ANALYSIS OF FLIGHT CONTROL AND TRAJECTORY PLANNING FOR AUTONOMOUS SHIP LANDING USING SMALL-SCALE UAVS

Christopher M. Hendrick and Joseph F. Horn, The Pennsylvania State University, University Park PA, USA

ABSTRACT

Due to the potential to expand flight envelopes and increase flight safety for sea based rotorcraft, there has been a drive to produce reliable autonomous ship landing algorithms. These algorithms are inherently complex and must be validated by extensive experimentation. Experimentation at model scale offers a controllable test bed that can be used to isolate the effects of individual parameter variations, helping gain insight into the limitations and vulnerabilities of a complex landing algorithm. In this paper, the development of an autonomous landing control mode suitable for use in scaled experiments of this nature is presented. The main goal in algorithm design was to create a landing solution representative of other methods published in the literature, while also allowing for easy variation of parameters during testing. The resulting algorithm features explicit model following controllers, quadratic programming trajectory generation, and deck motion prediction based on autoregressive models. Additionally, a scaling method based on Froude scaling is proposed and results from 230 scaled flight tests to a virtual deck are discussed. During testing, reference tracking bandwidths and jerk limitations were progressively reduced and deck prediction accuracy was degraded. Results show that the landing algorithm performs well for scaled moderate to high sea states and deck motion predictions help compensate for reduced aircraft mobility. It was also found that reduced tracking bandwidths effect landing accuracy, but that inclusion of reasonable jerk limitations can be important to prevent the autonomy algorithm from overcompensating for artificially reduced model scale bandwidths.

NOTATION

Variables

$\underline{\alpha}$	=	AR model coefficients
A, B, C, D	=	state space matrices
g	=	gravitational acceleration
j	=	jerk
$K_P,\ K_I,\ K_D$	=	PID control gains
N	=	prediction horizon
N_F	=	Froude number
p, q, r	=	UAV body axis angular rates
ϕ, θ, ψ	=	3-2-1 Euler angles
Q, R, S	=	cost function weight matrices
$T_{A/B}$	=	rotation matrix from B to A coordinates
τ_d	=	discrete delay
v_k	=	white noise vector at time step k
u_k	=	control input at time step k
u, v, w	=	UAV body axis translational rates
X^a, Y^a, Z^a	=	X, Y, and Z positions in frame a
$\vec{x}_k, \vec{y}_k$	=	state and output vectors at time step k

Superscripts and Subscripts

cmd	=	commanded value
d	=	deck state
dp	=	predicted deck state
dhf	=	deck heading frame
hf	=	UAV heading frame
I	=	inertial forward-right-down reference frame
uav	=	UAV state

1 INTRODUCTION

The difficulty of performing shipboard landings during moderate to high sea states results in restricted flight envelopes for sea-based rotorcraft. To help mitigate this issue, numerous research campaigns have investigated some or all aspects of autonomous rotorcraft shipboard landing solutions. This includes the associated deck state estimation, deck motion forecasting, and trajectory generation and control problems. In order to confidently field these autonomous landing systems on larger rotorcraft, extensive experimentation must be performed. While testing at full scale is inevitably necessary, testing at small scale can be used to bridge the gap between simulation and full scale tests. Key advantages of testing at model scale are:

1. It provides 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.

These advantages can be leveraged to gain insight into potential benefits, limitations, and design requirements prior to performing full scale tests. Care should be taken, however, to ensure that the scaled tests are dynamically similar to full scale use cases. For example, the frequency of ship motion, aircraft closed loop bandwidths, and limits on control authority should be systematically adjusted to be roughly consistent with reduction in scale.

To date, little attention has been given to the scaling of dynamics when exploring the autonomous launch and recovery problem. Unrelated research, however, has shown that Froude scaling can give a good approximate relation between small scale rotorcraft and dynamically similar larger scale prototypes. For example, in [1] the lateral dynamics of an IRIS+ quadcopter are compared to those of an XV-15. Froude scaling was reported to give a reasonable prediction of the IRIS+ modal frequencies based off the XV-15 dynamics, with the Froude number N_F taken as the ratio of full to small scale hub-to-hub distance. In [2] similar results were found when comparing Froude scaled stability and control derivatives of two different sized hexarotor UAVs. In [3] 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 off hub-to-hub distance is a promising method for establishing predictive handling qualities metrics, such as disturbance rejection bandwidth (DRB), for UAVs. These studies suggest that Froude scaling is an appropriate method for relating ship and aircraft dynamics across test scales.

This paper presents the development and application of autonomous trajectory generation and control laws suitable for scaled experiments, where experiments are designed using Froude scaling principles. The main goal when designing the landing algorithm was not to formulate a novel solution to the autonomous landing problem. Instead, the goal was to create a representative landing solution based off the current literature that also allows for easy variation of parameters during testing. Parameter variations can then be used to study sensitivity of the landing algorithm to a variety of factors, including degraded handling qualities, maneuverability constraints, and deck motion prediction accuracy. The resulting algorithm features an explicit model following (EMF) control architecture, quadratic programming (QP) based trajectory planning, and deck motion prediction based on autoregressive (AR) time series models.

The EMF control architecture was chosen because it offers easy tailoring of control bandwidths, which is essential for these experiments. The decision to use an optimal control based path planning method combined with deck motion predictions was made because similar methods have been used in past works. For example, in [4] a nonlinear model predictive control (MPC) algorithm named “Ensemble iLQR” was developed. The solver plans multiple trajectories simultaneously and incorporates deck motion predictions produced by AR time series models. In [5] a QP based shrinking horizon MPC method was applied to the ship landing problem, but the deck states at land time were assumed to be known perfectly a priori. In [6] a trajectory generation law was developed based on classical optimal control, with paths planned to forecasted deck states produced by minor components analysis. The QP based path planning method presented here is representative of the MPC style trajectory planning that is prevalent in the literature, and is computationally efficient enough to run on our onboard computer’s CPU in real time. A nonlinear algorithm like that presented in [4] has the potential to improve performance, but real time implementation requires leveraging highly parallelized GPU programming. Sufficient capabilities for our experiments are obtained with the QP architecture, and there is the additional benefit that hard constraints can be directly included in the QP optimization. These hard constraints can be used to experiment with degraded aircraft mobility.

In addition to the autonomous landing controller, flight tests to a virtual deck are discussed. During these tests reference tracking bandwidths, jerk limits, and deck prediction lead time were decreased and the effects of this were analyzed. The deck is referred to as virtual because time histories of deck data were simply forwarded to the drone. This was done due to lack of availability of a suitable landing platform, but future work will attempt landings to a model scale ship. In these tests the UAV operated as if an actual landing was being performed, except no throttle down was initiated due to lack of a physical landing deck.

2 SCALING METHODOLOGY

We first introduce our proposed scaling methodology, which will be referred to throughout the rest of this paper. Both the closed loop vehicle dynamics and the ship motion characteristics are designed based on Froude scaling. With this method, model scale distance is calculated by dividing full scale distance by the Froude number 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 we can derive the dynamic scaling laws shown in table 1 (note that we multiply by these scale factors to go from full to model scale). To illustrate this, we consider scaling jerk. Looking at the units of jerk, we have

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

where m denotes meters and s denotes seconds. This shows model scale jerk is increased from full scale by a factor of $\sqrt{N_F}$.

Table 1: Froude Scaling Factors

Value of Interest	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	$1/N_F^5$

The ratio of full to model scale hub-to-hub distance is usually taken as the Froude number when applying Froude scaling to multi-rotor aircraft control characteristics. Here, however, we propose using vehicle mass ratio to establish the scale factor. That is, our Froude number is calculated from

$$N_F = \left(\frac{M_{fs}}{M_{ms}}\right)^{1/3} \tag{2}$$

where M_{fs} and M_{ms} represent full and model scale vehicle masses, respectively. The reason for using mass ratio is that, for tracking a trajectory, the UAV can be modelled as a point mass with a thrust vector. The bandwidth with which the thrust vector can rotate is prescribed in our attitude controller, and the bandwidth with which the thrust magnitude can vary is prescribed in our Z position controller. Additionally, the relative thrust to weight ratio can be particularly important when performing ship landings, as ship motions are often aggressive in heave. The downside to using mass ratio is that it is undesirable to have handling qualities requirements scale with vehicle mass since mass depends on payload, fuel, and other factors. Here, though, our goal is not to establish strict handling qualities requirements. Furthermore, the scaling factor is found from a cube root of mass ratio. Changes in the mass of either aircraft therefore do not have a large impact on the scaling factor unless they are

very large compared to the nominal aircraft weight. Moving forward with our mass scaling law, the full scale aircraft case that is considered here is a medium weight helicopter roughly the size of a UH-60. The UAV used in experimentation weighs 6.6 pounds, so taking the nominal full scale aircraft weight to be approximately 17,500 pounds we have:

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

For the remainder of this paper, all scaling will be based on this value of $N_F = 13.8$.

3 SHIP DATA

The ship motion time histories are 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 [7]. The SCONE data provides 6 degree of freedom motion with three different levels of severity: low, moderate, and high. Additionally, data sets are identified as either roll dominant or heave dominant. For all data sets the ship is given an approximately constant forward velocity. In the tests discussed here, the moderate and high data sets were used for both the roll and heave dominant cases. For the remainder of this paper these data sets will be referred to using the SCONE naming convention - 2H refers to moderate heave dominant motion, 2R refers to moderate roll dominant, 3H refers to high heave dominant, and 3R refers to high roll dominant.

Several adjustments were made to the SCONE data prior to use in model scale testing. First, the scaling methods from section 2 were applied. The time vector in each SCONE data file was divided by $\sqrt{N_F}$, resulting in higher frequency deck motion. The remainder of the data was adjusted to model scale by multiplying by the scale factors listed in table 1. Due to limited lab space, it was also necessary to remove the mean surge velocity and associated forward deck motion from the model scale deck data. To achieve this, both the deck forward velocity and forward displacement were high pass filtered by a first order filter with a pole at -0.37. This is low enough frequency to avoid altering data around the peak of the ship amplitude spectrum while still removing the constant velocity trend from the data. Additionally, the mean position of the scaled deck data was altered to place the virtual deck at the desired location in the test facility.

4 TEST FACILITIES AND CONDITIONS

4.1 Motion Capture Lab and UAV Platform

Figure 1: UAV platform used in experiments.

Flight tests were performed in the Penn State Motion Capture Lab. The facility capture volume has a length and width of 25 feet and a height of 15 feet. The motion capture cameras and software were supplied by Vicon Motion Systems.

A coaxial hexacopter UAV (shown in Fig. 1) has been designed for use in deck landing experiments. The hexacopter has 10 inch rotors and weighs 6.6 pounds. The UAV is equipped with a Pixhawk flight controller running the PX4 flight control firmware and an Odroid XU4 on-board computer. The Odroid uses the Robotic Operating System (ROS) to communicate with the Pixhawk over a serial link and is also fitted with a WIFI module, allowing for communication with the ground station computer. The WIFI link is used to forward motion capture data to the UAVs as a substitute for GPS during indoor testing, and also to send deck motion data from the ground station to the UAV. Motion capture data was sent to the UAV at 100 Hz, while the deck motion data was streamed at 50 Hz.

A block diagram depicting the integrated test hardware is shown in Fig. 2. Note the deck forecasting algorithm, trajectory generator, and EMF control laws run on the Odroid in real time. The deck motion predictor and QP trajectory generator each update at a rate of 10 Hz, while the EMF control law updates at 100 Hz. The controller produces angular rate and throttle commands which are sent to the PX4 autopilot.

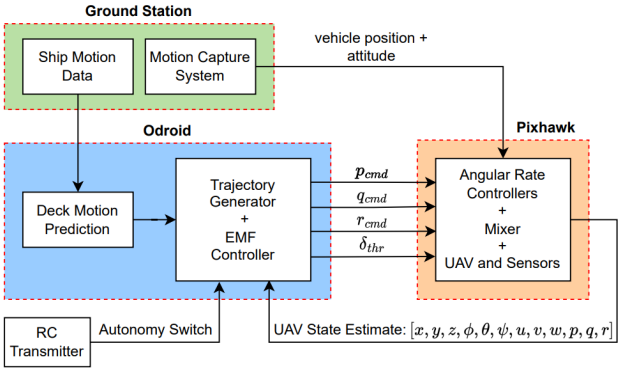

Figure 2: Test hardware integration.

4.2 Flight Test Cases

A total of 230 flight tests were performed, with reference tracking bandwidths and deck motion forecasting varied as shown in table 2. The high, medium, and low tracking cases refer to variations in reference tracking bandwidth and jerk constraints. Details on the control bandwidth variations for each case are given in section 5.2.1. Details on the jerk constraints chosen for each case are given in section 7.2.3. Referring to table 2, test cases 1-5 were each run 10 times for each of the 4 deck motion cases, accounting for the first 200 tests. During these tests the deck motion forecasting algorithm was executed without any modification, but reference tracking bandwidths and jerk constraints were varied.

Test cases 6-8 accounted for the final 30 tests, which were only run for the 3H deck motion case. During these tests 10 landings were performed for each test case, with varied vertical bandwidth and jerk constraints, degraded deck motion prediction, and the land time update (discussed in section 7.2.5) turned off. To degrade the deck motion predictions, the predicted deck states were frozen 1.3 seconds into the prediction horizon. For example, if we have a prediction horizon of 3 seconds, the deck predictions from the current time

Table 2: Landing Test Cases

Test Case	Roll/Pitch Tracking Case	Heave Tracking Case	Land Time Update	Degraded Deck Prediction	Deck Data Sets Used
1	High	High	on	off	2H,3H,2R,3R
2	med	High	on	off	2H,3H,2R,3R
3	low	High	on	off	2H,3H,2R,3R
4	High	med	on	off	2H,3H,2R,3R
5	High	low	on	off	2H,3H,2R,3R
6	High	High	off	on	3H
7	High	med	off	on	3H
8	High	low	off	on	3H

to 1.3 seconds in the future are unmodified. The predictions from 1.3 to 3 seconds into the future, however, are frozen at the 1.3 second ahead prediction. The reason for doing this was to see if including deck predictions past this point could help alleviate control bandwidth requirements, despite the fact that deck motion prediction accuracy drops off quickly after this point. The chosen freeze time of 1.3 seconds also corresponds to slightly under 5 seconds at full scale, which was used as the full scale prediction horizon in [6] and [8]. The land time update was turned off during these tests because it relies on the long term deck state prediction, and that information is effectively cut out with the degraded predictions.

Note that the results of these tests could be sensitive to the point in the deck data time history that landing is initiated. To prevent inconsistent land times across test cases, each deck data set was assigned ten start times in the deck motion time history. These same 10 start times were then used for each set of 10 tests run on a deck motion case. For example, if the first landing on deck data set 3H for test case 1 was assigned a start time of 100 seconds, then the first landing on deck data 3H for all other test cases was assigned the same start time of 100 seconds. The ten start times were chosen by producing uniform random numbers that lied within a 100 second segment in the middle of each deck data set. Additionally, the deck motion data was triggered to begin streaming by the UAV arriving at a specified waypoint, which gave the UAV an identical starting point for each test. Once the UAV reached the specified starting point and deck data began streaming, the UAV was commanded to hover at the starting point for 20 seconds. This gave consistent timing between the deck data stream starting and the landing maneuver initiating. The wait period was set to 20 seconds to allow time for the estimated parameters in the deck motion prediction algorithm to converge.

5 MODEL FOLLOWING CONTROL LAW

5.1 Control Architecture

A position command - position hold EMF controller similar to that presented in [9] was used in autonomous control modes. EMF uses a feedforward inversion model to approximately cancel plant dynamics and linear feedback compensation. A time delay on commands is included to capture phase lag due to higher frequency dynamics not included in the inversion model. With an accurate inversion model, the reference tracking response will approximate a specified ideal model (called the command filter here) with the added input delay. This allows for easy tailoring of reference tracking dynamics.

While the overall control design is similar to that presented in [9], here command filters are included for X^I, Y^I, and Z^I position commands. There is also no inner vertical rate loop, as the transfer function from throttle input to Z^I position output is inverted directly. Note that the UAV is modelled by decoupled transfer functions in roll, pitch, yaw, and heave. The models were derived using the system identification (ID) process presented in [9] and proved to be very accurate representations of the dynamics around hover. For designing the outer loop controllers, we consider roll and pitch attitudes as the driving inputs and neglect the higher bandwidth attitude dynamics. The resulting approximate models for X and Y heading frame accelerations are then given as

$$\ddot{X}^{hf}_{uav} \approx -g\theta \qquad \ddot{Y}^{hf}_{uav} \approx g\phi \tag{4}$$

The transformation from the UAV heading frame to the inertial frame is given as

$$T_{I/hf} = \begin{bmatrix} cos(\psi_{uav}) & -sin(\psi_{uav}) \\ sin(\psi_{uav}) & cos(\psi_{uav}) \end{bmatrix} \tag{5}$$

A block diagram depicting the X^I and Y^I outer loop controllers is shown in Fig. 3. Note that position command filter inputs u_x and u_y are given in the inertial frame. The X and Y velocity and position tracking errors, as well as the integral of position tracking error, are also calculated in the inertial frame. The tracking errors are then transformed to the aircraft heading frame prior to multiplying by the feedback gains. The same process is applied in the acceleration feedforward path. These transformations are necessary to produce appropriate pitch and roll commands, as our control gains are designed based off the heading frame model from Eq. (4). Also note that acceleration commands are low pass filtered. This is done because the QP trajectory generators produce new commands at 10 Hz, but the EMF control laws execute at 100 Hz, resulting in large oscillation in the acceleration command. The low pass filter was second order with frequency and damping ratio set to 25 rad/s and 0.707, respectively. This greatly attenuated the oscillations in the acceleration feedforward without introducing enough lag to significantly affect performance.

The Z^I position controller has a similar structure to the X^I and Y^I controller, with a second order command filter and PID feedback compensation. The acceleration feedforward is also low pass filtered as in the X^I and Y^I controller. The main differences are the absence of an inner loop for Z^I position control and that no frame transformations are necessary. Additionally, the Z^I position controller produces throttle inputs directly, rather than commanding a lower level rate controller (as is done by the attitude controller shown in Fig. 3). The Z^I position inversion model and command time delay are derived from system ID (see [9]).

Block diagrams depicting the roll, pitch, and yaw attitude command - attitude hold EMF controllers are given in [9],

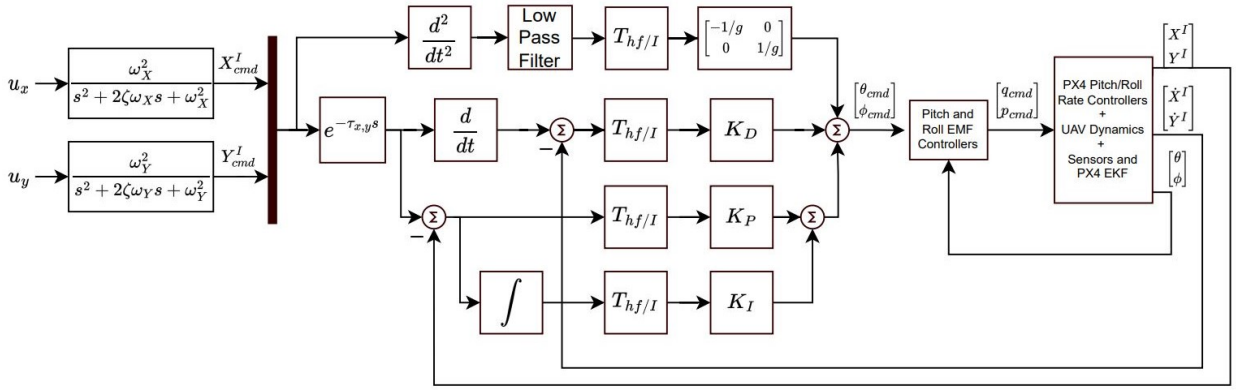

Figure 3: X and Y EMF position controller.

but are not reproduced here since the structure is identical. The attitude controllers use models obtained from system ID in the inversion feedforward and PI feedback compensation. Command filters are specified to be second order, just as is done for the position command filters shown in Fig. 3. The attitude controllers produce angular rate commands, which are input to the PX4 angular rate controller.

5.2 Control Bandwidth Scaling

The experiments we wish to perform with this control architecture include variations in reference tracking bandwidth and DRB (defined in [10]), where model scale bandwidths are chosen based off handling qualities criteria for full scale rotorcraft. Methods for systematically adjusting control parameters to achieve this are discussed in the following two subsections.

5.2.1 Scaled Reference Tracking

To exemplify the process used to determine our model scale control parameters, we will consider the longitudinal control axis. 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 2, we simply scale this by $\sqrt{N_F}$ to arrive at the model scale pitch command filter frequency:

$$\omega_{\theta,cf} = \omega_{\theta,fs}\sqrt{N_F} \tag{6}$$

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

$$\omega_{X,cf} = \frac{\omega_{\theta,cf}}{5} \tag{7}$$

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 $\tau_{x,y}$ in Fig. 3, is also calculated based off 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, we write

$$\frac{\theta_{uav}(s)}{\theta_{cmd}(s)} \approx \frac{\omega^2_{\theta,cf}}{s^2+2\zeta\omega_{\theta,cf}s+\omega^2_{\theta,cf}}e^{-\tau_\theta s} \tag{8}$$

To approximate the phase of the attitude dynamics in Eq. (8), we calculate $\tau_{x,y}$ as follows:

$$\tau_{x,y} = 1.65/\omega_{\theta,cf} + \tau_\theta \tag{9}$$

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. 3, note that $\tau_{x,y}$ delays both the X^I and Y^I position and velocity commands. While we only considered the pitch attitude tracking response when calculating $\tau_{x,y}$ in Eq. (9), the roll attitude command filter is specified to be identical to the pitch command filter for all tests here. The value of τ_θ in Eq. (9) is also very similar to the value of τ_ϕ for the UAV used here (τ_θ = 0.0273s and τ_ϕ = 0.0257s). The similarity in τ_θ and τ_ϕ has been found common when performing system ID on other UAV air frames, and larger differences between τ_θ and τ_ϕ would not have a large impact anyhow since the value of $\tau_{x,y}$ is dominated by the first term of Eq. (9). By scaling $\tau_{x,y}$ and $\omega_{X,cf}$ based off $\omega_{\theta,cf}$, the position and attitude tracking responses are consistently scaled based off just the full scale attitude tracking bandwidth $\omega_{\theta,fs}$. Note that the above process is used for scaling the X^I and Y^I controllers, but for Z^I and yaw control we just need to scale command filter frequencies by $\sqrt{N_F}$ as was done for pitch in Eq. (6).

Table 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

The chosen command filter frequencies for the high, medium, and low test cases discussed in section 4.2 are given in table 3. The full scale values were chosen to hit a range of realistic bandwidths for a full scale rotorcraft similar to the UH-60. The model scale values were calculated as discussed in the preceding paragraphs. The X^I and Y^I position command filter frequencies are not shown but can be calculated from Eq. (7). Note that the yaw bandwidth was set to correspond to 3.0 rad/s at full scale, which achieves level 1 handling qualities according to ADS-33E-PRF. This was not varied during testing as the deck does not yaw significantly in the ship motion data sets and yaw error is more tolerable than for other degrees of freedom.

5.2.2 Scaled Disturbance Rejection

To calculate our desired model scale DRB we follow the same process shown in Eq. (6), multiplying the full scale DRB by $\sqrt{N_F}$. To achieve this value we need to systematically tune our feedback compensators. Starting again with the longitudinal control axis, we first consider the pitch attitude loop. The pitch proportional gain $K_{P,\theta}$ is set to get 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 $K_{I,\theta} = 0.2K_{P,\theta}\omega_{co}$, where ω_{co} is the gain crossover frequency. This 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 heading frame. Modelling the heading frame X acceleration as shown in Eq. (4) and taking the Laplace transform, we get the following transfer function:

$$\frac{X^{hf}_{uav}(s)}{\theta_{uav}(s)} = \frac{-g}{s^2} \tag{10}$$

For position control we consider θ_{uav} as the input and close the loop on this model with PID feedback compensation. Considering the effect of an additive disturbance X^{hf}_{dst} at the X^{hf} position output on the position tracking error, we can derive the following expression for the error dynamics:

$$\frac{e(s)}{X^{hf}_{dst}(s)} = \frac{s^3}{s^3 - g(K_{D,X}s^2 + K_{P,X}s + K_{I,X})} \tag{11}$$
$$\textit{where}\ \ e(s) = X^{hf}_{cmd}(s) - X^{hf}_{uav}(s)$$

The denominator of Eq. (11) can then be factored into a second order and first order polynomial, yielding

$$\frac{e(s)}{X^{hf}_{p}(s)} = \frac{s^3}{(s^2 + 2\zeta_d\omega_d s + \omega_d^2)(s + p_d)} \tag{12}$$

where we set $\zeta_d = 1$, $p_d = 0.2\omega_d$, and ω_d is tuned to give a desired tradeoff 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 by Eq. (13). The same process was used for determining the Y position hold PID gains.

$$\begin{aligned} K_{P,X} &= -\frac{(\omega_d^2 + 2\zeta_d\omega_d p_d)}{g} \\ K_{I,X} &= -\frac{p_d\omega_d^2}{g} \\ K_{D,X} &= -\frac{(2\zeta_d\omega_d + p_d)}{g} \end{aligned} \tag{13}$$

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

$$K_Z(s) = \frac{K_{D,Z}s^2 + K_{P,Z}s + K_{I,Z}}{s} = \frac{K(s + z_1)(s + z_2)}{s} \tag{14}$$

The gain K in Eq. (14) sets 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 from Eq. (15), allowing gains to be set by only adjusting K.

$$K_{P,Z} = K(z_1 + z_2), \quad K_{I,Z} = z_1 z_2 K, \quad K_{D,Z} = K \tag{15}$$

The DRBs and disturbance rejection peaks (DRPs) for each control axis are shown in table 4. The control gains for roll, pitch, yaw, and heave were tuned to meet the scaled ADS-33E-PRF level 1 handling qualities criteria in both these metrics. These gains 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. Note that "High" refers to the high tracking bandwidth case from table 3. For medium and low tracking bandwidth, 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 in our simplified experiments since there are no significant disturbances introduced.

Table 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
$\boldsymbol{Z}$	1.04	0.28	1.72
$\boldsymbol{X, Y}$ **High**	0.67	0.18	2.74
$\boldsymbol{X, Y}$ **med**	0.59	0.16	3.05
$\boldsymbol{X, Y}$ **Low**	0.42	0.11	3.14

6 DECK MOTION PREDICTION

The deck motion prediction algorithm used here is based on the AR time series model. The use of a time series method is appealing as it requires no model of the ship dynamics and computational cost can be kept reasonable. Additionally, past works have found this class of methods to converge to reasonable predictions with sufficient lead time for performing autonomous landings (see [4], [6], [8], and [11]). Note that AR models were used in [4] and [11], and the minor component analysis (MCA) method was used in [6] and [8]. These methods performed comparably, so the AR method was chosen here due to its ease of implementation.

6.1 AR Model Estimation and Propagation

The estimation of model parameters was carried out in a manner similar to that presented in [4], 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

$$\vec{y}_k = \alpha_1\vec{y}_{k-1} + \alpha_2\vec{y}_{k-2}\cdots + \alpha_{N_{lag}}\vec{y}_{k-N_{lag}} + \vec{v}_k \tag{16}$$

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. (16) and writing in matrix form, we have

$$\vec{y}_k^T = \begin{bmatrix} \vec{y}_{k-1}^\intercal & \vec{y}_{k-2}^\intercal & \cdots & \vec{y}_{k-N_{lag}}^\intercal \end{bmatrix} \begin{bmatrix} \alpha_1^\intercal \\ \alpha_2^\intercal \\ \vdots \\ \alpha_{N_{lag}}^\intercal \end{bmatrix} + \vec{v}_k \tag{17}$$

or, in a more compact form,

$$\vec{y}_k^T = \bar{Y}_{lag}\underline{\alpha} + \vec{v}_k \tag{18}$$

This is then separated into one equation for each output:

$$\begin{aligned} y_1 &= \bar{Y}_{lag}\underline{\alpha}_1 + v_1 \\ y_2 &= \bar{Y}_{lag}\underline{\alpha}_2 + v_2 \\ &\vdots \\ y_m &= \bar{Y}_{lag}\underline{\alpha}_m + v_m \end{aligned} \tag{19}$$

Here $\underline{\alpha}_i$ represents the ith column of $\underline{\alpha}$. Each column vector $\underline{\alpha}_i$ 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 $\underline{\alpha}$ obtained, we propagate the AR model into the future to cover the desired prediction horizon:

$$\begin{aligned} \vec{y}_{k+1}^{\intercal} &= \begin{bmatrix} \vec{y}_k^{\intercal} & \vec{y}_{k-1}^{\intercal} & \cdots & \vec{y}_{k-N_{lag}+1}^{\intercal} \end{bmatrix} \underline{\alpha} \\ &\vdots \\ \vec{y}_{k+N}^{\intercal} &= \begin{bmatrix} \vec{y}_{k-1+N}^{\intercal} & \vec{y}_{k-2+N}^{\intercal} & \cdots & \vec{y}_{k-N_{lag}+N}^{\intercal} \end{bmatrix} \underline{\alpha} \end{aligned} \tag{20}$$

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

6.2 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 used for each AR model are

$$\begin{aligned} \vec{y}_{long} &= \begin{bmatrix} X_d^{dhf} & \dot{X}_d^{dhf} & \theta_d & Z_d^I & \dot{Z}_d^I \end{bmatrix}^{\intercal} \\ \vec{y}_{lat} &= \begin{bmatrix} Y_d^{dhf} & \dot{Y}_d^{dhf} & \phi_d & \psi_d \end{bmatrix}^{\intercal} \end{aligned} \tag{21}$$

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 help the AR predictions. Including all outputs in a single vector, however, was found to cause less reliable predictions at times.

In Eq. (21) X_d, Y_d, $\dot{X}_d$, and $\dot{Y}_d$ are expressed in the averaged deck heading frame. This frame is defined by 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 heading frame is computed as

$$\begin{bmatrix} X_d^{dhf} \\ Y_d^{dhf} \end{bmatrix} = \begin{bmatrix} cos(\bar{\psi}_d) & sin(\bar{\psi}_d) \\ -sin(\bar{\psi}_d) & cos(\bar{\psi}_d) \end{bmatrix} \begin{bmatrix} X_d^I \\ Y_d^I \end{bmatrix} \tag{22}$$

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.

To give an idea of prediction accuracy at various times in the future, Fig. 4 shows deck heave predictions at 0.5, 1.3. and 2 seconds ahead. These predictions were produced with the 2H deck data. Results with other data sets are similar, though a small increase in prediction error was observed when moving from moderate sea states (2R and 2H data) to higher sea states (3R and 3H data). Looking at Fig. 4, we see that the prediction is very accurate at 0.5 seconds. At 1.3 seconds, the prediction is much less accurate but still captures general trends in the data. By 2 seconds, the predictions are increasingly inaccurate and generally under predict deck motion by a considerable amount. Despite these inaccuracies at longer prediction horizons, the deck predictions proved to help the autonomy algorithm.

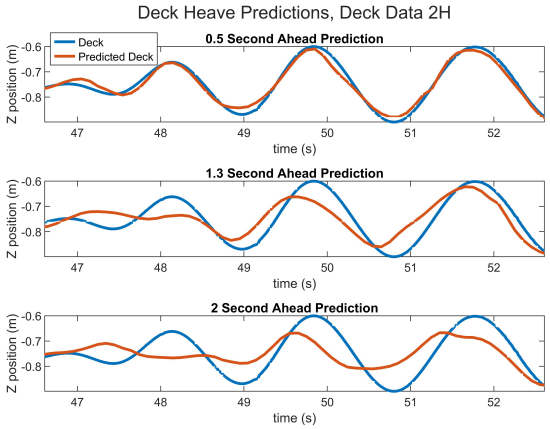

Figure 4: Deck Z^I predictions at 0.5, 1.3, and 2 sec.

7 TRAJECTORY GENERATION

The trajectory generation algorithm utilizes quadratic programming (QP) solvers for real time landing path generation. Here the MATLAB® function "mpcActiveSetSolver" was used to solve the optimization problem. The trajectory generator provides inertial X, Y, and Z position command inputs to the controller's outer loop command filters. Heading commands are handled separately without the use of an optimization solver, as this is unnecessarily complicated for the experiments this research aims to perform.

Three separate QP solvers are used for producing inertial X, Y, and Z commands. This choice was made 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. The QP trajectory generators and heading commands are discussed in detail in the remainder of this section.

7.1 Discrete Time Model

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

$$\begin{aligned} G_X(s) &= \frac{\omega_{X,cf}^2}{s^2 + 2\zeta\omega_{X,cf}s + \omega_{X,cf}^2} e^{-\tau_{x,y}s} \\ &\downarrow \\ \vec{x}_{X,k+1} &= A_X\vec{x}_{X,k} + B_X u_{X,k-\tau_d} \\ \vec{y}_{X,k} &= C_X\vec{x}_{X,k} + D_X u_{X,k-\tau_d} \end{aligned} \tag{23}$$

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.

7.2 Quadratic Program Transcription

The standard form for a QP problem is written as

$$\begin{aligned} J &= \frac{1}{2}\bar{U}^T H \bar{U} + F^T \bar{U} \\ s.t. \quad & A_c \bar{U} \leq b_0 \end{aligned} \tag{24}$$

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 that 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.

7.2.1 Future Output and Jerk Calculation

We start by rolling out the output vectors over the prediction horizon. Consider a generic LTI system (delays will be included in section 7.2.6):

$$\begin{aligned} \vec{x}_{k+1} &= A\vec{x}_k + Bu_k \\ \vec{y}_k &= C\vec{x}_k + Du_k \end{aligned} \tag{25}$$

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

$$\begin{aligned} \vec{y}_0 &= C\vec{x}_0 + Du_0 \\ \vec{y}_1 &= C\vec{x}_1 + Du_1 = CA\vec{x}_0 + CBu_0 + Du_1 \\ \vec{y}_2 &= C\vec{x}_2 + Du_2 = CA^2\vec{x}_0 + CABu_0 + CBu_1 + Du_2 \\ &\vdots \\ \vec{y}_N &= CA^N\vec{x}_0 + CA^{N-1}Bu_0 + CA^{N-2}Bu_1 + \cdots + Du_N \end{aligned} \tag{26}$$

where N represents the number of points in our prediction horizon. For convenience we write Eq. (26) in matrix form:

$$\bar{Y} = \hat{C}\vec{x}_0 + \hat{D}\bar{U} \tag{27}$$

where we use the notation

$$\bar{Y} = \begin{bmatrix} \vec{y}_0 \\ \vec{y}_1 \\ \vdots \\ \vec{y}_N \end{bmatrix} \quad \bar{U} = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{bmatrix} \quad \hat{C} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^N \end{bmatrix}$$

$$\hat{D} = \begin{bmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CAB & CB & D & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ CA^{N-1}B & CA^{N-2}B & CA^{N-3}B & \cdots & D \end{bmatrix} \tag{28}$$

Since our discrete state space models are determined by our 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. (27) and (28) are 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 makes sense to penalize and constrain instantaneous jumps in commanded acceleration.

In order to penalize and constrain jerk, we approximate jerk by back differencing the acceleration output. That is,

$$j_k = \frac{a_k - a_{k-1}}{\Delta t} \tag{29}$$

where Δt is the time step used when discretizing the model. In this paper this back difference is what the term "jerk" refers to. With this definition and the expressions in Eqs. (27) and (28), we can solve for the jerk over the full time horizon as

$$\bar{Y}_\Delta = \hat{C}_\Delta \vec{x}_0 + \hat{D}_\Delta \bar{U} \tag{30}$$

where we use the notation

$$\bar{Y}_\Delta = \begin{bmatrix} j_1 \\ j_2 \\ \vdots \\ j_N \end{bmatrix} \quad \hat{C}_\Delta = \begin{bmatrix} C_j(A-I) \\ C_jA(A-I) \\ \vdots \\ C_jA^{N-1}(A-I) \end{bmatrix}$$

$$\hat{D}_\Delta = \begin{bmatrix} C_jB - D_j & D_j & 0 & \cdots & 0 \\ C_j(A-I)B & C_jB - D_j & D_j & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ C_jA^{N-1}(A-I)B & C_jA^{N-2}(A-I)B & \cdots & \cdots & D_j \end{bmatrix} \tag{31}$$

Note that $\bar{U}$ in Eq. (31) is still defined as given in Eq. (28). Also note that C_j and D_j in Eq. (31) are calculated as shown in Eq. (32), 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.

$$C_j = \frac{1}{\Delta t} C_3 \quad D_j = \frac{1}{\Delta t} D_3 \tag{32}$$

7.2.2 Cost Function

The cost function used here is defined as

$$\begin{aligned} J = & \sum_{k=0}^{N-1} \left[(\vec{y}_{ref,k} - \vec{y}_k)^\intercal Q (\vec{y}_{ref,k} - \vec{y}_k) + u_k^\intercal R u_k \right] \\ & + \sum_{k=1}^{N-1} \left[j_k^\intercal Q_\Delta j_k \right] + u_N^\intercal R u_N + N \left[j_N^\intercal S_\Delta j_N \right. \\ & \left. + (\vec{y}_{ref,N} - \vec{y}_N)^\intercal S (\vec{y}_{ref,N} - \vec{y}_N) \right] \end{aligned} \tag{33}$$

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, separate weights are used for the terminal and running costs. This allows for emphasis to be placed on the final point of

the trajectory, which is particularly important. 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 is variable in length. When the horizon is at a maximum, 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.

Using the expressions developed in the previous subsection, we can write the cost function in the standard QP form. Expanding Eq. (33) and dropping terms not effecting the optimization we have

$$
\begin{aligned}
J = & \sum_{k=0}^{N-1} \left[\cancel{\vec{y}_{ref,k}^{\intercal} Q \vec{y}_{ref,k}} - 2\vec{y}_{ref,k}^{\intercal} Q \vec{y}_k + \vec{y}_k^{\intercal} Q \vec{y}_k + u_k^{\intercal} R u_k \right] \\
& + \sum_{k=1}^{N-1} \left[j_k^{\intercal} Q_\Delta j_k \right] + u_N^{\intercal} R u_N + N \left[j_N^{\intercal} S_\Delta j_N \right. \\
& \left. + \cancel{\vec{y}_{ref,N}^{\intercal} S \vec{y}_{ref,N}} - 2\vec{y}_{ref,N}^{\intercal} S \vec{y}_N + \vec{y}_N^{\intercal} S \vec{y}_N \right]
\end{aligned} \tag{34}
$$

Using Eqs. (27), (28), (30), and (31) we can write the above expression in the following form:

$$
J = -2\bar{Y}_{ref}^{\intercal} \bar{Q} \bar{Y} + \bar{Y}^{\intercal} \bar{Q} \bar{Y} + \bar{U}^{\intercal} \bar{R} \bar{U} + \bar{Y}_\Delta^{\intercal} \bar{Q}_\Delta \bar{Y}_\Delta \tag{35}
$$

where we define

$$
\begin{aligned}
\bar{Y}_{ref} &= \begin{bmatrix} \vec{y}_{ref,0} \\ \vec{y}_{ref,1} \\ \vdots \\ \vec{y}_{ref,N} \end{bmatrix} \qquad \bar{R} = \begin{bmatrix} R & & & \\ & R & & \\ & & \ddots & \\ & & & R \end{bmatrix} \\
\bar{Q} &= \begin{bmatrix} Q & & & \\ & Q & & \\ & & \ddots & \\ & & & S \end{bmatrix} \quad \bar{Q}_\Delta = \begin{bmatrix} Q_\Delta & & & \\ & Q_\Delta & & \\ & & \ddots & \\ & & & S_\Delta \end{bmatrix}
\end{aligned} \tag{36}
$$

Substituting the expressions for $\bar{Y}$ and $\bar{Y}_\Delta$ in Eqs. (27) and (30) into Eq. (35), expanding, and neglecting terms that do not contain the optimization variables $\bar{U}$, the following equation can be derived:

$$
\begin{aligned}
J = & \bar{U}^{\intercal} \left(\hat{D}^{\intercal} \bar{Q} \hat{D} + \hat{D}_\Delta^{\intercal} \bar{Q}_\Delta \hat{D}_\Delta + \bar{R} \right) \bar{U} \\
& + 2 \left(\vec{x}_0^{\intercal} \hat{C}^{\intercal} \bar{Q} \hat{D} + \vec{x}_0^{\intercal} \hat{C}_\Delta^{\intercal} \bar{Q}_\Delta \hat{D}_\Delta - \bar{Y}_{ref}^{\intercal} \bar{Q} \hat{D} \right) \bar{U}
\end{aligned} \tag{37}
$$

Dividing the above equation by two (which has no effect on the optimization) and comparing to the standard QP form in Eq. (24), the H and F matrices from Eq. (24) are found by inspection to be

$$
\begin{aligned}
H &= \hat{D}^{\intercal} \bar{Q} \hat{D} + \hat{D}_\Delta^{\intercal} \bar{Q}_\Delta \hat{D}_\Delta + \bar{R} \\
F &= \left(\hat{D}^{\intercal} \bar{Q} \hat{C} + \hat{D}_\Delta^{\intercal} \bar{Q}_\Delta \hat{C}_\Delta \right) \vec{x}_0 - \hat{D}^{\intercal} \bar{Q} \bar{Y}_{ref}
\end{aligned} \tag{38}
$$

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

7.2.3 Constraints

The expressions in Eqs. (27) and (30) can again be used to write the constraint equations. 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:

$$
\bar{Y}_{lim\,up} = \begin{bmatrix} \vec{y}_{lim\,up,0}^{\intercal} & \vec{y}_{lim\,up,1}^{\intercal} & \cdots & \vec{y}_{lim\,up,N}^{\intercal} \end{bmatrix}^{\intercal} \tag{39}
$$

Appealing to Eq. (27), the inequality defining the output upper bounds is then written as

$$
\hat{C}\vec{x}_0 + \hat{D}\bar{U} \leq \bar{Y}_{lim\,up} \rightarrow \hat{D}\bar{U} \leq \bar{Y}_{lim\,up} - \hat{C}\vec{x}_0 \tag{40}
$$

Doing the same for the output lower limits and the jerk upper and lower bounds, we have

$$
\begin{aligned}
-\hat{D}\bar{U} &\leq -\bar{Y}_{lim\,low} + \hat{C}\vec{x}_0 \\
\hat{D}_\Delta \bar{U} &\leq \bar{Y}_{\Delta,lim\,up} - \hat{C}_\Delta \vec{x}_0 \\
-\hat{D}_\Delta \bar{U} &\leq -\bar{Y}_{\Delta,lim\,low} + \hat{C}_\Delta \vec{x}_0
\end{aligned} \tag{41}
$$

where $\bar{Y}_{\Delta,lim\,up}$ and $\bar{Y}_{\Delta,lim\,low}$ contain the jerk upper and lower limits across the prediction horizon. Combining the inequalities in Eqs. (40) and (41) the A_c matrix and b_0 vector defining the QP constraints in Eq. (24) are found to be

$$
A_c = \begin{bmatrix} \hat{D} \\ -\hat{D} \\ \hat{D}_\Delta \\ -\hat{D}_\Delta \end{bmatrix} \quad b_0 = \begin{bmatrix} \bar{Y}_{lim\,up} - \hat{C}\vec{x}_0 \\ -\bar{Y}_{lim\,low} + \hat{C}\vec{x}_0 \\ \bar{Y}_{\Delta,lim\,up} - \hat{C}_\Delta \vec{x}_0 \\ -\bar{Y}_{\Delta,lim\,low} + \hat{C}_\Delta \vec{x}_0 \end{bmatrix} \tag{42}
$$

For the X^I and Y^I QP solvers, the position constraints were set to $\pm 1000\ m$ to effectively remove this constraint. The Z^I QP solver lower position limit was also set to $-1000\ m$ for the same reason, but the Z^I upper position limit was set to the predicted deck position at each point in the time horizon to prevent collision with the deck (this is the upper limit because down is taken as the positive Z^I direction). On all axes, the limits for the velocity and acceleration were set to $\pm 7\ m/s$ and $\pm 3.5\ m/s^2$, respectively. The jerk constraints were the same on all axes as well, but were varied for the high, medium, and low bandwidth test cases. The specific values were $\pm 9\ m/s^3$, $\pm 7\ m/s^3$, and $\pm 5\ m/s^3$ for the high, medium, and low bandwidth cases, respectively. The jerk limits were reduced with tracking bandwidth to help guard against the QP solver overcompensating for the reduced command filter frequency by over driving the command filter input. Over driving the command filter would not cause issues in the model scale setup with artificially degraded maneuverability, but for the full scale aircraft it could cause issues like saturated controls. The specific values were just selected as a best guess based off experience with the system, however. This was done due to lack of published information on jerk limitations for full scale aircraft.

7.2.4 Prediction Horizon and Reference Path

The maximum horizon length was set to $N_{max} = 30$. With a time step of 0.1 seconds for each QP solver, this equates to a 3 second prediction horizon. Limiting the horizon to 3 seconds kept the number of optimization variables and constraints to satisfy reasonable, but the prescribed time to perform the landing (described in section 7.2.5) 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.

When $N > Nmax$, an intermediate reference trajectory is prescribed for use in the cost function (Eq. (33)) by drawing a straight, constant velocity line toward the deck. For each axis, the desired velocity is calculated as the difference between the predicted deck position at the max horizon time

and the current UAV position, all divided by the time remaining. For example, for the X^I QP solver we have

$$v_{des} = \frac{X^I_{dp,N_{max}} - X^I_{uav}}{t_{rem}} \tag{43}$$

where $X^I_{dp,N_{max}}$ is the deck X^I prediction at N_{max} time steps in the future and t_{rem} is the time left until touchdown. The output reference values for the X^I QP solver (when $N > N_{max}$) are then given as

$$\vec{y}_{Xref,k} = \begin{bmatrix} X^I_{uav} + v_{des}t_k & v_{des} & 0 \end{bmatrix}^\intercal \tag{44}$$

where t_k is the time into the prediction horizon. For example, $t_0 = 0\,s$ and $t_{N_{max}} = 3\,s$.

When the prediction horizon covers the time remaining until touchdown ($N \leq Nmax$), different reference outputs are prescribed. For the X^I and Y^I trajectories, an identical process is used to set $\vec{y}_{ref}$. 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:

$$\vec{y}_{Xref,k} = \begin{bmatrix} X^I_{dp,N} & \dot{X}^I_{dp,N} & 0 \end{bmatrix}^\intercal \tag{45}$$

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

$$\vec{y}_{Xref,N} = \begin{bmatrix} X^I_{dp,N} & \dot{X}^I_{dp,N} & a^I_{Xdes,N} \end{bmatrix}^\intercal \tag{46}$$

The terminal reference acceleration is used to motivate matching attitude with the ship deck at touchdown. Here the desired X^I and Y^I accelerations at touchdown are computed by assuming the vehicle heading frame accelerations can be described by Eq. (4) and that the UAV will closely match deck heading at touchdown:

$$\begin{bmatrix} a^I_{Xdes,N} \\ a^I_{Ydes,N} \end{bmatrix} = \begin{bmatrix} cos(\psi_{dp,N}) & -sin(\psi_{dp,N}) \\ sin(\psi_{dp,N}) & cos(\psi_{dp,N}) \end{bmatrix} \begin{bmatrix} -g\theta_{dp,N} \\ g\phi_{dp,N} \end{bmatrix} \tag{47}$$

While for a full scale aircraft with significant wind the approximation in Eq. (4) may not be valid, any mapping (linear or nonlinear) from aircraft attitude 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. To formulate attitude matching directly, however, a nonlinear MPC method like that presented in [4] would be needed. For our model test this simplified approximation is reasonably accurate, though.

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 similarly to what is shown in Eq. (44). 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 5 cm above the deck:

$$\vec{y}_{Zref,N} = \begin{bmatrix} Z^I_{dp,N} + Z^I_{offset,N} & \dot{Z}^I_{dp,N} & 0 \end{bmatrix}^\intercal \tag{48}$$

The 5 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.

7.2.5 Land Time

Since the QP cost function does not include time as an optimization variable, a length of time to perform the landing must be assigned prior to running the solver. Here, an initial land time was chosen based off the tau guidance method described in [9], where land time was chosen to limit accelerations during the maneuver. Using X^I guidance as an example, the resulting equation is

$$t_{land,X} = 2.888\sqrt{\frac{|X^I_{uav,0} - X^I_{d,0}|}{\ddot{X}^I_{max}}} \tag{49}$$

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 we are not using the tau guidance method in this work, 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

$$t_{land} = \max \begin{pmatrix} 1.444\sqrt{\frac{|X^I_{uav,0} - X^I_{d,0}|}{\ddot{X}^I_{max}}} \\ 1.444\sqrt{\frac{|Y^I_{uav,0} - Y^I_{d,0}|}{\ddot{Y}^I_{max}}} \\ 1.444\sqrt{\frac{|Z^I_{uav,0} - Z^I_{d,0}|}{\ddot{Z}^I_{max}}} \end{pmatrix} \tag{50}$$

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

A flag can be set in the trajectory generation algorithm to allow updates to the initial land time based off the predicted deck states. If the update flag is set, the land time can be adjusted when there is between 1.5 and 3 seconds left in the maneuver. The lower bound was set at 1.5 seconds because changing the landing time, and therefore location, with less time than this remaining was found to leave insufficient time for the UAV to react. The 3 second upper limit was chosen because we produce deck predictions out to 3 seconds in the future. When in this window, the algorithm can shorten the remainder of the maneuver by up to 0.3 seconds or extend the land time out to 3 seconds. For example, if there is 1.7 seconds remaining in maneuver, then the land time update logic will assign a cost to each candidate land time from 1.4 seconds to 3 seconds in the future in 0.1 second increments. The time point in this interval with the lowest cost is then selected as the new land time. The cost for each time point in the considered landing window is given by

$$\begin{aligned} J_k = w_Z \left(Z^I_{dp,k} - \bar{Z}^I_{dp} \right) - w_{\dot{Z}} \dot{Z}^I_{dp,k} + w_\phi |\phi_{dp,k}| \\ + w_\theta |\theta_{dp,k}| + w_{\Delta t} |t_{land,k} - t_{land}| \end{aligned} \tag{51}$$

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 for the first two terms Z is positive down. In the last term, $t_{land,k}$ represents what the new land time would be if the candidate time point were selected for landing. This last term is included to penalize changes from the current land time, so that large changes in land time are not made due to small changes in the other terms. The first two terms in Eq. (51) 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. The weighting factors were set to $w_Z = 1$, $w_{\dot{Z}} = 0.5$, $w_{\Delta t} = 0.15$, and $w_\phi = w_\theta = 0.06 \cdot 180/pi$.

7.2.6 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. These stored inputs then predetermine the first $\tau_d - 1$ future outputs of the discrete time model, meaning that the first $\tau_d - 1$ outputs in the prediction horizon are no longer free and should be dropped from the cost function. To achieve this, we simply use the stored τ_d past inputs to simulate our discrete model from the current state to $\tau_d - 1$ sample periods into the prediction horizon:

$$\begin{aligned} \vec{y}_0 &= C\vec{x}_0 + Du_{-\tau_d}\ ,\ \vec{x}_1 = A\vec{x}_0 + Bu_{-\tau_d} \\ \vec{y}_1 &= C\vec{x}_1 + Du_{1-\tau_d}\ ,\ \vec{x}_2 = A\vec{x}_1 + Bu_{1-\tau_d} \\ &\vdots \\ \vec{y}_{\tau_d-1} &= C\vec{x}_{\tau_d-1} + Du_{-1}\ ,\ \vec{x}_{\tau_d} = A\vec{x}_{\tau_d-1} + Bu_{-1} \end{aligned} \tag{52}$$

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. (24) is set to $N - \tau_d$. What this does is fill the first $\tau_d - 1$ outputs into the prediction horizon $\vec{y}_0 \cdots \vec{y}_{\tau_d-1}$ by propagating the model forward. Then the QP solver computes the optimal inputs $u_0 \cdots u_{N-\tau_d}$ that define the outputs $\vec{y}_{\tau_d} \cdots \vec{y}_N$.

7.3 Heading Commands

For the tests performed here the UAV begins the landing sequence from behind the ship deck, and the deck yaw primarily oscillates around a mean value. It was therefore found sufficient to simply command the UAV heading to follow the deck heading passed through a low pass filter with a rate limit of 50 deg/s. The rate limit was included because when initiating the autonomy sequence the UAV heading may have a large offset from the deck heading. The UAV would first be commanded to align heading before proceeding to start the test, and the rate limit guarded against overly aggressive yaw commands due to this initial step input.

8 EXPERIMENTAL RESULTS

A time history of a landing approach to the virtual deck with deck case 3H, high bandwidth control, and no degradation of deck predictions is shown in Fig. 5. Note that "UAV Ideal" in Fig. 5 refers to the theoretical ideal response that would be achieved by the EMF controllers with a perfect inversion model, which is closely tracked by the UAV. From this landing trace, we see that the UAV begins the approach from a hover located about 3.25 m above and 4 m behind the mean deck position with heading aligned to the deck. The command to land is given around 20 seconds, and the UAV performs what it appears would have been a successful landing. Aggressive deck heave motion occurred during landing, hitting peak deviations from the mean of about 0.3 m, which corresponds to slightly over 4 m at full scale. Due to the inclusion of deck forecasting, the UAV does not follow deck heave motions until it attempts to match deck state at the end of the trajectory. Additionally, we can see that the prediction algorithm was able to identify the upward heave motion occurring at the initially chosen land time, allowing the trajectory generator to delay landing to a time with easier to match deck motion. At the updated land time, the relative vertical velocity is approximately 0.1 m/s. The X^I and Y^I position and velocities are also matched within 0.1 m and 0.1 m/s, though there is little deck motion in surge. The deck motion is also fairly mild in pitch and roll for this case, but the effects of the terminal acceleration weights on pitch and roll attitude matching can still be observed. The commanded pitch and roll angles begin to approach those of the deck at the end of the trajectory, giving a slight decrease in relative attitude error. This was found common for the heave dominant deck motion cases, but there were cases with the 3R deck data where the algorithm was unable to match attitude within 10 degrees while also reasonably matching position and velocity. The deck yaw motion is negligible, which is the case across all deck data sets.

The time history shown in Fig. 5 is representative of algorithm performance for the majority of test cases. For some of the more aggressive 3H and 3R deck cases, though, the algorithm did not perform well, even with high bandwidth control and no degraded forecasting. Additionally, the land time updates worked well in most cases, but in some instances produced land times that were more difficult than the initially chosen time. It was hypothesized that this could be improved by checking if the QP algorithm predicts the UAV will be able to match deck state at landing prior to updating the land time. Due to inaccuracies in long term predictions, though, the QP algorithm was found to always predict that it would match deck state at touchdown until approximately 0.5 seconds before landing. The land time update method used here managed to avoid land times with large upward heave in many cases only because the general trends in deck motion tend to be captured in long term predictions. Overall, flight tests showed the algorithm to be capable of performing landings in relatively challenging conditions though the land time update method could be improved. This is sufficient for the purpose of this research.

The position and velocity error root mean square (RMS) and maximum for each test case are reported in Fig. 6. Note that when analyzing the X^I and Y^I errors only the test cases with degraded X^I and Y^I command filter bandwidth are compared to the baseline (test case 1), as degraded Z^I tracking bandwidth did not have significant effect on X^I and Y^I errors. The same is true for the Z^I errors, where only cases with degraded vertical bandwidth are compared to the baseline. Also note that in Fig. 6 there is a vertical position error reported, as the tests were run until the designated land time. The Z^I position error is therefore taken as the vertical distance between the designated land point and the UAV at the chosen land time. There were a small number of cases, however, where the UAV would have made early contact with the deck. For these cases, the actual land time was taken as the time of contact. These cases were dropped

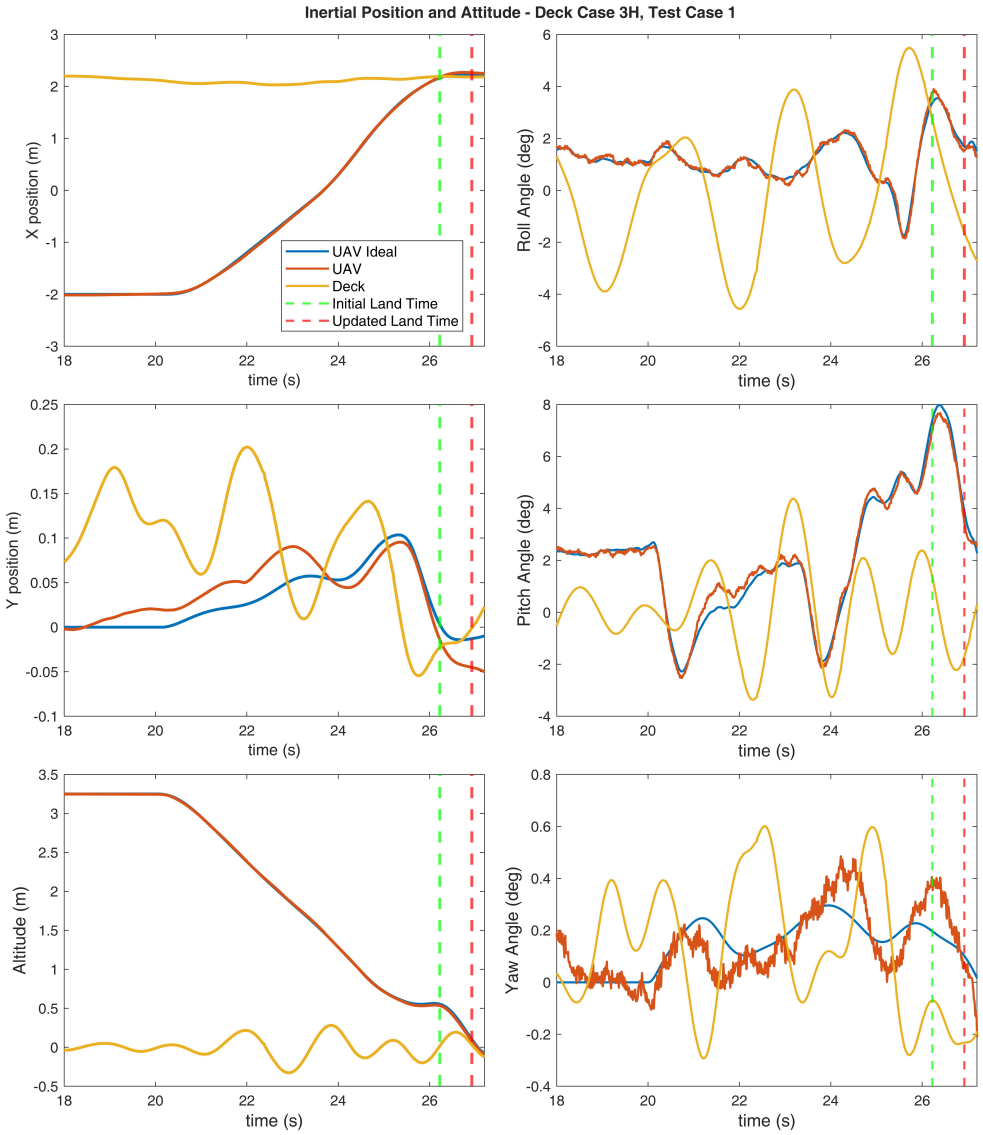

Figure 5: Landing time history.

when calculating Z^I position error RMS, as they have low position error that skews the RMS due to the land time being taken as when the UAV intersected the deck. These cases were not dropped when calculating velocity error RMS, however, as early landing can result in high relative velocity.

Looking at Fig. 6, we see that both the RMS and maximum of X^I velocity errors trend downward with reduced command filter bandwidth. This is also true for the X^I position error, though less pronounced in RMS where the actual values do not differ by more than a few centimeters for the same deck case. A notable feature of this data is the larger drop in performance from medium to low X^I and Y^I command filter bandwidth. The performance difference when moving from high to medium bandwidth, however, is minimal. This shows that even though the surge motion of the deck was very mild, the degraded mobility still had a clear effect on performance. It should be noted, though, that this behavior cannot be solely attributed to the reduced command filter bandwidth since jerk limits were lowered along with bandwidth. Several landings that resulted in larger X^I position and/or velocity errors for the low bandwidth case encountered the jerk limits for that case, but not for the medium and high bandwidth cases. This highlights the fact that the QP algorithm will try to compensate for reduced bandwidth by over driving the command filter input. For testing autonomy algorithms at scale, determining realistic jerk limits or relating limits to full scale control input saturation may therefore be important when assessing aircraft with reduced mobility.

The Y^I position and velocity error statistics shown in Fig. 6 do not show any clear trends across test cases. Additionally, the Y^I errors are generally found to be larger than for X^I and Z^I. It is believed that this outcome was largely due to poor Y^I deck position and velocity predictions. The algorithm generally struggled to predict these states much more so than for the other position and velocity degrees of freedom, and in some cases produced predictions that were simply a delayed value of the current deck state. Poor Y^I predictions were not observed when applying the prediction algorithm to the full scale SCONE data, however. It was found in [12] that AR prediction performance decreased with higher frequency ship motion, and this may be affecting the model scale Y^I predictions in a way that is not proportional

to reduction in scale. The prediction accuracy was adequate in other degrees of freedom, though, so other contributing factors are likely at play. Future work attempting landings on a small scale landing craft will help clarify if this is a consistent issue or a product of the data processing used to produce model scale time histories for this work.

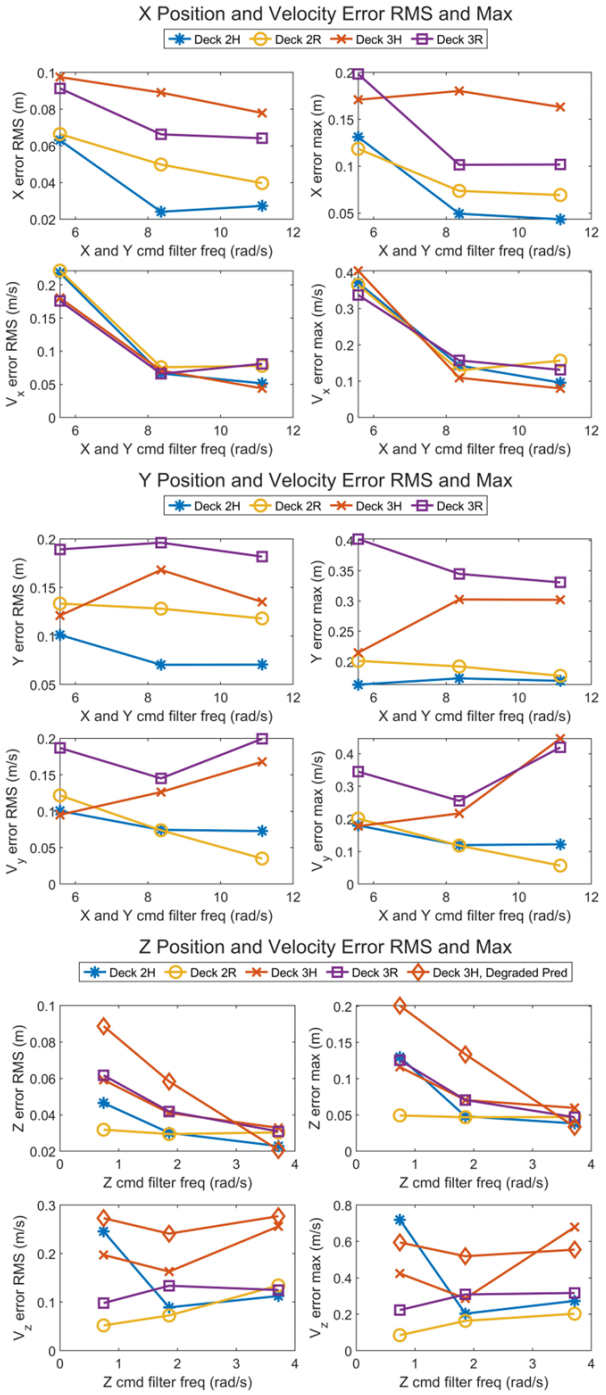

Figure 6: Position and velocity landing error statistics.

The Z^I position error statistics in Fig. 6 do not reveal any significant trends for the 3R, 3H, or 2R deck data without degraded predictions. The 3R and 3H deck data show slightly higher Z^I position errors for the low bandwidth case, but the difference in RMS error is very small. Additionally, the Z^I velocity errors were largely the same regardless of bandwidth for the 3R and 3H deck data, but with one outlier producing a high max velocity error for high bandwidth with 3H deck data. For the 2R deck data, performance was relatively consistent regardless of control configuration. For the 2H deck data, the lowest bandwidth case appears to have reduced performance in both position and velocity matching, but this is in part due to a QP solver failure that occurred during one run of the low bandwidth 2H deck case. This run resulted in the max errors shown in Fig. 6 for Z^I velocity and position with 2H deck data. This was one of two QP solver failures that occured during the 230 flight tests, both of which were with the lowest vertical bandwidth and jerk limits. More restrictive jerk limits make the algorithm more likely to fail, and it appears that reduced vertical mobility is more likely to cause a failure than reduced horizontal mobility due to the inclusion of hard position constraints defined by the deck.

Looking at the 3H deck case with degraded predictions and land time updates off, the high bandwidth configuration performs similarly to the 3H deck case with non-degraded deck predictions. The medium and low bandwidth cases with degraded predictions both show a large increase in position error RMS and max as bandwidth decreases, but the velocity errors remain relatively constant. Looking at data from individual runs, it was found that the QP algorithm would attempt to match position and velocity for high bandwidth cases. For low BW cases with degraded predictions, the QP algorithm would shift to matching velocity and accepting more position error. To demonstrate this, Fig. 7 shows the altitude during the final seconds of landing for low, medium, and high vertical bandwidth and degraded deck motion predictions. The low and high bandwidth cases here are assigned a land time within about 0.15 seconds of each other. At that assigned land time the high bandwidth case matches position and velocity well, but the low bandwidth case takes a larger position error. A similar result is observed for the medium bandwidth case. Looking at the QP jerk outputs plotted in Fig. 7, we see that this is clearly impacted by the jerk limitations. In all three cases the jerk limits are hit by the QP solver, with the low bandwidth case spending the largest amount of time on the limit. This reaffirms that a method for determining realistic jerk limits may be important when performing model scale testing for a less mobile aircraft.

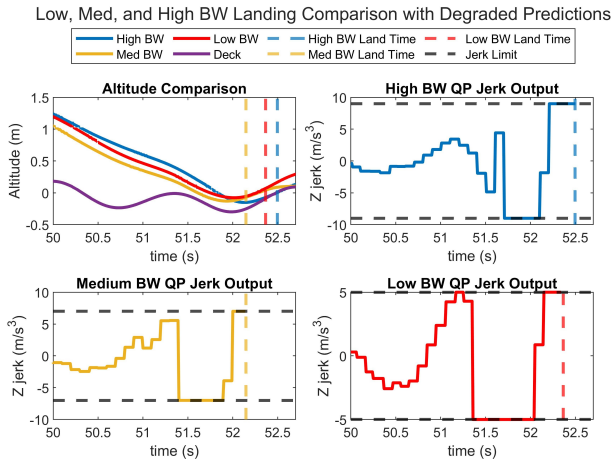

Figure 7: Comparison with degraded predictions.

While it is expected that reduced bandwidth and jerk limits will negatively impact robustness to poor deck predictions, it is interesting that such a significant reduction in perfor-

mance is seen for the low and medium bandwidth cases with the predictions cut off at 1.3 seconds. Referring back to Fig. 4, it was observed that the deck predictions start to become much less accurate around and after 1.3 seconds. Despite this, including longer term predictions appears to make the algorithm more robust to reduced mobility, as the low and medium bandwidth cases performed considerably worse with deck predictions degraded and land time updates turned off than with unaltered predictions.

9 CONCLUSIONS

This paper presented the development of an autonomous ship landing algorithm suitable for use in model scale experiments. The algorithm was tested in 230 flight tests to a virtual landing deck, with model scale deck data derived from simulations representative of a DDG-51 type ship. Both the ship data and autonomy algorithm parameters were adjusted to model scale via the proposed scaling method, which uses Froude scaling based on vehicle mass. The autonomy algorithm features EMF controllers, QP trajectory generation, and AR models for deck motion prediction. During testing, the EMF controller and QP algorithm jerk limits were used to progressively degrade aircraft mobility. Additional tests were performed with degraded deck motion predictions and no land time updates. From the flight test results, the following observations can be made:

1. The proposed algorithm is capable of reliably performing landings in scaled moderate sea states. In high sea states the algorithm performs well the majority of the time, but cases with large landing errors were also observed. This provides sufficient capabilities for the model scale tests discussed here.
2. If using model scale testing to gain insight on a full scale aircraft with low control authority, determining realistic jerk limits or relating jerk/output limits to full scale control saturation would be important. This was exemplified here, where the QP algorithm compensates for reduced bandwidth by over driving the command filter input until the jerk limits are encountered. If proper limitations are not enforced, then an optimal control algorithm will compensate for reduced bandwidth at model scale in a way that does not realistically translate to the full scale aircraft.
3. Despite inaccuracies in long term deck motion predictions, including long term predictions in the autonomy algorithm increased robustness to reduced mobility. This indicates that the trends in the data are captured well enough for long term predictions to still be useful for path planning. This was observed through the relative drop in performance for the low and medium bandwidth cases when deck predictions were degraded, as opposed to left unaltered.
4. The AR prediction algorithm performed much more poorly for Y^I deck position and velocity than other deck states. This was not observed on full scale deck data, and it is unclear if this is due to the prediction algorithm scaling poorly with increased deck frequency or if it is a product of the data processing used to derive model scale deck data. Future tests with a physical landing craft will clarify if this is a problem when landing on an actual model scale vessel.

ACKNOWLEDGEMENTS

This work was sponsored 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 ONR, the U.S. Navy, or the U.S. Government.

REFERENCES

[1] E L Tobias, F C Sanders, and M B Tischler. “Full-Envelope Stitched Simulation Model of a Quadcopter Using STITCH”. In: *AHS International 74th Annual Forum & Technology Display*. Phoenix, Arizona: AHS, 2018.

[2] Christina M. Ivler et al. “System Identification Guidance for Multirotor Aircraft: Dynamic Scaling and Test Techniques.” In: *Journal of the American Helicopter Society* 66.2 (2021), pp. 1–16. ISSN: 00028711.

[3] Christina M. Ivler et al. “Development and Flight Validation of Proposed Unmanned Aerial System Handling Qualities Requirements”. English. In: *Journal of the American Helicopter Society* 67.1 (2022).

[4] Jintasit Pravitra. “Shipboard UAS Operations with Optimized Landing Trajectories”. PhD dissertation. The Pennsylvania State University, 2021.

[5] William B Greer and Cornel Sultan. “Shrinking horizon model predictive control method for helicopter–ship touchdown”. In: *Journal of Guidance, Control, and Dynamics* 43.5 (2020), pp. 884–900.

[6] Joseph F Horn et al. “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). 2015, pp. 864–877.

[7] Alan Schwartz. “Systematic characterization of the naval environment (scone)–standard deck motion data for a generic surface combatant”. In: *Memorandum from Office of Naval Research and Naval Surface Warfare Center-Carderock Division* (2015).

[8] Junfeng Yang. “Autonomous Control Modes and Optimized Path Guidance for Shipboard Landing in High Sea States”. PhD dissertation. The Pennsylvania State University, 2018.

[9] Christopher M. Hendrick et al. “Scaled Experiments in Flight Control Design for Autonomous Landing in High Sea States”. In: *AIAA AVIATION 2022 Forum*. DOI: 10.2514/6.2022-3280. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2022-3280. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2022-3280.

[10] Chris L Blanken et al. *Proposed Revisions to Aeronautical Design Standard-33E (ADS-33E-PRF) Toward ADS-33F-PRF*. Tech. rep. CCDC AvMC, 2019.

[11] Rushabh Patel et al. “Gpc-based deck motion estimation for autonomous ship deck landing of an unmanned aircraft”. In: *AIAA Scitech 2021 Forum*. 2021, p. 1814.

[12] Hua Jiang et al. “Scale effects in AR model real-time ship motion prediction”. In: *Ocean Engineering* 203 (2020), p. 107202.