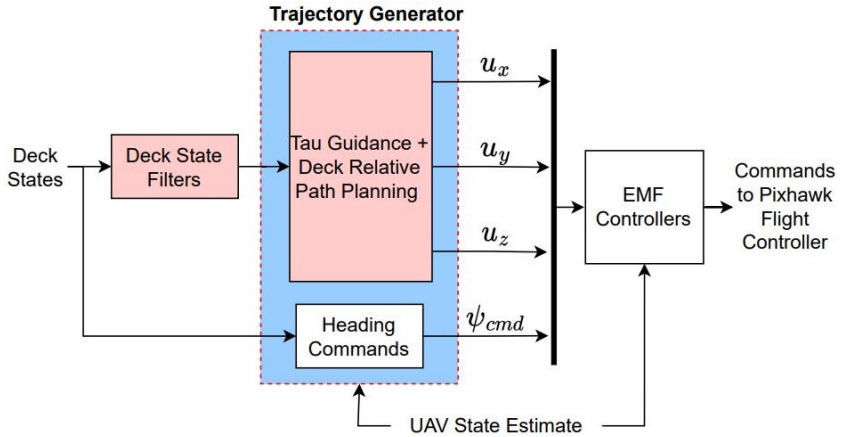
Note that the altitude controller feedforward acceleration term is low pass filtered, where the filter was specified to be identical to that used in the X and Y controller feedforward path. This low pass filter was included in the altitude controller for the same reason it was included in the X and Y position controller: the quadratic programming based guidance algorithm discussed in Chapter 3 runs at a lower rate than the EMF control laws. The result is high frequency oscillations in the command filter acceleration output, which is problematic in the lead-heavy feedforward channel without additional filtering.

Chapter 3
Autonomous Landing Guidance Algorithms

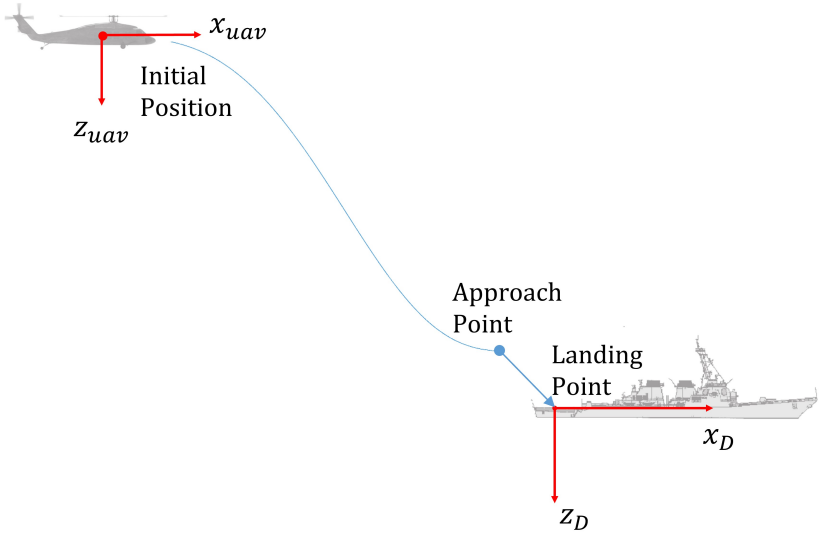
Two different guidance algorithms were developed for performing autonomous ship landing experiments. The first is a simple deck-tracking algorithm (referred to here as the baseline algorithm), which commands a deck-relative flight path, closing the distance between the aircraft and deck at a constant rate. The second is a quadratic programming (QP) based landing algorithm that performs real-time landing path optimization and plans the trajectory to a predicted deck state. This chapter will describe the both the baseline and QP guidance algorithms, including the method for producing the deck motion predictions incorporated into QP guidance method.

3.1 Baseline Guidance Algorithm

A high-level block diagram of the baseline guidance algorithm is shown in Fig, 3.1a and a schematic of the planned trajectory is shown in Fig. 3.1b. Referring to Fig. 3.1a, the baseline landing algorithm takes the current deck state measurements as input, filters the deck states, and then uses the filtered deck states to produce position and heading commands that are input to the EMF control law discussed in Chapter 2. As depicted in Fig. 3.1b, the produced trajectory commands are broken into two stages: an approach stage and a landing stage. The approach stage brings the aircraft from its initial position to the so-called "approach point", which is a fixed position relative to the mean motion of the landing deck. For the model-scale experiments that will be discussed in this work, the approach point was set to 0.5 meters behind and 0.75 meters above the mean deck motion. A height of 0.75 meters was chosen so that for the largest heave oscillations the UAV landing gear would remain approximately 0.25 meters above the deck, which



(a) High-level diagram of baseline guidance algorithm.



(b) Schematic of baseline landing trajectory.

Figure 3.1: Baseline guidance algorithm block diagram and trajectory schematic.

corresponds to about 3.5 meters (about 11 feet) at full-scale when applying the scaling method discussed in Chapter 4.1. The 0.5 meter offset aft of the deck was then chosen so that a camera onboard the UAV could maintain a view of the deck during the approach. The onboard camera was not used in this study, but it was desired to gather data with the camera as part of a separate research effort. Once the approach point is reached, the aircraft is commanded to hold this position for two seconds. After two seconds the landing phase is initiated and the aircraft is commanded to close the distance between the aircraft and deck at a constant rate, killing the throttle once the landing gear is determined to be less that five centimeters above the deck. This method was chosen as the baseline landing strategy as it is a simple trajectory that has been commonly

implemented in the literature (for example, see [6, 27, 29, 32, 35]).

In [35] an altitude-based phase-in of deck motion was integrated into the landing trajectory to limit tracking deck oscillations when still far from the deck, though the specifics of the phase-in method were not discussed. Here, a complimentary filter is used to phase deck oscillations into the X and Y trajectory commands given by the baseline guidance algorithm. The complimentary filter separates the low frequency and high frequency deck motion. A variable gain is then placed on the high frequency portion of deck motion, which is scheduled with deck relative altitude to cut out oscillatory motion from the deck-relative commands when the UAV is far from the deck. This was found in model-scale simulations to help limit unnecessary pitching and rolling of the UAV that occurred when tracking the unfiltered deck X and Y oscillations well before landing. The complementary filter that was utilized is depicted for the deck X position in Fig. 3.2. Note that the high pass channel gain is saturated to be in the range $0 \leq K_{HPF} \leq 1$ and the filter parameters were set to $\zeta = 0.707$ and $\omega_{LPF} = 0.5\ rad/s$ for all axes. Also note that, in the scheduling function shown in Fig. 3.2, P_1 represents the deck-relative altitude where the high pass channel begins fading in and P_2 represents the deck-relative altitude where $K_{HPF} = 1$ (high pass motion is fully phased in). Here P_1 was set equal to the approach point altitude (0.75 meters above the deck), and P_2 was set to 0.1 meters so as to include full X and Y deck motions before landing. For the Z axis, $K_{HPF} = 0$ was used during the approach and $K_{HPF} = 1$ was used for the entire landing phase (there is no phase in of the high pass channel). This was done to avoid early collisions with the deck that could occur if the full deck altitude was not used in the landing phase commands.

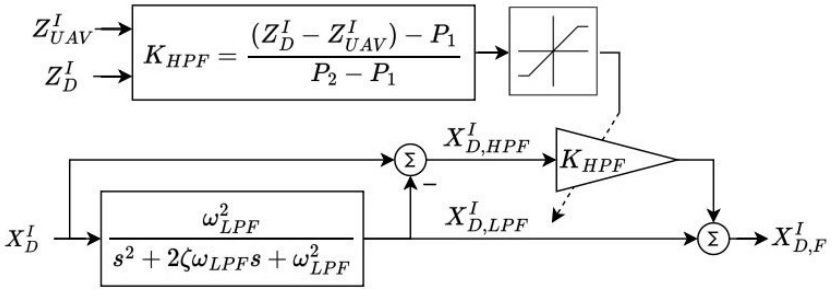


Figure 3.2: Deck filter used in baseline guidance algorithm.

To bring the aircraft from its initial location to the approach point, the higher order tau guidance algorithm presented by Holmes in [26] was used. This algorithm proposes a differential equation in terms of the distance to some goal and the velocity toward some

goal, denoted by χ and $\dot{\chi}$ respectively. The proposed differential equation is given as

$$\frac{\chi}{\dot{\chi}} = \frac{k}{3}\left(t - \frac{T_g^3}{t^2}\right) \tag{3.1}$$

where the parameter k can be used to adjust the aggressiveness of the trajectory, t is the time elapsed since the start of the trajectory, and T_g represents the total duration of the trajectory (i.e., at $t = T_g$, χ=0). Here a value of $k = 0.25$ was used, and T_g was calculated through the equation

$$T_g = 2.888\sqrt{\frac{|\chi_0|}{\ddot{\chi}_{max}}} \tag{3.2}$$

where χ_0 is the difference between the initial position and desired deck-relative position at the end of the approach phase. Determining T_g in this manner ensures that the maximum acceleration of the commanded trajectory is equal to $\ddot{\chi}_{max}$, which was set to $\ddot{\chi}_{max} = 0.5\ m/s^2$.

Solving Eq. (3.1) for $\chi(t)$ yields

$$\chi(t) = \frac{\chi_0}{T_g^{3/k}}\left(T_g^3 - t^3\right)^{1/k} \tag{3.3}$$

which is used to calculated a separate tau guidance trajectory for each translational degree of freedom. The initial gap χ_0 is therefore calculated for X, Y, and Z approach commands as

$$\begin{bmatrix}\chi_{0,X}^I \\ \chi_{0,Y}^I \\ \chi_{0,Z}^I\end{bmatrix} = \begin{bmatrix}X_{D,f}^I \\ Y_{D,f}^I \\ Z_{D,f}^I\end{bmatrix} + T_{I/dlf}\begin{bmatrix}X_{app}^{dlf} \\ Y_{app}^{dlf} \\ Z_{app}^{dlf}\end{bmatrix} - \begin{bmatrix}X_{0,UAV}^I \\ Y_{0,UAV}^I \\ Z_{0,uav}^I\end{bmatrix} \tag{3.4}$$

where $[X_{D,f}^I\ Y_{D,f}^I\ Z_{D,f}^I]^\intercal$ denotes the filtered deck positions expressed in the inertial frame, $[X_{app}^{dlf}\ Y_{app}^{dlf}\ Z_{app}^{dlf}]^\intercal$ denotes the desired end point of the approach expressed in the deck level frame, $T_{I/dlf}$ transforms the approach point from the deck level frame to the inertial frame, and $[X_{0,uav}^I\ Y_{0,uav}^I\ Z_{0,uav}^I]^\intercal$ is the position of the UAV at the start of the approach. The deck level frame is defined by rotating the inertial frame X and Y axes to align with the deck heading. The rotation matrix to transform from inertial to deck level frame

coordinates is therefore calculated as

$$T_{lf/I} = \begin{bmatrix} cos(\psi_D) & sin(\psi_D) & 0 \\ -sin(\psi_D) & cos(\psi_D) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

Using Eqs. (3.3) and (3.4), the inertial position command inputs for the EMF control law can be calculated as

$$\begin{bmatrix} u_x^I \\ u_y^I \\ u_z^I \end{bmatrix} = \begin{bmatrix} X_{D,f}^I \\ Y_{D,f}^I \\ Z_{D,f}^I \end{bmatrix} + T_{I/dlf} \begin{bmatrix} X_{app}^{dlf} \\ Y_{app}^{dlf} \\ Y_{app}^{dlf} \end{bmatrix} - \begin{bmatrix} \chi_X^I \\ \chi_Y^I \\ \chi_Z^I \end{bmatrix} \tag{3.6}$$

Once the approach phase is completed, the landing phase commences and altitude commands are calculated as

$$u_z^I = Z_{UAV,0}^I + (Z_D^I - Z_{D,0}^I) + \Delta t V_{z,b} \tag{3.7}$$

where subscript 0 denotes a position at the start of the landing phase, Δt is the time elapsed since the start of the landing phase, and $V_{z,b}$ is a constant descent rate relative to the deck. Here a value of $V_{z,b} = 0.25\ m/s$ was used. Similarly, the X and Y commands for the landing phase are calculated to follow the filtered deck positions, but with a small constant velocity bias. This is written as

$$\begin{bmatrix} u_x^I \\ u_y^I \end{bmatrix} = \begin{bmatrix} X_{UAV,0}^I + X_{D,f}^I - X_{D,f,0}^I + \Delta t V_{x,b} cos(\psi_D) \\ Y_{D,f}^I + \Delta t V_{x,b} sin(\psi_D) \end{bmatrix} \tag{3.8}$$

where $V_{x,b}$ is the velocity bias expressed along the X coordinate of the deck level frame. This term is included because the end location of the approach phase was specified to be 0.5 meters behind the landing point on the ship deck so that a camera onboard the UAV could maintain a view of the ship deck during the approach. The onboard camera was not used in this study, but it was desired to gather data with the camera as part of a separate research effort. This velocity bias was calculated so that the X position command at landing would be the center point of the deck if the UAV perfectly followed

to Z position commands:

$$V_{x,b} = \frac{X^{dlf}_{D,0} - X^{dlf}_{UAV,0}}{t_{land}}$$
$$t_{land} = \frac{|Z^{I}_{UAV,0} - Z^{I}_{UAV,0}|}{V_{z,b}} \tag{3.9}$$

The resulting values were typically around $V_{x,b} = 0.15$ m/s, which does not have a significant impact on the analysis presented here.

Heading commands are handled separately without the use of an optimization solver, as the ship does not typically change course rapidly during landing. Ship heading is therefore characterized by oscillations about a constant or slowly varying mean value. As a result, simply commanding the aircraft heading to match the ship heading passed through a low pass filter was found sufficient.

For both the approach and landing phase the UAV heading was simply commanded to follow low pass filtered deck heading, as the ship does not typically change course rapidly during landing. Ship heading is therefore characterized by oscillations about a constant or slowly varying mean value.

3.2 Quadratic Programming Based Guidance with Deck Motion Predictions

A high-level view of the QP guidance algorithm is shown in Fig. 3.3. The algorithm solves quadratic programming problems in real time to produce X, Y, and Z position command inputs to the EMF control laws, with the trajectory planned to match a prediction of the deck state at touchdown. The deck forecasts are updated continuously during the approach, and therefore become more accurate as the landing progresses. This method can provide more direct landing paths that, unlike the baseline algorithm, do not need to follow the oscillatory motions of the deck. This also eliminates the need to break the trajectory into an approach and landing stage.

To solve the QP optimizations, three separate instances of the MATLAB® function "mpcActiveSetSolver" were executed: one each for inertial X, Y, and Z position commands, all updating at 10 Hz. The choice was made to use three separate instances of the QP optimizer because the EMF controller forces the system to approximate the outer loop command filters with the addition of time delay. Since the command filters are uncoupled, separating into smaller QP problems is valid. This boosts computational efficiency as the three QP solvers can be run simultaneously on different CPU cores.

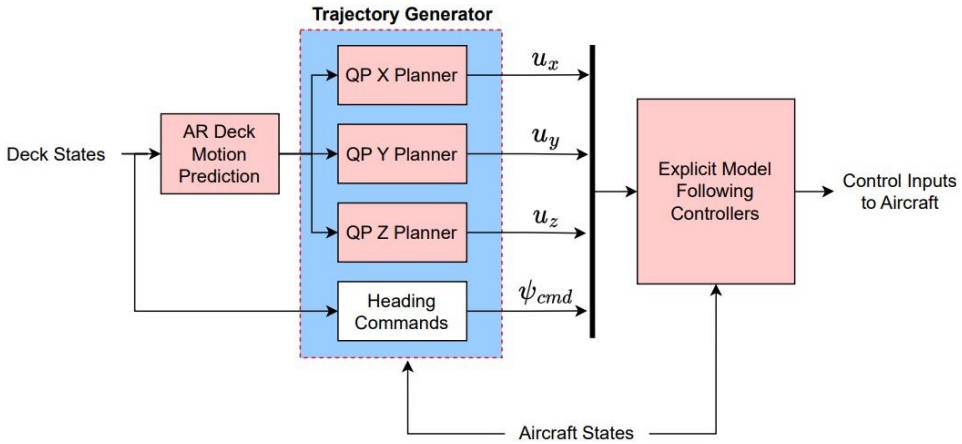


Figure 3.3: High level diagram of QP guidance algorithm.

Note that heading commands were not included in the optimization. Instead, it was found sufficient to command the UAV heading to follow a low pass filtered deck heading, as was done in the baseline guidance method. The remainder of this section will describe the formulation of the QP trajectory optimizations used to produce position commands.

3.2.1 Deck Motion Prediction

Autoregressive (AR) time series models were used to produce the deck motion predictions incorporated into the QP guidance algorithm. The AR method was chosen due to both its ease of implementation and the fact that past simulation studies have found this method to converge to reasonable predictions with sufficient lead time for performing autonomous landings (for example, see [5, 33]). Additionally, like all time-series based prediction methods, AR predictions are based on past measurements alone and therefore no information on the ship model is required.

3.2.1.1 AR Model Estimation and Propagation

The estimation of AR model parameters was carried out in a manner similar to that presented in [5], but written to allow for a recursive update. The AR model formulation assumes that the current output can be described by a linear combination of lagged outputs plus additive zero mean white noise. This is expressed as

$$\vec{y}_k = \alpha_1 \vec{y}_{k-1} + \alpha_2 \vec{y}_{k-2} \cdots + \alpha_{N_{lag}} \vec{y}_{k-N_{lag}} + \vec{v}_k \tag{3.10}$$

where output vectors $\vec{y} \in \mathbb{R}^m$, matrices $\alpha \in \mathbb{R}^{mxm}$, noise vector $\vec{v} \in \mathbb{R}^m$, and N_{lag} represents the number of lagged outputs. Here $N_{lag} = 15$ was used, as increases in accuracy leveled off when adding additional parameters. Transposing Eq. (3.10) and writing in matrix form gives

$$\vec{y}_k^\intercal = \begin{bmatrix} \vec{y}_{k-1}^\intercal & \vec{y}_{k-2}^\intercal & \cdots & \vec{y}_{k-N_{lag}}^\intercal \end{bmatrix} \begin{bmatrix} \alpha_1^\intercal \\ \alpha_2^\intercal \\ \vdots \\ \alpha_{N_{lag}}^\intercal \end{bmatrix} + \vec{v}_k^\intercal \tag{3.11}$$

or, in a more compact form,

$$\vec{y}_k^T = \bar{Y}_{lag}\bar{\alpha} + \vec{v}_k^\intercal \tag{3.12}$$

This is then separated into one equation for each output

$$\begin{aligned} y_1 &= \bar{Y}_{lag}\bar{\alpha}_1 + v_1 \\ y_2 &= \bar{Y}_{lag}\bar{\alpha}_2 + v_2 \\ &\vdots \\ y_m &= \bar{Y}_{lag}\bar{\alpha}_m + v_m \end{aligned} \tag{3.13}$$

where $\bar{\alpha}_i$ represents the ith column of $\bar{\alpha}$. Each column vector $\bar{\alpha}_i$ is then estimated using a standard recursive least squares (LS) algorithm. While the problem could be formulated as one larger recursive LS problem, breaking the parameter update into m smaller problems was found to be more computationally efficient due to smaller matrix inversions in the recursive update.

With the estimate of $\bar{\alpha}$ obtained, the AR model is propagated into the future to cover the desired prediction horizon:

$$\begin{aligned} \vec{y}_{k+1}^\intercal &= \begin{bmatrix} \vec{y}_k^\intercal & \vec{y}_{k-1}^\intercal & \cdots & \vec{y}_{k-N_{lag}+1}^\intercal \end{bmatrix} \bar{\alpha} \\ &\vdots \\ \vec{y}_{k+N}^\intercal &= \begin{bmatrix} \vec{y}_{k-1+N}^\intercal & \vec{y}_{k-2+N}^\intercal & \cdots & \vec{y}_{k-N_{lag}+N}^\intercal \end{bmatrix} \bar{\alpha} \end{aligned} \tag{3.14}$$

Note that for predictions we are interested in the expected future output, so the zero mean white noise term is dropped.

3.2.1.2 Use of AR Models for Deck State Predictions

For forecasting deck states, two separate AR models are used: one for longitudinal and vertical states and one for lateral states. The output vectors for each AR model are

$$\begin{aligned} \vec{y}_{long} &= \begin{bmatrix} X_D^{dlf} & \dot{X}_D^{dlf} & \theta_d & Z_D^I & \dot{Z}_D^I \end{bmatrix}^\intercal \\ \vec{y}_{lat} &= \begin{bmatrix} Y_D^{dlf} & \dot{Y}_D^{dlf} & \phi_D & \psi_D \end{bmatrix}^\intercal \end{aligned} \tag{3.15}$$

where the subscript D denotes these are deck states. Deck vertical position and velocity are included with the longitudinal outputs, as the correlation between ship pitch motion and landing deck heave was found to slightly improved the AR predictions. Including all outputs in a single vector, however, was found to cause less reliable predictions at times.

In Eq. (3.15) X_D, Y_D, $\dot{X}_D$, and $\dot{Y}_D$ are expressed in the averaged deck level frame. This frame is defined by rotating the inertial X and Y axes to align with the heading of the deck averaged over the lagged time history stored in the AR model. The transformation of X_D and Y_D from the inertial frame to the averaged deck level frame is computed as

$$\begin{bmatrix} X_D^{dlf} \\ Y_D^{dlf} \end{bmatrix} = \begin{bmatrix} cos(\bar{\psi}_D) & sin(\bar{\psi}_D) \\ -sin(\bar{\psi}_D) & cos(\bar{\psi}_D) \end{bmatrix} \begin{bmatrix} X_D^I \\ Y_D^I \end{bmatrix} \tag{3.16}$$

where $\bar{\psi}_D$ is the averaged deck heading. $\dot{X}_D$, and $\dot{Y}_D$ are transformed in the same manner. Once the AR model predictions are produced in this frame, they are transformed back to the inertial frame for use by the trajectory generator.

3.2.2 Discrete Time Model

A discrete system model is needed for each of the three QP solvers. For each axis the dynamics are approximated as the theoretical ideal result for the outer loop EMF controllers. The model is then discretized assuming a zero order hold. For example, for X^I trajectory generation, discretizing and converting to state space form gives

$$\begin{aligned} G_X(s) &= \frac{\omega_{X,cf}^2}{s^2 + 2\zeta\omega_{X,cf}s + \omega_{X,cf}^2} e^{-\tau_{\theta,\phi}s} \\ &\downarrow \\ \vec{x}_{X,k+1} &= A_X\vec{x}_{X,k} + B_X u_{X,k-\tau_d} \\ \vec{y}_{X,k} &= C_X\vec{x}_{X,k} + D_X u_{X,k-\tau_d} \end{aligned} \tag{3.17}$$

where subscript k represents the current time step and τ_d represents the number of full sample period delays. The fractional portion of the continuous time delay is absorbed into the discrete state space model. For example, if the continuous time model has a delay of 0.25 seconds and the sample period is 0.1 second (which is what was used here), then the discrete time model will delay the input by two sample periods (we set $\tau_d = 2$) and the remaining 0.05 seconds is absorbed into the state space matrices.

3.2.3 Quadratic Program Transcription

The standard form for a QP problem with inequality constraints is written as

$$\begin{aligned} J &= \frac{1}{2}\bar{U}^T H \bar{U} + F^T \bar{U} \\ s.t. \quad & A_c \bar{U} \leq b_0 \end{aligned} \tag{3.18}$$

where J represents the quadratic objective function, $\bar{U}$ represents the decision variables (in this case the controls), and H and F are constant matrices. In addition, the constant matrix A_c and constant vector b_0 define linear inequality constraints placed on the decision variables. Using our discrete time models, we can define a quadratic cost function and linear constraints in terms of system outputs, and then manipulate the equations to be a QP in terms of system inputs. The process to do this is essentially the same as the so-called "direct shooting" approach commonly used to apply QP to linear MPC. The main difference here is that the prediction horizon is variable in length, vanishing during the final stages of the landing sequence. The following subsections will cover this process.

3.2.3.1 Future Output and Jerk Calculation

We start by rolling out the output vectors over the prediction horizon. Consider a generic LTI system without delays (delays will be included in section 3.2.3.6):

$$\begin{aligned} \vec{x}_{k+1} &= A\vec{x}_k + Bu_k \\ \vec{y}_k &= C\vec{x}_k + Du_k \end{aligned} \tag{3.19}$$

The output at each time index can be written in terms of the initial state and the control inputs at each time step. Progressing through the time horizon, we have

$$\begin{aligned}
\vec{y}_0 &= C\vec{x}_0 + Du_0 \\
\vec{y}_1 &= C\vec{x}_1 + Du_1 = CA\vec{x}_0 + CBu_0 + Du_1 \\
\vec{y}_2 &= C\vec{x}_2 + Du_2 = CA^2\vec{x}_0 + CABu_0 + CBu_1 + Du_2 \\
&\vdots \\
\vec{y}_N &= CA^N\vec{x}_0 + CA^{N-1}Bu_0 + CA^{N-2}Bu_1 + \cdots + Du_N
\end{aligned} \tag{3.20}$$

where N represents the number of points in the prediction horizon. For convenience, Eq. (3.20) can be written in matrix form:

$$\bar{Y} = \hat{C}\vec{x}_0 + \hat{D}\bar{U} \tag{3.21}$$

where we use the notation

$$\begin{gathered}
\bar{Y} = \begin{bmatrix} \vec{y}_0 \\ \vec{y}_1 \\ \vdots \\ \vec{y}_N \end{bmatrix} \quad \bar{U} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{bmatrix} \quad \hat{C} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^N \end{bmatrix} \\
\hat{D} = \begin{bmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CAB & CB & D & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ CA^{N-1}B & CA^{N-2}B & CA^{N-3}B & \cdots & D \end{bmatrix}
\end{gathered} \tag{3.22}$$

Since the discrete state space models are determined by the second order command filters, the output vector $\vec{y}$ consists of position, velocity, and acceleration. To apply penalties and constraints to these outputs across the time horizon, the expressions shown in Eqs. (3.21) and (3.22) will be used. Constraining jerk is desirable as well, however, as jerk limitations are sometimes considered when evaluating full-scale ride quality. Additionally, an actual aircraft cannot achieve instantaneous acceleration, so it is logical to penalize and constrain instantaneous jumps in commanded acceleration.

In order to penalize and constrain jerk, the jerk is approximated by back differencing

the acceleration output. That is,

$$j_k = \frac{a_k - a_{k-1}}{\Delta t} \tag{3.23}$$

where Δt is the time step used when discretizing the model. In this work this back difference is what the term "jerk" refers to. With this definition and the expressions in Eqs. (3.21) and (3.22), the jerk can be solved for over the full time horizon as

$$\bar{Y}_\Delta = \hat{C}_\Delta \vec{x}_0 + \hat{D}_\Delta \bar{U} \tag{3.24}$$

where we use the notation

$$\begin{gathered} \bar{Y}_\Delta = \begin{bmatrix} j_1 \\ j_2 \\ \vdots \\ j_N \end{bmatrix} \quad \hat{C}_\Delta = \begin{bmatrix} C_j(A-I) \\ C_jA(A-I) \\ \vdots \\ C_jA^{N-1}(A-I) \end{bmatrix} \\ \hat{D}_\Delta = \begin{bmatrix} C_jB - D_j & D_j & 0 & \cdots & 0 \\ C_j(A-I)B & C_jB - D_j & D_j & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ C_jA^{N-1}(A-I)B & C_jA^{N-2}(A-I)B & \cdots & \cdots & D_j \end{bmatrix} \end{gathered} \tag{3.25}$$

Note that $\bar{U}$ in Eq. (3.25) is still defined as given in Eq. (3.22). Also note that C_j and D_j in Eq. (3.25) are calculated as

$$C_j = \frac{1}{\Delta t} C_3 \quad D_j = \frac{1}{\Delta t} D_3 \tag{3.26}$$

where the subscript 3 indicates that just the third row of the C and D matrices is used. This is because acceleration is the third output, so the jerk approximation only uses the state space coefficients defining the third output.

3.2.3.2 Cost Function

The cost function used here is defined as

$$\begin{aligned} J = &\sum_{k=0}^{N-1} \left[(\vec{y}_{ref,k} - \vec{y}_k)^\intercal Q (\vec{y}_{ref,k} - \vec{y}_k) + u_k^\intercal R u_k \right] \\ &+ \sum_{k=1}^{N-1} \left[j_k^\intercal Q_\Delta j_k \right] + u_N^\intercal R u_N + N \left[j_N^\intercal S_\Delta j_N \right. \\ &\left. + (\vec{y}_{ref,N} - \vec{y}_N)^\intercal S (\vec{y}_{ref,N} - \vec{y}_N) \right] \end{aligned} \tag{3.27}$$

where the weighting matrices Q, Q_Δ, S, S_Δ, and R are specified to be diagonal. This equation includes penalties on the deviation of position, velocity, and acceleration outputs from a reference trajectory $\vec{y}_{ref}$, as well as penalties on jerk and control inputs. For the output reference errors and jerk, the terminal cost weights S and S_Δ are high in magnitude relative to the running cost weights Q and Q_Δ. This places emphasis on the final point of the trajectory, which is particularly important for rendezvous problems like ship landing. Also note that the terminal cost associated with jerk and output reference error is weighted by the number of points in the prediction horizon N. This is done because the horizon length can vary at each time step, when the QP solver is re-run to update the commanded trajectory. When the horizon length is larger, the running cost is higher due to more points in the summation, so the terminal weighting factor is also increased to avoid its impact being diluted and to maintain consistent relative importance between the running and terminal cost. An approach similar to this was used in the variable-horizon ship landing guidance algorithm proposed by Pravitra in [5].

Using the expressions developed in the previous subsection, we can solve for the H and F matrices from Eq. (3.18) to write the cost function in the standard QP form. Expanding Eq. (3.27) and dropping constant terms not effecting the optimization gives

$$\begin{aligned} J = &\sum_{k=0}^{N-1} \left[\cancel{\vec{y}_{ref,k}^\intercal Q \vec{y}_{ref,k}} - 2\vec{y}_{ref,k}^\intercal Q \vec{y}_k + \vec{y}_k^\intercal Q \vec{y}_k + u_k^\intercal R u_k \right] \\ &+ \sum_{k=1}^{N-1} \left[j_k^\intercal Q_\Delta j_k \right] + u_N^\intercal R u_N + N \left[j_N^\intercal S_\Delta j_N \right. \\ &\left. + \cancel{\vec{y}_{ref,N}^\intercal S \vec{y}_{ref,N}} - 2\vec{y}_{ref,N}^\intercal S \vec{y}_N + \vec{y}_N^\intercal S \vec{y}_N \right] \end{aligned} \tag{3.28}$$

Using the expressions from Eqs. (3.22) and (3.25) that define the outputs, inputs, and jerk over the full prediction horizon, the cost function given above can be written in

matrix form:

$$J = -2\bar{Y}_{ref}^{\intercal}\bar{Q}\bar{Y} + \bar{Y}^{\intercal}\bar{Q}\bar{Y} + \bar{U}^{\intercal}\bar{R}\bar{U} + \bar{Y}_{\Delta}^{\intercal}\bar{Q}_{\Delta}\bar{Y}_{\Delta} \tag{3.29}$$

where we define

$$\begin{aligned} \bar{Y}_{ref} &= \begin{bmatrix} \vec{y}_{ref,0} \\ \vec{y}_{ref,1} \\ \vdots \\ \vec{y}_{ref,N} \end{bmatrix} \qquad \bar{R} = \begin{bmatrix} R & & & \\ & R & & \\ & & \ddots & \\ & & & R \end{bmatrix} \\ \bar{Q} &= \begin{bmatrix} Q & & & \\ & Q & & \\ & & \ddots & \\ & & & NS \end{bmatrix} \quad \bar{Q}_{\Delta} = \begin{bmatrix} Q_{\Delta} & & & \\ & Q_{\Delta} & & \\ & & \ddots & \\ & & & NS_{\Delta} \end{bmatrix} \end{aligned} \tag{3.30}$$

Substituting the expressions for $\bar{Y}$ and $\bar{Y}_{\Delta}$ given by Eqs. (3.21) and (3.24) into Eq. (3.29), we have:

$$\begin{aligned} J = &- 2\bar{Y}_{ref}^{\intercal}\bar{Q}\left(\hat{C}\vec{x}_0 + \hat{D}\bar{U}\right) + \left(\hat{C}\vec{x}_0 + \hat{D}\bar{U}\right)^{\intercal}\bar{Q}\left(\hat{C}\vec{x}_0 + \hat{D}\bar{U}\right) \\ &+ \bar{U}^{\intercal}\bar{R}\bar{U} + \left(\hat{C}_{\Delta}\vec{x}_0 + \hat{D}_{\Delta}\bar{U}\right)^{\intercal}\bar{Q}_{\Delta}\left(\hat{C}_{\Delta}\vec{x}_0 + \hat{D}_{\Delta}\bar{U}\right) \end{aligned} \tag{3.31}$$

Expanding and neglecting terms that do not contain the optimization variables $\bar{U}$, Eq. (3.31) reduces to

$$\begin{aligned} J = &\bar{U}^{\intercal}\left(\hat{D}^{\intercal}\bar{Q}\hat{D} + \hat{D}_{\Delta}^{\intercal}\bar{Q}_{\Delta}\hat{D}_{\Delta} + \bar{R}\right)\bar{U} \\ &+ 2\left(\vec{x}_0^{\intercal}\hat{C}^{\intercal}\bar{Q}\hat{D} + \vec{x}_0^{\intercal}\hat{C}_{\Delta}^{\intercal}\bar{Q}_{\Delta}\hat{D}_{\Delta} - \bar{Y}_{ref}^{\intercal}\bar{Q}\hat{D}\right)\bar{U} \end{aligned} \tag{3.32}$$

Dividing the above equation by two (which has no effect on the optimization) and comparing to the standard QP form in Eq. (3.18), the H and F matrices are found by inspection to be

$$\begin{aligned} H &= \hat{D}^{\intercal}\bar{Q}\hat{D} + \hat{D}_{\Delta}^{\intercal}\bar{Q}_{\Delta}\hat{D}_{\Delta} + \bar{R} \\ F &= \left(\hat{D}^{\intercal}\bar{Q}\hat{C} + \hat{D}_{\Delta}^{\intercal}\bar{Q}_{\Delta}\hat{C}_{\Delta}\right)\vec{x}_0 - \hat{D}^{\intercal}\bar{Q}\bar{Y}_{ref} \end{aligned} \tag{3.33}$$

These matrices are computed and passed to the QP solution routine at each time step.

3.2.3.3 Constraints

The expressions used to calculate the output and jerk over the prediction horizon (given in Eqs. (3.21) and (3.24), respectively) can also be used place constraints on the outputs and jerk. In the QP optimization formulation used here, the option is given to use

either hard or soft constraints. Hard constraints must be satisfied for the QP solution to be deemed feasible, while soft constraints allow some violation of the constraint boundaries but with a cost assigned proportional to the constraint violation. When using hard constraints, the current aircraft state estimate is not used as the initial state for re-planning the trajectory. Instead, the predicted output of the command model is used to initialize the next step in the trajectory generation solver. In other words, perfect command following is assumed and re-planning at each time step only compensates for changes in the predicted deck state at landing. This is done when using hard constraints because if feeding back estimated states for re-planning when operating near a constraint boundary, a slight perturbation in the aircraft state can place the initial state used in optimization over the constraint boundary resulting in an infeasible optimization problem. The use of soft constraints alleviates this concern by permitting some violation of the constraints, allowing the current state estimate to be fed back for re-planning. As will be seen later in this section, however, constraint softening also introduces additional optimization variables, making the QP solution more computationally expensive.

The hard constraint formulation is described as follows. Starting with the upper limits on position, velocity, and acceleration, we define an augmented vector containing the upper constraints at each time point in the prediction horizon:

$$\bar{Y}_{lim\,up} = \begin{bmatrix} \vec{y}_{lim\,up,0}^{\intercal} & \vec{y}_{lim\,up,1}^{\intercal} & \cdots & \vec{y}_{lim\,up,N}^{\intercal} \end{bmatrix}^{\intercal} \tag{3.34}$$

Appealing to Eq. (3.21), the inequality defining the output upper bounds is then written as

$$\hat{C}\vec{x}_0 + \hat{D}\bar{U} \leq \bar{Y}_{lim\,up} \rightarrow \hat{D}\bar{U} \leq \bar{Y}_{lim\,up} - \hat{C}\vec{x}_0 \tag{3.35}$$

Doing the same for the output lower limits, as well as the jerk upper and lower limits, yields

$$\begin{aligned} -\hat{D}\bar{U} &\leq -\bar{Y}_{lim\,low} + \hat{C}\vec{x}_0 \\ \hat{D}_\Delta\bar{U} &\leq \quad \bar{Y}_{\Delta,lim\,up} - \hat{C}_\Delta\vec{x}_0 \\ -\hat{D}_\Delta\bar{U} &\leq -\bar{Y}_{\Delta,lim\,low} + \hat{C}_\Delta\vec{x}_0 \end{aligned} \tag{3.36}$$

where $\bar{Y}_{\Delta,lim\,up}$ and $\bar{Y}_{\Delta,lim\,low}$ contain the jerk upper and lower limits across the prediction horizon. The last step in the hard constraint formulation is to combine the inequalities in Eqs. (3.35) and (3.36) and solve for the A_c matrix and b_0 vector defining the QP

constraints in Eq. (3.18), giving

$$A_c = \begin{bmatrix} \hat{D} \\ -\hat{D} \\ \hat{D}_\Delta \\ -\hat{D}_\Delta \end{bmatrix} \quad b_0 = \begin{bmatrix} \bar{Y}_{lim\,up} - \hat{C}\vec{x}_0 \\ -\bar{Y}_{lim\,low} + \hat{C}\vec{x}_0 \\ \bar{Y}_{\Delta,lim\,up} - \hat{C}_\Delta\vec{x}_0 \\ -\bar{Y}_{\Delta,lim\,low} + \hat{C}_\Delta\vec{x}_0 \end{bmatrix} \tag{3.37}$$

To implement soft constraints, the approach described above is modified by introducing slack variables. The process for this is as follows. First, the so-called slack variables are added to the upper and lower bounds of the constraint equations given in Eqs. (3.35) and (3.36), yielding

$$\begin{aligned} \hat{C}\vec{x}_0 + \hat{D}\bar{U} &\leq \bar{Y}_{lim\,up} + \vec{\epsilon} \\ -(\hat{C}\vec{x}_0 + \hat{D}\bar{U}) &\leq -(\bar{Y}_{lim\,low} - \vec{\epsilon}) \\ \hat{C}_\Delta\vec{x}_0 + \hat{D}_\Delta\bar{U} &\leq \bar{Y}_{\Delta,lim\,up} + \vec{\epsilon}_\Delta \\ -(\hat{C}_\Delta\vec{x}_0 + \hat{D}_\Delta\bar{U}) &\leq -(\bar{Y}_{\Delta,lim\,low} - \vec{\epsilon}_\Delta) \end{aligned} \tag{3.38}$$

where the slack variables $\vec{\epsilon}$ and $\vec{\epsilon_\Delta}$ represent the extent to which the output and jerk limits are violated, respectively. Next, the constraint equations are rearranged to place constant terms on the right hand side and $\bar{U}$ (the control vector over the time horizon) and slack variables on the left hand side:

$$\begin{aligned} \hat{D}\bar{U} - \vec{\epsilon} &\leq \bar{Y}_{lim\,up} - \hat{C}\vec{x}_0 \\ -\hat{D}\bar{U} - \vec{\epsilon} &\leq -\bar{Y}_{lim\,low} + \hat{C}\vec{x}_0 \\ \hat{D}_\Delta\bar{U} - \vec{\epsilon}_\Delta &\leq \bar{Y}_{\Delta,lim\,up} - \hat{C}_\Delta\vec{x}_0 \\ -\hat{D}_\Delta\bar{U} - \vec{\epsilon}_\Delta &\leq -\bar{Y}_{\Delta,lim\,low} + \hat{C}_\Delta\vec{x}_0 \end{aligned} \tag{3.39}$$

Considering the slack variables as an additional "input" to the QP solution routine, the constraint equations can be written as a linear inequality of the form

$$A_{c,soft}\bar{U}_{soft} \leq b_0 \tag{3.40}$$

where

$$A_{c,soft} = \begin{bmatrix} \hat{D} & -I & 0 \\ -\hat{D} & -I & 0 \\ \hat{D}_\Delta & 0 & -I_\Delta \\ -\hat{D}_\Delta & 0 & -I_\Delta \end{bmatrix} \quad \bar{U}_{soft} = \begin{bmatrix} \bar{U} \\ \vec{\epsilon} \\ \vec{\epsilon}_\Delta \end{bmatrix} \quad b_0 = \begin{bmatrix} \bar{Y}_{lim\,up} - \hat{C}\vec{x}_0 \\ -\bar{Y}_{lim\,low} + \hat{C}\vec{x}_0 \\ \bar{Y}_{\Delta,lim\,up} - \hat{C}_\Delta\vec{x}_0 \\ -\bar{Y}_{\Delta,lim\,low} + \hat{C}_\Delta\vec{x}_0 \end{bmatrix} \tag{3.41}$$

The QP cost function must then be re-written to penalize the use of slack variable "inputs" ϵ and ϵ_Δ. With the slack variables added, the quadratic cost function from Eq. (3.18) can be updated by augmenting the weighting matrices as follows:

$$H_{soft} = \begin{bmatrix} H & 0 \\ 0 & H_{slack} \end{bmatrix} \qquad F_{soft} = \begin{bmatrix} F \\ 0 \end{bmatrix} \tag{3.42}$$

where H and F in Eq. (3.42) are still calculated from Eq. (3.33), H_{slack} is a diagonal weighting matrix penalizing the square of the slack variables, and F_{soft} is augmented with a zero vector to account for the increased dimension of the augmented QP input vector $\bar{U}_{soft}$. The diagonal weights in H_{slack} can then be set to govern the "hardness" of the constraints. It should be noted, however, that the introduction of slack variables does increase the dimension of the optimization vector and therefore makes the QP solution more computationally expensive.

When using the hard constraint formulation, constraints are included on velocity, acceleration, and jerk. Additionally, altitude is given a lower bound defined by the predicted deck altitude at each point in the prediction horizon. This is included to prevent early contact with the deck. An altitude upper limit or limits on X^I and Y^I position were not included, however. When using the soft constraint formulation, constraints on altitude and acceleration were retained, but velocity and jerk constraints were removed. This was due to the additional computational cost associated with the introduction of slack variables, limiting the number of constraints that could be included while still running the QP algorithm onboard the aircraft at 10 Hz. Note that velocity and jerk were still incorporated into the QP cost function in both cases, however.

3.2.3.4 Prediction Horizon and Reference Path

The maximum horizon length was set to $N_{max} = 30$. With a time step of $dt_{QP} = 0.1$ seconds for each QP solver, this equates to a 3 second prediction horizon. Limiting the horizon to 3 seconds keeps the number of optimization variables and constraints to satisfy reasonable, but the prescribed time to perform the landing (which will be described in section 3.2.3.5) is generally higher than 3 seconds. To accommodate land times longer than the prediction horizon, the QP solver uses N_{max} as the horizon length if the time remaining in the landing sequence is greater than 3 seconds. If the time remaining is less than 3 seconds, then the QP horizon length N is defined as the time remaining divided

by the QP solver time step. Mathematically, that is

$$N = \min\left(\frac{t_{rem}}{dt_{QP}}\ ,\ N_{max}\right) \tag{3.43}$$

where t_{rem} is the time left until touchdown. The prediction horizon therefore vanishes as the landing is completed.

When $N > N_{max}$, an intermediate reference trajectory is prescribed for use in the cost function by drawing a straight, constant velocity line toward the deck. For example, for the X^I QP solver we have

$$v_{des} = \frac{X^I_{dp,f} - X^I_{UAV}}{t_{rem}} \tag{3.44}$$

where $X^I_{dp,f}$ is the deck X^I prediction at the desired land time. The output reference values for the X^I QP solver (when $N > N_{max}$) are then given as

$$\vec{y}_{Xref,k} = \begin{bmatrix} X^I_{UAV} + v_{des}t_k & v_{des} & 0 \end{bmatrix}^\intercal \tag{3.45}$$

where t_k is the time into the prediction horizon (i.e., $t_0 = 0\,s$ and $t_{N_{max}} = 3\,s$). An identical process is used for the Y^I and Z^I QP solvers.

When the prediction horizon covers the time remaining until touchdown ($N \leq N_{max}$), different reference outputs are prescribed. For the X^I and Y^I trajectories, an identical process is used to set the reference output $\vec{y}_{ref}$. Using the X^I axis to illustrate this, $\vec{y}_{ref}$ in the running cost is set to the predicted deck X^I position and velocity at touchdown and zero acceleration:

$$\vec{y}_{Xref,k} = \begin{bmatrix} X^I_{dp,N} & \dot{X}^I_{dp,N} & 0 \end{bmatrix}^\intercal \tag{3.46}$$

For the terminal cost, however, a non-zero reference acceleration is given:

$$\vec{y}_{Xref,N} = \begin{bmatrix} X^I_{dp,N} & \dot{X}^I_{dp,N} & a^I_{Xdes,N} \end{bmatrix}^\intercal \tag{3.47}$$

The terminal reference acceleration is used to motivate matching the pitch and roll attitude of the ship deck at touchdown. Attaining a close attitude match at touchdown is desired, as a large ship-relative attitude could result in issues with rotor tip clearance over the deck. Further, this can result in a single landing gear contacting the deck first, with the moment placed on the aircraft due to ground contact leading to higher impact velocities on the other landing gear. While helicopters are usually underactuated and

therefore cannot hold a set attitude without translating, it is theoretically possible to plan a trajectory such that a desired position, velocity, and attitude are obtained at one instant in time. The goal here is to motivate passing near this desired point in the state space at landing. Here the desired X^I and Y^I accelerations at touchdown are computed by assuming the vehicle level frame accelerations can be modeled by the equations

$$\begin{aligned} \ddot{X}^{lf} &= -g\theta \\ \ddot{Y}^{lf} &= g\phi \end{aligned} \tag{3.48}$$

and that the UAV will closely match deck heading at touchdown. The desired terminal X and Y accelerations are therefore calculated from the predicted deck attitude at touchdown as

$$\begin{bmatrix} a^I_{Xdes,N} \\ a^I_{Ydes,N} \end{bmatrix} = \begin{bmatrix} cos(\psi_{dp,N}) & -sin(\psi_{dp,N}) \\ sin(\psi_{dp,N}) & cos(\psi_{dp,N}) \end{bmatrix} \begin{bmatrix} -g\theta_{dp,N} \\ g\phi_{dp,N} \end{bmatrix} \tag{3.49}$$

While for a full-scale aircraft with significant wind the approximation in Eq. (3.48) may not be valid, any mapping (linear or nonlinear) from aircraft states to acceleration could be used here along with predicted deck and aircraft states. Given a model of a full-scale aircraft, this mapping could potentially be found.

For the Z^I trajectory, when $N \leq N_{max}$, the portion of $\vec{y}_{ref}$ used in the running cost was still set to give a straight line to the desired landing point. This was done in an identical manner to what is shown for the X axis in Eq. (3.45). For the Z^I solver terminal cost, the predicted deck position and velocity at land time were used to define $\vec{y}_{ref}$, but the desired land position was placed 3 cm above the deck:

$$\vec{y}_{Zref,N} = \begin{bmatrix} Z^I_{dp,N} + Z^I_{offset,N} & \dot{Z}^I_{dp,N} & 0 \end{bmatrix}^\intercal \tag{3.50}$$

The 3 cm offset $Z^I_{offset,N}$ gives a slight gap between the constraints and the desired end state in the QP solver, and it is close enough to begin throttling down when landing on an actual platform. Note that while there is discontinuity when moving from the portion of $\vec{y}_{ref}$ used in the running cost to the final reference point $\vec{y}_{ref,N}$ used in the terminal cost, this is smoothed by the dynamics enforced in the QP solver.

3.2.3.5 Land Time

Since the prediction horizon itself is fixed in each QP optimization, the duration of the landing maneuver must be assigned prior to running the QP solver. Methods employed by

other researchers to assign the landing duration include manually choosing a fixed length of time (for example, [12]) or performing a line search on the allotted time for landing (for example, [5]). Manually pre-determining the trajectory duration requires this value to be tuned when starting the trajectory at different distances from the deck. Performing a line search requires optimizing multiple trajectories, each targeting a different land time and therefore different predicted deck states at landing. This is more flexible to different initial conditions at the start of the trajectory and, if deck predictions are accurate enough, can be used to choose a land time that has a more favorable deck state at touchdown. The line search also adds additional computational burden, however, though in [5] Pravitra demonstrated that this can be done in parallel on a graphics processing unit at the expense of additional implementation effort. Here it was found both simple and sufficient to choose an initially targeted land time based on the tau guidance method described in [26], where the landing duration was chosen to limit accelerations during the maneuver. Using X^I guidance as an example, the resulting equation is

$$t_{land,X} = 2.888\sqrt{\frac{|X^I_{UAV,0} - X^I_{D,0}|}{\ddot{X}^I_{max}}} \tag{3.51}$$

where the subscript 0 indicates the start of the landing sequence and $\ddot{X}^I_{max}$ is the maximum acceleration in the tau guidance command profile. While the tau guidance method is not used in the QP guidance algorithm, this equation was found to produce reasonable times for the landing maneuver and to scale well across different initial distances from the deck. We therefore used this method to set the initial maneuver length t_{land}, but with a scale factor of 0.5 added to make the maneuver less aggressive. Mathematically, that is

$$t_{land} = \max \begin{pmatrix} 5.776\sqrt{\frac{|X^I_{UAV,0} - X^I_{D,0}|}{\ddot{X}^I_{max}}} \\ 5.776\sqrt{\frac{|Y^I_{UAV,0} - Y^I_{D,0}|}{\ddot{Y}^I_{max}}} \\ 5.776\sqrt{\frac{|Z^I_{UAV,0} - Z^I_{D,0}|}{\ddot{Z}^I_{max}}} \end{pmatrix} \tag{3.52}$$

where the maximum value across all three axes is used and the maximum accelerations are set equal to the QP planner acceleration limit.

If a flag is set in the trajectory generation algorithm, the initially chosen land time can be updated during the approach based on the predicted deck states. The process for doing this is depicted in Fig. 3.4 and is described as follows:

1. First an initial land time is chosen using Eq. (3.52).

2. Neighboring time points are then assigned a cost based on the predicted deck state.

3. The point with the lowest cost is then chosen as the new targeted time for landing.

The update is allowed to occur when in a pre-specified window, which was defined here to be when there is between 1.5 and 3 seconds left in the maneuver. The lower bound was chosen to avoid shifting the land point without leaving enough time for the UAV to react, and the specific value of 1.5 seconds was determined using model-scale simulations. The 3 second upper limit was chosen because deck predictions are known to be inaccurate at longer prediction horizons. When in this window, the algorithm can shorten the remainder of the maneuver by up to 3 time steps (0.3 seconds in this case) or extend the land time out to 3 seconds. Each time point in the considered landing window is assigned a cost, and the point with the lowest cost is selected as the new land time. Note that when applying this method to a full-scale landing scenario both the 1.5 second and 3 second limits should be scaled to account for slower developing aircraft dynamics, as well as the fact that valid deck state predictions can be made further into the future at full-scale due to lower frequency deck motion.

The cost for each time point in the considered landing window is given by

$$\begin{aligned} J_k = w_Z \left(Z^I_{dp,k} - \bar{Z}^I_{dp} \right) - w_{\dot{Z}} \dot{Z}^I_{dp,k} + w_\phi |\phi_{dp,k}| \\ + w_\theta |\theta_{dp,k}| + w_{\Delta t} |t_{land,k} - t_{land}| \end{aligned} \tag{3.53}$$

This equation weights the deck heave deviation from the mean, the heave velocity, and the deck roll and pitch angles in the first four terms. Note that Z and $\dot{Z}$ in the first two terms of Eq. (3.53) are positive down. These terms are designed to attempt targeting a point where the deck heave is near a high point, but is beginning to move back down. This was done to avoid attempting to mate with a deck that is heaving upward significantly, or near the bottom of an oscillation and about to heave upward, which is a harder state to match at landing. The third and fourth term of the cost function were included to discourage landing at a time with an extreme deck attitude, and the last term was included to penalize changes from the currently targeted land time, so that large changes in desired land time are not made due to small changes in the other terms. The weighting factors were set to $w_Z = 1$, $w_{\dot{Z}} = 0.5$, $w_{\Delta t} = 0.15$, and $w_\phi = w_\theta = 0.06 \cdot 180/pi$.

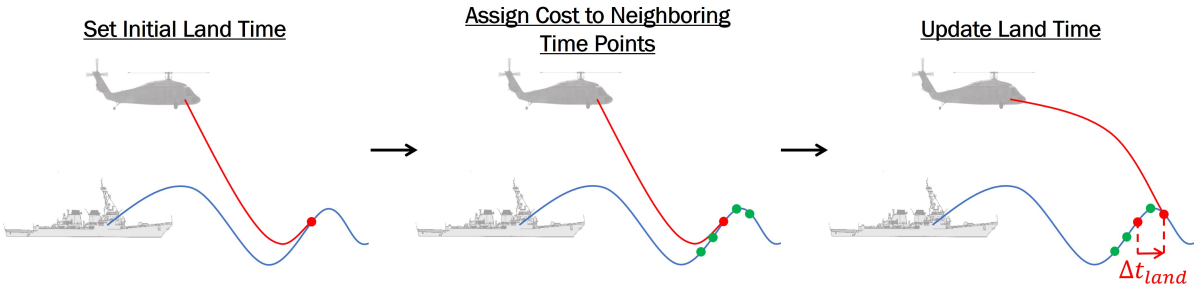


Figure 3.4: Schematic of the process for updating the targeted land time.

3.2.3.6 Inclusion of Discrete Time Delays

The preceding subsections considered a discrete LTI system without any input delays, but the discrete time models used here have full sample period delays included. Adding these delays into the QP framework is simple, however. If we have a discrete delay of τ_d sample periods, we simply store the previous τ_d inputs that were passed from the QP solver to the EMF controller. The stored inputs from the previous τ_d iterations then predetermine the first $\tau_d - 1$ future outputs $\vec{y}_0 \cdots \vec{y}_{\tau_d - 1}$ of the discrete time model, meaning that these outputs are no longer free and should be dropped from the cost function. To drop these no longer free terms, the stored τ_d past inputs are used to propagate the model forward, filling the first $\tau_d - 1$ outputs into the prediction horizon:

$$\begin{aligned} \vec{y}_0 &= C\vec{x}_0 + Du_{-\tau_d} \ , \ \vec{x}_1 = A\vec{x}_0 + Bu_{-\tau_d} \\ \vec{y}_1 &= C\vec{x}_1 + Du_{1-\tau_d} \ , \ \vec{x}_2 = A\vec{x}_1 + Bu_{1-\tau_d} \\ &\vdots \\ \vec{y}_{\tau_d - 1} &= C\vec{x}_{\tau_d - 1} + Du_{-1} \ , \ \vec{x}_{\tau_d} = A\vec{x}_{\tau_d - 1} + Bu_{-1} \end{aligned} \tag{3.54}$$

The resulting state $\vec{x}_{\tau_d}$ is then taken as $\vec{x}_0$ in the equations used to formulate the QP optimization, and the "prediction horizon" used when calculating H, F, A_c, and b_0 from Eq. (3.18) is set to $N - \tau_d$. The QP solver then computes the optimal inputs $u_0 \cdots u_{N-\tau_d}$ that define the outputs $\vec{y}_{\tau_d} \cdots \vec{y}_N$.

3.3 Wave Off Criterion

A "wave off" criterion for aborting landing would be a critical component to a complete autonomous landing solution. Here the development and evaluation of a wave off criterion was considered outside the scope of this research, but the importance of a well validated

method for aborting landing should be noted.

While the development of a wave off criteria targeted toward use on full-sized rotorcraft was considered out of scope here, a simple criteria for aborting landing was implemented as a safety mechanism for use in the model-scale tests discussed in Chapter 5. To initiate an abort, the position and velocity of the UAV relative to the deck were compared to predefined thresholds shortly before landing. The comparison was triggered once the UAV landing gear was 15 centimeters above the plane of the deck (corresponding to about 2 meters at full-scale when applying the scaling factor determined in Chapter 4). The height of the UAV landing gear above the plane of the deck can be calculated by examining the geometry sketched in Fig. 3.5. Referring to Fig. 3.5, $\vec{r}_p^{\,d}$ represents the position vector from the deck origin (the desired landing point) to the point on the deck plane that is directly below the UAV (denoted by point P). Additionally, ξ is defined as the altitude of point P above the deck origin and $h_{z\rightarrow UAV}$ is the altitude of the aircraft landing gear relative to point P. Noting that $\vec{r}_p^{\,d}$ is a 2 dimensional vector in the deck frame $X-Y$ plane, the following system of equations can be written:

$$\begin{bmatrix} X^I_{UAV} - X^I_D \\ Y^I_{UAV} - Y^I_D \\ \xi \end{bmatrix} = T_{I/D} \begin{bmatrix} r^d_{p,x} \\ r^d_{p,y} \\ 0 \end{bmatrix} \tag{3.55}$$

where $T_{I/D}$ represents the rotation matrix defining the transformation from the deck body frame to the inertial frame. Using the first two equations of the matrix relation, the distances $r^d_{p,x}$ and $r^d_{p,y}$ can be calculated. The values of $r^d_{p,x}$ and $r^d_{p,y}$ can then be used to solve for ξ. With ξ determined, the altitude of the aircraft landing gear relative to point P is calculated as

$$h_{z\rightarrow UAV} = \xi + Z^I_D - Z^I_{UAV} \tag{3.56}$$

where the deck and aircraft inertial Z positions use the positive down sign convention.

The X and Y position error thresholds used to trigger an abort were defined through the relation

$$T_{ldf/I} \begin{bmatrix} |X^I_{UAV} - X^I_D| \\ |Y^I_{UAV} - Y^I_D| \end{bmatrix} < \begin{bmatrix} e_{x,max} \\ e_{y,max} \end{bmatrix} \tag{3.57}$$

where $T_{ldf/I}$ is calculated as

$$T_{ldf/I} = \begin{bmatrix} cos(\psi_D) & sin(\psi_D) \\ -sin(\psi_D) & cos(\psi_D) \end{bmatrix} \tag{3.58}$$

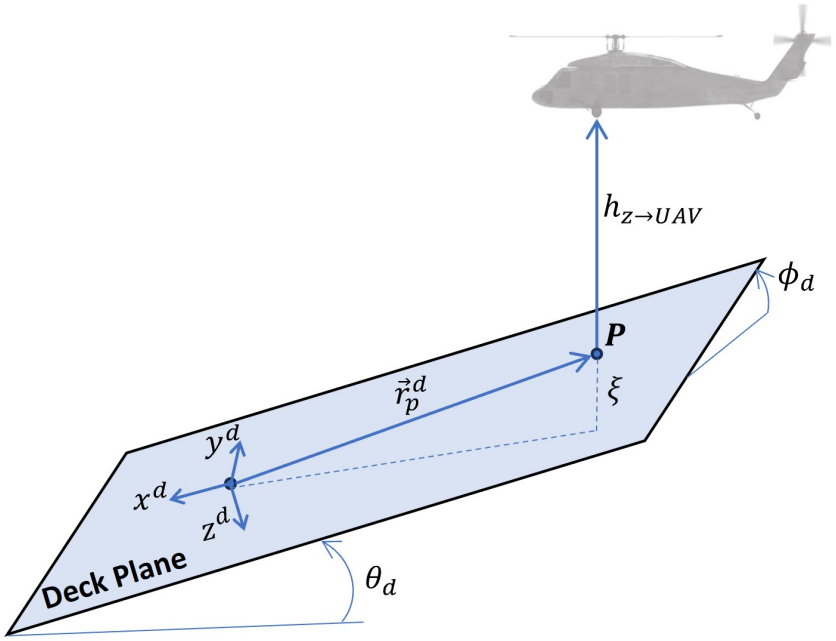


Figure 3.5: Schematic depicting UAV landing gear altitude relative to the deck plane.

The transformation is included to convert the deck-relative inertial positions to the level deck frame, aligning X position errors with the ship longitudinal axis and Y position errors with the ship lateral axis. The thresholds $e_{x,max}$ and $e_{y,max}$ were set to 0.5 meters and 0.38 meters, respectively. These values were determined based on the dimensions of the model-scale ship to ensure that the UAV was safely away from the edge of the deck before landing. The deck-relative X, Y, and Z velocity thresholds used to determine an abort were all set to 1 m/s, with the velocity comparison done in the inertial frame. This was found to be a high velocity limit that would still allow hard landings, but would avoid damage to the UAV hardware. This was deemed sufficient here, as the goal was not to test the abort procedure but to use it to avoid hardware damage that would effect further testing.

If any of the position or velocity thresholds were exceeded, an abort was initiated. The abort procedure was to command the UAV to hold the current X and Y position, while also commanding a step input into the altitude control law command filter such that the UAV would hover one meter above the model-scale deck.

Chapter 4
Scaling Methodology and Scaled Control Laws

In order to appropriately perform model-scale autonomous landing experiments, a systematic scaling procedure must be used to consistently scale aircraft closed-loop dynamics and ship motions. This chapter will first propose the use of Froude scaling to achieve this. The feasibility of using the EMF control law presented in Chapter 2 to attain a Froude scaled response to reference commands and disturbances will then be analyzed, and the methods used to select the EMF controller command model and feedback gain parameters are discussed.

4.1 Scaling Methodology

The model-scale aircraft controller parameters are determined by Froude scaling closed-loop reference tracking and disturbance rejection bandwidths from a full-scale use case (refer to Chapter 1.2.3 for additional discussion on Froude scaling). With the Froude scaling method, model-scale distance is calculated by dividing full-scale distance by the Froude scale factor N_F. Model-scale time is calculated by dividing full-scale time by $\sqrt{N_F}$, and frequency follows the inverse of this. Following these rules the dynamic scaling laws shown previously in table 1.1 can be derived, which are reproduced in table 4.1 for convenience (note that we multiply by these scale factors to go from full- to model-scale). To illustrate the process for deriving these laws, we consider scaling jerk. Looking at the units of jerk, we have

$$\frac{m_{ms}}{s_{ms}^3} = \frac{m_{fs}/N_F}{(s_{fs}/\sqrt{N_F})^3} = \frac{m_{fs}N_F^{3/2}N_F^{-1}}{s_{fs}^3} = \frac{m_{fs}\sqrt{N_F}}{s_{fs}^3} \tag{4.1}$$

where m denotes meters, s denotes seconds, and subscripts fs and ms denote full- and model-scale, respectively. This shows model-scale jerk is increased from full-scale by a factor of $\sqrt{N_F}$.

Table 4.1: Froude Scaling Factors

Dimension	**Scale Factor**
Time	$1/\sqrt{N_F}$
Frequency	$\sqrt{N_F}$
Position	$1/N_F$
Velocity	$1/\sqrt{N_F}$
Acceleration	1
Jerk	$\sqrt{N_F}$
Angles	1
Angular Rates	$\sqrt{N_F}$
Weight	$1/N_F^3$
Inertia	$1/N_F^5$

To use the scaling factors given in table 4.1, the Froude scale factor N_F must be determined. Here we propose using vehicle mass ratio to establish the scale factor. That is, the Froude scale factor is taken as

$$N_F = \left(\frac{M_{fs}}{M_{ms}}\right)^{1/3} \tag{4.2}$$

where M_{fs} and M_{ms} represent full and model-scale vehicle masses, respectively. In Chapter 4.2 and Chapter 4.3 it will be shown that the actual choice of scaling factor (for example, using mass ratio, length ratio, etc.) does not matter as long as the aircraft dynamics, control laws, guidance algorithms, and ship motion are all scaled consistently. Still, it is prudent to select a physically meaningful quantity, and mass has clear importance here as ship motions are typically significant in heave. Relative thrust-to-weight ratio is therefore important when maneuvering relative to the ship deck, and Froude scaling based on mass ratio retains approximate similarity in this regard.

In the tests discussed here, the full-scale aircraft considered was a medium weight helicopter roughly the size of a UH-60. Both the hexacopter and quadcopter UAVs used for model-scale experimentation weigh approximately 6.6 pounds, so taking the nominal full-scale aircraft weight to be approximately 17,500 pounds the Froude scale factor is

calculated as

$$N_F = \left(\frac{17500}{6.6}\right)^{1/3} \approx 13.8 \tag{4.3}$$

For the model scale tests that will be discussed in Chapter 5, all scaling was based on this value of $N_F = 13.8$.

4.2 Dynamic Similarity for Scaled Reference Tracking

It can be shown that the Froude scaling method proposed in the previous section retains dynamic similarity across test scales when analyzing the closed-loop reference tracking dynamics of the aircraft in the absence of disturbances. To show this, it is first noted that rotorcraft typically employ model-following control architectures that attempt to make the aircraft attitude and heave dynamics approximate a transfer function (the command model). This same approach was taken here using the EMF control architecture presented in Chapter 2. Assuming the attitude and altitude reference tracking dynamics approximate the command models, similarity parameters that must remain constant to preserve dynamic similarity are easily derived through the use of Buckingham's Pi theorem [62]. For example, consider the altitude dynamics of the UAV following the second order command model shown for the altitude controller in Chapter 2.6.2:

$$\ddot{Z}(t) + 2\zeta\omega\dot{Z}(t) + \omega^2 Z(t) = \omega^2 u_Z(t) \tag{4.4}$$

where u_Z is the input to the command filter, Z is the aircraft altitude, and ω and ζ are the command filter frequency and damping ratio. Buckingham's Pi theorem is then applied to Eq. (4.4) as follows:

1. **Identify the number of variables of the system**. Here there are five variables: u_Z $[m]$, Z $[m]$, ω $[s^{-1}]$, ζ, and t $[s]$, where the terms in square brackets represent the dimension of each variable with m representing meters and s representing seconds. Note ζ is dimensionless.

2. **Identify the total number of dimensions.** Here there are 2 dimensions: length (m) and time (s).

3. **Determine the number of dimensionless Π groups.** According to Buckingham's Pi theorem, the number of independent dimensionless Π groups (denoted here by k) is calculated as

$$k = n - j \tag{4.5}$$

where n is the number of system variables (item 1 of this list) and j is the number of independent dimensions (item 2 of this list). For this example, this gives $k = 5 - 2 = 3$ dimensionless Π groups.

4. **Determine a set of Π groups.** Structured methods for determining a non-unique set of Π groups are commonly used (for example, see [63]), but here a simple set of Π groups that can be derived from inspection is

$$\Pi_1 = \frac{Z}{u_Z} \quad , \quad \Pi_2 = \omega t \quad , \quad \Pi_3 = \zeta \tag{4.6}$$

With the dimensionless groups determined, we can show that these groups remain constant across test scales when the Froude scaling rules given in the previous section are applied. First considering Π_1 with the altitude output and altitude command both Froude scaled from model-scale to full-scale we have

$$\Pi_1 = \frac{Z_{ms}}{u_{Z,ms}} = \frac{Z_{fs}/N}{u_{Z,fs}/N} = \frac{Z_{fs}}{u_{Z,fs}} \quad \Rightarrow \quad \Pi_1 \text{ maintained} \tag{4.7}$$

which shows that Π_1 remains constant across scales. Considering Π_2 with both time and frequency Froude scaled we have

$$\Pi_2 = \omega_{ms} t_{ms} = \left(\omega_{fs}\sqrt{N_F}\right)\left(\frac{t_{fs}}{\sqrt{N_F}}\right) = \omega_{fs} t_{fs} \quad \Rightarrow \quad \Pi_2 \text{ maintained} \tag{4.8}$$

showing that Π_2 remains constant. Last, Π_3 is just the command filter damping ratio, which is a user-determined parameter that can be fixed to a constant value regardless of scale.

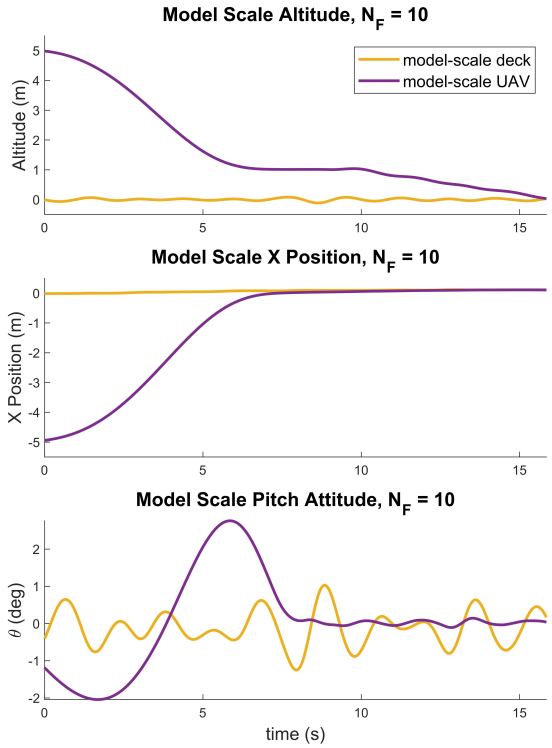
The preceding analysis shows that if the closed-loop altitude tracking dynamics approximate the command model, then dynamic similarity is retained regardless of the choice of Froude scaling factor. The same result is obtained when applying a similar process to the attitude and X and Y position dynamics. This result is expected, however, as the Froude scaling laws given in table 4.1 were derived to be dimensionally consistent (which was demonstrated for jerk scaling in Eq. (4.1)). Therefore, regardless of the system considered, any dimensionless Π group will remain unchanged if all variables are Froude scaled. For model-scale experiments the question then is, can the variables of each Π group really be assigned to match values determined by Froude scaling a target full-scale use case? With the reference tracking dynamics forced to approximate the command models, the answer is yes. To see this note that the output of each command

model is completely determined by the command model frequency, damping ratio, and input (i.e., u_Z for altitude). Therefore, if these three variables all satisfy the Froude scaling assumptions so will the output. For the frequency and damping ratio parameters the Froude scaling assumption is easily satisfied, as these are user defined parameters that can be set to a desired value in the control software. The input to the command models, on the other hand, is determined by the choice of autonomous landing guidance algorithm, and the commands produced by the guidance algorithm are inherently related to the ship motion. It can be shown though, that if the ship motion is also Froude scaled, then either of the guidance algorithms presented in Chapter 3 will produce appropriately scaled command inputs and dynamic similarity for the reference tracking dynamics is therefore retained.

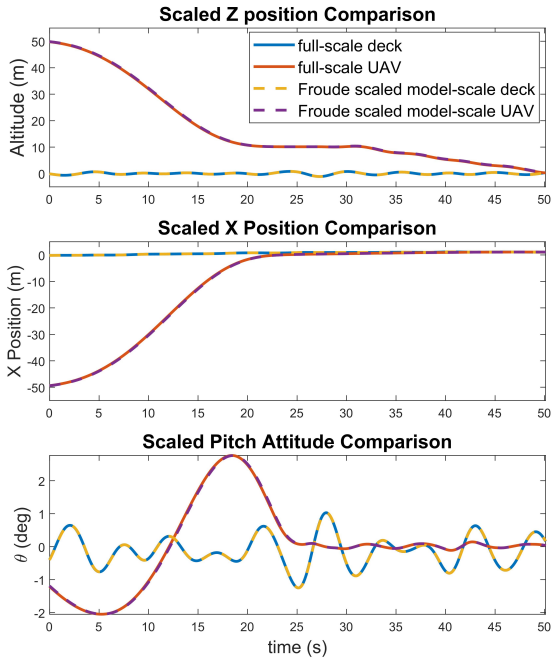
This is first demonstrated for the baseline guidance algorithm with a simplified simulation. For this simulation it was assumed that the aircraft pitch, roll, yaw, and altitude dynamics follow the EMF controller command models perfectly. Additionally, the inertial X and Y translational motions were modelled by the equation

$$\begin{bmatrix} \ddot{X}^I \\ \ddot{Y}^I \end{bmatrix} = \begin{bmatrix} cos(\psi) & -sin(\psi) \\ sin(\psi) & cos(\psi) \end{bmatrix} \begin{bmatrix} -g\theta \\ g\phi \end{bmatrix} \tag{4.9}$$

While this is a significantly simplified system model, this serves the purpose of demonstrating the scaling of the reference tracking dynamics. The ship data used for simulation was derived from simulation data of a generic surface combatant representative of a DDG-51 type ship. This database was developed by the U.S. Navy Office of Naval Research and the Naval Surface Warfare Center under the the Systematic Characterization of the Naval Environment (SCONE) program [47]. To demonstrate that dynamic similarity is retained for the reference tracking dynamics when the EMF control law is commanded by the baseline guidance algorithm, two cases were simulated: a "full-scale" case and a "model-scale" case. The full-scale case used roll, pitch, and yaw command filter frequencies of 3 rad/s and an altitude command filter frequency of 1 rad/s, with the SCONE ship data unscaled. The model-scale case used a scaling factor of $N_F = 10$ to Froude scale the command filter frequencies as well as the ship data. The results are shown in Fig. 4.1, with Fig. 4.1a showing the altitude, inertial X position, and pitch attitude of the aircraft obtained from the simulated model-scale landing. The data in Fig. 4.1a was then scaled up to full-scale and compared to landing data obtained directly from the full-scale simulation case in Fig. 4.1b. The results show agreement between the Froude scaled and simulated data sets, demonstrating dynamic similarity for the



(a) Model-scale simulation at $1/10^{\text{th}}$ scale.



(b) Scaled-up model-scale simulation compared to full-scale simulation.

Figure 4.1: Landing trajectories at full and $1/10^{\text{th}}$ scale obtained from simplified simulations with the baseline guidance algorithm.

reference tracking dynamics is retained with the baseline guidance algorithm.

To show a similar result for the QP guidance algorithm, scaling of the AR model deck state predictions (discussed in Chapter 3.2.1) must be considered since the predicted deck states affect the position commands produced from the QP optimization. Similar to what was demonstrated when applying Buckingham Pi theorem to the Froude scaled altitude command model, if all parameters of the AR models (including the AR model time step) are Froude scaled then the predictions that are produced will also satisfy the Froude scaling rules. Recall, however, that the AR models take the form

$$\vec{y}_k = \alpha_1 \vec{y}_{k-1} + \alpha_2 \vec{y}_{k-2} \cdots + \alpha_{N_{lag}} \vec{y}_{k-N_{lag}} + \vec{v}_k \tag{4.10}$$

where $\vec{y}_k$ is the current output, $\vec{v}_k$ zero mean white noise, and the α parameter matrices are determined via recursive least squares (RLS) estimation. Since the α parameter matrices are determined via RLS, the measurement noise covariance terms used in the RLS algorithm also need to be Froude scaled to maintain similarity. This is confirmed

by the simulated deck altitude predictions shown in Fig. 4.2, which were produced using the SCONE ship data. The plot on the lower left of Fig. 4.2 shows 4 second ahead deck altitude predictions compared to the true deck altitude for a "full-scale" case where predictions were produced using unscaled SCONE ship data. The plot on the upper left then shows altitude predictions for a "model-scale" case where the ship data, AR model parameters, AR model time step, and RLS noise variance terms were Froude scaled with the factor $N_F = 16$ ($1/16^{\text{th}}$ scale). Also, note that the model-scale predictions are compared to ship data 1 second ahead as opposed to 4 seconds ahead at full-scale, as time durations are also Froude scaled. Comparing these two plots, it is observed that they look visually identical, except that the length and time scales shown on the X and Y axes differ by their respective Froude scaling factors. This indicates that the predictions maintain similarity when all parameters are Froude scaled. This is further demonstrated by the plot on the right of Fig. 4.2, which shows the mean and 2σ bounds of the deck altitude prediction errors plotted against the length of time into the future we are predicting, with these statistics calculated from prediction errors produced from 280 simulations. This shows that when the model-scale prediction error statistics are scaled-up to full-scale and compared to prediction error statistics obtained directly from full-scale data, the mean and 2σ bounds tightly match. This again indicates similarity is retained.

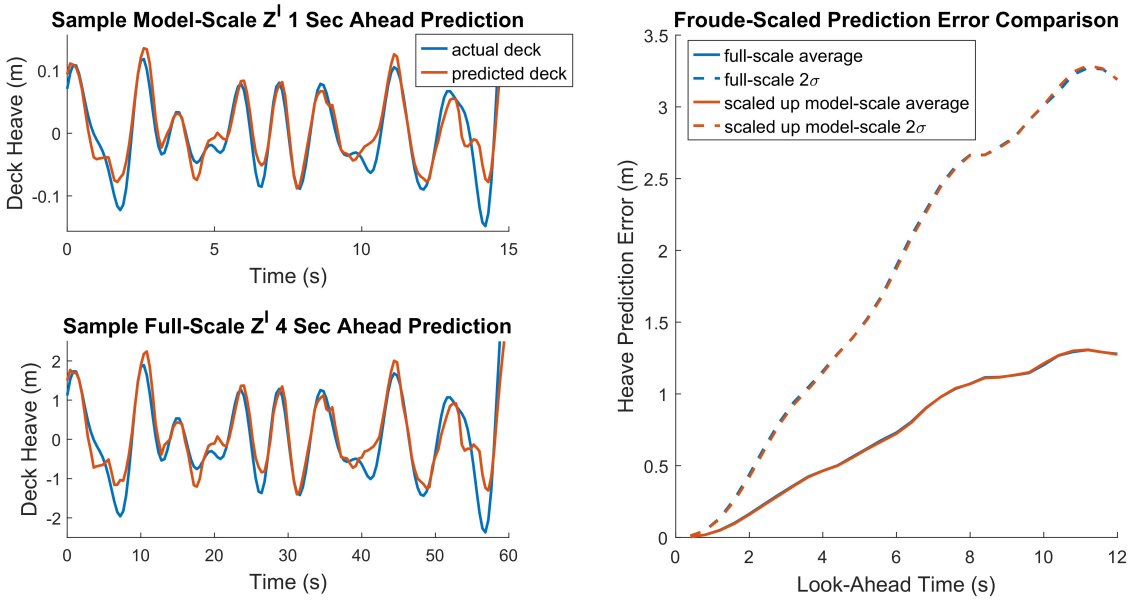
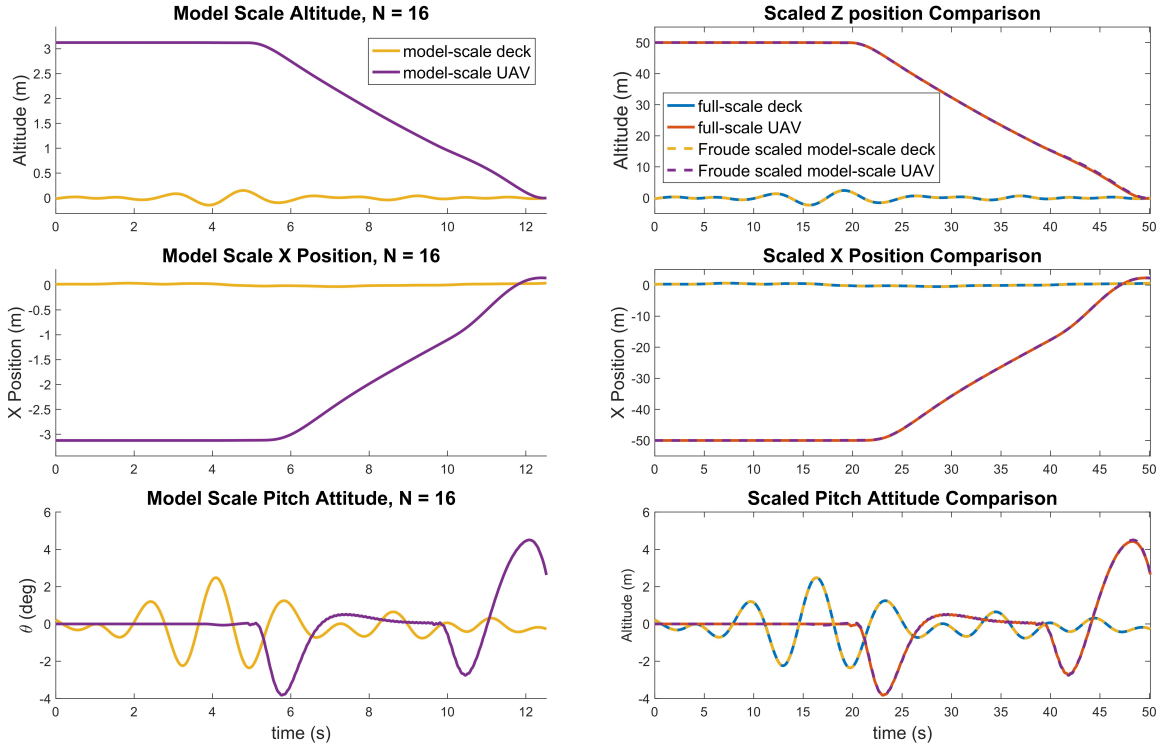


Figure 4.2: Comparison of deck heave predictions at full-scale ($N_F = 1$) and model-scale ($N_F = 16$) with Froude scaled AR model.

To confirm that dynamic similarity is retained for trajectories produced with the QP guidance algorithm, simulations were run using the SCONE data and simplified UAV model mentioned previously in this section. For these simulations the command model parameters, ship model, and AR prediction models were scaled as discussed in the preceding paragraphs. Additionally, the QP algorithm update rate, optimization horizon length, output constraints, and cost function weights were Froude scaled as well. The cost function weights need to be scaled, as they have units equivalent to the squared-inverse of the penalized term in order to give a unitless cost function. For example, the penalty on squared velocity error will have units of s^2/m^2, and must therefore be scaled accordingly. The results are shown in Fig. 4.3 in the same format as discussed previously for simulations using the baseline guidance method. When the model-scale simulation data from Fig. 4.3a is scaled up and compared to data obtained directly from the full-scale simulation (shown in Fig. 4.3b), we again see tight agreement between the two data sets.



(a) Model-scale simulation at $1/16^{\text{th}}$ scale.

(b) Scaled-up model-scale simulation compared to full-scale simulation.

Figure 4.3: Landing trajectories at full and $1/16^{\text{th}}$ scale obtained from simplified simulations with the QP guidance algorithm.

In summary, this section showed that if the aircraft dynamics approximate the command model then dynamic similarity is retained for the reference tracking dynamics regardless of choice of Froude scaling factor N_F, provided that the ship data, control law command models, deck state prediction algorithm, and guidance algorithm parameters are scaled consistently. This gives merit to scaling the reference tracking dynamics with the Froude scaling laws from table 4.1, but does not address the decision to use mass ratio to establish the scale factor. As mentioned in the previous section, the choice to scale based on mass ratio was made because mass is a physically relevant quantity and this scaling retains approximate similarity in relative thrust-to-weight ratio.

Lastly, while the tests that will be discussed in Chapter 5 were performed predominantly in a wave basin, it is worth noting that the results shown in this section offer an improvement to other more accessible test setups, such as those utilizing small UAVs and a motion platform to emulate the ship deck. As discussed in Chapter 1.2.4, many tests have used this setup and the majority of them do not include aerodynamic disturbances. The tests are therefore more focused on demonstrating that the proposed algorithm can compensate for deck motions in high sea states, and the closed-loop reference tracking dynamics and ship motions are what are most critical to scale in this scenario. While dynamic scaling has not been considered in such tests, the results in this section show that Froude scaling both the aircraft closed-loop reference tracking dynamics and full-scale ship data prior to applying it to a motion platform gives a better representation of the target full-scale use case.

4.3 Dynamic Similarity for Scaled Disturbance Rejection

When scaling the reference tracking dynamics, it was assumed that the closed-loop response to a reference input accurately approximated the command model, greatly simplifying the scaling relationships. For disturbance rejection it is more complicated, however, as the scaling of the open-loop dynamics comes into play. This can be seen by considering the closed-loop system shown in Fig. 4.4, where the controller $C(s)$ must compensate for disturbances and $C(s)$ and $G(s)$ are both SISO systems. Ideally, for perfect scaling of the closed-loop system to be attained, both the output response due to an input disturbance ($y(s)/u_d(s)$) and the output response due to an output disturbance ($y'(s)/y_d(s)$) should scale following the Froude scaling rules of table 4.1. In order to achieve a scaled input disturbance response $y(s)/u_d(s)$ and output disturbance response $y'(s)/y_d(s)$ without needing to consider the scaling of the bare airframe dynamics, it must

be possible to design a controller that simultaneously achieves this for both $y(s)/u_d(s)$ and $y'(s)/y_d(s)$ regardless of dissimilarity in the model-scale and full-scale open-loop dynamics. This, however, proves to be infeasible.

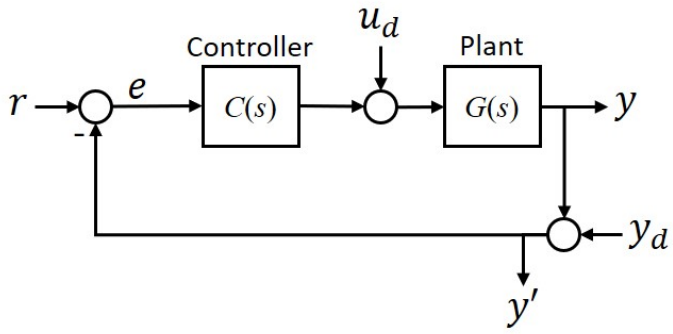


Figure 4.4: Simple feedback loop with input and output disturbances.

To see this, we first examine the output due to an input disturbance, as aerodynamic gusts would be more appropriately modelled through an equivalent input disturbance since gusts will apply forces and moments to the system. Even designing a control law to achieve similarity in $y(s)/u_d(s)$ alone has been found infeasible without the model-scale open-loop dynamics appropriately scaled, however. This is intuitive as the input disturbance u_d shown in Fig. 4.4 is directly injected into the plant model, affecting the output before the controller has the chance to act on the signal. In practice, this means an unimplementable controller with higher order derivative terms would be necessary to achieve similarity in $y(s)/u_d(s)$ while compensating for dissimilarity between the model-scale and full-scale dynamics.

Further, even if it was practical to implement a controller that achieves dynamic similarity in $y(s)/u_d(s)$ without accurately scaled open-loop dynamics, it is still not possible to achieve this for both $y(s)/u_d(s)$ and $y'(s)/y_d(s)$ simultaneously. To show this, we will again look at the input disturbance response $y(s)/u_d(s)$. Noting that the transfer function for $y(s)/u_d(s)$ is given as

$$\frac{y(s)}{u_d(s)} = \frac{G(s)}{1 + G(s)C(s)} \tag{4.11}$$

we can solve for a controller that theoretically achieves similarity in $y(s)/u_d(s)$ across scales. To do this it is assumed that the transfer function for $y(s)/u_d(s)$ is known from the full-scale closed-loop system, and the desired model-scale response can therefore be determined via Froude scaling. Using this assumption and Eq. (4.11), the controller is

solved for as

$$Fr\left[\left(\frac{y}{u_d}\right)_{fs}\right] = \frac{G_{ms}}{1+G_{ms}C_{ms,u_d}} \quad \Rightarrow \quad C_{ms,u_d} = Fr\left[\left(\frac{y}{u_d}\right)_{fs}^{-1}\right] - G_{ms}^{-1} \tag{4.12}$$

where the s notation is dropped for convenience and subscripts ms and fs are used to denote model-scale and full-scale transfer functions, respectively. Additionally, note that the subscript u_d denotes that the control law is derived to achieve a scaled input disturbance response and the operator $Fr[\cdot]$ is used to denote Froude scaling a system from full- to model-scale. Substituting for y/u_d of the full-scale system, Eq. (4.12) can be written as

$$C_{ms,u_d} = Fr\left[\left(\frac{G_{fs}}{1+G_{fs}C_{fs}}\right)^{-1}\right] - G_{ms}^{-1} \tag{4.13}$$

This controller will now be compared to the controller derived by examining the output response due to an output disturbance. Noting that the transfer function describing $y'(s)/y_d(s)$ is given by

$$\frac{y'(s)}{y_d(s)} = \frac{1}{1+G(s)C(s)} \tag{4.14}$$

we can solve for a controller that achieves similarity in $y'(s)/y_d(s)$ across scales. Assuming that a desired model-scale transfer function for $y'(s)/y_d(s)$ is known by Froude scaling that of the full-scale system and using Eq. (4.14), the controller is solved for as

$$Fr\left[\left(\frac{y'}{y_d}\right)_{fs}\right] = \frac{1}{1+G_{ms}C_{ms,y_d}} \quad \Rightarrow \quad C_{ms,y_d} = G_{ms}^{-1}\,Fr\left[\left(\frac{y'}{y_d}\right)_{fs}^{-1}\right] - G_{ms}^{-1} \tag{4.15}$$

where the subscript y_d denotes that the control law is derived to achieve a scaled output disturbance response. Substituting for $y'(s)/y_d(s)$ of the full-scale system yields

$$C_{ms,y_d} = G_{ms}^{-1}\,Fr\left[\left(\frac{1}{1+G_{fs}C_{fs}}\right)^{-1}\right] - G_{ms}^{-1} \tag{4.16}$$

In order for the closed loop response to both input and output disturbances to satisfy Froude scaling simultaneously, the control laws in Eq. (4.13) and Eq. (4.16) must be identical. Comparing Eq. (4.13) and Eq. (4.16), it is observed that the second term in both equations is identical but that the first term differs. Equating the first term of both

equations gives

$$Fr\left[\left(\frac{G_{fs}}{1+G_{fs}C_{fs}}\right)^{-1}\right] = G_{ms}^{-1}\,Fr\left[\left(\frac{1}{1+G_{fs}C_{fs}}\right)^{-1}\right] \tag{4.17}$$

Noting that Froude scaling in the Laplace domain is equivalent to scaling pole and zero frequencies as well as magnitude, and this can be done separately to polynomials in s that are multiplied, divided, or added together, this can be written as

$$\frac{Fr\left[1+G_{fs}C_{fs}\right]}{Fr\left[G_{fs}\right]} = \frac{Fr\left[1+G_{fs}C_{fs}\right]}{G_{ms}} \tag{4.18}$$

Comparing the denominator of the resulting equation, it is seen that the only way the controller in Eq. (4.13) (derived to achieve similarity in $y(s)/u_d(s)$) can be equivalent to the controller in Eq. (4.16) (derived to achieve similarity in $y'(s)/y_d(s)$) is if the model-scale open-loop dynamics are equivalent to those of the Froude-scaled full-scale aircraft ($Fr[G_{fs}] = G_{ms}$). Further, if the open-loop dynamics do relate across scales via Froude scaling, then further simplifying Eq. (4.16) and substituting $Fr[G_{fs}] = G_{ms}$ yields

$$\begin{aligned} C_{ms} &= \frac{Fr\left[1+G_{fs}C_{fs}\right]}{G_{ms}} - \frac{1}{G_{ms}} && (4.19)\\ &= \frac{1+Fr\left[G_{fs}\right]Fr\left[C_{fs}\right]}{G_{ms}} - \frac{1}{G_{ms}} && (4.20)\\ &= \frac{1+G_{ms}Fr\left[C_{fs}\right]}{G_{ms}} - \frac{1}{G_{ms}} && (4.21)\\ &= Fr\left[C_{fs}\right] && (4.22) \end{aligned}$$

This analysis shows that in order to achieve a scaled closed-loop output response to both input and output disturbances, both the model-scale open-loop dynamics and the model-scale controller should approximate the scaled open-loop dynamics and controller from the full-scale system.

While it has been shown that the most appropriate way to achieve a scaled response to disturbances is to design a model-scale system with both scaled open-loop dynamics and a scaled controller, the multi-rotor UAVs used in this work were not built to achieve scaled open-loop dynamics relative to any particular full-scale aircraft. This is an acknowledged drawback of the choice to use a generic multi-rotor UAV, with the trade-off being that a generic multi-rotor is easier to build than a Froude-scaled model of a particular aircraft.

In addition, the majority of the model-scale tests conducted for this work (discussed in Chapter 5.1) were conducted in the absence of significant gusts or turbulence, meaning that the tests were primarily focused on evaluating the ability of different guidance and control configurations to compensate for ship motion. For this scenario, the scaling of the reference tracking dynamics and ship motion are most important, as the UAV precisely follows the command models. The model-scale tests discussed in Chapter 5.2 (which did include significant aerodynamic disturbances) were focused on a comparison between different guidance algorithms, and a cruder scaling of closed-loop response to aerodynamic disturbances was deemed sufficient provided the lower-level control design was consistent across all guidance methods tested. If model-scale tests were conducted with the intent of determining necessary closed-loop disturbance rejection criteria to perform safe landings in a scaled aerodynamic environment, however, then utilizing a bare airframe with scaled open-loop dynamics would be required.

Even though the UAVs used for the tests conducted for this work were not designed to achieve accurately scaled open-loop dynamics, it should be noted that reasonable similarity in the scaled output disturbance response $y'(s)/y_d(s)$ could still be obtained. The reason for this is because the output disturbance y_d passes through both the plant and the controller to produce the perturbed output y'. This means that the scaling of the loop transfer function $G(s)C(s)$ is what matters for achieving similarity in $y'(s)/y_d(s)$ and the feedback compensator can be designed to capture important aspects of the scaled output disturbance response. This is meaningful, as commonly used metrics such as disturbance rejection bandwidth and disturbance rejection peak (defined in Chapter 4.4.2) are defined from $y'(s)/y_d(s)$. Physically, disturbance rejection bandwidth characterizes how fast the hold variable of the closed loop system will return to its commanded value after a disturbance. Disturbance rejection peak relates to the overshoot and damping of the hold variable as it returns to the commanded variable after the disturbance. As will be discussed in Chapter 4.4.3, the feedback compensators will be designed here to achieve disturbance rejection bandwidths and peaks determined by scaling representative values for full-scale rotorcraft. This at least provides a crude scaling that captures the salient characteristics of the response to disturbances.

4.4 Choosing Model-Scale Controller Parameters for Froude Scaled Closed-Loop Dynamics

4.4.1 Choosing Parameters for Scaled Reference Tracking

Based on the results presented in Chapter 4.2, Froude scaled reference tracking dynamics can be attained by simply Froude scaling the command model parameters of the EMF control law. To exemplify the process for doing this, the longitudinal control axis will first be considered. Say we have a full-scale model with a pitch tracking bandwidth denoted by $\omega_{\theta,fs}$. Following the Froude scaling framework discussed in section 4.1, we simply scale this by $\sqrt{N_F}$ to arrive at the model-scale pitch command filter frequency:

$$\omega_{\theta,cf} = \omega_{\theta,fs}\sqrt{N_F} \tag{4.23}$$

The outer loop position control command filter frequency is then determined by dividing $\omega_{\theta,cf}$ by 5:

$$\omega_{X,cf} = \frac{\omega_{\theta,cf}}{5} \tag{4.24}$$

This factor of 5 is a good rule of thumb for ensuring adequate frequency separation between the inner and outer control loops.

The time delay included on position and velocity commands, denoted by $\tau_{\theta,\phi}$ in Fig. 2.13, is also calculated based on the attitude command filter. For position control, the attitude dynamics are neglected in the inversion, so we capture the phase of the attitude dynamics with this delay. Recall that the attitude tracking dynamics are forced to approximate the attitude command filters plus some small time delay. For pitch with the command filter given in Eq. 2.25, ignoring the high frequency pole p_θ and focusing on the second order filter that governs the reference tracking response, this gives

$$\frac{\theta(s)}{\theta_{cmd}(s)} \approx \frac{\omega_{\theta,cf}^2}{s^2 + 2\zeta\omega_{\theta,cf}s + \omega_{\theta,cf}^2} e^{-\tau_q s} \tag{4.25}$$

To approximate the phase of the attitude dynamics in Eq. (4.25), we calculate $\tau_{\theta,\phi}$ as follows:

$$\tau_{\theta,\phi} = 1.65/\omega_{\theta,cf} + \tau_q \tag{4.26}$$

Note that this is valid with the attitude command filter damping ratio set to 0.8, which

was done for all command filters. Also, referring to Fig. 2.13, note that $\tau_{\theta,\phi}$ delays both the X^I and Y^I position and velocity commands. While we only considered the pitch attitude tracking response when calculating $\tau_{\theta,\phi}$ in Eq. (4.26), the roll attitude command filter is constrained in the control design to be identical to the pitch command filter. For a highly augmented aircraft with full authority flight control, this is a reasonable requirement. Additionally, the values of τ_q and τ_p are generally much smaller than the portion of $\tau_{\theta,\phi}$ attributed to the attitude command filters, meaning differences in τ_q and τ_p do not have a large impact on the value of $\tau_{\theta,\phi}$.

By scaling outer loop delay $\tau_{\theta,\phi}$ and command filter frequency $\omega_{X,cf}$ based on the inner loop command filter frequency $\omega_{\theta,cf}$, the X position and pitch attitude tracking responses are consistently scaled based on just the full-scale pitch attitude tracking bandwidth $\omega_{\theta,fs}$. This same process was used to scale the Y position and roll attitude tracking responses. For altitude and yaw control, however, there is no outer control loop and the reference tracking bandwidths can be set by scaling command filter frequencies by $\sqrt{N_F}$ as was done for pitch in Eq. (4.23). That is,

$$\omega_{Z,ms} = \omega_{Z,fs}\sqrt{N_F} \tag{4.27}$$

$$\omega_{\psi,ms} = \omega_{\psi,fs}\sqrt{N_F} \tag{4.28}$$

4.4.2 Definition of Disturbance Rejection Bandwidth and Disturbance Rejection Peak

In Chapter 4.3 it was shown that a Froude scaled response to both input and output disturbances cannot be attained here since the model-scale UAVs used for this work were not designed to match the scaled open-loop dynamics of any particular airframe. It is still possible to design the feedback compensator to attain similar properties in the response to output disturbances, however. Here feedback gains will be selected to match a Froude scaled disturbance rejection bandwidth (DRB), which is defined from the output disturbance response. This provides a crude scaling of the disturbance rejection properties of the system. The following subsection will present the method used for choosing control gains to meet a scaled DRB, but prior to discussing this process DRB should be formally defined. The discussion given here will summarize the definition given by Berger in [64].

DRB is a frequency domain metric obtained from the response of the control system hold variable to a disturbance at the hold variable output [64]. An example of this is

shown for a roll axis attitude-command-attitude-hold (ACAH) control system subject to a roll attitude disturbance ϕ_d in Fig. 4.5, where ϕ' is the measured output. The transfer function from ϕ_d to ϕ' is obtained from the sensitivity function S as

$$\frac{\phi'(s)}{\phi_d(s)} = S_{22} = [(I + GK)^{-1}]_{22} \tag{4.29}$$

where S_{22} represents the element at the second row and column of S, and G and K are defined as

$$\begin{aligned} G &= \begin{bmatrix} p/\delta_{lat} \\ p/\delta_{lat}/s \end{bmatrix} \\ K &= \begin{bmatrix} K_p & K_\phi \end{bmatrix} \end{aligned} \tag{4.30}$$

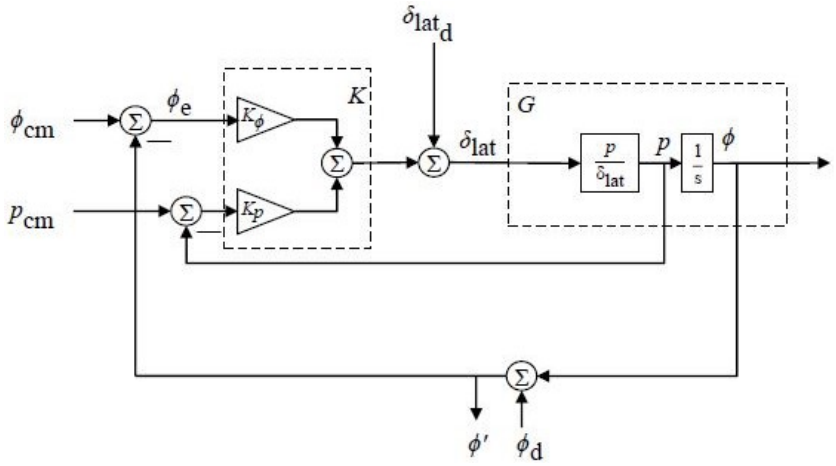


Figure 4.5: Example roll axis ACAH system (adapted from [64])

DRB is defined as the frequency where the magnitude of the hold variable response to a disturbance (i.e. $\frac{\phi'(s)}{\phi_d(s)}$ for this example) crosses -3 dB. This is exemplified graphically in Fig. 4.6. Note that an additional metric referred to as disturbance rejection peak (DRP) is also defined in Fig. 4.6: DRP is taken as the peak magnitude of the hold variable response to a disturbance. Physically, DRB gives a measure of how quickly the hold variable will return to its held value after a disturbance and DRP gives a measure of the overshoot and damping of the hold variable after a disturbance. These have become common metrics for rotorcraft flight control design guidance.

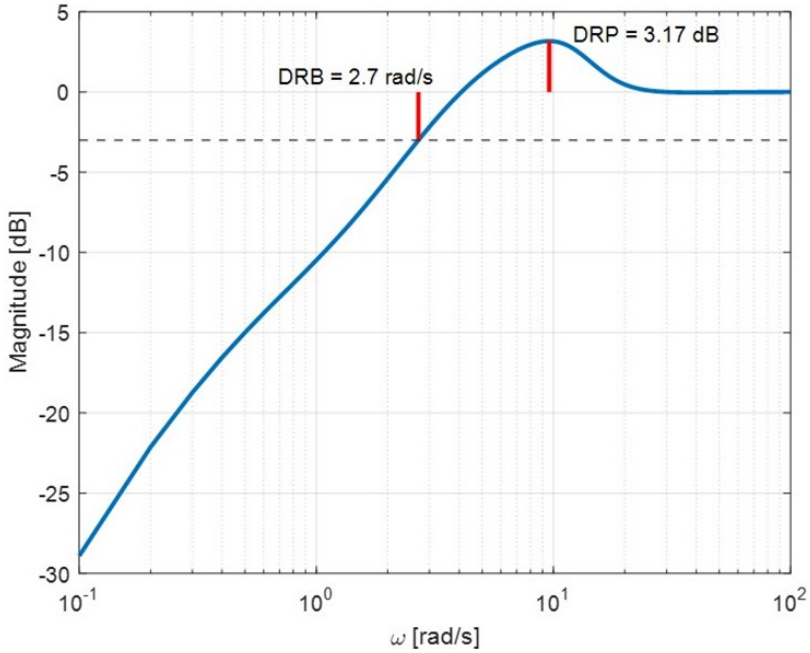


Figure 4.6: Graphical representation defining DRB and DRP.

4.4.3 Choosing Parameters for Scaled Disturbance Rejection Bandwidth

The desired model-scale DRB is calculated through the same process as shown in Eq. (4.23): multiplying the full-scale DRB by $\sqrt{N_F}$. To match this value the feedback compensators must be systematically tuned. Starting again with the longitudinal control axis, first consider the pitch attitude loop. The pitch proportional gain $K_{P,\theta}$ is set to achieve a desired gain crossover frequency in the loop transfer function. The zero in the loop transfer function at $K_{I,\theta}/K_{P,\theta}$ is then placed by setting

$$K_{I,\theta} = 0.2 K_{P,\theta} \omega_{co} \tag{4.31}$$

where ω_{co} is the gain crossover frequency. This is a commonly used rule-of-thumb to provide frequency separation between the integral and proportional compensators [61], and allows the pitch attitude DRB to be tuned by adjusting only $K_{P,\theta}$. This same method was used for tuning the roll and yaw PI controllers.

The X position hold control gains are chosen by approximately specifying the error dynamics in the UAV level frame. To do this the UAV level frame X position dynamics are modelled as shown in Eq. 2.31, where θ_{cmd} is considered the driving input. The loop is then closed on this model with the PID feedback compensator used in the X and Y

position control law. Then, considering the effect of an additive disturbance X^{lf}_{dst} at the X^{lf} position output on the position tracking error, the following expression for the error dynamics can be derived:

$$\frac{e(s)}{X^{lf}_{dst}(s)} = \frac{s^3}{s^3 - g(K_{D,X}s^2 + K_{P,X}s + K_{I,X})} \\ where \;\; e(s) = X^{lf}_{cmd}(s) - X^{lf}(s) \tag{4.32}$$

The denominator of Eq. (4.32) can then be factored into a second order and first order polynomial, yielding

$$\frac{e(s)}{X^{lf}_{dst}(s)} = \frac{s^3}{(s^2 + 2\zeta_d\omega_d s + \omega_d^2)(s + p_d)} \tag{4.33}$$

where we set $\zeta_d = 1$, $p_d = 0.2\omega_d$, and ω_d is tuned to give a desired trade-off between stability margins and DRB. This reduces the tuning process to adjusting only ω_d. The X position control PID gains are then related to ζ_d, ω_d, and p_d through the equations

$$\begin{aligned} K_{P,X} &= -\frac{(\omega_d^2 + 2\zeta_d\omega_d p_d)}{g} \\ K_{I,X} &= -\frac{p_d\omega_d^2}{g} \\ K_{D,X} &= -\frac{(2\zeta_d\omega_d + p_d)}{g} \end{aligned} \tag{4.34}$$

The same process was used for determining the Y position hold PID gains.

For scaling the Z^I controller DRB, a process similar to that applied to the attitude PI controllers is used. First, we factor the Z^I feedback compensator as shown below:

$$K_Z(s) = \frac{K_{D,Z}s^2 + K_{P,Z}s + K_{I,Z}}{s} = \frac{K(s + z_1)(s + z_2)}{s} \tag{4.35}$$

The gain K in Eq. (4.35) is then used to set the loop transfer function gain crossover frequency and zeros z_1 and z_2 are set to 0.1 times the gain crossover frequency. The PID gains are then found calculated from the equations

$$\begin{aligned} K_{P,Z} &= K(z_1 + z_2) \\ K_{I,Z} &= z_1 z_2 K \\ K_{D,Z} &= K \end{aligned} \tag{4.36}$$

allowing the altitude control gains to be tuned by only adjusting K.

4.5 Scaling Control Laws from Model-Scale to Full-Scale

The previous section discussed how controller parameters were chosen in this work to scale the reference tracking dynamics and disturbance rejection bandwidths from full-scale down to model-scale. The purpose in doing this is to provide reasonably scaled closed-loop dynamics when evaluating the guidance algorithms presented in Chapter 3 in model-scale experiments. But while this work uses representative full-scale closed-loop dynamics to determine control gains for model-scale experimental evaluations, it is worth noting that scaling could be applied in the opposite direction. For example, if the model-scale aircraft were built to exhibit approximately Froude scaled open-loop dynamics in the frequency range of interest for control design, then a new control mode could be designed and evaluated at model-scale. The control gains could then be scaled-up and applied to the full-scale aircraft.

To exemplify the process of scaling a control law from model-scale to full-scale, a simple example based on the longitudinal dynamics of the UH-60 in hover will be explored. A crude, decoupled model of the longitudinal dynamics for a helicopter at hover can be expressed as

$$\begin{bmatrix} \dot{u}_{fs} \\ \dot{q}_{fs} \\ \dot{\theta}_{fs} \\ \dot{X}_{fs} \end{bmatrix} = \begin{bmatrix} X_u & 0 & -g & 0 \\ M_u & M_q & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{fs} \\ q_{fs} \\ \theta_{fs} \\ X_{fs} \end{bmatrix} + \begin{bmatrix} X_{\delta_{lon}} \\ M_{\delta_{lon}} \\ 0 \\ 0 \end{bmatrix} \delta_{lon,fs} \tag{4.37}$$

where the stability derivatives X_u, M_u, and M_q and control derivatives $X_{\delta_{lon}}$ and $M_{\delta_{lon}}$ are determined from the linearization of the PSUHeloSim UH-60 model [65] and tabulated in table 4.2. Note that the subscript fs here is included to emphasize that this is a model of the full-scale aircraft.

Table 4.2: Full-Scale Stability and Control Derivatives

	$\boldsymbol{X_u}$ $[\boldsymbol{1/s}]$	$\boldsymbol{M_u}$ $[\boldsymbol{1/(ft\ s)}]$	$\boldsymbol{M_q}$ $[\boldsymbol{1/s}]$	$\boldsymbol{X_{\delta_{lon}}}$ $[\boldsymbol{ft/s^2}]$	$\boldsymbol{M_{\delta_{lon}}}$ $[\boldsymbol{1/s^2}]$
value	-0.0041	3.6518e-04	-0.1768	0.0402	0.0047

Since the goal here is to first design a controller at model-scale and then use the model-scale controller to determine the full-scale controller, a model of the model-scale dynamics is needed. For this simple example, Froude-scaled model-scale dynamics are obtained from Eq. (4.37) by Froude scaling each of the stability and control derivatives. For example, X_u has units of $1/s$ and therefore scales following the frequency scaling rule given in table 4.1. That is, $X_{u,ms} = X_{u,fs}\sqrt{N_F}$ where N_F is again the Froude scaling

factor. For this example a scale factor of $N_F = 10$ was used. Applying dimensional analysis as was done to determine the jerk scaling factor in Eq. (4.1), scale factors for the remaining stability and control derivatives can be determined. The model-scale dynamics that will be used in this example are then given by

$$\begin{bmatrix} \dot{u}_{ms} \\ \dot{q}_{ms} \\ \dot{\theta}_{ms} \\ \dot{X}_{ms} \end{bmatrix} = \begin{bmatrix} X_u\sqrt{N_F} & 0 & -g & 0 \\ M_u N_F^{3/2} & M_q\sqrt{N_F} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{ms} \\ q_{ms} \\ \theta_{ms} \\ X_{ms} \end{bmatrix} + \begin{bmatrix} X_{\delta_{lon}} \\ M_{\delta_{lon}} N_F \\ 0 \\ 0 \end{bmatrix} \delta_{lon,ms} \tag{4.38}$$

A controller can now be designed based on these model-scale dynamics.

Here the model-scale control design uses the EMF control architecture shown in Fig. 4.7, which is very similar to the position control law implemented on the UAVs in this work (see Chapter 2.6.1). The pitch attitude inversion model for the model-scale design is given by a simple first order approximation of the pitch dynamics:

$$\frac{\widehat{\theta(s)}}{\delta_{lon}(s)} = \frac{M_{\delta_{lon}} N_F}{s(s - M_q\sqrt{N_F})} \tag{4.39}$$

The pitch attitude command model at model-scale was given a damping ratio of $\zeta_\theta = 0.8$ and frequency of $\omega_{\theta,ms} = 10\ rad/s$, and the pitch PID controller gains were tuned to meet the scaled DRB guideline for level 1 handling qualities as specified in ADS33E-PRF. The model-scale position controller command model was given a damping ratio of $\zeta_x = 0.8$ and frequency of $\omega_{x,ms} = 2\ rad/s$. The outer loop time delay was determined from the pitch attitude command model frequency in the same manner as shown in Eq. (4.26). The position loop inversion model shown in Fig. 4.7 is based on the assumption that the X position dynamics can be reasonably modelled by $\ddot{X} = -g\theta$. The model-scale position

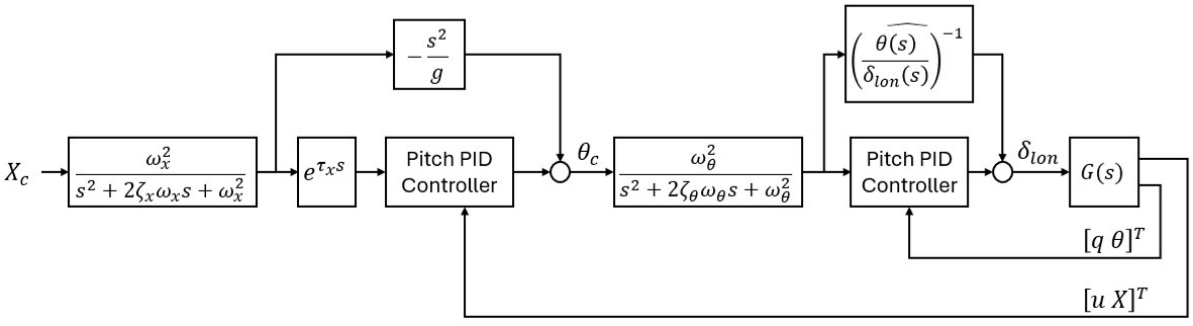


Figure 4.7: EMF control law used for scaled control design example.

control PID gains were chosen using this same assumption and approximately specifying the position tracking error dynamics, as was done to derive the position control PID gain formulas given in Eq. (4.34) where the frequency parameter ω_d defines all three gains. The position and attitude control gains from the model-scale design are given in table 4.3.

Table 4.3: Model-scale control gains

	$\boldsymbol{K_{P,\theta,ms}}$	$\boldsymbol{K_{I,\theta,ms}}$	$\boldsymbol{K_{D,\theta,,ms}}$	$\boldsymbol{K_{P,X,ms}}$	$\boldsymbol{K_{I,X,ms}}$	$\boldsymbol{K_{D,X,ms}}$
value	245.45	151.96	59.83	-0.0627	-0.0107	-0.0821

The full-scale controller parameters can now be set by Froude scaling the model-scale parameters. For example, the pitch-attitude derivative control gain $K_{D,\theta}$ acts on the pitch rate error, which has units of rad/s. Since the attitude control PID gains sum together to produce the dimensionless control input δ_{lon}, $K_{D,\theta}$ must have units of s/rad so that the product of pitch rate tracking error and $K_{D,\theta}$ is unitless. Scaling the units of $K_{D,\theta}$ gives

$$\frac{s_{fs}}{rad_{fs}} = \frac{s_{ms}\sqrt{N_F}}{rad_{ms}} \tag{4.40}$$

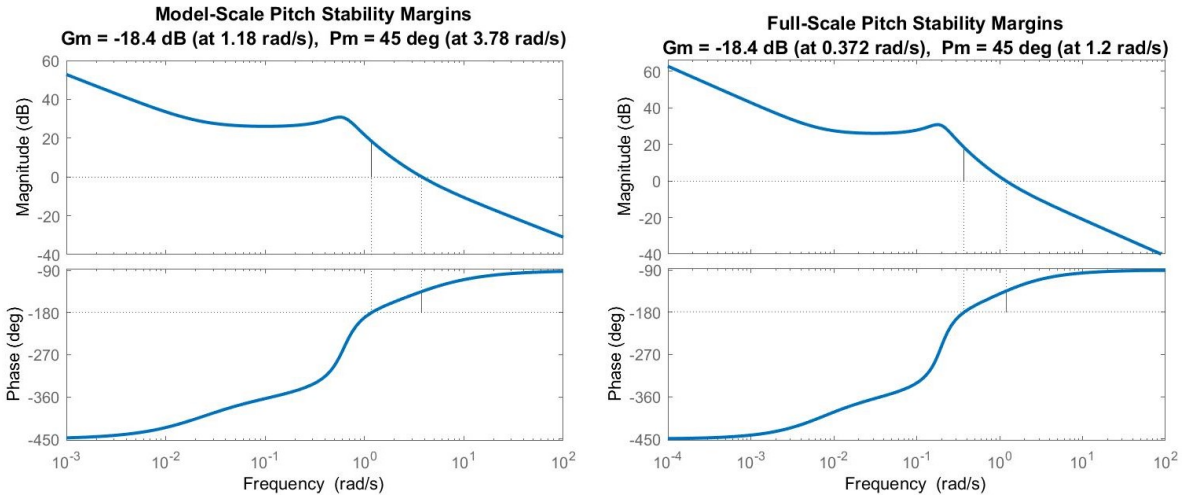
which shows that the full-scale pitch attitude derivative gain is increased from model-scale by $\sqrt{N_F}$ (i.e., $K_{D,\theta,fs} = K_{D,\theta,ms}\sqrt{N_F}$) for appropriate Froude scaling. Applying the same process to scale the pitch control proportional and integral gains, the equations for determining the full-scale gains from the model-scale design are

$$\begin{aligned} K_{P,\theta,fs} &= K_{P,\theta,ms} \\ K_{I,\theta,fs} &= K_{I,\theta,ms}/N_F \\ K_{D,\theta,fs} &= K_{D,\theta,ms}\sqrt{N_F} \end{aligned} \tag{4.41}$$

For the outer loop position controller, the control gains could be obtained in the same manner. More simply though, since the position control gains are determined by the frequency parameter ω_d as shown in Eq. (4.34), equivalent scaling is obtained by simply scaling ω_d. That is, setting $\omega_{d,fs} = \omega_{d,ms}/\sqrt{N_F}$. All other parameters in the full-scale control law are determined by Froude scaling as well. For example, full-scale command model frequencies are determined from model-scale using the aforementioned frequency scaling rule.

With Froude-scaled open-loop dynamics and the full-scale control law determined by Froude scaling the model-scale design, it is expected that key aspects of the closed-loop system like stability margins and disturbance rejection properties will be related by

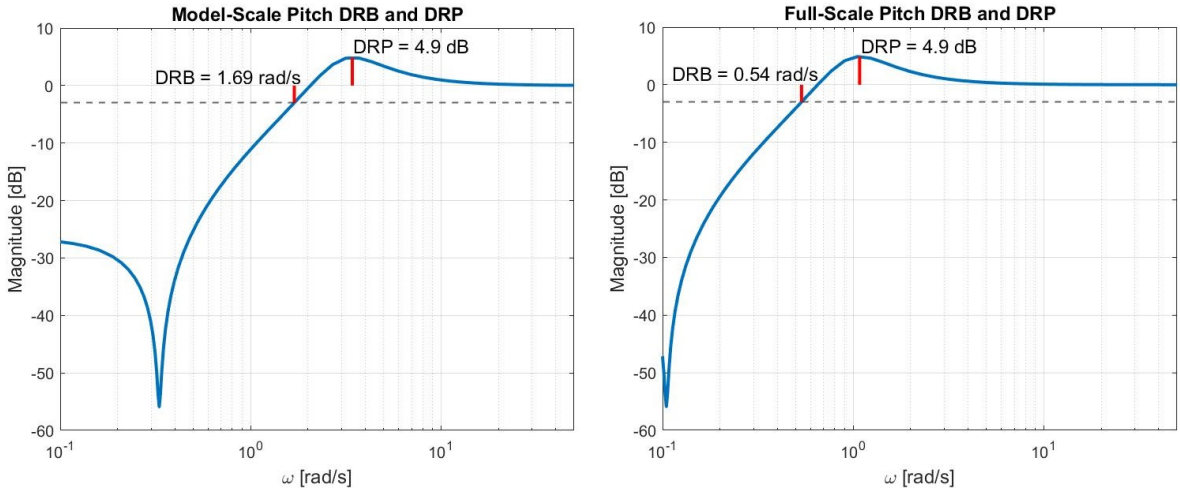
Froude scaling as well. First we will verify the scaling of gain and phase margin, as determined by breaking the loop at the control input. For phase margin, one-to-one scaling is expected since the phase angle is unitless. One-to-one scaling is also expected for gain margin since the transfer function from an input disturbance to the resulting control input is also unitless. These expectations are confirmed by comparing Fig. 4.8a and Fig. 4.8b, where we see stability margins are maintained across scales. Only the gain and phase crossover frequencies change, and they change following the square root scaling rule expected of Froude scaling. Similar expectations are confirmed when analyzing the disturbance rejection properties. For example, DRB will scale following the square root frequency scaling rule and DRP will be maintained across scales since the ratio of output over output disturbance will be unitless. This result is demonstrated by the DRB and DRP values reported in Fig. 4.9. Additionally, as expected, closed-loop reference tracking dynamics will scale appropriately. This is verified in Fig. 4.10 where a 1 ft position step command is applied to the model-scale system and a 10 ft step command (increased by scale factor N_F =10) is applied to the full-scale system. Comparing Figs. 4.10a and 4.10b, the plots appear identical but the x and y axis scales are off by the associated Froude scaling factors, verifying appropriate scaling.



(a) Model-scale margins.

(b) Full-scale margins.

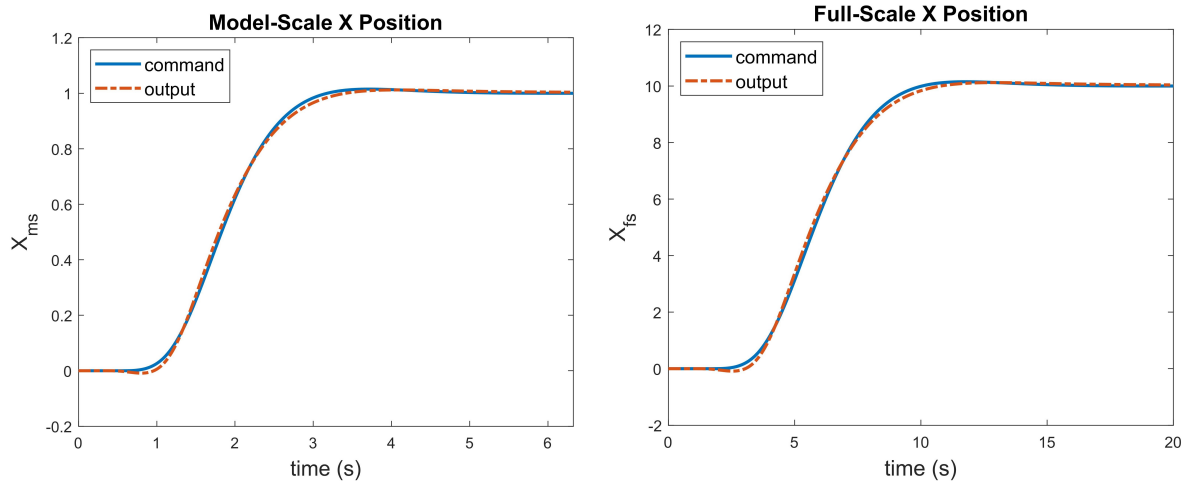
Figure 4.8: Pitch attitude stability margins with model-scale control design ($1/10^{\text{th}}$ scale, or $N_F = 10$) and full-scale control design determined from model-scale.



(a) Model-scale DRB and DRP.

(b) Full-scale DRB and DRP.

Figure 4.9: Pitch attitude DRB and DRP with model-scale control design ($1/10^{\text{th}}$ scale, or $N_F = 10$) and full-scale control design determined from model-scale.



(a) Model-scale step response.

(b) Full-scale step response.

Figure 4.10: Response to X position step command with model-scale control design ($1/10^{\text{th}}$ scale, or $N_F = 10$) and full-scale control design determined from model-scale.

The preceding example is simplified in that the open-loop dynamics relate perfectly across scales by Froude scaling, which will only be approximately true in a best-case scenario. Still, this does demonstrate that it is theoretically possible to perform control design at model-scale and translate the design and expected robustness and performance to a larger scale model. Further, work by other researchers has indicated that approximately Froude scaled open-loop dynamics can be attained by making this a goal in the design of

the model-scale aircraft. For example, rotorcraft built with roughly Froude scaled rotor radius, mass, and inertia have been shown in some studies to exhibit roughly Froude scaled open-loop dynamics [52–54]. Though it is not explored further in this work, the feasibility of designing a model-scale aircraft to achieve approximately Froude scaled dynamics, performing system modelling and control design with the aid of model-scale flight test, and then scaling the model-scale control design to full-scale would be an interesting area of future work.

Chapter 5
Model–Scale Autonomous Ship Landing Experiments

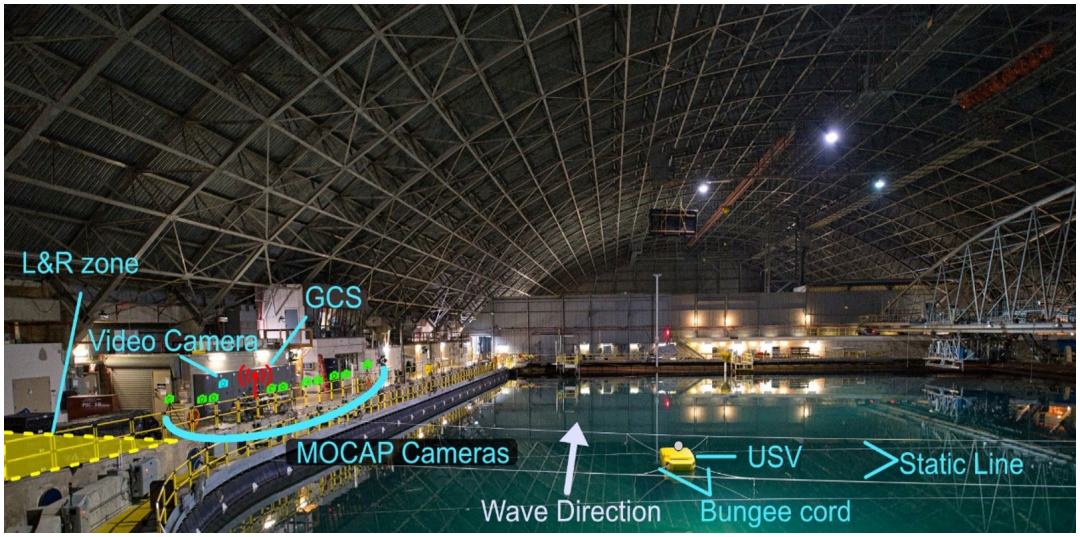
Using the scaling methodology from the previous chapter, the guidance algorithms presented in Chapter 3 were evaluated in a series of model-scale autonomous landing experiments. The majority of the experiments were performed at the Maneuvering and Seakeeping Basin (MASK) located at the Naval Surface Warfare Center Carderock Division (NSWCCD). For these tests aerodynamic disturbances were not included, and the focus was on evaluating the ability of the baseline and QP guidance algorithms to compensate for deck motion. Additionally, control law parameters were adjusted during testing to evaluate the ability of both guidance algorithms to compensate for reduced speed of response and maneuverability. A smaller set of tests were also performed at the Penn State Indoor Flight Facility. These tests incorporated wind gusts and were conducted to evaluate the ability of both guidance algorithms to perform in the presence of a significant aerodynamic disturbance. This chapter will describe the experimental setups and results obtained from both sets of model-scale tests.

5.1 Autonomous Landing Experiments at the Maneuvering and Seakeeping Basin

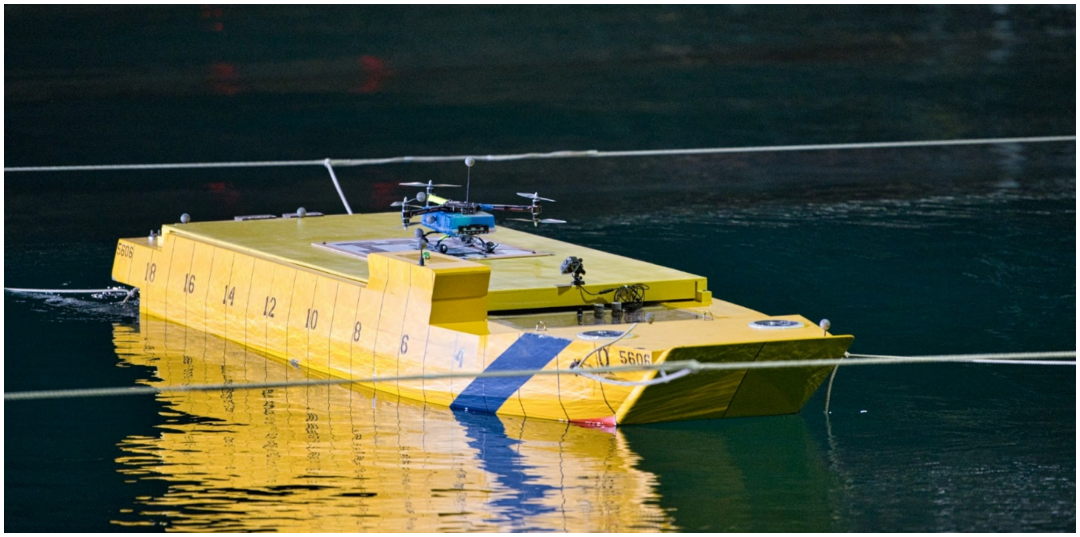
5.1.1 Experimental Setup

A picture of the experimental setup at the MASK facility is shown in Fig. 5.1a. The MASK facility contains a 360 foot long by 240 foot wide wave basin with 216 individually controlled electro-mechanical wave boards around the edge of the basin. The wave boards

can produce appropriately scaled wave conditions that are precisely controlled to match a specified wave spectrum. An OptiTrack motion capture system is installed in the MASK, with the motion capture software running on the ground control station (GCS). This allows for accurate tracking of deck and aircraft positions and attitudes. A 20 foot long unmanned ship vessel (USV) was loosely tethered in the wave pool, as shown in Fig. 5.1b. Tethering prevented unbounded drift in the ship's heading and position, but allowed significant oscillations in all six degrees of freedom. A launch and recovery (L&R) zone was designated off the shore of the basin for takeoff and landing between test runs.



(a)



(b)

Figure 5.1: (a) Test setup in the MASK facility at the NSWCCD. (b) Model-ship used during landings.

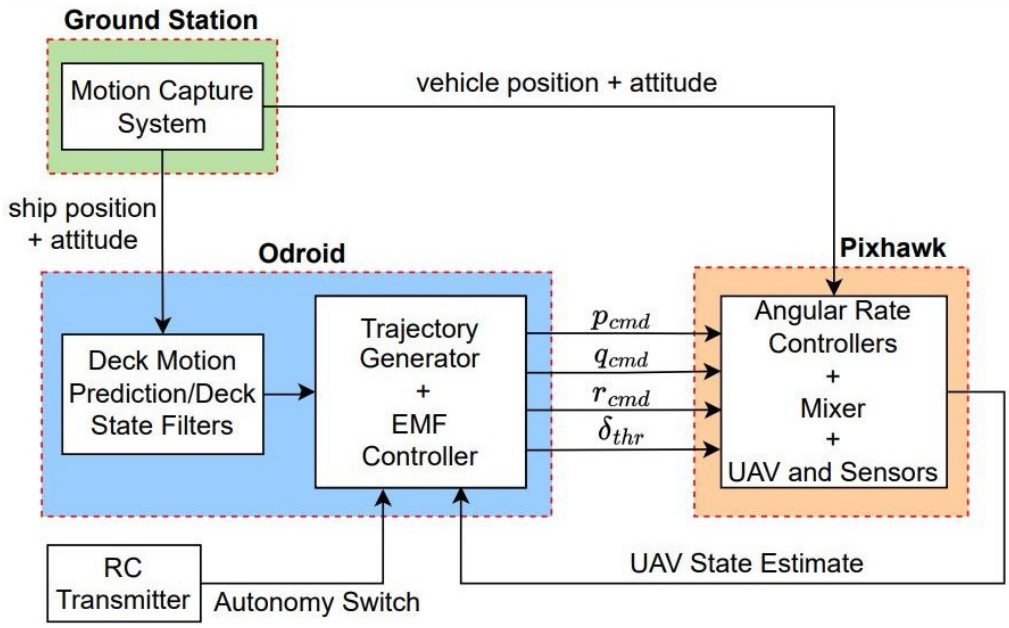


Figure 5.2: Test hardware and software integration.

The flight tests flown in the MASK facility used the hexacopter UAV described in section 2.1. The UAV is equipped with a WiFi module, which is used to receive data from the ground station computer as shown in Fig. 5.2. The ground station computer sends out position and attitude measurements of both the ship and UAV. The UAV measurements are picked up by the PX4 firmware for use in state estimation, and the estimated aircraft states are then sent to the UAV's Odroid XU4 onboard computer for use in trajectory generation and control. The ship position and attitude measurements are used directly in the trajectory generation algorithms.

5.1.2 Wave Conditions

Landings were performed in three separate stochastic wave conditions. The wave conditions are referred to as stochastic because a single sinusoidal wave form was not used. Rather, the waves are produced to match a realistic wave spectrum. The standard deviation (STD) and maximum difference from the mean are tabulated for each degree of freedom at each wave condition in table 5.1. The mean magnitude of ship motion for each degree of freedom is also tabulated in table 5.1. For example, the reported mean magnitude of deck heave motion is calculated as $mean\left(\vec{Z}_D^I - \bar{Z}_D^I\right)$ where $\vec{Z}_D^I$ is a vector of all ship altitude measurements taken at a given wave condition and $\bar{Z}_D^I$ is the average altitude of the deck at that wave condition. Ship heading and heading rate were excluded

from this table as oscillations in heading were mild for these tests. Note that ω_p in table 5.1 is the frequency corresponding to the peak in the wave amplitude spectrum, and ψ_w is the wave heading. For wave conditions 1 and 2 the waves approached the bow of the ship directly, but for wave condition 3 the waves approached at an angle of 45 degrees to starboard.

Wave condition 1 was the most mild, as reflected by the deck motion statistics shown in table 5.1. Wave conditions 2 and 3 were similar, but the 45 degree wave angle for condition 3 led to more aggressive roll and sway motion and less aggressive surge motion. Wave condition 3 was found to be the most difficult for the autonomy algorithms to perform landings in, though it should be noted that this may not be entirely due to larger lateral motions. The wave makers in the MASK dissipate less energy with a 45 degree wave heading than with 0 degree wave heading, which leads to slightly more aggressive motions on average.

Table 5.1: Ship motion properties for each wave condition.[1]

	Cond. 1 $\omega_p = 2.8$ rad/s $\psi_w = 0$ deg	**Cond. 2** $\omega_p = 2.3$ rad/s $\psi_w = 0$ deg	**Cond. 3** $\omega_p = 2.3$ rad/s $\psi_w = 45$ deg
DOF	STD/Max/Mean	STD/Max/Mean	STD/Max/Mean
Roll	1.46/5.24/1.16	1.89/8.04/1.49	3.74/16.20/2.97
Roll Rate	5.16/15.51/4.09	5.74/17.79/4.51	12.38/50.36/9.78
Pitch	2.82/10.32/2.26	3.82/15.58/3.04	2.30/10.96/2.37
Pitch Rate	9.20/25.76/7.36	10.66/35.42/8.47	8.71/30.94/6.89
Surge	0.11/0.41/0.09	0.15/0.64/0.12	0.10/0.41/0.08
Surge Rate	0.09/0.27/0.07	0.17/0.58/0.14	0.13/0.45/0.10
Sway	0.07/0.24/0.06	0.08/0.36/0.06	0.14/0.72/0.10
Sway Rate	0.05/0.17/0.04	0.09/0.28/0.07	0.15/0.54/0.12
Heave	0.05/0.23/0.04	0.11/0.44/0.08	0.11/0.41/0.09
Heave Rate	0.15/0.42/0.12	0.26/0.93/0.20	0.27/0.94/0.21

[1]Angles and angular rates use units of deg and deg/sec. Translation and translational rates use units of meters and meters/sec.

To give a qualitative feel for the deck motion, a segment of deck heave at wave condition 2 is plotted in Fig. 5.3. Only heave is plotted here because the deck motion was most aggressive in heave for all wave conditions, though significant oscillations in surge and sway were present as well. Referring to Fig. 5.3, two important aspects of the deck motion can be noted. First, the motions are irregular and difficult to predict, which is essential for accurate evaluation of the QP guidance algorithm as it incorporates

deck motion predictions. Also, it should be noted that the deck regularly undergoes periods of significant motion for the scale the tests were conducted at. At times the deck heave oscillated with amplitudes around 35 to 40 centimeters, meaning the deck heaves through a total throw of 70-80 centimeters, and this occurs in a time span of 1 to 2 seconds. Using the Froude scaling metrics, a model-scale throw of 70 centimeters in slightly over 1 second corresponds to a full-scale throw of approximately 10 meters (slightly under 120 percent the UH-60 rotor radius) in about 3.7 seconds.

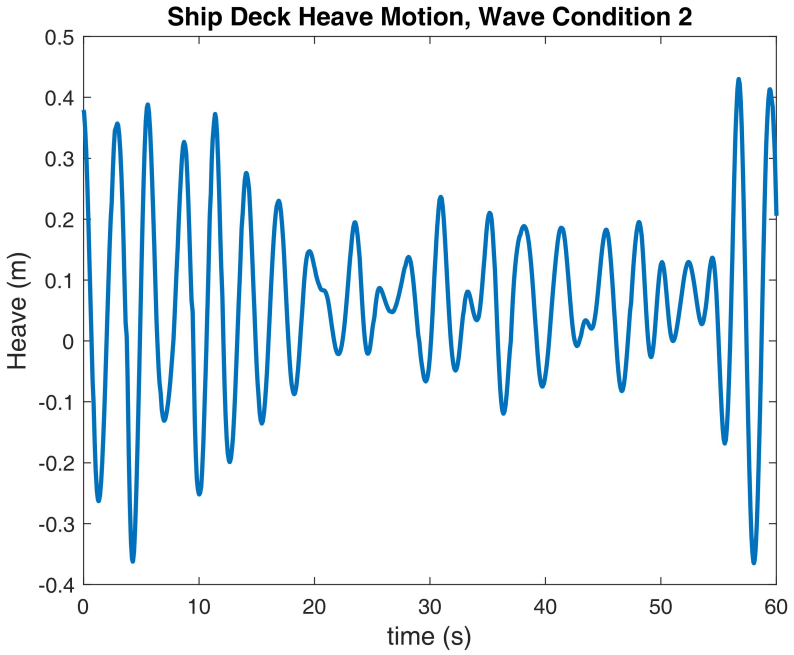


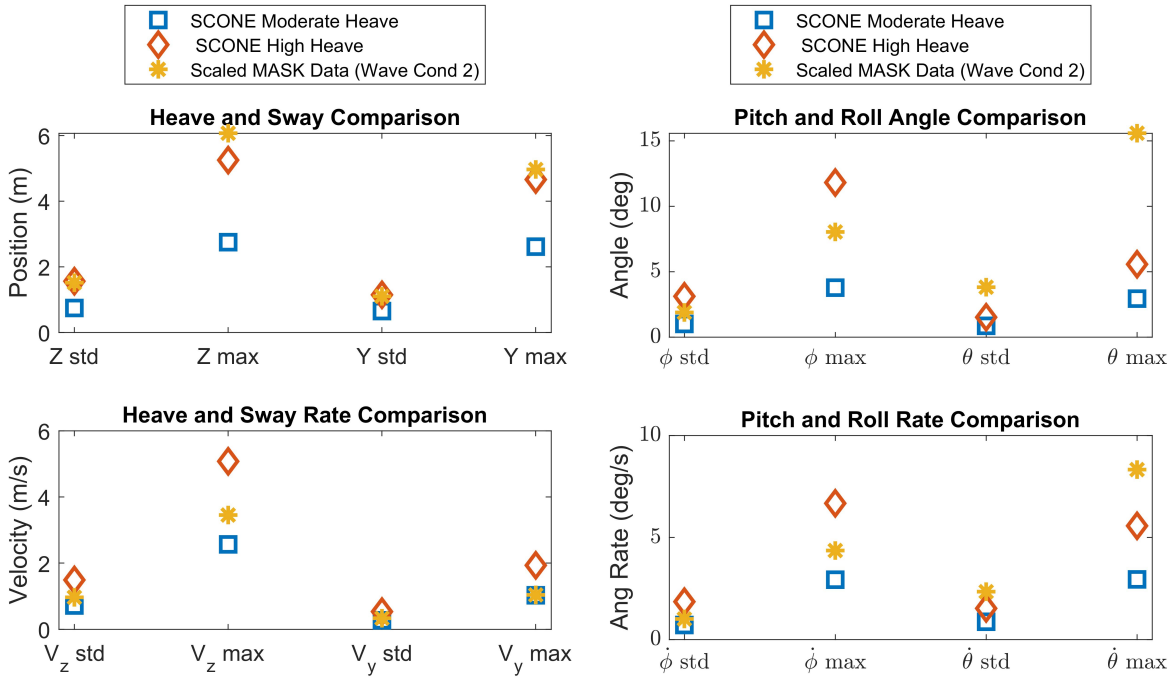
Figure 5.3: Sample deck heave motion time history.

5.1.3 Scaling of Model-Scale Ship Motions

To gain a better understanding of how the ship motion from the experiments performed in the MASK compares to representative full-scale ship motion, the ship data is compared to data released under the Systematic Characterization of the Naval Environment (SCONE) program. The SCONE database includes 6 degree of freedom simulation data of a generic surface combatant representative of a DDG-51 type ship [47]. The time histories are categorized into low, moderate, and high severity cases, which are also identified as exhibiting roll dominant or heave dominant motion.

Plots comparing the scaled ship motions are shown in Fig. 5.4. Comparisons are drawn with the heave dominant moderate and high severity SCONE cases, as the model-

scale ship motion during the MASK experiments was heave dominant. The model-scale data shown in Fig. 5.4 is adjusted to full-scale using the Froude scaling method discussed in Chapter 4.1. Only the model-scale data from wave condition 2 is shown here, as the vast majority of landings were performed in this wave condition. Additionally, ship surge motion was not compared here, as the ship moves forward at a constant velocity in the SCONE simulations. With the model-scale ship tethered in the wave basin, the model-scale ship experiences more significant oscillations in surge as it does not move forward through the waves with a significant velocity.



(a) Heave and sway comparison (b) Pitch and roll comparison

Figure 5.4: Comparison of scaled ship motion from MASK experiments with SCONE simulation data.

Looking at the data plotted in Fig. 5.4a, it is seen that both the scaled standard deviation (std) and maximum (max) displacement of the heave (Z) motion obtained from the MASK ship data is similar that obtained from the high-heave SCONE data. The same is true for the sway (Y) motion. The scaled std and max of the MASK heave rate (V_z) and sway rate (V_y) data compares well with the SCONE moderate-heave data, however. This is related to the fact that the frequency content of the model-scale ship motion was slightly low when compared to that of the SCONE data sets. This can be seen by first noting that ω_p in table 5.1 is approximately equivalent to the frequency at

which the peak in the ship heave spectrum occurs. Froude scaling ω_p for wave condition 2 gives a corresponding full-scale heave spectrum peak occurring at $\omega_p \approx 0.62\ rad/s$, while the SCONE data sets typically exhibit a peak in the heave spectrum at around 0.9 rad/s. The fact that the scaled ship motion from these experiments exhibited slightly low frequency content when compared to full-scale is due to hardware limitations in the wave board actuators at the MASK. Still, the heave and sway displacement and rate data in Fig. 5.4a demonstrates that ship translational motions in the MASK experiments were representative of a scaled moderate to high sea state. The same is true when looking at the std and max of the roll angle and roll rate shown in Fig. 5.4b, where the MASK data falls between the SCONE moderate and high heave data points. The std and max of the pitch and pitch rate data does show more significant pitch motion in the MASK experiments than even the SCONE high-heave data, however. This is again due to the fact that the model-scale ship in the MASK was did not translate forward with a constant velocity, and therefore did not have forward momentum to break the waves. Overall, the comparison given in Fig. 5.4 demonstrates that the MASK ship data is representative of ship motion in scaled moderate to high sea state.

5.1.4 Test Cases

A total of 149 autonomous landings were performed (79 with the QP guidance algorithm and 70 with the baseline algorithm), with 10 different flight controller configurations used. The controller configurations are summarized in table 5.2. At least 10 landings were performed for each combination of control case and wave condition. However, in instances where the UAV battery had enough capacity remaining to perform additional landings before switching test cases, additional landings were recorded.

Table 5.2: Landing Test Cases

Control Case	**Guidance Algorithm**	**Roll and Pitch[1] Tracking BW** (rad/s)	**X and Y Tracking BW** (rad/s)	**Heave Tracking[1] BW** (rad/s)	**X,Y/Z[2] Jerk Limit** (m/s^3)	**Wave Conds. Tested (No. of landings)[3]**
1	QP	High	2.23	High	9/9	1 (10), 2 (12), 3 (11)
2	QP	Med	1.67	High	7/9	2 (11)
3	QP	Low	1.11	High	5/9	2 (12)
4	QP	High	2.23	Med	9/7	2 (10)
5	QP	High	2.23	Low	9/5	2 (12)
6	baseline	High	10	30	NA	1 (10), 3 (10)
7	baseline	High	2.23	30	NA	2 (10)
8	baseline	High	2.23	15	NA	2 (10)
9	baseline	High	2.23	6	NA	2 (10)
10	baseline	High	2.23	High	NA	2 (10), 3 (10)

[1]Refer to table 5.3 for high, medium, and low values.
[2]Jerk limits in QP planner are reported as "X,Y limit" / "Z limit." For example, 7/9 for control case 2 means the X,Y jerk limit was $\pm 7\ m/s^3$ and the heave jerk limit was $\pm 9\ m/s^3$ for this case.
[3]The number in parentheses is the number of landings performed at that wave condition. For example, "2 (12)" for control case 3 means that 12 landings were performed at wave condition 2 for control case 3.

Referring to table 5.2, control cases 1 - 5 used the QP algorithm, but with varied pitch, roll, and heave tracking bandwidths to simulate reduced aircraft mobility. The values corresponding to high, medium, and low cases are reported in table 5.3, which were chosen by Froude scaling realistic values for an aircraft similar to the UH-60. Note that the "hard constraint" formulation of the QP guidance algorithm was used for these tests.

Table 5.3: Tracking bandwidth cases.

	Model Scale Bandwidth			**Full Scale Equivalent**		
	High	**Med**	**Low**	**High**	**Med**	**Low**
$\boldsymbol{\omega_{\theta,cf}}$	11.14	8.36	5.57	3.00	2.25	1.50
$\boldsymbol{\omega_{\phi,cf}}$	11.14	8.36	5.57	3.00	2.25	1.50
$\boldsymbol{\omega_{Z,cf}}$	3.71	1.86	0.74	1.00	0.50	0.20

Note: All entries in units of rad/s

Control cases 6-10 used the baseline algorithm, with pitch and roll tracking bandwidth fixed to the high configuration and heave tracking bandwidth varied. Note that control cases 6-9 used Z bandwidths that were significantly higher than the "High" configuration used for the QP algorithm. For these cases, the bandwidth is reported directly in table 5.2. This was done because the deck relative commands produced by the baseline guidance algorithm were passed directly into the UAV altitude controller command filter, and the UAV lags the deck motion increasingly as command filter frequency is reduced. The baseline controller was therefore first tested with a very high Z bandwidth (30 rad/s) to simulate a case where the aircraft can track the deck with minimal lag. The command filter frequency was then progressively reduced until the lag was too great to perform soft landings. This occurred by the time the Z bandwidth was reduced to the "High" case used for QP landings (i.e., control case 10).

For all control cases, with the exception of control case 6, the X and Y bandwidths were assigned by dividing the pitch and roll bandwidth by 5, ensuring adequate frequency separation between the inner and outer loops. For control case 6 the X and Y bandwidths were set to 10 rad/s. This gives a faster response, but with a higher potential for significant overshoot.

For cases with the QP algorithm, jerk limits were also varied with tracking bandwidth. For example, when heave was in the high bandwidth configuration, the maximum Z axis jerk constraint is set to $\pm 9\ m/s^3$. When heave was in the medium and low bandwidth configurations this was reduced to $j_{z,max} = \pm 7\ m/s^3$ and $j_{z,max} = \pm 5\ m/s^3$, respectively.

The X and Y jerk limits were tied to the pitch and roll bandwidths in the same manner. To simulate the QP algorithm controlling an aircraft with reduced maneuverability some restriction of output constraints is necessary, because if the constraints are too loose the optimal control algorithm can overdrive the command filter input to compensate for the reduced tracking bandwidth. For the UAVs used during testing this would not be problematic as the reduced bandwidth is artificial, but for a full-scale aircraft bandwidth limits are related to real physical limitations such as actuator rate saturations. Here jerk was restricted because it was found to be the most common constraint to approach a limit, and also is related to actuator rate saturation. The acceleration and velocity output constraints were fixed to $V_{max} = \pm 7\ m/s$ and $a_{max} = \pm 3.5 m/s^2$.

For all control cases, the feedback gains for roll, pitch, yaw, and heave were tuned to meet scaled ADS-33E-PRF level 1 handling qualities guidelines for DRB and DRP. The scaled DRBs and DRPs are shown in table 5.4. The gains for roll, pitch, yaw, and heave were fixed for all tests to isolate the effects of varied tracking bandwidths. The X and Y control gains, however, were not fixed. The gain values corresponding to the "X, Y High" case were tuned to just satisfy scaled level 1 DRB and DRP. These gains were used for control cases given in table 5.2 where the pitch and roll axes were in a high tracking bandwidth configuration. For medium and low pitch and roll tracking bandwidth configurations, however, the X and Y feedback gains were reduced to avoid excessively high DRP, reducing DRB. This occurs for degraded pitch and roll bandwidth because the pitch and roll attitude command filters are present in the loop closure when assessing outer loop disturbance rejection. This change in X and Y DRB does not have a large impact here as wind gusts and turbulence were not included in the test setup.

Table 5.4: Disturbance Rejection Properties

	Model Scale DRB (rad/s)	**Full Scale DRB (rad/s)**	**DRP (dB)**
Pitch	2.63	0.71	2.04
Roll	3.73	1.00	2.74
Yaw	3.67	0.99	1.70
Z	1.04	0.28	1.72
***X, Y* High**	0.67	0.18	2.74
***X, Y* med**	0.59	0.16	3.05
***X, Y* Low**	0.42	0.11	3.14

5.1.5 Results

5.1.5.1 Deck Predictions

Prior to assessing the performance of each landing algorithm, discussion of the deck state prediction accuracy is pertinent as the predictions are used directly in the QP guidance algorithm. To give a qualitative feel for the deck prediction accuracy, a representative sample of deck heave predictions produced during flight is given in Fig. 5.5. The plot was produced by taking a sample time segment of deck heave motion and comparing this to what it was predicted the deck heave would be at each time point, with the predictions produced 0.5, 1.5, and 2.5 seconds ago. The result shows that the predictions are very accurate at 0.5 seconds ahead, but the accuracy drops off quickly. When predicting just 1.5 seconds ahead, errors in both magnitude and phase of the predicted heave are significant. By 2.5 seconds ahead, the heave magnitude is regularly under-predicted.

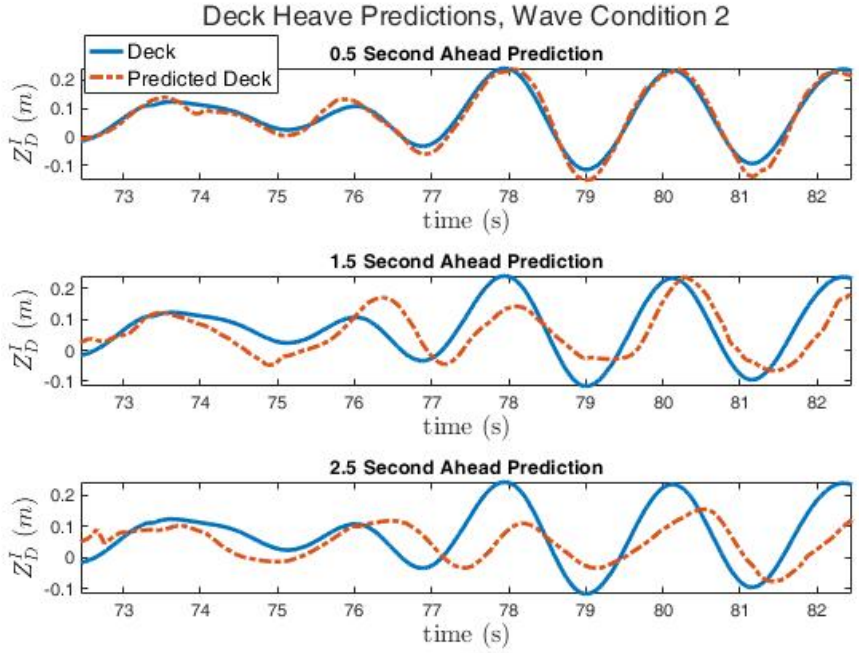


Figure 5.5: Sample deck heave predictions at 0.5, 1.5, and 2.5 seconds into the future.

These observations are supported by the prediction accuracy statistics calculated across trials shown in Figs. 5.7 and 5.8. The errors for a single "trial" were computed as shown in Fig. 5.6, described as follows:

1. A time point in a given landing flight test time history was chosen.
2. At the chosen time point, predictions of future deck motion spanning 0.1 to 3

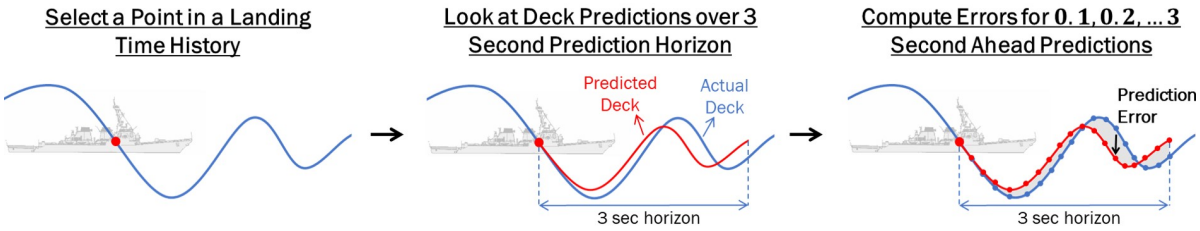


Figure 5.6: Process for calculating "trial" error vector used to compute prediction error statistics shown in Figs. 5.7 and 5.8.

seconds into the future were made during testing. These predictions were saved.

3. The saved predictions were compared to the actual deck motion at 0.1, 0.2, ... 3 seconds ahead of the time when the predictions were produced, generating a "trial" error vector.

The time points chosen to evaluate the predictions were spaced out by at least 3 seconds in all cases. This ensured that directly adjacent time points were not used when calculating prediction errors, providing trials taken at a more diverse set of conditions. The number of trials used to calculate the prediction error statistics was increased until the mean and standard deviation converged (here this occurred at 217 trials).

Looking at the wave condition 2 prediction errors shown in Fig. 5.7, the light gray lines show the error vectors computed for each trial and the darker lines show how the mean and standard deviation of the prediction errors vary with how far we predict into the future. From this plot it can be seen that the X and Y predictions start converging toward the true deck value at approximately 1.5 seconds out (as can be seen by the prediction error mean and standard deviation beginning to trend toward zero) and the heave predictions begin to converge toward the true deck heave between 1 and 1.5 seconds out. On average, the long term position prediction errors are slightly under the average deviation of deck position from its mean location. For example, the average magnitude of heave displacement from the mean deck altitude (reported in table 5.1) is about 9 cm, while the average heave prediction error at 2.5 seconds is about 6 cm. In many cases, though, the prediction errors at 1-3 seconds out are more than double this value indicating that predictions are only reliable in the very short term. The attitude errors also follow a similar trend, decreasing in the final second of the prediction horizon. The same is true for the velocity predictions, but the velocity predictions also exhibit a number of sharp spikes in prediction error, which sometimes occur with under 1 second remaining in the prediction horizon. This is related to occasional data buffering that occurred when sending deck measurements from the ground station to the UAV over wifi.

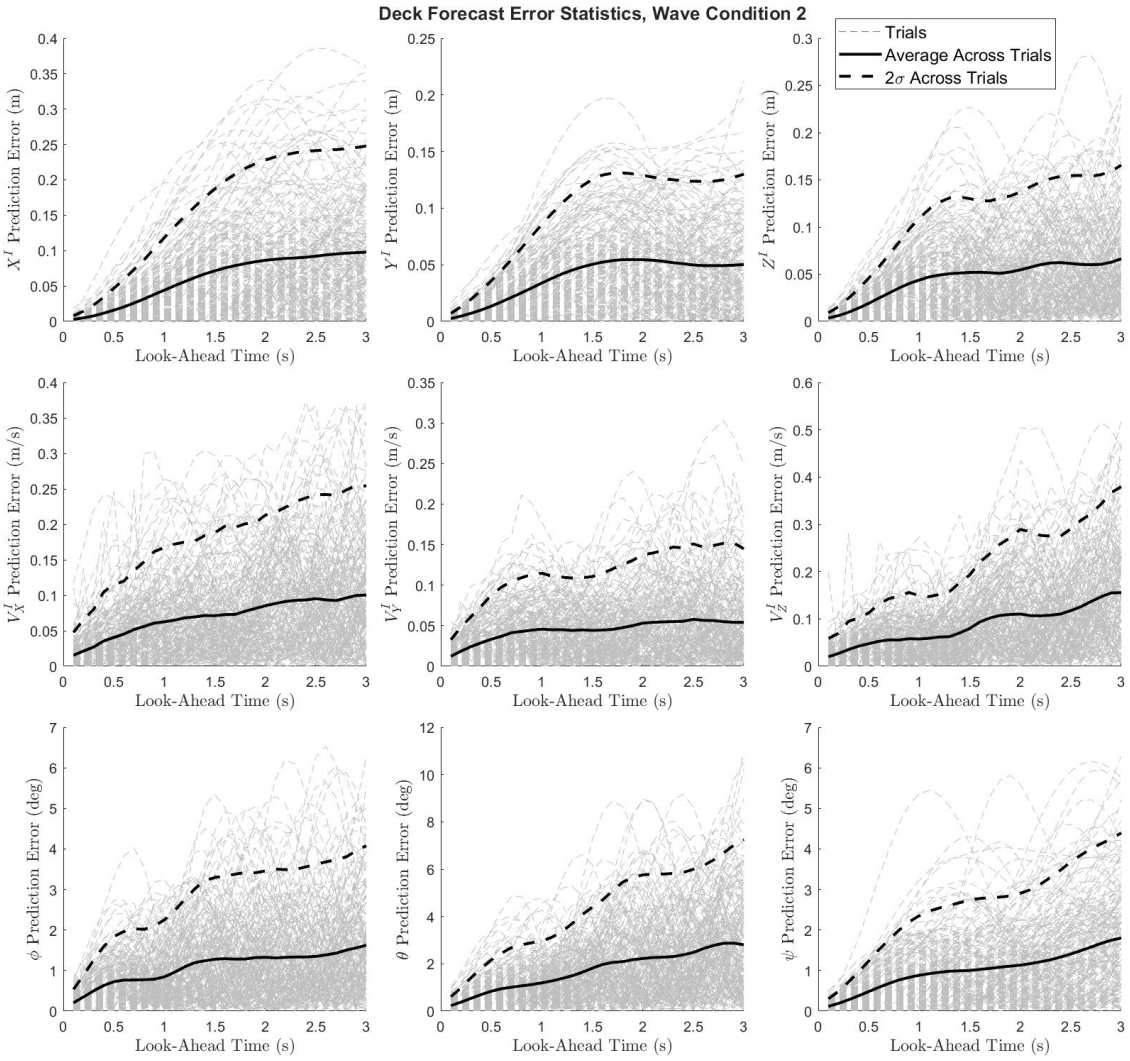


Figure 5.7: AR model deck state prediction error statistics vs look-ahead time for wave condition 2.

When this occurs, small step-like inputs are passed through the filters used to estimate deck velocity, resulting in spikes in the velocity estimates. Since the AR model calculates future outputs based on past outputs, this can result in sharp spikes appearing in the predictions. Despite these issues, it will be shown that QP algorithm was able to perform well with the quality of deck predictions obtained here.

It should also be noted that the prediction quality is dependent on both the frequency and magnitude of deck motion. For example, as shown when considering the scaling of the prediction algorithms in Chapter 4.2, the model-scale predictions beginning to converge toward the true deck states at 1.5 seconds ahead would Froude scale to full-scale predictions beginning to converge at slightly over 5 seconds ahead. Further, the effect

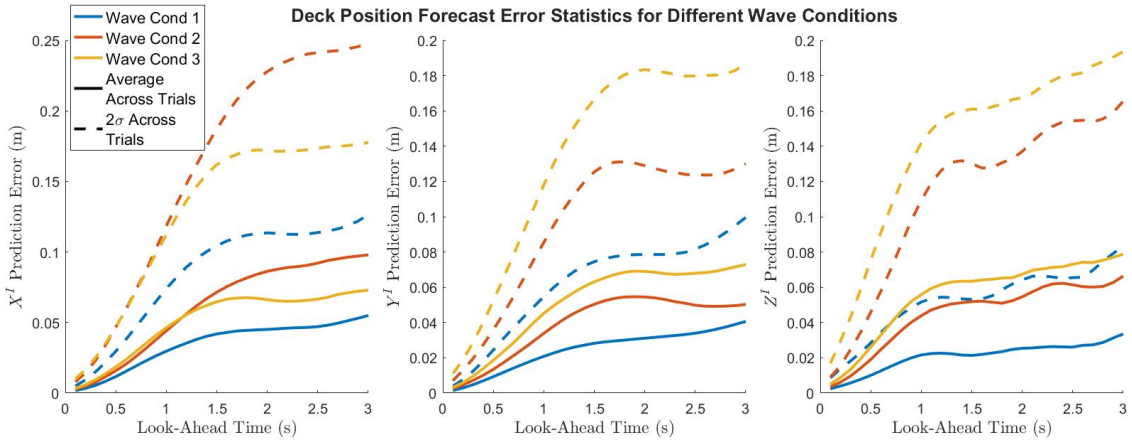


Figure 5.8: Position prediction errors vs look-ahead time for each wave condition.

of deck motion aggressiveness on prediction performance can be seen from the Fig. 5.8, where position prediction errors at wave condition 1 (the most mild wave condition) are notably lower than the errors calculated at wave conditions 2 and 3. The deck Y (sway) prediction errors are also highest at wave condition 3 because this was the condition with the highest amplitude sway oscillations (see table 5.1). Likewise, the X (surge) prediction errors are highest for wave condition 2, which was the wave condition with the highest amplitude oscillations in surge.

5.1.5.2 Sample Landing Time Histories

Sample landings performed at wave condition 2 with the baseline and QP guidance algorithms are shown in Figs. 5.9 and 5.10. In both cases, a successful landing was completed with low terminal deck-relative positions and velocities.

For the baseline landing case, the relative descent rate at touchdown was very close to the 0.25 m/s descent rate bias commanded in the baseline guidance algorithm. This is not surprising as the Z bandwidth for control case 7 was high and the UAV could effectively track the deck. The baseline algorithm also matched X and Y velocity well in this instance, though in other flight tests more significant X velocity errors were observed as the X and Y command filter frequency for control case 7 was low enough to introduce significant lag in X and Y translation. This will be discussed further in section 5.1.5.3. Also, note that the sharp changes in attitude that occur shortly after deck contact are due to a combination of rigid landing gear on the UAV and the throttle being cut several centimeters above the deck.

For the QP landing case, similar relative velocities were obtained at landing but deck

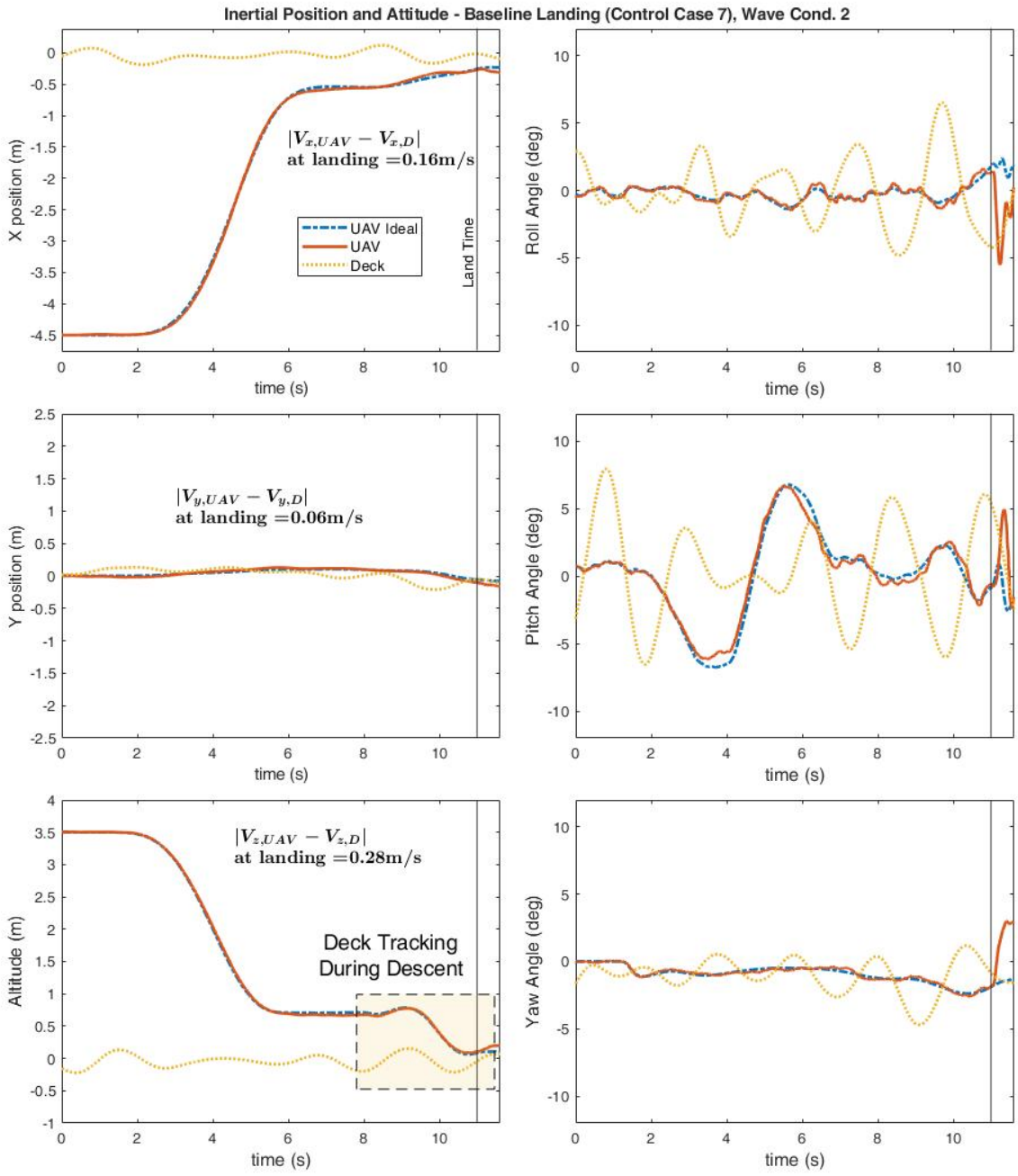


Figure 5.9: Sample landing time history with baseline guidance algorithm.

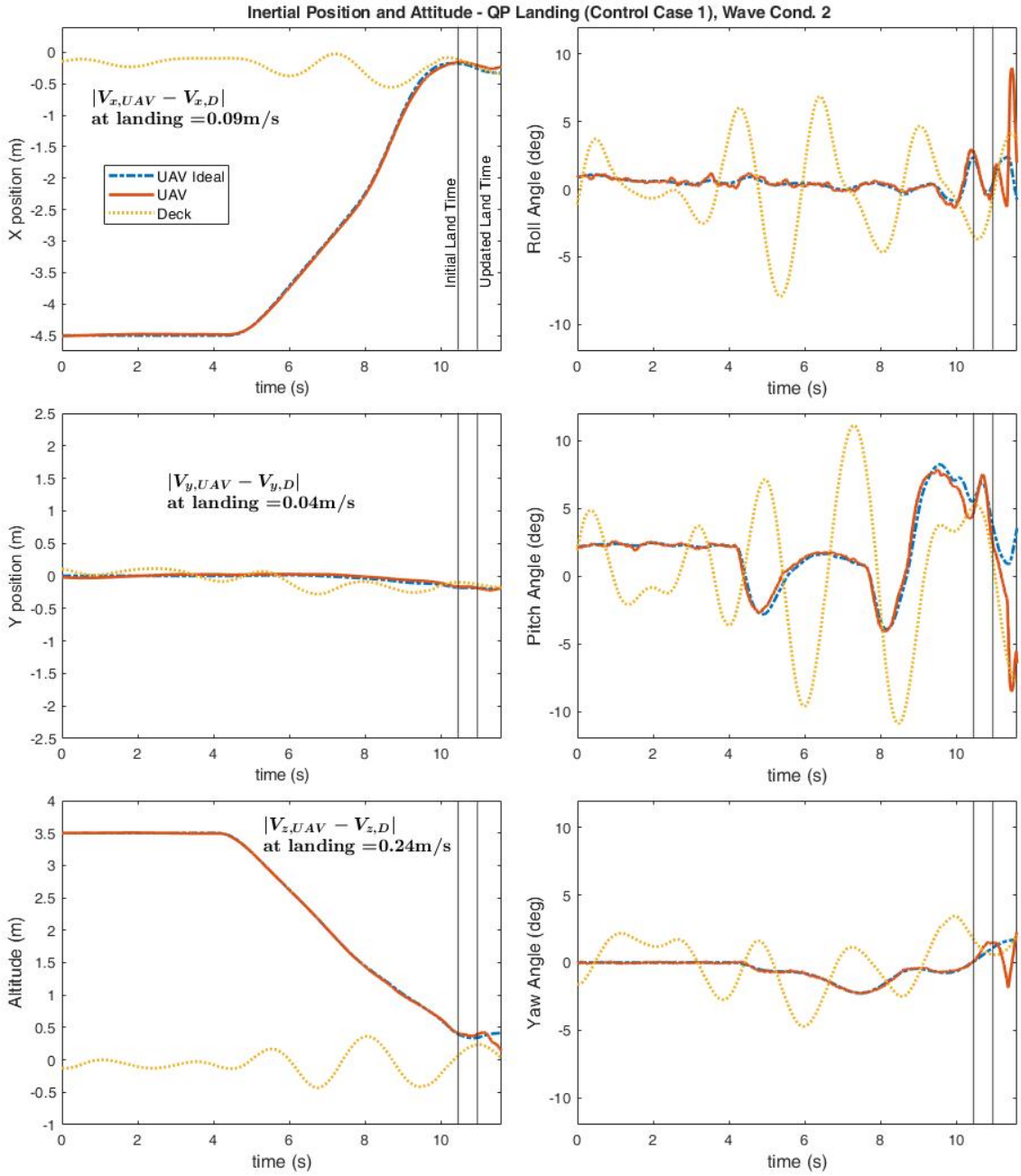


Figure 5.10: Sample landing time history with QP guidance algorithm.

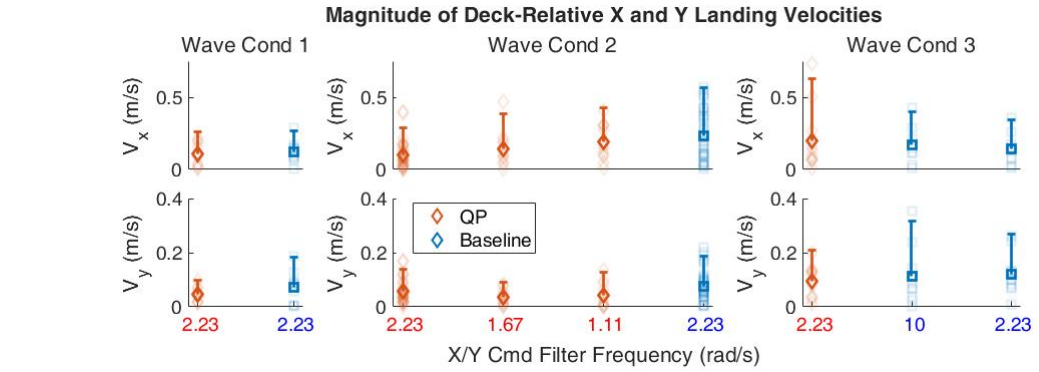
oscillations were largely ignored. This can be seen by examining the altitude time history plotted for the sample QP landing case shown in Fig. 5.10, where the UAV descends at a nearly constant velocity without needing to explicitly follow deck heave oscillations. This is in contrast to the baseline landing case shown in Fig. 5.9, where it is observed that the UAV follows the deck heave oscillations for approximately the last 3 seconds of the

landing. The results illustrate that the QP algorithm can plan a more direct landing path, though the baseline "deck tracking" method is simple and effective when the aircraft has the required control authority. Also, for the case shown in Fig. 5.10, the land-time update scheme included in the QP algorithm shifted the land time from a point with significant upward heave to a point near a peak in the oscillations, preventing the UAV from needing to change direction to match heave motion at landing. It will be shown in the following subsections, however, that the land time update harmed performance in many other cases. The QP algorithm also manages to match deck attitude in this case, though deck attitude was small at landing. The capability of the landing algorithms to consistently match position, velocity, and attitude will be clarified through the statistics presented in the following subsections.

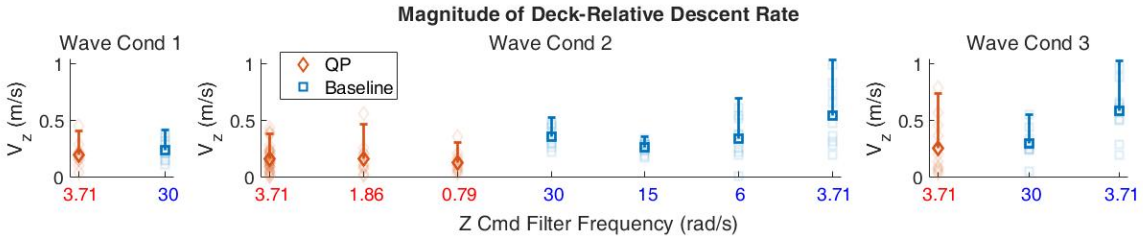
5.1.5.3 Position, Velocity, and Attitude Landing Errors

The deck-relative landing velocities are shown for all flight tests in Fig. 5.11. Note that Fig. 5.11a shows the X and Y landing velocity errors plotted versus the X/Y command filter frequency used for each test, and Fig. 5.11b plots the Z velocity errors as a function of Z command filter frequency. First focusing on the deck-relative descent rates reported for the baseline algorithm in Fig. 5.11b, the results show the baseline "deck tracking" guidance method to provide low relative descent rates when the control law was configured to provide a very high Z bandwidth. For example, with $\omega_{cf,Z} = 30\ rad/s$ at wave condition 1 and $\omega_{cf,Z} = 15\ rad/s$ at wave condition 2, deck-relative descent rates of under 0.4 m/s were obtained with averages close to the deck-relative descent rate of 0.25 m/s commanded by the baseline guidance algorithm. With $\omega_{cf,Z} = 30\ rad/s$ at wave condition 2 the results are similar, though two landings did end with relative descent rates of around 0.45 m/s. While testing, however, landings that were no longer visibly perceived as "soft" did not occur until relative descent rates of around 0.6 m/s. One such landing occurred with $\omega_{cf,Z} = 30\ rad/s$ in wave condition 3 (the most aggressive wave condition), though the other landings at this test condition had impact velocities of around 0.4 m/s or less.

The results reported in Fig. 5.11b also demonstrate the sensitivity of the baseline algorithm to reduced bandwidth. For example, moving from $\omega_{cf,Z} = 15\ rad/s$ to $\omega_{cf,Z} = 6\ rad/s$ at wave condition 2, a noticeable increase in the mean and variance of the Z velocity landing errors begins to emerge due to the effective command tracking delay added by the reduced bandwidth. When the bandwidth is further reduced to $\omega_{cf,Z} = 3.71\ rad/s$ at wave condition 2 the trend continues, resulting in excessively high



(a) Deck-relative X and Y landing velocities.



(b) Deck-relative Z landing velocities.

Figure 5.11: Deck-relative landing velocities vs. command filter frequency. Faded markers represent individual landings, bold markers show the average, and the error bar shows 2σ of the velocity error magnitude.

impact velocities. The same behavior is seen when moving from $\omega_{cf,Z} = 30\ rad/s$ to $\omega_{cf,Z} = 3.71\ rad/s$ with the baseline guidance algorithm at wave condition 3.

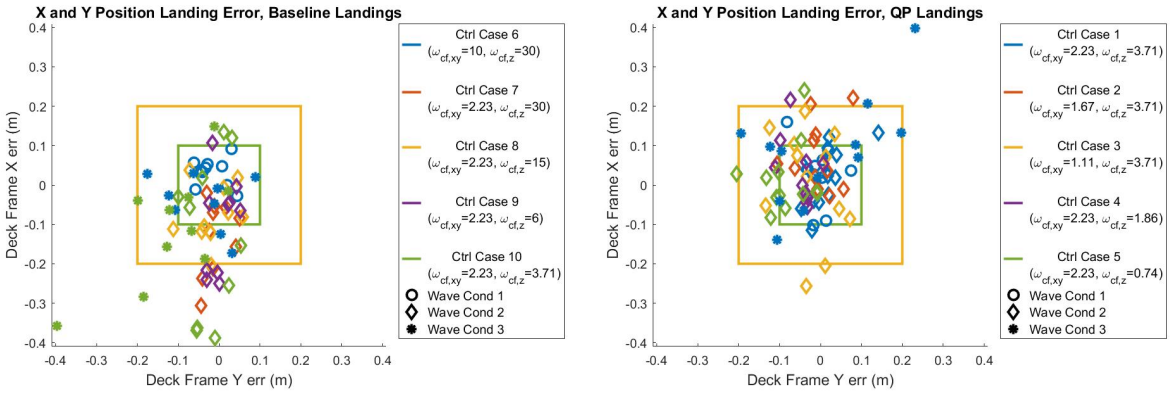
Relating these results back to full-scale, a heave tracking bandwidth of 6 rad/s at model-scale corresponds to 1.6 rad/s at full-scale. The specifications for height response characteristics in ADS-33E-PRF (Ref. 66) require a vertical axis time constant of 5 seconds or less for level 1 handling qualities. This roughly corresponds to a full-scale vertical axis bandwidth of 0.2 seconds. The results obtained here therefore suggest that the vertical axis tracking bandwidth would need to significantly exceed the level 1 requirement to consistently perform soft landings with the baseline "deck tracking" landing strategy in high sea states. Such a high bandwidth may be difficult for large or heavily loaded rotorcraft to attain.

The QP guidance algorithm, on the other hand, proved to be insensitive to reduced Z bandwidth. For example, looking at the results shown in Fig. 5.11b for $\omega_{cf,Z} = 3.71\ rad/s$ at wave condition 2, low deck-relative descent rates are obtained with the QP algorithm while the baseline algorithm performed poorly with the same Z bandwidth. Further, as $\omega_{cf,Z}$ is reduced down to $\omega_{cf,Z} = 0.79\ rad/s$, the QP algorithm still consistently obtained low deck-relative descent rates on average. Looking at Fig. 5.11a, it is observed that

the Y velocity errors also did not increase with the QP algorithm when reducing the X and Y command filter frequency from 2.23 rad/s to 1.11 rad/s at wave condition 2. The reduced X and Y bandwidth did result in an increase in the average X velocity errors, however, but the increase is slight and low relative velocities were still obtained.

A similar trend can be observed in the X position errors shown for the QP algorithm in Fig. 5.12b. Comparing the X position landing errors across all controller configurations tested with the QP algorithm at wave condition 2, the largest spread in the X position error is observed for control case 3 (the case with the lowest X/Y bandwidth). The difference here is again small, though, with the majority of QP X position landing errors grouped below 20 cm regardless of controller configuration. This is in contrast to the position errors reported for the baseline algorithm in Fig. 5.12a, where the position landing errors are clearly lowest for the case with the highest X and Y bandwidth (control case 6, $\omega_{cf,xy} = 10\ rad/s$). This occurs despite the lack of frequency separation between the inner and outer loops for control case 6 ($\omega_{cf,xy} = 10\ rad/s$ and $\omega_{cf,\theta\phi} = 11.14\ rad/s$), as the frequency of deck motion was low enough to avoid significant overshoot resulting from commanding the UAV to track the deck. If roll and pitch bandwidths are not high, however, simply using a high outer loop command filter frequency may not be a tractable solution. Using a command filter frequency that ensures adequate frequency separation may not provide a fast enough response, though. This is seen by the inflated position landing errors for control cases 7-10, where $\omega_{cf,xy}$ was reduced to $2.23\ rad/s$. For control case 10 position landing errors were highest, as using a Z command filter frequency of $\omega_{cf,Z} = 3.71\ rad/s$ with the baseline algorithm resulted in the UAV and deck altitudes being significantly out of phase during the descent. This caused the UAV to contact the deck prior to the expected land time, further inflating landing errors.

Aside from being more robust to reduced command tracking bandwidth, it should also be noted that the QP algorithm did result in lower average relative landing velocities than the baseline algorithm in nearly every case. For baseline landing cases with $\omega_{cf,Z} = 30\ rad/s$ and $\omega_{cf,Z} = 15\ rad/s$, though, the relative descent rate would likely have decreased at the expense of an increased duration of descent if the 0.25 m/s Z velocity bias included in the baseline algorithm was reduced. Also, it should not be overlooked that the QP algorithm did not perform well in several outlying cases. Looking at the relative landing velocities shown in Fig. 5.11 for the QP algorithm at wave condition 3, one landing terminated with a X velocity error of close to 0.75 m/s and another with a Z velocity error of about 0.75 m/s. One of these landings also terminated with a very large position error (the largest shown in Fig. 5.12b). The QP algorithm



(a) Baseline guidance position errors. (b) QP guidance position errors.

Figure 5.12: Deck-relative X and Y position at landing. Command filter frequencies for each control case are reported in the legend in units of rad/s.

also performed poorly in terms of attitude matching at touchdown. This is shown by the results plotted in Fig. 5.13, where the QP landings terminated with deck-relative pitch attitudes of close to 15 degrees in several instances. The pitch attitude errors were often greater than those obtained with the baseline algorithm, despite the fact that the QP algorithm included some motivation for attitude matching in the cost function (albeit through a simplified relationship between attitude and acceleration). It will be shown in the Chapter 5.1.5.5, however, that the vast majority of cases where the QP algorithm performed poorly likely would have been averted if the land time update method (discussed in Chapter 3.2.3.5) did not allow any reduction of the landing duration.

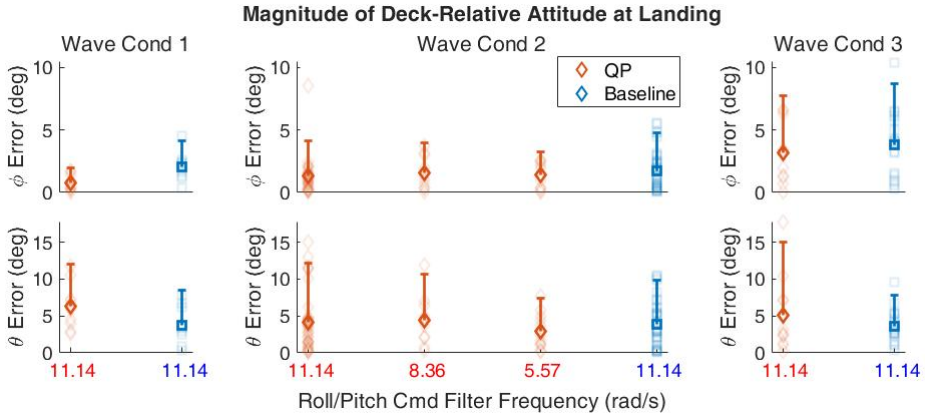


Figure 5.13: Deck relative landing attitude. The faded markers represent individual landings, the bold markers show the average value, and the error bar represents 2σ of deck-relative landing attitudes.

Despite the poor attitude matching performance, these results illustrate one of the major advantages of using a guidance algorithm like the QP method presented here. That is, integrating deck motion predictions and an optimal control solver allow the guidance algorithm to plan ahead for the aircraft dynamics and expected deck motions. Here, this allowed the QP algorithm to adequately match deck positions and velocities at lower bandwidths than the baseline landing algorithm. With a sequential loop closure architecture used to control X and Y translation, this in turn allows the outer loop controllers to be designed assuming adequate frequency separation. With the baseline algorithm, on the other hand, the outer loop command filter frequency was set to balance a trade-off between speed of response and overshoot due to lack of separation from the inner loop.

The QP algorithm also has the potential to plan ahead for more restrictive output constraints. For the QP algorithm, this is different than planning for reduced bandwidth. The highly weighted terminal cost used to motivate matching deck state at touchdown can outweigh the cost placed on acceleration and jerk. Without high enough penalties or constraints included on acceleration or jerk, the optimal control solution may therefore increase the magnitude of control inputs to compensate for reduced bandwidth. Here, the jerk penalty and constraints included in the QP algorithm were set to prevent the algorithm from overcompensating for reduced bandwidths. Additionally, tests with further restricted jerk constraints were included to study the ability of the QP algorithm to plan ahead for reduced output constraints. The hypothesis here was that, with more stringent output constraints included, the optimization algorithm may need to plan the landing path with the control action spread out over a larger time range, which also requires that deck motion predictions converge with enough lead time to avoid a rapidly changing desired terminal state at the end of the trajectory. The results showed the reduced jerk constraints to have little impact on performance, but the jerk constraints may simply have not been restricted enough to have a large impact. To better understand the extent to which the QP algorithm is capable of planning ahead for more stringent output constraints, additional testing with further restricted acceleration and jerk limits would need to be conducted.

5.1.5.4 Accelerations During Landing

Another hypothesized advantage of the QP algorithm was that accelerations during landing could be reduced, as deck motions do not need to be tracked. This proved not to be the case, however. Referring to the results shown in Fig. 5.14, the baseline

algorithm did result in large Z accelerations around 2-2.5 seconds before landing, but this is an artifact of the baseline guidance logic. It was around this time in the landing sequence where the baseline algorithm would switch from tracking low pass filtered deck motion to tracking actual deck motion while descending. This introduces a small step input, resulting in sharp throttle commands and more aggressive accelerations. This behavior could have been averted by phasing in the switch between the approach and landing stages of the baseline algorithm, but this does illustrate an advantage of the QP algorithm: the QP algorithm results in smoothly regulated accelerations without the need to manage transitions between flight modes.

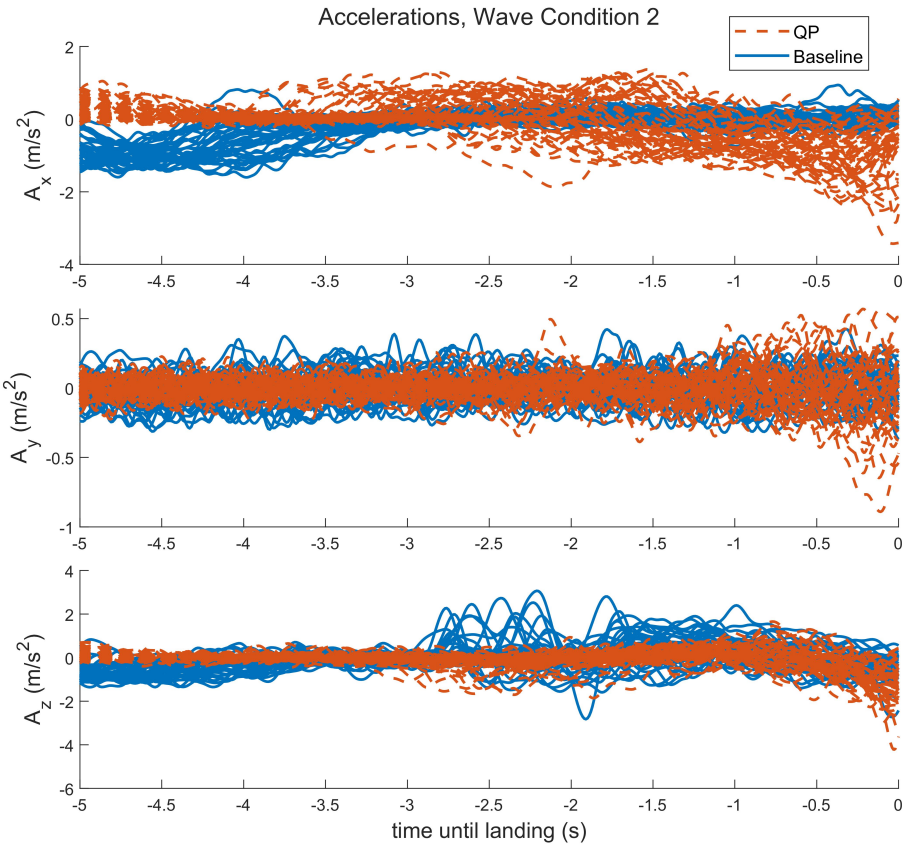


Figure 5.14: Accelerations during the last 5 seconds of landing for flights conducted at wave condition 2.

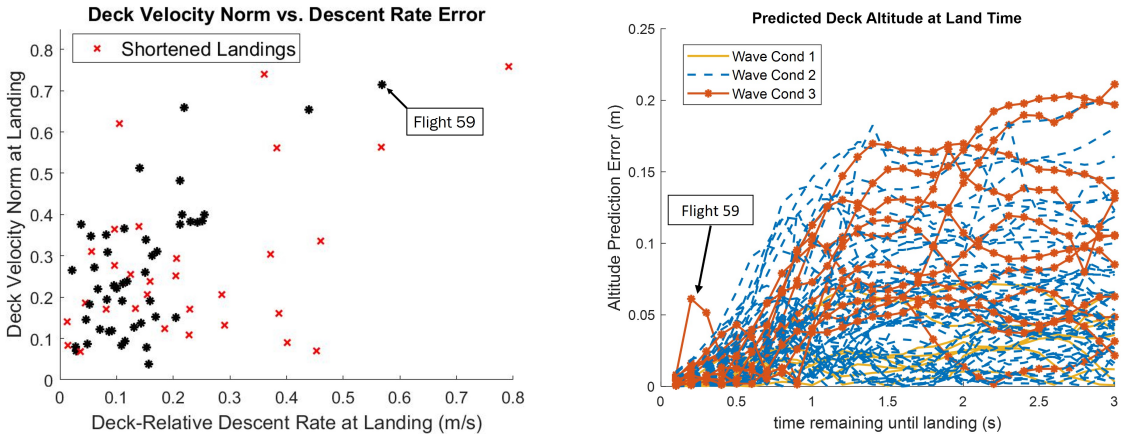
Looking at later time points in the landing sequence, the baseline and QP algorithms resulted in comparable Z acceleration magnitudes, though the baseline landing accelerations are defined by those of the deck and more aggressive deck motion may change this result. The QP guidance algorithm also generally resulted in higher X accelerations, but this was due to the X command filter frequency resulting in heavily filtered deck-relative

commands for all baseline tests conducted at wave condition 2. It should also be noted that the QP accelerations remained within the specified 3.5 m/s^2 constraint for all cases but one. This one instance was due to the landing gear contacting the deck slightly harder than usual at touchdown, resulting in a spike in the acceleration in the final 0.2 seconds of the plotted trajectory.

5.1.5.5 Factors Affecting QP Algorithm Performance

To understand the robustness of the QP algorithm, it is necessary to understand why the QP algorithm performed poorly in some instances. Here it was found that there were three contributing factors: the quality of deck motion predictions, the aggressiveness of deck motion at landing, and the land time update scheme included in the QP algorithm.

The individual impact of each of these factors is clarified by studying the plots shown in Fig. 5.15. Fig. 5.15a plots the velocity norm at landing against the relative-descent rate at touchdown for all QP landings. As the deck motion was dominated by translational oscillations, the velocity norm is representative of the aggressiveness of deck motion. This plot shows that the cases with the three highest impact velocities all landed at times where deck velocity norm is large, which is an expected result.



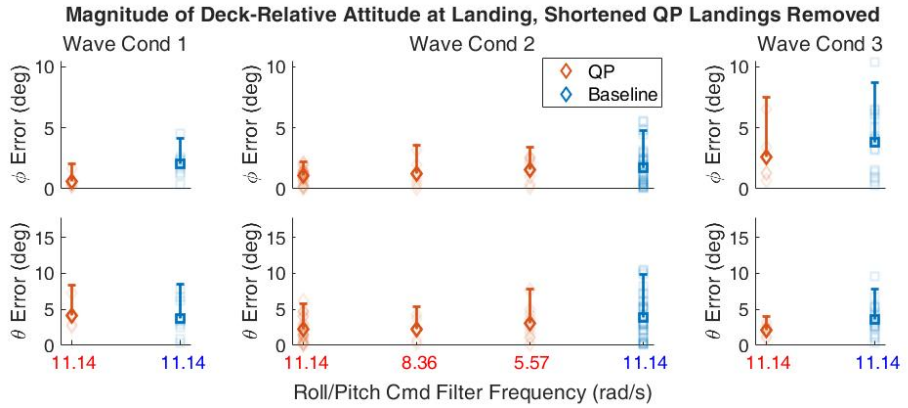
(a) (b)

Figure 5.15: (a) Deck velocity norm vs. deck-relative descent rate at landing with shortened QP flights denoted by the red “x". (b) Predicted deck altitude at landing during final 3 seconds of all QP flights.

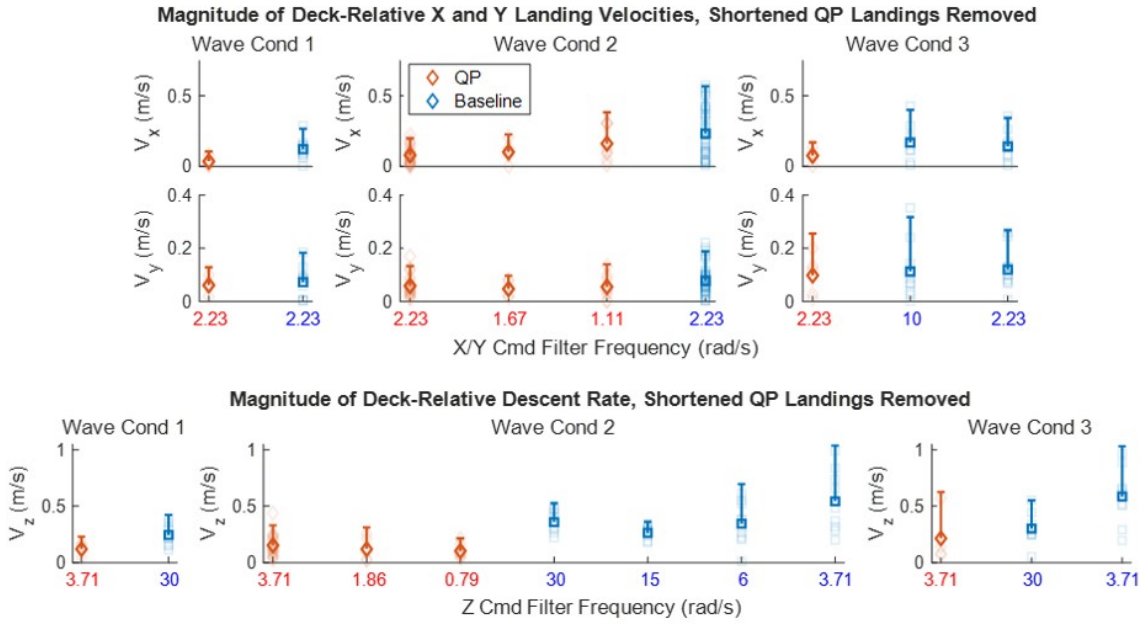
More interestingly, Fig. 5.15a also reveals the impact of the land time update scheme. The markers plotted as a red “x” in Fig. 5.15a represent cases where the land time update scheme elected to shorten the landing by a slight amount (the target land time was not

reduced by more than 0.3 seconds in any case), and the black "*" markers represent all other QP landings. The cases with shortened landing duration account for every landing where the deck velocity norm was under 0.6 m/s but the relative descent rate was over about 0.3 m/s. In addition, only one landing with a relative descent rate of greater than 0.5 m/s did not have a shortened landing duration. This landing is marked by "Flight 59" on Fig. 5.15a and, by comparing to Fig. 5.15b, it is seen that abnormally poor deck heave predictions occurred during the final 0.5 seconds of the landing. Further, the deck was moving aggressively at touchdown for this case. Also, it should be noted that while cases with a shortened maneuver duration account for slightly over 50 percent of the QP landing cases, these landings were not disproportionately biased toward cases with high deck velocity norms at landing (in other words, shortened landings did not happen to coincide with unfavorable deck motions on average). Landings with shortened duration were biased toward cases with high impact velocities, however.

The land time update scheme also negatively impacted QP algorithm attitude errors at touchdown. This can be observed by comparing Fig. 5.16a and Fig. 5.13. In Fig. 5.16a the shortened QP flights are removed from the plot, which removes all QP landings with a pitch attitude error greater than about 7.5 degrees. The reason for this is that the slightly shortened landing duration requires a more aggressive trajectory to be planned, resulting in the UAV flaring pitch attitude at the end of the trajectory to match deck surge velocity. The flared pitch attitudes often resulted in the rear landing gear contacting the deck, increasing the relative pitch attitude at touchdown. This also inflates the velocity errors at touchdown, which is clearly seen by comparing Fig. 5.16b and Fig. 5.11. In Fig. 5.16b the data points corresponding to the shortened QP landings are removed from the plot, which removes the data points with the largest X and Z velocity errors. These results are interesting, as a major point of concern prior to testing was the fidelity of the deck motion predictions. In the vast majority of cases, however, poor deck motion predictions proved not to be the culprit of poor performance and performance likely would have improved by simply not allowing the trajectory duration to be reduced during the descent.



(a) Deck-relative landing attitude.



(b) Deck-relative landing velocities.

Figure 5.16: Deck-relative landing velocity and attitude with shortened QP flights removed. Faded markers represent individual landings, bold markers show the average, and the error bar shows 2σ of the error magnitude.

5.2 Autonomous Landing Experiments at the Penn State Indoor Flight Facility

5.2.1 Experimental Setup

Additional landing flight tests were conducted with the quadcopter UAV in the Penn State Indoor Flight Facility, as seen in Fig. 5.17. The Indoor Flight Facility consists of a

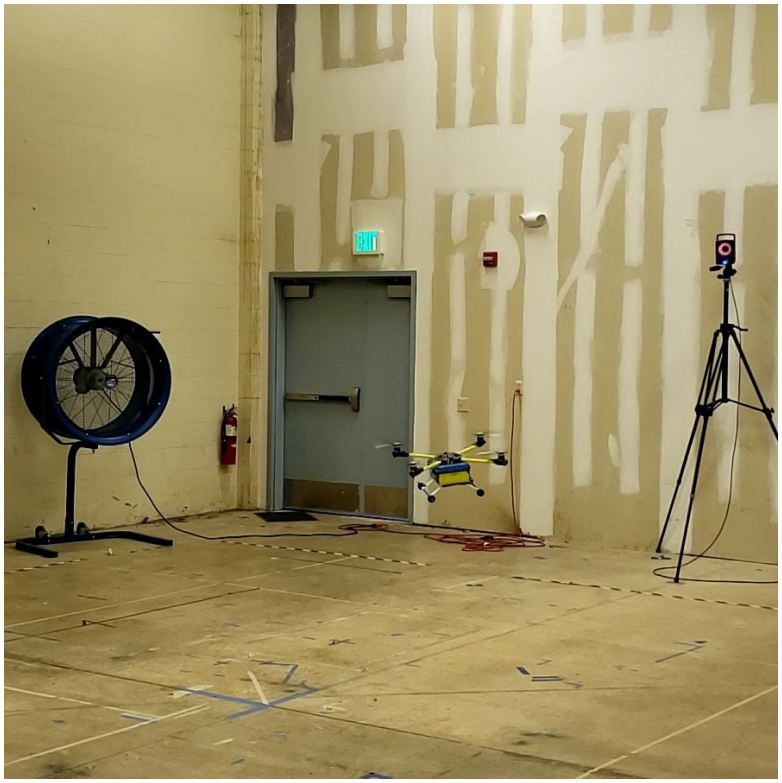


Figure 5.17: Quadcopter UAV flying in the Penn State Indoor Flight Facility.

25' x 25' x 15' space equipped with a 12 camera Vicon motion capture system. A landing platform capable of emulating ship motion was not available for use in this facility, so the flight tests were performed to a "virtual ship deck." The deck is referred to as virtual because time histories of deck motion recorded during the experiments performed at the Maneuvering and Seakeeping Basin (MASK) were simply forwarded to the drone, providing the landing algorithms with a "measured deck location" despite the lack of physical hardware. An industrial fan was added to the test setup to provide aerodynamic disturbances.

For the flight tests conducted in the Penn State Indoor Flight Facility, data was transferred between the motion capture system, UAV onboard computer, and UAV flight controller in a manner similar to what is shown for the tests conducted in the MASK facility in Fig. 5.2. The UAV WIFI link is still used to forward motion capture measurements of aircraft position and attitude from the ground station to the Odroid XU4 onboard computer. Additionally, ship motion data is still sent from the ground station to the UAV, but here the ship motion "measurements" are produced from ship data recorded at the MASK facility rather than measurements of a physical platform. During these tests the motion capture measurements and recorded ship motion time

histories were streamed to the UAV at rates of 100 Hz and 50 Hz, respectively.

5.2.2 Test Cases

The tests conducted in the MASK facility did not incorporate aerodynamic disturbances, which can have a significant impact on performance in a practical landing scenario. The purpose of these experiments was therefore to determine if the QP guidance algorithm offers any advantage over the baseline method when operating in the presence of a significant aerodynamic disturbance.

For this purpose, flight tests were performed with an industrial fan placed 15 feet from the mean location of the ship deck. The fan was located so that the wind approached the bow of the virtual ship at a 30 degree angle to port (as the UAV approached the deck, the gust would blow the aircraft backward and to the right). The height of the fan was set so that the aerodynamic disturbance would extend to approximately 1 meter above the mean height of the ship deck, so that the UAV was not affected by the gust until the final stages of the landing. Measured at a distance of 15 feet, the flow produced by the fan has a mean wind speed of approximately 6 m/s. The circulation of the fan blades combined with the rotor wake also introduced significant turbulence, though the degree of turbulence is difficult to quantify. Based on the Froude scaling method discussed in section 4.1, the mean wind speed of 6 m/s corresponds to a full-scale wind speed of approximately 40 kts, but it should be noted that the model-scale UAV was not designed to exhibit a response to aerodynamic disturbances similar to that of a UH-60. Still, this is a significant disturbance for the model-scale aircraft.

Using this setup, a total of 30 landings were performed: 15 with the QP guidance algorithm and 15 with the baseline algorithm. Here the "soft constraint" formulation of the QP guidance method was used, allowing the QP algorithm to re-plan the trajectory at each time step with the current state estimate used as the initial condition. In addition, the method included in the QP algorithm for selecting the targeted land time was altered to not allow any reduction of the landing maneuver duration. This adjustment was made based on the results presented in Chapter 5.1.5.5, where it was found that even slight reductions in the landing maneuver duration often had a negative impact on performance.

The EMF controller reference tracking parameters were fixed to high bandwidth configurations. More specifically, when using the QP guidance algorithm the command filter frequencies listed for control case 1 in the experiments performed at the MASK (see table 5.2) were used. When using the baseline guidance algorithm, the command filter frequencies listed for control case 6 in table 5.2 were used, but the heave tracking

bandwidth was reduced from 30 rad/s to 15 rad/s as this was found in the MASK experiments to be sufficiently high for performing landings with the baseline guidance method. The feedback gains were also fixed to give the DRB values listed in table 5.5, which were determined by Froude scaling level 1 handling qualities guidelines for full-sized rotorcraft.

Table 5.5: Quadcopter Disturbance Rejection Properties

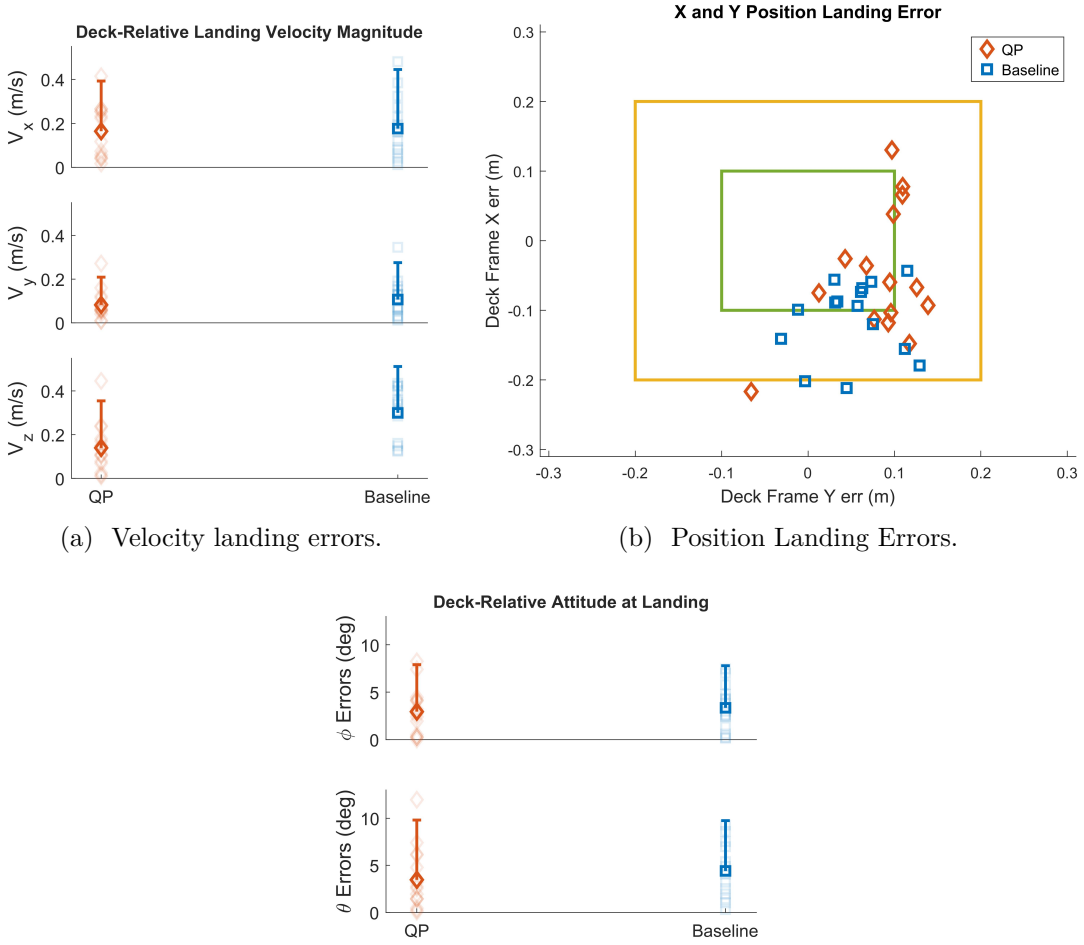
	Model Scale DRB (rad/s)	**Full Scale DRB (rad/s)**	**DRP (dB)**
Pitch	2.70	0.73	3.29
Roll	3.69	1.00	4.50
Yaw	3.18	0.85	3.72
$\boldsymbol{Z}$	1.02	0.27	2.36
$\boldsymbol{X, Y}$	0.67	0.18	2.92

The ship motion time histories used to drive the virtual deck were obtained from data recorded at wave condition 2 used in the MASK experiments (see Chapter 5.1.2). Fifteen segments of ship data were selected randomly in order to provide a range of wave conditions. The same 15 ship motion time histories were used for flight tests performed with the QP and baseline guidance algorithms.

5.2.3 Results

The velocity, position, and attitude landing errors are shown in Fig. 5.18. Overall, with feedback gains tuned to provide scaled DRBs meeting the guidelines for level 1 handling qualities and the baseline guidance algorithm in a high tracking bandwidth configuration, both the QP and baseline guidance methods were able to perform successful landings in the presence of the aerodynamic disturbance. The landing error statistics shown in Fig. 5.18 do not, however, indicate that the QP algorithm offers any definitive advantage over the baseline algorithm in terms of compensating for aerodynamic disturbances. Looking at Fig. 5.18a, the descent rate errors obtained with the QP algorithm were lower on average, but the baseline algorithm still performs well, with an average deck-relative descent rate close to the 0.25 m/s descent rate bias commanded by the baseline guidance algorithm. This result is similar to that obtained from the disturbance-free experiments performed in the MASK. The attitude landing errors also show a result similar to that obtained in the MASK experiments, with the QP algorithm resulting in slightly lower attitude errors on average, though one outlier case terminated with a pitch attitude error

slightly higher than the worst case for the baseline algorithm. Looking at the X and Y velocity errors, the QP guidance method does not provide any performance improvement and the results obtained with both guidance methods are very similar. Further, the position landing errors obtained with both algorithms were also comparable.



(a) Velocity landing errors.

(b) Position Landing Errors.

(c) Attitude Landing Errors.

Figure 5.18: Deck-relative velocity, position, and attitude at landing. On the velocity and attitude plots, faded markers represent individual landings, bold markers show the average, and error bars show 2σ of the error magnitude.

While the landing error statistics in Fig. 5.18 do not indicate that the QP algorithm provides significantly improved landing errors in a gusty environment, the QP method does allow for a more direct landing with a shorter duration than the baseline method. When operating in a gusty or turbulent environment this is advantageous, as the QP method will spend less time operating in the ship airwake. For the tests conducted for this section, the effect of this can be seen by looking at the control input time histories

shown in Fig. 5.19, which are plotted against the time remaining until landing. The aerodynamic disturbance here primarily affected the lateral and longitudinal axes, and an increased magnitude of the control inputs δ_{ail} and δ_{ele} can be observed once the UAV enters the flow induced by the fan. Since the baseline algorithm must slowly descend relative to the ship deck, δ_{ail} and δ_{ele} tend to be increased for approximately the final 5 seconds of the landing. With the QP guidance algorithm the UAV typically only operates in this region for the last 1.5 to 2 seconds of the trajectory, resulting in lower total control usage. Higher throttle inputs δ_{throt} are also seen for the baseline landings, though it should be noted that spikes in the baseline throttle inputs occurring with 1.75 to 4 seconds remaining are a product of the baseline guidance logic. As mentioned in Chapter 5.1.5.4, when the baseline algorithm switches from hovering over the deck to following deck motions while descending, a step in the position command can occur and result in a sharp throttle input. This behavior could potentially be averted by phasing in the switch between the approach and landing stages of the baseline algorithm.

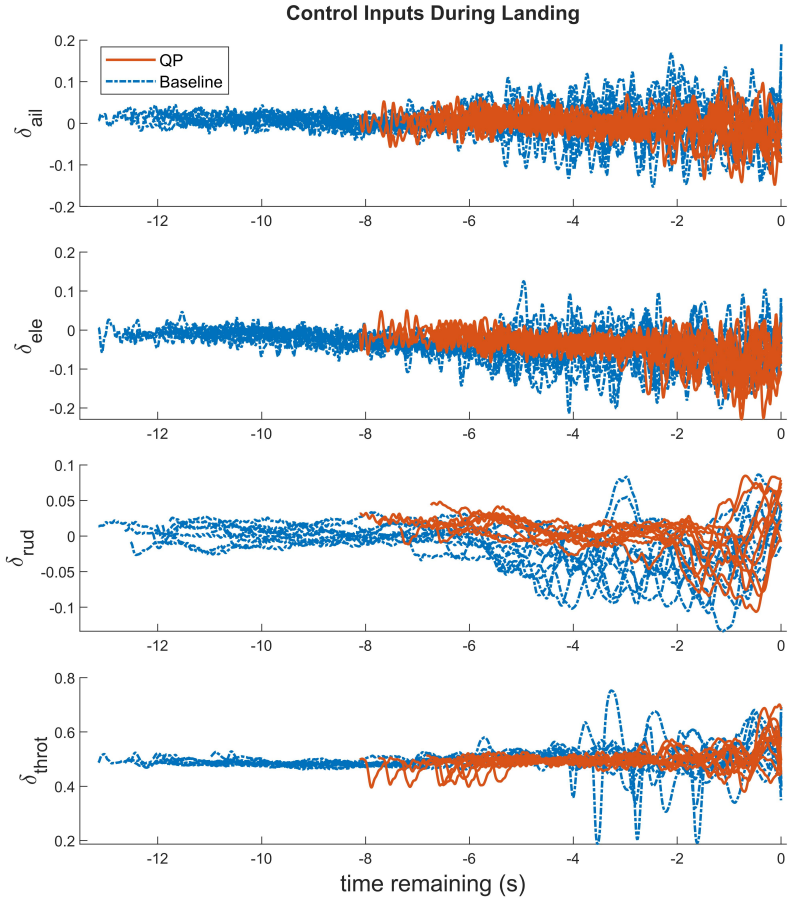


Figure 5.19: Control inputs plotted against time remaining until landing.

Chapter 6
Conclusions and Recommendations for Future Work

6.1 Conclusions

The rigorous experimental evaluation of autonomous ship landing guidance and control algorithms is lacking in the open literature. The vast majority of existing experimental data is from model-scale experiments, but the scaling of dynamics has not been considered in these tests. This work contributed toward addressing this gap by developing a methodology for performing model-scale autonomous landing experiments where both aircraft closed-loop dynamics and ship motions are related across test scales via Froude scaling. The validity of using Froude scaling to retain dynamic similarity in the closed-loop reference tracking dynamics and response to disturbances was then analyzed in Chapter 4. Based on this analysis, the following conclusions can be drawn:

1. If a model-following control law is designed such that the reference tracking dynamics of the closed-loop system approximate those of the command filter, then dynamic similarity can be retained for the reference tracking dynamics if the ship data, command models, deck state prediction algorithm, and guidance algorithm parameters are Froude scaled consistently. Even without considering the scaled response to aerodynamic disturbances, this offers an improvement to many of the more accessible test setups that have been used in the literature. For example, many tests have been performed with small UAVs and a motion platform to emulate the ship deck, and many of these tests have not included aerodynamic disturbances. The tests are therefore assessing the ability of a proposed algorithm to compensate for deck motions, and Froude scaling both the closed-loop reference tracking dy-

namics and ship motions will allow approximate dynamic similarity to be retained across scales in this scenario.

2. With a well designed model-following control law implemented on the model-scale aircraft, approximate similarity in the reference tracking dynamics can be attained without considering the scaling of the open-loop aircraft dynamics. In order to obtain an accurately scaled response to aerodynamic disturbances, however, the model-scale system must be designed to attain accurately scaled open-loop dynamics. If the model-scale open-loop dynamics are not designed to approximate those of a scaled full-scale model, then a cruder scaling of the response to external disturbances can be achieved by choosing feedback gains to match Froude scaled disturbance rejection bandwidths representative of full-scale aircraft.

Using the proposed scaling methodology, two sets of model-scale tests were performed to evaluate an advanced autonomous landing algorithm representative of the MPC methods prevalent in the literature, for which few experimental validations have been published. More specifically, the advanced landing algorithm utilizes quadratic programming optimization and plans the landing trajectory to deck state predictions produced by autoregressive time series models. A simpler "baseline" algorithm was also implemented for comparison, which commands a deck-relative flight path, closing the distance between the aircraft and deck at a steady rate. Both guidance algorithms provided position and heading commands to an explicit model following (EMF) control law, and the EMF controller command model parameters and feedback gains were chosen to match Froude scaled reference tracking dynamics and disturbance rejection bandwidths.

The first set of model-scale tests was performed at the Maneuvering and Seakeeping Basin (MASK) located at the Naval Surface Warfare Center Carderock Division. During these tests the EMF control law parameters were modified to impose artificial constraints on the maneuverability of the aircraft, providing insight into how well both guidance methods can cope with a less agile aircraft. Based on the experimental results, the following observations and conclusions can be made:

1. The baseline "deck tracking" landing strategy is both simple and effective when the UAV has the bandwidth required to track deck motions. This is supported by the low average velocity errors obtained with the baseline algorithm in high-bandwidth configurations.

2. The baseline landing method is sensitive to additional sources of lag, however. This was demonstrated by the tests performed with reduced vertical axis reference

tracking bandwidth. When the bandwidth was reduced to 6 rad/s with baseline guidance, several hard landings and a noticeable increase in the mean and variance of the deck-relative descent rates were observed. Based on the Froude scaling laws used here, a heave tracking bandwidth of 6 rad/s at model-scale corresponds to 1.6 rad/s at full-scale. The specifications for height response characteristics in ADS-33E-PRF [66] suggest a vertical axis time constant of 5 seconds or less for level 1 handling qualities. This roughly corresponds to a vertical axis bandwidth of 0.2 seconds. The results obtained here therefore suggest that the vertical axis tracking bandwidth would need to significantly exceed the level 1 handling qualities guideline to consistently perform soft landings with a "deck tracking" landing strategy in high sea states. Such a high bandwidth may be difficult for large or heavily loaded rotorcraft to attain.

3. The inclusion of deck state predictions and the ability to plan for the system dynamics allowed the QP algorithm to plan more direct landing paths, and also to perform landings with significantly lower tracking bandwidths than the baseline algorithm. In fact, consistently low deck-relative descent rates were obtained with the QP algorithm when the vertical axis bandwidth was reduced down to 0.79 rad/s. This bandwidth Froude scales to 0.2 rad/s at full-scale, approximately corresponding to the 5 second vertical axis time constant guideline for predicted level 1 handling qualities given in ADS-33E-PRF. This result suggests that a full-scale rotorcraft just meeting the level 1 height response guideline could feasibly achieve soft landings using a predictive landing strategy. It should be noted, though, that full-scale acceleration and jerk limits could be a limiting factor as well. For the tests conducted here, reducing the model-scale jerk limits down to 5 m/s^3 (1.34 m/s^3, or 0.13g per second, at full-scale) did not have a perceivable effect on QP algorithm performance. Additionally, the model-scale acceleration limit of 0.35g (also 0.35g at full-scale due to one-to-one scaling) was rarely approached, indicating that a lower acceleration limit could have been used without degrading QP algorithm performance. In order to better understand the ability of the QP algorithm to plan for more restrictive acceleration and jerk constraints, however, additional testing with more stringent output constraints is required.

4. Long term deck state predictions produced by the AR models were often unreliable, with the predictions typically beginning to converge toward the true deck state with 1.5 seconds or less remaining in the model-scale landings (this would correspond to

about 5.5 seconds or less at full-scale). Additionally, the velocity predictions often exhibited intermittent points with spikes in the prediction errors. This was due to intermittent buffering issues that occurred when sending ship measurement data over WiFi to the UAV.

5. Despite poor long term deck state predictions, the QP landing algorithm was able to match position and velocity well in the majority of cases. There were, however, several outlier cases in the MASK experiments with deck-relative landing velocities greater than 0.5 m/s. Additionally, there were a handful of cases where the QP algorithm led to deck-relative pitch angles of greater than 10 degrees at landing.

6. It was hypothesized that robustness to poor deck state predictions would be a limiting factor for the QP algorithm. These results, however, indicate that the land time update scheme was the culprit in the majority of cases where the QP algorithm showed poor performance. Removing cases from the experiments performed at the MASK where the maneuver duration was slightly decreased was found to remove almost all instances where a high deck-relative pitch attitude or velocity occurred at landing. The one exception to this was a case where deck altitude prediction errors were exceedingly poor during the last 0.5 seconds of the maneuver, and it is believed that this was related to data buffering issues that occurred occasionally during testing. These results give confidence in the feasibility of incorporating deck motion predictions directly in path planning, provided the deck state measurements used in the prediction scheme are not low quality.

The landings performed at the MASK did not include aerodynamic disturbances. Additional landings to a "virtual deck" were therefore performed at the Penn State Indoor Flight Facility to determine if the QP guidance algorithm offers any advantage over the baseline method when operating in the presence of a significant aerodynamic disturbance. Based on the results of these experiments, the following observations and conclusions can be made:

1. With feedback gains tuned to provide scaled DRBs meeting the guidelines for predicted level 1 handling qualities and the baseline guidance algorithm in a high tracking bandwidth configuration, both the QP and baseline guidance methods were able to perform successful landings in the presence of the aerodynamic disturbance.

2. The results did not indicate that recursively re-planning the trajectory from the perturbed aircraft state with the QP algorithm offered any advantage over the

baseline method (when the baseline algorithm is in a high tracking bandwidth configuration) in terms of matching deck position, velocity, and attitude while operating in a gusty environment. The QP algorithm did allow for landings with a shorter duration, however. For landings with the QP algorithm, the UAV therefore spent less time operating in the aerodynamic disturbance placed near the ship deck, resulting in lower total control usage.

6.2 Recommendations for Future Work

The scaling methodology proposed here could be used in a number of future model-scale studies on autonomous ship landing systems. For example, while tests conducted as part of this work examined the reference tracking properties necessary to compensate for ship motion with different landing guidance algorithms, a study on the necessary disturbance rejection properties was not performed. This is because the model-scale UAV built for these experiments was not designed to obtain Froude scaled open-loop dynamics, meaning an accurately scaled response to aerodynamic disturbances could not be guaranteed. Work by other researchers, however, has shown that small aircraft built with roughly Froude scaled properties such as rotor radius, mass, and inertia result in open-loop dynamics reasonably satisfying the Froude scaling assumption [52–54]. Utilizing a roughly scaled bare airframe, Froude scaled tests could be designed where a scaled ship hull is added to a platform used to emulate ship motion, and fans can be used to provide disturbances with a scaled mean wind velocity. A scaled control law could then be implemented, with control gains adjusted to vary disturbance rejection properties like DRB and DRP across test cases. Performing scaled tests such as this would give insight into disturbance rejection requirements for performing landings in a range of conditions. Evaluating this through model-scale testing is also advantageous as the near-ship aerodynamic environment is difficult to simulate.

Model-scale tests with varied disturbance rejection properties would also be pertinent to the evaluation of many proposed GPS-denied landing solutions. A number of papers have proposed methods for estimating deck-relative position and velocity for use in GPS-denied landings (for example, [26, 28, 31, 67, 68]), but the proposed control designs typically use the deck-relative position and velocity estimates in feedback as if they were inertial frame estimates. The unknown motion of the deck therefore appears as an output disturbance or sensor error to the control system. For small UAVs landing on a full-scale ship, this may not be problematic as standard linear feedback can provide high

bandwidth disturbance rejection that allows commands to be accurately tracked in the deck-relative frame. For larger rotorcraft, however, standard feedback control may not be able to sufficiently reject disturbances occurring in the frequency range of ship motion. Model-scale tests can be used to determine the necessary disturbance rejection properties needed for performing GPS-denied landings in various sea states. Additionally, adaptive mechanisms to estimate and compensate for the effective "disturbance" resulting from the unknown ship motion can be evaluated.

A combination of simulations and model-scale experiments could also be used to gain insight into the required accuracy of deck state predictions for use in a predictive landing strategy. Looking at the deck state prediction error statistics obtained from the MASK experiments (shown in Fig. 5.7), the plots show the expected shape where the mean and variance of the prediction error increases as the length of time into the future we are predicting increases. In many cases though, the prediction error mean and variance level off after a certain length of time into the future. This information could be used to perform simulations and model-scale tests with the purpose of determining requirements on prediction accuracy. For example, emulating the ship motion with pre-recorded time histories or simulated ship motion, the future states of the ship are known. Landings could therefore be performed where the known future ship motion is used to give a perfect "prediction" of ship motion across the optimization horizon. Landings could also be performed where the known ship data is corrupted and again used in the landing path optimization as the "predicted" ship motion. The ship data could be corrupted to give "prediction errors" that exhibit properties similar to the errors reported from the experiments conducted here, but with magnitude of prediction errors progressively increased until the landing algorithm can no longer perform adequately.

Bibliography

[1] "Naval helicopter history timeline 400BC till 1940," https://www.nhahistoricalsociety.org/indexphp/naval-helicopter-history-timeline-new/.

[2] Smith, Z. F. (2021) "Baseline for Virtual Dynamic Interface," in *AIAA AVIATION 2021 FORUM*, p. 2484.

[3] (2022), "Next generation ship-based autonomous helicopter provides expanded capabilities," https://news.northropgrumman.com/news/releases/northrop-grumman-built-mq-8c-fire-scout-makes-operational-deployment-with-the-us-navy?_gl=1*ss8xhv*_ga*Mjg2NzA1OTQzLjE2NzA2MTczOTc.*_ga_7YV3CDX0R2*MTY3MDYxNzM5Ny4xLjEuMTY3MDYxNzQyMS4wLjAuMA..

[4] (2022), "VSR700 autonomous take-off and landing capabilities tested at sea," . URL https://www.airbus.com/en/newsroom/press-releases/2022-03-vsr700-autonomous-take-off-and-landing-capabilities-tested-at-sea

[5] Pravitra, J. (2021) *Shipboard UAS Operations with Optimized Landing Trajectories*, PhD dissertation, The Pennsylvania State University.

[6] Horn, J. F., J. Yang, C. He, D. Lee, and J. K. Tritschler (2015) "Autonomous ship approach and landing using dynamic inversion control with deck motion prediction," in *41st European Rotorcraft Forum 2015, ERF 2015*, Deutsche Gesellschaft fuer Luft und Raumfahrt (DGLR), pp. 864–877.

[7] Hu, B., L. Lu, and S. Mishra (2018) "A Control Architecture for Time-Optimal Landing of a Quadrotor Onto a Moving Platform," *Asian Journal of Control*, **20**(5), pp. 1701–1712, https://onlinelibrary.wiley.com/doi/pdf/10.1002/asjc.1693.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/asjc.1693

[8] Tedrake, R. (2023) *Underactuated Robotics*.
URL https://underactuated.csail.mit.edu

[9] Ngo, T. D. and C. Sultan (2016) "Model predictive control for helicopter shipboard operations in the ship airwakes," *Journal of Guidance, Control, and Dynamics*, **39**(3), pp. 574–589.

[10] Persson, L. and B. Wahlberg (2019) "Model predictive control for autonomous ship landing in a search and rescue scenario," in *AIAA Scitech 2019 Forum*, p. 1169.

[11] Crouse, J. (2020) *Linear Model Predictive Control with Envelope Detection for Aerial Vehicle-Ship Intercept Scenarios*, MS thesis, The Pennsylvania State University.

[12] Greer, W. B. and C. Sultan (2020) "Shrinking horizon model predictive control method for helicopter–ship touchdown," *Journal of Guidance, Control, and Dynamics*, **43**(5), pp. 884–900.

[13] Greer, W. B. (2019) *Advanced Linear Model Predictive Control For Helicopter Shipboard Maneuvers*, PhD dissertation, Virginia Polytechnic Institute and State University.

[14] Ngo, T. D. and C. Sultan (2022) "Variable Horizon Model Predictive Control for Helicopter Landing on Moving Decks," *Journal of Guidance, Control, and Dynamics*, **45**(4), pp. 774–780.

[15] Baca, T., P. Stepan, V. Spurny, D. Hert, R. Penicka, M. Saska, J. Thomas, G. Loianno, and V. Kumar (2019) "Autonomous landing on a moving vehicle with an unmanned aerial vehicle," *Journal of Field Robotics*, **36**(5), pp. 874–891.

[16] Paris, A., B. T. Lopez, and J. P. How (2020) "Dynamic landing of an autonomous quadrotor on a moving platform in turbulent wind conditions," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 9577–9583.

[17] Guo, K., P. Tang, H. Wang, D. Lin, and X. Cui (2022) "Autonomous Landing of a Quadrotor on a Moving Platform via Model Predictive Control," *Aerospace*, **9**(1), p. 34.

[18] Nakamura, T., S. Haviland, D. Bershadsky, and E. N. Johnson (2016) "Vision-based optimal landing on a moving platform," Georgia Institute of Technology.

[19] Zhao, D., J. Krishnamurthi, S. Mishra, and F. Gandhi (2018) "A trajectory generation method for time-optimal helicopter shipboard landing," in *Annual Forum Proceedings-AHS International*, vol. 2018.

[20] Zhao, D., S. Mishra, and F. Gandhi (2019) "Real-time path planning for time-optimal helicopter shipboard landing via trajectory parametrization," .

[21] Comandur, V. and J. Prasad (2018) "Rotorcraft shipboard landing guidance using MPPI trajectory optimization," .

[22] Williams, G., A. Aldrich, and E. A. Theodorou (2017) "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, **40**(2), pp. 344–357.

[23] Yang, J. (2018) *Autonomous Control Modes and Optimized Path Guidance for Shipboard Landing in High Sea States*, PhD dissertation, The Pennsylvania State University.

[24] Lee, D. N. (1998) "Guiding movement by coupling taus," *Ecological psychology*, **10**(3-4), pp. 221–250.

[25] Voskuijl, M., G. Padfield, D. Walker, B. Manimala, and A. W. Gubbels (2010) "Simulation of automatic helicopter deck landings using nature inspired flight control," *The Aeronautical Journal*, **114**(1151), pp. 25–34.

[26] Holmes, W. K. (2017) *Vision-based Relative Deck State Estimation Used with Tau Based Landings*, MS thesis, The Pennsylvania State University.

[27] Lyu, Z., W. Ding, X. Sun, H. Sang, Y. Zhou, P. Yu, and L. Zheng (2021) "Dynamic landing Control of a Quadrotor on the Wave Glider," *Journal of Marine Science and Engineering*, **9**(10), p. 1119.

[28] Lee, B., V. Saj, D. Kalathil, and M. Benedict (2022) "Intelligent Vision-based Autonomous Ship Landing of VTOL UAVs," *Journal of the American Helicopter Society*.

[29] Lee, B. (2021) *On the Complete Automation of Vertical Flight Aircraft Ship Landing*, Ph.D. thesis, Texas A&M University.

[30] Xuan-Mung, N., S. K. Hong, N. P. Nguyen, T.-L. Le, et al. (2020) "Autonomous quadcopter precision landing onto a heaving platform: New method and experiment," *IEEE Access*, **8**, pp. 167192–167202.

[31] Wang, L. and X. Bai (2018) "Quadrotor autonomous approaching and landing on a vessel deck," *Journal of Intelligent & Robotic Systems*, **92**(1), pp. 125–143.

[32] Lu, Q., B. Ren, and S. Parameswaran (2018) "Shipboard landing control enabled by an uncertainty and disturbance estimator," *Journal of Guidance, Control, and Dynamics*, **41**(7), pp. 1502–1520.

[33] Patel, R., B. Le Floch, E. N. Johnson, and J. Crouse (2021) "Gpc-based deck motion estimation for autonomous ship deck landing of an unmanned aircraft," in *AIAA Scitech 2021 Forum*, p. 1814.

[34] Abujoub, S., J. Mcphee, C. Westin, and R. A. Irani (2018) "Unmanned aerial vehicle landing on maritime vessels using signal prediction of the ship motion," in *OCEANS 2018 MTS/IEEE Charleston*, IEEE, pp. 1–9.

[35] Fourie, C. K. (2015) *The autonomous landing of an unmanned helicopter on a moving platform*, Ph.D. thesis, Stellenbosch: Stellenbosch University.

[36] Vorwald, J., A. Schwartz, and C. Kent (2016) "Near Term Ship Motion Forecasting From Prior Motion," in *Fluids Engineering Division Summer Meeting*, vol. 50299, American Society of Mechanical Engineers, p. V01BT30A004.

[37] Kusters, J., K. Cockrell, B. Connell, J. Rudzinsky, and V. Vinciullo (2016) "FutureWaves™: A real-time Ship Motion Forecasting system employing advanced wave-sensing radar," in *OCEANS 2016 MTS/IEEE Monterey*, IEEE, pp. 1–9.

[38] Alford, L. K., R. F. Beck, J. T. Johnson, D. Lyzenga, O. Nwogu, and A. Zundel (2015) "A real-time system for forecasting extreme waves and vessel motions," in *International conference on offshore mechanics and arctic engineering*, vol. 56598, American Society of Mechanical Engineers, p. V011T12A056.

[39] Khan, A., C. Bil, and K. E. Marion (2005) "Ship motion prediction for launch and recovery of air vehicles," in *Proceedings of OCEANS 2005 MTS/IEEE*, IEEE, pp. 2795–2801.

[40] Wang, H., F. Wu, and D. Lei (2021) "Prediction of Ship Heave Motion Using Regularized BP Neural Network with Cross Entropy Error Function," *International Journal of Computational Intelligence Systems*, **14**(1), pp. 1–7.

[41] De Masi, G., F. Gaggiotti, R. Bruschi, and M. Venturi (2011) "Ship motion prediction by radial basis neural networks," in *2011 IEEE Workshop On Hybrid Intelligent Models And Applications*, IEEE, pp. 28–32.

[42] Monneau, A., N. K. M'Sirdi, S. Mavromatis, G. Varra, M. Salesse, and J. Sequeira (2020) "Adaptive prediction for ship motion in rotorcraft maritime operations," *CEAS Aeronautical Journal*, **11**(4), pp. 1071–1082.

[43] Küchler, S., T. Mahl, J. Neupert, K. Schneider, and O. Sawodny (2010) "Active control for an offshore crane using prediction of the vessel's motion," *IEEE/ASME transactions on mechatronics*, **16**(2), pp. 297–309.

[44] Zhao, X., R. Xu, and C. Kwan (2004) "Ship-motion prediction: algorithms and simulation results," in *2004 IEEE international conference on acoustics, speech, and signal processing*, vol. 5, IEEE, pp. V–125.

[45] Ma, J., T. Li, and G. Li (2006) "Comparison of representative method for time series prediction," in *2006 International Conference on Mechatronics and Automation*, IEEE, pp. 2448–2453.

[46] Lin, Z., Q. Yang, Z. Guo, and J. Li (2011) "An improved autoregressive method with kalman filtering theory for vessel motion predication," *International Journal of Intelligent Engineering and Systems*, **4**(4), pp. 11–18.

[47] Schwartz, A. (2015) "Systematic characterization of the naval environment (scone)–standard deck motion data for a generic surface combatant," *Memorandum from Office of Naval Research and Naval Surface Warfare Center-Carderock Division*.

[48] Mettler, B., C. Dever, and E. Feron (2004) "Scaling effects and dynamic characteristics of miniature rotorcraft," *Journal of guidance, control, and dynamics*, **27**(3), pp. 466–478.

[49] Mettler, B. (2013) *Identification modeling and characteristics of miniature rotorcraft*, Springer Science & Business Media.

[50] Hunt, G. (1973) "Similarity requirements for aeroelastic models of helicopter rotors," .

[51] Friedmann, P. (2004) "Aeroelastic scaling for rotary-wing aircraft with applications," *Journal of fluids and structures*, **19**(5), pp. 635–650.

[52] Ivler, C. M., E. S. Rowe, J. Martin, M. J. S. Lopez, and M. B. Tischler (2021) "System Identification Guidance for Multirotor Aircraft: Dynamic Scaling and Test Techniques." *Journal of the American Helicopter Society*, **66**(2), pp. 1 – 16.

[53] Tobias, E. L., F. C. Sanders, and M. B. Tischler (2018) "Full-Envelope Stitched Simulation Model of a Quadcopter Using STITCH," in *AHS International 74th Annual Forum & Technology Display*, AHS, Phoenix, Arizona.

[54] Ivler, C. M., K. Truong, D. Kerwin, J. Otomize, D. Parmer, M. B. Tischler, and N. Gowans (2022) "Development and Flight Validation of Proposed Unmanned Aerial System Handling Qualities Requirements," *Journal of the American Helicopter Society*, **67**(1).

[55] Ambroziak, L., M. Ciężkowski, A. Wolniakowski, S. Romaniuk, A. Bożko, D. Ołdziej, and C. Kownacki (2022) "Experimental tests of hybrid VTOL unmanned aerial vehicle designed for surveillance missions and operations in maritime conditions from ship-based helipads," *Journal of Field Robotics*, **39**(3), pp. 203–217.

[56] Tischler, M. B. and R. K. Remple (2012) *Aircraft and rotorcraft system identification*, American Institute of Aeronautics and Astronautics Reston, VA.

[57] Bendat, J. S. and A. G. Piersol (2011) *Random data: analysis and measurement procedures*, John Wiley & Sons.

[58] Wei, W., M. B. Tischler, and K. Cohen (2017) "System identification and controller optimization of a quadrotor unmanned aerial vehicle in hover," *Journal of the American Helicopter Society*, **62**(4), pp. 1–9.

[59] Saetti, U., J. F. Horn, S. Lakhmani, C. Lagoa, and T. F. Berger (2020) "Design of Dynamic Inversion and Explicit Model Following Flight Control Laws for Quadrotor UAS," *Journal of the American Helicopter Society*, **65**(3), pp. 1–16.

[60] Cho, S. H., S. Bhandari, F. C. Sanders, M. B. Tischler, and K. Cheung (2019) "System identification and controller optimization of coaxial quadrotor UAV in hover," in *AIAA Scitech 2019 Forum*, p. 1075.

[61] Tischler, M. B., T. Berger, C. M. Ivler, M. H. Mansur, K. K. Cheung, and J. Y. Soong (2017) *Practical methods for aircraft and rotorcraft flight control design: an optimization-based approach*, American Institute of Aeronautics and Astronautics, Inc.

[62] Buckingham, E. (1914) "On physically similar systems; illustrations of the use of dimensional equations," *Physical review*, **4**(4), p. 345.

[63] Kittirungsi, B. (2008) *A Scaling Methodology for Dynamic Systems: Quantification of Approximate Similitude and Use in Multiobjective Design.*, PhD dissertation, The University of Michigan.

[64] Berger, T., C. Ivler, M. G. Berrios, M. B. Tischler, and D. Miller (2016) "Disturbance rejection handling qualities criteria for rotorcraft," in *72nd Annual Forum of the American Helicopter Society, West Palm Beach, USA*.

[65] Horn, J. F. (2019) "Non-linear dynamic inversion control design for rotorcraft," *Aerospace*, **6**(3), p. 38.

[66] Blanken, C. L., M. B. Tischler, J. A. Lusardi, T. Berger, C. M. Ivler, and R. Lehmann (2019) *Proposed Revisions to Aeronautical Design Standard-33E (ADS-33E-PRF) Toward ADS-33F-PRF*, *Tech. rep.*, CCDC AvMC.

[67] Arora, S., S. Jain, S. Scherer, S. Nuske, L. Chamberlain, and S. Singh (2013) "Infrastructure-free shipdeck tracking for autonomous landing," in *2013 IEEE international conference on robotics and automation*, IEEE, pp. 323–330.

[68] Sanchez-Lopez, J. L., J. Pestana, S. Saripalli, and P. Campoy (2014) "An approach toward visual autonomous ship board landing of a VTOL UAV," *Journal of Intelligent & Robotic Systems*, **74**, pp. 113–127.

Vita

Christopher M. Hendrick

Christopher Hendrick was born in Syracuse, New York. In August 2014 he began his undergraduate education at the University at Buffalo, graduating in May 2018 with Bachelor of Science degrees in mechanical and aerospace engineering. Christopher then enrolled in the Master of Science program in mechanical engineering at the Pennsylvania State University, where he studied control theoretic models of human pilots applied to rotorcraft flight simulation. After completing his Master's degree in August 2020, Christopher went on to pursue a Ph.D. in Aerospace Engineering at Penn State, where his research has focused on flight control, optimal guidance, and model-scale analysis of autonomous ship landing systems for rotorcraft.