System Identification-based Fault Detection and Model-Based Control of an Uncrewed Aerial System

Robert Bowes

Submitted to the graduate degree program in Department of Aerospace Engineering and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master's of Science.

Dr. Shawn Keshmiri, Chairperson

Committee members

Dr. Mark Ewing

Dr. Weizhang Huang

Date defended: May 11, 2023

The Thesis Committee for Robert Bowes certifies
that this is the approved version of the following thesis:

System Identification-based Fault Detection and Model-Based Control of an Uncrewed Aerial System

Dr. Shawn Keshmiri, Chairperson

Date approved: May 11, 2023

Abstract

As the wide-scale use of uncrewed aerial systems, or UAS, proceeds towards civilian airspace, guarantees of operational safety over the entire flight envelope become more critical. However, public opinion is mixed on the general safety of autonomous systems, especially within close proximity to densely populated spaces. Because the opinion of the general public is of paramount importance to the general adoption of this technology, focus must be placed on the development of dependable systems before operations can even be considered.

With the stringent payload capacity and size constraints of UAS, advanced control algorithms can fill the gap caused by the reduced redundancy expected of UAS. One such branch of control algorithms is active fault tolerant control, which uses knowledge of failures to update the control loop and maintain performance. However, most works on active fault tolerant control present in literature trivialize fault detection and diagnosis to hard to achieve assumptions of perfect knowledge of failures or the presence of actuator feedback systems, which are currently not widely used. These weight and cost penalties increases the appeal of a software-based system for fault detection, which solely uses measurements from sensors already necessary for autonomous flight, completely eliminating the disadvantages of hardware-based systems.

As system dynamics change during failures, system identification is an intuitive way to measure changes in dynamics to estimate actuation failures, especially flight-critical actuators which have a large impact on system performance. In this work, a time domain system identification method based on the eigensystem realization algorithm (ERA) and observer Kalman filter identification method (OKID) is used. These methods used in tandem can generate a model capturing the maximum input-output information from any arbitrary set of data, providing a more generalizable approach than is possible with most system identification

techniques. Taking advantage of available flight tests data which contained aileron and rudder failure cases, this combination is applied to flight test data, and the models generated are used to investigate the change in dynamics present during failures, with changes in model parameters correlated to failures.

Knowledge of failures can be leveraged by the control loop by an adaptive flight controller if the dynamic model of the aircraft is available after failure. This research presents applications of two different control algorithms capable of adapting to changing dynamic models, model predictive control (MPC) and dynamic inversion. This work adds a non-traditional control increment term, Δu, to the MPC cost function to directly punish large control rates to prevent oscillations and out of phase behavior. The presented dynamic inversion flight controller allows for a more direct approach to system performance design. Its theory implies the ability to maintain performance in the case of faults as long as adequate updates to the model are provided and enough overall control authority is available to make the commanded maneuvers.

Typical approaches to simulation validation of fault tolerant control methods are based heavily in assumptions on how failures impact the dynamic model. In direct contrast to this, validation models for this work are based on collected failure flight test data and generated using a Monte Carlo-based technique. This method varies model parameters based on a given normal distribution to find a dynamic model which best fits the measured aircraft states. Nominal simulations results show a similar level of performance of both control algorithms, while failure simulations highlight the satisfactory performance of both controllers as well as the different benefits offered by each algorithm.

This work builds off of a previously published conference paper [1] which presents preliminary results of the fault detection method with one data set and only performs failure validation simulation for the LQR and dynamic inversion controllers.

Acknowledgements

I would like to start by thanking my advisor, Dr. Shawn Keshmiri. His leadership by example has taught me lessons about dedication and motivation which I will apply in my future career. Through my time working under him at the Flight Research Lab, I have had the opportunity to not only grow as an engineer, but also as a leader and a person. He placed his trust in me when I joined his team and maintained it over my last two years of learning. For this, I am truly grateful.

I would like to thank U.S. Air Force, Department of Defense, Federal Aviation Administration, and the National Science Foundation for their funding. I was able to gain a wide variety of experience, like flight testing, systems engineering, control design, and manufacturing, through working on multiple projects. This breadth of experience is something I treasure about my graduate experience and will carry with me in my future career.

I would like to thank my fellow students at the flight research lab, namely Dr. Mozammal Chowdhury, Aaron McKinnis, Jeffrey Xu, Hady Benyamen, and Justin Clough. Getting to work together with them over the last two years was a privilege, and I greatly appreciate the experiences we shared at the FRL.

Finally, I would like to thank my mom and dad for their constant support in my academic life. They taught me about the importance of education at a young age and have always pushed me to be a better student. Their countless sacrifices put me on the life trajectory I am on now, and I will always be grateful for the opportunities they gave me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Due to large advancements in technologies such as processing, sensors, and electric propulsion [2], uncrewed aerial systems (UASs) have grown increasingly popular among civilian, military, and industrial applications. Although UASs have been in use for several decades, their widespread industrial adaptation outside of military and defense has grown exponentially in recent years. UASs are highly versatile and flexible in their capabilities, with diverse uses like photography, surveillance, mapping and surveying, search and rescue, and delivery among many other uses. This adaptability, along with reduced cost of operation compared to non-autonomous methods and a steadily increasing capability of onboard computing and sensors, suggests that UASs will likely play an integral part in solving many societal problems.

One such issue is congested roads due to heavy traffic, specifically in urban areas. With estimates of 68% of global population living in urban areas by 2050 [3], traffic in both intercity streets and urban highways will continue to grow. Although it is regarded as a part of life, traffic congestion has large negative impacts on multiple facets of society. According to the Texas A&M Transportation Institute, the average urban commuter sits in traffic for 54 hours a year, and urban traffic costs the US economy a total of $179 billion each year [4]. Long commutes are also correlated with several physical and social health risks, such as increased risk of heart disease, increased feelings of loneliness due to less time for social activities, and increased stress levels [5–7].

One solution proposed to help alleviate urban road congestion is urban air mobility

(UAM). Urban air mobility refers to the use of UASs for the transport of passengers and cargo in urban areas. Urban air mobility offers unique promise to relieve congestion in both urban roads and highways. UAM could alleviate road congestion by reducing the number of vehicles moving passenger and cargo on the road. Due to the increased popularity of on-demand delivery services, the use of UASs for last mile delivery operations could eliminate a significant amount of traffic from inner city streets.

Despite the promise of UAM, its adoption faces many challenges. In addition to requirements for new infrastructure and established regulations, the safety of integrating autonomous vehicles into civilian airspace is a prevalent public concern. In 2020, the urban air mobility group at Airbus conducted a survey of 1500 people in four areas where new transportation methods are likely to be adopted [8]. However, only 41% of people surveyed deemed urban air mobility to be safe or very safe. New innovations, especially those which affect the safety of the general public, are scrutinized heavily in the court of public opinion and require pristine operational histories to cultivate trust in a wide user base. Therefore, a large emphasis must be placed on the safety of these vehicles in their design. Contrary to other aerospace systems, traditional methods of safety via redundancy may not be possible in these vehicles due to size, weight, and power constraints. These restraints require alternate methods to have comparable safety limits seen in traditional aircraft.

One method to increase the level of safety is fault tolerant control (FTC), a method of adaptive control with the goals of:

- for small faults, maintaining the nominal level of performance.
- for large faults, degrading performance in a more graceful manner than a base controller.

FTC can be separated into two subcategories: passive fault tolerant control and active fault tolerant control. Passive fault tolerant control models the system with a possible range of uncertain failures and takes an approach similar to traditional robust control to guarantee a certain level of performance. Active fault tolerant control instead uses a fault detection

and diagnosis (FDD) algorithm to monitor failures and update the controller accordingly. Still, due to the size, weight, and power requirements of adding a hardware-based system for FDD, the possibility of a software-based, data-driven form of FDD is appealing for this application. This research provides the mathematical basis and discusses the preliminary results and limitations of a system identification-based fault detection and diagnosis algorithm. Investigation is also made into the ability of several control architectures to adapt to onboard adverse conditions given adequate model updates.

Chapter 2

System Identification-based Fault Detection and Model-based Control of an Uncrewed Aerial System

Abstract

As the wide-scale use of uncrewed aerial systems proceeds towards civilian airspace, guarantees of operational safety over the entire flight envelope become more critical. In this paper, a fault detection and diagnosis method based on system identification is proposed and the adaptation capabilities of various control algorithms given knowledge of failures are investigated. For fault detection and diagnosis, a reduced order model is estimated from measurements based on relevant dynamic modes for a defined window of flight data. This system identification algorithm allows faults to be estimated via tracking of calculated model parameters. These failures will be parameterized and fed into the control loop to update the dynamic model and maintain dynamic model accuracy in the presence of faults. The initial results of the system identification algorithm as well as performance of the discussed controllers for nominal models is shown. The adaptivity of different controllers to failure is investigated through the use of validation models derived from collected flight test data for simulated failure cases.

2.1 Introduction

Uncrewed aerial systems (UAS) have begun to break into the national airspace with many industries looking to employ these systems in unique and challenging scenarios. These complex

situations are frequently described as Advanced Aerial Mobility (AAM) and can range from long line linear infrastructure inspection to agricultural spraying and mapping. Within these situations, it is crucial for the UAS to be able to tightly follow given flight trajectories since large deviations will decrease endurance and, in cases like mapping and spraying, require repeating the flight altogether. This tracking problem is greatly exacerbated when the scope is narrowed from large open areas to a spatially constrained areas, like that of the modern city.

When UAS are asked to operate in urban environments, the nomenclature shifts and is commonly referred to as Urban Air Mobility (UAM). UAM presents many unique tracking challenges such as static and dynamic obstacle avoidance, non-linear wind fields, and poor connection to GPS/GNSS [9]. A multitude of techniques have been applied to remedy these problems at the source. Poor tracking in this type of environment can lead to costly reroutes around city blocks or even collision.

Even with a base controller able to comfortably attain all given attitudes desired by the path planning, faults in UAS subsystems can cause drastic reduction in performance, with consequences ranging from slight degradation of tracking to complete loss of system. This issue is solved through the use of system redundancy in traditional aircraft systems, like transport jets. However, due to the rigid size, weight, and power constraints present in UAS, highly redundant systems may be prohibitively heavy, reducing potential use cases due to diminished endurance or payload capacity. A solution to increase operational safety without these drawbacks is the use of fault tolerant control (FTC). Fault tolerant control algorithms can be classified in two different groups: active and passive algorithms. Passive fault tolerant control (PFTC) algorithms [10–12] use a predetermined controller structure and address fault tolerance using an approach similar to robust control design through the use of expected faults and system uncertainties. Active fault tolerant control (AFTC) algorithms instead leverage a fault detection and diagnosis (FDD) mechanism to realize the occurrence of faults in real time, and then update the controller via changing of parameters fed into the control

loop.

In Reference [13], a fault tolerant control scheme is designed for a UAS in the presence of actuator failures. It accounted for dynamic uncertainties, but assumes that whenever a failure occurs, it is detected and quantified. Reference [14] presents a dynamic inversion-based fault tolerant control scheme for a hybrid quadrotor-fixed wing UAS which uses the vertical motors in the case of elevator effectiveness loss and floating failures. However, it also assumes perfect detection and quantification of actuation failure. Reference [15] uses incremental dynamic inversion as well as a nonlinear dynamic inversion method with model identification for fault tolerant aircraft trajectory control. However, the model identification method is not used in the case of faults, and no discussion is made to the detection or diagnosis of faults. In Reference [16], a Lyapunov-based technique is used to develop fault tolerant control architecture for a quadrotor UAS with partial actuation failures and parametric uncertainties. Again, no mention is made to the detection or quantification of actuation faults, but knowledge of these failures is still used in simulation. Reference [17] designs a model predictive control-based fault tolerant control algorithm for a hypersonic vehicle. Actuation saturation is taken into account as a cost function constraint, but faults are modelled in simulation as a simple linear reduction to control derivatives, and no explanation is made into failure detection. Reference [18] designs both a model predictive control algorithm and a linear quadratic regulator for fault tolerant control of a small quadrotor UAS. It uses the linear quadratic regulator as a comparison case, as it has no inherent ability to adapt to changes in model. However, it also treated failures as linear reductions to control derivatives in simulation and made no mention to failure detection. Reference [19] develops an active fault tolerant control scheme which uses direct measurement of motor speed for fault detection and diagnosis. However, achieving feedback on actuation systems like servos and motors requires additional hardware. This may be practically infeasible due to excessive cost or weight, especially since additional weight could simply be used for redundant systems.

This research proposes a software-based application of failure detection for a small,

commercial-off-the-shelf (COTS) fixed-wing aircraft. This application contrasts that of Reference [14] where all actuation strength analysis was done off-board and was used to determine a system's level of over-actuation. Instead, this actuation analysis will be used to determine degradation of actuation strength through reduced order system identification and the tracking of calculated model parameters.

2.2 Methodology

These sections introduce the the theory behind the main components of the AFTC design. Section 2.2.1 describes the method proposed to estimate control effectiveness based upon calculation and monitoring of reduced-order models. Section 2.2.2 details the base design of a dynamic inversion controller as explained in Reference [20]. Section 2.2.3 outlines the process of implementing the discussed control methods into an inner-outer loop structure such that simulations can be performed in a manner as similar to flight test as possible. Section 2.2.4 introduces the method for generating validation models from actual flight test data. This method was used for failure flight test data to provide a more realistic assessment of the controller's adaptability to failures.

2.2.1 Failure Detection and Diagnosis

In this work, a method for finding a reduced order model given a set of arbitrary input-output data, eigensystem realization algorithm and observer Kalman filter identification, as described in Reference [21]. These methods have a long history in application for time domain system identification and have shown great results in generating accurate dynamic models [22–25].

2.2.1.1 Eigensystem Realization Algorithm

The eigensystem realization algorithm (ERA) is a system identification method introduced by Juang and Pappa in 1985 [26] and based upon minimal realization theory, originally

published by Ho and Kalman in 1966 [27]. Initially applied to identify structural models for spacecraft, ERA takes an input of sensor measurements from an impulse response and outputs a low-dimensional linear model without the need for prior model knowledge. Given the discrete linear system:

$$x_{k+1} = A_d x_k + B_d u_k \tag{2.1}$$

$$y_K = C_d x_k + D_d u_k \tag{2.2}$$

and a discrete impulse input - measured output pair of:

$$u^\delta(k\Delta t) = I, \;\; k = 0 \tag{2.3}$$

$$u^\delta(k\Delta t) = 0, \;\; k = 1, 2, 3, \ldots \tag{2.4}$$

a measured response of:

$$y^\delta(k\Delta t) = D_d, \;\; k = 0 \tag{2.5}$$

$$y^\delta(k\Delta t) = C_d A_d^{k-1} B_d, \;\; k = 1, 2, 3, \ldots \tag{2.6}$$

should be observed. The impulse responses, m, are collected, corresponding to one impulse response for each control input. These responses are then stored in a matrix y, where each column corresponds to a separate impulse response. A Hankel matrix, H can then be formed by stacking a shifted time series of impulse response measurements.

$$H = \begin{bmatrix} y_1^\delta & y_2^\delta & \cdots & y_{m_c}^\delta \\ y_2^\delta & y_3^\delta & \cdots & y_{m_c+1}^\delta \\ \vdots & \vdots & \ddots & \vdots \\ y_{m_o}^\delta & y_{m_o+1}^\delta & \cdots & y_{m_t-1}^\delta \end{bmatrix} \tag{2.7}$$

$$= \begin{bmatrix} C_d B_d & C_d A_d B_d & \cdots & C_d A_d^{m_c-1} B_d \\ C_d A_d B_d & C_d A_d^2 B_d & \cdots & C_d A_d^{m_c} B_d \\ \vdots & \vdots & \ddots & \vdots \\ C_d A_d^{m_o-1} B_d & C_d A_d^{m_o} B_d & \dots & C_d A_d^{m_t-2} B_d \end{bmatrix} \tag{2.8}$$

m_t is introduced to maintain conciseness of notation, where $m_t = m_o + m_c$. This matrix can be formed without any knowledge of the underlying linear system. The only requirement is a measured impulse input and corresponding response data. Singular value decomposition can then be performed on H to find the dominant trends in the contained time-series data.

$$H = U\Sigma V^\intercal = \begin{bmatrix} \tilde{U} & U_t \end{bmatrix} \begin{bmatrix} \tilde{\Sigma} & 0 \\ 0 & \Sigma_t \end{bmatrix} \begin{bmatrix} \tilde{V}^\intercal \\ V_t^\intercal \end{bmatrix} \tag{2.9}$$

Small singular values, Σ_t, are truncated, and only the first r singular values are kept. U and V describe the eigen-time-delay coordinates. Now, a second Hankel matrix H', shifted by one time step, can be formed as:

$$H' = \begin{bmatrix} y_2^\delta & y_3^\delta & \cdots & y_{m_c+1}^\delta \\ y_3^\delta & y_4^\delta & \cdots & y_{m_c+2}^\delta \\ \vdots & \vdots & \ddots & \vdots \\ y_{m_o+1}^\delta & y_{m_o+2}^\delta & \cdots & y_{m_t}^\delta \end{bmatrix} \tag{2.10}$$

$$= \begin{bmatrix} C_d A_d B_d & C_d A_d^2 B_d & \cdots & C_d A_d^{m_c} B_d \\ C_d A_d^2 B_d & C_d A_d^3 B_d & \cdots & C_d A_d^{m_c+1} B_d \\ \vdots & \vdots & \ddots & \vdots \\ C_d A_d^{m_o} B_d & C_d A_d^{m_o+1} B_d & \dots & C_d A_d^{m_t-1} B_d \end{bmatrix} \tag{2.11}$$

Using matrices H and H', a model can be constructed as follows:

$$\tilde{A} = \tilde{\Sigma}^{-1/2} \tilde{U} H' \tilde{V} \tilde{\Sigma}^{-1/2} \tag{2.12}$$

$$\tilde{B} = \tilde{\Sigma}^{-1/2} \tilde{V}^\intercal \begin{bmatrix} I_p & 0 \\ 0 & 0 \end{bmatrix} \tag{2.13}$$

$$\tilde{C} = \begin{bmatrix} I_q & 0 \\ 0 & 0 \end{bmatrix} \tilde{U} \tilde{\Sigma}^{-1/2} \tag{2.14}$$

Where I_p is the identity matrix of dimension p which extracts the first p columns and I_q is the identity matrix of dimension q which extracts the first q columns. This method estimates a lower order model of the overall state space system. One major benefit of using ERA is it approximately balances the controllability and observability Gramians, which are metrics of the general controllability and observability of the system. This balanced model captures the most input-output relation possible given the measured data. The performance of this approximation in balancing the controllability and observability Gramians is heavily dependent on the amount of data collected and the damping of the system. If not enough

data is collected, the system will not have time for full decay of the impulse signal, so the ERA will not exactly balance the system. The typical solution for avoiding this problem is to collect sufficient data such that the Hankel matrix H is numerically full-rank, meaning its truncated singular values are below an identified tolerance.

2.2.1.2 Observer Kalman Filter Identification

Impulse control inputs are atypical in UAS operations and conducting impulse experiments is challenging due to sensor noise and external disturbances. Data collection can also be difficult, as systems with slow decay of transients require massive amounts of data. Because traditional system identification methods require inversion of a matrix containing this data, identification can become less computationally viable, especially for use in a real-time system. These problems are resolved using observer Kalman filter identification (OKID), first introduced in Reference [28]. Instead of requiring a large amount of data, OKID uses an asymptotically stable observer to construct a discrete state space model. This makes identifying the underlying system simpler. Through design of the observer, a specified decay rate can be assigned via pole placement of the observer system.

This method takes any set of arbitrary input-output data and approximates the impulse response. Consider a discrete linear time invariant state space system:

$$x_{k+1} = Ax_k + Bu_k \tag{2.15}$$

$$y_k = Cx_k + Du_k \tag{2.16}$$

Where $x_k \in \mathbb{R}^n$, $y_k \in \mathbb{R}^q$, and $u_k \in \mathbb{R}^m$. Assuming zero initial conditions, or $x_0 = 0$, the measurement portion of these equations can be written for a sequence of i as:

$$y = \mathbf{YU} \tag{2.17}$$

where y, Y, and U are defined as:

$$y = [y_0 \ y_1 \ y_2 \ \cdots \ y_{l-1}] \tag{2.18}$$

$$Y = [D \ CB \ CAB \ \cdots \ CA^{l-2}B] \tag{2.19}$$

$$U = \begin{bmatrix} u_0 & u_1 & u_2 & \cdots & u_{l-1} \\ 0 & u_0 & u_1 & \cdots & u_{l-2} \\ 0 & 0 & u_0 & \cdots & u_{l-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & u_0 \end{bmatrix} \tag{2.20}$$

This system of equations describes the measured input-output history of the system for the sequence i. The matrix y is a $q \times l$ matrix of output data, where q is the number of outputs and l is the length of data sampled. The matrix Y is a $q \times ml$ matrix which contains all Markov parameters to be estimated, which in this case is $D, CB, CAB, \cdots, CA^{l-2}B$. The matrix U is a $ml \times l$ upper block triangular matrix of control inputs. These equations show that, for multiple control inputs ($m > 1$), there are $q \times ml$ unknown Markov parameters but only $q \times l$ equations. This implies a lack of uniqueness of the solution for $\mathbf{Y}$. Even so, a solution still may not be possible for a single control input for conditions such as zero initial input, $u_0 = 0$, or if the input matrix is rank-deficient. Either of these cases result in a ill-conditioned $\mathbf{U}$ matrix and make calculation of Markov parameters using the pseudo-inverse of the matrix, $\mathbf{Y} = y\mathbf{U}^{-1}$ inaccurate. To solve this problem, instead consider a discrete linear system with an asymptotically stable A matrix. With this assumption, a window size p can be selected such that for all times steps $i \geq p$, $A^i = 0$. This assumption can then be used to

redefine the Markov parameter identification problem as:

$$y = \begin{bmatrix} y_0 & y_1 & y_2 & \cdots & y_p & \cdots & y_{l-1} \end{bmatrix} \tag{2.21}$$

$$Y = \begin{bmatrix} D & CB & CAB & \cdots & CA^{p-1}B \end{bmatrix} \tag{2.22}$$

$$U = \begin{bmatrix} u_0 & u_1 & u_2 & \cdots & u_p & \cdots & u_{l-1} \\ 0 & u_0 & u_1 & \cdots & u_{p-1} & \cdots & u_{l-2} \\ 0 & 0 & u_0 & \cdots & u_{p-2} & \cdots & u_{l-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \dots & u_0 & \cdots & u_{l-p-1} \end{bmatrix} \tag{2.23}$$

In this case, U and Y are truncated forms of $\mathbf{U}$ and $\mathbf{Y}$, as they are $mp \times l$ and $q \times mp$ size matrices instead of full size $ml \times l$ and $q \times ml$ matrices, respectively. By selecting a window size l greater than mp, there will be more data points in y than necessary Markov parameters in Y. This solves the problem of uniqueness encountered in the previous case, as a unique solution for Y is available. This solution, however, raises the issue of system identification for low damped systems. For a system with low damping, a large amount of time would be required to satisfy the condition $A^i = 0$, resulting in a large value of p. Since $l \geq mp$, a low damped system with a large amount of control inputs m would require a vast amount of data, which would make solving these equations computationally expensive. The most straight-forward way to decrease the computational expense would be to artificially increase the damping of the system via the use of a feedback loop. However, this is not always practically possible, or in the case of open-loop system identification, counter-intuitive. Instead, an optimal observer may be added to the state function to form the following optimal observer function:

$$\hat{x}_{k+1} = A_d\hat{x}_k + K_f(y_k - \hat{y}_k) + B_d u_k \tag{2.24}$$

$$\hat{y}_k = C_d\hat{x}_k + D_d u_k \tag{2.25}$$

which may be written in simplified form:

$$\hat{x}_{k+1} = (A_d - K_f C_d)\hat{x}_k + \begin{bmatrix} B_d - K_f D_d, & K_f \end{bmatrix} \begin{bmatrix} u_k \\ y_k \end{bmatrix} \tag{2.26}$$

As long as the original discrete state space system is observable, the poles of this optimal observer function can be placed anywhere, allowing full control over system damping. However, the observer gains are typically decided based on measurement noise, process noise, and model uncertainty using a linear-quadratic estimator, or Kalman filter. Using the dynamics of the observer system, the observer Markov parameters can be found as outlined in Equation 2.27. First, the window size of observer Markov parameters to identify, l must be determined. Based upon the number of observer Markov parameters necessary, data matrices S and V are constructed:

$$S = \begin{bmatrix} y_0 & y_1 & \dots & y_l & \dots & y_m \end{bmatrix} \tag{2.27}$$

$$V = \begin{bmatrix} u_0 & u_1 & \dots & y_l & \dots & y_m \\ 0 & v_0 & \dots & v_{l-1} & \dots & v_{m-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & v_0 & \dots & v_{m-l} \end{bmatrix} \tag{2.28}$$

This matrix follows a similar pattern to the U matrix discussed in the previous methods

of finding Markov parameters, except due to the added optimal observer, the addition of measured outputs through the use of $v_i = \begin{bmatrix} u_i^T & y_i^T \end{bmatrix}^T$ is necessary. These matrices then form the equation $S = \bar{S}V$, which can be used to solve for $\bar{S}$ using the pseudo-inverse of V. The Markov parameters of the original system are then reconstructed from the observer Markov parameters by first ordering the observer Markov parameters as:

$$\bar{S}_0^\delta = D \tag{2.29}$$

$$\bar{S}_k^\delta = \begin{bmatrix} (\bar{S}^\delta)_k^{(1)} & (\bar{S}^\delta)_k^{(2)} \end{bmatrix}, \quad k \geq 1 \tag{2.30}$$

where $(\bar{S}^\delta)_k^{(1)} \in \mathbb{R}^{q \times p}$, $(\bar{S}^\delta)_k^{(2)} \in \mathbb{R}^{q \times q}$, and $y_0^\delta = D$. The remaining Markov parameters can be reconstructed from observer Markov parameters $\bar{S}_k^\delta$ using the following equation:

$$y_k^\delta = (\bar{S}^\delta)_k^{(1)} + \sum_{i=1}^{k} (\bar{S}^\delta)_k^{(2)} y_{k-i}^\delta, \quad k \geq 1 \tag{2.31}$$

The full set of identified system Markov parameters estimate the impulse response of the system and can be fed directly into the eigensystem realization algorithm for model identification.

2.2.1.3 Applications for Failure Detection

Although this form of system identification is normally used to construct reduced-order open loop models for control design or system analysis, closed loop models can be used for identification and quantification of failures. For the case of lateral-directional control as discussed in this work, the main goal of a controller is to increase the damping of the oscillatory lateral-directional mode, Dutch roll. Therefore, the states to be investigated in the reduced-order model are determined by both the eigenvector of the state matrix and the control effectiveness matrix formed by multiplying the control matrix B by the inverse of the

state matrix eigenvector, shown below in Tables 2.1 and 2.2.

Table 2.1: SkyHunter State Matrix Eigenvectors

	Roll	Dutch Roll	Spiral
β	0.012	-0.061 ± 0.024i	-0.033
ϕ	-0.075	-0.013 ± 0.0962i	-0.8015
P	0.991	0.845	0.1699
R	0.113	-0.269 ± 0.4478i	-0.5724

Table 2.2: SkyHunter Control Effectiveness Matrix

	Aileron (δa)	Rudder (δr)
Roll	90.77	-6.380
Dutch Roll	18.62 ± 20.20i	5.442 ± 22.40i
Spiral	-4.258	5.796

The Dutch roll mode of this aircraft is mainly a function of P and R, so these variables were used for formation of the reduced order model for failure detection. As damping is the main change desired from the addition of a controller, the main variables tracked in this work were $\mathcal{L}_p$, which relates rolling moment $\mathcal{L}$ to roll rate p and describes the roll damping of the system and $\mathcal{N}_r$, which relates yawing moment $\mathcal{N}$ to yaw rate r. For aileron failures, $\mathcal{L}_p$ is tracked as aileron has the larger impact on modes which are a function of p. For rudder failures, $\mathcal{N}_r$ is tracked as rudder has a larger impact on modes which are a function of r. However, as shown in Table 2.2, aileron has an overall larger impact on the aircraft dynamic modes than rudder. This means that an aileron-based fault would have a much larger impact on system dynamics. As this is more critical to system performance, aileron-based faults are the case focused on in this work.

2.2.2 Linear Model Predictive Control

Consider a discrete linear time-invariant system given previously in Equation 2.1. The optimal control problem for reference tracking is defined by the following cost function:

$$V(x,U) = \sum_{i=0}^{N-1} ((x_i - x_r)^\intercal Q(x_i - x_r) + U_i^\intercal R U_i) + (x_N - x_r)^\intercal Q(x_N - x_r). \tag{2.32}$$

$$s.t.\ |u| \leq u_{lim}. \tag{2.33}$$

Here, x_r denotes the reference state to be tracked generated by the guidance. This reference is considered as a constant reference over the time horizon, N. U denotes the vector $\begin{bmatrix} u^\intercal & \Delta u^\intercal \end{bmatrix}^\intercal$, where u is control inputs and Δu is the discrete control increment, defined by $\Delta u = u_k - u_{k-1}$. Control increment was added to the traditional cost function formulation to discourage oscillations of the control input signal. The only constraint set on the system is a saturation of total control surface deflection, as large control surface deflections reduce the accuracy of the linear assumptions upon which the control force and moment model is based. The model predictive control (MPC) algorithm takes the discrete dynamic model from Equation 2.15 as well as the current state vector x_o as inputs and solves the optimization problem of minimizing the cost function V over the given horizon N. The output of this optimization is the optimal control sequence $u^* = \begin{bmatrix} u_0^* & u_1^* & \dots & u_N^* \end{bmatrix}$. The optimal control for the current time step, u_0^*, is then applied to the system and optimization is repeated at every time step. This control method offers more promise of fault tolerance than offline optimization-based systems, like the linear quadratic regulator (LQR), as real-time updates to the dynamic model are utilizable in future optimization steps, unlike the static gain matrix used by LQR-based controllers.

2.2.3 Linear Dynamic Inversion Control

Consider a continuous linear time-invariant system given in the state-space form:

$$\dot{x} = Ax + Bu \tag{2.34}$$

$$y = Cx + Du \tag{2.35}$$

Assuming that $D = 0$, Equation 2.35 can be differentiated such that:

$$\dot{y} = C\dot{x} = CAx + CBu \tag{2.36}$$

Solving this equation for the control output gives the following solution:

$$u = (CB)^{-1}(v - CAx) \tag{2.37}$$

This solution reveals that the control output can only be found if the product of the observability and control matrices, CB, is invertible. Substituting Equation 2.37 into Equation 2.36 yields the simple, decoupled integrator dynamics:

$$\dot{y} = v \tag{2.38}$$

where v is known as the pseudo-control vector. This work specifies the pseudo-control vector as a combination of reference rate of change and gain-multiplied reference error:

$$v = \dot{y}^* + K(y - y^*) \tag{2.39}$$

With perfect inversion, this pseudo-control vector directly specifies the rate of change of observed variables. This direct control of output response is the biggest advantage for the use of dynamic inversion in active fault tolerant control. As long as the fault detection and diagnosis algorithm can sufficiently update the model to prevent inversion error, and adequate control authority is still present, this control technique can guarantee the capability to maintain a desired response in the presence of adverse on-board conditions.

2.2.4 Control Loop Design

The focus of control loop design was to minimize differences between software-in-the-loop design and flight test design. Therefore, flight test algorithms were implemented in the simulation control loop. The first step necessary was the implementation of guidance logic. Although some similar works on the topic of fault tolerant control use step inputs, doublets, or transfer function transformation of similar signals to validate tracking of control designs, this does not accurately emulate the dynamic characteristics of an actual guidance logic. This aspect was especially important for the design of dynamic inversion controllers, as rate of change of reference plays a substantial role in performance when rapid changes occur, such as turns or changes of commanded altitude. Without a real guidance logic, a controller's robustness to dynamic changes in error will not be known and could cause major issues in real-world testing. To provide dynamic commands in this work, the guidance logic developed in [29] was used. This look-ahead guidance logic takes a waypoint path as an input and generates roll commands to track the path. However, since commanding roll with dynamic inversion for an aircraft dynamic model would lead to a non-invertible CB, the error between this roll command and measured roll was converted to a roll rate command through the use of a proportional-integral-derivative (PID) outer-loop controller. The roll rate command output by the PID was then fed into the dynamic inversion controller. A block diagram of the described control loop is shown below in Figure 2.1.

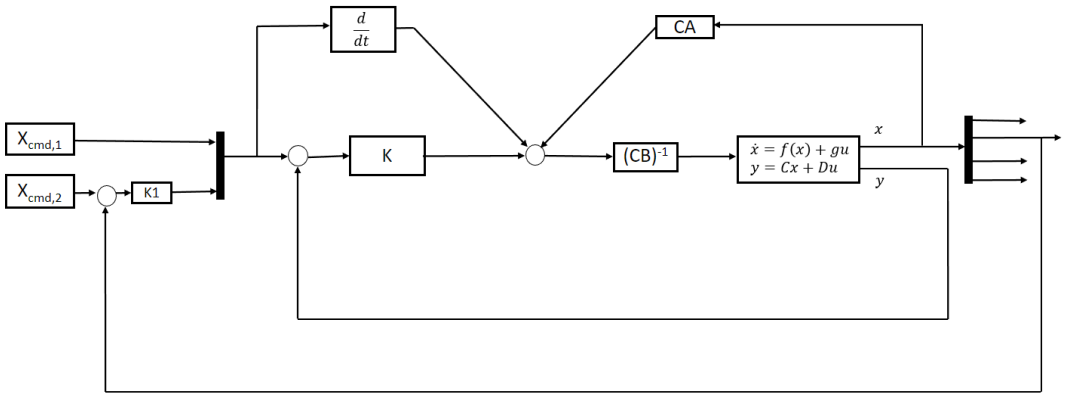


Figure 2.1: Block Diagram for Lateral Dynamic Inversion Control

2.2.5 Validation Model Design

Monte Carlo (MC) simulations[a] were performed and compared to the recorded flight data to identify a dynamic model that most closely resembles the aircraft flight dynamics. At the core of the Monte Carlo simulations is a base dynamic model. This base model uses the non-linear six degree of freedom fixed-wing aircraft equations of motion. The aerodynamic forces and moments are modeled using first order Taylor series approximations and stability and control derivatives (SCDs) found using Advanced Aircraft Analysis (AAA). AAA is an aircraft design software which uses physics-based and semi-empirical methods to estimate the SCDs. The cost of developing aircraft models using geometry-based methods is relatively low and therefore the obtained model is expected to have low fidelity. The base model uses a set of 22 SCDs.

The Monte Carlo simulations are performed to identify stability and control derivatives that result in a dynamic model which most closely resembles the flight data. The goal is to perform a large number of simulations in which the dynamic model parameters are allowed to randomly vary within specified uncertainty ranges. The simulations are then compared to the recorded flight data to identify which simulation parameters yield the smallest simulation errors. Each of the derivatives is allowed to vary according to a normal distribution for which

[a]This work was done in collaboration with Hady Benyamen.

the mean is the AAA SCD estimate and the standard deviation is specified by the user. 3000 simulations are performed in the MC analysis. Each simulation starts from an initial condition obtained from the flight data and at each simulation step, the aircraft controls are obtained from the recorded flight data. The root mean square error (RMSE) between each simulation and the flight data is calculated at the end of the simulations. The simulation which results in the minimum RMSE is identified and the SCDs used in that simulation are obtained. These SCDs result in the closest resemblance to the flight data. Comparison was made between the MC simulation results and the flight data for the lateral-directional rotational rates, the roll and yaw rates (P and R). The models generated are shown below, with the subscript 100 denoting the state space model generated from 100% aileron effectiveness LQR data, and the subscript 50 denoting the model generated from 50% aileron effectiveness LQR data.

$$\dot{x}_{100} = \begin{bmatrix} -0.3297 & 0.7721 & 0.0008 & -0.9752 \\ 0 & 0 & 1 & 0 \\ -157.7178 & 0 & -8.6919 & 5.8740 \\ 45.2848 & 0 & -0.8798 & -3.0807 \end{bmatrix} x_{100} + \begin{bmatrix} 0 & 0.1185 \\ 0 & 0 \\ 149.2587 & 2.9993 \\ -9.2588 & -20.5755 \end{bmatrix} u_{100} \quad (2.40)$$

$$\dot{x}_{50} = \begin{bmatrix} -0.3978 & 0.7762 & 0.0026 & -0.9801 \\ 0 & 0 & 1 & 0 \\ -132.1390 & 0 & -9.4014 & 2.8891 \\ 42.6573 & 0 & -0.9804 & -3.5932 \end{bmatrix} x_{50} + \begin{bmatrix} 0 & 0.1658 \\ 0 & 0 \\ 61.1433 & 3.7177 \\ -10.3573 & -18.0211 \end{bmatrix} u_{50} \quad (2.41)$$

2.3 Software in the Loop Simulations

This work is validated using an in-house designed Python simulator. The flight controller provides control inputs to the six degree of freedom dynamic model, which propagates the states forward using a Runge-Kutta fourth order discrete integrator. By separating the simulator into distinct code objects, the environment is flexible, allowing for modular

changing of guidance and control algorithms.

2.3.1 Computer Hardware

These simulations are validated on a single tower Dell PC. It houses a Intel I7-9700K CPU at 3.60GHz and 16GB of DDR4 RAM. The performance of this machine allows for the simulation of 20000 iterations to be completed in between 25 and 27 seconds of computation time.

2.3.2 UAS Dynamic Model

The UAS chosen for this research is a COTS twin-boom platform known as the Skyhunter. This UAS has a cruise speed of approximately $15ms^{-1}$, wingspan of $1.9m$ and an endurance of 20 minutes. The base aerodynamic stability and control derivatives of this UAS are found as described previously. The propulsive derivatives are found through repeated motor testing, collection of thrust data versus throttle percentage, and fitting a cubic regression to the resulting thrust curves. The servo dynamics for all servos are modelled as a first-order delay of 30 Hz. The final product of this modelling is a linear set of stability and control derivatives, which are used to find aerodynamic forces and moments for the software-in-the-loop simulator.The salient characteristics for UAS flight are outlined in Table 2.3.

Table 2.3: Skyhunter Specifications

Wingspan	$1.9m$
Mass	$4.0kg$
Cruise Velocity	$15.2m \cdot s^{-1}$
I_{xx}	$0.43kg \cdot m^2$
I_{yy}	$0.73kg \cdot m^2$
I_{zz}	$0.31kg \cdot m^2$

The model for this aircraft has also been refined through system identification resulting from over 300 autonomous flights. With this plethora of data, the UAS model has been fine-tuned to a point of confidence and displays the least difference between performance in simulation and flight test. Future work for this control scheme includes flight test validation

and verification, so the use of a aircraft with a high quality dynamic model was necessary. Having a high accuracy model would mean lower inversion error, which is one of the biggest issues facing the application of dynamic inversion. Figure 2.2 showcases the Skyhunter UAS in a recent flight test.

Figure 2.2: Skyhunter UAS

2.3.3 Environment Setup

The simulation environment is a featureless area of 300 by 600 meters in which a rectangular waypoint track is centered. For consistency, the UAS begins at the same trim state, about 30 meters inside the waypoint box for every simulation. The simulation runs for 200 seconds, collecting state and control data for later inspection.The six degree of freedom equations of motion, described in [30], are

$$\dot{V} = F_{a+p}/m + g - \widetilde{\omega} \times V \tag{2.42}$$

$$I\dot{\omega} = M_{a+p} - \widetilde{\omega} \times I\omega \tag{2.43}$$

These equations, as well as the previously discussed control loop, were iterated at 100

Hz and integrated using a Runge-Kutta fourth order discrete integrator for all simulations. For the model predictive control results discussed in this work, the Python toolbox do-mpc describe in [31] was used. This environment leverages the CasADi symbolic framework [32] and the Multifrontal Massively Parallel Sparse direct solver (MUMPS) from the Interior Point Optimizer (IPOPT) open source optimization library. This method allowed for straightforward implementation and testing as required in this work.

2.4 Results and Discussion

2.4.1 Preliminary Flight Test of Actuation Failure

UAS failures can occur in a multitude of ways. However, for the sake of practicality, actuation failures were considered as degradation of control effectiveness. Degradation of control effectiveness occurs when, for any reason, the control effectors of the UAS generate less aerodynamic forces and moments than usual. This could be caused by power issues, such as low input voltage to the servo decreasing its range of deflection, or by structural failures, such as a partial break of a faulty propeller. These were approximated using a fractional multiplier on the control outputs. For example, the magenta region of Figure 2.4 show flight test data from a 70% control effectiveness test, where the control outputs were decreased by 30%. Preliminary flight tests were performed with the Skyhunter UAS and its base linear quadratic regulator controller to investigate the effects of control degradation when controllers have no mechanism for fault tolerance. As shown in Figures 2.3 and 2.4, lateral error, specifically overshoot from turns, worsens when the control effectiveness is reduced to 70%, and at 15% the controller is unable to track the waypoint path given the wind conditions. These results emphasize the importance of fault tolerance in UAS, as well-tuned, high performance controllers cannot guarantee performance in the presence of actuation failures.

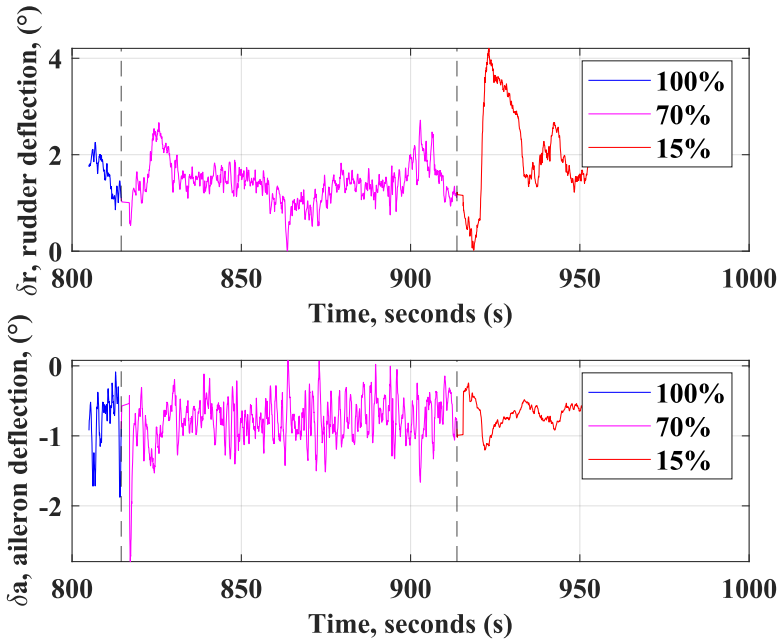


Figure 2.3: Control Outputs of Degradation Flight Test

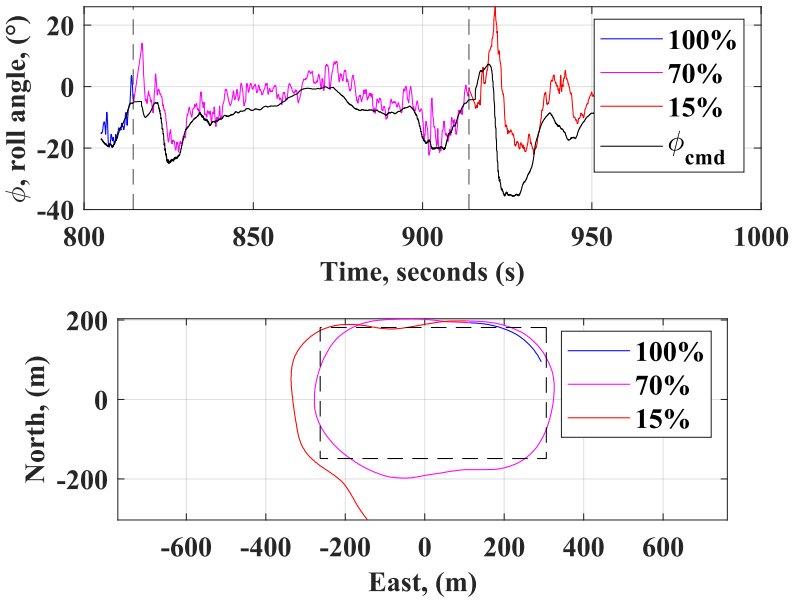


Figure 2.4: Lateral State and Path Tracking of Degradation Flight Test

2.4.2 Failure Detection Results

The method of failure detection described in Section 2.2.1 was applied to two sets of aileron degradation SkyHunter flight test data. Linear quadratic regulator (LQR) controllers were

used for both longitudinal and lateral-directional control of the SkyHunter, and the look-ahead guidance logic described in [29] was used to provide roll and pitch angle commands. These two data sets provided a stark contrast in results, highlighting the design decisions and operational limitations of the proposed method.

The first data set, discussed in the previous section, was collected for 100%, 70%, and 15% aileron degradation of control effectiveness on a day with a nominal wind of 16 kilometers per hour and gusts reaching up to 24 kilometers per hour. These wind conditions are extremely high for usual SkyHunter flight, with peak gusts reaching almost half of the SkyHunter's 50 kilometer per hour cruise speed. This flight test data was input to the system identification method described above in Sections 2.2.1.1 and 2.2.1.2. Reduced order discrete models were generated at one second intervals for different window sizes of flight test data, and a sample of calculated derivatives are shown by histograms below in Figure 2.5.

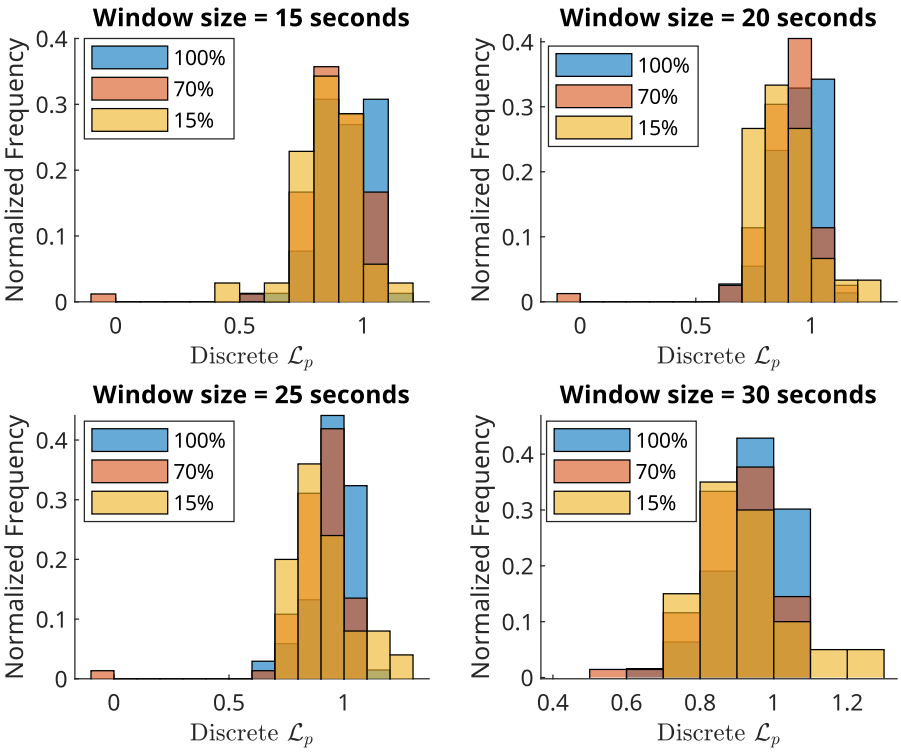


Figure 2.5: Discrete $\mathcal{L}_p$ for Varying Windows of High Wind Data

A second data set was collected on a day with low wind conditions, with a constant wind

from the East of 5 kilometers per hour with gusts up to 8 kilometers per hour. These wind conditions are more typical of SkyHunter flight test. Reduced order discrete models were again created as discussed above, and the results for the same data window sizes are given in Figure 2.6.

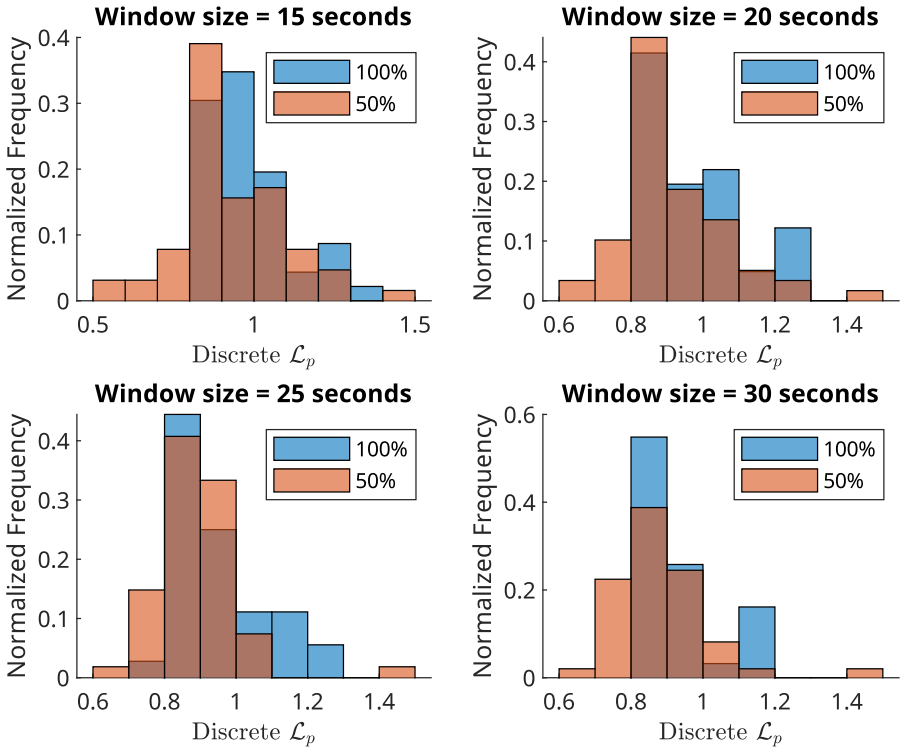


Figure 2.6: Discrete $\mathcal{L}_p$ for Varying Windows of Low Wind Data

Results from the first set of data show no consistent trend in terms of calculated $\mathcal{L}_p$ over the differing window sizes other than a decreased standard deviation. It can be hypothesized as a larger window size would result in more data being used in multiple model calculations. However, the second set of data also exhibits the trend expected in mean values, as the mean $\mathcal{L}_p$ value for the degraded data is lower than the nominal data for all window sizes. These results are shown below in Figure 2.7.

The absence of this trend in the first data set is likely due to the presence of high wind. System identification methods typically struggle with large, unmeasured external disturbances [33]. As the effects of disturbances, like wind, can overpower the actual dynamics of the

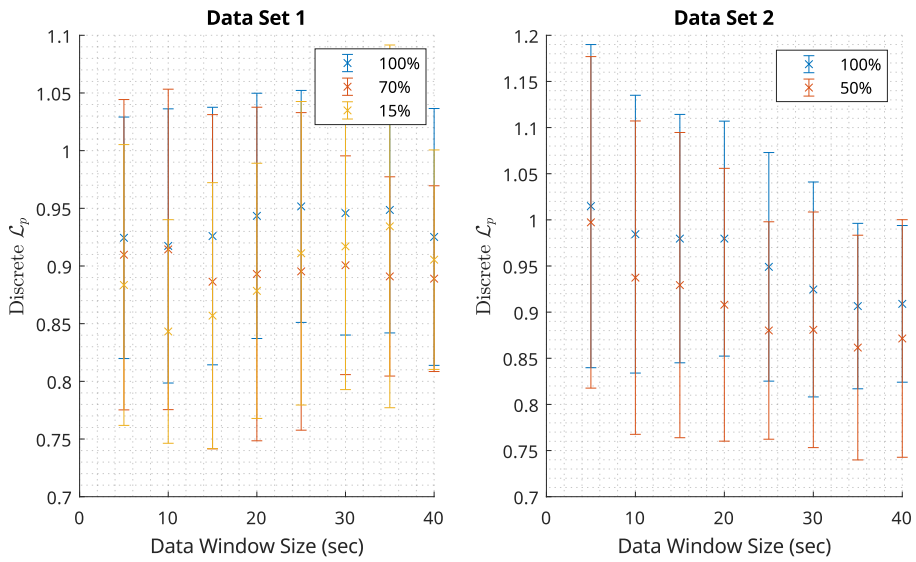


Figure 2.7: Mean and Standard Deviation of Calculated $\mathcal{L}_p$ for Varying Windows of Low Wind Data

system, estimated models can vary widely, even over the course of a single flight. These impacts are much less important for larger aircraft, since their larger mass and moment of inertia naturally damp external disturbances. However, as the SkyHunter is a small UAS without the large damping stability derivatives common in large aircraft, a moving window was applied to the calculated $\mathcal{L}_p$ values of the second data set to moderate the fluctuations over time. To determine the best data window size to apply the moving average to, the metric $\frac{\mu_{nom}-\mu_{deg}}{\sigma_{nom}}$ was used to estimate the difference between calculated ranges of $\mathcal{L}_p$, where μ and σ are the mean and standard deviation of $\mathcal{L}_p$, respectively. The 20 second data window maximized this metric and was used for subsequent moving averages. Moving averages of different periods were used to investigate the validity of this method for real-time failure detection. The results of four different periods is shown below in Figure 2.8.

This window size can be a function of the aircraft's dynamic characteristics, including mass, moment of inertia, geometry, and cruise velocity. The consistency of the trend can be observed after the moving window period increases beyond 30 seconds. This shows there is a minimum amount of points required for successful fault identification. However, the values do not stay within unique bounds, as the parameter values near the end of the full

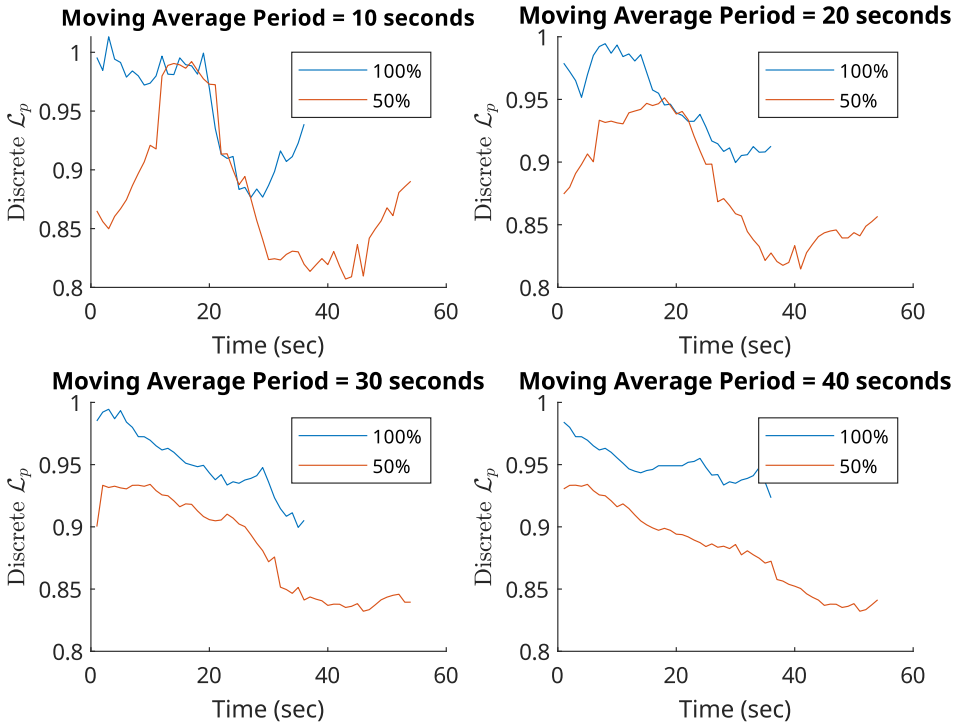


Figure 2.8: Discrete $\mathcal{L}_p$ for Varying Moving Average Periods

control authority data dip below the maximum of the parameter values for the degraded data. These results show the validity of this method to be used on its own as a method of failure detection, but also demonstrates the need for some additional classifier to determine exact levels of degradation, either via standard regression methods or using advanced techniques such as machine learning.

2.4.3 Base Dynamic Inversion Controller Results

For the preliminary design of the active fault-tolerant control, a base dynamic inversion controller is necessary. This controller is designed with an LTI model described previously, which has been verified with extensive flight test and system identification. Results from software in the loop simulations are shown below in Figures 2.9 through 2.11. Roll tracking root mean square error was used as a metric to determine the phase lag of tracking guidance commands, as well as a metric to quantify tracking error for controllers of similar lag. Dynamic

inversion had a roll tracking root mean square error of 7.82 degrees, compared to a roll tracking RMSE of 9.06 degrees for a well-tuned LQR. This shows that dynamic inversion realized commands from guidance much faster than the LQR. Although the aircraft begins almost 30 meters off the desired path, it quickly converges and shows great tracking along the North and South legs. However, considering the short distances between waypoints and the aircraft's cruise velocity, the tracking is impacted on the short legs, resulting in a mean absolute cross track error of 25.41 feet off the lateral path. The linear model used for dynamic inversion is based upon the assumption of operation via small perturbations from a linearized trim point. Since sharp maneuvers such as the 90 degree turns of this waypoint box require large roll angles and the model is linearized around a wings-level operation point ($\phi = 0$), inversion error caused by model mismatch increases and causes the decrease in tracking of roll angle during turns. This issue could be solved either via the use of a higher-order model and non-linear dynamic inversion or through scheduling the use of different linear models for steady state level turn.

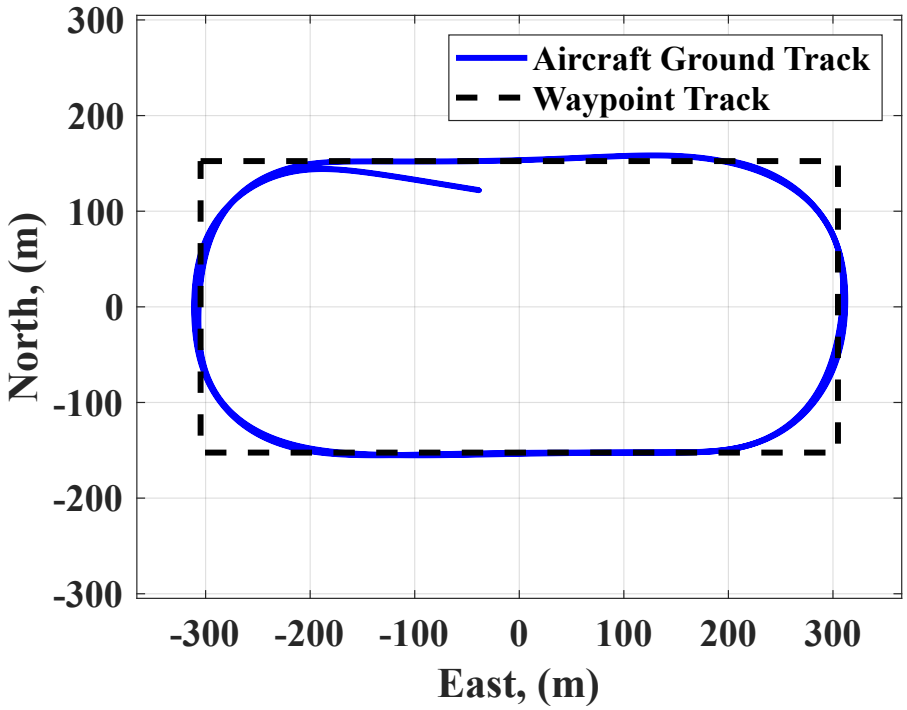


Figure 2.9: Lateral Path Tracking of Dynamic Inversion Controller

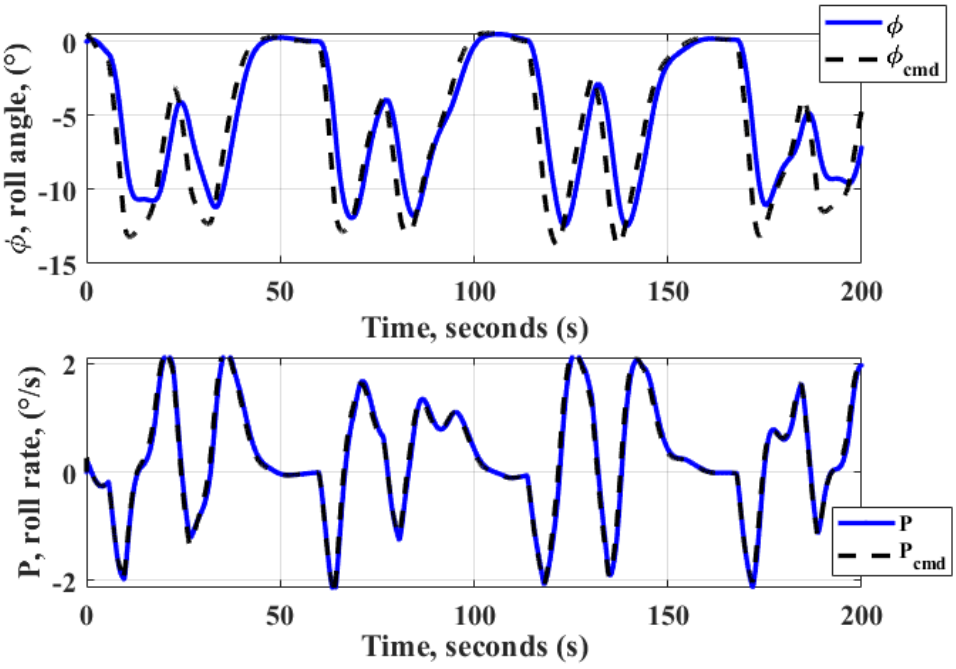


Figure 2.10: Roll and Roll Rate Tracking of Dynamic Inversion Controller

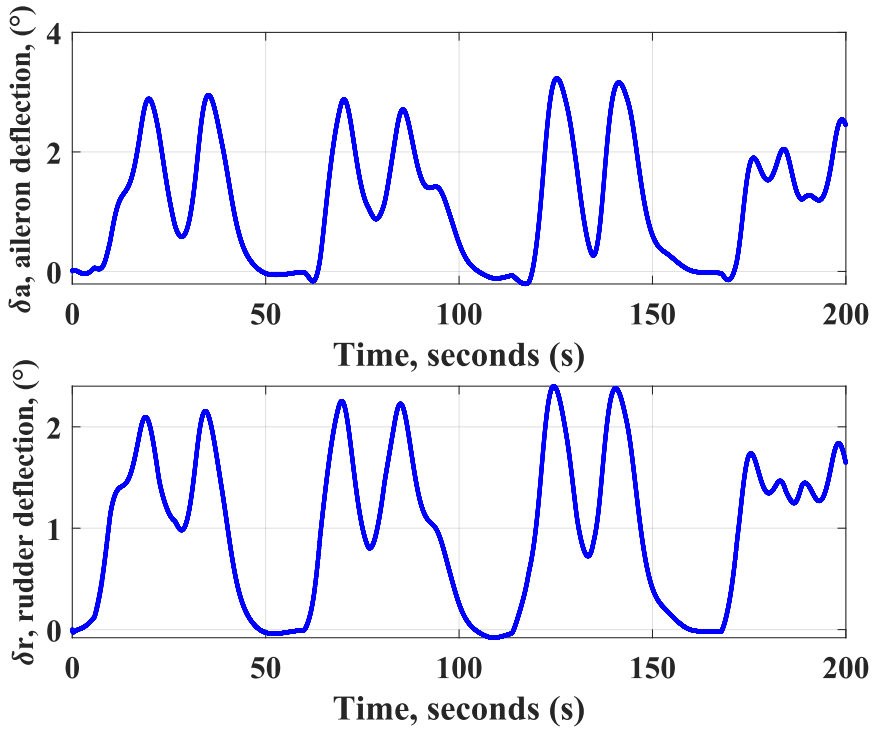


Figure 2.11: Aileron and Rudder Outputs of Dynamic Inversion Controller

2.4.4 Base Model Predictive Control Results

The base model predictive controller, designed using the same linear dynamic model as the dynamic inversion controller, was also tested as a comparison to the base dynamic inversion controller. For these results, the MPC cost function was optimized over a prediction horizon of 1 second at an update rate of 100 Hz. However, as this controller directly controlled roll angle, no commanded roll angular rate is shown. The lateral path tracking, roll angle tracking , and control outputs are shown below in Figure 2.12. Tracking in this nominal case achieves the similar levels as the dynamic inversion base controller, with a roll tracking root mean square error of 7.78 degrees. Similar to linear dynamic inversion, the MPC quickly realizes commands given by the guidance, shown by its quick convergence to the commanded lateral path. The MPC does not perfectly track large roll commands due to the same linear model assumptions as the dynamic inversion controller. However, its predictive capability along with the explicitly defined penalties on control surface deflections allow for satisfactory tracking of reference signals with much lower control surface deflections than the base dynamic inversion controller, resulting in a slightly better mean absolute cross track error of 22.18 feet. The added Δu term adequately limited the amplitude of control input oscillation, but oscillations in control input rate still occur. This could be due to the abnormal formulation of the reference tracking cost function, as typical MPC reference tracking cost functions only punish reference errors, while this cost function also punishes the roll and yaw rates, which relate to the transient performance leading to tracking.

This demonstrates one of the major benefits of model predictive control over dynamic inversion, as the cost function optimization of the MPC allows for explicit punishments of large control deflections via the control cost matrix, R, without major sacrifices in tracking performance. Dynamic inversion, however, has no similar explicit method of reducing control surface deflections. Instead, performance of dynamic inversion is only altered through the design and subsequent tuning of the pseudo-control vector, v. This leads to a more complex design process than the intuitive cost function tuning provided by the MPC.

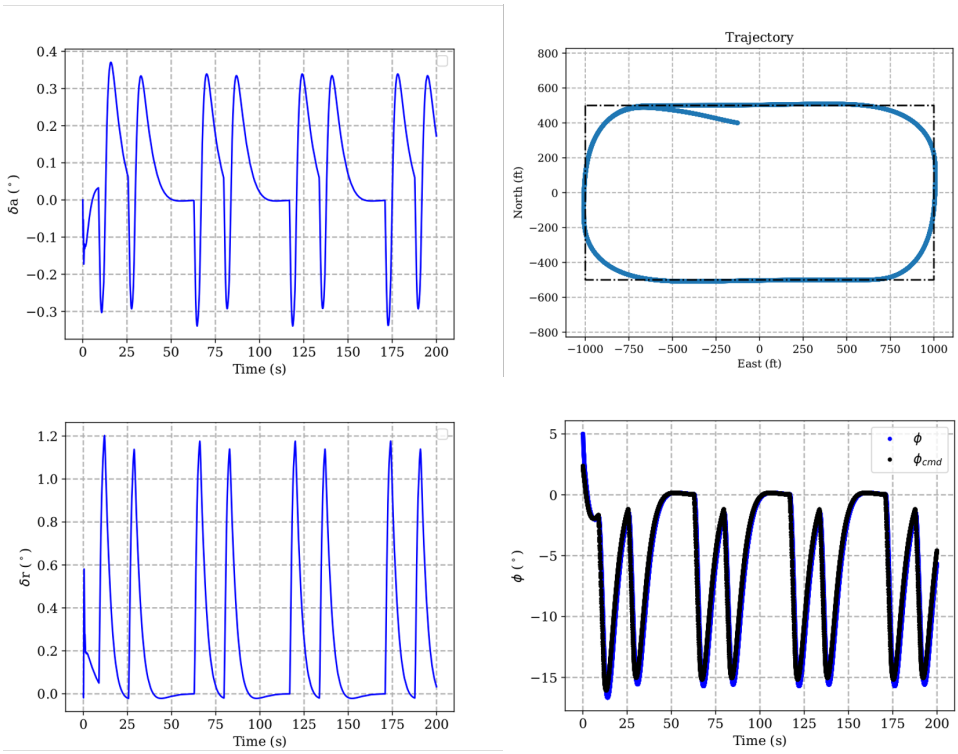


Figure 2.12: Model Predictive Control Base Control Results

2.4.5 Controller Results with Validation Models

Given successful failure detection from the previous section, new models can be generated as discussed in Section 2.2.5 to adapt the onboard controllers. For the simulations done in this work, validation models were generated using the Monte Carlo method to test the performance of the previously discussed controllers with models impacted by failures. These failure models were taken from the 50% aileron degradation test case for data set 2, as this failure event showed the most consistent results from the failure detection algorithm. Models were selected based on minimum RMSE of roll rate and yaw rate between flight test data and simulation data with generated models, and an average model of the top five simulations was used. Since this model is rooted in flight test data, it provides a better comparison to real-world failure cases than the assumptions conventionally used in AFTC literature. Full results are shown in the appendices A, B, and C. The resulting roll angle command and lateral path tracking is shown below in Figure 2.13. The top, middle, and bottom rows

show results from the dynamic inversion controller, model predictive control, and linear quadratic regulator, respectively. The dynamic inversion controller root mean square error of roll tracking is 7.79 degrees, the MPC root mean square error of roll tracking is 7.85 degrees, and the linear quadratic regulator controller root mean square error of roll tracking is 9.13 degrees. Both the dynamic inversion and MPC control methods have a higher peak roll rate of around 11 degrees per second; whereas, the non-adaptive LQR only reaches 8 degrees per second. The only negative aspect of the dynamic inversion controller is its higher control rates, with a peak aileron rate of 23 degrees per second and a peak rudder rate of 10 degrees per second. MPC maintains low control rates, with a maximum aileron rate of 1.5 degrees per second and a maximum rudder rate of 4 degrees per second.

The roll tracking of the linear quadratic regulator controller is significantly worse when roll command is high, as it does not adapt for the reduced aileron strength. This lack of adaptation can also be seen in the roll rate, as the MPC and dynamic inversion controllers maintain a roll rate similar to the nominal case to continue to accurately follow the trajectory, whereas the LQR has a lower maximum roll rate. This difference is due to the ways in which the different controllers leverage the dynamic model in their loop structure. Since LQR optimizes its gain matrix offline for a given cost function and dynamic model, it is incapable of using model updates with its traditional implementation. Unlike LQR, MPC uses online optimization to find new gain matrices in real time. Through this optimization, MPC has the ability to leverage information from updated models and adapts its outputs for the reduced control authority. Dynamic inversion has similar capabilities, as through inversion of the updated dynamic model, the system inherently uses the information from model updates to solve for the new control output.

Cross track error off of the lateral path is also critical, as not following the given trajectory in spatially constrained environments like UAM could results in collisions with buildings or other infrastructure. Typically, mean absolute error(MAE) is a good metric to quantify tracking performance for controllers of similar performance. However, the different perfor-

mance characteristics of these controllers make this comparison more difficult. For the data shown, the MAE for dynamic inversion, MPC, and LQR was 25.78 feet, 22.94 feet, and 24.85 feet, respectively. As the LQR controller does not adapt for the reduced effectiveness of the new model, it turns much slower than the dynamic inversion and MPC controllers, thereby reducing the peak cross track error in corners by around 30 feet, compensating for the large overshoots. This is the biggest contributor to mean absolute cross track error for the other two controllers, so comparisons between the adaptive MPC and dynamic inversion and non-adaptive LQR using this metric are unclear. Because of this, the overshoot of cross track error after turns was analyzed for all controllers to characterize their convergence back to the specified trajectory after the turns. The adaptive benefits of dynamic inversion and MPC can be seen in this metric, as dynamic inversion has a maximum cross track overshoot of 29.7 feet and MPC has a maximum cross track overshoot of 3.1 feet, compared to the maximum cross track overshoot of 50.4 feet for LQR. It can be hypothesized that the large difference in overshoot between dynamic inversion and MPC is caused by the coupled effect of difference in tracking aggressiveness and the tuning of guidance parameters. The guidance values used for this work were tuned specifically for the LQR base controller typically used in flight test. This means that the guidance might be more successful for controllers of similar aggressiveness. As MPC punishes control surface use through penalties on both deflection magnitude and rate, its level of aggressiveness may be more similar to the LQR controller, meaning it is more in-phase with the guidance. On the other hand, dynamic inversion has no mechanism for penalizing control outputs, and control outputs are generated with the sole purpose of following the response of the give pseudo-control vector, v. Depending on the design of v, this may mean a higher level of control outputs as seen in this work, resulting in dynamic inversion being less in-phase with the guidance.

The MPC algorithm uses significantly lower control rates due to the added penalty to control increment, Δu, in its cost function. This explicit form of authority over control rates is a major benefit to the use of MPC for fault tolerant control. Limiting control rates for

dynamic inversion is possible, but through more complex means such as design of the pseudo-control vector v or saturation limits on reference rate. Although lower control rates may result in a worse tracking performance overall, they decrease the chance of controller-induced instability, since high control rates increase the overall system energy dramatically, causing large oscillations and possibly loss of control.

The tracking of MPC and dynamic inversion is similar in both roll angle and lateral tracking, but the increase in roll tracking error and cross track MAE from the nominal case to the failure validation model is much worse for the MPC, as dynamic inversion roll tracking actually improves for the failure case and cross track mean a. Even though the performance of the base linear quadratic regulator is acceptable and the performance of the MPC is excellent, the preservation of performance shown by dynamic inversion further validates its ability to adapt to adverse onboard conditions given sufficient model updates. These results show that as the magnitude of failures increases, dynamic inversion provides the most graceful degradation out of the investigated controllers as long as accurate model updates are provided. Given data from a wide variety of failure conditions, models could be generated with the Monte Carlo model matching algorithm and used in a method similar to a look up table to provide low degradation of performance over a wide operational envelope.

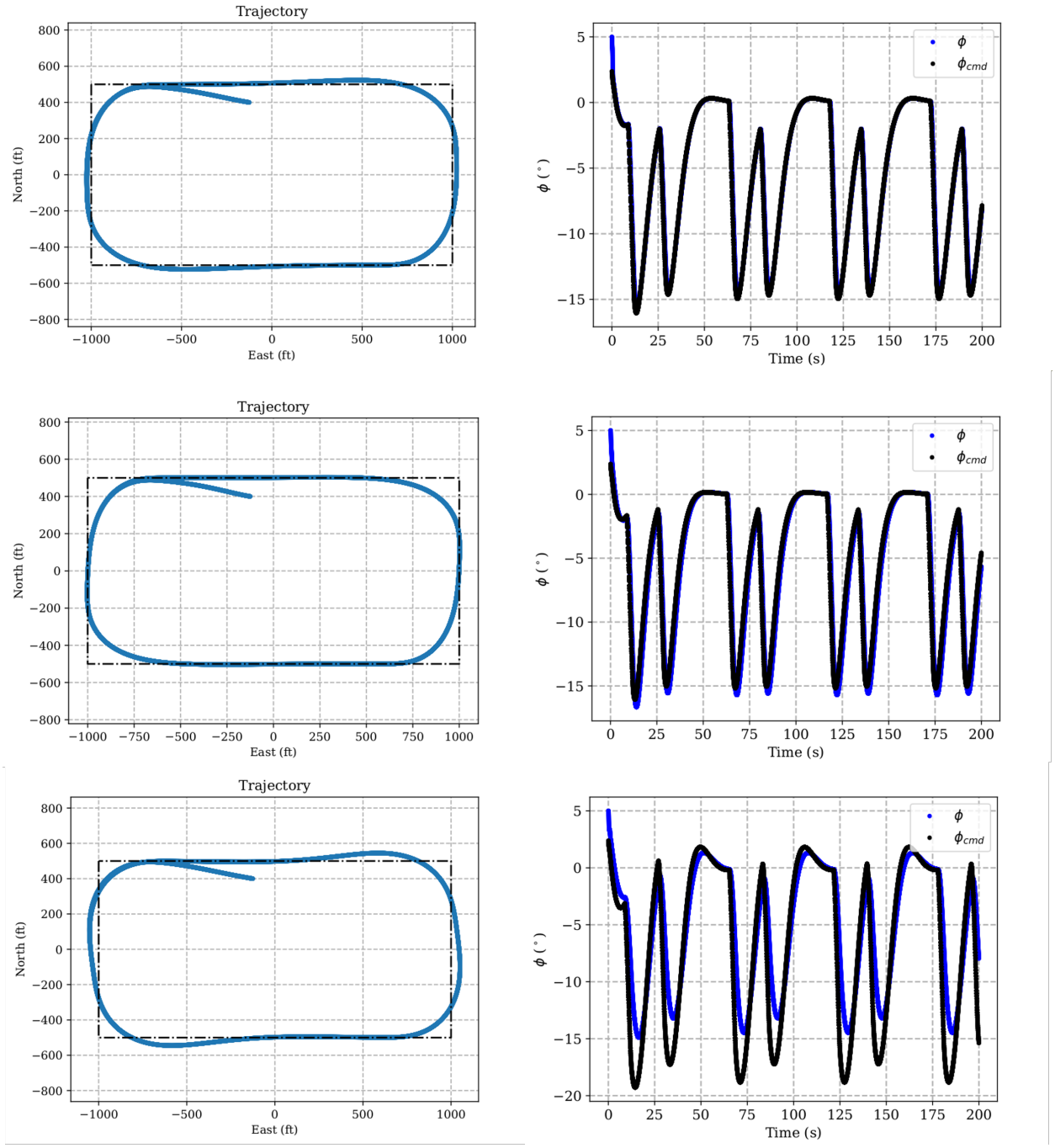


Figure 2.13: Lateral Path and Roll Angle Tracking during 50% Aileron Degradation

2.5 Conclusions and Future Works

In this work, the mathematical basis behind a system identification-based fault detection and diagnosis mechanism was introduced. The impact of data window size was quantified, as

measured $\mathcal{L}_p$ distributions became more distinctly separate and the standard deviation of $\mathcal{L}_p$ decreased as window size increased. The large impact of wind on this method implies that this failure detection and diagnosis scheme would work better for systems with larger mass and moment of inertia due to their inherent damping towards external disturbances. However, this method could be improved for smaller platforms if wind was estimated and accounted for in system identification. The design of both a base lateral linear dynamic inversion controller and linear MPC were explained, as well as their integration into a six degree of freedom simulation environment with a look-ahead guidance logic. Results of simulation were shown for both nominal and Monte Carlo generated failure models, with the failure scenario showing the promise of adaptivity to adverse conditions possible with both MPC and dynamic inversion. Although control rates were much higher for dynamic inversion, trade-offs between base tracking performance and control rates can be made by applying a saturation limit to the reference rate in the dynamic inversion loop. Given adequate knowledge of any current failures via model updates, dynamic inversion showed the most graceful degradation of performance, a quality necessary for UAS operations in civilian airspace.

Future work for the linear dynamic inversion control design will include both the hardware in the loop and flight test of the base linear dynamic inversion controller to judge both its ability to run in real time and its robustness to external disturbances and modelling uncertainties. Additional tuning could be done to further improve the in-phase behavior of the dynamic inversion controller with the look-ahead guidance logic. The reduction of inversion error through the use of model scheduling or higher order models with nonlinear dynamic inversion could be investigated. To improve the adaptivity of the MPC, investigations can be made into scheduling different cost matrices, Q and R. More flight test data will be collected for different intensities of wind and failure conditions to better understand the ability of the proposed failure detection method to accurately diagnose failure levels, and investigation will be made into using classical regression methods or machine learning-based classification methods for failure diagnosis.

Bibliography

[1] R. Bowes, H. Benyamen, and S. Keshmiri, "System identification-based fault detection and dynamic inversion control of an uncrewed aerial system," in *2023 International Conference on Unmanned Aerial Systems*. IEEE, 2013, pp. 661–666.

[2] M. Waibel, "What technologies enabled drones to proliferate?" Jun 2021. [Online]. Available: https://spectrum.ieee.org/quadcopter-hexacopter-octocopter-uavs

[3] "68% of the world population projected to live in urban areas by 2050, says un | un desa department of economic and social affairs." [Online]. Available: https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html

[4] D. schrank, L. Albert, B. Eisele, and T. Lomax, "2021 urban mobility report," Texas A&M Transportation Institute, Tech. Rep., 2021.

[5] C. Hoehner, C. Barlow, P. Allen, and M. Schootman, "Commuting distance, cardiorespiratory fitness, and metabolic risk," *American Journal of Preventative Medicine*, vol. 42, no. 6, pp. 571–578, 2012.

[6] M. Petrov, J. Weng, and K. Reid, "Commuting and sleep: Results from the hispanic community health study/study of latinos sueño ancillary study," *American Journal of Preventative Medicine*, vol. 54, no. 3, pp. 49–57, 2018.

[7] M. Hilbrecht, B. Smale, and S. E. Mock, "Highway to health? commute time and well-being among canadian adults," *World Leisure Journal*, vol. 56, no. 2, pp. 151–163, 2014. [Online]. Available: https://doi.org/10.1080/16078055.2014.903723

[8] P. Yedavalli and J. Mooberry, "Community perception study," Available at https://www.airbusutm.com/uam-resources-community-perception (2023/04/12).

[9] S. Hasan, "Urban air mobility (uam) market study," NASA, Tech. Rep., 2019.

[10] A.-R. Merheb, H. Noura, and F. Bateman, "Passive fault tolerant control of quadrotor uav using regular and cascaded sliding mode control," in *2013 Conference on Control and Fault-Tolerant Systems (SysTol)*, 2013, pp. 330–335.

[11] R. R. Benrezki, M. Tadjine, F. Yacef, and O. Kermia, "Passive fault tolerant control of quadrotor uav using a nonlinear pid," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2015, pp. 1285–1290.

[12] S. Y. Vural, J. Dasdemir, and C. Hajiyev, "Passive fault tolerant lateral controller design for an uav," *IFAC-PapersOnLine*, vol. 51, no. 30, pp. 446–451, 2018, 18th IFAC Conference on Technology, Culture and International Stability TECIS 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S240589631832994X

[13] J. O. Pedro and T. B. Tshabalala, "Fault-tolerant control of fixed-wing uav using ga-optimised control allocation technique," in *2017 11th Asian Control Conference (ASCC)*, 2017, pp. 371–676.

[14] K. F. Prochazka, T. Ritz, and H. Eduardo, "Over-actuation analysis and fault-tolerant control of a hybrid unmanned aerial vehicle," *5th CEAS Conference on Guidance, Navigation, and Control*, 2019.

[15] P. Lu, E.-J. Van Kampen, C. De Visser, and Q. Chu, "Aircraft fault-tolerant trajectory control using incremental nonlinear dynamic inversion," *Control Engineering Practice*, vol. 57, pp. 126–141, 2016.

[16] X. Zhang and Y. Zhang, "Fault tolerant control for quad-rotor uav by employing

lyapunov-based adaptive control approach," in *AIAA Guidance, Navigation, and Control Conference*, 2010.

[17] X. Hu, H. R. Karimi, L. Wu, and Y. Guo, "Model predictive control-based non-linear fault tolerant control for air-breathing hypersonic vehicles," *IET Control Theory & Applications*, vol. 8, no. 13, pp. 1147–1153, 2014.

[18] B. Yu, Y. Zhang, I. Minchala, and Y. Qu, "Fault-tolerant control with linear quadratic and model predictive control techniques against actuator faults in a quadrotor uav," in *2013 Conference on Control and Fault-Tolerant Systems (SysTol)*. IEEE, 2013, pp. 661–666.

[19] M. Saied, B. Lussier, I. Fantoni, H. Shraim, and C. Francis, "Fault diagnosis and fault-tolerant control of an octorotor uav using motors speeds measurements," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5263–5268, 2017, 20th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896317308315

[20] J. F. Horn, "Non-linear dynamic inversion control design for rotorcraft," *Aerospace*, vol. 6, no. 3, 2019.

[21] S. L. Brunton and J. N. Kutz, *Balanced Models for Control*. Cambridge University Press, 2019, p. 321–344.

[22] T. D. Woodbury, J. Valasek, and F. Arthurs, "Flight test results of observer/kalman filter identification of the pegasus unmanned vehicle," in *AIAA Atmospheric Flight Mechanics Conference*, 2015, p. 1481.

[23] C. M. Pappalardo and D. Guida, "System identification algorithm for computing the modal parameters of linear mechanical systems," *Machines*, vol. 6, no. 2, p. 12, 2018.

[24] J. S. S. Júnior, E. B. M. Costa, and L. M. M. TORRES, "Multivariable fuzzy identification

of unmanned aerial vehicles," in *Congresso Brasileiro de Automática-CBA*, vol. 1, no. 1, 2019.

[25] J. Valasek and W. Chen, "Observer/kalman filter identification for online system identification of aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 2, pp. 347–353, 2003.

[26] J.-N. Juang and R. S. Pappa, "An eigensystem realization algorithm for modal parameter identification and model reduction," *Journal of Guidance, Control, and Dynamics*, vol. 8, no. 5, pp. 620–627, 1985. [Online]. Available: https://doi.org/10.2514/3.20031

[27] B. HO and R. E. Kálmán, "Effective construction of linear state-variable models from input/output functions," *at-Automatisierungstechnik*, vol. 14, no. 1-12, pp. 545–548, 1966.

[28] J.-N. Juang, M. Phan, L. G. Horta, and R. W. Longman, "Identification of observer kalman filter markov parameters: Theory and experiments," *Journal Of Guidance, Control, and Dynamics*, vol. 16, no. 2, pp. 320–329, 1993.

[29] J. Xu, A. McKinnis, S. Keshmiri, and R. Bowes, "Flight test of the novel fixed-wing multireference multiscale ln guidance logic for complex path following," *Journal of Intelligent & Robotic Systems*, 2022.

[30] R. Stengel, *Flight Dynamics*. Princeton University Press, 2004.

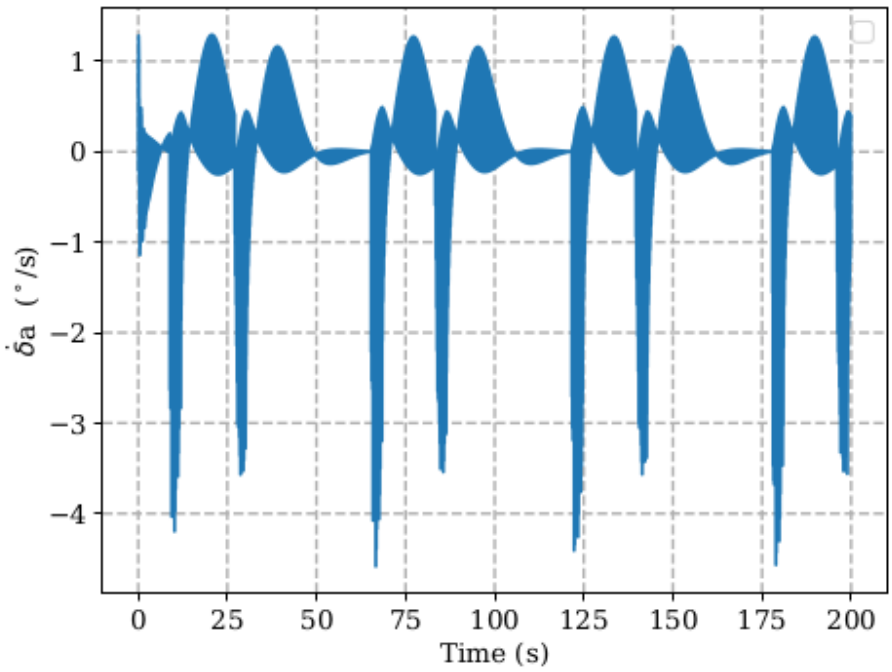
[31] S. Lucia, A. Tatulea-Codrean, C. Schoppmeyer, and S. Engell, "An environment for the efficient testing and implementation of robust nmpc," *IEEE Conference on Control Applications*, 2014.

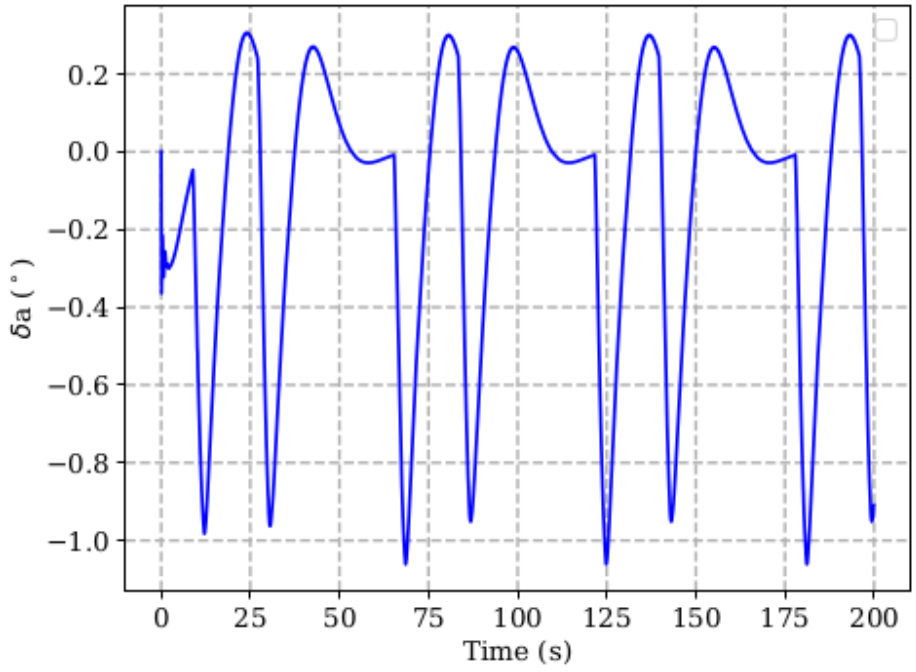
[32] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

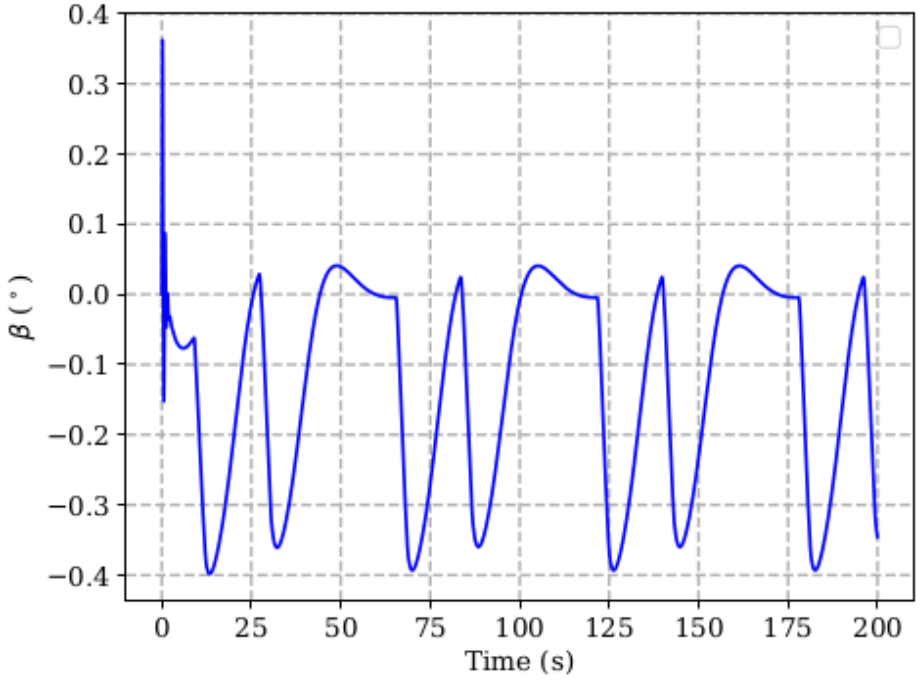
[33] S. S. Keshmiri, E. Lan, and R. Hale, "Nonlinear aerodynamics of an unmanned aircraft in wind shear," *Aircraft Engineering and Aerospace Technology*, vol. 89, no. 1, pp. 39–51, 2017.

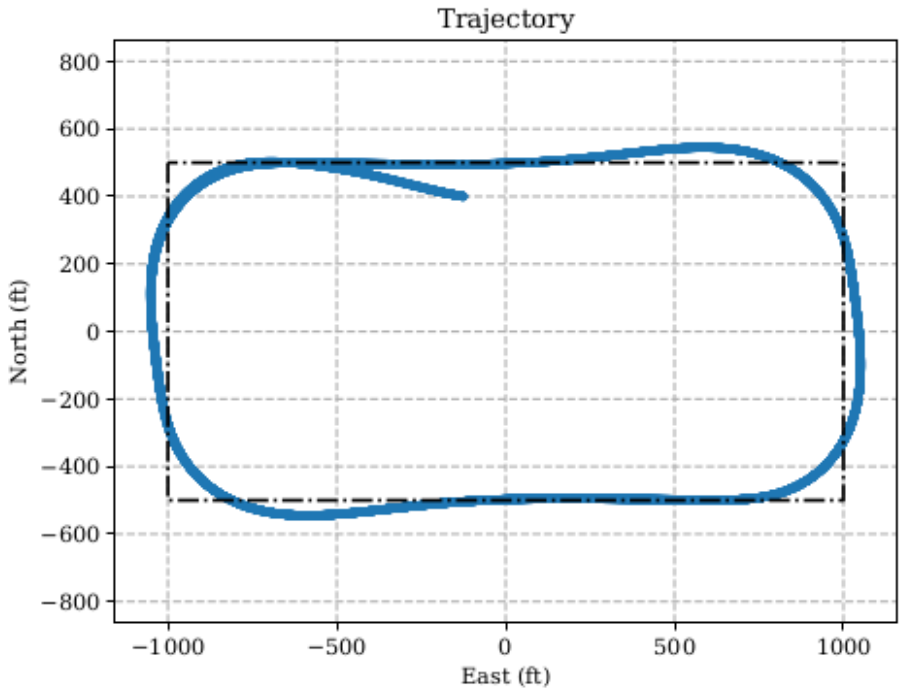
Appendix A

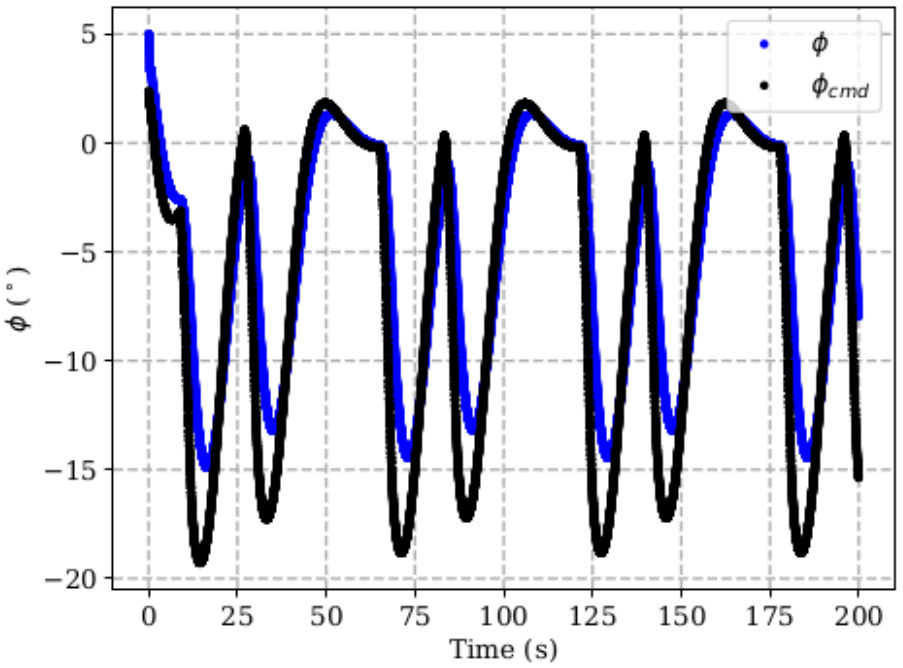
LQR Control Lateral Plots for 50% Aileron Degradation

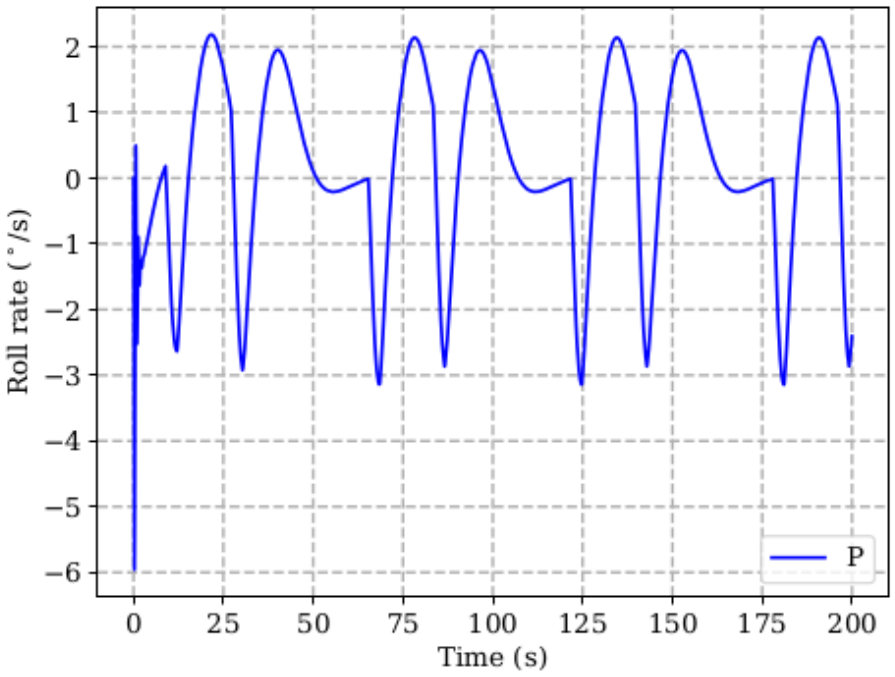


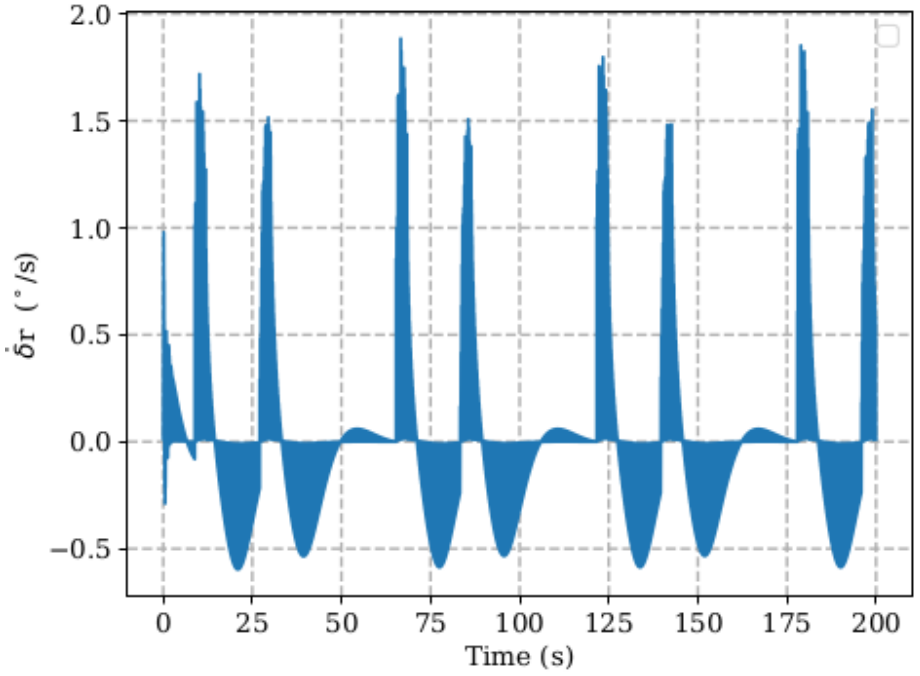

0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
δa (°)
0
25
50
75
100
125
150
175
200
Time (s)

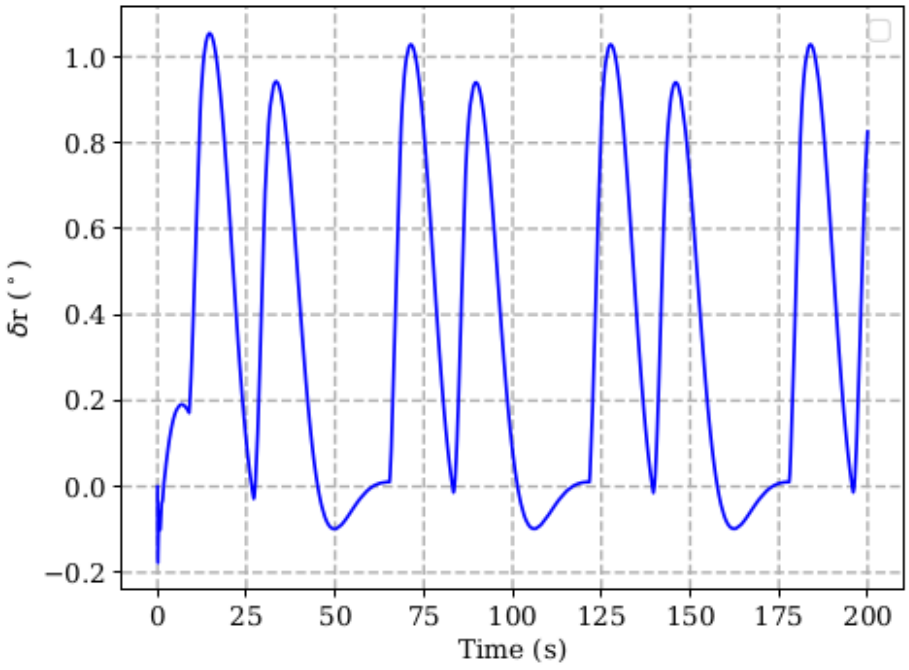

0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
β (°)
0
25
50
75
100
125
150
175
200
Time (s)

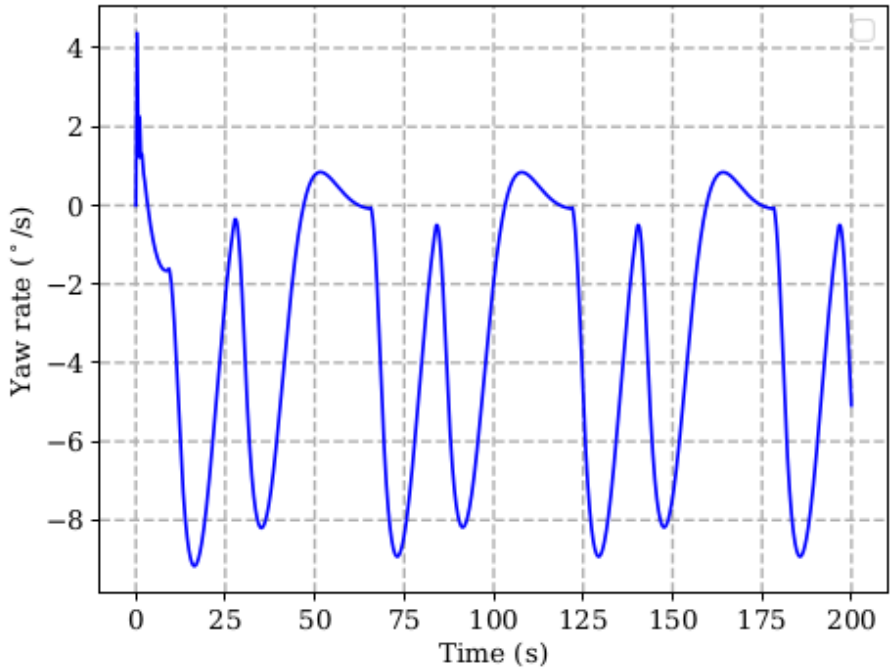

Trajectory
North (ft)
East (ft)
800
600
400
200
0
−200
−400
−600
−800
−1000
−500
0
500
1000


ϕ
ϕ_{cmd}
ϕ (°)
Time (s)
5
0
−5
−10
−15
−20
0
25
50
75
100
125
150
175
200


Roll rate (°/s)
Time (s)
P

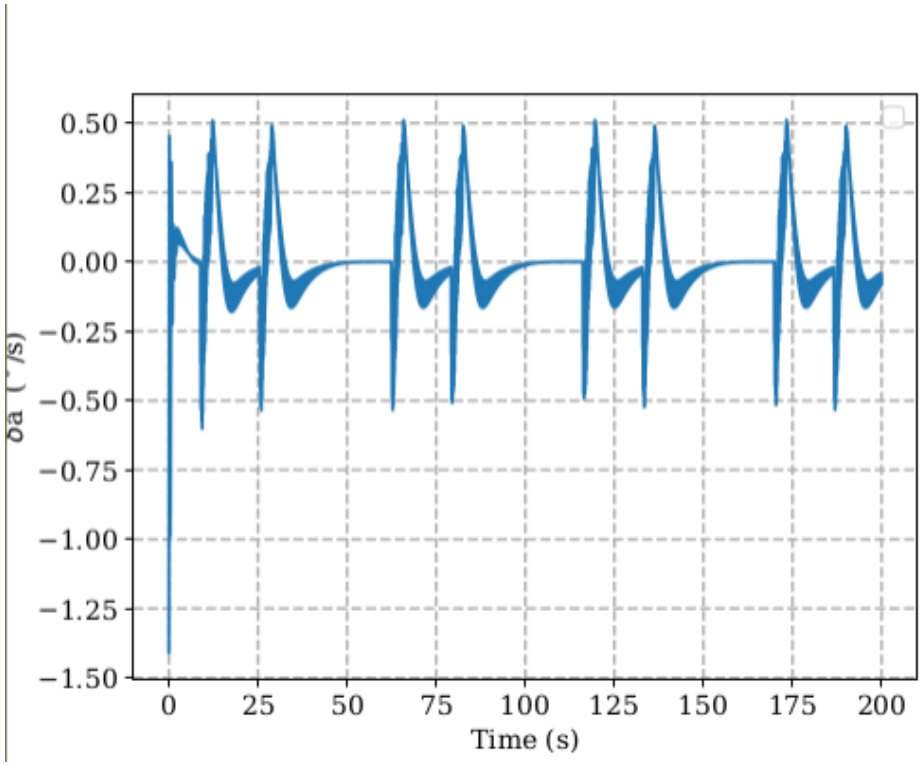

$\dot{\delta}_R$ (°/s)
Time (s)

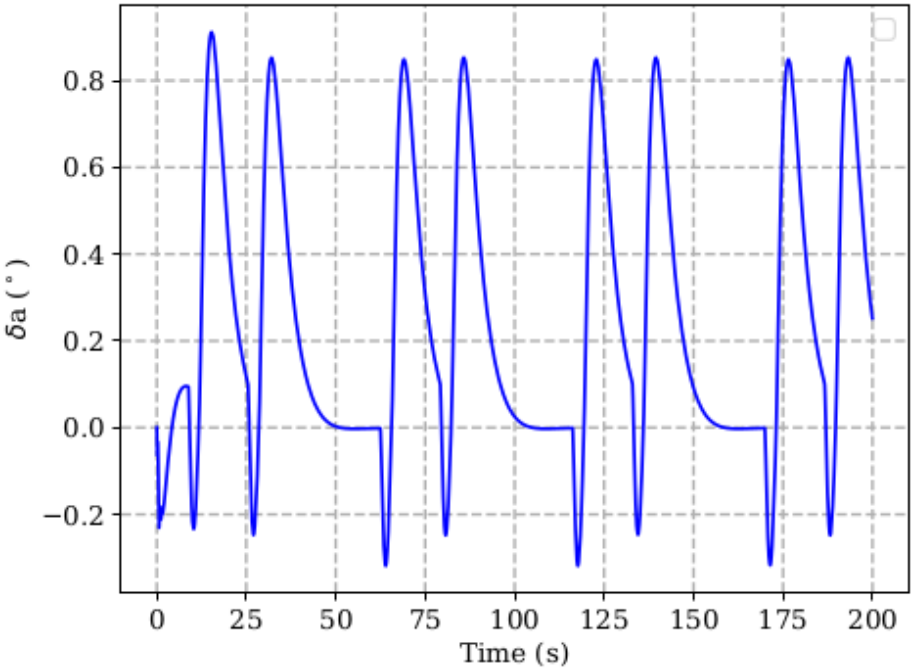

1.0
0.8
0.6
0.4
0.2
0.0
−0.2
δr (°)
0
25
50
75
100
125
150
175
200
Time (s)

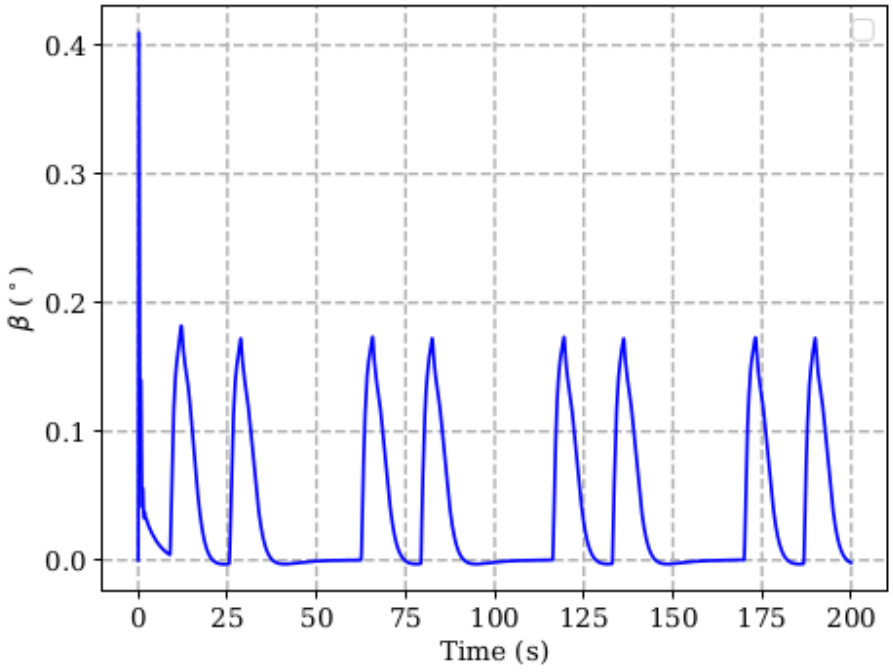

4
2
0
−2
−4
−6
−8
Yaw rate (°/s)
0
25
50
75
100
125
150
175
200
Time (s)

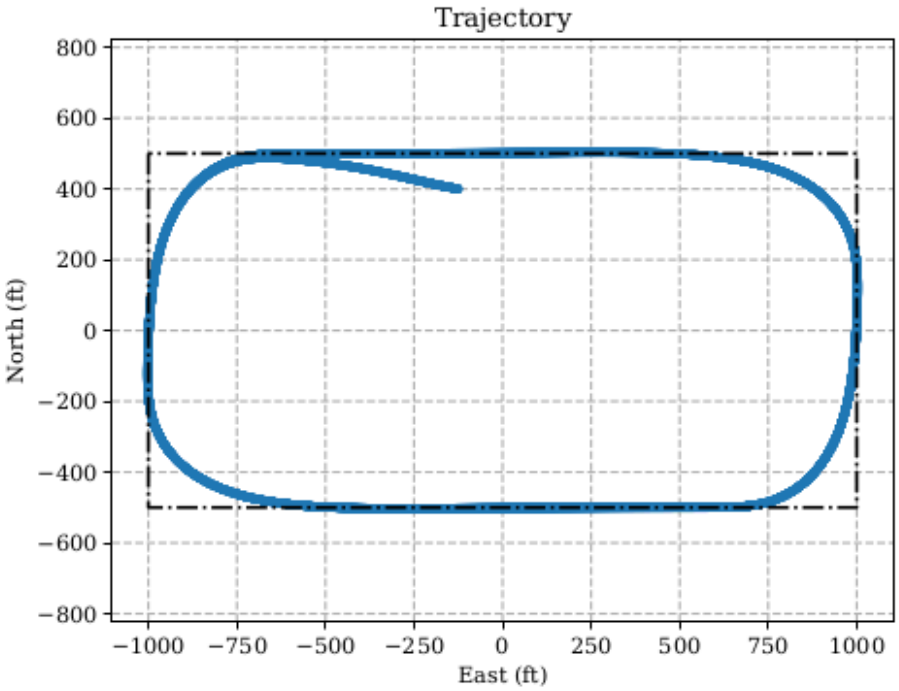
Appendix B

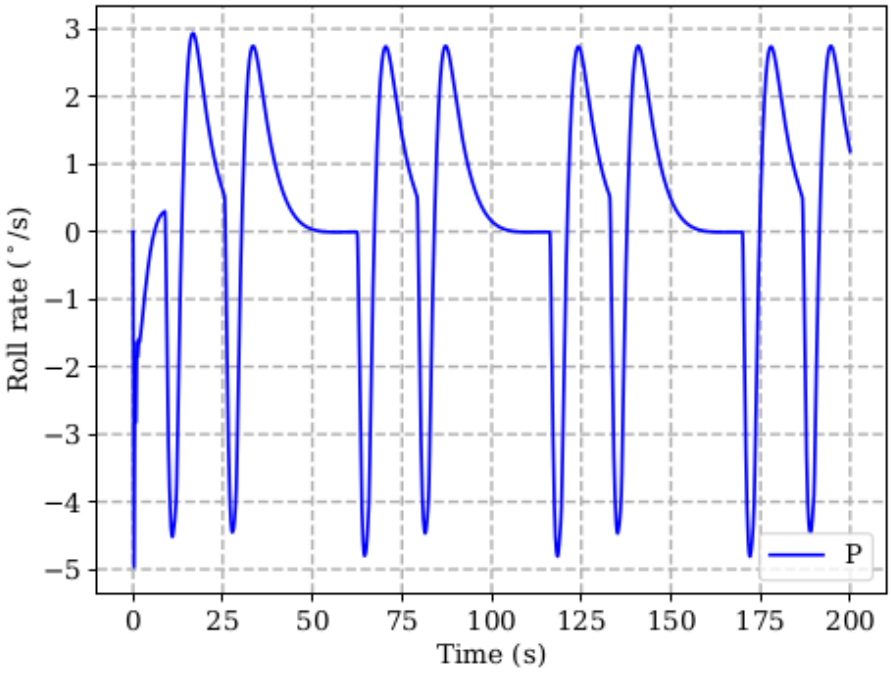
MPC Lateral Plots for 50% Aileron Degradation

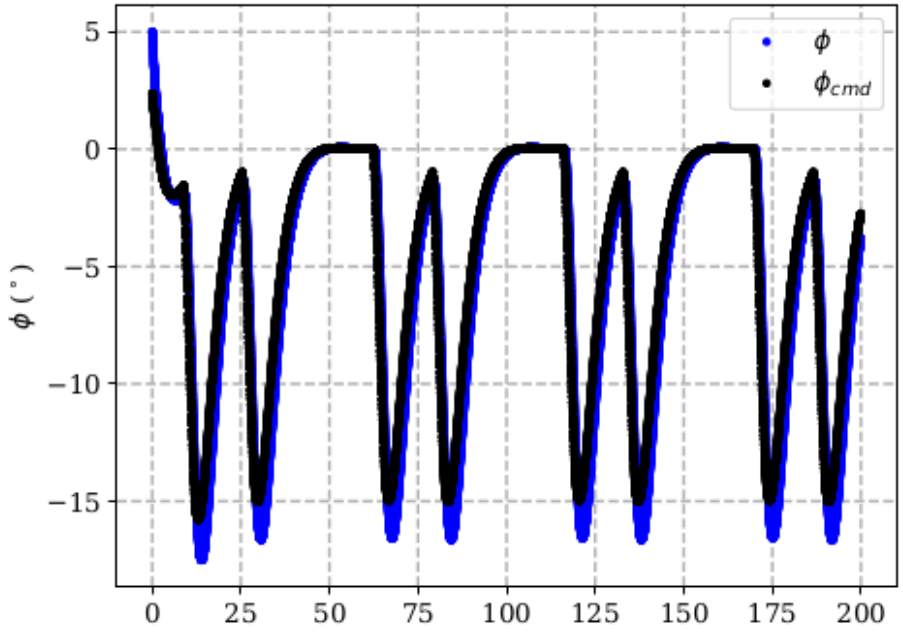


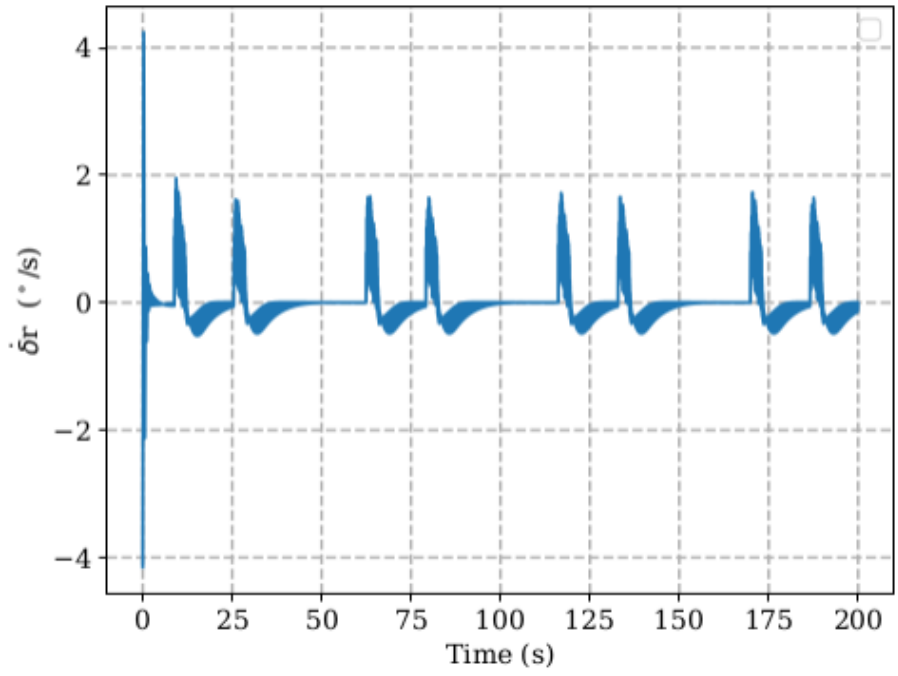

0.8
0.6
0.4
0.2
0.0
−0.2
δa (°)
0
25
50
75
100
125
150
175
200
Time (s)

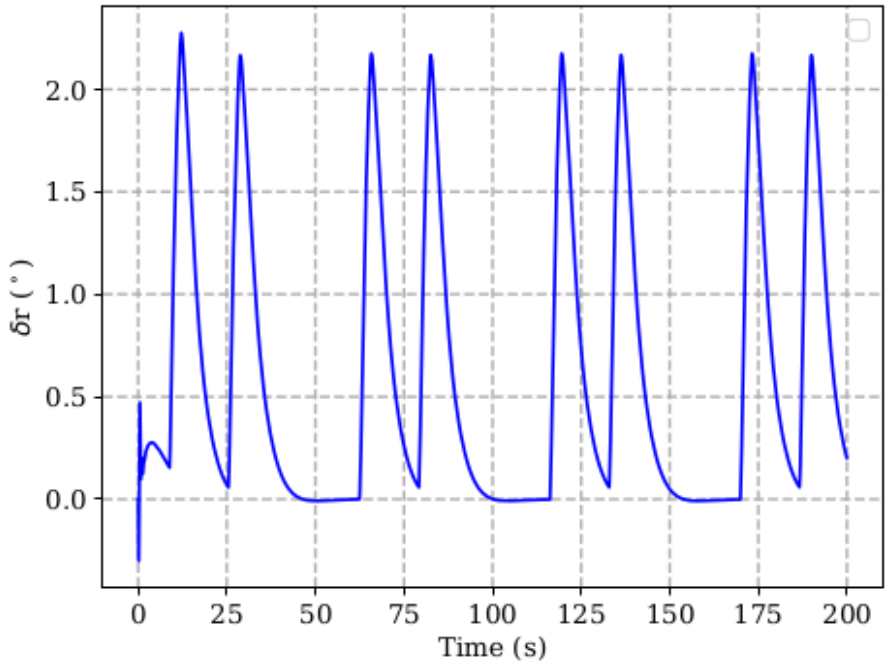

0.4
0.3
0.2
0.1
0.0
β (°)
0
25
50
75
100
125
150
175
200
Time (s)

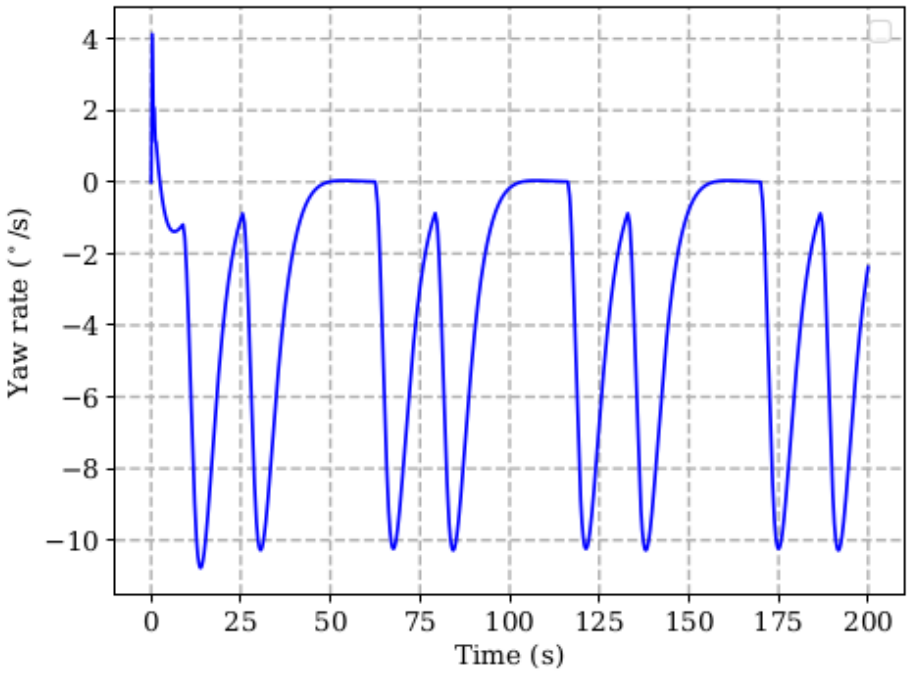

Trajectory
North (ft)
East (ft)
800
600
400
200
0
−200
−400
−600
−800
−1000
−750
−500
−250
0
250
500
750
1000


Roll rate (°/s)
Time (s)
P
3
2
1
0
−1
−2
−3
−4
−5
0
25
50
75
100
125
150
175
200


ϕ
ϕ_{cmd}
ϕ (°)
5
0
−5
−10
−15
0
25
50
75
100
125
150
175
200

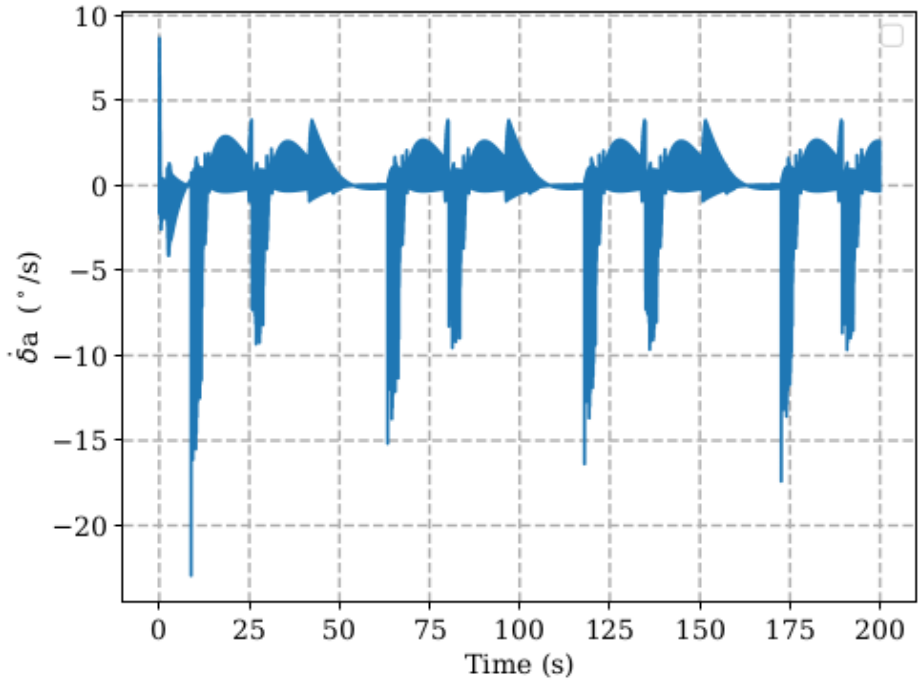

$\dot{\delta}$r (°/s)
4
2
0
−2
−4
0
25
50
75
100
125
150
175
200
Time (s)

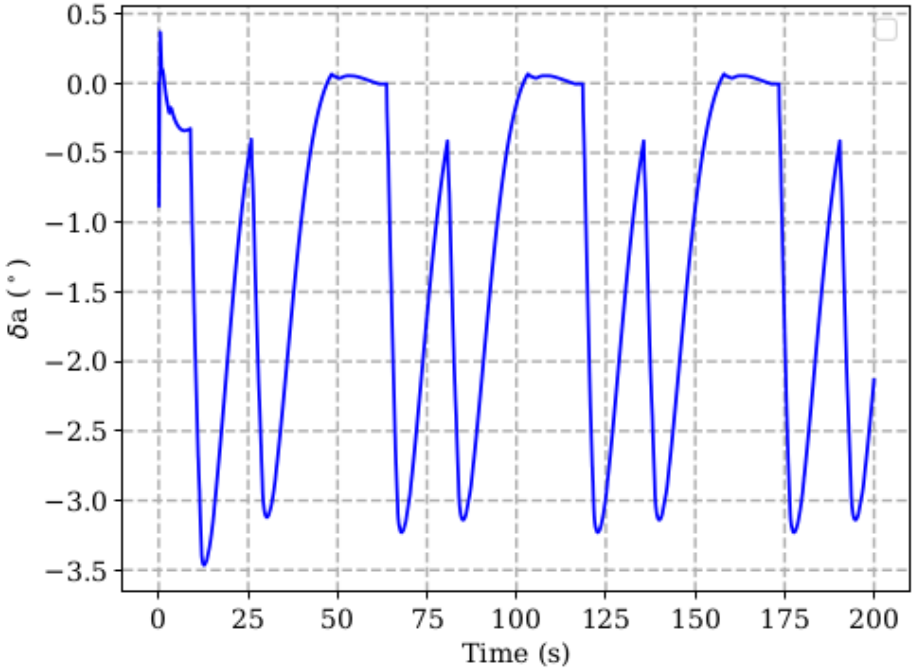

2.0
1.5
1.0
0.5
0.0
δr (°)
0
25
50
75
100
125
150
175
200
Time (s)

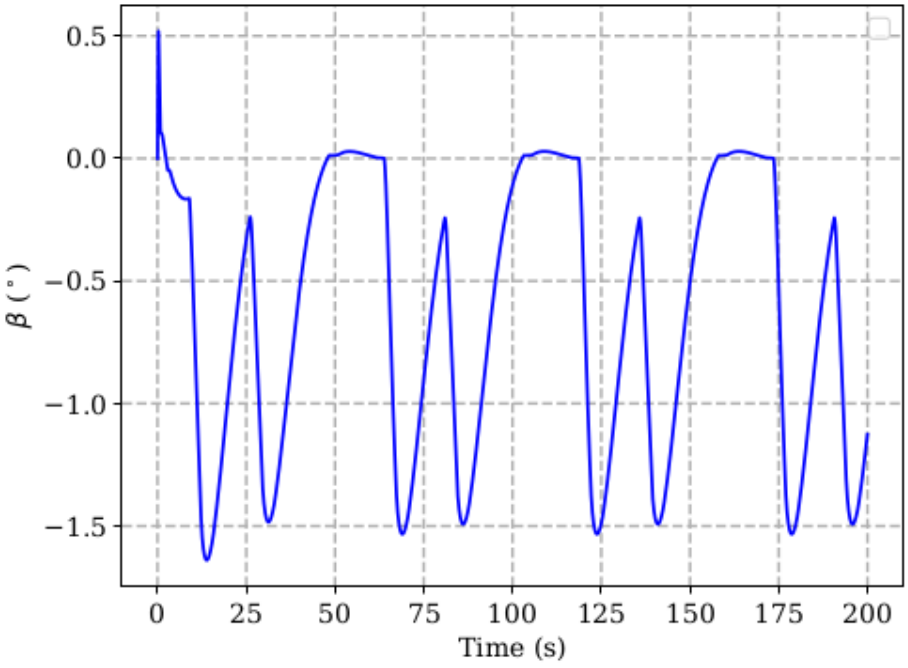

4
2
0
−2
−4
−6
−8
−10
Yaw rate (° /s)
0
25
50
75
100
125
150
175
200
Time (s)

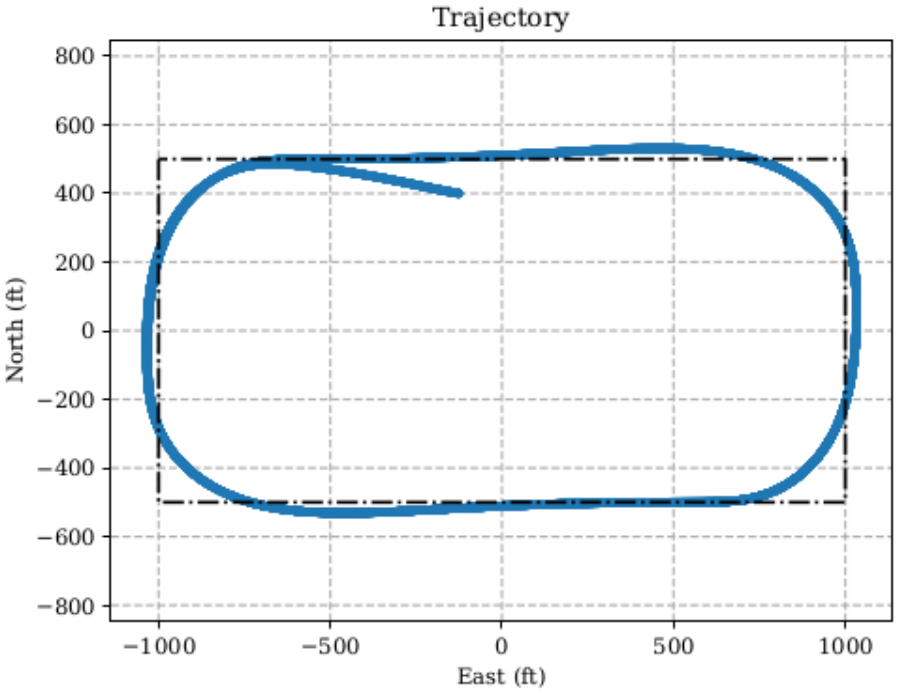
Appendix C

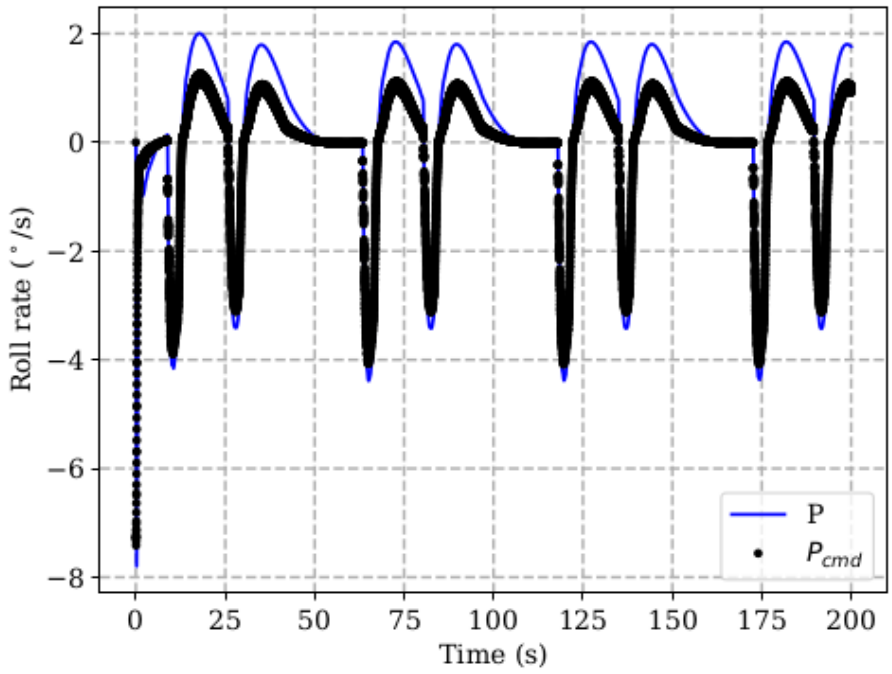
Dynamic Inversion Lateral Plots for 50% Aileron Degradation

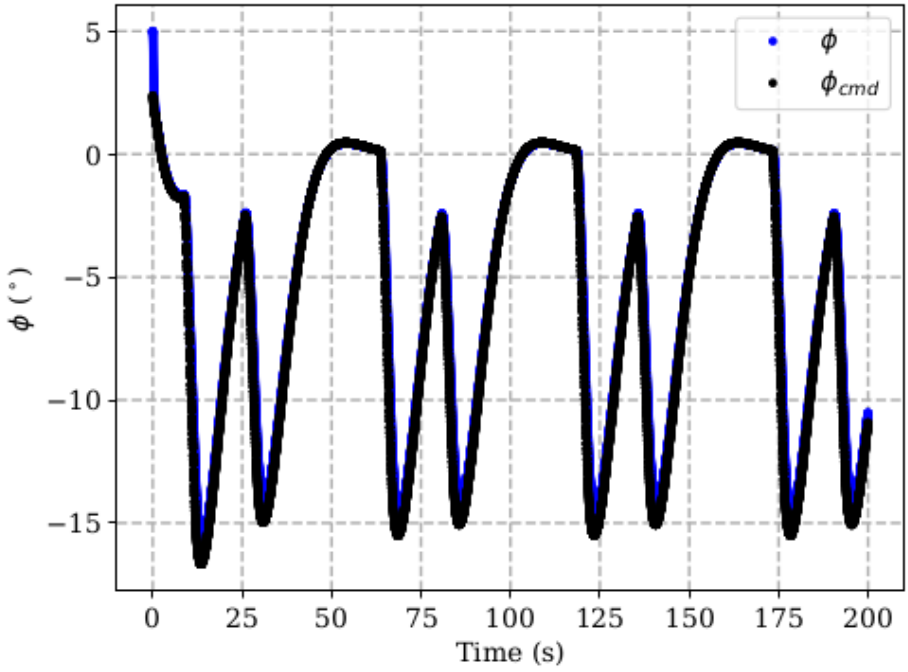


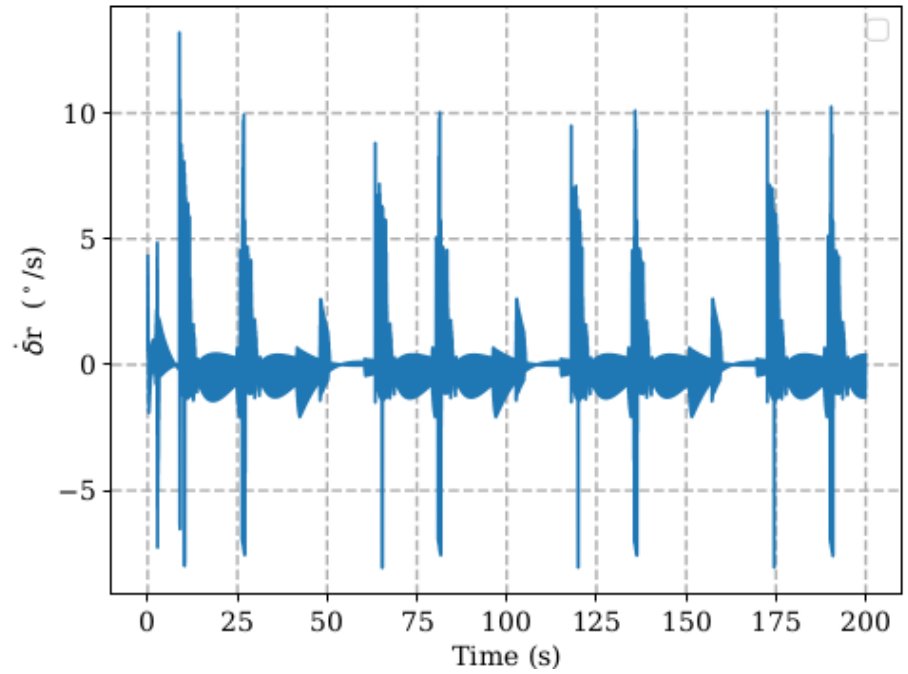

0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
δa (°)
0
25
50
75
100
125
150
175
200
Time (s)

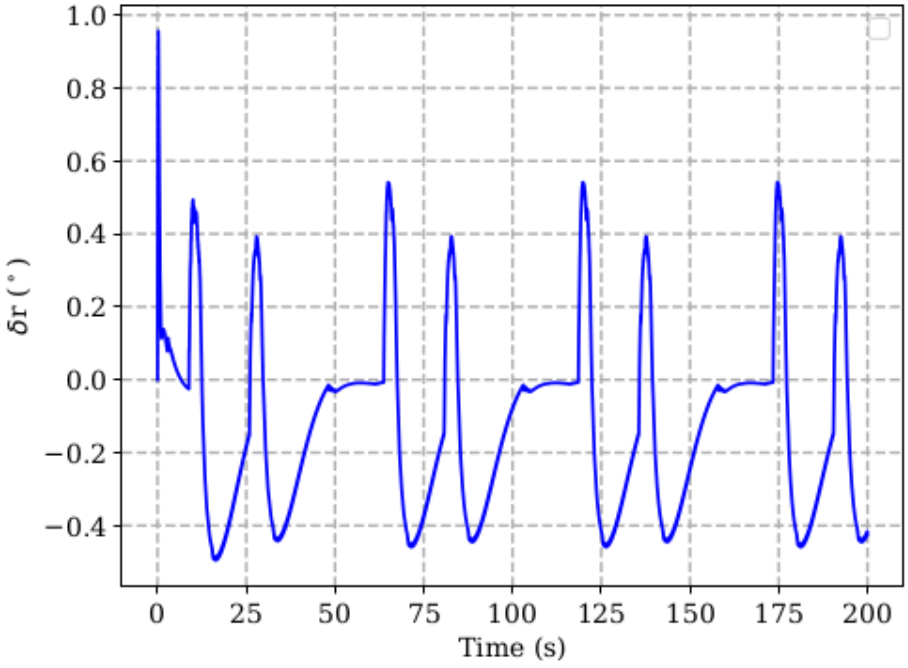

0.5
0.0
−0.5
−1.0
−1.5
β (°)
0
25
50
75
100
125
150
175
200
Time (s)

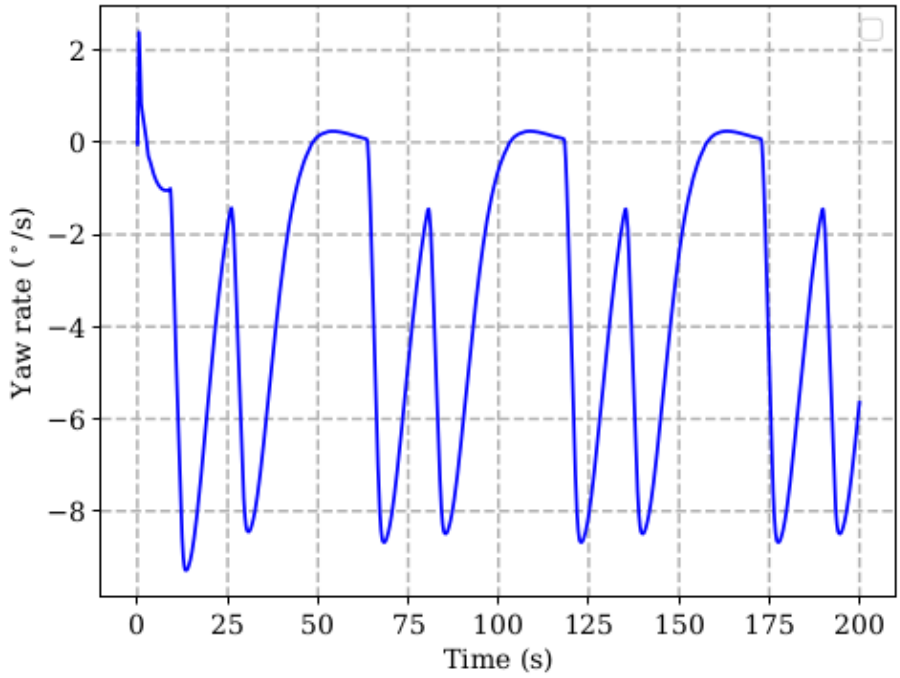

Trajectory
North (ft)
East (ft)
800
600
400
200
0
−200
−400
−600
−800
−1000
−500
0
500
1000


Roll rate (° /s)
Time (s)
P
P_{cmd}
2
0
−2
−4
−6
−8
0
25
50
75
100
125
150
175
200


ϕ
ϕ_{cmd}
5
0
−5
−10
−15
ϕ (°)
0
25
50
75
100
125
150
175
200
Time (s)


10
5
0
−5
$\dot{\delta}$r (°/s)
0
25
50
75
100
125
150
175
200
Time (s)


1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
δr (°)
0
25
50
75
100
125
150
175
200
Time (s)


2
0
−2
−4
−6
−8
Yaw rate (°/s)
0
25
50
75
100
125
150
175
200
Time (s)