# PROPELLER FAULT DIAGNOSIS IN MULTIROTOR UAVS: A DATA-DRIVEN SYSTEM DYNAMICS APPROACH

BY

**MOHSSEN ELSHAAR**

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

In Partial Fulfillment of the Requirements
for the Degree of

**MASTER OF SCIENCE**

**IN**

**AEROSPACE ENGINEERING**

KING FAHD UNIVERSITY
OF PETROLEUM & MINERALS

Dhahran, Saudi Arabia

MARCH 2025

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **MOHSSEN ELSHAAR** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN AEROSPACE ENGINEERING**.

**Thesis Committee**

Dr. Ayman Abdallah (Adviser)

Prof. Amro Al-Qutub (Member)

Dr. Mohamed Ismail (Member)

Dr. Ayman Abdallah
Department Chairman

Dr. Suliman Al-Homidan
Dean of Graduate Studies

Date

*To the one whose time, wisdom, and determined support have shaped the person I am — who, even in his darkest moments, in the depths of unbearable conditions, still found the strength to stand by me.*
*Your sacrifices were a continuous rhythm, weaving through every challenge I faced. Your belief in me never faltered, even when I stumbled and made mistakes. Your presence a refuge, your strength a lesson, your kindness a warmth that taught me how to be kind.*
*This work is not mine; it carries the imprint of your endurance, your generosity, and your unshaken resolve.*
*To my father,*
*I dedicate this to you.*

# ACKNOWLEDGMENTS

*To my mother, whose unwavering love and quiet strength have been my foundation, your selflessness has been my greatest support.*

*To my advisors, Dr. Ayman and Dr. Mohamed, without whom this work would not have been possible, your guidance has been invaluable.*

*To Professor Amro, whose experience and sharp insights have added much to this work.*

*To Balshy, Zeyad, Belal, Meral, and Pansie, who filled this journey with relentless discussions, shared frustrations, and well-timed nagging, your companionship made the process bearable.*

*To my dearest Ruaa, whom I hope to soar with and find home.*

*Thank you all.*

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# THESIS ABSTRACT


**NAME:** Mohssen ELshaar


**TITLE OF STUDY:** Propeller Fault Diagnosis in Multirotor UAVs: A Data-Driven System Dynamics Approach

**MAJOR FIELD:** Aerospace Engineering

**DATE OF DEGREE:** March 2025


*Propeller faults in multirotor UAVs pose a significant challenge to flight stability and control, necessitating reliable fault detection and diagnosis methods. This study presents a data-driven system dynamics approach for propeller fault diagnosis, leveraging system identification techniques to establish a healthy drone model baseline. Controlled fault injection experiments are conducted to analyze deviations in dynamic response, utilizing residual-based marginal analysis for fault detection. The study focuses on three of the most common and distinct propeller fault types: edge cuts, cracks, and surface unbalances. Each fault affects the propeller differently, with edge cuts altering aerodynamic properties, cracks reducing structural stiffness, and surface unbalances introducing mass asymmetry. Faults are classified based on their characteristics, and severity trends are studied through variations in residual margins.*

*Experimental results demonstrated the effectiveness of residual-based marginal analysis in detecting faults, grouping them into distinct categories, and analyzing their severity trends. The findings contribute to improved UAV reliability by enabling early fault detection and classification, enhancing predictive maintenance strategies in autonomous aerial systems.*

# ملخص الرسالة

**الاسم:** محسن الشعار

**عنوان الدراسة:** تشخيص أعطال المراوح في الطائرات متعددة الدوارات: بالاعتماد على ديناميكيات النظم البياناتية

**التخصص:** هندسة الطيران والفضاء

**تاريخ الدرجة العلمية:** مارس 5202

تمثل أعطال المراوح الدافعة في الطائرات متعددة الدوّارات تحديًا كبيرًا لاستقرار الطيران والتحكم، مما يستدعي تطوير منهجيات موثوقة للكشف عن الأعطال وتشخيصها. يقدم هذا البحث نهجًا قائمًا على تحليل ديناميكيات النظم لتشخيص أعطال المراوح، بالاعتماد على التقنيات البياناتية لتحديد النظام، وذلك لإنشاء نموذج أساسي للطائرة في الحالة السليمة. نفذت تجارب محكمة لإدراج الأعطال المختلفة، ومن ثم محاولة كشفها. يكون كشف الأعطال من خلال تحليل انحرافات الاستجابة الديناميكية في حالة كل عطل، وذلك بالمقارنة الهامشية بين استجابات النموذج السليم واستجابات النظم الموسومة بالأعطال، وتحليل الفروقات الهامشية الناتجة عن تلك المقارنة. يركز البحث على ثلاثة أعطال، هم من أكثر أنواع أعطال المراوح شيوعًا وتميزًا: القطوع الحدية، والشقوق، والاختلالات السطحية. لكل نوع من هذه الأعطال تأثير مختلف على المروحة؛ حيث تؤثر القطوع الحدية على الخصائص الديناميكية الهوائية، بينما تضعف الشقوق من الصلابة الهيكلية، في حين تؤدي الاختلالات السطحية إلى فقد التوازن الكتلي. صنفت الأعطال بناءً على خصائصها، ودرست العلاقة بين شدة الأعطال والتغيرات الملحوظة في الفروقات الهامشية. أظهرت النتائج التجريبية فعالية تحليل الفروقات الهامشية في الكشف عن الأعطال وتصنيفها ضمن فئات متميزة، بالإضافة إلى تحليل اتجاهات وتأثير شدتها. تساهم هذه النتائج في رفع موثوقية الطائرات بدون طيار، وفي إتاحة الاكتشاف المبكر للأعطال وتصنيفها، مما يعزز استراتيجيات الصيانة التنبؤية في النظم الجوية الآلية.

# CHAPTER 1

# BACKGROUND

This chapter provides an overview of fault detection in multirotor UAVs, focusing on different fault types and their classification. It introduces mathematical modeling techniques used to represent UAV faults, distinguishing between additive and multiplicative faults. The chapter then explores various Fault Detection and Identification (FDI) methods, including model-based, signal-based, and AI-driven techniques, each with its own advantages and challenges. A comparative analysis of these approaches highlights the trade-offs between computational efficiency, detection accuracy, and real-time applicability. By establishing a theoretical foundation for fault detection in UAVs, this chapter lays the groundwork for the following discussions on UAV modeling, experimental validation, and the development of robust fault detection frameworks.

## 1.1 Introduction

As technology advances, unmanned aerial vehicles (UAVs) benefit from improved flight times, enhanced power units, and sophisticated autonomous modes, broadening their applications. Multicopters, known for their exceptional maneuverability due to the independent control of each motor's RPM, are particularly susceptible to damage because their propulsion systems also serve as lift surfaces. Propellers, often made of plastics or carbon laminates [**?**], are prone to wear and damage from abrasive particles, collisions, and adverse weather conditions [**?**]. Such damage can lead to increased vibration [**?**], reduced battery life [**?**], and potential UAV crashes. With the growing use of UAVs, ensuring the reliability of propulsion systems has become critical. Effective monitoring and preventive measures are necessary to prevent severe damage or crashes. Current research emphasizes vibration characteristics and sensor-based monitoring for propeller failures [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**]. However, to the best of our knowledge, existing studies often focus on isolated experiments and fail to comprehensively address various fault types and severity levels within a single framework. Moreover, only a few works have explored fault analysis and detection through flight dynamics [**?**], with even fewer bridging the gap between experimental findings and model-based detection techniques grounded in flight dynamics [**?**], [**?**]. A critical gap in the literature is the lack of attention to subtle, hard-to-detect faults that do not significantly impair a drone's ability to maintain accurate flight. Such

faults, while seemingly minor, are crucial for advancing fault detection techniques and ensuring the reliability and safety of UAV operations.

## 1.2 Fault Classification in Multirotor UAVs

In multirotor UAVs, a fault is identified as a significant deviation of at least one key variable from its expected behavior [**?**]. The primary types of faults include communication failures, software malfunctions, and physical defects. Communication issues can occur due to signal interference, router problems, loss of range, or delays in data transmission [**?**]. Software malfunctions, such as errors in navigation and stabilization algorithms like the Kalman filter or improper parameter initialization [**?**], can greatly impact the UAV's performance. Moreover, changes in system characteristics—such as variations in payload—result in non-linear dynamic flight scenarios that linear control laws find difficult to handle, potentially leading to faults in the stabilization algorithm [**?**].

Physical defects, or hardware faults, are those appearing in the system's physical components [**?**]. They are categorized based on the defective component and the mathematical modeling of the failure [**?**]. Depending on their behavior over time, hardware faults are further classified as abrupt, intermittent, or incipient. Abrupt faults, such as actuator failures [**?**] or sensor bias faults [**?**], occur suddenly and are usually easier to detect but can lead to equipment damage if persistent. Intermittent faults are repetitive and reset after a random interval [**?**]. Incipient faults develop slowly, leading to gradual degradation, making them harder to detect but inevitable

due to wear and tear in mechanical components like motors and blades [**?**].

### 1.2.1 Mathematical Modeling of Faults

Mathematically, faults are generally classified as either additive or multiplicative [**?**].

**Additive faults** Additive faults are modeled as additional constant or time-varying terms in the system equations, often referred to as bias or offset terms. These faults appear in actuators or sensor offsets and can be represented as:

$$\zeta(t) = y(x(t)) + b + n(t)$$

where $\zeta(t)$ is the measured value, $y(x(t))$ is the nominal output of the system, $b$ is the bias term representing the fault, and $n(t)$ is the measurement noise [**?**].

**Multiplicative faults** On the other hand, multiplicative faults scale or multiply the values of certain parameters or variables in the system's equations. These faults describe degradation in actuators or sensors and can be modeled as:

$$\zeta(t) = \lambda y(x(t))$$

where $\lambda$ is a scaling factor that might depend on time, state variables, inputs, or other parameters affecting the system [**?**].

### 1.2.2 Sensor Faults

Sensor faults can lead to performance degradation and potential loss of control. Common issues include complete or partial sensor failures that hinder accurate data collection. Data drift, a gradual change in sensor output, can result in incorrect readings over time due to environmental influences or aging components. Bias errors, which are systematic discrepancies between measured and true values, are often caused by calibration problems. Additionally, noise interference caused by random fluctuations distorts sensor readings, and is frequently due to electrical interference or environmental conditions [**?**]. Specific examples include height sensor faults, where barometric pressure is affected by airflow [**?**]; IMU faults, appearing as bias in accelerometer and gyroscope measurements [**?**]; and position sensor faults, such as GPS, due to degradation, environmental factors, and operational stress [**?**].

### 1.2.3 Component and Actuator Faults

Component and actuator faults in multirotor UAVs primarily affect mechanical parts, such as the airframe, motors, propellers, and electronic speed controllers (ESCs). Frame damage often results from collisions or extreme weather, leading to material fatigue over time [**?**]. High-speed motors and propellers are susceptible to performance issues caused by vibrations, which can lead to potential loss of control (LOC). Motor failures can degrade propeller performance due to prolonged use or particles in the motor housing, reducing thrust and possibly causing seizing [**?**]. ESCs are vulnerable to overheating, which can disrupt power distribution and affect flight stability.

Failures may arise from improper startup sequences, environmental hazards, or full degradation [?]. Finally, batteries may pose significant fire hazards if damaged or improperly handled [?]. According to [?], 19% of research articles focusing on a specific group of drone subsystems that investigate actuator faults have studied propeller faults (Figure 1.1). This indicates that nearly one-fifth of current research efforts in actuator fault diagnosis are dedicated to propeller-related issues, highlighting their significance in UAV reliability and performance. The substantial proportion of studies in this area underscores the critical importance of advancing fault detection and mitigation strategies for propellers.

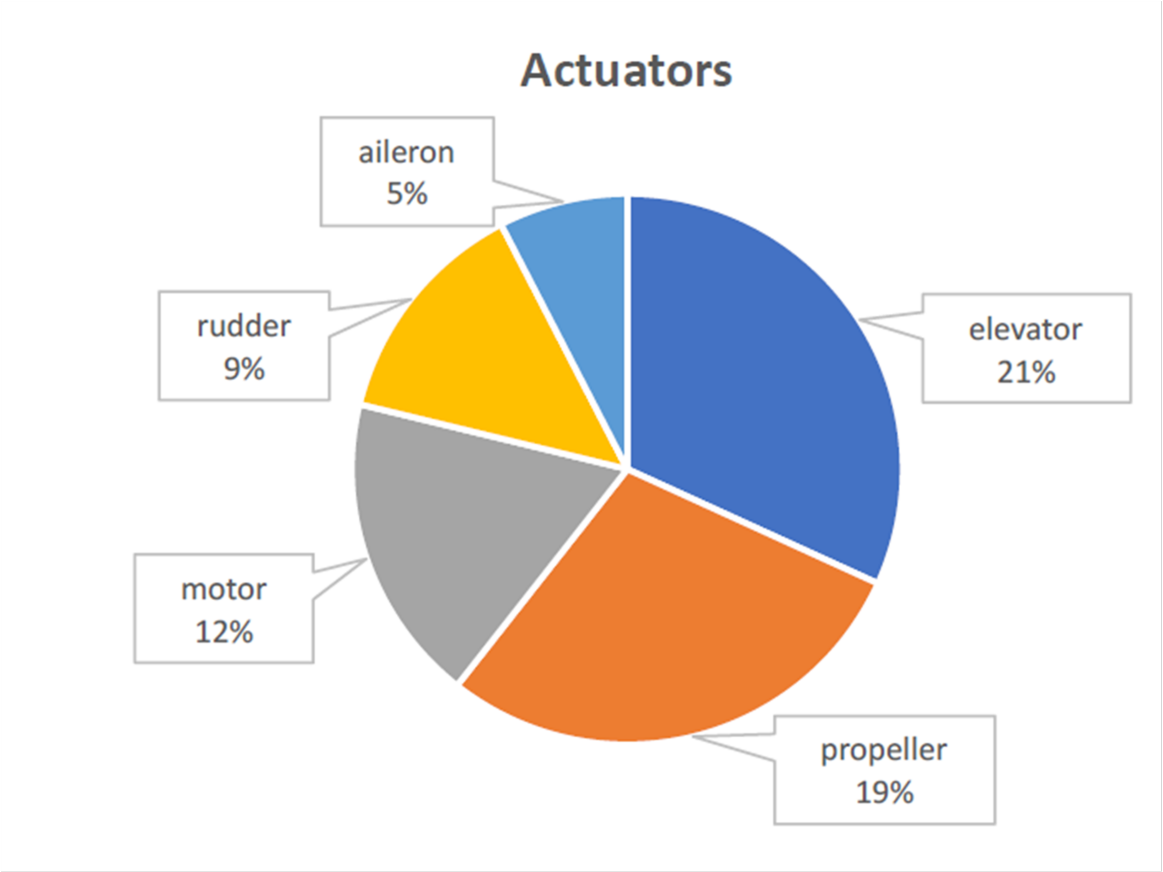



Figure 1.1: Distribution of research articles on actuator faults across drone subsystems. Propeller faults account for 19% of all studied actuators.

### 1.2.4 Propeller Fault Injection

In simulation, propeller damage can be represented simply by manipulating motor speeds or by altering the produced thrust [**?**]. However, injecting faults in hardware presents more challenges. As an intermediate step, fault emulation using filters and mathematical dynamics was introduced as a bridge between model-in-the-loop and hardware-in-the-loop systems in [**?**], where propeller slippage damage was modeled. Actual (experimental) propellers fault injection in the literature can be summarized by the methods presented in the following subsection.

- **Blade Chipping**: A small portion of the propeller blade is chipped or broken [**?**], [**?**], [**?**], [**?**], [**?**].
- **Blade Bending**: A part of the blade tip is bent by different angles [**?**].
- **Mass UnBalance**: Small weights are attached to one side of the blade to simulate unbalanced propeller dynamics [**?**], [**?**]. In [**?**] wing turbine blade faults were injected by creating unbalances on a single blade, either by adding mass to mimic conditions like dirt accumulation or by removing mass to replicate scenarios such as a broken blade.
- **Aerodynamic Properties Manipulation**: Textures are added to the propeller blades, either symmetrically or asymmetrically, to alter lift and drag generation [**?**].
- **Motor Input Corruption**: Intentionally altering the motor input to cause it

to spin slower or faster than intended [?].

- **Physical Detachment**: To test the complete failure of a propeller, where it can no longer generate lift, the motor voltage is set to zero, as demonstrated in [?], [?]. A propeller ejection method is proposed in [?], where friction is removed from the locking mechanism, allowing aerodynamic forces to initially keep the propeller attached. The propeller is then ejected by rapidly reducing motor speed, leveraging the fact that the propeller's rotational deceleration from aerodynamic torque is slower than the motor's deceleration.

All of these methods were used to consider different combinations of defective propellers and their effect on the flight performance (e.g. Fig 1.2).

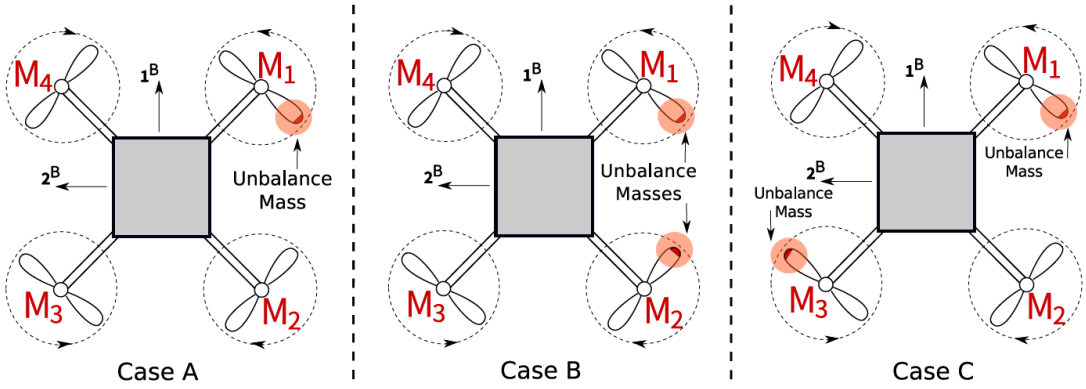
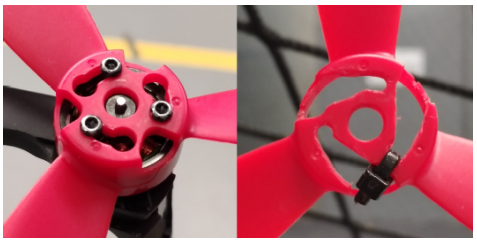



Figure 1.2: Fault combinations using different numbers and locations of defective propellers [?].

## 1.3 Fault Detection and Identification

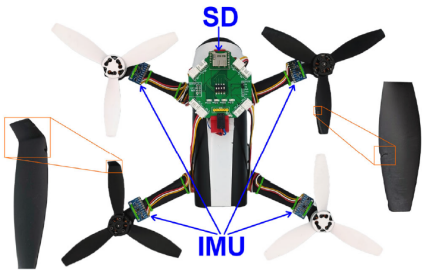
Fault detection and identification (FDI) methods are classified model-based, signal-based, AI-based, and hybrid methods [?]. Each method has unique advantages and


(a) Tip chipping [**?**].



(b) Friction removal from the propeller lock [**?**].

(c) Texture addition using tape [**?**].




(d) Tip bending and edge chipping [**?**].

Figure 1.3: Propeller faults in literature.

challenges, particularly when applied to nonlinear systems like quadrotors.

### 1.3.1 Model-Based Methods

Model-based methods use mathematical models of system dynamics to detect faults by comparing the expected system behavior with actual measurements. Discrepancies, known as residuals, signal the presence of faults [**?**]. The residual $\mathbf{r}(t)$ is defined as:

$$\mathbf{r}(t) = \mathbf{y}(t) - \hat{\mathbf{y}}(t) \tag{1.1}$$

$$\|\mathbf{r}(t)\| > \epsilon \tag{1.2}$$

where $\mathbf{y}(t)$ is the measured output (position, velocity, orientation), $\hat{\mathbf{y}}(t)$ is the output predicted by the model, and $\epsilon$ is a predefined threshold.

Model-based methods are effective for early fault detection and identification but require accurate models of the system. They include Kalman-based methods, observer-based methods, and multiple model-based methods, each suited to different scenarios based on system complexity and computational requirements.

### Kalman-Based Methods

Kalman-based methods are widely used in control systems for state estimation, where the goal is to estimate the internal states of a system (e.g., position, velocity) based on noisy sensor measurements. In fault detection, Kalman Filters (KF) estimate the system state and compute the residuals (the difference between expected and actual sensor data). Significant deviations in these residuals are indicative of faults.

- Kalman Filter (KF): This filter is optimal for systems with linear dynamics and Gaussian noise. It assumes that both the process noise (uncertainties in system dynamics) and measurement noise follow Gaussian distributions. By continuously updating state estimates based on noisy sensor inputs, the KF provides an estimate of the true state. However, quadrotor dynamics are often nonlinear, which limits the KF's direct application [**?**].

- Extended Kalman Filter (EKF): The EKF extends the Kalman filter to nonlinear systems by linearizing the system dynamics around the current state esti-

mate. At each step, the EKF approximates the nonlinear system with a linear model, enabling it to handle systems with moderate nonlinearities. However, this approximation can introduce errors, especially in highly nonlinear systems, and it may not be sufficient to detect faults accurately in such cases [**?**].

- Unscented Kalman Filter (UKF): The UKF improves upon the EKF by using a more sophisticated method for estimating the distribution of states. It uses a deterministic sampling technique (unscented transform) to better propagate the state and noise distributions through nonlinear dynamics. The UKF generally provides more accurate state estimates than the EKF, but it comes with increased computational cost, making it more challenging to implement in real-time on systems with limited processing power [**?**].

Kalman-based methods are suitable for systems with well-modeled dynamics and known noise characteristics. However, in quadrotors, where external disturbances and complex non-linearities exist, these methods may struggle without appropriate adaptations. This limitation has prompted the development of Particle Filters [**?**] and advanced variants to improve fault detection under such conditions.

### Observer-Based Methods

Observer-based methods are another class of fault detection techniques that involve estimating the system's internal state using a state observer. The observer generates an estimate of the system states based on inputs and outputs, and the difference between estimated and actual outputs is used to detect faults.

- State Observer: A state observer is a dynamic system that replicates the behavior of the actual system. By observing the system inputs and outputs, it estimates the internal states. When a fault occurs, there is a discrepancy between the estimated and measured outputs, known as the residual, which signals a fault.

- Sliding Mode Observer (SMO): The SMO is particularly well-suited for systems with high levels of uncertainty or disturbance. It is designed to handle model inaccuracies and external disturbances by applying a discontinuous control action that forces the system to "slide" along a set of desired states. This makes the SMO robust to disturbances and modeling errors, which are common in quadrotor systems [**?**].

Observer-based methods provide robust fault detection in systems with non-linear dynamics, but they require an accurate mathematical model of the system to generate reliable state estimates.

### Multiple Model-Based Methods

Multiple model-based methods extend fault detection by using a set of models, each corresponding to a different possible system state (healthy or faulty). The system behavior is compared against all models simultaneously, and the model that best matches the observed behavior indicates the current state of the system. The method uses a bank of models, where each model corresponds to a different fault condition (e.g., normal operation, partial rotor failure, full rotor failure). By calculating the

residuals for each model and comparing them to the observed system outputs, the method identifies which model fits best. While this approach provides accurate fault identification, it is computationally expensive, especially in real-time applications like quadrotor flight. This is because multiple models need to run in parallel, each processing inputs and generating residuals for comparison. Optimizing model selection (using fewer, simpler models) can help mitigate this computational burden [**?**].

Multiple model-based methods offer high precision for fault identification but are often computationally demanding, limiting their practical use in resource-constrained environments.

### 1.3.2 Signal-Based Methods

Signal-based methods are commonly used for fault detection by analyzing the signals (sensor outputs) generated by the system. These methods detect abnormalities or deviations from normal signal patterns, which indicate faults.

- Threshold-Based Methods: This is one of the simplest signal-based methods. Predefined thresholds are set for key system variables (e.g., rotor speed, temperature), and any deviation beyond the threshold is flagged as a fault. While this method is computationally inexpensive, it can lead to false positives in noisy environments, which are common in quadrotors [**?**].

- Wavelet Transforms and PCA: More advanced signal processing techniques like

Wavelet Transforms decompose signals into different frequency components, enabling the detection of sudden changes that may indicate faults. Similarly, Principal Component Analysis (PCA) can reduce noise and improve the accuracy of fault detection by isolating significant signal components from less important ones. Wavelet transforms are particularly effective in quadrotors, where dynamic flight operations generate non-stationary signals [**?**].

Signal-based methods are fast and straightforward to implement, but they can be sensitive to noise, which may affect the accuracy of fault detection in dynamic and harsh environments like those faced by quadrotors.

**Vibration-Based Methods Integrated with Signal Analysis**

Mechanical components such as propellers and motors often produce distinct vibration patterns during faults. Vibration-based methods analyze these patterns to detect mechanical issues.

- Spectral Analysis (FFT + PCA): Fast Fourier Transform (FFT) is used to convert time-domain vibration signals into the frequency domain. When combined with Principal Component Analysis (PCA), it can detect specific vibration frequencies associated with faults such as damaged propellers [**?**].

- Trajectory-Based Vibration Methods: These methods rely on vibration data collected during different flight maneuvers. By analyzing how the vibrations change with motor speed or flight conditions, trajectory-dependent faults (e.g., rotor imbalance) can be detected. However, the effectiveness of this approach

is highly dependent on flight trajectory, which may limit its use in general fault detection [**?**].

- Discrete Wavelet Transform (DWT) + FFT: This hybrid approach integrates Wavelet Transforms with FFT to filter noise and detect subtle changes in vibration signals, providing better accuracy in identifying both minor and major mechanical faults in real-time [**?**].

Vibration-based methods are highly effective for detecting mechanical faults, especially in components like propellers. However, their reliance on specific flight conditions and trajectories can limit their general applicability.

### 1.3.3 AI-Based Methods for FDI

Artificial intelligence (AI) methods, particularly machine learning models, are increasingly used for fault detection due to their ability to learn complex patterns from large datasets. These methods can either complement traditional model-based approaches or function as standalone systems for FDI.

#### Neural Networks and Deep Learning Approaches

Artificial Neural Networks (ANNs) are widely used for processing sensor data to detect faults. These models excel at identifying patterns in noisy, high-dimensional data, making them ideal for detecting complex, non-linear faults in quadrotors. In [**?**], ANNs were applied to classify vibration data and detect faults such as motor degradation and propeller damage. ANNs learn to recognize fault signatures directly from the

data, making them highly adaptable to different flight environments and operational conditions.

Neural networks are powerful tools for fault detection, but they require large datasets for training and may be computationally expensive to implement in real-time systems.

### Fuzzy Logic and Neuro-Fuzzy Systems

Fuzzy Logic Systems (FLS) handle uncertainty and imprecision in data, making them useful in environments where sensor readings are noisy or incomplete, such as quadrotors. Neuro-Fuzzy Systems (NFS) combine the adaptability of neural networks with the decision-making process of fuzzy logic to handle complex fault scenarios in real-time. In [**?**], a neuro-fuzzy system was used to process vibration data from multirotor arms. This approach achieved high fault detection accuracy, even in challenging environments with noisy sensor data.

### Reinforcement Learning in FDI

Reinforcement Learning (RL) is an emerging AI method where a learning agent interacts with its environment and learns to detect faults by trial and error. Over time, the agent optimizes its fault detection strategy by rewarding accurate fault identification and penalizing incorrect detections. RL is particularly useful in dynamic environments where predefined models cannot capture all variations of possible faults.

#### AI for Vibration Analysis

AI techniques such as neural networks and fuzzy logic are highly effective for processing large amounts of vibration data, identifying subtle patterns that traditional signal-processing methods might miss. By combining AI techniques with Wavelet Transforms, more accurate fault detection systems can be created [**?**], [**?**].

### 1.3.4 Hybrid Methods

Hybrid methods combine model-based approaches (e.g., Kalman Filters) with signal-based approaches (e.g., wavelet analysis) to leverage the advantages of both. For example, a Kalman Filter can be used for initial fault detection, and the fault can then be confirmed using wavelet analysis [**?**]. Hybrid methods aim to balance accuracy and response time but are often complex to implement, particularly in real-time systems.

### 1.3.5 Comparison of Methods

Table 1.1 provides a summary of the various methods discussed in this review, comparing their advantages and challenges.

Table 1.1: Comparison of Fault Detection and Identification (FDI) Methods

| **Method** | **Advantages** | **Challenges** |
|---|---|---|
| Kalman-Based | Accurate for linear systems, good for state estimation | Struggles with nonlinearities, requires known noise characteristics |
| Observer-Based | Effective for nonlinear systems, robust to uncertainties | Requires an accurate system model, sensitive to modeling errors |
| Multiple Model-Based | High precision, capable of detecting multiple fault types | Computationally expensive, requires multiple model evaluations |
| Signal-Based | Fast and simple, does not require system modeling | Sensitive to noise, may produce false positives |
| Vibration-Based | Effective for detecting mechanical and structural faults | Dependent on flight conditions and operational trajectory |
| AI-Based | Learns from complex patterns, adaptable to various fault conditions | Requires large datasets, computationally expensive |
| Hybrid | Combines strengths of multiple approaches, improves fault detection accuracy | Complex to implement, may require high computational resources |

# Summary

This chapter introduced the importance of UAV reliability and the increasing need for advanced fault detection techniques. It also provided a comprehensive background on fault detection in multirotor UAVs, with a specific focus on propeller fault detection.

## Fault Classification in Multirotor UAVs

Faults in multirotor UAVs were classified into three main categories:

- **Sensor Faults:** Errors in measurement due to sensor drift, noise interference, or complete sensor failure.
- **Component and Actuator Faults:** Physical damage to UAV components, including motors, electronic speed controllers (ESCs), and propellers, often leading to stability issues.
- **Propeller Faults:** Specific faults affecting the propulsion system, such as unbalance, cracks, or material loss, which impact UAV performance and safety.

The mathematical modeling of faults was also discussed, distinguishing between additive faults (biases in system output) and multiplicative faults (scaling effects on system parameters).

## Fault Detection and Identification (FDI) Methods

Various FDI approaches were analyzed:

- **Model-Based Methods:** These rely on system dynamics and residual analysis to detect faults. Techniques such as Kalman filtering, state observers, and multiple-model-based estimation were discussed.
- **Signal-Based Methods:** These analyze sensor data trends using threshold-based techniques, Fourier transforms, and wavelet decomposition to identify anomalies.
- **AI-Based Methods:** Machine learning models, including artificial neural networks (ANNs), fuzzy logic, and reinforcement learning, are used to classify and predict UAV faults.
- **Hybrid Methods:** A combination of model-based and signal-based techniques to improve detection accuracy and robustness in complex UAV environments.

The discussion of FDI techniques laid the foundation for later chapters, where specific fault detection methodologies will be implemented and experimentally validated.

# CHAPTER 2

# QUADCOPTER DYNAMICS AND CONTROL

Accurate modeling and control are essential for ensuring stable flight performance and effective fault detection in UAVs. This chapter introduces the Quanser QDrone 2, a research-oriented quadcopter platform used for experimental validation of UAV fault detection techniques. The chapter first explores the quadrotor dynamics, discussing fundamental force and moment equations that govern UAV motion. It then details the visual pose estimation system, which utilizes motion capture technology to track the UAV's position and orientation with high precision. Finally, it presents the control architecture of QDrone 2, which employs a cascaded control strategy to maintain stability and respond to external disturbances. By understanding the dynamic behavior and control mechanisms of the QDrone 2, this chapter establishes a foundation for analyzing how faults affect UAV performance and how detection methods can be integrated into the control system.

## 2.1 Quadcopter Dynamics

A quadcopter has six degrees of freedom: three translational ($x$, $y$, $z$) and three rotational (roll $\phi$, pitch $\theta$, yaw $\psi$). However, due to its underactuated nature, direct control is achieved through four independent inputs: the angular speeds of the four rotors.

### 2.1.1 Free Body Diagram and Force Equations

The dynamics of the QDrone 2 quadcopter can be derived from its free body diagram, as shown in Figure 2.1. The net force acting on the quadcopter is the sum of the forces generated by the four motors, minus the gravitational force. This can be expressed as:

$$F_{\text{Net}} = F_{m1} + F_{m2} + F_{m3} + F_{m4} - F_g, \tag{2.1}$$

where:

- $F_{m1}, F_{m2}, F_{m3}, F_{m4}$ are the forces generated by motors 1, 2, 3, and 4, respectively,
- $F_g$ is the gravitational force acting on the quadcopter.

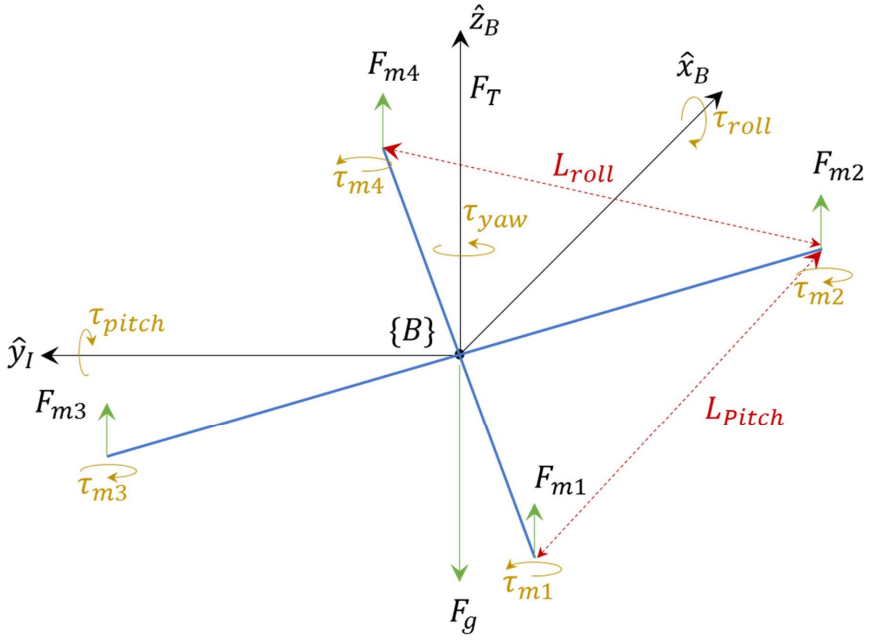



Figure 2.1: Free body diagram of the QDrone 2 quadrotor.

### 2.1.2 Moments and Torques

The desired moments about the three principal axes (roll, pitch, and yaw) are given by the following equations:

$$M_x = \tau_{\text{roll}} = \frac{L_{\text{roll}}}{2}(-F_{m1} - F_{m2} + F_{m3} + F_{m4}), \tag{2.2}$$

$$M_y = \tau_{\text{pitch}} = \frac{L_{\text{pitch}}}{2}(F_{m1} - F_{m2} + F_{m3} - F_{m4}), \tag{2.3}$$

$$M_z = \tau_{\text{yaw}} = \tau_{m1} - \tau_{m2} - \tau_{m3} + \tau_{m4}, \tag{2.4}$$

where:

- $L_{\text{roll}}$ and $L_{\text{pitch}}$ are the distances between the motors along the roll and pitch axes, respectively,

- $\tau_{m1}, \tau_{m2}, \tau_{m3}, \tau_{m4}$ are the torques generated by the motors.

The torque generated by each motor is proportional to the force it produces, and this relationship can be expressed as:

$$\tau = K_\tau F, \tag{2.5}$$

where $K_\tau$ is the torque constant. Using $K_{\tau 0}$ (determined experimentally), the yaw moment can be rewritten as:

$$M_z = \tau_{\text{yaw}} = K_{\tau 0}(F_{m1} - F_{m2} - F_{m3} + F_{m4}). \tag{2.6}$$

### 2.1.3 Motor Mapping Matrix

The forces and torques can be related to the motor forces through the motor mapping matrix:

$$\begin{bmatrix} F \\ \tau_{\text{roll}} \\ \tau_{\text{pitch}} \\ \tau_{\text{yaw}} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -\frac{L_{\text{roll}}}{2} & -\frac{L_{\text{roll}}}{2} & \frac{L_{\text{roll}}}{2} & \frac{L_{\text{roll}}}{2} \\ \frac{L_{\text{pitch}}}{2} & -\frac{L_{\text{pitch}}}{2} & \frac{L_{\text{pitch}}}{2} & -\frac{L_{\text{pitch}}}{2} \\ \frac{1}{K_{\tau 0}} & -\frac{1}{K_{\tau 0}} & -\frac{1}{K_{\tau 0}} & \frac{1}{K_{\tau 0}} \end{bmatrix} \begin{bmatrix} F_{m1} \\ F_{m2} \\ F_{m3} \\ F_{m4} \end{bmatrix}. \tag{2.7}$$

To calculate the desired motor thrust based on the higher-level controller outputs,

we use the inverse of the motor mapping matrix:

$$\begin{bmatrix} F_{m1} \\ F_{m2} \\ F_{m3} \\ F_{m4} \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & -\frac{1}{2L_{\text{roll}}} & \frac{1}{2L_{\text{pitch}}} & \frac{K_{\tau 0}}{4} \\ \frac{1}{4} & -\frac{1}{2L_{\text{roll}}} & -\frac{1}{2L_{\text{pitch}}} & -\frac{K_{\tau 0}}{4} \\ \frac{1}{4} & \frac{1}{2L_{\text{roll}}} & \frac{1}{2L_{\text{pitch}}} & -\frac{K_{\tau 0}}{4} \\ \frac{1}{4} & \frac{1}{2L_{\text{roll}}} & -\frac{1}{2L_{\text{pitch}}} & \frac{K_{\tau 0}}{4} \end{bmatrix} \begin{bmatrix} F \\ \tau_{\text{roll}} \\ \tau_{\text{pitch}} \\ \tau_{\text{yaw}} \end{bmatrix}. \tag{2.8}$$

### 2.1.4 Thrust to Voltage Conversion

Once the motor thrust values are calculated, they are converted to voltage commands using the relationship between thrust and voltage. The thrust $F_m$ is proportional to the square of the voltage $V$:

$$F_m = K_V V^2, \tag{2.9}$$

where $K_V$ is a constant that depends on the motor and propeller characteristics. The voltage command is then converted to a duty cycle percentage, which is sent to the electronic speed controller (ESC) to regulate the motor speed.

### 2.1.5 Mapping Angular Velocities to Forces

The forces and torques acting on the quadcopter are generated by the combined effect of the angular velocities (RPMs) of the four motors. Each motor's angular velocity $\omega_i$ (where $i = 1, 2, 3, 4$) is directly related to the voltage percentage $V_{\%,i}$ applied to the

motor. This relationship can be expressed as:

$$\omega_i = K_\omega V_{\%,i}, \tag{2.10}$$

where:

- $\omega_i$ is the angular velocity of the $i$-th motor in RPM,
- $V_{\%,i}$ is the voltage percentage applied to the $i$-th motor (ranging from 0% to 100%),
- $K_\omega$ is a constant that depends on the motor's characteristics and the battery voltage.

The force $F_{m,i}$ generated by each motor is proportional to the square of its angular velocity:

$$F_{m,i} = K_F \omega_i^2, \tag{2.11}$$

where $K_F$ is a constant that depends on the motor and propeller design. Combining Equations 2.10 and 2.11, the force generated by each motor can be expressed in terms of the voltage percentage:

$$F_{m,i} = K_F (K_\omega V_{\%,i})^2 = K_F K_\omega^2 V_{\%,i}^2. \tag{2.12}$$

The net force $F_{\text{Net}}$ and the moments $M_x, M_y, M_z$ are then determined by the combination of the forces generated by the four motors. Using the motor mapping

matrix (Equation 2.7), the net force and moments can be expressed as:

$$F_{\text{Net}} = F_{m1} + F_{m2} + F_{m3} + F_{m4}, \tag{2.13}$$

$$M_x = \tau_{\text{roll}} = \frac{L_{\text{roll}}}{2}(-F_{m1} - F_{m2} + F_{m3} + F_{m4}), \tag{2.14}$$

$$M_y = \tau_{\text{pitch}} = \frac{L_{\text{pitch}}}{2}(F_{m1} - F_{m2} + F_{m3} - F_{m4}), \tag{2.15}$$

$$M_z = \tau_{\text{yaw}} = K_{\tau 0}(F_{m1} - F_{m2} - F_{m3} + F_{m4}). \tag{2.16}$$

Substituting the motor forces from Equation 2.11, the net force and moments can be rewritten in terms of the motor angular velocities:

$$F_{\text{Net}} = K_F(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2), \tag{2.17}$$

$$M_x = \tau_{\text{roll}} = \frac{K_F L_{\text{roll}}}{2}(-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2), \tag{2.18}$$

$$M_y = \tau_{\text{pitch}} = \frac{K_F L_{\text{pitch}}}{2}(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2), \tag{2.19}$$

$$M_z = \tau_{\text{yaw}} = K_{\tau 0} K_F(\omega_1^2 - \omega_2^2 - \omega_3^2 + \omega_4^2). \tag{2.20}$$

These equations demonstrate how the net force and moments are a direct combination of the angular velocities of the four motors. By controlling the RPMs of the motors through voltage percentages, the quadcopter's motion can be precisely regulated.

## 2.2 Visual Pose Estimation of Rigid Bodies

Visual pose estimation is a cornerstone problem in computer vision and robotics, involving the determination of an object's position and orientation—collectively referred to as its *pose*—within a given reference frame. For rigid bodies, which do not deform during motion, the task simplifies to estimating a six-degree-of-freedom (6DoF) pose: three translational and three rotational parameters. These parameters allow for a comprehensive description of the object's location and orientation in space, which is critical for applications such as autonomous navigation, robotics, and augmented reality.

### 2.2.1 Problem Definition

The goal of rigid body pose estimation is to compute the transformation that aligns a known 3D model of an object to its 2D or 3D observations captured by sensors. This transformation includes translation, which represents the displacement along the X, Y, and Z axes, and rotation, which describes the orientation around these axes. Rotations can be represented using Euler angles, rotation matrices, or quaternions, depending on the application's requirements for computational efficiency and numerical stability.

Formally, given a set of 3D points $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$ representing the object model and their corresponding observed points $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_n\}$, the objective is to find the rotation matrix $\mathbf{R} \in SO(3)$ and translation vector $\mathbf{t} \in \mathbf{R}^3$ that minimize the alignment error as expressed in Equation (1). This optimization problem aims to

reduce the sum of squared differences between observed points and their transformed counterparts from the model:

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{n} \|\mathbf{q}_i - (\mathbf{R}\mathbf{p}_i + \mathbf{t})\|^2 \tag{2.21}$$

### 2.2.2 Methods and Approaches

Approaches for visual pose estimation of rigid bodies can be broadly categorized into feature-based methods, template matching, and learning-based methods.

Feature-based methods rely on detecting and matching keypoints between the object model and the observed data. This process involves identifying distinctive features, such as corners or edges, in both the model and the observation. Once detected, these features are matched to establish correspondences, and algorithms like the Perspective-n-Point (PnP) method [**?**] are used to compute the transformation that best aligns the matched features.

Template matching, on the other hand, involves comparing observed images with a database of pre-stored templates representing the object's appearance from various viewpoints. By finding the best match, the system can infer the object's pose with reasonable accuracy.

With the advent of machine learning, particularly deep learning, learning-based methods have gained prominence. These approaches involve training neural networks on large datasets of labeled images to learn the mapping from image space to pose parameters. Once trained, these networks can predict the pose of new images di-

rectly [?], offering a robust solution in scenarios with complex backgrounds or partial occlusions.

### 2.2.3 Rigid Body Pose Estimation Using Motive

Motive, developed by OptiTrack, enables precise tracking of rigid bodies using reflective markers and infrared cameras [?]. The process starts with rigid body configuration, where reflective markers are strategically placed on the object. These markers must form a unique, non-symmetrical pattern to prevent ambiguities in orientation detection. For example, symmetrical arrangements can cause confusion in differentiating rotations of 90°, leading to inaccurate pose estimation.

After marker placement, the next step is the rigid body definition within Motive. This process involves calibrating the motion capture system to ensure optimal accuracy, selecting the markers to define the rigid body, adjusting the pivot point to align with the object's geometric center, and saving the configuration for real-time applications. Calibration is critical as it corrects for lens distortions and aligns the coordinate system across multiple cameras, ensuring that the tracking data is accurate and reliable.

The data obtained from Motive integrates seamlessly with control systems for robotics and autonomous vehicles, enabling applications such as navigation, obstacle avoidance, and multi-robot coordination. However, challenges like marker occlusion, environmental conditions affecting infrared tracking, and the necessity for regular calibration to maintain system accuracy must be managed effectively.

In the case of QDrone 2, Motive is utilized to achieve accurate pose estimation by

leveraging a well-calibrated motion capture system. Reflective markers are strategically placed on the drone to form a unique, non-symmetrical pattern, ensuring reliable tracking and preventing orientation ambiguities. This configuration allows Motive to consistently identify the drone's position and orientation in real-time, even during dynamic maneuvers. Figure 2.2 illustrates sample marker arrangement used on QDrone 2.


Figure 2.2: Sample unique marker configurations on QDrone 2 used to avoid symmetrical ambiguities, ensuring accurate pose estimation.

## 2.3 QDrone 2 Control Architecture

QDrone 2 control system follows a cascaded Proportional-Integral-Velocity (PIV) control architecture to convert pose errors into motor commands (Figure 2.4). The control pipeline consists of:

1. Compute pose error from the reference pose and estimated pose.

2. Generate desired thrust, roll, pitch, and yaw rate using the PIV position controller.

3. Compute attitude error using the reference and estimated attitude (roll and pitch) angles.

4. Generate desired angular rate commands using the PD attitude controller.

5. Compute the final torque commands using a second-stage PD controller.

6. Map the final thrust and torques to motor commands via the motor mixing matrix.

### 2.3.1 Pose Controller (Outer Loop): Mapping Pose Error to Desired Attitude and Thrust

The estimated poses of the drone are determined using a visual localization camera system discussed in 2.2.3. In this system, the drone is modeled as a rigid body, which allows for more precise tracking of its position and orientation in three-dimensional space. By integrating data from multiple camera feeds, the system leverages sensor fusion techniques to enhance the accuracy and robustness of the pose estimation process. The system is capable of estimating both the three-coordinate translational position (along the $x$, $y$, and $z$ axes) and the yaw angle (rotation about $z$) of the drone.

The pose controller computes errors in the inertial frame:

$$\mathbf{e}_p = \mathbf{p}_{\text{ref}} - \mathbf{p}, \tag{2.22}$$

$$\mathbf{e}_v = \mathbf{v}_{\text{ref}} - \mathbf{v}, \tag{2.23}$$

$$\tag{2.24}$$

where:

- $\mathbf{e}_p = [e_x, e_y, e_z, e_\psi]^T$ is the pose error in $x, y, z$, and $\psi$,
- $\mathbf{p} = [x, y, z, \psi]^T$ and $\mathbf{p}_{\text{ref}}$ are the current and reference poses.
- $\mathbf{e}_v = [e_{\dot{x}}, e_{\dot{y}}, e_{\dot{z}}, e_{\dot{\psi}}]^T$ is the pose rate error in $\dot{x}, \dot{y}, \dot{z}$, and $\dot{\psi}$,
- $\mathbf{v} = [\dot{x}, \dot{y}, \dot{z}, \dot{\psi}]^T$ and $\mathbf{v}_{\text{ref}}$ are the current and reference poses.

A PIV controller generates the reference thrust & attitude commands:

$$T_{\text{ref}} = K_{p,z} e_{p,z} + K_{d,\dot{z}} e_{v,\dot{z}} + K_{i,z} \int e_{p,z} dt, \tag{2.25}$$

$$\phi_{\text{ref}} = -K_{p,y} e_{p,y} - K_{d,\dot{y}} e_{v,\dot{y}} - K_{i,y} \int e_{p,y} dt, \tag{2.26}$$

$$\theta_{\text{ref}} = K_{p,x} e_{p,x} + K_{d,\dot{x}} e_{v,\dot{x}} + K_{i,x} \int e_{p,x} dt, \tag{2.27}$$

$$\dot{\psi}_{\text{ref}} = K_{p,\psi} e_{p,\psi} + K_{d,\dot{\psi}} e_{v,\dot{\psi}} + K_{i,\psi} \int e_{p,\psi} dt. \tag{2.28}$$

Here, $T_{\text{ref}}$, $\phi_{\text{ref}}$, $\theta_{\text{ref}}$, and $\dot{\psi}_{\text{ref}}$ are the thrust, roll, pitch and rate of yaw commands

respectively. The commands could be written in the vector form as following:

$$[\theta_{\text{ref}}, -\phi_{\text{ref}}, T_{\text{ref}}, \dot{\psi}_{\text{ref}}]^T = \mathbf{K_p e_p} + \mathbf{K_d e_v} + \mathbf{K_i} \int \mathbf{e_p} dt \tag{2.29}$$

### 2.3.2 Attitude Controller (Inner Loop): Mapping Desired Attitude commands to Generalized Force and Torque Commands

Accurate attitude control is fundamental for maintaining the stability and maneuverability of UAVs and autonomous robots. This section delves into the estimation and control strategies that enable precise tracking of desired roll and pitch angles, ensuring robust performance in dynamic environments. The process begins with the conversion of raw Inertial Measurement Unit (IMU) data into meaningful attitude states, followed by a cascaded proportional-derivative (PD) controller to track the generate the system desired forced and torques, which will be later mapped to motor force commands,

#### IMU Data Processing for Attitude State Estimation

The conversion of IMU readings to a comprehensive set of nine attitude states—including roll, pitch, and yaw angles, their corresponding angular rates, and angular accelerations—is fundamental for robust state estimation. This process integrates data from gyroscopes and accelerometers to estimate the full attitude vector through sensor fusion techniques.

IMUs typically provide raw data such as angular velocity (in rad/s) from gyroscopes and linear acceleration (in $m/s^2$) from accelerometers. In the presented estimation framework (see Figure 2.3), this data undergoes several key processing stages to yield accurate attitude estimates.

The accelerometer data is first filtered and processed to approximate the roll and pitch angles. This is achieved using an accelerometer filter combined with an attitude approximation function, which calculates the roll and pitch based on the accelerometer's readings in the body frame. To ensure numerical stability, the accelerometer vector $\mathbf{a}_{body} = [a_x, a_y, a_z]^T$ is normalized as follows:

$$\mathbf{a}_{norm} = \begin{cases} \dfrac{\mathbf{a}_{body}}{\|\mathbf{a}_{body}\|_2}, & \text{if } \|\mathbf{a}_{body}\|_2 > \epsilon \\ [0, 0, 1]^T, & \text{otherwise} \end{cases} \tag{2.30}$$

Here, $\|\mathbf{a}_{body}\|_2$ represents the Euclidean norm of the accelerometer vector, and $\epsilon$ is a small threshold to prevent division by zero.

The roll angle ($\phi_{acc}$) and pitch angle ($\theta_{acc}$) are computed using:

$$\phi_{acc} = \arctan 2(a_y, a_z) \tag{2.31}$$

$$\theta_{acc} = \arctan 2\left(-a_x, a_y \sin(\phi_{acc}) + a_z \cos(\phi_{acc})\right) \tag{2.32}$$

The roll angle $\phi_{acc}$ is determined by the ratio of the lateral ($a_y$) to vertical ($a_z$) acceleration components. The pitch angle $\theta_{acc}$ is calculated considering both the longitudinal acceleration ($-a_x$) and the projection of $a_y$ and $a_z$ on the roll-adjusted

plane.

Concurrently, gyroscope data provides real-time measurements of angular velocities. This data is crucial for capturing dynamic orientation changes that accelerometers cannot accurately measure due to their sensitivity to linear motion and external forces. The gyroscope data is filtered to reduce noise, followed by a derivative operation to estimate angular accelerations.

The sensor fusion block integrates the filtered accelerometer and gyroscope data. This block utilizes complementary filtering techniques where accelerometer-based estimates, which are stable in the long term but noisy, are combined with gyroscope-based estimates, which are accurate in the short term but prone to drift. The fusion process balances these characteristics to produce reliable roll and pitch estimates.

The final attitude estimation is achieved through the combination of roll, pitch, and yaw angles (with yaw primarily derived from gyroscopic integration), their respective angular rates, and angular accelerations. These are represented as a nine-dimensional state vector:

$$\mathbf{x}_{att} = [\phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}, \ddot{\phi}, \ddot{\theta}, \ddot{\psi}]^T \tag{2.33}$$

This integrated estimation approach ensures high accuracy and robustness, making it suitable for real-time applications where precise attitude information is critical for control and navigation tasks. The redundancy in data sources and the fusion methodology mitigate individual sensor limitations, enhancing the reliability of attitude estimates.

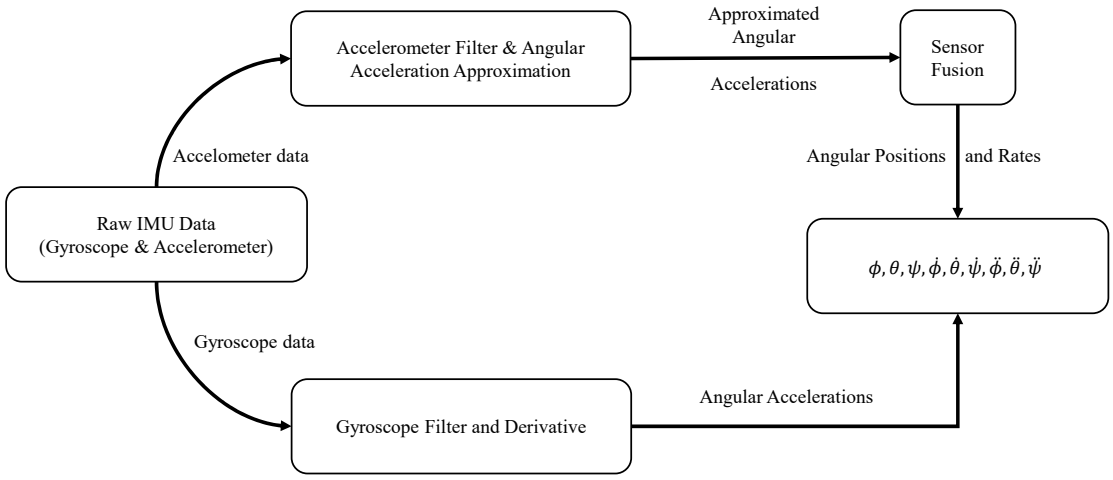



Figure 2.3: Estimation framework for converting IMU reading to the nine attitude states.

### Attitude Controller I: Mapping Desired Attitude commands to Desired Attitude Rate commands

The first attitude controller tracks the desired roll and pitch by generating the required angular rate commands. The error between the desired and current attitudes is defined as:

$$\mathbf{e}_\eta = \boldsymbol{\eta}_{\text{ref}} - \boldsymbol{\eta}, \tag{2.34}$$

$$\tag{2.35}$$

where $\boldsymbol{\eta} = [\phi, \theta]^T$ represents roll and pitch, and $\dot{\boldsymbol{\eta}} = [\dot{\phi}, \dot{\theta}]^T$ represents roll and pitch rates, all estimated as explained in section 2.3.2. The controller computes the roll and pitch rate commands using a proportional-derivative (PD) control strategy,

while the yaw rate command is taken from the pose controller output:

$$[\dot{\phi}_{\text{ref}}, \dot{\theta}_{\text{ref}}]^T = K_{p,\eta}\mathbf{e}_\eta + K_{d,\eta}\dot{\eta}, \tag{2.36}$$

$$\boldsymbol{\omega}_{\text{ref}} = [\dot{\phi}_{\text{ref}}, \dot{\theta}_{\text{ref}}, \dot{\psi}_{\text{ref}}]^T. \tag{2.37}$$

**Attitude Controller II: Mapping Desired Attitude Rate Commands to Generalized System Forces**

The second attitude controller computes torque commands based on angular rate errors to maintain the desired attitude.

$$\mathbf{e}_\omega = \boldsymbol{\omega}_{\text{ref}} - \boldsymbol{\omega}, \tag{2.38}$$

$$\tag{2.39}$$

where $\boldsymbol{\omega} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ represents roll, pitch and yaw rates, and $\boldsymbol{\alpha} = [\ddot{\phi}, \ddot{\theta}, \ddot{\psi}]^T$ represents roll, pitch and yaw accelerations, all estimated as explained in section 2.3.2.

$$\boldsymbol{\tau} = K_{p,\omega}\mathbf{e}_\omega + K_{d,\alpha}\alpha. \tag{2.40}$$

The thrust command is taken directly from the pose controller output. So, the final generalized thrust and torques are:

$$F = T_{\text{ref}}, \tag{2.41}$$

$$\tau_x = \tau_\phi, \quad \tau_y = \tau_\theta, \quad \tau_z = \tau_\psi. \tag{2.42}$$

### 2.3.3 Motor Forces Mapping

The generalized force & torque command values are mapped to motor forces using:

$$\begin{bmatrix} F_{m1} \\ F_{m2} \\ F_{m3} \\ F_{m4} \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & -\frac{1}{2L_{\text{roll}}} & \frac{1}{2L_{\text{pitch}}} & \frac{K_\tau}{4} \\ \frac{1}{4} & -\frac{1}{2L_{\text{roll}}} & -\frac{1}{2L_{\text{pitch}}} & -\frac{K_\tau}{4} \\ \frac{1}{4} & \frac{1}{2L_{\text{roll}}} & \frac{1}{2L_{\text{pitch}}} & -\frac{K_\tau}{4} \\ \frac{1}{4} & \frac{1}{2L_{\text{roll}}} & -\frac{1}{2L_{\text{pitch}}} & \frac{K_\tau}{4} \end{bmatrix} \begin{bmatrix} F \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix}. \tag{2.43}$$

where:

- $L_{roll}$ and $L_{pitch}$ are the roll and pitch arm lengths of the quadrotor,
- $K_\tau$ is the yaw torque coefficient,
- $F_{m1}, F_{m2}, F_{m3}, F_{m4}$ are the forces from motors 1 to 4.

The motor thrusts are then converted into duty cycle percentages using the thrust-to-voltage relationship:

$$V_i = \sqrt{\frac{F_{m,i}}{K_f}}. \tag{2.44}$$

clo

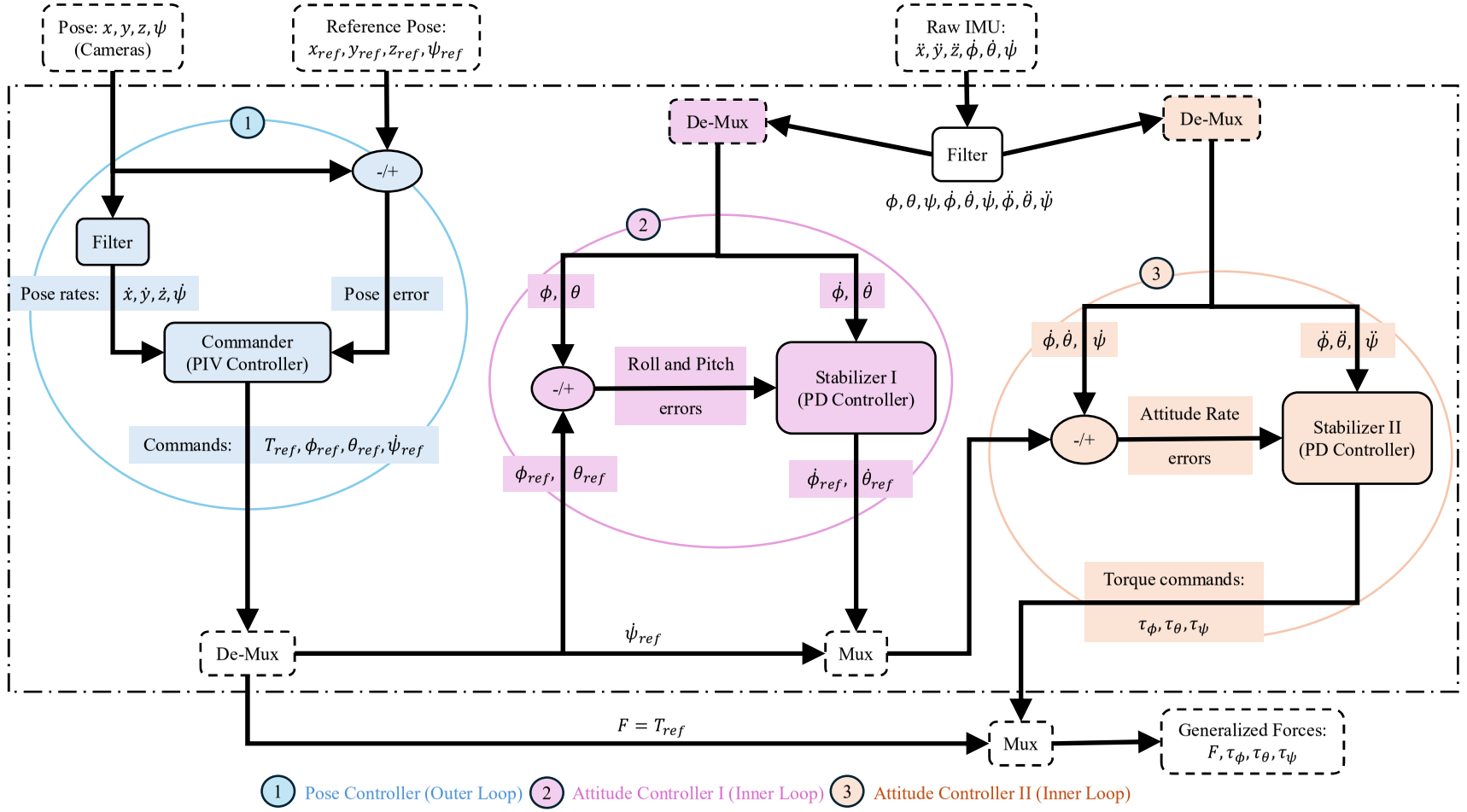



Figure 2.4: The Cascaded PIV Control Architecture Implemented by Quanser$^{TM}$. The commander pose controller processes pose and velocity errors, derived from camera-based localization, to generate thrust and attitude reference commands. The first attitude controller computes angular rate commands based on attitude errors and IMU-estimated angular rates. Finally, the second attitude controller produces generalized thrust and torque commands, which are mapped later to individual motor forces.

where $K_f$ is the thrust coefficient. This ensures the desired pose is achieved by dynamically adjusting the quadrotor's thrust and torques.

# Summary

This chapter provided an in-depth examination of the Quanser QDrone 2 platform, highlighting its dynamic modeling, pose estimation system, and control architecture.

## Quadcopter Dynamics

The chapter first explored the fundamental dynamics of quadcopters, focusing on the relationship between motor thrust, external forces, and UAV motion. The net forces acting on the quadcopter result from a balance between motor thrust and gravity, while rotational forces are controlled by adjusting the thrust distribution among the four motors. These interactions are mathematically modeled to describe how thrust variations influence UAV movement. A direct relationship between motor voltages and the corresponding thrust outputs is established, providing a complete framework for understanding quadcopter flight dynamics.

## Visual Pose Estimation

The pose estimation system used in QDrone 2 relies on a motion capture setup, which determines the UAV's position and yaw orientation. The OptiTrack motion capture system is used to achieve highly accurate real-time tracking.

## QDrone 2 Control Architecture

A cascaded Proportional-Integral-Velocity (PIV) control strategy is implemented in QDrone 2 for accurate flight stabilization. The control pipeline consists of:

- **Pose Controller (Outer Loop):** Computes pose errors and generates desired thrust and attitude commands.
- **Attitude Controller (Inner Loop):** Converts attitude errors into angular rate commands and determines the necessary thrust and torques for correction.
- **Motor Forces Mapping:** The final step, where control signals are converted into motor thrusts.

This chapter established a comprehensive understanding of the QDrone 2 system, including its dynamic equations, pose estimation process, and control mechanisms. These elements form the basis for the next phase of research, where fault injection and detection techniques will be explored to evaluate UAV reliability.

# CHAPTER 3

# PHYSICAL CHARACTERIZATION OF QUADCOPTER PROPELLER FAULTS

Propeller integrity is crucial for UAV stability and longevity, as structural inconsistencies can lead to performance degradation over time. This chapter investigates common propeller faults by categorizing them into distinct types and conducting physical experiments to measure and analyze their characteristics. The chapter first classifies propeller faults into unbalance and surface defects, describing their origins and potential impact on UAV operation. It then details the experimental investigation of these faults through propeller weighing and physical measurements to quantify structural inconsistencies. Additionally, the chapter introduces ISO 1940-1:2003, a standard for

rotor balancing, to establish a formal methodology for measuring and quantifying propeller unbalance. These investigations provide a baseline characterization of propeller defects, forming a reference for subsequent chapters that explore fault detection through flight dynamics.

## 3.1 Propeller Fault Types

Propeller faults significantly impact the performance, stability, and reliability of UAVs, making their detection and mitigation essential for safe and efficient operation. These faults can arise from structural damage, aerodynamic disturbances, or mass distribution irregularities, each affecting flight dynamics in different ways. Understanding their origins, effects, and the mechanisms through which they develop is crucial for designing robust diagnostic systems and effective maintenance strategies.

The classification of propeller faults in UAV operation is typically focused on three fault types: unbalance, edge cuts, and cracks which are widely regarded in the literature as the most important due to their pronounced impact on performance and reliability [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**]. This section provides a detailed analysis of each fault type, examining their causes and effects.

### 3.1.1 Unbalance

Unbalance in a propeller occurs when there is asymmetry in the mass distribution around its center of rotation, leading to radial forces and out-of-plane moment couples. This can result from material defects, uneven repairs, or damage to the blades. Propeller unbalance is commonly addressed through two methods: static and dynamic balancing. Static balancing corrects radial unbalance by repositioning the center of gravity to align with the axis of rotation. This is performed by placing the pro-

peller on a specialized fixture and making weight adjustments or material removal until it remains level without external forces [**?**]. Since this method is conducted while the propeller is stationary, it effectively mitigates mass-induced asymmetry but does not account for imbalances arising during rotation. Dynamic balancing, in contrast, accounts for both radial and axial unbalance by detecting and compensating for imbalance while the propeller is in motion. This is achieved by adding or subtracting counterweights in real-time to minimize vibrations caused by the combined effects of the propeller, hub, and motor assembly [**?**]. Unlike static balancing, this method ensures smooth operation by addressing aerodynamic disturbances and rotational asymmetries that emerge during flight. However, both methods primarily focus on mass distribution imbalances and may not fully resolve out-of-plane moment unbalances, which can introduce additional vibration sources and instability at high rotational speeds [**?**].

Unbalance can arise from uneven repairs, where inconsistent material removal or addition during maintenance creates mass asymmetry, or from material loss caused by erosion or unnoticed damage. Surface damage, such as material removal or erosion, further contributes to asymmetry in rotation. In small-scale UAVs, propeller unbalance has been shown to induce significant centrifugal forces during operation [**?**], leading to vibrations that can adversely affect onboard sensors and compromise flight stability and control systems.

Mathematically, propeller unbalance can be modeled by considering the mass difference ($\Delta m$) between the blades and the distance ($r$) from the axis of rotation. The

centrifugal force ($F_c$) generated due to this unbalance is expressed as:

$$F_c = \Delta m \cdot r \cdot \omega^2$$

where $\omega$ is the angular velocity of the propeller. This equation shows that even minor unbalances can generate substantial forces at high rotational speeds, resulting in vibrations that may interfere with sensor accuracy and degrade overall UAV performance.

Detecting propeller unbalance faults is challenging. The On-Rotor Sensing (ORS) monitoring method addresses this by using a cost-efficient MEMS accelerometer mounted at the center of the motor-propeller system to directly capture vibration signals for analysis [**?**]. Experimental results show that it can detect minor unbalance faults as low as 0.25% of the propeller weight, making it an economical and robust solution for large industrial drones with potential for detecting other early-stage faults [**?**].

### 3.1.2 Edge Cuts

Edge cuts refer to damage or deformations along the leading or trailing edges of UAV propeller blades. These faults are particularly prevalent due to the high aerodynamic forces concentrated in these regions, combined with their direct exposure to environmental hazards. The leading edge, which is the first point of contact with incoming airflow and debris, is especially vulnerable to impacts that can cause nicks, chips, or other structural compromises [**?**]. While the trailing edge is less exposed, it is still

susceptible to damage from the turbulent airflow it encounters during operation. Over time, even minor flaws on either edge can disrupt the smooth flow of air, leading to flow separation, increased drag, and reduced thrust efficiency.

Damage to the trailing edge exacerbates turbulence and vortex shedding, further degrading overall propeller performance and increasing energy consumption. As noted in [**?**], metal tipping is often applied along the leading edge and the tips of blades to protect against damage from airborne particles encountered during flight stages (Figure 3.1). This solution highlights the critical vulnerability of leading edges to edge cuts, which can significantly compromise aerodynamic performance. By shielding these high-risk areas, metal tipping helps mitigate the risks associated with edge cuts, ensuring better durability and consistent propeller efficiency.

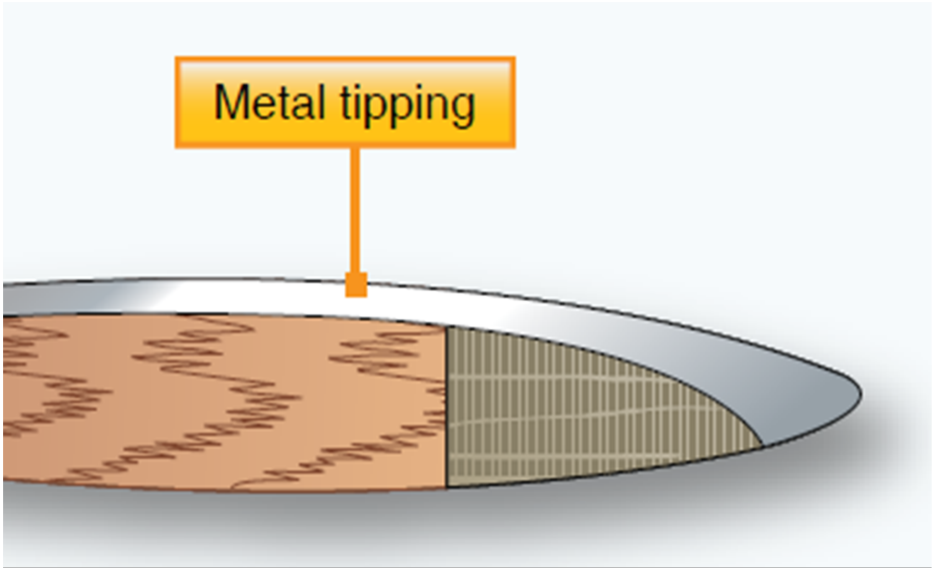



Figure 3.1: Metal tipping for edge protection [**?**].

### 3.1.3 Cracks

Cracks in UAV propellers compromise aerodynamic performance and structural integrity, posing risks to efficiency, stability, and safety. These defects arise from fatigue, impact damage, material inconsistencies, or environmental factors. Their presence disrupts airflow, induces vibrations, and accelerates failure due to cyclic stresses. The presence of cracks in the propeller increases the likelihood of crack propagation, leading to further deterioration of the blade structure [**?**], [**?**].

Generally, blade cracks significantly alter aerodynamic performance by disturbing the airflow and modifying pressure distributions. Under aerodynamic and centrifugal loads, cracks propagate differently; aerodynamic loads drive cracks towards the trailing edge, whereas centrifugal loads direct them towards the leading edge [**?**]. This directional propagation affects lift production and introduces localized flow separations, leading to efficiency losses. Furthermore, cracks on the trailing edge generate tonal noise [**?**].

Cracks can be classified based on their direction relative to the blade structure, each with distinct impacts on performance (Figure 3.2). Transverse cracks, which form perpendicular to the blade span, typically result from fatigue or impact and significantly alter the blade's stiffness distribution, leading to changes in natural frequencies and mode shapes [**?**]. The reduction in stiffness due to a transverse crack causes localized flexibility, which in turn amplifies vibratory responses and increases the likelihood of resonance [**?**]. Longitudinal cracks, aligned with the blade span, arise from tensile loads and progressive fatigue, leading to rigidity loss, pitch deviations,

and increased vibration levels [?].

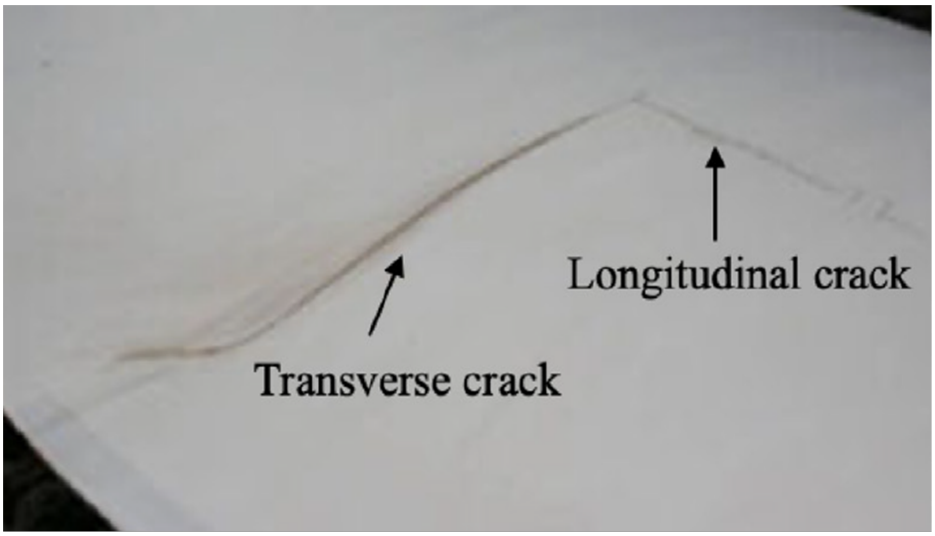



Figure 3.2: Crack types on propeller blades (Transverse & Longitudinal) [?].

Vibrations in propellers, particularly those caused by unbalances, play a significant role in the formation and propagation of transverse cracks. When a propeller is unbalanced, it generates excessive cyclic loads due to periodic forces acting on the structure at high rotational speeds (Figure 3.3). These forces cause oscillations that induce alternating stress concentrations at specific locations, particularly near the hub, the blade root, and regions of variable cross-section [?]. As rotational speed increases, the vibration amplitude follows a linear dependency, and beyond a critical limit, these vibrations can exceed material fatigue thresholds, initiating micro-cracks that grow with continuous operation.

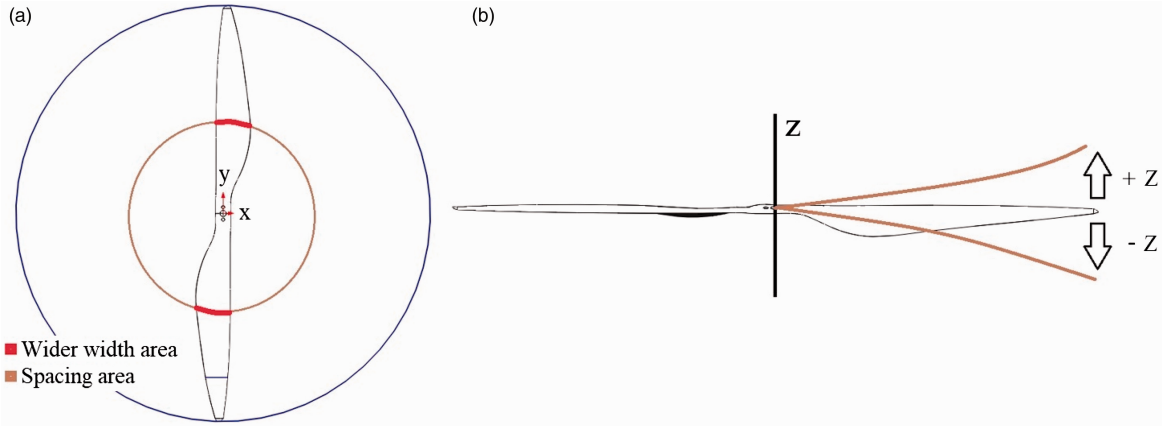
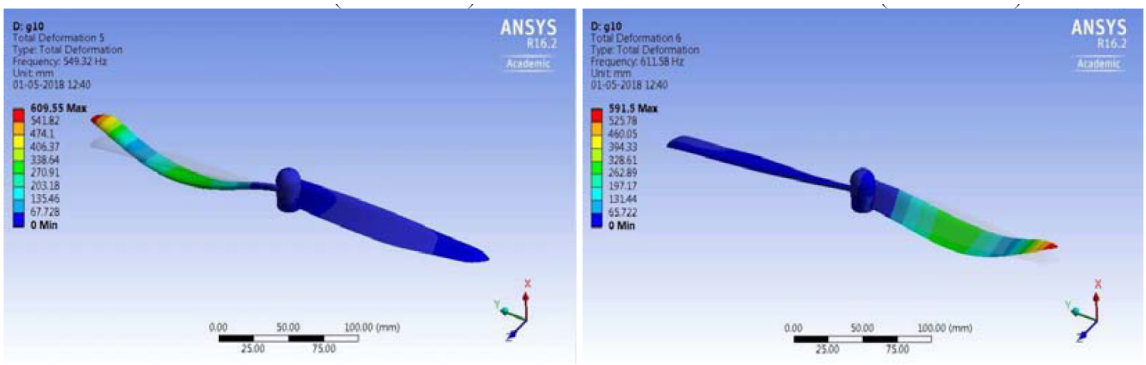



Figure 3.3: Propeller vibration induced by rotational motion. The top-down view (a) highlights regions of varying width and spacing that contribute to asymmetric aerodynamic forces, which in turn generate oscillatory motion. The side view (b) demonstrates the displacement along the $z$-axis, showing how the unbalance causes alternating movement in the positive and negative directions. These vibrations, if excessive, can lead to structural fatigue and crack formation [**?**].

The transverse cracks typically develop due to fatigue, as cyclic loading leads to material degradation over time. The regions experiencing the highest vibrational amplitudes, such as the blade tips [**?**] and areas with abrupt geometric changes, are particularly vulnerable (Figure 3.4). As the crack propagates, it further exacerbates the unbalance, leading to an unstable feedback loop where increasing vibration accelerates structural failure [**?**]. Additionally, the presence of aerodynamic forces contributes to asymmetric stress distributions, further amplifying crack growth.

The deformation of a propeller under vibrational loading illustrates how high-frequency oscillations induce structural stress and potential failure. Figure 3.4 highlights the vibrational response of a rotating propeller, where deformation due to dynamic excitation can lead to fatigue accumulation over time. As these vibrations persist, localized stress concentrations promote crack initiation and propagation.




Mode 5(549.32 HZ) Mode 6(611.58 HZ)

Figure 3.4: Finite Element Analysis (FEA) illustrating propeller deformation under vibrational loading. High-frequency oscillations contribute to stress accumulation, leading to crack formation and propagation [**?**].

To mitigate these risks, vibration monitoring, such as the laser vibrometer approach used in [**?**], helps in early fault detection before critical failure occurs. Proper balancing and material selection, such as the use of high-rigidity carbon fiber, enhance the durability of the propeller against vibrational fatigue. Understanding and controlling vibration-induced stress is essential for ensuring the longevity and reliability of UAV propellers.

## 3.2 Experimental Analysis of Propeller Faults

All faults often induce a combination of aerodynamic and mass effects [**?**], which together lead to performance degradation. For example, edge cuts or cracks affect the aerodynamic flow, causing increased drag and reduced lift. Simultaneously, structural damage can lead to unbalance, resulting in vibrations that compromise stability and control. The interaction between these effects intensifies the overall impact. Cracks, for instance, not only disrupt pressure distribution but also lead to uneven

mass distribution, exacerbating the unbalance. Similarly, edge cuts influence aerodynamic efficiency and contribute to mass asymmetry, amplifying the consequences of vibration-induced instability.

Since many faults simultaneously affect both aerodynamics and mass distribution, unbalance serves as a quantifiable and widely recognized indicator of structural integrity. Even minor deviations in mass symmetry can generate significant vibratory responses, making balance quality assessment a reliable method for estimating fault severity. By evaluating the extent of unbalance and its effect on system dynamics, engineers can systematically diagnose faults, anticipate performance degradation, and implement corrective actions to enhance stability and operational reliability.

### 3.2.1 Balance Quality Requirements for Rotors

To systematically quantify and control unbalance, the ISO 1940-1:2003 [**?**] standard establishes balance quality requirements for rigid rotors across various industrial applications.

This standard defines permissible residual unbalance levels based on rotor mass, rotational speed, and operational constraints. It classifies balance quality grades, ranging from G4000 for heavy-duty applications such as marine engines to G0.4 for high-precision components like gyroscopes. Additionally, ISO 1940-1:2003 provides standardized measurement techniques, verification methods, and guidelines for correcting unbalance to ensure compliance with industry-specific tolerances.

By adhering to these specifications, engineers can systematically assess and mitigate unbalance effects, optimizing rotor performance, extending service life, and re-

ducing maintenance costs.

### Residual Unbalance

Residual unbalance is the amount of unbalance that remains in a rotor after the balancing process. It is quantified as the product of the unbalanced mass $m_u$ and its eccentricity $r_u$, typically measured in gram-millimeters ($\mathrm{g} \cdot \mathrm{mm}$). Mathematically, it is defined as:

$$U_{\mathrm{res}} = m_u \cdot r_u \tag{3.1}$$

where:

- $U_{\mathrm{res}}$ is the residual unbalance ($\mathrm{g} \cdot \mathrm{mm}$),
- $m_u$ is the unbalanced mass (g),
- $r_u$ is the radial distance of the unbalanced mass from the rotational axis (mm).

According to ISO 1940-1:2003, unbalance can be represented as a single resultant vector or decomposed into components in different correction planes for practical balancing.

### Permissible Residual Unbalance

For satisfactory rotor performance, the residual unbalance should not exceed a permissible limit. The permissible residual unbalance $U_{\mathrm{per}}$ is derived based on the balance quality grade $G$, rotor mass $m$, and operational angular velocity $\Omega$, as defined by:

$$U_{\text{per}} = \frac{G \cdot m}{\Omega} \tag{3.2}$$

where:

- $G$ is the balance quality grade (mm/s),
- $m$ is the rotor mass (kg),
- $\Omega$ is the angular velocity (rad/s).

Alternatively, the permissible residual specific unbalance per unit mass $e_{\text{per}}$ can be expressed as:

$$e_{\text{per}} = \frac{U_{\text{per}}}{m} = 9594 \times \frac{G}{\Omega} \tag{3.3}$$

where $e_{\text{per}}$ is typically measured in micrometers ($\mu$m) or (g.mm\kg). This specific unbalance provides a direct measure of how much unbalance per unit mass is acceptable in a system.

Figure 3.5 illustrates the permissible residual specific unbalance as a function of the balance quality grade $G$ and service speed $n$, as defined in ISO 1940-1:2003. The white area represents the commonly used range based on industry experience, providing a guideline for selecting appropriate balance quality levels for different rotor applications. This chart serves as a crucial reference for determining acceptable unbalance thresholds to ensure safe and efficient rotor performance.

**Balance Quality Grades ($G$)**

The balance quality grade $G$ is a standardized parameter that defines the allowable vibration severity based on rotor application. ISO 1940-1:2003 categorizes rotors into different grades depending on their function, operational speed, and acceptable vibration levels. The grades range from G4000 (extremely high unbalance tolerance, such as slow marine diesel engines) to G0.4 (high-precision applications like gyroscopes). Typical applications of balance quality grades include:

For most industrial rotors, balance quality grades between G6.3 and G2.5 (White region in figure 3.5 chart) are commonly used, ensuring optimal operational efficiency while minimizing vibration-induced wear and fatigue.

### 3.2.2 Experimental Classification of Propeller Faults

The propellers used in this study are HQ Durable Prop 7x4.5 Light Grey propellers [**?**], designed for high-performance drone applications. These propellers are made of durable polycarbonate material, with a diameter of 7 inches and a pitch of 4.5 inches. They are the standard propellers used by the Quanser QDrone 2. Figure 3.6 illustrates the HQ Durable Prop 7x4.5 propeller used as the reference model for fault analysis in this study.

The propeller faults analyzed in this study are categorized into three groups: Edge Cut Faults, Crack Faults, and Surface Unbalance Faults. Each group is divided into three severity levels: low, medium, and high, based on their impact on the aerodynamic and structural performance of the propeller. Figure 3.7 provides a visual

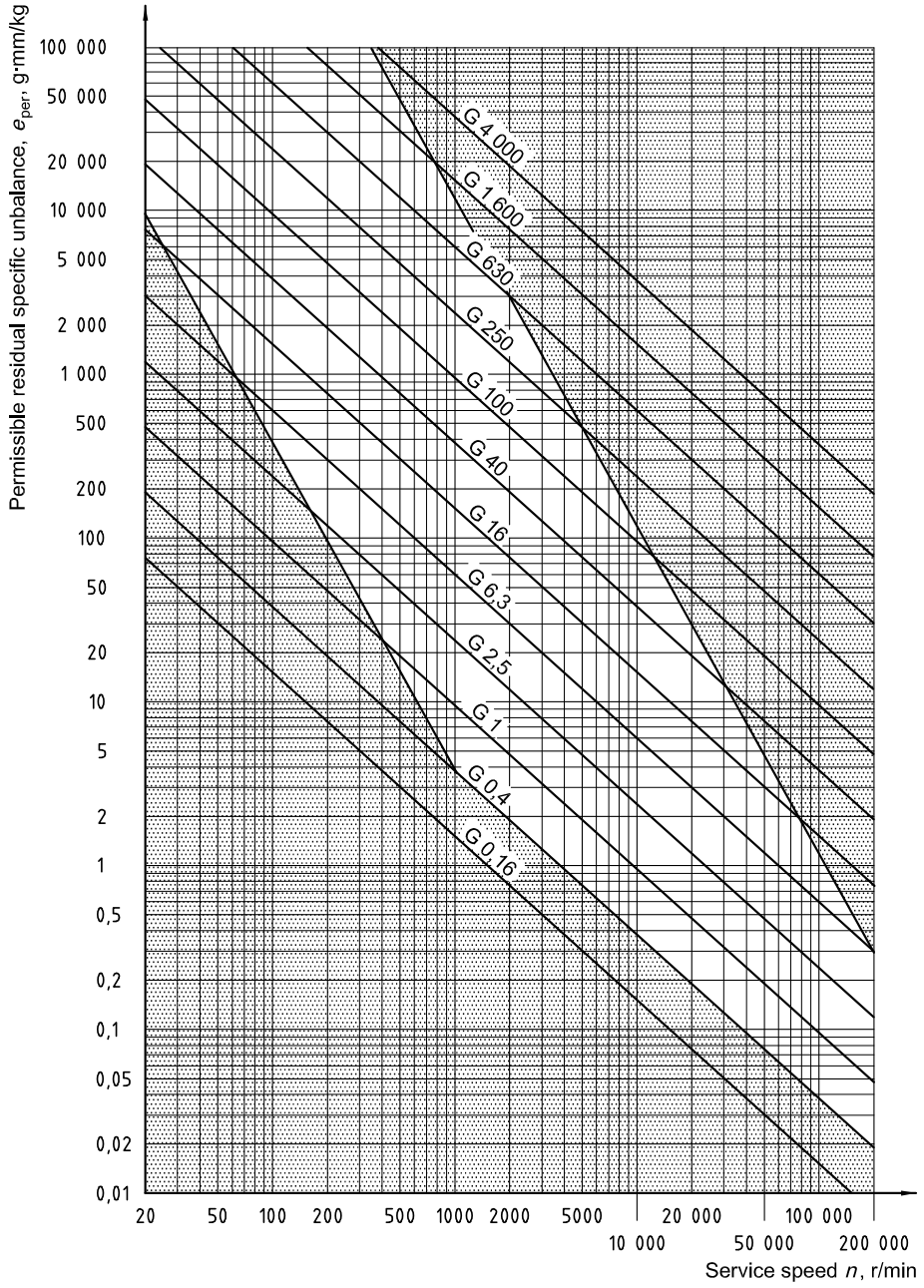



Figure 3.5: Permissible residual specific unbalance as a function of balance quality grade $G$ and service speed $n$, adapted from ISO 1940-1:2003. The white area indicates the commonly used range based on industry standards.

Figure 3.6: HQ Durable Prop 7x4.5 Light Grey propeller used in this study as the reference model for fault analysis. These propellers are standard for the Quanser QDrone 2.

representation of the different fault types. These faults were intentionally introduced while the drone was in normal operation to observe their effects across various flight phases [**?**].

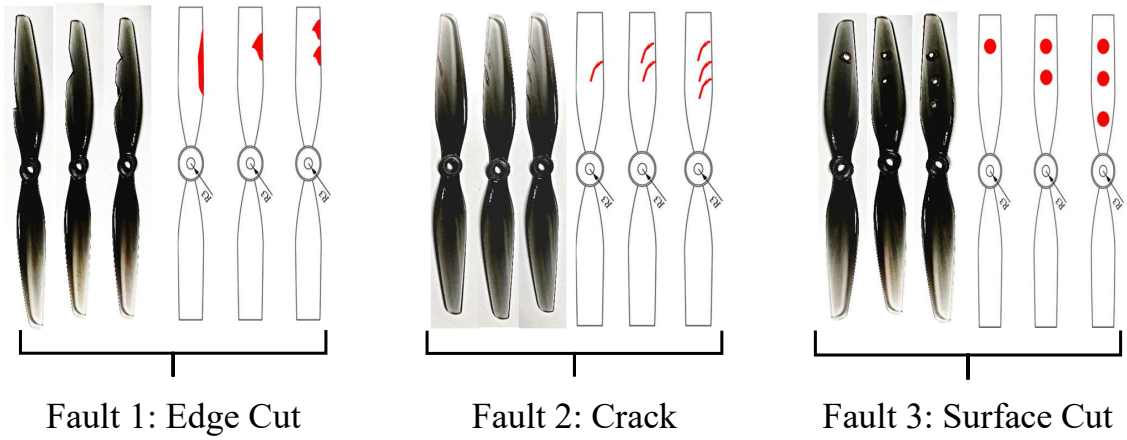



Figure 3.7: Photographs of the three fault groups for the drone propeller.

**Edge Cut Faults** Edge cut faults are caused by physical removal of material along the blade edges, affecting the balance and aerodynamics of the propeller. Increasing fault severity corresponds to larger material removal, which disrupts airflow and increases vibration. Three levels of severity were introduced: a 2-mm single edge cut, a 5-mm single edge cut, and two separate 5-mm edge cuts.

**Crack Faults** Crack faults occur due to fractures along the propeller blade, weakening its structural integrity and causing potential failure during high-speed rotations. To replicate different levels of severity, one, two, and three inclined cracks were created, each measuring 10 mm in length.

**Surface Unbalance Faults** These faults are introduced by drilling holes into the propeller surface, creating an asymmetry in mass distribution. Such defects significantly increase rotational imbalance and contribute to instability. The severity levels were defined by drilling one, two, and three holes, each with a 6-mm diameter, into a single blade.

Table 3.1 summarizes the fault classification, severity levels, and the corresponding physical modifications made to the propellers.

### Fault Residual Unbalance Calculations

The classification of fault types based on physical damage to the propeller provides a structured approach to understanding their impact on performance. However, to quantify the effects of these faults, it is necessary to assess the imbalance they introduce into the system. Unbalance in rotating systems is a crucial factor that affects

Table 3.1: Fault severity levels and corresponding sizes for all nine faults

| **Fault Type** | **Fault Code** | **Severity Level** | **Severity Code** | **Fault Size / Configuration** |
|---|---|---|---|---|
| Edge Cut | F1 | Low | SV1 | Single edge cut<br>2 mm removed |
| | | Medium | SV2 | Single edge cut<br>5 mm removed |
| | | High | SV3 | Two edge cuts<br>5 mm removed each |
| Crack | F2 | Low | SV1 | Single inclined crack<br>10 mm length |
| | | Medium | SV2 | Two inclined cracks<br>10 mm length each |
| | | High | SV3 | Three inclined cracks<br>10 mm length each |
| Surface Unbalance | F3 | Low | SV1 | Single hole<br>6 mm diameter |
| | | Medium | SV2 | Two holes<br>4 mm diameter each |
| | | High | SV3 | Three holes<br>4 mm diameter each |

stability, efficiency, and longevity. To evaluate the severity of each fault, the mass difference and its radial distance from the axis of rotation are measured, allowing the computation of the resultant unbalance.

The unbalance measurements are based on the resultant unbalance equation:

$$U_{\text{res}} = \sum_{i=1}^{n} m_i \cdot r_i \tag{3.4}$$

where $U_{\text{res}}$ represents the resultant unbalance, $m_i$ is the mass difference at a specific fault location, and $r_i$ is the corresponding radial distance from the rotation axis. This equation provides a quantitative measure of the imbalance introduced by each fault, helping to determine its severity in terms of dynamic instability.

A Hobby Fans Tru-Spin Propeller Balancer (Figure 3.8) was employed to ensure that the healthy propeller was properly balanced before conducting measurements. This device provides a low-friction pivot mechanism, allowing the propeller to rotate freely and settle in a position dictated by any residual unbalance. A perfectly balanced propeller remains stationary at any orientation (Figure 3.9), whereas an unbalanced propeller tilts toward the heavier side due to gravitational effects (Figure 3.10). Any imbalance detected was corrected by adding small amounts of material to the lighter side. This step is crucial in ensuring that the mass discrepancy observed in faulty propellers was solely attributed to the induced fault rather than pre-existing manufacturing imbalances.

A digital carat milligram scale balance (Figure 3.11) with a measurement range of 30 grams and an accuracy of 1 milligram was used to measure the mass of both

Figure 3.8: The Hobby Fans Tru-Spin Propeller Balancer used to assess and correct propeller balance. The low-friction pivot system allows precise detection of mass asymmetry.

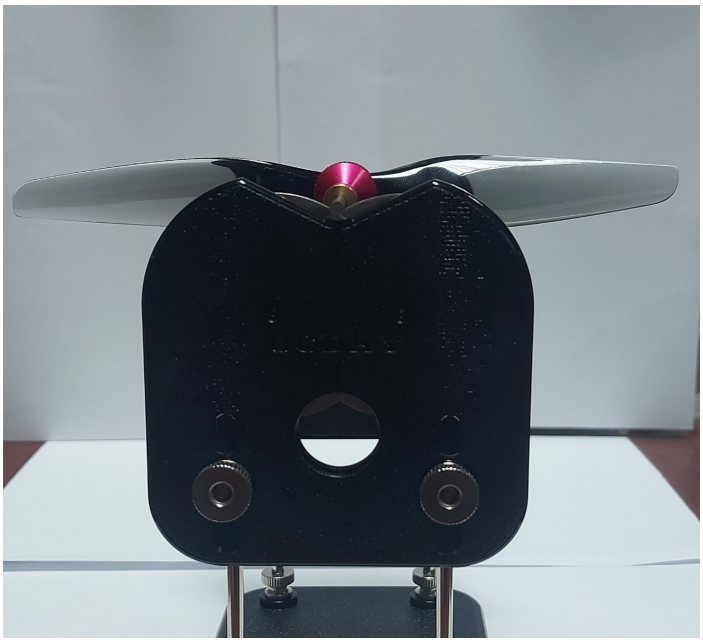


Figure 3.9: A properly balanced propeller remains stationary at any orientation on the Hobby Fans Tru-Spin propeller balancer, indicating minimal residual mass asymmetry.

Figure 3.10: An unbalanced propeller tilts towards the heavier side when placed on the Hobby Fans Tru-Spin propeller balancer, revealing mass asymmetry that must be corrected.


Figure 3.11: Digital carat milligram scale balance.

healthy and faulty propellers. Given the resolution of the balance, any recorded weight value carries an inherent uncertainty of ±0.5 milligrams. The difference in mass between the healthy and faulty propellers represents the material removed due to fault introduction. This mass difference directly corresponds to the unbalance introduced in the system. By determining the center of the removed mass, the resulting unbalance moment can be calculated, which is essential for quantifying the fault-induced asymmetry and its effect on the rotor dynamics.

Table 3.2 presents the fault severity levels and the corresponding measured unbalance values for all nine fault conditions. The relationship between increasing fault severity and resultant unbalance is evident, as greater material removal or deformation leads to higher mass differences and larger radial distances. This, in turn, results in greater unbalance forces, which can affect the operational stability of the drone.

Table 3.2: Fault severity levels and measured unbalance for different fault types

| **Fault Code** | **Severity Code** | **Fault Description** | **Mass (g)** | **Mass Difference (mg)** | **Distance (mm)** | **Measured Unbalance (g · mm)** |
|---|---|---|---|---|---|---|
| Healthy | - | No Fault | 7.198 | 0 | 0 | 0 |
| | - | No Fault | 7.197 | 0 | 0 | 0 |
| | - | No Fault | 7.198 | 0 | 0 | 0 |
| F1 | SV1 | Edge Cut (1 2-mm cut) | 7.163 | 35 | 40 | 0.035×40 = 1.400 |
| | SV2 | Edge Cut (1 5-mm cut) | 7.113 | 85 | 60 | 0.085×60 = 5.1000 |
| | SV3 | Edge Cut (2 5-mm cuts) | 7.102 | 96 | 65/40 | (0.048×65) + (0.048×50) = 5.520 |
| F2 | SV1 | Crack Fault (1 crack) | 7.194 | 4 | 55 | 0.004×55 = 0.220 |
| | SV2 | Crack Fault (2 cracks) | 7.148 | 50 | 40/70 | (0.025×40) + (0.025×70) = 2.750 |
| | SV3 | Crack Fault (3 cracks) | 7.145 | 53 | 70/55/40 | (0.018×70) + (0.018×55) + (0.018×40) = 2.640 |
| F3 | SV1 | Unbalance (1 hole) | 7.140 | 58 | 65 | 0.058×65 = 3.770 |
| | SV2 | Unbalance (2 holes) | 7.131 | 67 | 65/50 | (0.034×65) + (0.034×50) = 3.91 |
| | SV3 | Unbalance (3 holes) | 7.099 | 99 | 65/50/35 | (0.033×65) + (0.033×50) + (0.033×35) = 4.950 |

**The Permissible Residual Unbalance of the Rotor System**

For the propeller and rotor system in this study, the permissible residual unbalance is calculated using the balance quality grade $G$, rotor mass $m$, and operational angular velocity $\Omega$, in accordance with ISO 1940-1:2003. The mass of the rotor system is approximately 30 g (0.03 kg). The average angular velocity of the rotors for QDrone 2, operating without additional payload, was determined based on the specified motion speed discussed in [**?**]. At this speed, the measured average rotor speed was found to be approximately 10,000 rpm.

Using a balance quality grade of G 6.3, the permissible residual specific unbalance is obtained from the chart in Figure 3.5:

$$e_{\text{per}} = 6\,\text{g} \cdot \text{mm/kg} \tag{3.5}$$

Thus, the permissible residual unbalance is calculated as:

$$U_{\text{per}} = e_{\text{per}} \times m = 6 \times 0.3 = 0.18\,\text{g} \cdot \text{mm} \tag{3.6}$$

This result indicates that all fault types, across all severity levels, introduce unbalance values that exceed the system's permissible limits. Consequently, these unbalances cannot be effectively absorbed or neglected, leading to potential instability and increased vibration within the system.

### Fault Classification and Surface Area / Unbalance Distribution

After determining that the measured unbalance values exceed the permissible unbalance limits, further investigation is necessary to analyze how different fault types contribute to both unbalance magnitude and affected surface area. Since unbalance alone does not provide a complete picture of the severity of each fault, examining the correlation between unbalance and the physical extent of damage is crucial. Larger surface areas may indicate significant structural degradation, while high unbalance values suggest pronounced dynamic instability. Understanding this relationship helps in assessing the potential impact of each fault type on propeller performance and overall flight stability.

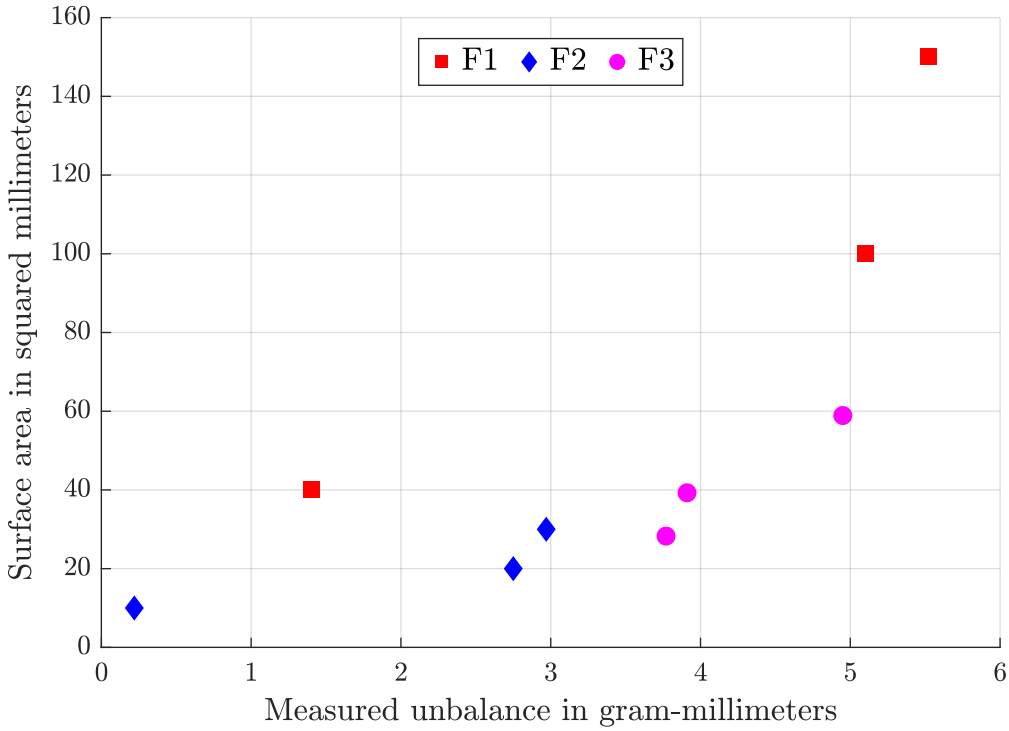



Figure 3.12: Distribution of Fault Types Based on Measured Unbalance and Surface Area

Figure 3.12 illustrates the distribution of different fault types with respect to their measured unbalance ($g \cdot mm$) and affected surface area ($mm^2$). The plot shows the three fault groups groups: F1 (Edge Cuts), F2 (Cracks), and F3 (Surface Unbalance), each represented by different markers.

Overall, faults exhibit a clear pattern in how they affect both unbalance and surface area, forming distinct categories based on severity. Edge cut faults (F1) demonstrate the largest affected surface areas, with severity increasing from moderate surface and medium unbalance (F1\SV1) to large surface and high unbalance (F1\SV3). This indicates that as more material is removed from the blade edges, aerodynamic disturbances and dynamic instability rise significantly.

Crack faults (F2), on the other hand, consistently show small surface areas across all severity levels but vary in unbalance, ranging from low (F2\SV1) to medium (F2\SV3). This suggests that cracks primarily weaken structural integrity without significantly affecting mass distribution, making them less impactful on unbalance compared to edge cuts and drilled holes.

Surface unbalance faults (F3), caused by drilled holes, are characterized by small to moderate surface areas but introduce high unbalance values across all severity levels. The small surface, high unbalance classification of F3\SV1 demonstrates that even a small mass removal at a localized position can induce substantial rotational asymmetry. As severity increases, the faults transition to moderate surface, high unbalance (F3\SV3), reinforcing the idea that concentrated mass loss plays a dominant role in unbalance severity.

Table 5.2 provides a detailed classification of fault types, illustrating how edge

cuts primarily impact surface area and unbalance simultaneously, while cracks mainly affect structural integrity with minor unbalance variations, and drilled holes result in high unbalance even with moderate surface areas.

Table 3.3: Summary of fault groups with measured unbalance, surface area, and category

| **Fault Type** | **Fault Code** | **Surface Area ($mm^2$)** | **Measured Unbalance ($g \cdot mm$)** | **Category** |
|---|---|---|---|---|
| Edge Cuts | F1\SV1 | Moderate (40 $mm^2$) | Medium (1.5 $g \cdot mm$) | Moderate Surface Medium Unbalance |
| | F1\SV2 | Large (100 $mm^2$) | High (5.0 $g \cdot mm$) | Large Surface High Unbalance |
| | F1\SV3 | Large (150 $mm^2$) | High (6.0 $g \cdot mm$) | Large Surface High Unbalance |
| Cracks | F2\SV1 | Small (10 $mm^2$) | Low (0.5 $g \cdot mm$) | Small Surface Low Unbalance |
| | F2\SV2 | Small (20 $mm^2$) | Medium (2.5 $g \cdot mm$) | Small Surface Medium Unbalance |
| | F2\SV3 | Small (30 $mm^2$) | Medium (3.0 $g \cdot mm$) | Small Surface Medium Unbalance |
| Surface Unbalances | F3\SV1 | Small (30 $mm^2$) | High (4.0 $g \cdot mm$) | Small Surface High Unbalance |
| | F3\SV2 | Moderate (50 $mm^2$) | High (5.0 $g \cdot mm$) | Moderate Surface High Unbalance |
| | F3\SV3 | Moderate (70 $mm^2$) | High (5.5 $g \cdot mm$) | Moderate Surface High Unbalance |

### 3.2.3 Unbalance Estimation Using the Flight Stand

Accurate estimation of balance is crucial for diagnosing and mitigating unbalance-related issues in a UAV system. A flight stand provides a controlled environment to assess the effects of mass distribution and rotational asymmetry, allowing for systematic analysis of imbalance and its impact on dynamic response.

The Tytorobotics Flight Stand 15 [**?**] is a specialized test platform designed for precise measurement and evaluation of UAV propulsion systems, particularly for assessing balance, thrust, torque, and efficiency in controlled conditions. By securely mounting the UAV or its propulsion unit, the stand enables real-time data acquisition, allowing identification of imbalances, measurement of power consumption, and recording the thrust generation profile. As shown in Figure 3.13, the system integrates high-resolution sensors to capture thrust variations and dynamic loads, which are crucial for diagnosing unbalanced rotors or structural misalignment.

The balance estimation process on the flight stand involves measuring vibration levels and rotational accelerations under controlled conditions. According to industry best practices [**?**], diagnosing imbalance requires detecting excessive vibration amplitudes, which correlate directly with unbalance-induced forces. The following steps outline the process:

1. **Initial Baseline Measurement**: The UAV is mounted on the flight stand, and initial data on rotational stability, motor performance, and vibration levels is collected. This serves as the baseline reference.

Figure 3.13: Tytorobotics Flight Stand 15, a precision measurement system for UAV propulsion testing [**?**].

2. **Controlled Excitation**: By operating the UAV at different speeds and recording IMU data, the response to induced loads is analyzed. Higher rotational speeds amplify unbalance effects, making detection easier.

3. **Spectral Analysis**: Using frequency domain techniques such as Fast Fourier Transform (FFT), dominant vibration frequencies are identified. A clear peak at the rotational frequency suggests mass imbalance.

Acceleration measurements were evaluated across different axes ($x$, $y$, and $z$) to determine the most effective direction for detecting imbalance. The $z$-axis, being out of the plane of rotation, exhibited minimal sensitivity to unbalance forces, as the primary effects of mass asymmetry introduce in the plane of rotation. The $x$-axis, while within the plane, showed lower vibration amplitudes, where force distribution remains relatively symmetrical. In contrast, the $y$-axis, being the radial direction, captured

the strongest response to imbalance-induced forces, as unbalance primarily generates radial oscillations. This made the $y$-axis the most suitable choice for analyzing unbalance effects, providing a clear and consistent correlation between acceleration peaks and estimated unbalance values.

Figure 3.14 presents the frequency response of the system, illustrating acceleration in the $y$-direction as a function of frequency for different fault conditions (F1, F2, and F3) and severity levels (SV1, SV2, and SV3). The dominant peak around 83.3 Hz highlights the primary resonance frequency of the system, where unbalance effects become most pronounced. Variations in peak amplitude across fault conditions suggest that imbalance significantly influences the system's dynamic response. Notably, F1 and F3 fault cases exhibit higher acceleration peaks compared to the healthy baseline conditions (H1, H2, H3), indicating a stronger correlation with unbalance-related forces.

To establish a quantitative relationship between acceleration peaks and measured unbalance, a linear scaling factor was determined through least squares fitting. This factor allows for the conversion of acceleration values ($m/s^2$) into unbalance ($g \cdot mm$).

Figure 3.15 illustrates the estimated unbalance in gram-millimeters ($g \cdot mm$) across different fault conditions and severity levels. The trend confirms that increasing fault severity generally leads to higher unbalance magnitudes. The blue bars (F1) represent the highest unbalance values, which align with the pronounced acceleration peaks observed in Figure 3.14. Meanwhile, the F2 faults exhibit lower unbalance levels, correlating with the relatively smaller acceleration peaks. The F3 cases show varied unbalance levels, with F3/SV2 reaching the highest recorded unbalance.

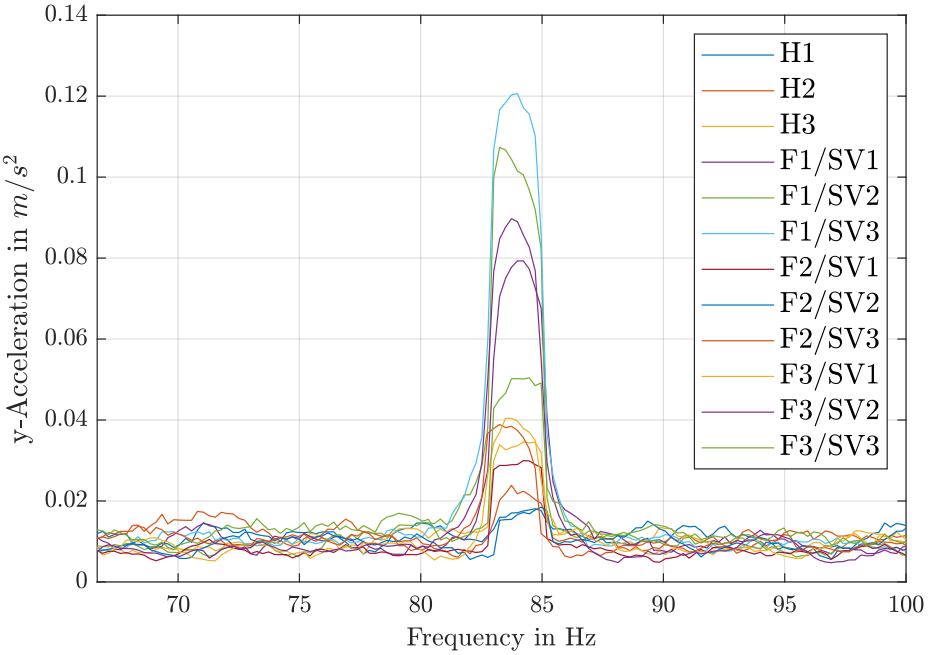



Figure 3.14: Frequency response analysis showing acceleration in the $y$-direction as a function of frequency for different fault conditions and severity levels. The peak at 83.3 Hz indicates the primary resonance frequency.

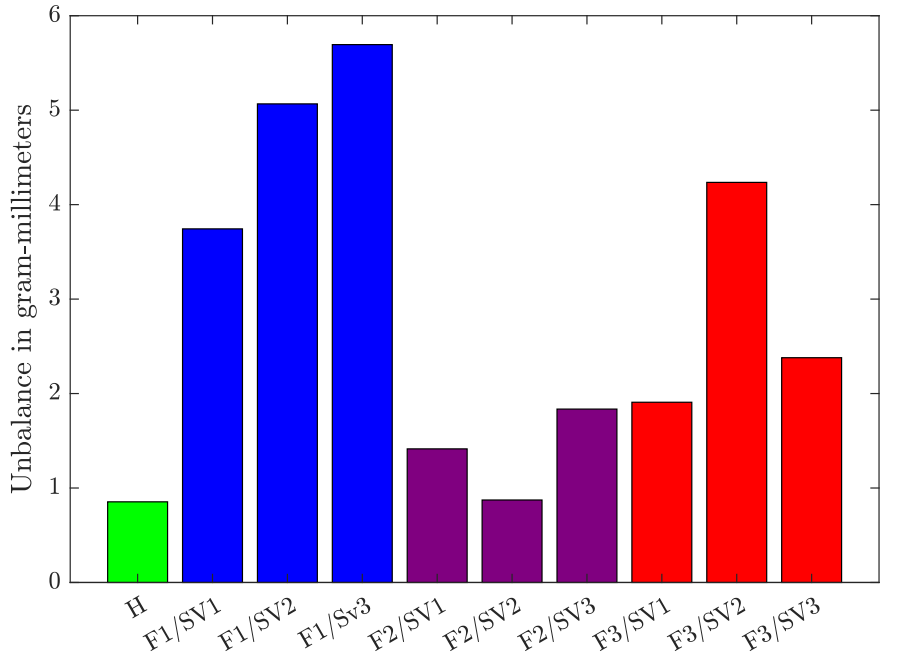



Figure 3.15: Unbalance estimation in gram-millimeters (g $\cdot$ mm) across different fault conditions and severity levels. The measured unbalance values confirm that increasing fault severity correlates with higher imbalance.

## Summary

This chapter conducted an experimental investigation of propeller faults, categorizing them based on structural characteristics and analyzing their physical properties. The study focused on three major fault types: unbalance, edge cuts, and cracks, which were systematically introduced to assess their characteristics and severity.

- **Unbalance** Caused by mass asymmetry along the propeller blades, leading to uneven weight distribution. Unbalance faults were experimentally introduced by modifying mass distribution, such as adding or removing material at specific points along the blade.

- **Edge Cuts** Physical damage at the leading edge of the propeller, disrupting airflow and affecting aerodynamic efficiency. These faults were introduced by systematically cutting sections from the edge of the blade to simulate erosion or impact damage.

- **Cracks** Structural fractures in the propeller material that can propagate under stress, compromising structural integrity. Cracks were artificially induced through controlled structural stress to simulate fatigue or manufacturing defects.

- **Experimental Fault Injection and Categorization** Faults were categorized into three severity levels: low, medium, and high, based on the extent of material loss and structural damage.

- **Measurement and Analysis** Faults were analyzed by weighing the modified propellers and measuring their physical deviations to establish baseline severity classifications.

The analysis of fault types revealed distinct differences in how unbalance and affected surface area contribute to fault severity. Edge cuts (F1) exhibited the largest affected surface areas, with increasing severity leading to higher unbalance values. This suggests that material loss from the blade edges has a significant aerodynamic and dynamic impact. Crack faults (F2) consistently showed small surface areas and low unbalance levels, indicating that cracks primarily weaken structural integrity without significantly altering mass distribution. Surface unbalance faults (F3), caused by drilled holes, resulted in high unbalance values despite having small affected surface areas, demonstrating that localized mass removal induces substantial rotational asymmetry. These findings establish ground-truth insights into how different fault types affect propeller characteristics, forming the basis for later assessments of flight dynamics-based fault detection methods.

# CHAPTER 4

# FAULT DETECTION PRINCIPLE

Accurately modeling the dynamics of a quadcopter is essential for understanding its behavior, analyzing system responses, and developing effective diagnostic techniques for fault detection. In this chapter, we integrate advanced system identification methods with experimental techniques to establish a robust framework for quadcopter fault detection. The dynamic modeling approaches—comprising data-based mechanistic modeling, ARX models, and Savitzky-Golay filtering—lay the foundation for the fault detection methodology presented later.

## 4.1 System Identification

System identification is a crucial process in modeling dynamic systems, involving the estimation of mathematical models from input-output data. Among the most prominent algorithms used for this purpose are the Prediction Error Method (PEM), frequency-domain techniques, and recursive algorithms.

**Prediction Error Method (PEM)** is a widely used time-domain approach that identifies system parameters by minimizing the difference between the measured outputs and the predicted outputs of a model. This method is particularly effective when input-output relationships are influenced by noise, as it provides robust parameter estimates through optimization techniques.

**Frequency-Domain Techniques** analyze the system's response to periodic or sinusoidal inputs, offering a complementary perspective to time-domain methods. These techniques are especially valuable for identifying resonant frequencies, damping characteristics, and other frequency-specific dynamics of the system.

**Recursive Algorithms** enable online identification by continuously updating model parameters as new data becomes available. This feature makes them highly suitable for systems with time-varying dynamics, where traditional offline methods may struggle to keep up with rapid changes.

### 4.1.1 Data-Based Mechanistic (DBM) Modeling

Data-Based Mechanistic (DBM) modeling emphasizes deriving models from empirical data while maintaining simplicity, interpretability, and physical relevance [**?**]. Unlike traditional deterministic approaches that rely on complex, detailed simulation models, DBM focuses on capturing the dominant modes of system behavior through simpler, more efficient models. The primary goal is to provide models that are both statistically valid and physically meaningful, even in the presence of noisy, irregular, or sparse data. The DBM approach's capability to handle uncertainty and noise makes it particularly well-suited for identifying minor faults, which may cause small but critical changes in system dynamics that are not easily detectable through conventional methods.

### 4.1.2 ARX Models

In this work, the MATLAB System Identification Toolbox is employed to estimate the system dynamics using AutoRegressive with eXogenous inputs (ARX) models. ARX models are widely used due to their simplicity, computational efficiency, and ability to capture linear dynamics effectively [**?**]. The ARX model is defined by the following difference equation:

$$y(t) + a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) = b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) + e(t), \quad (4.1)$$

where:

- $y(t)$ is the output at time $t$,

- $u(t)$ is the input at time $t$,
- $e(t)$ is the noise term,
- $a_1, \ldots, a_{n_a}$ and $b_1, \ldots, b_{n_b}$ are the model parameters,
- $n_a$ and $n_b$ are the orders of the autoregressive and exogenous terms, respectively.

The ARX model can be expressed compactly in transfer function form as:

$$y(t) = \frac{B(Z^{-1})}{A(Z^{-1})}u(t) + \frac{1}{A(Z^{-1})}e(t), \tag{4.2}$$

where $Z^{-1}$ is the unit delay operator, meaning that for any discrete-time signal $x(t)$,

$$Z^{-1}x(t) = x(t-1). \tag{4.3}$$

The polynomials $A(Z^{-1})$ and $B(Z^{-1})$ are defined as:

$$A(Z^{-1}) = 1 + a_1 Z^{-1} + a_2 Z^{-2} + \cdots + a_n Z^{-n}, \tag{4.4}$$

$$B(Z^{-1}) = b_0 + b_1 Z^{-1} + b_2 Z^{-2} + \cdots + b_m Z^{-m}. \tag{4.5}$$

Here, $A(Z^{-1})$ represents the characteristic polynomial of the system, and $B(Z^{-1})$ represents the input-related polynomial. The noise term $e(t)$ is filtered through the inverse of $A(Z^{-1})$, capturing the stochastic properties of the system.

### 4.1.3 SISO, MISO, and MIMO Systems

System identification can be applied to various configurations, including:

- **Single-Input Single-Output (SISO)**: A system with one input and one output.
- **Multiple-Input Single-Output (MISO)**: A system with multiple inputs and one output.
- **Multiple-Input Multiple-Output (MIMO)**: A system with multiple inputs and multiple outputs.

While MISO and MIMO models are more general and can capture interactions between multiple inputs and outputs, they are computationally more complex and require larger datasets for accurate parameter estimation [**?**]. For this work, SISO models are chosen due to their simplicity and the ability to isolate individual input-output relationships, which is particularly useful for fault detection and diagnosis in quadrotor systems. The choice of SISO models is motivated by the following factors:

- **Simplicity**: SISO models are easier to implement, analyze, and interpret compared to MISO or MIMO models.
- **Computational Efficiency**: SISO models require fewer parameters, reducing the computational burden during identification and validation.
- **Fault Isolation**: By focusing on individual input-output pairs, SISO models enable precise fault isolation, which is critical for diagnosing specific system

anomalies.

The MATLAB Identification Toolbox provides a user-friendly interface for estimating ARX models from input-output data. The toolbox uses prediction error minimization (PEM) techniques to optimize the model parameters, ensuring a good fit between the model and the experimental data [**?**].

**Validation and Model Selection**

Model validation is performed using statistical metrics such as the **Fit Percentage** and the **Mean Squared Error (MSE)**. The Fit Percentage measures how well the model output matches the experimental data, while the MSE quantifies the average squared difference between the predicted and actual outputs. Additionally, residual analysis is conducted to ensure that the residuals are uncorrelated and resemble white noise, indicating a well-fitted model.

## 4.2 Inner Loop Dynamics

In control systems, outer-loop states and inner-loop states refer to different levels of control hierarchy and dynamics. Inner-loop states are typically associated with fast dynamics and are responsible for stabilizing the system. These states often include rotational dynamics, such as angular rates and attitudes ($\phi, \theta, \psi$ and their derivatives), and are regulated by high-frequency control loops designed to ensure stability and quick responses to disturbances. On the other hand, outer-loop states are related to slower dynamics and higher-level objectives, such as position $(x, y, z)$ or trajectory

tracking. The outer loop provides reference commands for the inner loop to follow, often involving path planning or guidance laws. The interplay between these loops is critical: the inner loop ensures stability and responsiveness, while the outer loop ensures that the system achieves broader operational goals. Proper tuning of both loops is essential to maintain system performance and avoid issues such as overshoot, instability, or sluggish response.

In the context of system identification, the choice between outer-loop and inner-loop states depends on the goals and requirements of the application. Inner-loop states are typically easier to measure with high-frequency sensors and provide detailed information about the system's dynamic stability. These states are particularly useful when identifying control models that prioritize rapid stabilization and responsiveness. Conversely, outer-loop states are more suitable for identifying models focused on trajectory tracking, navigation, or high-level system objectives. However, outer-loop states can be affected by slower dynamics and external disturbances, which may complicate the identification process. While inner-loop states often yield more robust models for control purposes due to their direct relationship with the system's dynamics, the use of outer-loop states can be advantageous for applications requiring precise path-following or environmental interaction.

For propeller fault detection, inner-loop states are generally more effective because propeller faults primarily manifest as changes in rotational dynamics and stability. These faults can be directly observed through deviations in angular rates and attitudes, making inner-loop states more sensitive to detecting subtle changes caused by faults. Outer-loop states, while useful for tracking overall system performance, may

fail to capture the immediate effects of propeller faults due to their slower dynamics and integration of multiple subsystems. Thus, inner-loop states are better suited for precise and timely detection of propeller faults.

## 4.3 Savitzky-Golay Differentiation and Filtering

The Savitzky-Golay (SGolay) filter is a widely used digital signal processing technique for smoothing and differentiating noisy data. Introduced in [**?**], the SGolay filter is particularly effective in applications where preserving the shape and features of the signal is critical, such as in quadcopter sensor data processing [**?**]. The filter achieves this by fitting a polynomial to a sliding window of data points and using the polynomial coefficients to compute smoothed values and derivatives.

The SGolay filter offers several advantages over traditional filtering methods. Unlike simple moving average filters, it preserves the shape and peaks of the signal, ensuring that key features remain intact. Additionally, it enables simultaneous smoothing and differentiation, allowing both smoothed values and their derivatives to be computed in a single step. The filter coefficients are precomputed, making the process computationally efficient. However, the selection of the window size and polynomial degree requires careful tuning to balance noise reduction and signal distortion. The filter also suffers from boundary effects, as it performs poorly at the edges of the data where a full window of data points is not available.

### 4.3.1 Mathematical Formulation

The SGolay filter operates on a sliding window of $2m + 1$ data points, where $m$ is the half-width of the window. For each window, a polynomial of degree $n$ is fitted to the data using least squares regression. The polynomial is of the form:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n, \tag{4.6}$$

where $x$ represents the index of the data points within the window, and $a_0, a_1, \ldots, a_n$ are the polynomial coefficients.

The smoothed value at the center of the window is given by the zeroth-order coefficient $a_0$. The first and second derivatives are obtained from the first and second-order coefficients, respectively:

$$\text{Smoothed value} = a_0, \tag{4.7}$$

$$\text{First derivative} = a_1, \quad \text{where } a_1 \text{ is the first-order coefficient}, \tag{4.8}$$

$$\text{Second derivative} = 2a_2, \tag{4.9}$$

where the second derivative is derived from differentiating $a_2 x^2$, yielding $2a_2 x$, which when evaluated at $x = 0$ gives $2a_2$.

### 4.3.2 Filter Coefficients

The SGolay filter coefficients are precomputed using a design matrix $\mathbf{A}$, which is constructed from the polynomial basis functions evaluated at the window indices:

$$\mathbf{A} = \begin{bmatrix} 1 & -m & (-m)^2 & \dots & (-m)^n \\ 1 & -(m-1) & (-(m-1))^2 & \dots & (-(m-1))^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & m & m^2 & \dots & m^n \end{bmatrix}. \tag{4.10}$$

The filter coefficients for smoothing and differentiation are obtained from the pseudoinverse of $\mathbf{A}$:

$$\mathbf{C} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T, \tag{4.11}$$

where $\mathbf{C}$ is the matrix of filter coefficients. Each row of $\mathbf{C}$ corresponds to the coefficients for computing the smoothed value or its derivatives at the center of the window.

### 4.3.3 Applications in Quadcopter Data Processing

The SGolay filter is particularly useful for processing noisy sensor data from quadcopters, such as accelerometer and gyroscope measurements. By applying the SGolay filter, high-frequency noise can be effectively removed while preserving the underlying signal trends. Additionally, the filter's ability to compute derivatives directly from the polynomial coefficients makes it ideal for estimating velocities and accelerations from position data.

For example, consider a noisy position signal $y(t)$. The smoothed position $\hat{y}(t)$ and its derivatives can be computed as:

$$\hat{y}(t) = \sum_{k=-m}^{m} c_k y(t+k), \tag{4.12}$$

$$\dot{\hat{y}}(t) = \sum_{k=-m}^{m} d_k y(t+k), \tag{4.13}$$

$$\ddot{\hat{y}}(t) = \sum_{k=-m}^{m} e_k y(t+k), \tag{4.14}$$

where $c_k$, $d_k$, and $e_k$ are the filter coefficients for smoothing, first derivative, and second derivative, respectively.

### 4.3.4 Inner Loop Reference Commands Estimation

The SGolay filter is not only useful for smoothing sensor data but also plays a critical role in processing reference commands for the inner loop control system. As discussed in Section 4.2, the inner loop is responsible for tracking the reference roll ($\phi_{\text{ref}}$) and pitch ($\theta_{\text{ref}}$) angles, which are generated by the outer loop or a higher-level controller. To analyze the system's performance, it is essential to examine the relationship between the reference derivatives—namely, the desired roll rate ($\dot{\phi}_{\text{ref}}$) and pitch rate ($\dot{\theta}_{\text{ref}}$)—and the actual angular rates measured by the gyroscope.

In this work, the SGolay filter is applied as a pre-processing step to smooth the logged reference roll and pitch angles and estimate their derivatives. The filter is particularly useful in this context, as it allows for simultaneous noise reduction and accurate derivative computation. By fitting a polynomial to a sliding window of

reference data, the SGolay filter provides smoothed estimates of $\phi_{\text{ref}}$ and $\theta_{\text{ref}}$, as well as their first derivatives $\dot{\phi}_{\text{ref}}$ and $\dot{\theta}_{\text{ref}}$.

To ensure a fair and accurate comparison, the gyroscope measurements ($\dot{\phi}_{\text{gyro}}$ and $\dot{\theta}_{\text{gyro}}$) are also filtered using the SGolay filter to remove high-frequency noise. This filtering step is critical, as gyroscope measurements are often contaminated with noise due to sensor imperfections, vibrations, and external disturbances. The filtered gyroscope measurements are then compared with the smoothed reference derivatives to facilitate system identification.

Moreover, filtering the gyroscope readings using the same derivative window size and polynomial order as the reference derivatives ensures that the introduced delays in both signals remain consistent. This consistency is crucial for system identification, as it minimizes phase discrepancies between the reference commands and the actual gyroscope measurements. By maintaining uniform filtering parameters, the system preserves the temporal relationship between input and output signals, leading to more reliable identification of the dynamic mapping between the reference altitude rates and their corresponding gyroscope readings.

### 4.3.5 Optimization of the Derivative Window Size for System Identification

To maximize the effectiveness of the SGolay filter, it is essential to optimize the window size and polynomial order. This subsection explores the impact of these parameters on system identification accuracy. Accurate system identification requires a well-

conditioned dataset where noise is minimized while preserving the essential dynamics of the system. In this work, the SGolay filter is used as a pre-processing step to estimate the derivatives of the reference roll and pitch angles. A critical aspect of this process is selecting an appropriate window size $2m + 1$, as it directly influences the balance between noise reduction and signal fidelity.

A small window size allows for rapid response to high-frequency variations in the signal but may amplify measurement noise, leading to poor identification results. Conversely, a large window size aggressively smooths the signal and reduces noise but introduces excessive lag and may distort transient dynamics. To determine the optimal window size for system identification, a systematic analysis was conducted by evaluating different values of $m$ and assessing their impact on the accuracy of the identified model.

**Evaluation Criteria**

The selection of the optimal window size was based on the performance of the identified model, evaluated using the fitting $R^2$ score. A second-order polynomial was chosen for the SGolay filter because it provides a good balance between flexibility and computational efficiency, capturing the essential dynamics of the system without overfitting.

The modeling techniques discussed above are crucial for capturing the quadcopter's dynamic behavior, which directly informs the design and optimization of our fault detection strategy.

## 4.4 Proposed Fault Detection Method

This section presents the methodology for detecting propeller faults in multi-rotor UAVs through a systematic experimental framework. The proposed approach integrates system identification techniques with data-driven modeling, enabling a comprehensive evaluation of drone behavior under both healthy and faulty conditions. The methodology consists of four key components: system identification, fault injection, experimental validation, and fault detection, forming a structured workflow for fault diagnosis and classification.

### 4.4.1 System Identification and Healthy Model Baseline

To establish a reference model for UAV behavior, system identification is performed on a new, healthy drone. This step ensures a reliable baseline for comparison when diagnosing faulty conditions.

- Healthy Drone Selection: Two brand-new QDrones (Drone 2 & 3) are selected to ensure that no prior wear or damage affects the modeling process.
- Dynamic Model Identification: The selected QDrones are subjected to controlled flights, where inner loop dynamic inputs and outputs are recorded along a predefined trajectory.
- Model Validation: An additional QDrone (Drone 1) with more operational hours is included in the validation process to establish acceptable performance margins.

By capturing the drone's response to specific control inputs, a dynamic model of the UAV's healthy state is identified. This model serves as the benchmark for evaluating deviations caused by faults.

### 4.4.2 Fault Injection and Testing

Once the healthy model is established, faults are intentionally introduced into the propellers to assess their impact on quadrotor dynamics.

- Fault Injection: All nine faults investigated in Section 3.2.2 are examined, covering edge cuts, cracks, and surface unbalances, each tested at three severity levels.
- Controlled Test Flights: The same drone used for healthy model validation (Drone 1) is also utilized for faulty propeller cases. This ensures that all test conditions, including environmental factors and flight trajectories, remain consistent across both healthy and faulty scenarios, allowing for accurate comparative analysis.
- Data Collection: Flight data, including sensor readings and system responses, is recorded for comparison with the healthy model.

This step ensures that the fault effects are systematically captured and can be quantified for subsequent detection.

### 4.4.3 Experimental Validation and Performance Margin Estimation

To ensure the reliability of fault detection, the three QDrones (Drone 1, 2, and 3) are flown under identical healthy conditions for validation. This step accounts for natural variability in UAV behavior and establishes a performance boundary for distinguishing normal deviations from fault-induced anomalies.

- Multi-Drone Validation: Data from the three healthy QDrones is used to define nominal performance limits.
- Residual Analysis: Differences between actual and estimated outputs for healthy drones are analyzed to determine baseline residual variability.
- Threshold Definition: The 90th percentile of absolute residuals is used as the threshold beyond which deviations are classified as faults.

By incorporating multiple drones, the methodology ensures that the fault detection system is robust to minor variations in drone behavior.

### 4.4.4 Fault Detection using Residual-Based Marginal Analysis

Residual-Based Marginal Analysis forms the core fault detection mechanism, utilizing the absolute residuals between expected and observed outputs.

The approach consists of:

- Computing Residuals: The difference between actual and estimated outputs for both healthy and faulty drones.

- Threshold Comparison: If the residual exceeds the defined healthy performance boundary, the deviation is flagged as a fault signature.

The detection algorithm follows a structured approach:

**Algorithm 1** Residual-Based Fault Detection

**Input:**
  Faulty system data: Estimated outputs $\hat{y}_{faulty}$, actual outputs $y_{faulty}$
  Healthy system data: Estimated outputs $\hat{y}_{healthy}$, reference outputs $y_{healthy}$ from $n$ healthy drones

**Output:** Fault signature occurrences

**Step 1: Determine Residual Limit**
Compute residuals: $r_{healthy} = y_{healthy} - \hat{y}_{healthy}$
Set residual threshold $T_{max} = \max(|r_{healthy}|)$

**Step 2: Fault Detection**
Calculate residuals for faulty system: $r = y_{faulty} - \hat{y}_{faulty}$
**for** each time step $t$ **do**
    **if** $|r(t)| > T_{max}$ **then**
        Mark $t$ as a fault signature occurrence
    **end if**
**end for**

### 4.4.5 Fault Latency Analysis

Fault latency refers to the time interval between the onset of a fault and its detection by the system. Short-lived deviations in residual values may not necessarily indicate a fault, as they can result from sensor noise, transient disturbances, or minor fluctuations in system behavior. To prevent false fault detections, only residuals that remain above the predefined threshold for a sufficient duration are considered valid fault signatures.

The fault detection process evaluates latency by analyzing the exceeding period (P) in which the residual surpasses the pre-defined threshold before a fault is confirmed.

- Exceeding Period Calculation: The duration for which the residual remains above the healthy residual boundary (90th percentile) is measured.
- Fault Decision Criteria: If the exceeding period surpasses the Fault Minimal Latency (FML), the system classifies the event as a fault. Otherwise, it is neglected and considered noise.

This approach ensures that only persistent deviations contribute to fault detection, while transient anomalies that do not meet the latency criteria are filtered out as noise.

Figure 4.1 provides a detailed overview of the fault detection process, starting from the initial margin estimation phase and progressing through residual computation, threshold comparison, and fault latency evaluation. The flowchart outlines the step-by-step approach used to differentiate between normal variations and actual faults by defining a healthy residual boundary (90th percentile).

The process begins with obtaining both healthy and faulty drone outputs, followed by the calculation of absolute residuals for each case. These residuals are then compared against the pre-defined threshold to determine whether the deviations exceed acceptable limits. If the residual surpasses the threshold for a sustained period (exceeding period P), the system evaluates whether this duration is greater than the Fault Minimal Latency (FML). Only if the exceeding period meets or exceeds this minimum latency is the deviation classified as a fault; otherwise, it is discarded as noise.

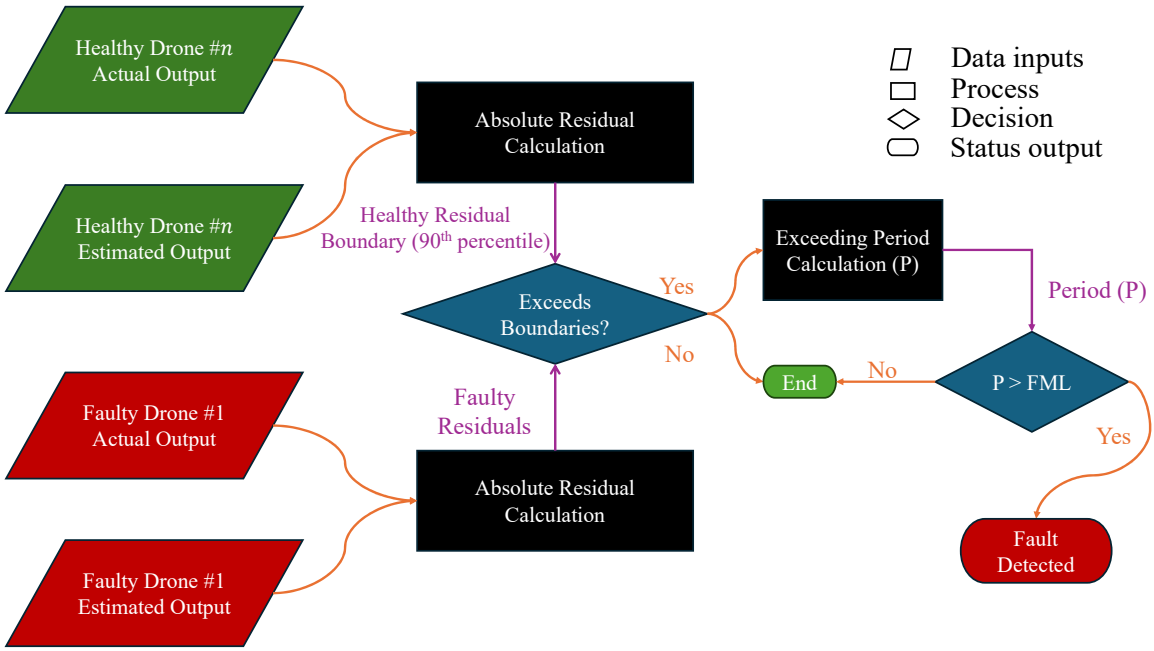



Figure 4.1: Comprehensive fault detection flowchart, illustrating the process from margin estimation to fault latency evaluation, including residual computation, threshold comparison, and final fault decision.

### 4.4.6 Methodology Workflow

The proposed methodology integrates system identification, fault injection, validation, and detection into a structured workflow (Figure 4.2).

- Healthy Model Identification: Establishing a reference dynamic model using two brand-new UAVs.
- Fault Testing: Conducting controlled flights with faulty propellers to record deviations from healthy behavior.
- Baseline Validation: Flying multiple healthy UAVs to determine acceptable performance variations.
- Fault Detection Logic: Comparing faulty drone performance against the established baseline to detect and classify faults.

## 4.5 Summary

This chapter presented a comprehensive framework for UAV fault detection by merging dynamic modeling with a structured fault detection methodology. Key points include:

- **Dynamic Modeling Foundation**: The use of system identification, ARX models, and Savitzky-Golay filtering to accurately capture quadcopter dynamics.
- **Integrated Fault Detection**: Leveraging the established models to inform the design of a robust fault detection strategy.

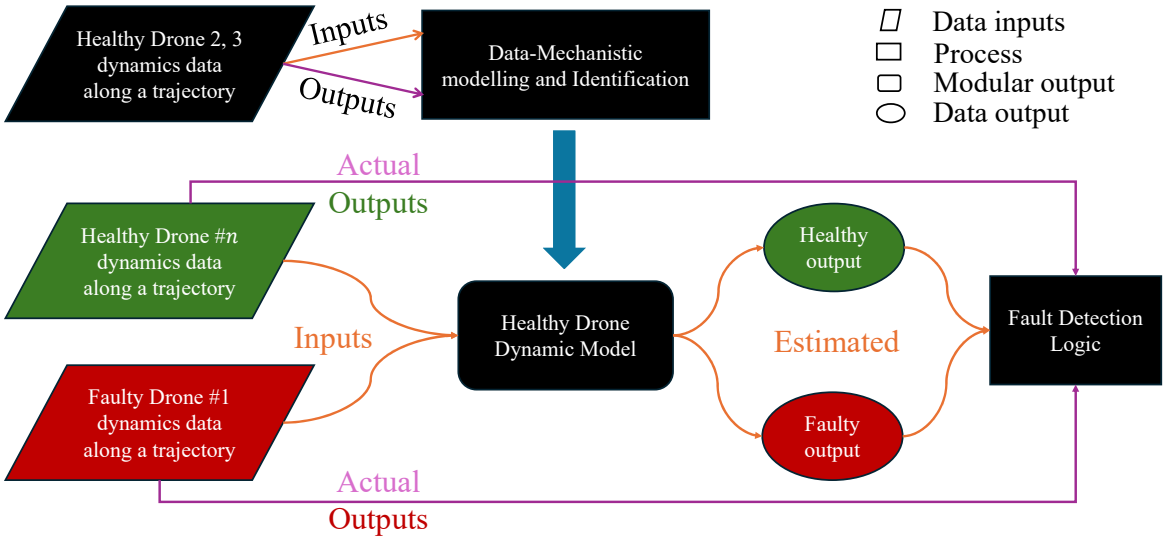



Figure 4.2: Methodology workflow overview.

- **Practical Relevance**: The techniques discussed offer a clear path from theoretical modeling to experimental validation in UAV diagnostic applications.

By uniting these approaches, the chapter provides a solid basis for subsequent experimental validation and further system analysis.

# CHAPTER 5

# EXPERIMENTAL WORK

This chapter presents the experimental validation of the fault detection methodology for multi-rotor drones by analyzing propeller faults and their impact on system dynamics. The experimental setup involved multiple drones flying a predefined square trajectory under controlled conditions, ensuring repeatable and structured inner-loop excitations for consistent fault analysis. A structured approach was followed to establish a system identification framework, inject faults into the quadcopter propellers, and evaluate fault detection performance. A data-driven methodology was employed, where a healthy quadcopter model was first identified as a baseline for fault detection. Multiple drones were tested to ensure robust validation across different platforms. A residual-based fault detection approach was applied to quantify anomalies of faulty system responses. Beyond fault detection, further analysis was conducted to extend the methodology to fault classification and severity identification, ensuring a comprehensive understanding of how different fault types affect quadcopter dynamics.

## 5.1 Experimental Setup

The experimental setup (Table 5.1) utilizes the Quanser™ Qdrone 2 with advanced onboard sensors, including two 6-DoF IMUs (IIM-42652) and a ToF height sensor, alongside a comprehensive onboard data acquisition system. Position and trajectory data were captured using a motion capture system with 14 OptiTrack X-Prime 13 cameras and processed at a sampling frequency of 1 kHz. A high-performance ground control station, equipped with an Intel Core i7 processor, NVIDIA RTX GPU, and 32GB of RAM, manages data acquisition and real-time processing. This setup (Fig. 5.1 & 5.2) ensures precise tracking and high-resolution dynamics data for evaluating propeller faults.

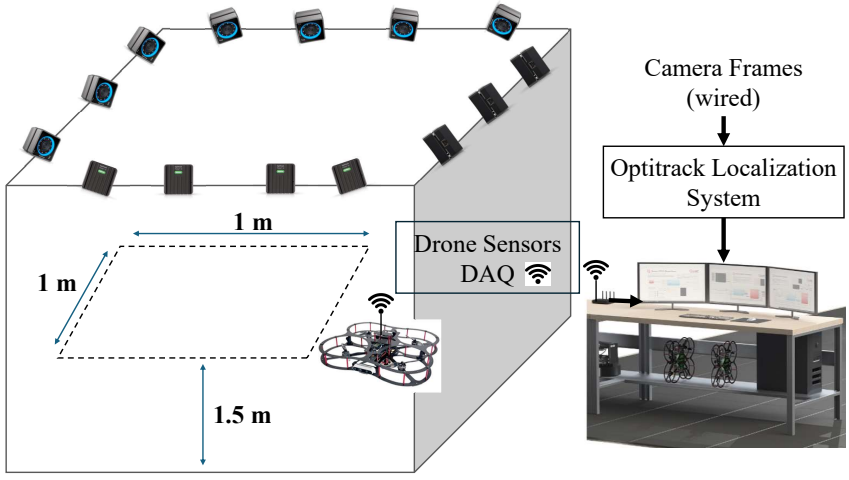



Figure 5.1: Experimental setup flow.

Table 5.1: Features of the Experimental System Setup

| **QDrone 2** | |
|---|---|
| **Dimensions** | 50 x 50 x 15 cm |
| **$L_{roll}$** | Length of roll axis: 0.254 m |
| **$L_{pitch}$** | Length of pitch axis: 0.2032 m |
| **Mass of QDrone 2** $M_d$ | 1.146 kg |
| **Mass of 3700 mAh Battery** $M_b$ | 0.358 kg |
| **Takeoff weight** $M_t$ | 1.504 kg |
| **Max Payload** | 300 g |
| **Propeller Model** | HQ Durable Prop 7x4.5 |
| **Power** | 4S 14.8V LiPo (3700mAh) with XT60 connector |
| **Flight Time** | 7-8 minutes (hover per battery charge) |
| **Motor torque constant** $K_t$ | 68.9055 N/N.m |
| **Motor speed constant** $K_v$ | 1300 RPM/V |
| **Onboard Sensors** | 2x 6-DOF IMU (gyroscope and accelerometer), 1x ToF height sensor |
| **Connectivity** | WiFi 802.11 a/b/g/n/ac (867Mbps), Micro HDMI port |
| **Camera System** | |
| **Camera Model** | 14 X-Prime 13 cameras |
| **Data Acquisition Unit** | Single OptiTrack hub |
| **Ground Control Station** | |
| **Processor** | Intel Core i7 processor |
| **GPU** | NVIDIA RTX GPU |
| **RAM** | 32GB |

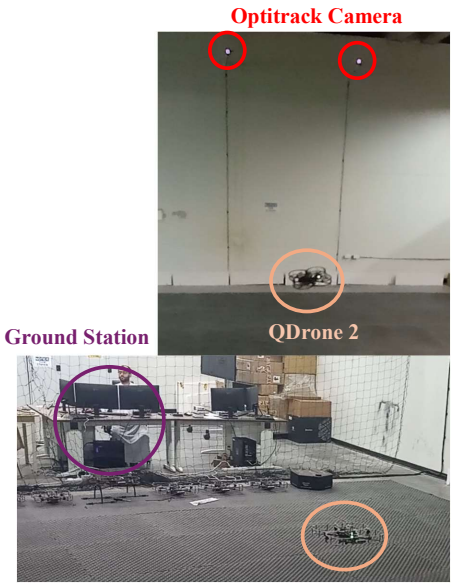



Figure 5.2: Physical experimental setup.

## 5.2 Experimental Trajectory Design

The selection of a square trajectory for the experimental flights was motivated by its ability to introduce controlled and repeatable directional changes, thereby exciting the drone's inner-loop control states in a structured manner. As shown in Figure 5.3, the drone follows a predefined $1 \times 1$ m square path at a constant altitude, ensuring a standardized motion profile across all test cases.

Given the nature of multi-rotor dynamics, each segment of the square trajectory induces specific control challenges. During straight-line motion along the edges, the quadrotor maintains a relatively stable roll ($\phi$) and pitch ($\theta$) state, with minor corrective adjustments due to environmental disturbances and inherent system imperfections. However, at the sharp 90-degree turns at each vertex, a distinct excitation in the inner-loop attitude states occurs. Figure 5.4 illustrates the variations in roll and

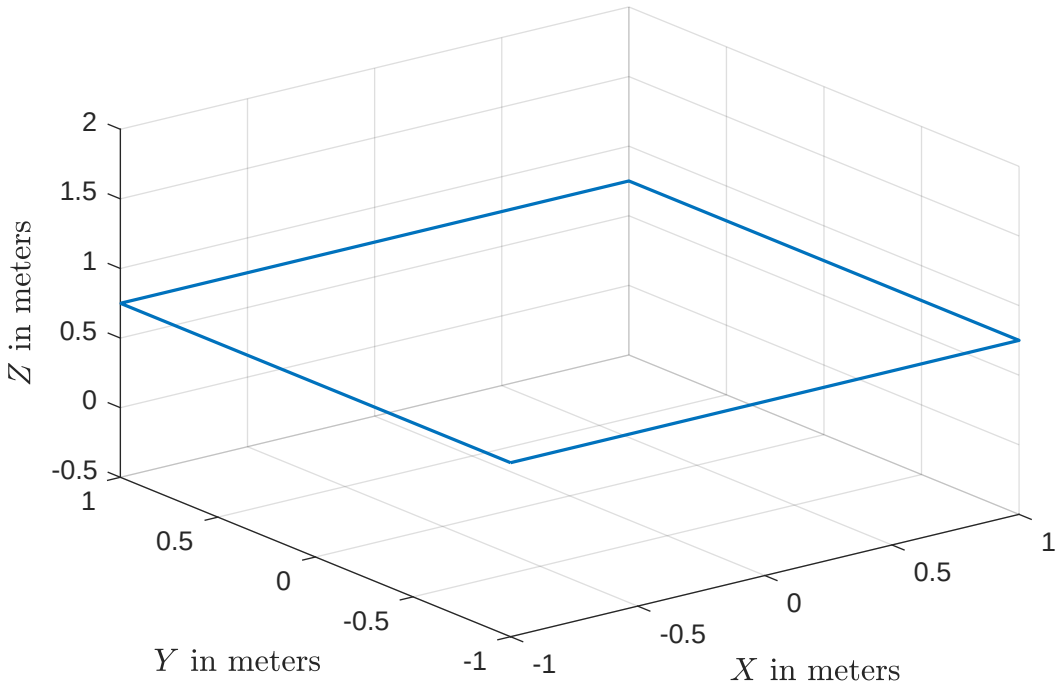



Figure 5.3: The predefined square trajectory used for all experimental analysis.

pitch angles over time, highlighting the transient disturbances introduced at the trajectory corners. Each turn demands a coordinated control response to rapidly adjust attitude states and redirect the drone's velocity vector while compensating for inertial effects. The sharp spikes in $\phi$ and $\theta$ at these transition points indicate the system's effort to maintain tracking accuracy while executing aggressive orientation changes. This controlled excitation is crucial for fault detection, as deviations from expected response patterns can signal potential abnormalities in propeller performance or control effectiveness.

To analyze the dynamic behavior of the drone along the square trajectory, it is essential to consider the respective roll and pitch rates ($\dot{\phi}$ and $\dot{\theta}$). The first derivatives provide a more comprehensive representation of the system's response, particularly

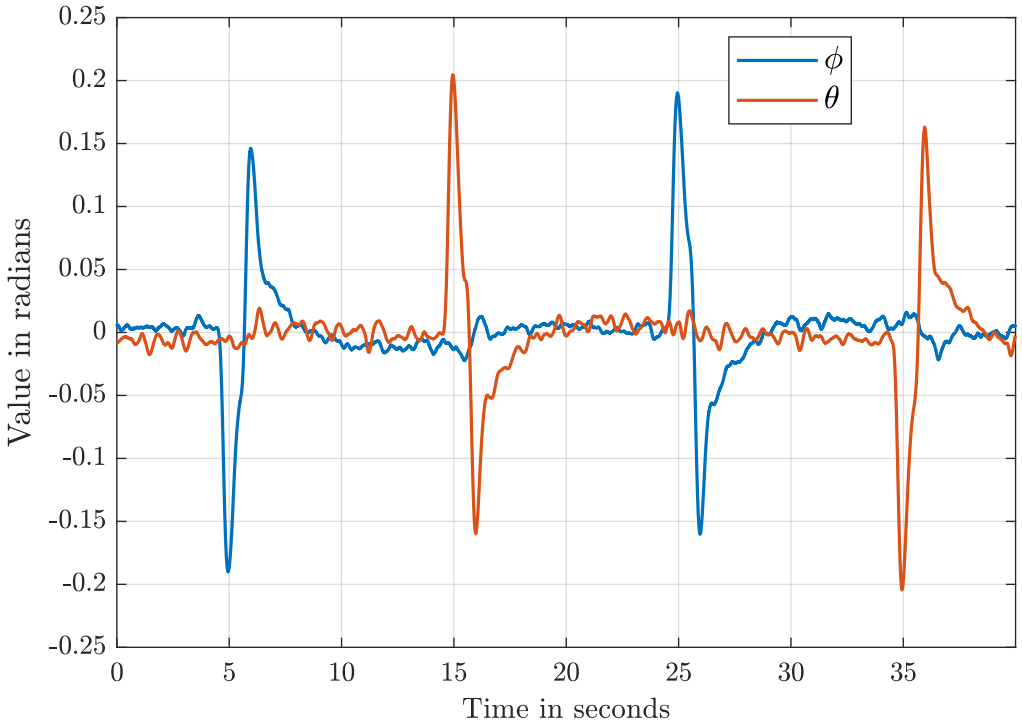



Figure 5.4: Recorded roll ($\phi$) and pitch ($\theta$) states during the square trajectory flight. Increased attitude excitations occur at trajectory corners due to rapid directional transitions.

during sharp turns, where rapid variations in attitude demand precise control corrections. By examining the rates of roll and pitch, the transient behavior and the control effort required to stabilize the drone become more evident, offering deeper insights into fault-induced anomalies. If the excitation observed in the first derivatives is insufficient to capture the full dynamics, higher-order derivatives ($\ddot{\phi}$ and $\ddot{\theta}$) may be analyzed to reveal additional frequency components associated with the system's response to disturbances.

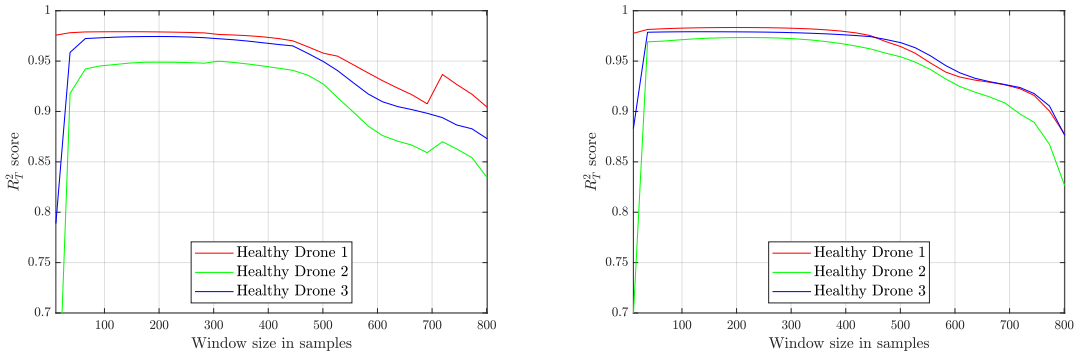
## 5.3 Savitzky-Golay Filter Parameter Optimization

The roll rate ($\dot{\phi}$) and pitch rate ($\dot{\theta}$) are derived from two primary sources: the reference attitude commands ($\phi_{\text{ref}}$ and $\theta_{\text{ref}}$), which are differentiated to obtain the desired rates, and the actual rates, which are measured directly from the gyroscope sensor onboard the UAV.

Different window sizes were tested to determine the optimal differentiation and smoothing parameters for system identification. The identification process was conducted on data collected from a single healthy drone (Drone 1) using MATLAB identification toolbox, ensuring that the derived model accurately captured the system dynamics. To assess the generalizability of the identified $\dot{\phi}$ and $\dot{\theta}$ models, validation was performed using flight data from all three healthy drones (Drone 1, 2 & 3), with their respective validation scores plotted for comparison (Figures 5.5a & 5.5b).

The results indicate that a window size of 100 samples provided an acceptable trade-off between noise reduction and signal fidelity. Smaller window sizes introduced

excessive noise, while larger sizes initially preserved similar performance without significant improvement, and eventually led to excessive smoothing, distorting the system's response.




(a) Roll rate ($\dot{\phi}$) signal.

(b) Pitch rate ($\dot{\theta}$) signal.

Figure 5.5: Effect of different window sizes on the differentiation of roll rate ($\dot{\phi}$) and pitch rate ($\dot{\theta}$) signals. A window size of 100 samples was found to provide a balance between noise suppression and signal fidelity. Smaller window sizes resulted in excessive noise amplification, while larger window sizes introduced over-smoothing, distorting the system's response.

## 5.4 Inner Loop Models Identification

Two linear ARX models were identified using data from a healthy flight experiment of the new healthy drones (Drone 2 & 3), where the reference inner-loop states $\dot{\phi}_{\text{ref}}$ and $\dot{\theta}_{\text{ref}}$ served as model inputs, and the actual measured states $\dot{\phi}$ and $\dot{\theta}$, obtained from the gyroscope, were used as outputs during training. The model structure was systematically evaluated by adjusting the denominator and numerator orders to balance accuracy and complexity.

A second-order denominator was selected as it provided a reasonable trade-off between model simplicity and fitting performance, achieving an $R_T^2$ score of 70%, which

indicates a reliable level of explanatory power while avoiding excessive complexity. Increasing the denominator order to three resulted in a significantly lower $\varepsilon_{\text{fit}}$ error but introduced overfitting, with one of the poles exceeding unity, leading to an unstable model representation. The numerator order was tuned between 0 and 4, with the best-fitting models featuring first and second-order numerators. Higher-order numerators resulted in marginal improvement while increasing the risk of capturing noise rather than system dynamics. Delays were explored in the range of 0 to 4 time steps, ensuring proper alignment between inputs and outputs.

The identified linear ARX models for the inner-loop roll and pitch rates, represented in terms of $Z^{-1}$, are given as follows:

$$G_{\dot{\phi}}(Z^{-1}) = \frac{0.0743Z^{-1} + 0.0616Z^{-2}}{1 - 1.5326Z^{-1} + 0.5333Z^{-2}} \tag{5.1}$$

$$G_{\dot{\theta}}(Z^{-1}) = \frac{0.1014Z^{-1} + 0.0573Z^{-2}}{1 - 1.9363Z^{-1} + 0.9368Z^{-2}} \tag{5.2}$$

where $G_{\dot{\phi}}$ and $G_{\dot{\theta}}$ represent the transfer functions for the roll and pitch responses, respectively. The shift operator $Z^{-1}$ represents a single time-step delay in discrete time.

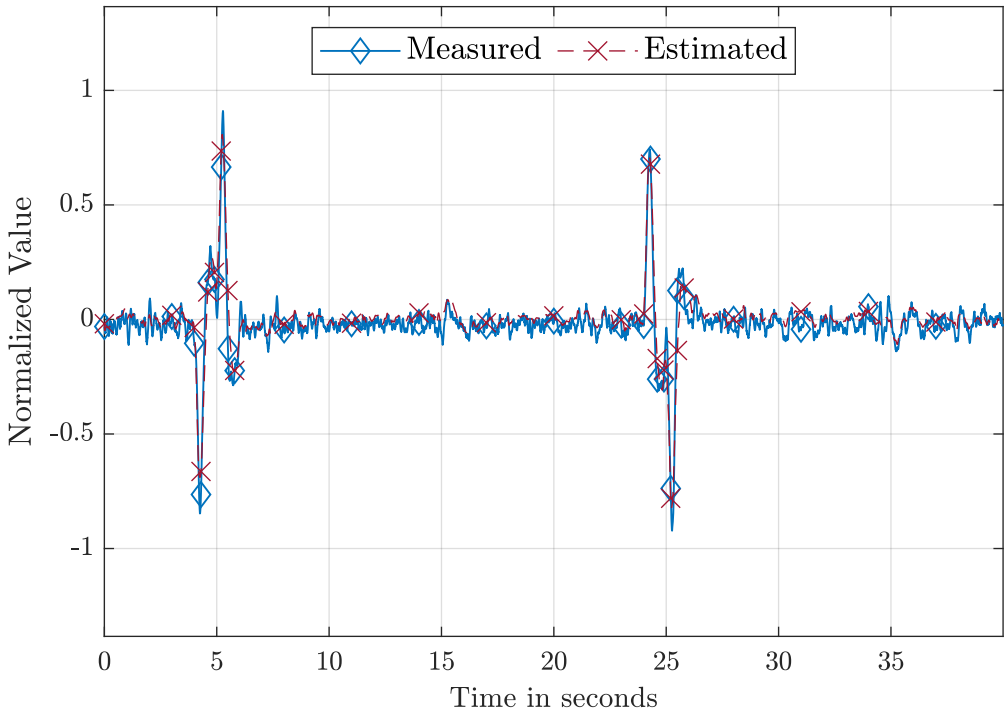
These transfer functions illustrate key differences in system behavior. The roll model (5.1) shows a moderate coefficient magnitude in the denominator, resulting in better generalizability across multiple validation datasets. On the other hand, the pitch model (5.2) has significantly larger denominator coefficients, which suggests longer system memory and reduced robustness.

Figures 5.6a and 5.6b present the model fitting results for roll and pitch rates, respectively. The fitting results confirm that while both models successfully capture the inner-loop system dynamics.

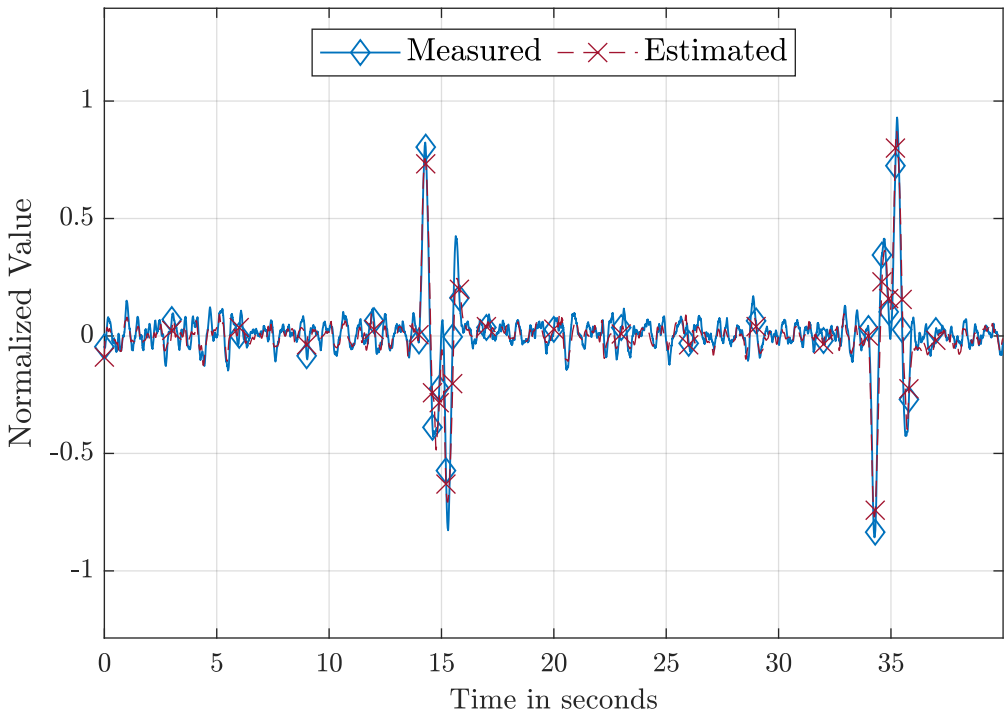
## 5.5 Residual-Based Marginal Analysis

Based on the performance margin estimation method explained in section 4.4.3, three healthy trajectories, each corresponding to one of the three drones, are utilized to quantify the acceptable residual margin for general healthy systems based on the estimated models. This process involves inputting each desired state into its corresponding model and computing the residuals, which are defined as the differences between the estimated output states and the actual measured states. These residuals serve as a metric to evaluate the accuracy and reliability of the estimated models under healthy operating conditions. Subsequently, the faulty trajectories are passed through the same estimated models to monitor the residual behavior. The goal is to compare these residuals against the predefined acceptable error margin, which is derived from the healthy system data. Figures 5.7a and 5.7b illustrates the absolute residuals for the roll and pitch rates healthy models, providing a visual representation of the deviations observed in the system. The residuals are obtained by subtracting the actual output states from the estimated ones. The acceptable error margin is chosen to include more than 90% of the absolute residual data points under healthy conditions.

The validation results show that both $G_{\dot{\phi}}$ and $G_{\dot{\theta}}$ models effectively captured the




(a) Model fitting for roll rate $\dot{\phi}$.



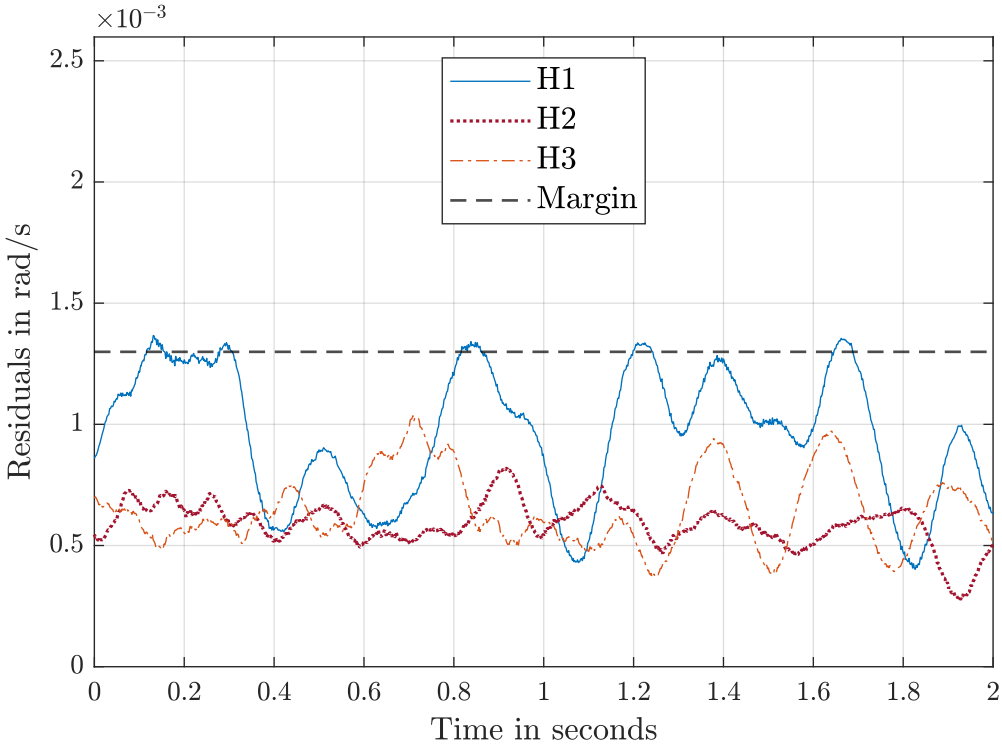

(b) Model fitting for pitch rate $\dot{\theta}$.

Figure 5.6: Model validation results for the identified $G_{\dot{\phi}}$ and $G_{\dot{\theta}}$ transfer functions.

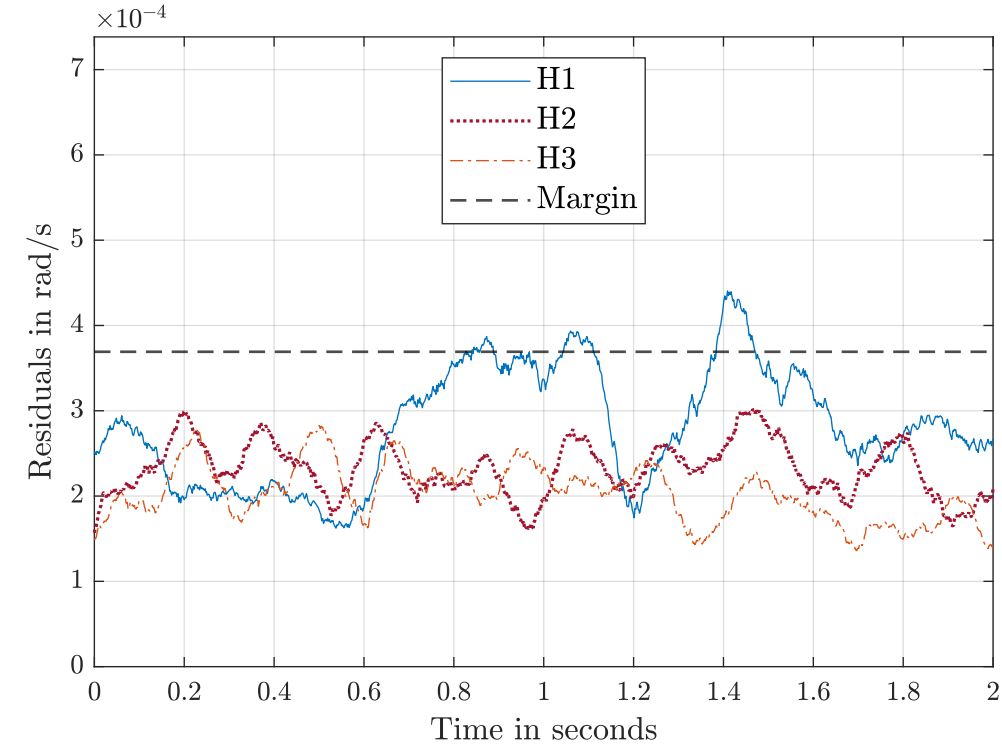
system dynamics, exhibiting minimal error margins in milli-units. The maximum error observed in the roll rate model $G_{\dot{\phi}}$ was approximately $1.3 \times 10^{-3}$, whereas the pitch rate model $G_{\dot{\theta}}$ exhibited a significantly higher maximum error exceeding $4 \times 10^{-3}$. This indicates that while both models performed well, the pitch rate model demonstrated reduced generalizability, particularly when validated on Drone 1, which had a longer operational period, deviating from the baseline healthy model derived from newer drones.

To minimize computational cost while maintaining reliable fault detection, the roll rate model $G_{\dot{\phi}}$ is chosen as the primary representation of the inner-loop dynamics. The roll model exhibited lower residual errors and better generalization across different drones, making it the more robust candidate for dynamics representation. Given the increased error in the pitch model and its sensitivity to variations in drone condition, relying solely on the roll model provides a computationally efficient solution for real-time implementation without significant accuracy trade-offs.

From $G_{\dot{\phi}}$ results in figure 5.7a, it is evident that the newer drones (drones 2 and 3), which have accumulated fewer operating hours, exhibit a closer match to the healthy model with minimal deviations. Specifically, the residuals for these drones range between $0.5 \times 10^{-3}$ and $1 \times 10^{-3}$. In contrast, drone 1, which has been operational for a significantly longer period, also aligns with the healthy model but demonstrates slightly higher deviations, with residuals ranging from $0.5 \times 10^{-3}$ to $1.3 \times 10^{-3}$. This discrepancy can be attributed to the effects of prolonged usage, which may introduce minor changes in the system's model. These observations highlight the importance of considering operational history when evaluating the health and performance of




(a) Residuals of the roll rate model $G_{\dot{\phi}}$ across three healthy drones.




(b) Residuals of the pitch rate model $G_{\dot{\theta}}$ across three healthy drones.

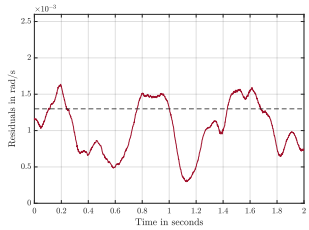
Figure 5.7: Absolute residuals for roll rate ($\dot{\phi}$) and pitch rate ($\dot{\theta}$) across three healthy drones. The roll model $G_{\dot{\phi}}$ demonstrates lower residual errors and better generalizability, making it the preferred choice for the next steps of fault detection.

dynamical systems.

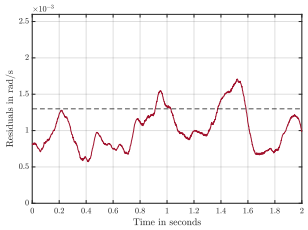
### 5.5.1 Fault Detection

Following the establishment of the acceptable residual margin using healthy trajectories, all nine fault samples were systematically tested against this margin to evaluate their residual behavior as explained in section 4.4.4. Each fault sample was passed through the estimated models, and the residuals were computed by comparing the estimated output states with the actual measured states. Figures 5.8a through 5.8i illustrate the residuals of all nine fault samples plotted against the predefined healthy margin. Each figure corresponds to a specific fault, showcasing the deviations of the residuals from the healthy margin over time. These plots highlight the magnitude and duration of residual excursions beyond the acceptable margin.

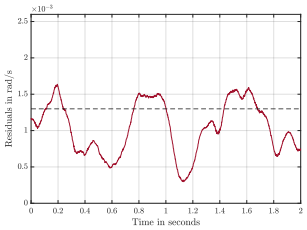
To detect faults in the system, the fault residuals that exceed the predefined healthy margin are identified, and the - fault signature duration (FSD) - duration for which these residuals remain outside the acceptable margin is calculated. This duration is a critical metric for distinguishing between transient anomalies and persistent faults. Figure 5.9 illustrates the durations of residual margin crossings for nine fault samples and three healthy samples. From the figure, it is evident that the residuals of all fault samples exceed the healthy margin multiple times, with varying FSDs. In contrast, the healthy samples exhibit only very short FSDs and infrequent crossings, which are attributed to noise or negligible disturbances.



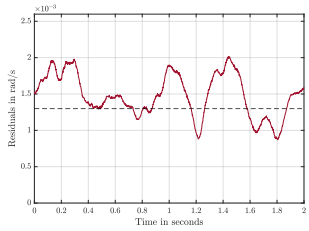

(a) Fault group 1 - Severity level 1 (F1/SV1) residuals



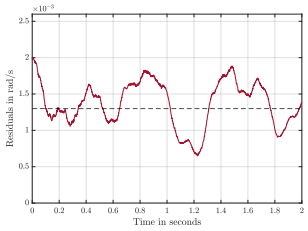

(b) Fault group 1 - Severity level 2 (F1/SV2) residuals



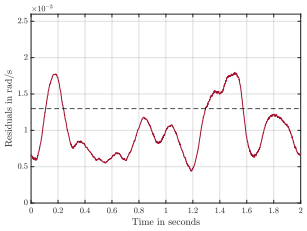

(c) Fault group 1 - Severity level 3 (F1/SV3) residuals



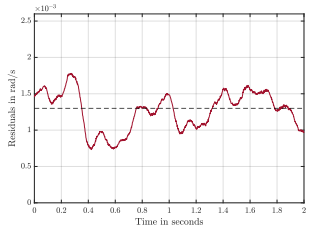

(d) Fault group 2 - Severity level 1 (F2/SV1) residuals



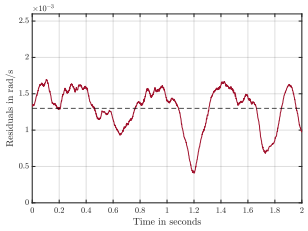

(e) Fault group 2 - Severity level 2 (F2/SV2) residuals



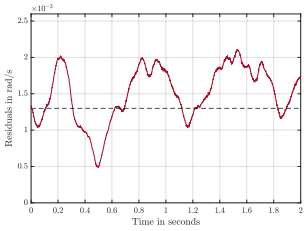

(f) Fault group 2 - Severity level 3 (F2/SV3) residuals




(g) Fault group 3 - Severity level 1 (F3/SV1) residuals




(h) Fault group 3 - Severity level 2 (F3/SV2) residuals




(i) Fault group 3 - Severity level 3 (F3/SV3) residuals

Figure 5.8: Residuals across the healthy margin for all nine fault samples. Each subfigure corresponds to a specific fault, showing the deviations of the residuals from the healthy margin over time.

### Fault Minimal Latency

To ensure robust fault detection, a minimum duration threshold - or Fault Minimal Latency (FML) - is established. A residual crossing is classified as an actual fault signature only if it persists beyond this threshold, where the FSD is longer than the FML. The maximum margin crossing period observed in the healthy samples is used as the baseline for defining the FML (Figure 5.9). Any sample exhibiting an FSD that exceed this baseline is considered faulty. This approach minimizes false positives and

ensures reliable fault identification in the system.

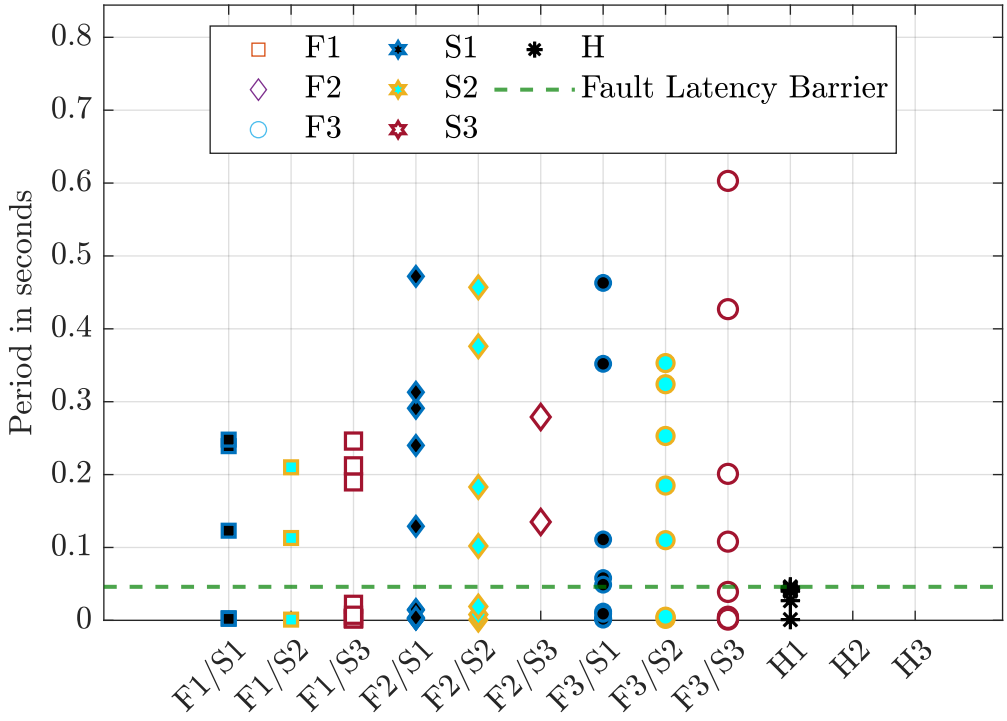



Figure 5.9: Durations of residual margin of the roll rate $\dot{\phi}$ crossings for nine fault samples and three healthy samples (FSDs). The plot highlights the periods during which the residuals exceed the predefined healthy margin, with fault samples showing prolonged deviations compared to healthy samples. The maximum margin crossing period of the healthy samples is used to define the FML threshold.

### 5.5.2 Fault Classification

The fault classification process involves analyzing the maximum residual margin for each fault. This is achieved by calculating the difference between the residual values and the predefined healthy margin line during the maximum FSD of each fault. To ensure accurate analysis, transient noise —defined as residual deviations with durations shorter than the FML — is neglected. This filtering step is crucial to eliminate false positives and focus on persistent fault signatures.

Figure 5.10 presents the maximum residual margin for each fault type, plotted against the corresponding maximum fault signature durations (FSDs). The results reveal a distinct separation between the three fault categories, reinforcing their unique dynamic characteristics and enabling effective classification.

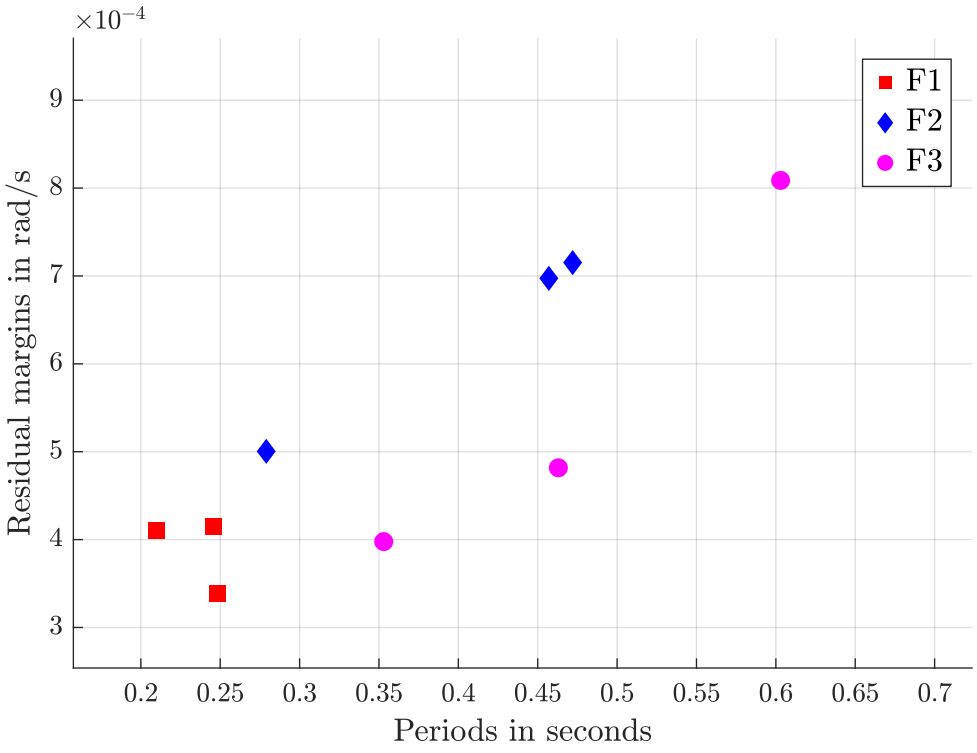



Figure 5.10: Maximum residual margin plotted against the maximum fault periods for each fault type. The plot demonstrates clear separation between fault types, enabling effective classification.

As discussed in Section 3.2.2 and illustrated in Figure 3.12, the three fault categories — edge cuts (F1), cracks (F2), and surface unbalance faults (F3) — form well-separated groups in terms of unbalance and affected surface area. This separation is further supported by the findings in Figure 5.10, where each fault type exhibits a unique range of residual margins and FSDs. These results highlight the influence of different fault mechanisms on the drone's dynamic response, confirming that un-

balance alone does not fully capture the severity of a fault. Instead, factors such as stiffness reduction (in cracks) and localized mass asymmetry (in drilled holes) contribute significantly to the system's stability and fault detectability.

Edge cut faults (F1) demonstrate the most pronounced aerodynamic disturbances due to significant material removal from the blade edges, beside having high unbalance values. The aerodynamic disturbance is due to large affected surface areas. However, despite these substantial geometric changes, F1 faults exhibit relatively short FSDs and low residual margins. This suggests that as more material is removed from the blade edges, the reduction in weight is accompanied by a proportional decrease in the generated lift, potentially mitigating the overall dynamic effect. The simultaneous reduction of mass and aerodynamic force may counteract each other, leading to lower-than-expected deviations in the system response. This suggests that while the fault is severe in terms of structural modification and unbalance, the aerodynamic effects may help compensate for its impact, likely due to the opposing directions of lift and weight forces.

Figure 5.11 illustrates the effect of edge cuts on the aerodynamic and gravitational forces acting on the propeller. The front view clarifies how this mass reduction affects the force distribution. Since lift is generated by the blade's surface area and airfoil characteristics, removing material from the edges decreases both weight and lift simultaneously. Given that lift acts in the opposite direction to weight, the reduction of both forces may create a balancing effect that minimizes the net change in system response.

However, this hypothesis requires further validation. The redistribution of aerody-

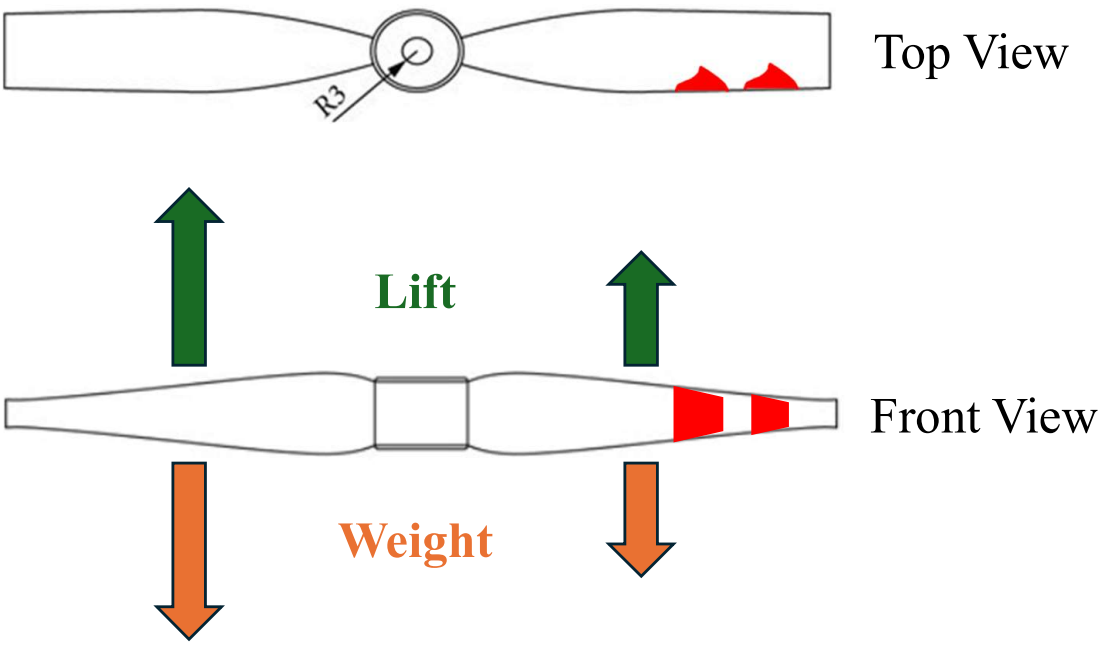



Figure 5.11: Effect of edge cuts on lift and weight distribution. The top view illustrates the removal of material from the blade edges, reducing overall mass. The front view highlights the corresponding reduction in both lift and weight. Since lift acts in the opposite direction to weight, the simultaneous decrease in both forces may mitigate the overall dynamic effect, leading to minimal residual deviations despite increasing unbalance.

namic forces along the propeller blade may result in altered flow properties, potentially influencing thrust efficiency and stability. Computational fluid dynamics (CFD) simulations are necessary to analyze how the airflow around the modified blade structure interacts with the remaining aerodynamic surfaces. Such an analysis would help determine whether the observed mitigation effect is due to true aerodynamic balancing or another compensatory mechanism within the UAV's flight dynamics.

Crack faults (F2), on the other hand, present an entirely different dynamic behavior. Despite having the smallest affected surface area and lowest contribution to unbalance, F2 faults result in the high residual margins and long FSDs. This suggests that cracks influence the structural stiffness of the propeller in a way that impacts the thrust generation more than simple mass removal or aerodynamic alterations. The prolonged FSDs indicate that the drone experiences extended instability periods as it struggles to compensate for the underlying structural weakness introduced by the cracks. These findings emphasize that crack propagation alters the load distribution on the blades, leading to significant performance degradation over time.

Surface unbalance faults (F3), caused by drilled holes, introduce high unbalance despite affecting only small to moderate surface areas. Their dynamic impact falls between F1 and F2 faults, with residual margins increasing from low to high as severity increases, along with consistently long FSDs. The concentrated mass loss from drilled holes generates significant rotational asymmetry, making the drone's response highly dependent on the severity and positioning of the faults. Unlike edge cuts, which gradually affect the aerodynamics across a larger surface, or cracks, which influence stiffness, F3 faults create localized imbalance that leads to high-frequency oscillatory

disturbances. This explains their moderate yet persistent impact on the drone's residual response and dynamic stability.

Table 5.2: Summary of Fault Types with Affected Area, Unbalance, Residual Margins, and Fault Signature Durations (FSDs)

| **Fault Type** | **Affected Area and Unbalance** | **Residual Margins (rad/s)** | **Fault Signature Duration (FSDs) (s)** | **Physical Interpretation** |
|---|---|---|---|---|
| Edge Cut (F1) | Large affected area High unbalance | Low | Short | Dynamic response adapts relatively quickly to aerodynamic disturbance |
| Crack Fault (F2) | Small affected area Low unbalance | High | Long | Excessive stiffness loss significantly impacts thrust generation |
| Surface Unbalance (F3) | Small affected area High unbalance | Mixed | Long | Localized mass removal causes rotational asymmetry, leading to oscillatory disturbances |

### 5.5.3 Fault Severity Identification

Following the classification of fault types, an additional analysis was conducted to examine how increasing severity affects the system response. Specifically, the relationship between residual margins and measured unbalance was investigated for each fault category. Figure 5.12 illustrates the variation in residual margins as a function of measured unbalance across different severity levels of edge cut faults (F1), crack faults (F2), and surface unbalance faults (F3). The observed trends highlight distinct severity-dependent behaviors across these fault categories, emphasizing that unbalance alone does not fully determine a fault's impact.

Surface unbalance faults (F3) represent the most direct fault mechanism, where severity is primarily dictated by the amount of mass removed and the resulting unbalance. The trend observed in Figure 5.12 confirms that increasing the severity of surface unbalance faults directly increases the residual margins, emphasizing that the dynamic effects stem largely from the induced rotational asymmetry. Unlike edge

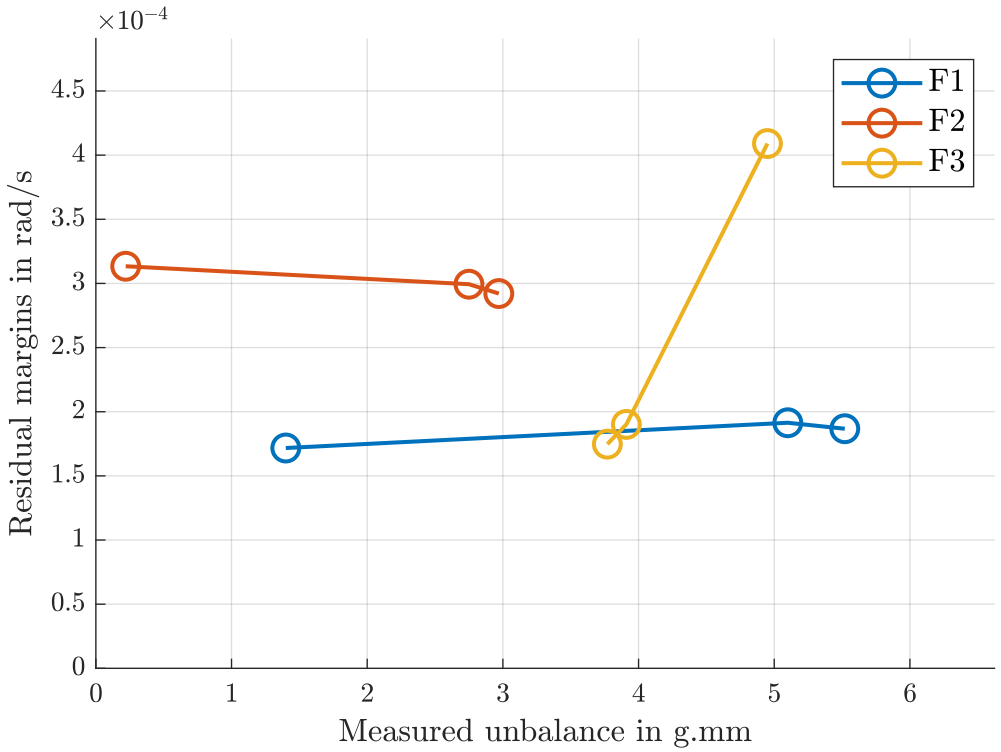



Figure 5.12: Analysis of fault severity effects across different fault types. The relationship between residual margins and measured unbalance highlights distinct severity trends for each fault category.

cuts, where aerodynamic interactions may stabilize the effect, or cracks, where stiffness dominates, surface unbalance faults follow a nearly linear relationship between severity and residual impact, making them highly predictable in terms of fault detection.

Despite the clear correlation observed for surface unbalance faults (F3), the same level of sensitivity to increasing severity was not evident in edge cut faults (F1) and crack faults (F2). As seen in Figure 5.12, the residual margins for F1 and F2 exhibited minimal variation across different severity levels, suggesting that additional factors beyond unbalance may be influencing their detectability. In the case of edge cut faults, aerodynamic effects might play a stabilizing role, reducing the impact of incremental mass removal on residual behavior. Similarly, for crack faults, structural stiffness variations could dominate the response, preventing a straightforward increase in residuals with severity. This lack of strong residual differentiation underlines the importance of incorporating multi-faceted diagnostic criteria beyond simple unbalance measurements to improve fault severity classification in rotating systems.

This severity-based analysis reinforces the idea that fault classification must extend beyond unbalance and affected surface area to consider additional mechanical and aerodynamic influences. By incorporating insights into severity-dependent variations, fault detection methodologies can be refined to distinguish between minor performance deviations and critical system faults more effectively.

# Summary

The experimental study validated the fault detection framework by analyzing propeller faults and their impact on quadcopter dynamics. A data-driven system identification approach was used to establish a healthy baseline model, against which nine fault cases—categorized as edge cuts (F1), cracks (F2), and surface unbalance faults (F3)—were tested through controlled flights. Fault detection relied on residual margin analysis to quantify deviations from the healthy model, while fault latency filtering ensured only sustained anomalies were classified. drones were flown along a structured square trajectory to introduce repeatable inner-loop excitations, aiding in fault differentiation. Savitzky-Golay filtering improved roll and pitch derivative estimation, refining residual calculations. Severity-based analysis further distinguished fault impact across categories, emphasizing the roles of aerodynamic effects, stiffness reduction, and mass asymmetry.

Key findings include:

- Edge cut faults (F1) caused high unbalance but moderate residual margins. At higher severity levels, the residuals plateaued, suggesting aerodynamic redistribution mitigates disturbances. Further CFD analysis is needed to confirm this effect.

- Crack faults (F2) had minimal unbalance impact but high residual margins, indicating that stiffness reduction rather than mass distribution influences thrust generation and system behavior.

- Surface unbalance faults (F3) showed a direct correlation between unbalance and residual margins, confirming mass removal as the primary factor affecting quadcopter instability. Higher severity led to proportionally greater residual deviations.

The findings validate the proposed fault detection approach and provide a foundation for a more advanced fault classification and severity assessment.

# CHAPTER 6

# CONCLUSION AND FUTURE WORK

This thesis presented a systematic approach to fault detection in multirotor UAVs by analyzing the effects of propeller faults on flight dynamics. A data-driven methodology was employed, integrating system identification techniques with experimental validation to establish a robust fault detection framework. The key contributions of this work include:

- A comprehensive study of propeller faults, categorizing their impact based on aerodynamic and structural characteristics.
- Implementation of a system identification approach to model quadrotor dynamics under both healthy and faulty conditions.
- Development of a residual-based fault detection mechanism that distinguishes between different fault types and severity levels.

- Application of Savitzky-Golay filtering to enhance signal processing for reliable residual estimation.
- Validation of the proposed methodology through controlled flight experiments, demonstrating its effectiveness in detecting and classifying faults.

The results indicate that different fault types exhibit distinct residual patterns, reinforcing the necessity for fault-specific classification techniques. Moreover, the analysis confirmed that aerodynamic effects, mass distribution, and structural integrity all contribute to fault severity, highlighting the complexity of propeller fault diagnosis in UAVs.

While this research provides a solid foundation for UAV fault detection, several areas warrant further exploration and improvement:

- **Enhanced Fault Classification Models:** Future work can explore advanced machine learning and deep learning techniques to improve fault classification accuracy. Neural networks, particularly convolutional and recurrent architectures, can be trained on a larger dataset to refine the detection of fault signatures and enhance robustness against environmental variations.
- **Integration of Real-Time Onboard Processing:** The current methodology relies on post-processing flight data. Implementing real-time fault detection onboard using embedded systems, such as NVIDIA Jetson or Raspberry Pi, could enable immediate corrective actions during flight, enhancing UAV resilience and operational safety.

- **Incorporation of Computational Fluid Dynamics (CFD) Analysis:** A deeper understanding of the aerodynamic effects of propeller faults can be achieved through CFD simulations. This would allow for a more detailed exploration of airflow redistribution and its influence on UAV stability, improving predictive modeling for fault diagnosis.

- **Adaptive Control Strategies for Fault Mitigation:** Beyond detection, future research should investigate adaptive control techniques that compensate for identified faults in real time. Model predictive control (MPC) and reinforcement learning-based controllers could dynamically adjust thrust allocation and flight parameters to maintain stability under faulty conditions.

- **Extension to a Broader Range of UAV Configurations:** This research primarily focused on quadrotors; however, the methodology can be extended to other UAV configurations, such as hexacopters and octocopters. This would involve modifications in system modeling and fault impact analysis tailored to different rotor arrangements.

This thesis contributes to the growing field of UAV reliability by providing a structured framework for detecting and analyzing propeller faults. By leveraging data-driven methods and experimental validation, it lays the groundwork for more advanced fault detection and mitigation strategies. As UAV applications continue to expand, further advancements in autonomous fault recovery and adaptive control will be crucial for ensuring operational safety and efficiency.

# End of text

# VITAE

- Name: Mohssen Essam Elshaar
- Nationality: Egypt
- Date of Birth: 09-July-1999
- Email: *mohssen.elshaar@gmail.com*
- Permenant Address: 1015, 1st, 6th of October, Giza, Egypt

- Bachelor's of Science (2017 - 2022): Aerospace Engineering, University of Science and Technology at Zewail City