Constraint Aware Reinforcement Learning for Aeroelastic Aircraft

A hybridization of Reinforcement Learning with Model Predictive Control

Patrick Kostelac

TU Delft

Constraint Aware Reinforcement Learning for Aeroelastic Aircraft

A hybridization of Reinforcement Learning with Model Predictive Control

Thesis report

by

Patrick Kostelac

to obtain the degree of Master of Science
at the Delft University of Technology
to be defended publicly on July 2, 2025 at 13:00

Thesis committee:

Chair:	Dr.ir. Erik-Jan van Kampen
Supervisors:	Dr. Anahita Jamshidnejad
	Dr. Xuerui Wang
External examiner:	Dr. Azita Dabiri
Place:	Faculty of Aerospace Engineering, Delft
Project Duration:	December, 2024 - July, 2025
Student number:	5238676

An electronic version of this thesis is available at
`https://brightspace.tudelft.nl/d2l/le/content/560234/Home`.

Faculty of Aerospace Engineering · Delft University of Technology

Preface

The completion of this thesis marks the end of my Master's degree in Aerospace Engineering. It has been one of the most rewarding parts of my academic journey, it has been challenging, engaging, and genuinely enjoyable. This project gave me the opportunity to explore a topic I am deeply interested in, while also pushing me to grow as a researcher. From long hours spent puzzling over technical issues to the satisfaction of seeing ideas finally come together, the experience has been both intellectually and personally fulfilling.

I am grateful to my friends and family for their support throughout this process. While their help wasn't always technical, their encouragement, patience, and ability to lift my spirits made all the difference during more difficult periods. I truly appreciate their continued support in anything I do. I would also like to thank my mentors, Anahita and Sherry, for their guidance throughout the project. Their input helped steer my work in the right direction and challenged me to approach problems more critically.

Thank you for reading my work.

Patrick Kostelac
Delft, June 2025

Contents

Nomenclature

List of Abbreviations

ADP	Approximate Dynamic Programming
AI	Artificial Intelligence
ANN	Artificial Neural Network
CAPS	Consistent Action Policy Smoothing
D-MPC	Distributed Model Predictive Control
DDP	Dynamic Programming
DDPG	Deep Deterministic Policy Gradient
DP	Dynamic Programming
DPG	Deterministic Policy Gradient
DQN	Deep Q-Networks
DRL	Deep Reinforcement Learning
GLA	Gust Load Alleviation
IADP	Incremental Approximate Dynamic Programming
KS	Kreisselmeier-Steinhauser function
LPV	Linear Parameter-Varying
LQR	Linear Quadratic Regulator
LSTM	Long Short-Term Memory
LTI	Linear Time-Invariant
MARL	Multi-Agent Reinforcement Learning
MAV	Micro Aerial Vehicle
MBRL	Model-Based Reinforcement Learning
MC	Monte Carlo
MCTS	Monte Carlo Tree Search
MDP	Markov Decision Processes
MIMO	Multiple Input Multi Output
ML	Machine Learning
MLA	Maneuver Load Alleviation
MOR	Model Order Reduction
MPC	Model Predictive Control
NLP	Nonlinear Programming
NMPC	Nonlinear Model Predictive Control
PID	Proportional-Integral-Derivative
PPO	Proximal Policy Optimization
qlMPC	Quasi-Linear MPC Framework
QP	Quadratic Programming
RL	Reinforcement Learning
RMPC	Robust Model Predictive Control
ROM	Reduced Order Model
SAC	Soft Actor-Critic
SQP	Sequential Quadratic Programming
TD	Temporal Difference
TD3	Twin Delayed Deep Deterministic Policy Gradient
UVLM	Unsteady Vortex Lattice Method
VTOL	Vertical Take-Off and Landing Aircraft
ZOH	Zero-Order Hold

List of Symbols

α	Learning rate for value function updates
$\boldsymbol{a}_t$	Action vector at time step t
$\boldsymbol{s}_t$	State vector at time step t
$\boldsymbol{u}_i$	Predicted control input at step i
$\boldsymbol{u}_k$	Control input at step k
$\boldsymbol{x}_0$	Initial state
$\boldsymbol{x}_i$	Predicted state at step i in the horizon
$\boldsymbol{x}_{\text{other},k}$	State of agents other than i at step k

$\boldsymbol{x}_{N^{\mathsf{P}}}$ Terminal state at the end of prediction horizon

$\boldsymbol{y}(t)$ Continuous-time output

$\boldsymbol{y}_k$ Vector-valued output at time step k

$\delta\boldsymbol{x}$ Deviation from nominal trajectory

Δt Discrete time step

γ Discount factor for future rewards

$\hat{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)$ Approximate system model

$\mathcal{A}$ Action space

$\mathcal{C}$ Set of system constraints

$\mathcal{S}$ State space

$\mathcal{U}_\theta$ Input constraint set parameterized by θ

$\mathcal{X}_\theta$ State constraint set parameterized by θ

$\nabla_\theta J(\theta)$ Policy gradient of expected return with respect to parameters θ

$\pi'(s)$ Greedy policy w.r.t. current value function

$\pi(a \mid s)$ Probability of taking action a in state s under policy π

$\pi^*(s)$ Optimal policy

$\pi^{\mathsf{MPC}}(\boldsymbol{s})$ MPC-derived policy

$\pi_\theta(\boldsymbol{a} \mid \boldsymbol{s})$ Policy parameterized by θ

τ Trajectory tuple in reinforcement learning

Θ Set of admissible policy parameters

θ Policy parameters for RL agent

$\tilde{A}$ Stacked state transition matrix for MPC prediction

$\tilde{B}$ Stacked input matrix for MPC prediction

A Discrete-time state matrix

$A(\boldsymbol{\rho}_k)$ State matrix parameterized by scheduling variables

A^c Continuous-time state matrix

a_t Action at time step t

B Discrete-time input matrix

$B(\boldsymbol{\rho}_k)$ Input matrix parameterized by scheduling variables

B^c Continuous-time input matrix

C Discrete-time output matrix

$C(\boldsymbol{\rho}_k)$ Output matrix parameterized by scheduling variables

C^c Continuous-time output matrix

d_k Disturbance at time step k

E Matrix representing inequality constraints on inputs

$f(\boldsymbol{x}_k, \boldsymbol{u}_k)$ True system dynamics

G Vector representing bounds in input inequality constraints

G_t Return starting from time step t

J Total cost over the prediction horizon

l Stage cost function

N^{C} Control horizon length

N^{P} Prediction horizon length

P Terminal cost weighting matrix

$P(s' \mid s, a)$ Transition probability of reaching state s' from state s after taking action a

Q State weighting matrix in the cost function

$Q(\boldsymbol{s}, \boldsymbol{a})$ Action-value function

$Q^*(s, a)$ Optimal action-value function

$Q_\pi(s, a)$ Action-value function under policy π

R Control weighting matrix in the cost function

r_t Reward at time step t

s_t State at time step t

T Episode length or time horizon

T_s Sampling time used for discretization

$V(\boldsymbol{s})$ State-value function

$V^*(s)$ Optimal state-value function

$V^{\mathsf{f}}(\boldsymbol{x}_{N^{\mathsf{P}}})$ Terminal cost function

$V_\pi(s)$ State-value function under policy π

y_k System output at discrete time k

List of Figures

List of Tables

Part I

Literature Review & Research Definition

1

General Introduction

The evolution of aerospace engineering has led to lightweight, flexible structures aimed at enhancing fuel efficiency and performance [1, 2]. While beneficial, these designs introduce complex challenges in maintaining flight stability and control. aeroelastic wings are particularly prone to aeroelastic phenomena such as flutter, divergence, and control reversal, where the interaction between aerodynamic forces and structural deformations can compromise structural integrity and flight safety [3]. Effectively managing these rapid, nonlinear dynamics remains a critical challenge, requiring control strategies capable of adapting online to the unique, time-varying behavior of aeroelastic structures.

The growing use of aeroelastic aircraft in commercial and research applications has created a need for advanced control strategies capable of managing the unique challenges of aeroelastic behavior [3, 4]. These designs involve complex interactions between structural deformation, unsteady aerodynamics, and nonlinear dynamics, making precise control challenging, especially during rapid state changes and in the presence of external disturbances [1, 5]. Traditional methods such as Linear Quadratic Regulators (LQR) [6, 7] and Model Predictive Control (MPC) [8], rely on linear models, struggle with the coupled, nonlinear dynamics of aeroelastic structures and can break down under large deformations or transonic effects, leading to safety risks [9, 10, 11]. In contrast, Reinforcement Learning (RL) provides a data-driven alternative that can adapt to complex, nonlinear dynamics without precise model linearization [12, 13] but lacks the stability guarantees needed for safety-critical aerospace applications, highlighting the need for innovative control strategies that can address these challenges without compromising performance or safety [3, 14, 4].

MPC and RL have traditionally been used separately, but when combined, they offer a promising approach for aeroelastic aircraft control [15, 16, 17]. MPC provides model-based stability and constraint handling, while RL introduces adaptability for complex, nonlinear dynamics, making the combination well-suited for both predictable behavior and unmodeled disturbances, the main challenges of aeroelastic control [8, 18, 13, 19]. However, existing hybrid methods often either embed MPC within RL [20] or use RL to tune MPC parameters [21], without fully exploiting the complementary strengths of both. These approaches tend to address specific challenges without offering a comprehensive, adaptable solution for various real-world conditions. Given the rapid state changes, model mismatch, and external disturbances typical of aeroelastic wings, there is still significant potential for improving the efficiency, scalability, and robustness of these hybrid controllers. Particularly for real-time, scalable control of highly flexible, gust-sensitive structures, where maintaining stability under unpredictable loads remains a critical challenge [22, 23].

The objective of this thesis is to investigate the strengths and weaknesses of MPC and RL as standalone controllers for aeroelastic aircraft, identify their limitations in handling nonlinear dynamics, rapid state changes, and external disturbances, and then develop a complementary hybrid approach that integrates the advantages of both MPC and RL. This will involve analyzing each method individually to understand their limitations, followed by designing a combined controller that leverages the insights gained from the analysis to integrate the most promising features of both MPC and RL. The resulting hybrid controller should maintain stability and performance across a wide range of conditions, including severe gusts and rapid structural deformations, while remaining computationally efficient for real-time deployment. The thesis is expected to contribute to the development of more robust and adaptable control strategies for aeroelastic aircraft, improving their safety and performance in real-world conditions. By integrating MPC and RL into a

scalable, online framework, this work aims to address critical gaps in existing control methods, potentially enhancing flight stability, reducing operational risks, and supporting the next generation of aeroelastic, high-performance aircraft. While the primary focus is on aeroelastic flight control, the underlying approach may also be extended to other systems with nonlinear dynamics and rapidly changing environments.

2

Literature Review

To address the challenges identified in the previous sections, this literature review examines the theoretical foundations and practical implementations of MPC and RL for aeroelastic aircraft. It first outlines the principles, advantages, and limitations of MPC, followed by a discussion on the role of RL in adaptive control for nonlinear, time-varying systems. The review then explores existing efforts to combine these approaches, highlighting the potential benefits of integrating MPC and RL for enhanced performance, scalability, and robustness. While general hybrid methods have been proposed, few have specifically addressed the unique demands of aeroelastic aerospace structures, underscoring the need for more specialized solutions, which this thesis aims to address.

2.1. Model Predictive Control

This section introduces MPC, an optimization-based control strategy that has gained prominence in applications requiring constraint handling and multivariable control. Unlike classical controllers such as Proportional-Integral-Derivative (PID) or Linear Quadratic Regulators (LQR), MPC explicitly optimizes future control actions over a finite prediction horizon, considering system dynamics, constraints, and disturbances. By solving an optimization problem at each time step, MPC provides real-time adaptability to system changes. The fundamental principle of MPC is to minimize a cost function that is typically made up of state deviations, control effort, and sometimes control smoothness while ensuring compliance with constraints on inputs, outputs, and system states.

MPC is particularly well-suited for systems that require constraint handling, multivariable control, and future prediction. It is widely used in process control, autonomous systems, robotics, power systems, and aerospace applications, where safe and efficient operation depends on the enforcement of actuation limits, state constraints, and trajectory optimization. Unlike traditional controllers that respond reactively to system changes, MPC proactively adjusts control inputs by solving an optimization problem at each time step, allowing it to anticipate disturbances and optimize performance over a defined horizon. Its ability to handle multi-input multi-output (MIMO) systems makes it a powerful choice for complex interconnected dynamics, particularly in applications requiring trajectory planning, energy-efficient actuation, and precision control. However, the computational cost of repeatedly solving optimization problems remains a challenge, especially for high-speed or embedded applications, necessitating efficient solvers and real-time implementation strategies.

This chapter is divided into two sections. The first section, 2.1.1 MPC: Principles and Formulation, details the mathematical foundations of MPC, including its optimization structure, cost function formulation, and constraint handling. It also discusses different classes of MPC, such as Linear, Nonlinear, and Linear Parameter-Varying (LPV) MPC, with an emphasis on LPV approaches for handling nonlinear systems. The second section, 2.1.2 MPC in Aerospace, explores the use of MPC in aerospace applications, summarizing existing research and identifying gaps that motivate further innovation with the focus on the assumptions and simplifications taken in the current research.

2.1.1. MPC: Principles and Formulation

MPC is a class of optimization-based control strategies that utilizes an explicit system model to predict and optimize control inputs over a finite prediction horizon [24]. Unlike classical controllers such as

Proportional-Integral-Derivative (PID) and Linear Quadratic Regulators (LQR), MPC explicitly handles constraints and can accommodate multi-variable interactions, making it particularly advantageous for complex and constrained systems[25].

Cost Function and Objective Formulation
MPC operates by solving an optimization problem at each time step, minimizing a quadratic cost function subject to system dynamics and constraints. The cost function typically penalizes deviations from a desired reference trajectory as well as excessive control effort:

$$J(\boldsymbol{x}_k, \boldsymbol{u}_k) = \sum_{i=k}^{k+N^{\mathrm{p}}-1} \left(\boldsymbol{x}_i^\top Q \boldsymbol{x}_i + \boldsymbol{u}_i^\top R \boldsymbol{u}_i\right) + \boldsymbol{x}_{k+N^{\mathrm{p}}}^\top P \boldsymbol{x}_{k+N^{\mathrm{p}}} \tag{2.1}$$

where $\boldsymbol{x}_i$ represents the system state, $\boldsymbol{u}_i$ the control input, and Q, R, and P are positive semi-definite weight matrices that penalize deviations in the state, control effort, and terminal state, respectively. Only the first optimal control input is applied at each iteration, and the procedure is repeated at the next time step in a receding horizon fashion.

A key distinction between MPC and traditional control methods lies not just in the use of a model, but in how that model is utilized within the control framework. Traditional control methods compute control actions based solely on the current error signals, without explicitly considering how the system will evolve over time. This structure is illustrated in Figure 2.2, where the controller reacts to the present state without forecasting future behavior. In contrast, as shown in Figure 2.1, MPC incorporates an internal model within an optimization block that predicts future system behavior and determines the control inputs over a specified prediction horizon. At each time step, MPC simulates the effect of a sequence of control inputs on the model to evaluate how the system is expected to respond. Based on this prediction, it selects the optimal control sequence, but only the first control input is applied to the real system. This predictive optimization allows MPC to anticipate future events, enforce constraints directly, and apply corrective actions before deviations occur, thereby improving overall system performance.

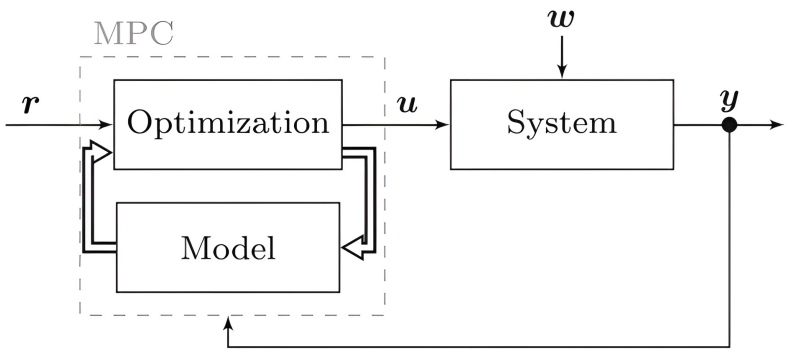


Figure 2.1: Simplified block diagram of an MPC-based control loop. The optimization block computes control inputs based on the predictive model. Symbols: r is the reference signal, u the control input, y the system output, and w external disturbances.

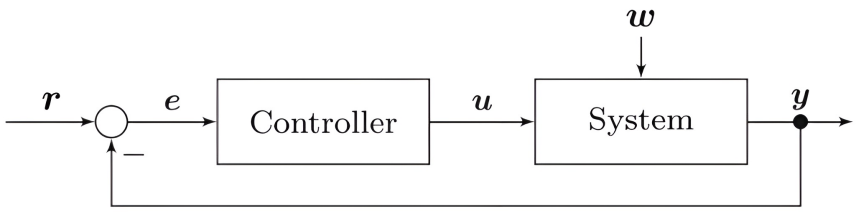


Figure 2.2: Standard feedback control loop. The control action is computed based only on current error signals.

Predictive Models and State-Space Representation
MPC relies on a predictive model of the system, often described using a discrete-time state-space representation:

$$x_{k+1} = Ax_k + Bu_k, \quad y_k = Cx_k, \tag{2.2}$$

where A, B, and C define the system dynamics, control influence, and measured outputs, respectively. Given this model, the future system states can be predicted over the control horizon N_p using recursion:

$$\mathbf{x} = \tilde{A}x_k + \tilde{B}\mathbf{u}, \tag{2.3}$$

where **x** is a stacked vector containing future states, and **u** contains future control inputs. The matrices $\tilde{A}$ and $\tilde{B}$ are constructed as follows:

$$\tilde{A} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N_p} \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1}B & A^{N_p-2}B & \dots & B \end{bmatrix}. \tag{2.4}$$

This formulation allows the prediction of system trajectories over the horizon and facilitates the formulation of the optimization problem [8].

The control input sequence is computed by minimizing the following quadratic cost function [8]:

$$J(x_k, \mathbf{u}) = \mathbf{x}^T Q_N \mathbf{x} + \mathbf{u}^T R_N \mathbf{u}, \tag{2.5}$$

where Q_N and R_N are block diagonal matrices constructed from Q and R. The control inputs must satisfy input and state constraints, which can be expressed as:

$$E\mathbf{u} \leq G, \tag{2.6}$$

where E and G define the feasible control region. These constraints ensure that the computed control actions remain within physically realizable limits.

Linear vs Nonlinear MPC
MPC can be broadly categorized into linear MPC and nonlinear MPC (NMPC), depending on the nature of the system model. Linear MPC assumes that the system dynamics can be approximated by a linear time-invariant (LTI) models, which simplify the control problem significantly. In this setting, a quadratic programming (QP) problem is solved at each time step to minimize a cost function of the form previously defined in (2.1). This cost function penalizes both state deviations and control effort over a finite prediction horizon using weight matrices Q, R, and a terminal weight P. Linear MPC is computationally efficient and well-suited for systems where linear models provide a sufficiently accurate approximation over the prediction horizon.

While linear MPC assumes that the system dynamics can be approximated by a LTI model, many real-world systems, particularly in aerospace applications, exhibit strong nonlinearities arising from aerodynamic forces or structural flexibility. In such cases, the assumption of linearity can lead to suboptimal or even unstable control performance. To account for these effects, we consider systems governed by nonlinear dynamics of the form:

$$\boldsymbol{x}_{i+1} = \boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{u}_i), \tag{2.7}$$

where $\boldsymbol{f}(\cdot)$ captures the nonlinear system behavior. Importantly, we retain the quadratic cost structure introduced in (2.1), as it enables computationally efficient optimization formulations. This can be taken a step further by solving a fully nonlinear problem, in which both the dynamics and the cost function are nonlinear. However, this approach results in a nonlinear programming (NLP) problem at each time

step, which incurs a high computational burden and may be unsuitable for real-time applications. Instead, we focus on controlling systems with nonlinear dynamics using a quadratic cost, and seek alternative formulations that preserve tractability while capturing the essential nonlinear behavior.

NMPC has been successfully applied in various industries, including **chemical processes**, **robotics**, and **aerospace**, where accurate handling of nonlinear dynamics is crucial for stability and performance [26]. However, the increased computational burden and the potential for local minima make NMPC challenging to implement in real-time systems.

This challenge is particularly relevant for onboard aerospace applications, where computational resources are limited. Real-time execution requires that optimization problems be solved within tight timing constraints, making the high computational cost of NMPC a significant drawback. As a result, many practical implementations favor linear or LPV-based MPC formulations, which offer a compromise between model accuracy and feasibility for embedded systems.

Linear Parameter Varying MPC

To handle nonlinearities efficiently while maintaining computational tractability, **Linear Parameter-Varying (LPV) MPC** is often used. LPV-MPC dynamically updates linearized models around the current operating point, providing a trade-off between computational efficiency and model accuracy. This approach constructs a set of linear models that approximate the nonlinear system behavior over different operating regions, allowing the controller to adapt as the system state changes [11].

An LPV system can be represented as a linear system with parameters that vary over time, often written in the form:

$$\boldsymbol{x}_{k+1} = A(\boldsymbol{\rho}_k)\boldsymbol{x}_k + B(\boldsymbol{\rho}_k)\boldsymbol{u}_k, \quad \boldsymbol{y}_k = C(\boldsymbol{\rho}_k)\boldsymbol{x}_k, \tag{2.8}$$

where $A(\boldsymbol{\rho}_k)$, $B(\boldsymbol{\rho}_k)$, and $C(\boldsymbol{\rho}_k)$ are parameter-varying system matrices, and $\boldsymbol{\rho}_k$ is a vector of scheduling parameters that capture the current operating conditions. These parameters are typically chosen to reflect key nonlinear aspects of the system dynamics, such as the angle of attack in an aircraft or the flow rate in a chemical reactor.

The cost function for LPV-MPC retains the quadratic structure defined in (2.1), but is evaluated using a parameter-varying system model. Specifically, the system dynamics at each step are governed by a locally linearized model of the form:

$$\boldsymbol{x}_{k+1} = A(\boldsymbol{\rho}_k)\boldsymbol{x}_k + B(\boldsymbol{\rho}_k)\boldsymbol{u}_k. \tag{2.9}$$

This local linearization allows LPV-MPC to handle moderate nonlinearities without incurring the full computational overhead of NMPC. However, it relies on accurate parameter scheduling to ensure that the linear approximations remain valid across the entire operating range [11].

It is important to distinguish LPV-MPC from gain-scheduled MPC. While both use scheduling parameters to account for changing operating conditions, gain-scheduled MPC typically relies on a predefined set of linear controllers, each tuned for a specific operating point. The controller switches between these based on the current value of the scheduling variable. In contrast, LPV-MPC updates the model continuously and solves a new optimization problem at each time step using the current parameter values. This allows LPV-MPC to adapt more smoothly and flexibly to nonlinear behavior without relying on discrete controller switching.

LPV-MPC has been successfully applied in controlling complex systems such as **autonomous underwater vehicles**, **chemical reactors**, and **aeroelastic aircraft**, demonstrating its versatility and effectiveness [27]. By switching or interpolating between these models during operation, LPV-MPC can effectively manage nonlinearities with reduced computational burden compared to fully nonlinear approaches.

Discrete vs. Continuous-Time MPC

MPC can be implemented in both discrete-time and continuous-time domains, depending on the nature of the system and the desired control performance. In practice, **Discrete-Time MPC** is more commonly used, particularly in digital control applications where the controller operates at fixed sampling intervals. This approach relies on a discretized version of the system model, which can be expressed as:

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k, \quad \boldsymbol{y}_k = C\boldsymbol{x}_k, \tag{2.10}$$

where A, B, and C are the discrete-time system matrices, typically obtained from a continuous-time model using a zero-order hold (ZOH) or other discretization methods. The relationship between the continuous-time and discrete-time system matrices is given by:

$$A = e^{A^{\mathsf{c}} T_s}, \tag{2.11}$$

$$B = \int_0^{T_s} e^{A^{\mathsf{c}} \tau} B^{\mathsf{c}} \, d\tau, \tag{2.12}$$

where T_s is the sampling time, and A^{c}, B^{c} are the continuous-time system matrices. The matrix exponential $e^{A^{\mathsf{c}} T_s}$ captures the effect of continuous-time dynamics over a discrete sampling period.

In contrast, **Continuous-Time MPC** operates directly on the continuous-time system equations:

$$\dot{\boldsymbol{x}}(t) = A^{\mathsf{c}}\boldsymbol{x}(t) + B^{\mathsf{c}}\boldsymbol{u}(t), \quad \boldsymbol{y}(t) = C^{\mathsf{c}}\boldsymbol{x}(t), \tag{2.13}$$

where the control actions are computed continuously over time. While this approach can offer more accurate representations of high-speed dynamics, it is computationally more intensive and often requires more sophisticated solvers, such as differential dynamic programming (DDP) or direct collocation methods.

The cost function for continuous-time MPC is typically formulated as:

$$J(\boldsymbol{x}(0), \boldsymbol{u}(\cdot)) = \int_0^T \left(\boldsymbol{x}(t)^\top Q \boldsymbol{x}(t) + \boldsymbol{u}(t)^\top R \boldsymbol{u}(t) \right) dt, \tag{2.14}$$

where the integral accounts for the continuous nature of the system. In contrast, discrete-time MPC uses a summation over a fixed prediction horizon, as shown in (2.1), highlighting a key difference in how the control objectives are defined and optimized.

Despite the theoretical advantages of continuous-time MPC, most practical implementations involve discretizing the system model to facilitate numerical computations on digital platforms. This process can introduce approximation errors and increase computational complexity, as the continuous dynamics must be accurately captured within the discrete framework. For this reason, many industrial MPC applications rely on discrete-time formulations, which are directly supported by widely used control software packages such as MATLAB's Model Predictive Control Toolbox [28].

2.1.2. MPC in Aerospace

MPC has been widely applied in aerospace for tasks ranging from flight control to disturbance rejection and structural load alleviation. Its predictive capabilities and constraint handling capabilities make it well suited for aerospace applications, where safety, performance, and robustness are paramount. Key applications include gust load alleviation (GLA) and maneuver load alleviation (MLA) to reduce structural loads from atmospheric disturbances and aggressive maneuvers, vibration suppression to mitigate aeroelastic instabilities in aeroelastic aircraft, and flight control for trajectory tracking and stability enhancement. Additionally, MPC is used in some fault tolerance cases, where the controller performance needs to be maintained under actuator or system failures. As all of these are real-time implementations the computational feasibility of MPC is also a topic of high interest. These implementations often rely on simplifying assumptions regarding system dynamics, disturbances, and computational constraints. The following sections reviews these primary areas where MPC has been employed in aerospace, highlighting the approaches taken to tackle them.

GLA and MLA

One of the major challenges in aircraft control is mitigating gust-induced structural loads, which can significantly impact passenger comfort, structural integrity, and aircraft performance. MPC has been widely used in GLA systems, leveraging its predictive nature to proactively counteract gust disturbances. Many studies employ LPV MPC, where the controller adapts dynamically to variations in flight conditions, structural deformations, and aerodynamic effects [11]. The response errors between the LPV and the nonlinear models remain small, making it reasonable to use the LPV model to represent nonlinear dynamics [29]. Some implementations integrate real-time gust preview data from sensors such as LIDAR or nose-mounted probes to enhance disturbance rejection by allowing the controller to anticipate incoming gusts before they impact the aircraft [30]. The integration of unsteady aerodynamic modeling has played a crucial role in improving GLA strategies. Some studies employ high-fidelity aerodynamic solvers such as the Unsteady Vortex Lattice Method (UVLM), which effectively captures the dynamic response of aeroelastic aircraft under gust disturbances [31]. Additionally, model order reduction (MOR) techniques, including balanced truncation and oblique projection, are often used to retain the dominant aeroelastic modes while simplifying computational complexity [11]. The use of time-varying Kalman filters to estimate unmeasured states has further improved the real-time applicability of these methods [29].

Many GLA strategies assume that the environment is static and that future gust disturbances can be accurately measured within the prediction horizon [30] [29] and that the aerodynamic response to these disturbances is sufficiently captured by reduced-order linearized models [31]. There are a lot of additional assumptions on the gust itself, some papers simplify gusts and assume them to be uniform [31] [11]. As well as the assumptions on the gusts, there are also assumptions on the deformations caused by the gusts, deformations are sometimes based only on the steady state and ignore the transient behavior [11]. Additionally, the interaction between flexible structures and aerodynamics is often simplified, limiting the ability of these controllers to address highly nonlinear effects and strong gust interactions [11].

MPC is also extensively used for manoeuvre load alleviation (MLA) to reduce structural stresses induced by aggressive manoeuvres. This is particularly relevant for aeroelastic aircraft, where load management is critical to ensuring structural longevity. MPC-based MLA frameworks typically regulate control surface deflections to redistribute aerodynamic loads, reducing peak structural stress without compromising handling performance [32].Some studies implement distributed MPC (D-MPC) for MLA, where multiple local controllers work in parallel to optimize load distribution across the aircraft[10]. This decentralized control architecture allows for scalability and modularity, making it suitable for large, aeroelastic airframes. Additionally, state-dependent MPC formulations have been introduced to enhance adaptability by adjusting control gains based on the instantaneous structural state [32].

While MLA strategies are effective for reducing maneuver-induced loads, some studies highlight the potential implications of neglecting gust disturbances during maneuvers. One study emphasizes the reduction of static loads through MLA strategies but acknowledges that while MLA effectively mitigates maneuver-induced loads, it may inadvertently increase the impact of dynamic gust loads on the final design, underlining the importance of considering gust effects in load alleviation strategies [33]. In contrast, another study investigates the combined effects of gusts and maneuvers on aircraft loads, exploring the use of coordinated deflections of outer ailerons and spoilers to alleviate both gust and maneuver loads. This approach demonstrates the necessity of control strategies that address both factors simultaneously,

ensuring that load alleviation measures do not inadvertently introduce new challenges under varying flight conditions [34].

Many studies also adopt quasi-static load redistribution models, neglecting transient aeroelastic effects that might introduce additional challenges in real-world conditions[32] [10]. Importantly, many MLA strategies are designed under the assumption that loads are solely maneuver-induced, without explicitly accounting for gust disturbances that may occur simultaneously with a maneuver [32] [10] [33]. Since these approaches do not incorporate GLA techniques, the same challenges associated with gust load alleviation also persist in these MLA implementations, further limiting their robustness in realistic flight scenarios. These can also lead to underestimated load predictions and potentially insufficient control actions if unexpected gusts increase structural stresses beyond the anticipated maneuver-induced loads [34].

Vibration Suppression
The increasing trend towards lightweight, high-aspect-ratio wings has introduced new challenges related to aeroelastic vibrations, which can degrade structural integrity and flight performance. MPC has been extensively employed to mitigate these vibrations by actively adjusting control inputs based on predicted structural responses [11]. Many implementations leverage reduced-order models (ROMs) to approximate the aeroelastic dynamics while maintaining computational efficiency [29]. These ROMs, often derived using modal truncation techniques, aim to capture the dominant aeroelastic modes while filtering out less significant dynamics. However, simplifications in the modeling process can lead to inaccuracies in real-time implementations.

Some studies enhance MPC-based vibration suppression by incorporating Kalman filters to estimate unmeasured structural states, ensuring robust performance under varying flight conditions [31]. Additionally, Linear Parameter-Varying (LPV) MPC formulations have been introduced to account for changing structural characteristics due to different flight regimes and external disturbances [35]. These controllers dynamically adjust their parameters based on real-time flight conditions, improving adaptability while still maintaining computational feasibility. Moreover, recent approaches have integrated smooth-switching LPV control techniques to enhance vibration suppression by reducing abrupt control transitions between different operating conditions, thereby improving stability and robustness in highly aeroelastic aircraft [35]. Additionally, certain control approaches assume that scheduling parameters, can be measured in real-time without uncertainty, neglecting potential sensor noise or delays that could affect controller performance [35].

Many vibration control strategies assume that wing deformations can be effectively captured using modal approximations, which focus on dominant structural modes while neglecting higher-order or nonlinear interactions [11]. Furthermore, some methods assume uniform gust distributions across the aircraft, simplifying turbulence effects but potentially misrepresenting localized aerodynamic interactions [35]. Whereas some approaches assume that the environment is static and that it can be measured by LIDAR and thus predicted [29].

Flight Control
MPC has been widely applied in flight control to enhance trajectory tracking, attitude stabilization, and fault-tolerant capabilities. Unlike conventional controllers such as PID or LQR, MPC explicitly accounts for input and state constraints, making it particularly suitable for aerospace applications where actuator saturation and operational limits must be strictly enforced. Many flight control implementations utilize Linear Time-Invariant (LTI) or LPV models to balance computational feasibility with accuracy, allowing controllers to adapt dynamically to changing flight conditions [36].

Several studies have demonstrated the effectiveness of MPC in different flight control contexts. It can be particularly useful with highly aeroelastic aircraft where structural deformations should also be minimized. One approach applies real-time MPC for controlling unmanned aeroelastic aircraft, showcasing its capability to improve stability and disturbance rejection despite the presence of structural deformations and aerodynamics-induced uncertainties. The study highlights the practical implementation of MPC in flight tests and discusses the computational challenges associated with real-time operation [37]. Another study explores MPC for aeroelastic aircraft dynamics, focusing on the use of nonlinear reduced-order models to capture structural deformations while maintaining computational efficiency. This method allows controllers to anticipate structural oscillations and adjust control inputs accordingly, significantly improving handling qualities in highly aeroelastic aircraft [36].

MPC has also been applied to fault-tolerant flight control, particularly in cases of severe actuator failures. A case study on El Al Flight 1862 demonstrates how an MPC-based reconfiguration strategy could have helped prevent the fatal crash by redistributing available control authority to compensate for actuator failures. This study employs a reference model-based approach, where MPC continuously updates its internal model based on fault-detection inputs, allowing the aircraft to maintain stable flight despite extensive control surface failures [38]. A similar MPC framework is proposed for tiltrotor eVTOL aircraft, incorporating data-driven control to improve performance and adaptively compensate for the unmodeled dynamics. This approach leverages both model-based and data-driven predictive control strategies, showing promising results in increasing fault tolerance while ensuring stability in novel aircraft configurations [39].

Several studies have focused on reducing the computational burden of MPC for flight control, particularly in applications with large-scale constrained optimization problems. One approach introduces constraint aggregation using the Kreisselmeier-Steinhauser (KS) function, which replaces multiple individual constraints with a smooth, nonlinear approximation. This method reduces the number of constraints explicitly handled in the optimization process, making it easier to solve in real-time using Sequential Quadratic Programming (SQP). While this approach can significantly decrease computational cost and allow for longer prediction horizons, it also introduces a trade-off—overly conservative constraints may limit controller performance more than necessary [40]. Another approach improves computational efficiency by using Laguerre orthonormal basis functions to approximate control trajectories, which simplifies the MPC formulation for aeroelastic aircraft. By integrating this method into a quasi-linear MPC framework (qLMPC) with a Kalman filter, this approach enables accurate reference tracking while reducing overall computing time by a factor of approximately 8 compared to traditional MPC implementations. While the Laguerre-based method can improve tracking performance, it also introduces a trade-off—controllers designed with strict stability guarantees may be less efficient than those that prioritize computational speed [41].

Many MPC-based flight control implementations assume linearized or quasi-linear system representations, which may not fully capture the nonlinear aerodynamic and structural interactions present in real-world flight conditions. This is particularly relevant in aeroelastic aircraft dynamics, where nonlinearities significantly affect the control response [36]. Despite the heavy reliance on the linearized systems not many papers study the effect of model mismatch. During strong gusts and fast maneuvers high accelerations can cause significant structure deformations which leads to large model mismatch during which the controllers were found to perform poorly [40]

Additionally, some methods rely on accurate aerodynamic models, which may not generalize well across varying flight regimes especially when the fault tolerant control is in question, this can lead to potential control performance degradation under unmodeled disturbances [38]. Certain studies assume perfect fault detection and identification, meaning that control reconfiguration is based on real-time, accurate information about actuator failures. However, this may not always be the case in practice, as fault diagnosis and estimation errors could lead to incorrect control actions [38]. Some approaches also neglect measurement uncertainties, assuming perfect state estimation with no sensor noise or time delays, which may not hold in real-world conditions where sensor inaccuracies impact control performance [37].

MPC implementations for very aeroelastic aircraft assume that constraint aggregation using Kreisselmeier-Steinhauser (KS) functions sufficiently preserves the feasible control region, despite the fact that these aggregations introduce conservatism and may eliminate optimal control actions that could have been feasible with a full constraint set. This introduces a trade-off between computational efficiency and controller performance [40]. Additionally, some control strategies assume that reducing the number of constraints will not significantly affect stability, implicitly assuming that the controller's stability and effectiveness remain intact. However, this may not hold in cases where strongly coupled aeroelastic effects dominate the aircraft dynamics, potentially leading to overly conservative control actions or loss of performance [41]. Additionally, most of the MPC formulations assume that disturbance effects, such as gusts or modeling errors in transition-phase dynamics, can be adequately handled by simple predefined disturbance models. For example, disturbances in [39] are modeled as an external vertical acceleration of the form:

$$d(t) = M\frac{1-\cos(2\pi t/p)}{2} \tag{2.15}$$

where M and p denote the magnitude and period of the disturbance. This simplified approach assumes that disturbances can be represented by periodic or well-characterized acceleration profiles, which may

not be representative of real-world turbulent conditions or more complex unsteady aerodynamic effects. As a result, control schemes based on such assumptions may fail to compensate for highly nonlinear or rapidly changing disturbances, leading to degraded tracking performance and stability in operational environments.

2.1.3. Conclusion

The current research on MPC-based control in aerospace highlights its applications in GLA, MLA, vibration suppression, and flight control. Across these domains, MPC has been adopted due to its ability to handle constraints and predict system behavior over a finite horizon. Many implementations employ LPV-MPC to adapt to changing flight conditions, structural deformations, and aerodynamic effects [11]. LPV-MPC also provides a practical balance between the simplicity of linear MPC and the high computational demands of full nonlinear MPC, making it well-suited for onboard applications with limited processing capacity. Some research integrates real-time state estimation techniques such as Kalman filtering to compensate for unmeasured disturbances [31], and some employ reduced-order models (ROMs) to retain computational efficiency while capturing dominant aeroelastic behaviors [29]. Despite these advancements, many existing approaches rely on assumptions that limit their applicability in broader aerospace scenarios.

One of the major limitations is that most studies focus on a single objective, either reference tracking, load alleviation, or vibration suppression, without addressing how these goals interact in a unified control framework. For instance, while MPC-based MLA methods effectively redistribute loads to minimize peak structural stresses [32], they often do not account for simultaneous gust disturbances that could exacerbate structural loads [33]. On the other hand some GLA approaches do not incorporate MLA strategies, potentially leading to conflicting control objectives when both disturbances and maneuvers occur simultaneously [34]. Some GLA approaches rely on previewed gust information from LIDAR [30] and assume perfect knowledge of the upcoming gusts which is not a realistic scenario as the environment is dynamic rather than static.

Disturbance modeling is a significant limitation in many studies. Many studies only consider simplified disturbances, often assuming uniform gusts or predefined periodic profiles [31] [11]. Some approaches assume perfect knowledge of future disturbances through LIDAR-based sensing [30], which may not be reliable in all operational conditions. With realistic turbulence models such as Dryden or Von Kármán turbulence models, which introduce stochastic and multi-scale disturbance effects, being often neglected, limiting the robustness of these controllers in real-world conditions [29].

Actuator delay is also frequently overlooked. Most MPC implementations assume instantaneous actuation, overlooking the time lag between command execution and response. However, even small delays can degrade control performance, especially in fast-dynamic aerospace systems. Studies in other domains have shown that actuator delay can significantly impact stability and tracking accuracy [42], yet it remains largely unaddressed in aerospace MPC research. Without compensating for this delay, controllers risk generating suboptimal inputs.

Model mismatch presents an additional challenge. Many controllers assume that the underlying system dynamics remain constant and accurately modeled, disregarding variations due to environmental conditions, structural changes, or damage. However, real-world aerospace systems experience model deviations due to factors such as icing, changing altitudes, or structural degradation over time [40]. Without incorporating robust adaptation mechanisms or considering potential model mismatch, these MPC implementations may fail to maintain performance in off-nominal conditions.

Although MPC has been demonstrated to work effectively for many specific scenarios, there has yet to be a generalized framework that can simultaneously handle multiple objectives, account for realistic disturbances, and adapt to model uncertainties. Future research must address these limitations to develop more robust and universally applicable MPC strategies for aerospace applications, ensuring enhanced reliability and performance across diverse flight conditions.

2.2. Reinforcement Learning

This section introduces RL, a learning-based control strategy that has gained prominence in applications requiring adaptive decision-making in uncertain and high-dimensional environments. Unlike classical control methods such as PID, LQR, or MPC, RL does not necessarily require an explicit system model and instead learns optimal policies through interaction with the environment. By leveraging experience, RL can adapt to changing conditions, optimize performance over time, and handle complex state-action relationships that may be difficult to model explicitly.

RL is particularly well-suited for applications where the system dynamics are highly nonlinear, uncertain, or difficult to model accurately. It is widely used in robotics, autonomous systems, aerospace, power systems, and various real-time decision-making tasks. Unlike traditional controllers that rely on pre-defined rules or optimization-based receding horizon planning, RL-based controllers learn optimal behaviors by maximizing cumulative rewards over sequential interactions. This makes RL a powerful tool for complex control problems, including trajectory optimization, adaptive flight control, and autonomous decision-making. However, RL faces challenges such as high sample complexity, stability concerns, and safety-critical constraints, which require careful formulation and robust training methodologies to ensure reliable deployment.

This chapter is divided into two sections. The first section, 2.2.1 RL: Principles and Formulation, details the mathematical foundations of RL, including Markov Decision Processes (MDPs), value functions, policy optimization, and model-based versus model-free approaches. The second section, 2.2.2 RL in Aerospace, explores the use of RL in aerospace applications, summarizing existing research and identifying gaps that motivate further innovation.

2.2.1. RL: Principles and Formulation

Fundamentals of Reinforcement Learning

RL is a framework for sequential decision-making, where an agent interacts with an environment to maximize cumulative rewards over time. Unlike traditional control methods that rely on explicit system models, RL enables learning from experience, making it particularly useful in complex and uncertain environments.

The core idea of RL is to allow an agent to associate rewards with actions that contribute to achieving a goal. The agent is not explicitly told what the goal is but instead learns by executing actions, observing state transitions, and receiving feedback in the form of rewards. This learning mechanism resembles natural trial-and-error learning, similar to how a child learns to walk or an animal learns conditioned behavior. RL has been successfully applied to high-dimensional tasks, such as playing board games at a superhuman level and controlling robotic arms to mimic human dexterity [43, 44].

A typical reinforcement learning (RL) system consists of:

- **Agent**: The decision-making entity.
- **Environment**: The system in which the agent operates.
- **State** (s_t): The representation of the environment at time t, where $s_t \in \mathcal{S}$.
- **Action** (a_t): A choice made by the agent at time t, where $a_t \in \mathcal{A}$, that affects the environment.
- **Reward** (r_t): A scalar signal received at time t, indicating how beneficial the previous action was.
- **Policy** (π): A strategy that maps states to actions, i.e., $\pi : \mathcal{S} \to \mathcal{A}$.

The interaction between the agent and the environment in reinforcement learning is typically modeled as a **Markov Decision Process (MDP)**. In this framework, the agent observes the current state s_t, selects an action a_t according to its policy, and receives a reward r_t from the environment. The environment then transitions to a new state s_{t+1}, which the agent uses to refine its decision-making strategy. This ongoing cycle is visually represented in Figure 2.3, which illustrates the flow of information between the agent and the environment [45].

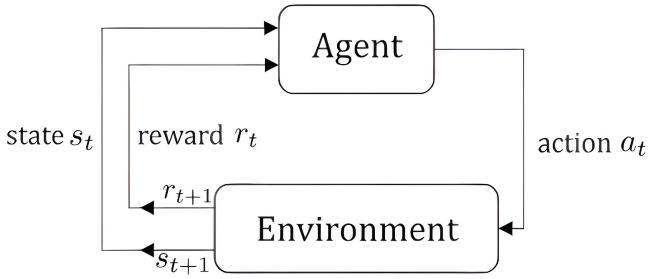


Figure 2.3: Agent-Environment Interaction in Reinforcement Learning. The agent selects an action a_t based on the current state s_t, receives a reward r_t, and transitions to a new state s_{t+1}. This interaction follows the MDP framework. Adapted from [45]

This process can be captured mathematically as a Markov Decision Process (MDP), which is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$:

- $\mathcal{S}$ is the set of possible states.
- $\mathcal{A}$ is the set of possible actions.
- $P(s' \mid s, a)$ is the transition probability function, which defines the likelihood of moving from state s to state s' after taking action a.
- $r(s, a)$ is the reward function, providing scalar feedback for each state-action pair.
- γ is the discount factor ($0 \leq \gamma \leq 1$) that determines the importance of future rewards.

One of the key properties of MDPs is the **Markov property**, which states that the future state s_{t+1} depends only on the current state s_t and action a_t, and not on any past states or actions. This memoryless property simplifies the decision process and allows RL algorithms to make predictions and optimize policies efficiently [45]:

$$P(s_{t+1}, r_t \mid s_t, a_t) = P(s_{t+1}, r_t \mid s_1, s_2, \ldots, s_t, a_1, a_2, \ldots, a_t). \tag{2.16}$$

The sequence of interactions between the agent and the environment is recorded as a **trajectory** τ, which consists of a chain of state-action-reward-next state tuples over time [45]:

$$\tau = (s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_T, a_T, r_T). \tag{2.17}$$

A transition in reinforcement learning can be either **deterministic** or **stochastic**. In a deterministic process, the next state is uniquely determined by the current state and action [45]:

$$s_{t+1} = f(s_t, a_t). \tag{2.18}$$

In contrast, in a stochastic process, the next state is sampled from a probability distribution:

$$s_{t+1} \sim p(s_{t+1} \mid s_t, a_t). \tag{2.19}$$

A sequence of transitions forms a **trajectory** or **episode**, which represents a complete interaction from an initial to a terminal state, such as an entire game session.

A fundamental aspect of reinforcement learning is the reward signal, which serves as the primary measure of an agent's performance. The reward is designed to incentivize behavior that leads to desirable outcomes. The cumulative sum of rewards over time is known as the **return** G_t [12]:

$$G_t = G(\boldsymbol{s}_{t:T-1}, \boldsymbol{a}_{t:T-1}) = r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \cdots + \gamma^{T-t-1} r(s_{T-1}, a_{T-1}). \tag{2.20}$$

The discount factor $\gamma \in [0, 1]$ determines how much future rewards contribute to the return. A higher γ makes the agent more "far-sighted," prioritizing long-term rewards, while a lower γ makes the agent "myopic," focusing on immediate gains.

The agent's decision-making process in reinforcement learning is governed by a **policy** $\pi(a \mid s)$, which defines the probability of selecting action a given state s [12]:

$$\pi(a \mid s) = P(a_t = a \mid s_t = s) \tag{2.21}$$

In this formulation, we use lowercase notation a_t and s_t to refer to both random variables and their realizations when the distinction is clear from context, as is common in reinforcement learning literature. Policies can be either **deterministic**, where a specific action is always chosen for a given state, or **stochastic**, where actions are selected based on probability distributions. The primary objective of reinforcement learning is to optimize the policy π such that it maximizes the expected return over time.

The Bellman Equation and Value Functions

Value functions are central to reinforcement learning, as they quantify the expected cumulative rewards an agent can obtain from a given state or state-action pair. These functions serve as the foundation for optimizing policies and predicting future rewards.

The **state-value function** $V_\pi(s)$ represents the expected return when starting from a state s and following a policy π [45]:

$$V_\pi(s) = \mathbb{E}_\pi \left[G_t \mid s_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right]. \tag{2.22}$$

The **action-value function** $Q_\pi(s, a)$ extends this definition by considering the expected return when taking a specific action a in state s, and then following the policy π thereafter [45]:

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid s_t = s, a_t = a \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right]. \tag{2.23}$$

These functions are critical for evaluating the long-term desirability of different states and actions, guiding decision-making in RL algorithms.

A fundamental property of value functions in reinforcement learning is that they satisfy recursive relationships, known as the **Bellman equations**. These equations express that the value of a state or state-action pair can be decomposed into immediate rewards and the discounted value of successor states.

For a given policy π, the Bellman equation for the state-value function $V_\pi(s)$ is derived as follows [12]:

$$\begin{aligned}
V_\pi(s) &= \mathbb{E}_\pi \left[G_t \mid s_t = s \right] && (2.24)\\
&= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right] && (2.25)\\
&= \mathbb{E}_\pi \left[r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s \right] && (2.26)\\
&= \mathbb{E}_\pi \left[r_{t+1} + \gamma V_\pi(s_{t+1}) \mid s_t = s \right] && (2.27)\\
&= \sum_{s', r} p(s', r \mid s, \pi(s)) \left[r + \gamma V_\pi(s') \right] && (2.28)
\end{aligned}$$

This recursive relationship states that the value of a state is equal to the immediate reward plus the discounted expected value of the next state.

Similarly, the Bellman equation for the action-value function $Q_\pi(s, a)$ is [12]:

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a \right]. \tag{2.29}$$

These equations form the backbone of many reinforcement learning algorithms, enabling iterative value updates.

The goal of reinforcement learning is to find an optimal policy π^* that maximizes expected returns. The corresponding optimal value functions satisfy the **Bellman optimality equations** [45]:

$$V^*(s) = \max_{a \in \mathcal{A}} \{\mathbb{E}\left[r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a\right]\} \tag{2.30}$$

$$Q^*(s,a) = \mathbb{E}\left[r_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a\right] \tag{2.31}$$

These equations form the basis of control algorithms including Q-learning, which iteratively refines estimates to approximate the optimal policy.

In practical RL algorithms, the **Bellman equation** is used to iteratively update value function estimates. A commonly used update rule is given by [45]:

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right], \tag{2.32}$$

where α is the learning rate that controls the step size of updates. This formulation is widely used in temporal difference (TD) learning and serves as the foundation for many RL methods. The Bellman equation serves as a fundamental building block for RL algorithms, enabling agents to evaluate actions, optimize policies, and make long-term decisions.

The **Bellman equation** is central to optimizing policies in reinforcement learning. By recursively estimating values, RL agents can evaluate state or action quality by estimating expected future rewards, refine policies iteratively through value-based methods, and improve decision-making by leveraging recursive updates. Many RL algorithms, including Dynamic Programming (DP), Monte Carlo (MC), and Temporal Difference (TD) methods, use these recursive formulations to update policies and enhance learning efficiency. The Bellman equation remains a key component in reinforcement learning research, enabling effective policy evaluation and control.

Dynamic Programming
Dynamic Programming (DP) is a class of algorithms used to solve MDPs when the full model of the environment is known. DP methods leverage the Bellman equation to iteratively compute value functions and improve policies in a structured manner. These methods provide exact solutions to MDPs under certain assumptions, but their applicability is limited due to computational constraints in large-scale problems.

DP approaches solve MDPs by systematically computing value functions and improving policies based on recursive updates. Given a known transition model $P(s' \mid s, a)$ and reward function $r(s,a)$, DP algorithms iteratively update the value function until convergence. The fundamental idea behind DP is to decompose the problem into smaller subproblems, solving them optimally in a bottom-up manner. The two main DP-based approaches for solving MDPs are **Value Iteration** and **Policy Iteration**.

Value Iteration is an approach that directly computes the optimal value function by iteratively applying the Bellman optimality equation [45]:

$$V^*(s) = \max_{a \in \mathcal{A}} \sum_{s'} P(s' \mid s, a) \left[r(s,a) + \gamma V^*(s')\right]. \tag{2.33}$$

The algorithm initializes $V(s)$ arbitrarily and updates it iteratively until convergence. Once $V^*(s)$ stabilizes, an optimal policy can be extracted by selecting actions that maximize expected returns:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} \sum_{s'} P(s' \mid s, a) \left[r(s,a) + \gamma V^*(s')\right]. \tag{2.34}$$

Value Iteration is computationally efficient for small MDPs but becomes intractable as the state space grows due to the need to update every state in each iteration.

Policy Iteration is a dynamic programming method that alternates between two key steps [45]:

1. **Policy Evaluation**: Compute the value function $V_\pi(s)$ for the current policy π:

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) \left[r(s,a) + \gamma V_\pi(s')\right]. \tag{2.35}$$

2. **Policy Improvement**: Update the policy by selecting actions that maximize expected returns:

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} \sum_{s'} P(s' \mid s, a) \left[r(s, a) + \gamma V_\pi(s') \right]. \tag{2.36}$$

These steps are repeated until the policy stabilizes, ensuring convergence to an optimal policy π^*. Policy Iteration is often more sample-efficient than Value Iteration, as it typically requires fewer iterations to converge.

While **DP** methods provide exact solutions, they suffer from several practical limitations. First, they are affected by the **Curse of Dimensionality**, where the number of possible states grows exponentially in high-dimensional problems, making DP computationally infeasible. Additionally, DP methods rely on a fully known transition model $P(s' \mid s, a)$, which is often unavailable in real-world applications, creating a significant barrier to practical use. Finally, the memory and computation costs associated with storing and updating value functions for all states and actions become prohibitive for large-scale problems.

Due to these limitations, DP is rarely used for real-world reinforcement learning tasks. Instead, approximate and model-free methods, such as MC and TD learning, are preferred. While Dynamic Programming provides exact solutions, it requires full environment knowledge and becomes impractical in large state spaces. Model-Free RL overcomes this by learning optimal policies directly from experience, making it more scalable and adaptable to complex environments.

Model-Free RL vs. Model-Based RL

Reinforcement learning methods can be broadly categorized into model-free and model-based approaches, depending on whether they rely on an explicit model of the environment. This distinction plays a crucial role in determining the efficiency, stability, and applicability of RL algorithms.

Model-Based Reinforcement Learning (MBRL) explicitly constructs a predictive model of the environment dynamics, allowing the agent to simulate interactions before executing real actions. Given a learned transition model $P(s' \mid s, a)$ and reward function $r(s, a)$, the agent can plan optimal actions using various methods, including **Dynamic Programming (DP)**, which utilizes full knowledge of the environment to iteratively solve for optimal policies, **MPC**, which optimizes actions over a finite horizon using the learned model, and **Monte Carlo Tree Search (MCTS)**, which simulates future trajectories to guide decision-making in discrete action spaces [46]. MBRL is generally more sample-efficient than model-free methods, as it can leverage simulations instead of relying purely on real interactions. However, the accuracy of the learned model is critical, model errors can lead to poor decision-making and instability.

Model-Free Reinforcement Learning does not require an explicit model of the environment. Instead, it learns optimal policies directly from interactions by estimating value functions or optimizing policies. Common model-free approaches include **Q-Learning**, which learns the optimal action-value function $Q^*(s, a)$ through trial and error, **Deep Q-Networks (DQN)**, which use neural networks to approximate $Q^*(s, a)$ in high-dimensional spaces, and **Policy Optimization Methods**, which directly optimize policies using algorithms notably Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) [46]. While model-free methods are typically more flexible and robust to model inaccuracies, they often require a large number of interactions to learn optimal behavior, making them less sample-efficient than model-based approaches.

Two key model-free learning paradigms in reinforcement learning are **Monte Carlo (MC) Methods** and **Temporal Difference (TD) Learning**.

MC methods estimate value functions by averaging observed returns from complete trajectories [45]:

$$V(s) \approx \frac{1}{N} \sum_{i=1}^{N} G_i(s) \tag{2.37}$$

where $G_i(s)$ is the total return obtained from an episode starting at state s. MC methods do not require a model but can only update value estimates at the end of an episode, making them inefficient in continuous learning environments.

In contrast, TD Learning updates value estimates incrementally after each step using the Bellman equation [45]:

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right]. \tag{2.38}$$

TD methods leverage bootstrapping, allowing them to update values without waiting for entire episodes to finish, making them more suitable for ongoing learning and large-scale applications.

Comparison of Monte Carlo and TD Learning

- Monte Carlo methods provide unbiased estimates but suffer from high variance.
- Temporal Difference methods introduce bias but have lower variance and enable more frequent updates.
- TD methods are generally preferred for online learning, while MC methods are more useful in episodic settings.

Trade-offs Between Model-Free and Model-Based RL

- Model-Based RL is more sample-efficient but sensitive to model inaccuracies.
- Model-Free RL is more robust but requires extensive interactions with the environment.
- The choice between the two depends on computational constraints, data availability, and application requirements.

Hybrid approaches that combine model-based and model-free techniques are an active area of research, aiming to balance sample efficiency and robustness.

Policy-Based RL and Policy Optimization

Reinforcement learning methods can be broadly categorized into value-based and policy-based approaches. While value-based methods, such as Q-learning and Deep Q-Networks (DQN), estimate value functions to derive optimal policies, policy-based methods directly optimize policies without requiring value function approximation.

Policy-based reinforcement learning is centered around **policy gradient methods**, which aim to optimize a parameterized policy $\pi_\theta(a \mid s)$ by directly maximizing expected returns. Unlike value-based methods, policy gradient approaches do not rely on value functions to determine optimal actions, but instead learn a direct mapping from states to actions through optimization techniques. The policy gradient theorem defines how the policy parameters θ are updated using the gradient of expected rewards [45]:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\nabla_\theta (\log \pi_\theta(a|s)) Q^\pi(s,a) \right]. \tag{2.39}$$

This approach forms the basis of several widely used algorithms, including [47]:

- **REINFORCE**: A Monte Carlo policy gradient method that updates policies based on complete episodes.
- **Proximal Policy Optimization (PPO)**: A method that constrains policy updates to improve stability and sample efficiency.
- **Trust Region Policy Optimization (TRPO)**: A policy optimization method that enforces a trust-region constraint to prevent excessively large updates.
- **Soft Actor-Critic (SAC)**: A method that incorporates entropy regularization to improve exploration and robustness.

Policy-based methods offer several advantages over value-based methods [48]:

- **Continuous Action Spaces**: Policy gradient methods are well-suited for problems with continuous actions, whereas value-based methods struggle in such settings.
- **Stochastic Policies**: Policy-based approaches can naturally model stochastic policies, which are beneficial in multi-agent and uncertain environments.
- **High-Dimensional Tasks**: Value-based approaches suffer from overestimation bias and instability in complex, high-dimensional problems, making policy-based methods preferable in deep RL applications.

However, policy-based methods also have drawbacks:

- **Sample Inefficiency**: Policy gradient methods require large amounts of data to converge, making them less sample-efficient than value-based approaches.
- **High Variance**: The estimation of policy gradients introduces variance, requiring variance reduction techniques such as baseline functions or entropy regularization.

In many real-world applications, RL is combined with traditional control strategies to improve stability and performance. **Residual learning** is an approach where RL refines an existing control policy, such as MPC, PID, or LQR, rather than learning from scratch. This method allows RL to handle dynamic and uncertain environments while leveraging established control methodologies.

Hybrid RL-model-based control techniques are also an area of active research, combining model-based planning with policy-based learning to improve sample efficiency and robustness. These hybrid approaches aim to strike a balance between the data efficiency of model-based methods and the flexibility of policy-based learning, making them particularly suited for high-stakes applications such as robotics and autonomous systems.

Policy-based RL remains a key area of study, enabling RL agents to directly optimize policies in complex environments. The next section will explore further advancements, including model-based approaches and the integration of RL with control systems.

2.2.2. Reinforcement Learning in Aerospace

RL has emerged as a powerful tool for aerospace applications, offering adaptive and data-driven control solutions in environments where traditional model-based approaches face limitations. Unlike conventional controllers that rely on explicit system models, RL learns control policies through interaction with the environment, making it particularly suited for handling nonlinearities, uncertainties, and high-dimensional state spaces.

The adaptability of RL has led to its exploration in several key aerospace domains. In autonomous flight control, RL has been investigated for aircraft stabilization, adaptive maneuvering, and fault-tolerant control, allowing aircraft to respond dynamically to changing conditions. In guidance and navigation, RL-based controllers optimize flight trajectories, improve obstacle avoidance, and enhance decision-making for UAVs and autonomous aircraft. Additionally, RL has been applied to structural load alleviation, where it learns control strategies to mitigate gust-induced and maneuver-induced structural stresses in aeroelastic aircraft. Another emerging application is model mismatch approximation, where RL is leveraged to compensate for discrepancies between theoretical models and real-world behavior, improving the robustness of model-based controllers. While RL presents significant potential, its application in aerospace comes with challenges, including sample inefficiency, safety concerns, and computational feasibility. The following sections delve into these RL applications, highlighting key developments and challenges in each area.

Autonomous Flight Control

Reinforcement Learning has been increasingly explored in aerospace for adaptive flight control, especially where traditional model-based controllers struggle with nonlinearity, uncertainties, and unmodeled disturbances. Unlike conventional controllers that require predefined models and tuning for specific operating conditions, RL-based controllers have demonstrated the ability to learn optimal control strategies dynamically. Several studies have investigated the use of RL for aircraft stabilization, adaptive maneuvering, and fault-tolerant control, offering promising results in handling unpredictable flight conditions. Among these, model-free RL controllers have been applied to aircraft attitude stabilization [18, 49], deep reinforcement learning (DRL) techniques have been utilized for trajectory tracking and maneuvering [50, 19], and hybrid RL-based methods have been explored for fault-tolerant control in failure scenarios [51, 52].

The studies on RL-based aircraft stabilization primarily focus on developing controllers that can regulate attitude in real-time without relying on explicit system models. In [18], an actor-critic neural network framework is used to approximate the optimal control policy for an aircraft attitude system. The controller works online without requiring any knowledge of the dynamics of the system. The controller adapts to varying flight conditions, making it effective for stabilization tasks where traditional methods might require precise system identification. Similarly, [49] introduces a Q-learning approach enhanced with fuzzy logic

to improve discrete action selection, ensuring smoother attitude control. Both approaches demonstrate how RL can be used to replace or complement classical flight controllers in environments where accurate system models are unavailable or where dynamic adaptation is required.

In the domain of adaptive maneuvering, RL has been explored to enable aircraft to learn optimal control policies for trajectory tracking and aggressive maneuvers. [50] presents an adaptive attitude and altitude controller that uses an RL-based algorithm to estimate controller parameters during deployment, enhancing stability and accuracy in nonlinear systems. The approach outperforms conventional PID controllers in trajectory tracking and altitude control, demonstrating improved adaptability under dynamic conditions. [19] develops an Incremental Approximate Dynamic Programming (iADP) controller, which continuously identifies a locally linearized model and optimizes control policies over an infinite horizon. Unlike offline-synthesized control laws, iADP adapts in online to changing aircraft conditions, making it effective for failure recovery. The study details controller integration and validation, culminating in real-world flight tests on a CS-25 class aircraft, marking the first demonstration of an online RL-based automatic flight control system for this category.

Fault-tolerant control has also been a major area of research, leveraging RL to handle actuator failures and system uncertainties. [51] employs Soft Actor-Critic (SAC) reinforcement learning to develop a controller capable of stabilizing an aircraft under actuator degradation and other failure scenarios. The study highlights the advantages of RL in discovering control strategies that traditional methods might not be able to predefine. Another approach is presented in [52], where a neural-network-based observer is integrated with RL to estimate system states and compensate for unknown faults. This hybrid strategy allows for real-time adjustments to unexpected changes in actuator effectiveness, making it highly relevant for fault-tolerant aerospace applications.

Despite these advancements, several limitations remain, making these controllers unsuitable for direct implementation in my work. Many of these studies assume that RL controllers will have access to sufficient training data and the ability to conduct extensive offline learning, which is not always feasible for real-world aerospace applications. [51] highlights the extensive offline training required for the Soft Actor-Critic (SAC) controller, stating that achieving a reliable policy necessitated training on normal plant dynamics for over 10^6 time steps before deployment. Safe exploration is another challenge, as many RL methods rely on trial-and-error learning, which is impractical in flight applications. [19] suggests that deterministic policy-based methods such as Incremental Approximate Dynamic Programming (iADP) improve training stability but at the cost of reducing adaptability to unforeseen conditions. While RL-based controllers have been tested in simulations and, in some cases, real-world trials, they often assume structured environments with predefined disturbances, reducing their robustness to unexpected failures. [49] presents an RL-based stabilization controller that utilizes Q-learning but highlights challenges in non-linearity and stability, requiring careful reward function design and hyperparameter tuning to ensure convergence, suggesting potential limitations when applied to highly dynamic and uncertain flight conditions. Model-free approaches, while adaptive, suffer from sample inefficiency and long training times. In [50], a reinforcement learning-based controller is used to actively estimate and tune control parameters, yet the study emphasizes the difficulty of obtaining a well-trained policy without extensive iterative tuning. Many RL implementations focus on specific, predefined failure cases, limiting their generalizability across different flight conditions. [52] evaluates an RL-based fault-tolerant controller that integrates a neural network-based observer to estimate system disturbances online. While this enables adaptation, the method inherently relies on accurate estimation of system parameters, suggesting that performance may depend on how well the controller has learned from previous conditions. While RL has been successfully applied to aerospace control, existing approaches remain highly specialized and task-specific, lacking a generalizable solution for diverse conditions and disturbances. Dependence on structured environments, predefined failure cases, and extensive tuning limits adaptability to unforeseen scenarios. Trade-offs between safety, sample efficiency, and robustness further highlight the need for careful design and controlled application of RL-based controllers in aerospace.

Guidance and Navigation

Aircraft navigation and trajectory planning require controllers that can efficiently adapt to changing conditions, optimizing factors such as fuel efficiency, airspace constraints, and obstacle avoidance. Here, RL offers a data-driven approach that allows controllers to learn adaptive policies through interaction with dynamic environments, making it well-suited for handling uncertainties and optimizing complex decision-making processes. RL has been applied to waypoint tracking, collision avoidance, and landing optimization. [53, 54]

explore RL-based waypoint navigation, demonstrating robust trajectory tracking and maneuver execution. For collision avoidance, [55] develops a multi-agent RL framework for UAV coordination in congested airspace. [56] integrates RL with PID control to enhance UAV landing stability under environmental disturbances. While these studies highlight RL's potential, challenges remain in generalization and real-time feasibility.

Waypoint navigation has been a key focus area where RL enables aircraft to determine efficient flight paths dynamically. In [53], a deep reinforcement learning controller was developed for a high-performance aircraft using a deep deterministic policy gradient (DDPG) approach. The study highlighted the capability of RL to handle high-dimensional control tasks such as rapid waypoint transitions and maneuvering in complex scenarios. Additionally, [54] introduced a Twin Delayed Deep Deterministic (TD3) Policy Gradient algorithm for quadrotor waypoint navigation, demonstrating successful waypoint tracking in both nominal conditions and under disturbances. These studies underscore RL's ability to adapt to different navigation environments and optimize waypoint selection strategies dynamically.

For collision avoidance, RL-based multi-agent decision-making has been investigated to prevent mid-air conflicts between multiple aircraft. [55] presents a deep multi-agent reinforcement learning (MARL) framework that applies an actor-critic model to train UAVs for cooperative collision avoidance. The study formulates the problem as a multi-agent Markov game, where agents interact and learn real-time policies to avoid collisions while maintaining efficient route planning. The use of long short-term memory (LSTM) networks within the guidance decision-making neural network enhances scalability and adaptability, allowing UAVs to navigate congested airspace more safely.

Landing and approach optimization have also been explored using RL to enhance landing precision under challenging conditions. [56] integrates RL with a conventional PID-based guidance system to improve UAV landing stability, demonstrating how RL can be used to enhance existing controllers rather than replace them. The study leverages deep Q-networks (DQN) for adaptive PID tuning, allowing real-time adjustment of landing parameters based on feedback. By optimizing control gains through reinforcement learning, the approach improves the UAV's ability to handle environmental disturbances such as wind shear and crosswinds.

Despite these advancements, challenges remain in generalizing RL-based navigation policies across different operating conditions. Many studies assume structured training environments where disturbances are predictable and sensor feedback is noise-free. For example, [54] highlights that while its trained controller successfully navigates waypoints, its generalization is limited when faced with significant disturbances not encountered during training. Similarly, [55] formulates RL-based collision avoidance as a multi-agent guidance problem trained in a structured simulation environment. While the approach effectively reduces conflicts in controlled settings, further optimization is needed for real-time implementation, particularly in handling unexpected agent behaviors and rapidly changing airspace conditions. Furthermore, [56] describes extensive training processes, including thousands of training episodes and careful hyperparameter tuning to ensure controller stability. While the study demonstrates RL's ability to optimize UAV landing performance, the reliance on computationally intensive training and parameter adjustments suggests challenges in achieving real-time implementation in dynamic, safety-critical environments. While RL has demonstrated its ability to improve aircraft navigation through waypoint tracking, collision avoidance, and landing optimization, its application remains constrained by computational demands and limited generalization across varying conditions. Many approaches rely on extensive offline training in structured environments, making them less adaptable to real-time changes and unforeseen disturbances. As a result, RL-based navigation controllers require careful tuning and cannot be deployed for online adaptation without prior training on a sufficiently diverse range of scenarios.

Load Alleviation

Aircraft are subjected to significant structural loads due to atmospheric disturbances and maneuvering forces, which impact both safety and operational efficiency. RL has been explored as a data-driven approach to mitigate these effects by enabling real-time adaptation to changing aerodynamic conditions. Recent studies have focused on RL-based strategies for GLA [13, 57], MLA [58, 59], and active aeroelastic control for vibration suppression [60]. These approaches highlight RL's ability to optimize control responses dynamically, reducing reliance on predefined load alleviation strategies.

Gust Load Alleviation (GLA) has been a key application where RL is used to counteract wind gusts by

adjusting control surfaces dynamically. [13] investigates a learning-based approach using a Soft Actor-Critic (SAC) algorithm to train an adaptive control strategy for micro aerial vehicles (MAVs) operating under wind disturbances. The study compares RL-based control with traditional model-based adaptive control and demonstrates that RL can achieve superior disturbance rejection without requiring explicit system modeling. Similarly, [57] develops an RL-based gust alleviation framework for camber-morphing wings, significantly reducing gust impact by autonomously adjusting wing shape in real time. Notably, the study finds that RL-enabled controllers require fewer onboard sensors while maintaining effective performance, reducing computational and sensing overhead.

In the domain of maneuver load alleviation (MLA), RL has been employed to optimize aerodynamic load distribution during high-G maneuvers. [58] applies deep reinforcement learning to an airfoil control system, optimizing control surface deflections to reduce aerodynamic loads under aggressive maneuvers. The study demonstrates that RL-based MLA can improve structural longevity by dynamically distributing forces across the airframe. Meanwhile, [59] explores RL-derived high-alpha aerobatic maneuvers, where reinforcement learning is used to refine control strategies for maintaining stability in confined spaces. These studies highlight RL's capability to enhance maneuverability while minimizing structural stress.

Active aeroelastic control has also been studied using RL for vibration suppression in aeroelastic-wing aircraft. [60] develops an RL-based strategy for active flutter suppression using trailing-edge circulation control, reducing oscillation amplitudes by 92%. The study emphasizes the benefits of model-free learning in handling nonlinear aeroelastic interactions, which are difficult to capture using conventional modeling approaches. By iteratively optimizing control actions in a wind tunnel environment, the RL-based controller learns to adjust jet intensity dynamically, minimizing flutter-induced vibrations.

Despite these advancements, RL-based load alleviation faces several challenges, particularly in real-time applicability and generalization across different flight conditions. Just as in autonomous flight control and guidance and navigation, similar challenges persist, including the reliance on structured training environments, computational efficiency concerns, and the need for stability guarantees. Many RL-based load alleviation studies train controllers in structured environments where disturbances are modeled within predefined parameters. [13] demonstrates an RL-based gust load alleviation strategy trained in a controlled simulation environment, while [57] applies RL to camber-morphing wings, focusing on predefined gust rejection scenarios. While these approaches show promise, their adaptability to unmodeled perturbations remains an open challenge. Additionally, computational efficiency remains a concern, as RL controllers often involve complex training processes and hyperparameter tuning before deployment. [60] applies deep reinforcement learning to active vibration control, highlighting the need for extensive parameter adjustments to achieve stability and performance. Another limitation is the need for stability guarantees, as RL policies may require additional constraints to prevent excessive control actions. [58] applies reinforcement learning to maneuver load alleviation but restricts actuation to discrete control values, highlighting the need for predefined constraints to ensure safe operation.

These factors indicate that while RL-based load alleviation has demonstrated potential in mitigating structural loads and improving aircraft performance, its real-world deployment remains challenging. Issues such as computational complexity, stability concerns, and limited adaptability to unmodeled disturbances must be addressed before RL can be reliably integrated into operational aerospace systems. Future research should focus on improving real-time learning efficiency, incorporating safety constraints, and enhancing generalization across diverse flight conditions to ensure robust and practical implementation.

Model Mismatch Approximation and System Identification

The accuracy of aerospace control systems is often limited by discrepancies between theoretical models and real-world aircraft behavior due to factors such as environmental variability, structural deformations, and unmodeled dynamics. RL has been explored as a tool for improving model fidelity by learning system dynamics, enhancing state estimation, and compensating for inaccuracies in control frameworks. Several studies have investigated RL's role in system identification and adaptive estimation, demonstrating its ability to refine predictive models in dynamic aerospace environments [61, 62, 63].

System identification has been a longstanding challenge in aerospace applications, where accurate models are essential for robust control. [61] presents an RL-based approach to system identification, where forward models are learned through reinforcement signals rather than supervised learning, allowing the system to adapt to nonlinear and time-varying dynamics. Unlike conventional supervised methods that

suffer from error accumulation over long prediction horizons, RL-based techniques can iteratively refine the model by minimizing long-term prediction errors. Additionally, [63] introduces a parameter-informed RL method that improves aircraft system identification by leveraging both prior knowledge and real-time sensor data, ensuring enhanced adaptability to varying flight conditions.

State estimation is another critical aspect where RL has been applied to improve accuracy in high-dimensional nonlinear systems. [62] proposes a reinforcement learning-based reduced-order estimator (RL-ROE), which incorporates a stochastic policy to correct errors in reduced-order models (ROMs). The study highlights RL's ability to compensate for inaccuracies in traditional model reduction techniques by dynamically adjusting the estimator based on real-time sensor measurements. By integrating RL into state estimation, the method demonstrates robustness in handling complex dynamical systems, outperforming conventional Kalman filtering approaches in specific scenarios.

Despite these advancements, RL-based model approximation and state estimation face several challenges, particularly in real-time implementation and stability. Many studies assume structured training environments with predefined dynamical models, limiting their generalizability to unmodeled perturbations [61]. Additionally, computational efficiency remains a concern, as RL-based estimation methods often require extensive training to achieve reliable performance [62]. The lack of formal stability guarantees further complicates deployment in safety-critical aerospace applications, as RL-generated policies may introduce unforeseen control actions [63]. While RL has demonstrated promise in improving system identification and model mismatch compensation, further research is needed to ensure its robustness in real-world aerospace operations. Future efforts should focus on hybrid approaches that integrate RL with physics-based models, safe RL frameworks to enforce stability constraints, and efficient training techniques to reduce computational costs for real-time applications.

2.2.3. Conclusion

RL has been explored across various aerospace applications, demonstrating its potential to improve flight control, navigation, load alleviation, and system identification. In flight control, RL-based controllers have shown adaptability in handling nonlinearities, enhancing maneuvering capabilities, and providing fault tolerance in aircraft stabilization [18, 49, 50, 19, 51, 52]. RL has also been applied to guidance and navigation tasks such as waypoint tracking, collision avoidance, and landing optimization, where it enables improved decision-making under uncertain conditions [53, 54, 55, 56]. In load alleviation, RL-driven strategies have been employed for gust load alleviation (GLA), maneuver load alleviation (MLA), and active vibration suppression, demonstrating effectiveness in dynamically reducing structural stresses [13, 57, 58, 59, 60]. Furthermore, RL has been leveraged in system identification to refine predictive models and improve state estimation accuracy [61, 62, 63].

Despite the increase in popularity, RL-based methods continue to face significant challenges across all domains. One major limitation is the reliance on structured training environments that assume well-defined disturbances and ideal sensor feedback, limiting the ability of RL-based controllers to generalize to real-world uncertainties [49, 19, 13, 54, 55]. Many RL implementations assume access to large amounts of training data and the ability to conduct extensive offline learning, which is often infeasible in aerospace applications where real-world testing is costly and constrained [51, 57]. This results in significant sample inefficiency, as RL-based controllers typically require millions of training steps to develop reliable policies [50, 60]. Safe exploration remains another critical concern, as RL methods traditionally rely on trial-and-error learning, which is impractical in safety-critical aerospace environments [19].

Computational constraints present another major limitation, particularly during the training phase. Many RL algorithms require high-performance computing resources and extensive simulation time to converge, which poses a significant challenge in aerospace applications where access to large-scale data is limited and testing certain failure modes on real systems is often infeasible [52, 61]. However, once training is completed, the resulting RL policies are typically lightweight and can be executed efficiently in real-time on embedded aerospace systems. Additionally, stability and safety guarantees remain a major concern. RL-based controllers lack formal stability assurances, making them difficult to certify for safety-critical applications [63]. Convergence guarantees are also generally absent, with a risk of the policy settling into suboptimal or unstable behavior. The opaque nature of many RL architectures, particularly deep neural networks, further limits explainability, making it difficult to verify, interpret, or trust the policy's decision-making process. The risk of unforeseen control actions during operation further complicates their

adoption, as RL policies may generate unpredictable responses when faced with unmodeled disturbances [58]. Some studies have integrated neural network-based observers to estimate system disturbances, but these approaches often rely on highly accurate prior data, which reduces their adaptability to dynamic flight conditions [52].

While these limitations are substantial, they do not preclude the future use of RL in aerospace. Instead, they highlight the need for refined methodologies that address these challenges. One promising direction is the integration of RL with traditional model-based controllers, such as MPC or adaptive robust control, to leverage the strengths of both approaches. These hybrid frameworks can provide stability guarantees while allowing RL to adapt to complex and uncertain conditions. Further research should also focus on safe RL techniques that enforce stability constraints and prevent unpredictable control actions [51, 60]. Additionally, improving sim-to-real transfer methods will help RL-trained policies generalize better to real-world conditions, reducing their reliance on structured training environments [54, 13]. Finally, optimizing RL architectures for sample efficiency and computational feasibility will be essential to ensure practical real-time implementation in aerospace systems.

While RL has shown great promise in aerospace applications, the literature highlights several fundamental limitations that must be addressed before it can be widely deployed in operational systems. Future work should prioritize hybrid approaches, safe RL methodologies, and computational optimizations to enhance RL's reliability and robustness for aerospace control and decision-making systems.

2.3. Combining MPC and RL

RL and MPC represent two distinct yet complementary approaches to control design. RL provides a data-driven mechanism for learning optimal policies through interaction with an environment, while MPC relies on explicit optimization to generate control inputs that satisfy system constraints and performance objectives. The integration of RL and MPC has gained significant attention as it enables the combination of learning-based adaptability with model-based optimization, addressing many of the limitations inherent in each individual approach [22, 64].

A primary motivation for combining RL with MPC is balancing exploration with constraint satisfaction. While RL excels at discovering novel control strategies through trial-and-error learning, it often struggles with enforcing safety and stability constraints, particularly in high-risk applications such as autonomous driving, robotics, and power systems [65]. In contrast, MPC provides a structured decision-making framework that guarantees constraint satisfaction at each time step but is limited by its reliance on an accurate system model. By integrating RL and MPC, control systems can retain the feasibility guarantees of MPC while benefiting from RL's capacity for adaptation and learning [20].

The integration of RL and MPC can be structured in various ways depending on the control objective. One widely explored approach is using MPC as a policy within RL, where MPC serves as a function approximator for generating control actions, providing structured decision-making within an RL training loop [66]. This formulation ensures that learned policies remain feasible while RL gradually refines cost function weights, constraints, or prediction models. Another approach is RL tuning MPC parameters, where RL operates as an outer-loop adaptation mechanism that dynamically adjusts MPC's internal parameters, such as weight matrices, prediction horizons, or system constraints, optimizing the trade-off between control performance and computational cost [67].

In addition to these direct integrations, RL and MPC can also function in hierarchical or parallel structures. In hierarchical frameworks, MPC supervises an RL controller, ensuring that RL-generated actions remain safe by filtering out unsafe control inputs before execution [65]. Conversely, RL can refine MPC outputs, modifying MPC-generated control decisions to account for unmodeled disturbances, improving adaptability without compromising feasibility [68]. Parallel architectures allow RL and MPC to operate independently while coordinating through shared objectives, where RL optimizes long-term decision-making while MPC ensures short-term feasibility, as seen in microgrid energy management and multi-agent coordination problems [69].

Despite the advantages of combining RL and MPC, several challenges remain. Computational complexity is a significant concern, as RL learning processes require extensive exploration, and MPC involves solving optimization problems at every control step [70]. Furthermore, stability and robustness must be carefully managed, as RL-driven adaptations can introduce parameter fluctuations that destabilize an MPC-controlled system if not properly regulated [71]. Additionally, ensuring data efficiency in RL remains an open challenge, as many RL algorithms require large amounts of training data, which may be impractical in real-world control applications [23].

This section explores various methods by which RL and MPC are integrated, including MPC as a policy within RL, RL-tuned MPC, hierarchical RL-MPC frameworks, and parallel RL-MPC architectures. By reviewing existing approaches, challenges, and real-world applications, it provides insight into how combining RL and MPC enables flexible, efficient, and safe control strategies across diverse domains [64, 22].

2.3.1. MPC as a Policy within RL

MPC has emerged as a powerful alternative to traditional policy representations in RL, providing a structured optimization-based approach for decision-making. Unlike conventional RL policies, which often rely on function approximators such as neural networks, MPC explicitly solves an optimization problem at each time step, ensuring constraint satisfaction and stability in control applications [22]. This makes MPC particularly attractive for safety-critical systems, robotics, and autonomous control, where model-based decision-making can provide improved interpretability and reliability [65].

One of the main motivations for integrating MPC into RL is to leverage its ability to incorporate system constraints directly into the decision-making process. Traditional RL policies often struggle with safety and feasibility constraints, requiring additional mechanisms such as constrained optimization layers or safety filters to ensure compliance. In contrast, MPC naturally enforces these constraints within its optimization

framework, making it a well-suited policy representation for applications requiring high reliability [20]. Furthermore, while standard RL policies rely on extensive trial-and-error learning, MPC incorporates a model of system dynamics, allowing it to make informed predictions about future states and reducing the amount of data required for effective learning [64].

Incorporating MPC within RL can take multiple forms, each offering unique advantages depending on the application context. One approach is to use MPC as a policy approximation, where RL leverages MPC's optimization framework to generate control actions while learning to refine the underlying cost functions, constraints, or prediction models over time [20]. In multi-agent reinforcement learning (MARL), MPC-based policies can be extended to decentralized, cooperative, or competitive settings, where multiple agents interact within a shared environment and optimize their respective objectives [66]. Furthermore, ensuring safety and robustness in RL-MPC policies remains a key research challenge, with techniques such as safe exploration strategies, Lyapunov stability constraints, and uncertainty-aware control being explored to mitigate risks associated with RL-driven decision-making [65].

A fundamental characteristic of MPC in RL is its ability to optimize control actions based on a predictive model of system dynamics, rather than relying purely on trial-and-error learning. By solving an optimization problem at each step, MPC provides a structured way to determine actions that balance immediate rewards with long-term performance. A general framework for MPC as a policy can be defined as:

$$\pi^{\mathsf{MPC}}(\boldsymbol{s}) = \arg\min_{\boldsymbol{u}} \sum_{k=0}^{N^{p-1}} l(\boldsymbol{x}_k, \boldsymbol{u}_k) + V^{\mathsf{f}}(\boldsymbol{x}_{N^p}) \tag{2.40}$$

subject to the system dynamics:

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{2.41}$$

and operational constraints:

$$(\boldsymbol{x}_k, \boldsymbol{u}_k) \in \mathcal{C}, \quad \forall k \in \{0, \ldots, N_p - 1\} \tag{2.42}$$

where $V_{\mathsf{f}}(\boldsymbol{x}_{N_p})$ represents a terminal cost function guiding long-term behavior, and $\mathcal{C}$ is the set of admissible state-action pairs. Within an RL framework, the MPC policy can either remain fixed while RL optimizes high-level tuning parameters, or be adapted iteratively as RL refines its decision-making based on interaction data [22].

While MPC offers many advantages in RL, it also introduces computational challenges. Unlike standard RL policies that execute pre-trained neural networks in constant time, MPC requires solving an optimization problem at each step, which can be computationally expensive, particularly in high-dimensional systems [20]. Additionally, the effectiveness of MPC depends on the accuracy of its predictive model—if the model does not accurately capture real-world dynamics, the resulting policy may be suboptimal or even unsafe [64]. Research efforts have explored techniques for adaptive model learning and exploration strategies informed by value function uncertainty to address these limitations and improve the robustness of RL-MPC frameworks [72].

To illustrate how MPC is used as a policy within RL, the following pseudo-code outlines a basic structure where RL interacts with an MPC-based policy without modifying its core optimization structure dynamically:

Algorithm: MPC as a Function Approximator for RL Policies

Initialize MPC parameters θ (e.g., cost function weights, constraints)
for each episode do
  Observe current state s_t
  Compute action $a_t \leftarrow MPC\_policy(s_t, \theta)$
  Execute action a_t, observe reward r_t, and next state s_{t+1}
  Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer
end for

This framework highlights how RL leverages MPC as a structured control policy while gradually improving decision-making through learning. The following sections will explore the various ways in which MPC is integrated as a policy within RL, including its role as a function approximator, its application in multi-agent settings, and strategies for ensuring safety and robustness in RL-MPC frameworks.

MPC as a Policy Approximation

MPC has gained attention as a structured policy approximation method within RL, providing an alternative to black-box function approximators such as deep neural networks [22]. Unlike standard RL policies, which learn a direct mapping from states to actions through function approximation, MPC explicitly solves an optimization problem at each step, considering system constraints and dynamics [65]. This structured decision-making process enables RL to integrate model-based control strategies while still adapting to real-time data.

One of the key ways MPC is used as a policy approximation is by embedding the optimization problem within the RL loop, where RL tunes the parameters of the MPC controller, such as cost function weights and constraints, to enhance the closed-loop system performance [64]. The resulting MPC-based policy is defined by the optimization problem in Equation (2.40), subject to the system dynamics in Equation (2.41) and the operational constraints in Equation (2.42), where $V_{\mathrm{f}}(\boldsymbol{x}_{N_p})$ serves as a terminal cost function to guide long-term optimization.

This formulation ensures that RL improves MPC parameters over time, refining the policy based on real-world interactions [20]. By structuring RL in this manner, MPC serves as an adaptive policy that evolves through RL's learning process while maintaining constraint satisfaction and stability [64].

The primary advantage of using MPC as a policy approximation is its structured nature, which enables explicit handling of constraints and improved interpretability compared to function approximators [22]. Because MPC optimizes over a receding horizon, it considers both short-term and long-term objectives, leading to higher sample efficiency compared to model-free RL approaches, which often require vast amounts of training data [65]. Another crucial benefit is safety and stability, as MPC inherently satisfies system constraints and prevents violations of operational limits [73]. Unlike neural network-based RL policies, which require additional modifications such as constrained optimization layers, MPC directly integrates constraints within its decision-making process [64]. However, a significant drawback of using MPC as a policy approximation is computational complexity. Unlike traditional RL policies that execute a trained model in constant time, MPC requires solving an optimization problem at each time step, which can be computationally expensive [20]. This challenge is particularly relevant in high-dimensional systems or applications that demand fast control rates [72]. Additionally, the accuracy of the predictive model used within MPC is a critical factor. If the model poorly represents the real system dynamics, the performance of the RL-MPC combination may be suboptimal [22].

Several studies have explored MPC as a policy approximation within RL. One study investigates how RL can adjust the cost function weights of an MPC-based policy to improve adaptability and optimize performance in dynamic environments [22]. Another study examines different ways in which MPC can be integrated into RL frameworks, distinguishing between approaches that use MPC as a direct policy versus those that use it as part of an adaptive control strategy [20]. Additional research has introduced techniques to address MPC's typically conservative decision-making tendencies by incorporating variance-based exploration strategies [72]. Since traditional MPC formulations tend to prioritize feasibility and stability, they often limit exploration in RL settings. By integrating mechanisms that allow for controlled exploration, these approaches ensure that the RL agent can refine the MPC-based policy while still benefiting from its structured optimization process. Other studies further explore how MPC-based policies can be leveraged to improve generalization across tasks, particularly in cases where system constraints and physical limitations play a crucial role in policy execution. These works demonstrate the versatility of MPC when used as a policy within RL, offering a structured yet adaptable alternative to conventional policy representations that can be fine-tuned to different control scenarios.

MPC as a policy approximation offers a structured and interpretable alternative to traditional RL policies while significantly enhancing sample efficiency and stability [64]. However, computational cost and model dependency remain key challenges [20]. Future research directions include reducing computational overhead, improving adaptive model learning, and enhancing exploration strategies to ensure RL-MPC frameworks can be widely deployed across various applications [22].

Multi-Agent RL with MPC as Policy

In multi-agent reinforcement learning (MARL), multiple agents interact within a shared environment, each optimizing their respective objectives while often influencing one another. When MPC is used as the policy for each agent, the system benefits from the structured decision-making of MPC while leveraging RL

to enhance coordination, adaptability, and learning efficiency [66]. Unlike traditional MARL approaches, where agents use learned policies parameterized by neural networks, MPC provides an optimization-based control framework that inherently respects system constraints, making it particularly suitable for cooperative, competitive, or decentralized multi-agent settings.

In a multi-agent setup, each agent i follows an MPC policy π_i^{MPC}, where actions are computed through optimization over a receding horizon:

$$\pi_i^{\mathsf{MPC}}(\boldsymbol{s}_{i,t}) = \arg\min_{\boldsymbol{u}_{i,t}} \sum_{k=0}^{N^{P-1}} l_i(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}) + V_i^{\mathsf{f}}(\boldsymbol{x}_{i,N_p}) \tag{2.43}$$

subject to the system dynamics:

$$\boldsymbol{x}_{i,k+1} = f_i(\boldsymbol{x}_{i,k}, \boldsymbol{u}_{i,k}, \boldsymbol{x}_{\mathsf{other},k}) \tag{2.44}$$

where $\boldsymbol{x}_{\mathsf{other},k}$ represents the states of other agents that influence agent i's dynamics. The inclusion of multi-agent dependencies increases the complexity of the optimization problem, necessitating coordination strategies such as decentralized MPC, hierarchical control, or game-theoretic formulations to ensure agents reach an optimal equilibrium [22].

The integration of RL with multi-agent MPC-based policies is typically structured in different ways. In decentralized RL-MPC, each agent independently uses an RL algorithm to fine-tune its own MPC parameters, leading to self-optimization without direct communication [66]. In cooperative RL-MPC, agents collaborate by sharing state information and learning a joint strategy for improving system-wide coordination [20]. In competitive RL-MPC, agents use RL to adaptively adjust MPC-based strategies in adversarial environments, such as autonomous driving or robotic competitions, where optimizing for individual objectives might conflict with others [64].

Using MPC within MARL offers several advantages. First, MPC inherently ensures constraint satisfaction, which prevents agents from violating physical limitations or safety-critical conditions [65]. Second, MPC improves decision interpretability, as actions are derived from explicit optimization objectives rather than black-box neural networks. Third, MPC can stabilize learning in MARL, where instability often arises due to non-stationarity introduced by multiple agents learning simultaneously [72]. However, MARL with MPC also presents challenges. Computational complexity increases significantly, as each agent must solve an optimization problem at every step, requiring efficient solvers and distributed computation. Additionally, coordination is non-trivial, especially in decentralized settings where agents do not share full state information, making convergence more difficult [66].

Several studies have investigated the use of MPC as a policy in MARL. One study explores decentralized MPC in a multi-agent robotic control scenario, where each robot uses RL to optimize its own MPC parameters while maintaining real-time feasibility [66]. Another study examines cooperative RL-MPC frameworks in autonomous traffic management, where vehicles coordinate using shared learning signals to optimize throughput and safety [20]. Competitive RL-MPC has also been explored in adversarial drone control, where agents adjust their predictive strategies dynamically to counter opponents [64]. These works highlight how MPC-based policies can enhance MARL applications by introducing structure, constraint-handling, and interpretable decision-making.

The combination of RL with multi-agent MPC policies presents a powerful approach for complex, interactive environments. Future research will likely focus on improving computational efficiency, designing robust coordination mechanisms, and scaling MARL-MPC frameworks to handle high-dimensional, real-world problems.

Safety and Robustness in RL-MPC Policies

Ensuring safety and robustness in RL-based MPC policies is critical for applications where failures can have severe consequences, such as autonomous systems, robotics, and industrial process control [65]. While MPC inherently provides constraint satisfaction and stability guarantees through its optimization framework, integrating RL introduces challenges in maintaining these guarantees under uncertainty, exploration, and learning-based adaptations.

One approach to enhancing safety in RL-MPC policies is to incorporate safe exploration strategies, ensuring that RL does not violate safety constraints while learning optimal control strategies. This can be

achieved through constraint satisfaction techniques, where RL modifies MPC parameters while ensuring that feasibility conditions remain met at all times [73]. Mathematically, this can be expressed as:

$$\theta^* = \arg\max_{\theta \in \Theta} \mathbb{E}\left[\sum_{t=0}^{T} r_t \mid \pi_\theta(s_t), \mathcal{C}\right] \tag{2.45}$$

where $\mathcal{C}$ represents system constraints that must hold throughout the learning process, and Θ is the admissible set of policy parameters considered by the RL agent. Unlike unconstrained RL, this formulation ensures that MPC remains within a predefined safe operating space while RL adapts its control strategy.

Another method to improve robustness is distributionally robust MPC, where the MPC optimization accounts for uncertainty in system dynamics and disturbances [64]. By modeling uncertainty explicitly, RL can learn adaptive control strategies that perform well under worst-case conditions, reducing the risk of performance degradation in deployment [72]. This is particularly useful in scenarios where RL must operate in environments with dynamic disturbances or partial observability.

While RL enhances adaptability in MPC-based policies, it also introduces stability risks if parameters are modified too aggressively. To mitigate this, Lyapunov-based constraints can be used to enforce stability guarantees during learning [65]. These constraints ensure that RL modifications to MPC's cost function or constraints do not lead to unstable control policies. Additionally, robustness constraints can be introduced to ensure that RL-driven adaptations remain within feasible margins, preventing drastic parameter updates that could destabilize the system [20].

Several studies have explored safety and robustness in RL-MPC policies. One study investigates safe RL for MPC by using barrier functions to enforce safety constraints while allowing RL to optimize control performance [65]. Another study explores robust MPC tuning via RL, where RL modifies MPC cost weights based on uncertainty estimates, ensuring performance under worst-case conditions [73]. Further research introduces trust-region RL-MPC approaches, which prevent RL from making large, destabilizing changes to MPC parameters by enforcing gradual updates [22]. These works demonstrate the potential of RL-MPC frameworks in maintaining safety and robustness while improving adaptability.

Despite these advancements, challenges remain in balancing exploration and constraint satisfaction, ensuring real-time feasibility of RL-driven MPC tuning, and designing generalizable safety mechanisms that scale across different applications. Future work is likely to focus on integrating learning-based uncertainty estimation with MPC to improve robustness and developing certified-safe RL-MPC frameworks for critical real-world applications.

2.3.2. RL adjusting MPC parameters

MPC is a widely used optimization-based control strategy that relies on solving a constrained optimization problem at each time step to determine the optimal control inputs. While MPC provides a structured approach to decision-making, its performance is highly dependent on manually tuned parameters such as cost function weights, prediction horizons, and system constraints. Improper tuning of these parameters can lead to suboptimal performance, infeasibility, or excessive computational burden. In dynamic or uncertain environments, static MPC parameters may fail to adapt to changing system conditions, limiting control efficiency and robustness [22].RL has emerged as a promising approach for automating MPC parameter tuning, allowing the controller to adjust its parameters based on real-time feedback rather than relying on offline manual tuning. Unlike traditional tuning methods, which require extensive empirical testing, RL enables continuous adaptation of MPC parameters, improving control performance in uncertain and time-varying environments. By leveraging RL's ability to learn from experience, MPC can dynamically adjust cost function weights, modify constraints, and refine prediction models, enhancing both efficiency and robustness [20, 21].

The integration of RL and MPC can be structured in multiple ways, depending on the specific objectives of parameter adaptation. RL can be used to tune cost function and constraint parameters, improving control trade-offs between performance and feasibility. It can also be applied to adaptive parameterization, where the RL agent modifies internal MPC structures such as prediction horizons and model representations. Furthermore, safe RL methods ensure that RL-driven adaptations do not destabilize the control system by enforcing Lyapunov constraints, trust-region constraints, or robust optimization techniques [71]. These approaches enable RL to refine MPC tuning while maintaining stability and feasibility, making RL-tuned

MPC suitable for safety-critical applications such as autonomous driving, energy grid management, and robotics [23, 67].

Despite its advantages, RL-driven MPC tuning presents several challenges, including computational complexity, stability concerns, and learning efficiency. Since MPC already involves solving an optimization problem at every control step, adding RL-based adaptation further increases computational demand, making real-time feasibility a key consideration. Additionally, aggressive RL updates can introduce instability, requiring safe RL mechanisms to ensure smooth adaptation. Furthermore, traditional RL algorithms rely on extensive exploration, which is impractical in real-world control systems where unsafe exploration can lead to failures or inefficiencies. Addressing these challenges requires the development of hybrid RL-MPC architectures, uncertainty-aware adaptation strategies, and scalable RL methods that balance performance, safety, and computational efficiency [70, 74].

This section explores the role of RL in automating MPC parameter tuning, discussing its applications in cost function and constraint adaptation, adaptive parameterization, and safe RL strategies. Additionally, challenges and future directions are examined, highlighting ongoing research efforts to improve the practicality and reliability of RL-driven MPC tuning. Through a comprehensive review of existing approaches, this section aims to provide insights into how RL can enhance MPC's adaptability while ensuring robustness and stability in real-world applications [71, 23].

Reinforcement Learning for MPC Cost Function and Constraint Tuning

MPC relies on a predefined cost function and a set of constraints to optimize control decisions over a prediction horizon. However, traditional MPC formulations often require manual tuning of these components, which can lead to suboptimal performance in dynamic or uncertain environments. RL offers a data-driven approach to adaptively tuning MPC parameters, allowing for real-time optimization of cost function weights and constraints to improve control performance [20, 21, 75].

The integration of RL into MPC parameter tuning is typically structured as an outer-loop learning process, where the RL agent modifies key MPC parameters while the MPC controller continues to execute real-time optimization at each control step. The RL agent does not replace MPC but instead acts as a high-level tuning mechanism, updating MPC's cost function or constraints based on observed system performance [67]. This approach allows MPC to retain its optimization-based decision-making structure while gaining the flexibility of RL adaptation, making it well-suited for dynamic environments where fixed tuning parameters may become suboptimal over time.

For cost function tuning, RL interacts with the MPC controller by adjusting the weight matrices that define the trade-offs between state tracking and control effort. The MPC optimization problem is typically expressed as:

$$J(\boldsymbol{x}_0, Q, R) = \sum_{k=0}^{N^{\mathrm{p}}-1} l(\boldsymbol{x}_k, \boldsymbol{u}_k, Q, R) + V^{\mathrm{f}}(\boldsymbol{x}_{N^{\mathrm{p}}}) \tag{2.46}$$

where Q and R are the state and control weighting matrices. These parameters significantly influence the behavior of the control policy, determining how aggressively or conservatively the system responds to disturbances. Manually tuning Q and R is often inefficient, as it requires trial-and-error adjustments that may not generalize well across different operating conditions. By leveraging RL, the controller can dynamically modify these parameters based on real-time feedback, optimizing performance under changing conditions while avoiding the need for repeated manual recalibration [70].

In addition to cost function tuning, RL can also adjust the constraint definitions within MPC. Standard MPC constraints are defined as:

$$x_k \in \mathcal{X}_\theta, \quad u_k \in \mathcal{U}_\theta, \tag{2.47}$$

where θ represents tunable constraint parameters. RL dynamically modifies these parameters to prevent infeasibility in cases where rigid constraint definitions could lead to control failures. This flexibility is particularly beneficial for systems operating under time-varying or uncertain conditions, such as power

systems, robotics, and transportation networks, where static constraints may either be too restrictive or insufficiently conservative [71].

The combination of RL and MPC for parameter tuning offers several advantages. By continuously adjusting MPC parameters, RL enables improved adaptability, allowing the controller to respond to varying environmental conditions without requiring manual intervention. This automated optimization leads to more efficient trade-offs between control effort and performance, ensuring that the system operates optimally under different scenarios. Additionally, RL-enhanced MPC can handle uncertainty more effectively, as it learns to adjust constraints dynamically, reducing the risk of infeasibility in constrained optimization problems [74]. However, these benefits come at the cost of increased computational complexity. The inclusion of an RL agent requires additional learning iterations, and the adaptation of MPC parameters in real time can be computationally demanding. Furthermore, stability concerns arise when RL updates parameters too aggressively, potentially leading to control instability. To mitigate these risks, techniques such as trust-region policy updates, Lyapunov-based constraints, and robust MPC constraints have been explored to ensure that RL modifications remain within acceptable stability bounds [70].

Several real-world applications demonstrate the effectiveness of RL-based cost function and constraint tuning in MPC. In greenhouse climate control, RL dynamically adjusts MPC cost function weights to optimize temperature and humidity regulation while minimizing energy consumption [76]. Similarly, in microgrid energy management, RL modifies economic MPC weights in response to fluctuating electricity prices and demand, improving efficiency while maintaining operational constraints [77]. Another example is highway ramp metering, where RL adapts the cost function to balance congestion levels and traffic throughput, improving efficiency in transportation networks [74]. In autonomous vehicles, RL-tuned MPC is used to adapt constraints in nonlinear vehicle models, ensuring stability and robustness under diverse driving conditions [71]. These examples highlight the versatility of RL-tuned MPC frameworks, demonstrating their ability to improve adaptability and efficiency in dynamic environments.

Overall, RL-based cost function and constraint tuning provides a powerful approach for enhancing MPC's adaptability and performance, particularly in systems with high uncertainty or variable operating conditions. By leveraging RL to automate parameter tuning, MPC can continuously optimize its decision-making process in response to real-world variations. However, the computational burden and safety concerns remain significant challenges, requiring further research into efficient RL algorithms, safe exploration methods, and real-time feasibility improvements [71, 70].

Adaptive and Learning-Based Parameterization of MPC

Traditional MPC relies on a fixed set of parameters, including cost function weights, prediction horizons, and system constraints, which are often tuned manually based on domain knowledge and empirical testing. However, static parameterization limits MPC's adaptability in dynamic environments where system characteristics, disturbances, or objectives change over time. RL provides a data-driven approach to adaptively updating these parameters, allowing the controller to modify its internal structure based on real-time observations and historical performance [67, 75, 23].

Unlike direct RL tuning of cost functions or constraints, adaptive MPC parameterization focuses on modifying the controller's internal structure rather than just its optimization weights. This includes adjusting elements such as prediction horizon lengths, control input constraints, terminal costs, and disturbance rejection parameters. By embedding RL within an adaptive framework, the controller can actively learn how changes in these parameters influence system performance, adjusting them accordingly to enhance efficiency, stability, and robustness [70].

A key application of RL-driven adaptive MPC is real-time system identification, where the controller continuously updates its predictive model based on observed discrepancies between actual and expected system behavior. Traditional MPC requires a well-defined model of system dynamics, which may degrade due to unmodeled disturbances, parameter drift, or external influences. RL can be integrated into system identification-based adaptive MPC, where it refines model parameters dynamically, ensuring that the predictive model remains accurate even in uncertain environments [78]. This approach is particularly useful in applications such as industrial process control and autonomous vehicles, where environmental conditions fluctuate, requiring continuous model adaptation.

Another advantage of RL-based MPC parameterization is the ability to manage computational complexity dynamically. Standard MPC formulations rely on a fixed prediction horizon and control update rate,

which may not always be optimal under varying conditions. RL enables adaptive horizon tuning, where the controller modifies its prediction depth based on real-time computational constraints and control requirements. A longer prediction horizon improves decision-making in highly uncertain environments but increases computational burden, while a shorter horizon reduces complexity but may sacrifice optimality. RL provides an automated mechanism to balance these trade-offs, ensuring computational efficiency without compromising performance [74]. Similarly, RL can dynamically adjust constraint relaxation levels, allowing the controller to prioritize feasibility or conservatism depending on system state [21].

Despite these benefits, adaptive MPC parameterization via RL introduces challenges. One concern is parameter instability, where excessive exploration or aggressive updates can lead to oscillatory or unpredictable control behavior. In safety-critical applications such as aerospace systems and energy networks, abrupt parameter shifts can degrade system performance or even cause operational failures. To mitigate this, structured RL exploration methods and stability-aware RL techniques such as Lyapunov-based adaptation and trust-region constraint tuning are employed to ensure smooth parameter evolution while preserving adaptability [71]. Another challenge is data efficiency, as RL-driven adaptation requires a balance between exploration and exploitation to refine parameters effectively without introducing unnecessary variability [23].

Several real-world applications demonstrate the effectiveness of RL-based adaptive MPC parameterization. In vertical takeoff and landing (VTOL) aircraft control, RL enhances flight stability by dynamically tuning MPC parameters related to aerodynamic modeling and actuator constraints [79]. In traffic control, RL-based adaptation improves highway ramp metering by modifying MPC's prediction horizon and relaxation constraints, allowing for better congestion management under varying demand conditions [74]. Similarly, in microgrid energy management, RL tunes operational constraints in economic MPC models to optimize power distribution while adapting to fluctuations in renewable energy supply and demand [77]. These applications highlight how adaptive parameterization enables MPC to function efficiently across complex, changing environments.

Overall, RL-based adaptive parameterization enhances MPC's ability to operate in uncertain and evolving conditions by continuously refining its internal models and optimization settings. This approach significantly improves control adaptability while reducing the reliance on manually tuned parameters. However, ensuring stability, managing computational overhead, and refining RL exploration strategies remain key challenges. Future research will likely focus on developing robust adaptive learning techniques, hybrid RL-MPC architectures, and real-time feasibility improvements to further enhance the reliability and efficiency of RL-driven MPC parameterization [71, 23].

Safe RL for Stabilizing MPC Parameters

RL has demonstrated significant potential in adjusting MPC parameters dynamically, improving adaptability and performance across various applications. However, a major challenge in RL-driven MPC tuning is ensuring that parameter modifications do not destabilize the system or violate critical safety constraints. Unlike traditional MPC, which guarantees constraint satisfaction through explicit optimization, RL-based adaptation introduces uncertainty, as learned updates may unintentionally degrade control stability. To address this, safe RL approaches have been developed to ensure that RL-driven MPC tuning maintains feasibility and robustness while optimizing performance [71, 20, 21].

Safe RL methods for stabilizing MPC parameters focus on incorporating safety guarantees within the RL learning process to prevent destabilizing parameter updates. One approach is to enforce hard safety constraints directly into the RL optimization, ensuring that parameter updates remain within predefined stability regions. This is achieved by embedding safety constraints within the reward function or modifying the RL exploration process to restrict parameter search to safe, feasible regions of the state space. In this framework, RL learns an optimal tuning policy while respecting stability bounds defined by the underlying control system [23].

Another widely used strategy is Lyapunov-based constraint enforcement, where RL parameter updates are evaluated using Lyapunov stability conditions to ensure that they do not introduce system instability. This approach prevents RL from making aggressive parameter modifications that could lead to control divergence. By incorporating Lyapunov constraints into the learning process, the RL agent is restricted to tuning MPC parameters in a manner that guarantees asymptotic stability, improving the reliability of adaptive MPC frameworks [71].

In addition to Lyapunov constraints, trust-region RL methods have been explored as a way to stabilize MPC parameter tuning. These methods restrict the magnitude of RL updates at each learning step, ensuring that parameter changes remain gradual and do not cause abrupt shifts in control behavior. By limiting the step size in parameter adaptation, trust-region methods reduce the risk of overcorrection, allowing RL to refine MPC parameters without inducing oscillatory or unstable responses [70]. This is particularly beneficial in safety-critical applications such as autonomous flight, energy grid management, and industrial process control, where abrupt changes in control policy can have severe consequences.

Another key aspect of safe RL for MPC stabilization is robust constraint adaptation, where RL adjusts soft constraints dynamically while ensuring that safety-critical constraints remain intact. In traditional MPC, feasibility constraints are typically fixed, leading to potential infeasibility in highly uncertain environments. Safe RL frameworks introduce adaptive constraint tuning, where RL modifies constraint boundaries in response to real-time conditions while ensuring that system-critical safety margins are never violated [67]. This ensures that even as RL optimizes for performance, it does not compromise the controller's ability to maintain safe operating conditions.

Despite the advantages of safe RL methods, challenges remain in balancing learning efficiency with stability guarantees. Constraining RL updates through Lyapunov functions, trust regions, or robust constraints can slow down the learning process, requiring additional computational resources to ensure safe adaptation. Furthermore, safe RL techniques must be carefully designed to avoid excessive conservatism, where constraints are so restrictive that the RL agent is unable to explore meaningful parameter improvements. Finding the right balance between exploration, constraint enforcement, and stability guarantees is an ongoing research challenge [71].

Several real-world applications highlight the importance of safe RL-driven MPC tuning. In industrial process control, RL has been used to tune economic MPC parameters while ensuring that stability constraints remain satisfied, preventing process fluctuations that could result in inefficiencies or safety violations [77]. In autonomous vehicle navigation, RL-driven MPC has been constrained using trust-region and Lyapunov-based stability enforcement to prevent unsafe driving maneuvers in high-speed environments [71]. Similarly, in power grid management, safe RL methods ensure that MPC tuning maintains grid stability under fluctuating demand and generation conditions, preventing voltage or frequency instabilities [23].

Overall, safe RL for stabilizing MPC parameters enables adaptive control improvements while maintaining robustness, feasibility, and safety. By integrating techniques such as Lyapunov constraints, trust-region updates, and robust constraint adaptation, RL can refine MPC tuning without introducing instability. However, ensuring scalability, computational efficiency, and an optimal balance between safety and exploration remains an area of ongoing research. Future work is likely to focus on hybrid safe RL-MPC architectures, uncertainty-aware learning strategies, and real-time constraint adaptation to further enhance the safety and reliability of RL-driven MPC frameworks [71, 70].

Challenges and Future Directions in RL-Tuned MPC

While RL has demonstrated significant potential in tuning MPC parameters, several challenges remain in achieving reliable, efficient, and scalable implementations. The integration of RL with MPC introduces additional computational overhead, stability concerns, and difficulties in balancing exploration with constraint satisfaction. Addressing these challenges is critical for ensuring that RL-driven MPC tuning is viable for real-time applications and safety-critical systems [71, 74, 23].

One of the primary challenges in RL-tuned MPC is computational complexity. Standard MPC formulations already require solving an optimization problem at every control step, and adding RL-based adaptation further increases computational demand. This is particularly problematic for applications with strict real-time constraints, such as robotic control, aerospace systems, and autonomous driving, where delays in control execution can compromise system safety. Although techniques such as adaptive horizon tuning and parallel computation have been explored to mitigate this issue, real-time feasibility remains a key limitation in RL-tuned MPC implementations [70]. Future research may focus on developing lightweight RL architectures, model-based acceleration techniques, and approximate dynamic programming methods to reduce computational overhead without sacrificing performance [67].

Maintaining stability and robustness remains a key challenge in integrating RL with control frameworks. Unlike traditional MPC, where constraints and stability guarantees are explicitly embedded in the optimization problem, RL-based adaptation introduces uncertainty in parameter updates. If RL modifies cost function

weights, constraints, or model parameters too aggressively, it can lead to oscillatory behavior, infeasibility, or control divergence. Safe RL methods, such as Lyapunov-based adaptation, trust-region constraints, and robust optimization, have been proposed to mitigate instability risks. However, these methods often introduce conservatism, limiting the ability of RL to explore optimal parameter adjustments [71]. Future research could focus on hybrid safe RL-MPC frameworks that dynamically balance exploration and safety, ensuring stable adaptation while maintaining performance improvements [23].

Improving sample efficiency and learning speed is a central challenge in RL-driven MPC tuning. Traditional RL methods require extensive exploration to learn optimal policies, which is impractical for real-world control systems where data collection is costly or time-consuming. Most RL algorithms rely on trial-and-error interactions, which may not be feasible in safety-critical applications such as nuclear plant control, industrial automation, or high-speed transportation systems. Methods such as meta-learning, transfer learning, and model-based RL offer promising directions to improve learning efficiency by leveraging past experiences to accelerate adaptation [20]. Future work may focus on uncertainty-aware learning approaches that use Bayesian inference or probabilistic models to reduce exploration requirements and improve data efficiency [67].

A fundamental challenge in RL-tuned MPC is interpretable and explainable adaptation. Unlike traditional control design, where parameter selection follows analytical reasoning, RL-based tuning operates as a black-box process, making it difficult to interpret why certain parameter updates are made. This lack of transparency can limit the adoption of RL-driven MPC in safety-critical industries where regulatory compliance and human oversight are required. Future research should explore explainable RL techniques, where adaptive parameter tuning is augmented with sensitivity analysis, human-in-the-loop adjustments, and uncertainty quantification, ensuring that decision-making processes remain interpretable and trustworthy [71].

Despite these challenges, RL-driven MPC tuning offers exciting opportunities for future advancements. One promising direction is hybrid RL-MPC architectures, where RL operates alongside adaptive model-based methods, constraint-aware learning, and online system identification. By combining RL's learning capabilities with MPC's structured optimization, these hybrid approaches could provide scalable, robust, and real-time adaptable control frameworks [74]. Another direction is multi-agent RL for decentralized MPC tuning, where multiple RL agents collaboratively optimize MPC controllers across interconnected systems, such as autonomous vehicle fleets, smart energy grids, and industrial robotics networks [23]. Additionally, distributed reinforcement learning could be leveraged to parallelize MPC tuning across edge computing platforms, reducing computational bottlenecks and enabling large-scale adaptive control [70].

Overall, while RL-tuned MPC presents notable challenges in terms of computation, stability, learning efficiency, and interpretability, ongoing research is actively addressing these limitations. Future innovations in safe RL, hybrid learning architectures, uncertainty-aware optimization, and explainability techniques will further enhance the practicality of RL-driven MPC tuning, paving the way for its broader adoption in autonomous systems, industrial automation, and complex multi-agent control environments [71, 23].

2.3.3. RL modifying MPC outputs

In hybrid control frameworks, RL is often employed to refine or adapt the outputs of an MPC controller, leveraging the strengths of both approaches to improve control performance. MPC provides structured decision-making by solving an optimization problem at each time step, ensuring constraint satisfaction and stability. However, its effectiveness is often limited by model inaccuracies, disturbances, and system nonlinearities that are difficult to capture in real time. RL, on the other hand, excels at learning complex, data-driven corrections that can enhance adaptability and robustness. By integrating RL as a refinement layer, the controller benefits from MPC's ability to generate feasible and safe control actions while allowing RL to introduce dynamic adjustments that improve response to uncertainties and optimize performance in ways that traditional MPC may not achieve. This approach enables better handling of unknown disturbances, reduces reliance on highly accurate system models, and enhances adaptability without sacrificing the stability and constraint satisfaction inherent to MPC.

One implementation of this approach is seen in both off-road autonomous driving and bipedal locomotion, where RL is used to refine MPC-generated actions to improve adaptability and robustness [68, 80]. In off-road driving, an Actor-Critic Reinforcement Learning Compensated MPC (AC2MPC) framework modifies the acceleration and steering inputs computed by MPC, compensating for unmodeled terrain interactions.

While MPC provides a baseline control policy based on an approximated vehicle model, RL learns a residual correction to account for factors that MPC does not explicitly model, such as tire-soil interactions. A similar concept is applied in bipedal locomotion, where MPC generates initial footstep placements based on a simplified model, and RL refines these steps to better account for whole-body dynamics. This refinement allows the robot to improve stability and agility when responding to external disturbances. In both cases, RL does not replace MPC but instead learns to make targeted modifications, ensuring that control decisions remain safe and feasible while improving adaptability. The effectiveness of these frameworks lies in their ability to retain the stability guarantees of MPC while allowing RL to introduce data-driven improvements that respond to uncertainties in real time.

A similar integration is found in urban traffic control, where RL dynamically adjusts MPC-generated traffic signal timings in response to congestion patterns [81]. Traditional MPC-based traffic signal control relies on predefined models of traffic flow, which may not always capture real-world variability. By incorporating RL as an adaptive correction mechanism, the system can modify green time distributions computed by MPC to better respond to unexpected congestion or disturbances. Here, MPC provides a structured decision-making process that ensures feasibility, while RL introduces flexibility by learning optimal modifications based on traffic conditions. This hybrid approach leads to improved efficiency over standalone MPC, particularly in uncertain environments. However, the challenge lies in ensuring that RL learns modifications that enhance, rather than degrade, MPC's performance, as poorly trained RL agents may introduce suboptimal changes.

A hierarchical RL-MPC framework has also been proposed for freeway traffic control, where RL modifies MPC-generated speed limits and ramp metering decisions [82]. In this framework, MPC operates at a lower frequency, solving an optimization problem to provide baseline control inputs, while RL works at a higher frequency to refine these outputs in real time. The advantage of this structure is that MPC establishes a constraint-satisfying foundation for traffic management, ensuring regulatory compliance and system stability, while RL learns to adjust the MPC decisions dynamically, compensating for external disturbances and changing traffic conditions. By decoupling long-term strategic control (handled by MPC) from real-time adaptability (handled by RL), this framework efficiently balances computational complexity and control effectiveness. However, a key challenge remains in tuning the coordination between the two controllers to ensure stability and prevent excessive corrections by RL that could lead to oscillatory or inefficient behaviors.

The integration of RL to refine MPC outputs enhances adaptability while maintaining constraint satisfaction and stability. This synergy reduces reliance on highly accurate models and improves performance across diverse applications. However, challenges remain in ensuring RL's modifications enhance rather than disrupt MPC's decisions. Poorly tuned RL policies can introduce instability, making careful coordination essential. Future research should focus on structured RL refinements, uncertainty-aware learning, and improved sample efficiency to ensure reliable and safe deployment. Balancing RL's flexibility with MPC's robustness remains key to unlocking the full potential of this hybrid approach.

2.3.4. MPC Supervising an RL Controller

RL has shown promise in control applications but struggles with enforcing safety constraints and maintaining stability in real-world deployments. MPC, with its ability to enforce explicit constraints and optimize control actions over a prediction horizon, can be integrated as a supervisory mechanism to ensure RL-generated decisions remain feasible and safe. Several studies have investigated how MPC can serve as a supervisory layer for RL, preventing unsafe actions while still allowing RL the flexibility to explore and learn optimal policies.

One study applies robust MPC as a safety constraint mechanism in RL-based control systems, ensuring that state constraints are always satisfied even under model uncertainty [65]. In this framework, RL is responsible for optimizing control policies, but instead of directly executing the RL-generated actions, MPC acts as a filtering mechanism that refines and enforces safe control inputs. Specifically, the RL agent proposes a control action based on learned policies, and MPC evaluates this action in real-time, modifying or rejecting it if it violates safety constraints. This ensures that the final applied control input remains within a feasible and stable region. The key advantage of this approach is that RL does not need to learn safety constraints through penalties or additional training, reducing the sample complexity and improving learning efficiency. However, this method depends heavily on the accuracy of the system model used for MPC, as

errors in the model could lead to suboptimal control decisions. Additionally, the robust constraints enforced by MPC may lead to overly cautious control actions, which could limit performance by preventing the RL agent from exploring aggressive but potentially optimal strategies.

Another study focuses on RL for microgrid energy management, where MPC is integrated to maintain feasibility and optimize real-time decision-making [69]. Microgrids involve both discrete decisions, such as switching between power sources, and continuous control actions, such as regulating power distribution. The framework decouples these two decision-making processes, allowing RL to handle the discrete, high-level operational decisions while MPC is responsible for low-level continuous control, ensuring feasibility and stability. This separation allows both controllers to operate independently while still complementing each other—RL determines optimal long-term switching strategies, and MPC refines the execution by optimizing power flow within predefined constraints. By acting as a real-time feasibility filter, MPC supervises RL's decisions, ensuring that every action taken adheres to operational and safety constraints. This structure not only enhances learning efficiency but also ensures that RL-generated policies remain valid under real-world conditions, preventing the agent from making infeasible or unsafe choices.

A different approach is presented in a study that employs NMPC as a safety filter in RL-driven control applications [71]. Unlike methods that integrate MPC within the RL learning process, this study keeps RL as the primary decision-making mechanism but places an NMPC layer between the RL agent and the system to override unsafe actions. By dynamically adjusting RL's policy, NMPC prevents RL from violating safety constraints while allowing it to focus on optimizing performance. This approach enables RL to explore more aggressively while maintaining stability and avoiding catastrophic failures. However, it introduces additional computational overhead due to the need for real-time NMPC evaluations, and the extent to which NMPC interventions restrict RL's learning dynamics must be carefully tuned to balance safety enforcement with exploration.

The integration of MPC as a supervisory mechanism for RL demonstrates the potential to balance safety, feasibility, and optimization in control applications. By leveraging MPC's constraint-handling capabilities, RL can focus on learning high-performance policies without the risk of unsafe actions. However, challenges such as computational complexity, model dependency, and balancing exploration with constraint enforcement remain key areas for further research. Future work may explore hybrid learning-adaptive approaches that allow for dynamic adjustments to the supervisory framework, ensuring greater flexibility while maintaining robust performance.

2.3.5. RL and MPC Running in Parallel

In many control applications, RL and MPC can be utilized in parallel, where both controllers operate independently while shaping system behavior through distinct mechanisms. Unlike approaches where one method explicitly influences or is embedded within the other, this parallel structure allows RL and MPC to function autonomously, each addressing different aspects of the control problem. RL explores and refines long-term strategies based on data-driven learning, while MPC ensures short-term feasibility and constraint satisfaction through real-time optimization. By maintaining separate yet complementary roles, this approach leverages the adaptability of RL without compromising the stability and robustness provided by MPC, resulting in a control framework that balances learning-based flexibility with formal constraint handling.

A key advantage of this structure is that both controllers can be tuned separately, allowing RL to focus on long-term decision-making while MPC ensures feasibility in the short term. This modular design provides flexibility, as each controller can be optimized for its specific role without requiring significant modifications to the other. Additionally, the parallel configuration enhances robustness, as MPC can act as a stabilizing mechanism even if the RL controller encounters unexpected conditions or takes suboptimal actions during its learning process. However, coordination between the two controllers remains a challenge, as the interaction between their decisions must be carefully managed to avoid conflicting control actions. This parallel setup is particularly beneficial in systems where disturbances, uncertainties, or hybrid control structures require a combination of data-driven learning and explicit optimization, ensuring adaptability while maintaining operational safety and efficiency.

A parallel structure is explored in microgrid energy management and power distribution networks, where RL and MPC optimize different aspects of system operation while running independently but in coordination [69, 83]. In both cases, RL is responsible for high-level strategic decision-making, while MPC ensures

real-time feasibility and optimal control execution. In the microgrid energy management setup, RL learns optimal scheduling policies for energy resources, making long-term planning decisions based on past demand and supply patterns. Similarly, in power distribution networks, RL dynamically determines reserve requirements to hedge against uncertainty in renewable generation and load demand. Meanwhile, in both cases, MPC plays a critical role in ensuring feasibility by adjusting continuous control variables—either optimizing power distribution within the grid's operational constraints or managing power flow to maximize load restoration while minimizing curtailment.

By decoupling these tasks, both approaches significantly reduce computational complexity. RL determines the discrete high-level scheduling or reserve allocation in advance, enabling MPC to focus solely on refining real-time execution without solving complex mixed-integer optimization problems online. However, despite their independence, the two controllers remain tightly coupled, as RL's outputs define the constraints and decision space for MPC, while MPC's real-time optimizations provide feedback that indirectly influences RL's learning process. Ensuring consistency between RL-generated schedules or reserves and MPC's dynamic optimizations remains a key challenge, misalignment between the two can lead to inefficient energy allocation, infeasible operations, or suboptimal load restoration strategies.

A parallel RL and MPC structure is also utilized for quadrotor guidance and control, where the two controllers operate in distinct yet complementary roles [84]. In this framework, MPC is responsible for vehicle control and obstacle avoidance, ensuring that the quadrotor follows dynamically feasible trajectories within predefined constraints. Meanwhile, RL is tasked with high-level path planning, guiding the quadrotor through unknown environments where backtracking and adaptive decision-making are necessary. This division of tasks allows MPC to focus on real-time execution and constraint satisfaction while RL optimizes exploration strategies based on environmental feedback. The main advantage of this approach is its robustness in navigating complex spaces without requiring a predefined global map. However, coordination remains critical, as RL's path selection must align with MPC's control feasibility to prevent inefficient or unsafe trajectories. The combination of learning-based exploration with optimization-based control highlights the strengths of both methods, enabling adaptive navigation while maintaining stability and constraint enforcement.

The parallel operation of RL and MPC presents a promising framework that leverages the strengths of both methods without direct hierarchical control. By allowing RL to focus on long-term strategy development and MPC to ensure short-term feasibility and constraint satisfaction, this approach achieves a balance between adaptability and stability. The modular nature of this structure enables independent tuning of each controller, enhancing flexibility while reducing computational burden. While some implementations already incorporate coordination mechanisms where RL's outputs define constraints for MPC and MPC provides feedback that refines RL's decisions, ensuring seamless interaction remains a challenge. Misalignment between high-level RL strategies and real-time MPC optimizations can still lead to inefficiencies or suboptimal control actions, highlighting the need for continued refinement of integration techniques to improve consistency and performance across different applications.

The goal of this study is to develop a hybrid control framework for aeroelastic aircraft by integrating Model Predictive Control (MPC) and Reinforcement Learning (RL). The controller aims to perform flutter suppression and load alleviation by optimizing tracking accuracy, enhancing disturbance rejection, improving settling characteristics and maintaining stability under severe gust disturbances, all while satisfying system constraints. The study begins by evaluating standalone MPC and RL controllers to identify their respective limitations, which then guide the design of the hybrid framework. Final performance is evaluated through 1000 Monte Carlo runs to assess robustness and effectiveness.

3

Research Questions

This research aims to develop a real-time control framework for aeroelastic aircraft, integrating MPC and RL to enhance stability, adaptability, and performance under severe gusts and rapid state changes. Specifically, the research objective is as follows:

Research Objective

The goal of this study is to develop a hybrid control framework for aeroelastic aircraft by integrating Model Predictive Control (MPC) and Reinforcement Learning (RL). The controller aims to perform flutter suppression and load alleviation by optimizing tracking accuracy, enhancing disturbance rejection, improving settling characteristics and maintaining stability under severe gust disturbances, all while satisfying system constraints. The study begins by evaluating standalone MPC and RL controllers to identify their respective limitations, which then guide the design of the hybrid framework. Final performance is evaluated through 1000 Monte Carlo runs to assess robustness and effectiveness.

The objective is divided into three main research questions, each addressing a different phase of the work, from evaluating standalone controllers to developing the hybrid approach and validating its effectiveness.

Research Question 1

How do standalone MPC and RL controllers perform for aeroelastic aircraft under gust disturbances, and what are their respective strengths and limitations?

Research Question 2

What are the most effective strategies for combining MPC and RL for aeroelastic control, considering safety and operational constraints, fast aeroelastic responses, and the computational demands of online implementation?

Research Question 3

How can the proposed hybrid MPC-RL approach be theoretically justified for safety and empirically validated through computer-based simulations for both safety and performance, and does it provide a measurable improvement over standalone controllers in aeroelastic control scenarios?

4

Project Plan

4.1. Methodology

The methodology consists of five major activities:

1. **Literature survey.** Conduct a thorough review of existing MPC and RL techniques applied to aeroelastic control, identifying each approach's strengths and limitations in handling gust disturbances, nonlinear elasticity, and real□time constraints.
2. **RL-MPC combinations exploration.** Analyze state-of-the-art methods for combining MPC and RL, to determine which architectures best address the specific requirements of aeroelastic□wing gust rejection.
3. **Baseline controller implementation.** Develop two independent simulation models: one using a standard MPC scheme, the other using an RL policy trained on generic stability/performance objectives.
4. **Limitation analysis and integration concept.** Evaluate the two baselines to pinpoint where each falls short (e.g. in disturbance rejection or constraint enforcement), then find which MPC–RL integration from the literature best addresses those specific gaps.
5. **Hybrid controller development and evaluation.** Build the combined controller, then conduct systematic simulation tests—varying gust profiles and operating points—to assess improvements in stability, tracking, and constraint satisfaction relative to the baselines.

4.2. Expected Results

With the proposed methodology, the expected results are the following:

- A comprehensive review of current MPC and RL approaches for aeroelastic disturbance rejection, including identification of complementary features and integration opportunities.
- Standalone simulation implementations of an MPC controller and an RL policy, each tuned for gust and aeroelastic disturbance mitigation.
- A two-stage MPC–RL controller that combines the safety guarantees of MPC with the adaptability of RL into a unified real□time policy.
- A formal demonstration that every action selected by the hybrid controller respects the prescribed actuator and state constraints.
- A MATLAB framework comparing the hybrid design against the two baselines, yielding quantitative evidence of improved disturbance rejection, tighter tracking, and preserved safety margins.

4.3. Planning

The detailed project planning can be seen in the Gantt chart on the following page.

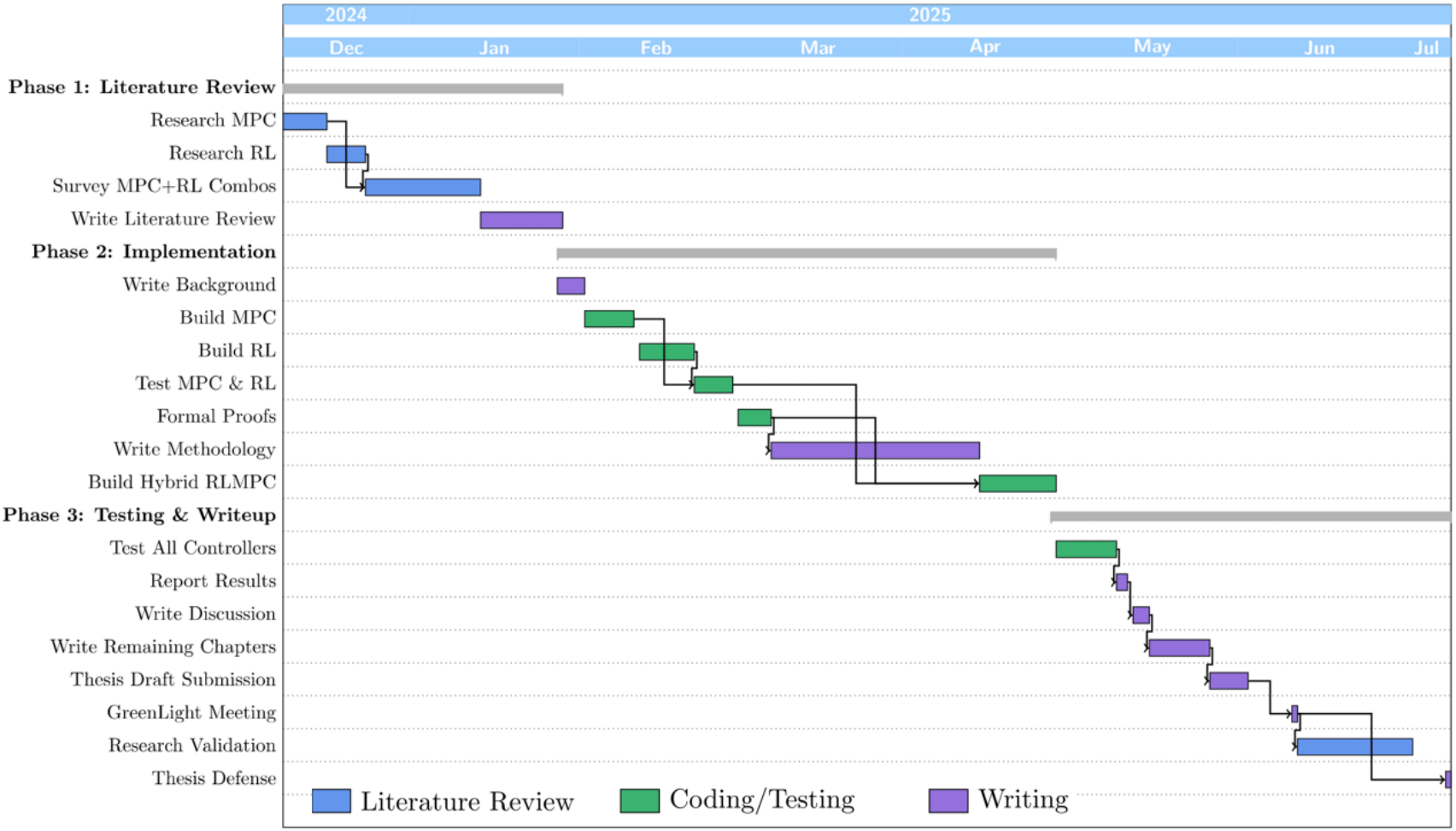


Figure 4.1: Project Timeline

References

[1] Qinfeng Guo et al. “Effects of Wing Flexibility on Aerodynamic Performance of an Aircraft Model”. In: *Chinese Journal of Aeronautics* 34.9 (2021), pp. 133–142. DOI: 10.1016/j.cja.2021.01.012. URL: https://doi.org/10.1016/j.cja.2021.01.012.

[2] Tobias Franziskus Wunderlich et al. “Global Aerostructural Design Optimization of More Flexible Wings for Commercial Aircraft”. In: *Journal of Aircraft* 58.849 (2021). DOI: 10.2514/1.C036301. URL: https://doi.org/10.2514/1.C036301.

[3] Eli Livne. *Aircraft Active Flutter Suppression — State of the Art and Technology Maturation Needs*. Final Report DOT/FAA/TC-18/47. Available through the National Technical Information Services (NTIS), Springfield, Virginia 22161. Also available from the Federal Aviation Administration William J. Hughes Technical Center at actlibrary.tc.faa.gov. University of Washington, William E. Boeing Department of Aeronautics and Astronautics, Guggenheim Hall, Suite 211, Seattle, WA 98195-2400: U.S. Department of Transportation, Federal Aviation Administration, William J. Hughes Technical Center, June 2019, p. 106.

[4] Tobias Franziskus Wunderlich et al. “Global Aerostructural Design Optimization of More Flexible Wings for Commercial Aircraft”. In: *Journal of Aircraft* 58.849 (2021). DOI: 10.2514/1.C036301. URL: https://doi.org/10.2514/1.C036301.

[5] Yuyang Chai et al. “Aeroelastic Analysis and Flutter Control of Wings and Panels: A Review”. In: *International Journal of Mechanical System Dynamics (IJMSD)* (Nov. 2021). Open Access, Review Article. DOI: 10.1002/msd2.12015. URL: https://doi.org/10.1002/msd2.12015.

[6] William L. Garrard et al. “Active Flutter Suppression Using Eigenspace and Linear Quadratic Design Techniques”. In: *Journal of Guidance, Control, and Dynamics* 8.3 (1985), pp. 304–311. DOI: 10.2514/3.19980. URL: https://doi.org/10.2514/3.19980.

[7] Labane Chrif et al. “Aircraft Control System Using LQG and LQR Controller with Optimal Estimation–Kalman Filter Design”. In: *Procedia Engineering* 80 (2014), pp. 245–257. DOI: 10.1016/j.proeng.2014.09.084. URL: https://doi.org/10.1016/j.proeng.2014.09.084.

[8] Eduardo F. Camacho et al. *Model Predictive Control*. 2nd. Springer, 2004. DOI: 10.1007/978-3-319-24853-0.

[9] Brian D. O. Anderson et al. *Optimal Control: Linear Quadratic Methods*. [Edition unavailable]. Dover Publications, 2014. URL: https://www.perlego.com/book/1444508/optimal-control-linear-quadratic-methods-pdf.

[10] Guilherme V. Pereira et al. “Model Predictive Control for Maneuver Load Alleviation in Flexible Airliners”. In: *AIAA Atmospheric Flight Mechanics Conference*. 2019. DOI: 10.2514/6.2019-1109.

[11] Tianyi He et al. “Robust control of gust-induced vibration of highly flexible aircraft”. In: *Aerospace Science and Technology* 143 (2023). DOI: 10.1016/j.ast.2023.108703.

[12] Richard S. Sutton et al. *Reinforcement Learning: An Introduction*. 2nd. MIT Press, 1998. URL: http://incompleteideas.net/book/the-book-2nd.html.

[13] Thomas Chaffre et al. “Learning-based vs Model-free Adaptive Control of a MAV under Wind Gust”. In: *arXiv preprint arXiv:2101.12501* (2021). DOI: 10.48550/arXiv.2101.12501.

[14] Sohrab Haghighat et al. “A Model Predictive Gust Load Alleviation Controller for a Highly Flexible Aircraft”. In: *Journal of Guidance, Control, and Dynamics* 35.5 (2012), pp. 1501–1512. DOI: 10.2514/1.57013. URL: https://doi.org/10.2514/1.57013.

[15] Patrick Hoffmann et al. “Comparison of Reinforcement Learning and Model Predictive Control for Automated Generation of Optimal Control for Dynamic Systems within a Design Space Exploration Framework”. In: *International Journal of Automotive Engineering* 15.1 (2024). License: CC BY-NC-SA 4.0, pp. 19–26. DOI: 10.20485/jsaejae.15.1_19. URL: https://doi.org/10.20485/jsaejae.15.1_19.

[16] Ali Mesbah et al. “Fusion of Machine Learning and MPC under Uncertainty: What Advances Are on the Horizon?” In: *Proceedings of the 2022 American Control Conference (ACC)*. Date of Conference: 08–10 June 2022. IEEE. Atlanta, GA, USA: IEEE, 2022. DOI: 10.23919/ACC53348.2022.9867643.

[17] Yating Hu et al. “Reinforcement learning for gust load control of an elastic wing via camber morphing at arbitrary sinusoidal gusts”. In: *Aerospace Science and Technology* 162 (2025), p. 110174. DOI: 10.1016/j.ast.2025.110174. URL: https://doi.org/10.1016/j.ast.2025.110174.

[18] Dingcui Huang et al. “Model-free Based Reinforcement Learning Control Strategy of Aircraft Attitude Systems”. In: *2020 Chinese Automation Congress (CAC)*. IEEE, 2020, pp. 1–6. DOI: 10.1109/CAC51589.2020.9327563.

[19] R. Konatala et al. “Flight Testing Reinforcement Learning based Online Adaptive Flight Control Laws on CS-25 Class Aircraft”. In: *Proceedings of the AIAA SCITECH 2024 Forum*. American Institute of Aeronautics and Astronautics (AIAA), 2024. DOI: 10.2514/6.2024-2402.

[20] Sébastien Gros et al. “Reinforcement Learning based on MPC and the Stochastic Policy Gradient Method”. In: *Proceedings of the 2021 American Control Conference (ACC)*. Conference Dates: 25-28 May 2021, Added to IEEE Xplore on 28 July 2021. New Orleans, LA, USA: IEEE, 2021. DOI: 10.23919/ACC50511.2021.9482765. URL: https://doi.org/10.23919/ACC50511.2021.9482765.

[21] Saket Adhau et al. “Reinforcement learning based MPC with neural dynamical models”. In: *European Journal of Control* 101 (2024). Open Access under Creative Commons license. DOI: 10.1016/j.ejcon.2024.101048. URL: https://doi.org/10.1016/j.ejcon.2024.101048.

[22] Katrine Seel et al. “Variance-Based Exploration for Learning Model Predictive Control”. In: *IEEE Access* 11 (2023). Published under Creative Commons License, pp. 60724–60736. DOI: 10.1109/ACCESS.2023.3282842. URL: https://doi.org/10.1109/ACCESS.2023.3282842.

[23] Hossein Nejatbakhsh Esfahani et al. “Approximate Robust NMPC using Reinforcement Learning”. In: *arXiv preprint arXiv:2104.02743* (2021). Accepted to 2021 European Control Conference (ECC). URL: https://doi.org/10.48550/arXiv.2104.02743.

[24] Max Schwenzer et al. “Review on model predictive control: an engineering perspective”. In: *The International Journal of Advanced Manufacturing Technology* 117.5 (2021), pp. 1327–1349.

[25] David Q. Mayne et al. “Constrained model predictive control: Stability and optimality”. In: *Automatica* 36.6 (2000), pp. 789–814.

[26] Frank Allgöwer et al. “Nonlinear model predictive control”. In: *Assessment and Future Directions of Nonlinear Model Predictive Control*. Springer, 2012, pp. 3–16. DOI: 10.1007/978-3-0348-8407-5_21.

[27] Juan Ferrer et al. “Linear Parameter Varying Model Predictive Control: A review on stability, feasibility, and applications”. In: *Applied Sciences* 11.10 (2021), p. 4368. DOI: 10.3390/app11104368.

[28] Liuping Wang. *Model Predictive Control System Design and Implementation Using MATLAB*. Springer, 2009.

[29] Wei Gao et al. “Gust load alleviation of a flexible flying wing with linear parameter-varying modeling and model predictive control”. In: *Aerospace Science and Technology* 155 (2024), p. 109671. DOI: 10.1016/j.ast.2024.109671.

[30] R. Bittner et al. “Model Predictive Control for Maneuver Load Alleviation”. In: *IFAC Proceedings Volumes*. Vol. 45. 9. 2012, pp. 199–204. DOI: 10.3182/20120823-5-NL-3013.00049.

[31] Tianyi He et al. “Gust Alleviation of Highly Flexible Aircraft with Model Predictive Control”. In: *AIAA SCITECH 2023 Forum*. 2023. DOI: 10.2514/6.2023-0586.

[32] G. V. Pereira et al. “Model Predictive Control Architectures for Maneuver Load Alleviation in Very Flexible Aircraft”. In: *AIAA Scitech 2019 Forum*. 2019. DOI: 10.2514/6.2019-1591.

[33] Marco Tito Bordogna et al. “Static and Dynamic Aeroelastic Tailoring with Composite Blending and Manoeuvre Load Alleviation”. In: *Structural and Multidisciplinary Optimization* 61 (2020), pp. 2193–2216. DOI: 10.1007/s00158-019-02446-w.

[34] Abdel R Darwich Ajjour. “Gust and Manoeuvre Loads Alleviation Using Upper and Lower Surface Spoilers”. PhD thesis. University of Bristol, 2022. URL: https://research-information.bris.ac.uk/en/studentTheses/gust-and-manoeuvre-loads-alleviation-using-upper-and-lower-surfac.

[35] Tianyi He et al. “Smooth-switching LPV control for vibration suppression of a flexible airplane wing”. In: *Aerospace Science and Technology* 84 (2019), pp. 895–903. DOI: 10.1016/j.ast.2018.11.029.

[36] Yinan Wang et al. “Model-Predictive Control of Flexible Aircraft Dynamics using Nonlinear Reduced-Order Models”. In: *AIAA SciTech* (2016). DOI: 10.2514/6.2016-0711.

[37] Benjamin Herrmann et al. “Flight Testing of Real-Time Model Predictive Flight Control for Unmanned Flexible Aircraft”. In: *EuroGNC Conference, CEAS 2024*. Conference Dates: 11-13 June 2024. Bristol, United Kingdom: Council of European Aerospace Societies (CEAS), June 2024, p. 054. URL: https://eurognc.ceas.org/archive/EuroGNC2024/pdf/CEAS-GNC-2024-054.pdf.

[38] Jan M. Maciejowski et al. “MPC Fault-Tolerant Flight Control Case Study: Flight 1862”. In: *IFAC Proceedings Volumes* 36.5 (June 2003), pp. 119–124. DOI: 10.1016/S1474-6670(17)36480-7. URL: https://doi.org/10.1016/S1474-6670(17)36480-7.

[39] Shen Qu et al. “Mixed Model Predictive Control and Data-Driven Control of a Tiltrotor eVTOL Aircraft”. In: *AIAA SCITECH 2024 Forum*. Orlando, FL, Jan. 2024. DOI: 10.2514/6.2024-0517. URL: https://doi.org/10.2514/6.2024-0517.

[40] Mateus de F.V. Pereira et al. “Model Predictive Control with Constraint Aggregation Applied to Conventional and Very Flexible Aircraft”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019. DOI: 10.1109/CDC.2019.9029614. URL: https://ieeexplore.ieee.org/document/9029614.

[41] Leif Rieck et al. “Efficient Quasi-Linear Model Predictive Control of a Flexible Aircraft Based on Laguerre Functions”. In: *2023 American Control Conference (ACC)*. IEEE, 2023. DOI: 10.23919/ACC2023.1234567. URL: https://ieeexplore.ieee.org/document/1234567.

[42] A. Orani et al. “Model predictive control for actuator delay compensation in vehicle stability control”. In: *Mechatronics* 56 (2019), pp. 59–68. DOI: 10.1016/j.mechatronics.2018.11.007.

[43] David Silver et al. “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676 (2017), pp. 354–359. DOI: 10.1038/nature24270.

[44] Ilge Akkaya et al. “Solving Rubik’s Cube with a robot hand”. In: *arXiv preprint arXiv:1910.07113* (2019).

[45] Zihan Ding et al. “Introduction to Reinforcement Learning”. In: *Deep Reinforcement Learning: Fundamentals, Research, and Applications*. Ed. by Hao Dong et al. Springer Nature, 2020. Chap. 2, pp. 47–124. URL: http://www.deepreinforcementlearningbook.org.

[46] Phillip Swazinna et al. “Comparing Model-Free and Model-Based Algorithms for Offline Reinforcement Learning”. In: *IFAC-PapersOnLine* 55.15 (2022), pp. 19–26. DOI: 10.1016/j.ifacol.2022.07.602.

[47] Matthias Lehmann. “The Definitive Guide to Policy Gradients in Deep Reinforcement Learning: Theory, Algorithms and Implementations”. In: *arXiv preprint arXiv:2401.13662* (2024). URL: https://doi.org/10.48550/arXiv.2401.13662.

[48] Giacomo Arcieri et al. “A Comparison of Value-Based and Policy-Based Reinforcement Learning for Monitoring-Informed Railway Maintenance Planning”. In: *Proceedings of the International Workshop on Structural Health Monitoring (IWSHM)*. ETH Zurich, 2023. DOI: 10.3929/ethz-b-000635011. URL: https://doi.org/10.3929/ethz-b-000635011.

[49] Mohsen Zahmatkesh et al. “Attitude Control of Highly Maneuverable Aircraft Using an Improved Q-learning”. In: *arXiv preprint arXiv:2210.12317* (2022). URL: https://doi.org/10.48550/arXiv.2210.12317.

[50] Ali Barzegar et al. “Deep Reinforcement Learning-Based Adaptive Controller for Trajectory Tracking and Altitude Control of an Aerial Robot”. In: *Applied Sciences* 12.9 (2022), p. 4764. DOI: 10.3390/app12094764. URL: https://doi.org/10.3390/app12094764.

[51] Killian Dally et al. “Soft Actor-Critic Deep Reinforcement Learning for Fault Tolerant Flight Control”. In: *AIAA SCITECH 2022 Forum*. 2022. DOI: 10.2514/6.2022-2078. arXiv: 2202.09262.

[52] Shaolong Yang et al. “Reinforcement Learning Based Attitude Fault-Tolerant Control of Spacecraft with Unknown System Model”. In: *SSRN* (2024). Posted: 11 Jan 2024, p. 33. URL: https://ssrn.com/abstract=XXXXXXX.

[53] Agostino De Marco et al. “A deep reinforcement learning control approach for high-performance aircraft”. In: *Nonlinear Dynamics* 111 (Aug. 2023), pp. 17037–17077. DOI: 10.1007/s11071-023-08725-y. URL: https://link.springer.com/article/10.1007/s11071-023-08725-y.

[54] K. Himanshu et al. “Waypoint Navigation of Quadrotor using Deep Reinforcement Learning”. In: *IFAC-PapersOnLine* 55 (22 2022), pp. 281–286. DOI: 10.1016/j.ifacol.2023.03.047. URL: https://doi.org/10.1016/j.ifacol.2023.03.047.

[55] Zhao Yu et al. “Reinforcement Learning-Based Collision Avoidance Guidance Algorithm for Fixed-Wing UAVs”. In: *Complexity* 2021 (2021), pp. 1–12. DOI: 10.1155/2021/8818013. URL: https://doi.org/10.1155/2021/8818013.

[56] Jinghang Li et al. “Automatic Landing Control for Fixed-Wing UAV in Longitudinal Channel Based on Deep Reinforcement Learning”. In: *Drones* 8.10 (2024), p. 568. DOI: 10.3390/drones8100568. URL: https://doi.org/10.3390/drones8100568.

[57] Kevin PT Haughn et al. “Deep reinforcement learning reveals fewer sensors are needed for autonomous gust alleviation”. In: *arXiv preprint arXiv:2304.03133* (2023). DOI: 10.48550/arXiv.2304.03133.

[58] Esteban A. Hufstedler et al. “Loads Alleviation on an Airfoil via Reinforcement Learning”. In: *AIAA Scitech 2019 Forum*. San Diego, California: American Institute of Aeronautics and Astronautics, Jan. 2019. DOI: 10.2514/6.2019-0404. URL: https://doi.org/10.2514/6.2019-0404.

[59] Robert Clarke et al. “Reinforcement Learning Derived High-Alpha Aerobatic Manoeuvres for Fixed Wing Operation in Confined Spaces”. In: *Algorithms* 16.8 (Aug. 2023), p. 384. DOI: 10.3390/a16080384. URL: https://doi.org/10.3390/a16080384.

[60] Zhen Chen et al. “Active flutter suppression for a flexible wing model with trailing-edge circulation control via reinforcement learning”. In: *AIP Advances* 13.1 (2023), p. 015317. DOI: 10.1063/5.0130370.

[61] Jose Antonio Martín H. et al. “Reinforcement Learning in System Identification”. In: *arXiv preprint* arXiv:2212.07123 (2022). DOI: 10.48550/arXiv.2212.07123. URL: https://doi.org/10.48550/arXiv.2212.07123.

[62] Saviz Mowlavi et al. “Reinforcement Learning State Estimation for High-Dimensional Nonlinear Systems”. In: *ICLR 2022 Conference (Withdrawn Submission)* (Sept. 2021). Modified: 14 February 2023.

[63] Nathan Schaff. “Online Aircraft System Identification Using a Novel Parameter Informed Reinforcement Learning Method”. In: *Engineering Applications of Artificial Intelligence* (2023). URL: https://commons.erau.edu/edt/779/.

[64] Rudolf Reiter et al. “Synthesis of Model Predictive Control and Reinforcement Learning: Survey and Classification”. In: *arXiv preprint arXiv:2502.02133* (2025). arXiv:2502.02133v1 [eess.SY]. DOI: 10.48550/arXiv.2502.02133. URL: https://doi.org/10.48550/arXiv.2502.02133.

[65] Mario Zanon et al. “Safe Reinforcement Learning Using Robust MPC”. In: *IEEE Transactions on Automatic Control* (2021). arXiv:1906.04005v2 [eess.SY]. DOI: 10.1109/TAC.2020.3024161. URL: https://doi.org/10.48550/arXiv.1906.04005.

[66] Samuel Mallick et al. “Multi-Agent Reinforcement Learning via Distributed MPC as a Function Approximator”. In: *arXiv preprint arXiv:2312.05166* (2024). Accepted for publication in Automatica, arXiv:2312.05166v4 [eess.SY]. DOI: 10.48550/arXiv.2312.05166. URL: https://doi.org/10.48550/arXiv.2312.05166.

[67] Dingshan Sun et al. “Adaptive parameterized model predictive control based on reinforcement learning: A synthesis framework”. In: *Engineering Applications of Artificial Intelligence* 109 (2024). Published under a Creative Commons license, p. 109009. DOI: 10.1016/j.engappai.2024.109009. URL: https://doi.org/10.1016/j.engappai.2024.109009.

[68] Prakhar Gupta et al. “Reinforcement Learning Compensated Model Predictive Control for Off-road Driving on Unknown Deformable Terrain”. In: *arXiv:2408.09253 [cs.RO]* (2024). Submitted on 17 Aug 2024. URL: https://doi.org/10.48550/arXiv.2408.09253.

[69] Caio Fabio Oliveira da Silva et al. “Integrating Reinforcement Learning and Model Predictive Control with Applications to Microgrids”. In: *arXiv preprint arXiv:2409.11267* (2024). arXiv: 2409.11267 [eess.SY]. URL: https://doi.org/10.48550/arXiv.2409.11267.

[70] Mario Zanon et al. “Reinforcement Learning Based on Real-Time Iteration NMPC”. In: *arXiv preprint arXiv:2005.05225* (2020). Accepted for the IFAC World Congress 2020, arXiv:2005.05225v1 [eess.SY]. DOI: 10.48550/arXiv.2005.05225. URL: https://doi.org/10.48550/arXiv.2005.05225.

[71] Sébastien Gros et al. “Towards Safe Reinforcement Learning Using NMPC and Policy Gradients: Part I - Stochastic case”. In: *arXiv preprint arXiv:1906.04057* (2019). Published in arXiv:1906.04057v1 [eess.SY]. DOI: 10.48550/arXiv.1906.04057. URL: https://doi.org/10.48550/arXiv.1906.04057.

[72] Arash Bahari Kordabad et al. “Reinforcement Learning for MPC: Fundamentals and Current Challenges”. In: *IFAC-PapersOnLine* 56.2 (2023). Published in January 2023, pp. 5773–5780. DOI: 10.1016/j.ifacol.2023.10.548. URL: https://doi.org/10.1016/j.ifacol.2023.10.548.

[73] Sébastien Gros et al. “Safe Reinforcement Learning via Projection on a Safe Set: How to Achieve Optimality?” In: *arXiv preprint arXiv:2004.00915* (2020). Accepted at IFAC 2020, arXiv:2004.00915v1 [eess.SY]. DOI: 10.48550/arXiv.2004.00915. URL: https://doi.org/10.48550/arXiv.2004.00915.

[74] Filippo Airaldi et al. “Reinforcement Learning with Model Predictive Control for Highway Ramp Metering”. In: *arXiv preprint arXiv:2311.08820* (2023). Submitted to IEEE Transactions on Intelligent Transportation Systems. URL: https://doi.org/10.48550/arXiv.2311.08820.

[75] Katrine Seel et al. “Combining Q-learning and Deterministic Policy Gradient for Learning-Based MPC”. In: *2023 62nd IEEE Conference on Decision and Control (CDC)*. Singapore, Singapore: IEEE, 2023, p. 10383562. DOI: 10.1109/CDC49753.2023.10383562. URL: https://doi.org/10.1109/CDC49753.2023.10383562.

[76] Samuel Mallick et al. “Reinforcement learning-based model predictive control for greenhouse climate control”. In: *Smart Agricultural Technology* 10 (2025). Under a Creative Commons license, Open access, p. 100751. DOI: 10.1016/j.atech.2024.100751. URL: https://doi.org/10.1016/j.atech.2024.100751.

[77] Sébastien Gros et al. “Data-driven Economic NMPC using Reinforcement Learning”. In: *arXiv preprint arXiv:1904.04152* (2019). Published in IEEE Transactions on Automatic Control, vol. 65, no. 2, pp. 636-648, Feb. 2020, arXiv:1904.04152v1 [cs.SY]. DOI: 10.48550/arXiv.1904.04152. URL: https://doi.org/10.48550/arXiv.1904.04152.

[78] Andreas B. Martinsen et al. “Combining system identification with reinforcement learning-based MPC”. In: *arXiv preprint arXiv:2004.03265* (2020). Accepted to the IFAC 2020, arXiv:2004.03265v1

[eess.SY]. DOI: 10.48550/arXiv.2004.03265. URL: https://doi.org/10.48550/arXiv.2004.03265.

[79] Shen Qu et al. “Mixed Model Predictive Control and Data-Driven Control of a Tiltrotor eVTOL Aircraft”. In: *AIAA 2024-0517: Control Techniques for AAM Autonomy*. Published Online: 4 Jan 2024. AIAA, 2024. DOI: 10.2514/6.2024-0517. URL: https://doi.org/10.2514/6.2024-0517.

[80] Seung Hyeon Bang et al. “RL-augmented MPC Framework for Agile and Robust Bipedal Footstep Locomotion Planning and Control”. In: *arXiv:2407.17683 [cs.RO]* (2024). Submitted on 25 Jul 2024. URL: https://doi.org/10.48550/arXiv.2407.17683.

[81] Willemijn Remmerswaal et al. “Combined MPC and Reinforcement Learning for Traffic Signal Control in Urban Traffic Networks”. In: *Proceedings of the 26th International Conference on System Theory, Control and Computing, ICSTCC 2022*. IEEE, 2022, pp. 432–439. DOI: 10.1109/ICSTCC55426.2022.9931771.

[82] Dingshan Sun et al. “A Novel Framework Combining MPC and Deep Reinforcement Learning With Application to Freeway Traffic Control”. In: *IEEE Transactions on Intelligent Transportation Systems* 25.7 (2024). Published under a Creative Commons License, pp. 6756–6769. DOI: 10.1109/TITS.2023.3342651.

[83] Abinet Tesfaye Eseye et al. “A Hybrid Reinforcement Learning-MPC Approach for Distribution System Critical Load Restoration”. In: *2022 IEEE Power & Energy Society General Meeting (PESGM)*. Denver, CO, USA: IEEE, 2022, pp. 1–5. DOI: 10.1109/PESGM48719.2022.9916743. URL: https://doi.org/10.1109/PESGM48719.2022.9916743.

[84] Colin Greatwood et al. “Reinforcement learning and model predictive control for robust embedded quadrotor guidance and control”. In: *Autonomous Robots* 43 (Oct. 2019). DOI: 10.1007/s10514-019-09829-4.

Part II

Scientific Article

Constraint-Safe Gust Rejection for Nonlinear Aeroelasitc Wings via MPC-Driven Reinforcement Learning

P. Kostelac *
Delft University of Technology, Faculty of Aerospace Engineering, 2629 HS Delft, The Netherlands

The evolution of aerospace engineering has led to aircraft with lightweight, flexible structures that improve fuel and aerodynamic performance. However, this flexibility increases sensitivity to gust disturbances, demanding control systems that ensure safety and performance under strict real-time constraints. These challenges highlight the need for new control architectures capable of robust, adaptive, and constraint-aware behavior in nonlinear, time-varying environments. Linear Parameter Varying Model Predictive Control (LPV-MPC) has emerged as a promising tool for aeroelastic aircraft control due to its structured constraint handling and robustness. However, its effectiveness can be limited under turbulence and real-time constraints. RL offers adaptive, lightweight control in uncertain nonlinear environments, but lacks built-in safety guarantees. Their complementary strengths have motivated hybrid approaches combining reliability with flexibility. However, most current integrations either embed MPC within RL or use RL to tune MPC, without fully leveraging their complementary strengths. This paper proposes a novel architecture where MPC computes safe control bounds during training that bracket the system's response. These bounds guide an RL agent, which learns to select effective actions within the safe region, ensuring constraint satisfaction by construction. The approach leverages long-horizon MPC during training to enable fast, lightweight execution at deployment. Simulations show improved gust rejection, constraint satisfaction, and reduced online computational load compared to standalone controllers.

Nomenclature

$\boldsymbol{x}_k$	=	state vector at time step k, $[x_1, x_2, x_3, x_4, x_5]^\top = [h_k, \alpha_k, \dot{h}_k, \dot{\alpha}_k, \beta_k]^\top$
$x_1 ... x_5$	=	individual states: $x_1 = h$ (m), $x_2 = \alpha$ (rad), $x_3 = \dot{h}$ (m/s), $x_4 = \dot{\alpha}$ (rad/s), $x_5 = \beta$ (rad)
v_k^h, a_k^h, j_k^h, s_k^h	=	plunge velocity, acceleration, jerk, snap (m/s, m/s^2, m/s^3, m/s^4)
v_k^α, a_k^α, $j_{\alpha,k}$, s_k^α	=	pitch velocity, acceleration, jerk, snap (rad/s, rad/s^2, rad/s^3, rad/s^4)
$\partial v_k^h / \partial x_i$, $\partial v_k^\alpha / \partial x_i$	=	partial derivatives of velocity w.r.t. state variable x_i

*MSc. Student, Faculty of Aerospace Engineering, Control and Simulation Division, Delft University of Technology.

$\partial a_k^h/\partial x_i$, $\partial a_k^\alpha/\partial x_i$	=	partial derivatives of acceleration w.r.t. state variable x_i
u^{flap}	=	commanded flap deflection (rad)
$\lvert\Delta u_k\rvert$	=	per-step elevator increment $\lvert u_k - u_{k-1}\rvert$ (rad)
k_β	=	flap actuator bandwidth (s^{-1})
M^{mass}	=	structural inertia matrix (kg or kg·m^2)
C	=	damping matrix (kg/s or kg·m^2/s)
K	=	stiffness matrix (N/m or Nm/rad)
L^{aero}	=	aerodynamic lift force (N)
M^{aero}	=	aerodynamic pitching moment (N·m)
$\boldsymbol{f}_k$	=	system dynamics evaluated at $\boldsymbol{x}_k$, $f(\boldsymbol{x}_k, u_k, w_k)$
$\nabla_x \boldsymbol{f}_k$	=	Jacobian of dynamics f w.r.t. x at time k
w_k	=	turbulence at time step k (m/s)
ρ	=	air density (kg/m^3)
U	=	freestream airspeed (m/s)
$c_{L\alpha}$, $c_{L\beta}$	=	lift coefficients per angle of attack and flap deflection (1/rad)
$c_{m\alpha}$, $c_{m\beta}$	=	moment coefficients per angle of attack and flap deflection (1/rad)
α^{eff}	=	effective angle of attack (rad)
a_b	=	aerodynamic center offset coefficient (unitless)
b	=	wing chord (m)
Δt	=	discrete time step (s)
T_s	=	controller sampling period (s)
N^{P}, N^{C}, N^{delay}	=	prediction, control, and delay compensation horizons (steps)
Q, R	=	state and input weighting matrices (dimensionless)
k_{NN}	=	number of neighbours in k-NN policy (–)
ε	=	small inverse-distance regulariser (–)
L_x, L_u	=	Lipschitz constants for state and input (–)
e_h, e_α, $e_{h,\text{RMS}}$, $e_{\alpha,\text{RMS}}$	=	instantaneous and RMS errors in h, α (m, rad)
O_h, O_α	=	maximum overshoot in h, α (mm, rad)
$T_{s,h}$, $T_{s,\alpha}$	=	settling times in h, α (s)
E^h_{Low}, E^h_{Mid}, E^h_{High}	=	plunge error energy in 0.1–1, 1–5, 5–50 Hz bands (mm^2·s)
E^α_{Low}, E^α_{Mid}, E^α_{High}	=	AoA error energy in 0.1–1, 1–5, 5–50 Hz bands (deg^2·s)

I. Introduction

The evolution of aerospace engineering has ushered in a new era of aircraft design, characterized by lightweight, flexible wings aimed at improving fuel efficiency and maneuverability [1, 2]. While beneficial, these structures introduce significant control challenges due to high-order, nonlinear dynamics and strong coupling between structural deformation and unsteady aerodynamics. Flexible wings are particularly susceptible to aeroelastic phenomena such as flutter, divergence, and control reversal, which threaten both structural integrity and overall flight safety [3]. These risks are further exacerbated under atmospheric turbulence, where sudden gusts introduce rapidly varying loads that demand precise real-time control despite actuator delays and strict input and state constraints. Traditional control frameworks, often fall short in addressing these coupled demands, particularly when nonlinearities, transport delays, and constraint satisfaction must all be handled simultaneously [3]. To overcome these limitations, a hybrid RL–MPC controller is developed by combining long-horizon LPV-MPC rollouts with a lightweight, constraint-aware Q-learning policy. The result is a turbulence-resilient, delay-compensating controller capable of achieving real-time flutter suppression and load alleviation while respecting both computational and physical constraints.

Existing methods span from classical linear techniques to advanced nonlinear controllers, yet few are equipped to handle the full spectrum of modern aeroelastic challenges. Linear controllers such as LQR, LQG, MPC and $\mathcal{H}_\infty$ [4–6] remain widely used due to their simplicity, fast computation, and ease of gain tuning. Paired with gain scheduling [7], they offer limited adaptability to changing conditions but rely heavily on accurate linearizations. In highly flexible wings subject to turbulence and delay, linear control assumptions often fail, resulting in model inaccuracies performance degradation [8, 9].

Nonlinear control methods such as Nonlinear Dynamic Inversion (NDI), Incremental Nonlinear Dynamic Inversion (INDI), and backstepping [10–12] offer improved handling of nonlinear dynamics but pose serious challenges in practical aeroelastic control. NDI depends on precise model knowledge and is sensitive to internal dynamic instabilities and inversion singularities, which are problematic in complex aeroelastic systems. INDI reduces model dependency but relies on accurate and delay-free measurement of state derivatives and control effectiveness, which are difficult to guarantee under turbulence and sensor noise. Backstepping, while systematic and theoretically robust, suffers from the so-called explosion of complexity in high-order systems, leading to computational burdens and difficulties in implementation. Moreover, all three methods typically assume full state availability and ideal actuator behavior assumptions that break down under the delays, uncertainties, and input constraints characteristic of aeroelastic models. These limitations highlight the need for control strategies that offer nonlinear robustness while remaining feasible for real-time deployment with constrained sensing and actuation.

One promising solution that bridges the gap between linear controllers and full nonlinear designs is Linear Parameter-Varying (LPV) control. Structured approaches like LPV-MPC have gained traction in flexible aircraft control, offering constraint handling, multivariable coordination, and real-time adaptability across operating regimes [13–15]. These methods have shown promise in load alleviation and flutter suppression [14–17]. However, many imple-

mentations rely on reduced-order models [18], or use preview-based gust estimation [19], assumptions that may break down in realistic turbulent conditions. Furthermore, LPV-MPC strategies often neglect nonlinear actuator transport delay [20], suffer from model mismatch, and their performance hinges on accurate linearization quality, all of which are challenged under turbulence and fast dynamics [20, 21]. LPV-MPC also suffers from substantial computational demands due to repeated real-time optimization and parameter updates, which limits its practicality in onboard, safety-critical systems [22].

On the other end of the spectrum, RL has emerged as a data-driven alternative capable of handling nonlinear dynamics, model mismatch, and uncertain environments through adaptive policy learning [23, 24]. Its ability to bypass explicit modeling makes it attractive for aeroelastic aircraft, where accurate system identification is often infeasible. RL has demonstrated promise across tasks such as gust load alleviation, adaptive flight control, and fault-tolerant operation [25–27]. Additionally, RL policies can be lightweight to deploy, especially when using simple architectures such as Q-learning [28]. However, RL often lacks inherent guarantees for stability and constraint satisfaction, making it ill-suited for direct deployment in safety-critical settings [24, 25]. These contrasting characteristics reveal a natural synergy: MPC offers stability, constraint satisfaction, and structured decision-making, while RL provides a data-driven approach capable of handling nonlinear dynamics, model mismatch, and uncertain environments. Together, they form a complementary framework well-suited for addressing the demands of real-time aeroelastic control under uncertainty and delay.

While hybrid architectures combining RL and MPC have been explored, they are typically applied to slow or structured systems such as energy networks [29] or traffic [30]. The most common strategies embed MPC as a policy within RL for safety and interpretability [31–33], or use RL to tune MPC parameters like weights and horizons for improved adaptability [34–36]. These approaches often rely on one-way integration and do not fully leverage the complementary strengths of both methods. As a result, they fall short in addressing the demands of fast, nonlinear systems like aeroelastic aircraft operating under turbulence and actuator delay. This work introduces a new parallel RL-MPC architecture tailored for real-time aeroelastic control. During training, two LPV-based MPC controllers compute state and disturbance-dependent input bounds under known best and worst-case gust scenarios, using long horizons and accurate scheduling without computational constraints. A Q-learning agent learns to select actions within these bounds, using a rollout strategy that respects actuator transport delay and ensures constraint satisfaction. During deployment, a certified interpolation strategy ensures constraint satisfaction when encountering unseen states. The approach balances model-based safety and data-driven adaptability, enabling lightweight, real-time control under turbulence and uncertainty.

II. Methodology

This section presents the control methodology proposed for safe learning and control of a flexible aeroelastic aircraft model. The approach combines an LPV-based MPC to generate safe bounds on control actions and a RL policy to select feasible control inputs within those bounds. To validate this architecture, both theoretical guarantees and supporting numerical analyses are developed. A high-level

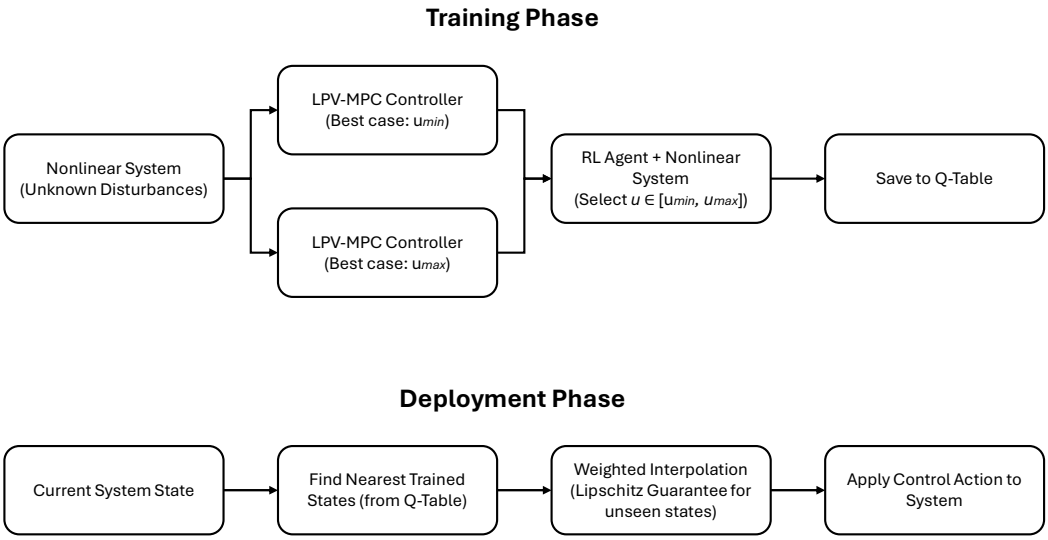


Fig. 1 Overview of Control Framework Architecture

overview of the control structure is shown in Figure 1.

During training, two LPV-based MPC controllers are used to compute safe upper and lower bounds on the admissible control actions for the flexible aircraft system. Since training is performed offline, long prediction horizons can be used to enhance accuracy without concern for real-time computational limits. Moreover, the training environment includes known gust realizations, allowing accurate modeling of the system dynamics and their response to disturbances. The best-case and worst-case scenarios are constructed based on experienced gust evolutions leveraging the fact that, given a particular system state, the underlying gust profile that induced it can be approximately inferred. This enables bounding of the control input over a plausible range of disturbance behaviors and bracketing the possible system responses.

RL agent then selects a control input within these bounds, based on the predicted gust information and the current system state. To ensure that the control effect is observable and meaningful during learning, the control input is held constant for two steps. This choice reflects the presence of actuator dynamics: a single time step is not enough for the control input to propagate through the system and produce a visible effect on all relevant states. Two steps were therefore chosen as the shortest possible duration that fully captures the influence of the input on the system. This is the only relevant criterion: the rollout must be long enough for the control effect to be visible, but no longer than necessary to stay consistent with deployment. In systems with more complex actuator dynamics or different structural responses, a longer rollout might be needed to meet this condition. The RL agent's reward function is formulated as a quadratic cost, $R = -(e^T Q e + r_u u^2)$, designed to drive the system towards a desired decaying state while penalizing control effort. It considers performance under predicted, best-case, and worst-case gust scenarios, ensuring robustness to different disturbance realizations. Through interaction with the environment, the RL agent

learns a control policy that selects the most effective input while respecting the safe control range, storing the policy in a Q-table.

During deployment, new system states may be encountered that do not exactly match the training data. To handle this, a weighted interpolation strategy is applied between nearby trained states in the Q-table. The deviation between the new and trained states must be small enough to guarantee that the interpolated control input does not violate system constraints. This distance depends not only on the proximity to the trained states but also on how close the system is to its operational limits. These bounds will be derived later in the chapter. The goal of this methodology is to formalize and validate the control architecture proposed for safe reinforcement learning. To ensure that this architecture is suitable for safety-critical operation, a series of theoretical guarantees and supporting numerical validations are established throughout this chapter.

A. System Modeling and Setup

1. System Definition and Continuous-Time Dynamics

The aircraft model considered in this work originates from a flexible wing structure with two primary degrees of freedom, plunge and pitch, together with a third dynamic state representing the flap deflection. The full state vector is defined compactly as

$$\boldsymbol{x}(t) = \begin{bmatrix} h(t) & \alpha(t) & \dot{h}(t) & \dot{\alpha}(t) & \beta(t) \end{bmatrix}^\top, \tag{1}$$

where $h(t)$ denotes the vertical displacement (plunge), $\alpha(t)$ denotes the pitch angle, $\dot{h}(t)$ and $\dot{\alpha}(t)$ represent the corresponding velocities, and $\beta(t)$ denotes the flap deflection.

The continuous-time evolution of the system is governed by a set of coupled differential equations that capture both structural and aerodynamic effects. The structural dynamics are modeled through second-order nonlinear equations of the form

$$\begin{bmatrix} \ddot{h}(t) \\ \ddot{\alpha}(t) \end{bmatrix} = (M^{\text{mass}})^{-1} \left(- \begin{bmatrix} L^{\text{aero}} \\ M^{\text{aero}} \end{bmatrix} - C \begin{bmatrix} \dot{h}(t) \\ \dot{\alpha}(t) \end{bmatrix} - K \begin{bmatrix} h(t) \\ \alpha(t) \end{bmatrix} \right), \tag{2}$$

where $M^{\text{mass}} \in \mathbb{R}^{2\times 2}$ is the mass matrix, $C \in \mathbb{R}^{2\times 2}$ is the damping matrix, and $K \in \mathbb{R}^{2\times 2}$ is the stiffness matrix. The aerodynamic lift L^{aero} and aerodynamic moment M^{aero} act as external forces and moments on the structure.

Nonlinearities arise both from the structural and aerodynamic components of the system. Specifically, the stiffness terms in K depend nonlinearly on the states $h(t)$ and $\alpha(t)$ shown in Equation (3), while the aerodynamic forces L^{aero} and M^{aero} are nonlinear functions of the effective angle of attack, which itself depends on the system's states. Consequently, the overall system exhibits nonlinear behavior.

$$\begin{aligned} k_h(h(t)) &= 2844 \cdot \left(1 + 0.9\, h(t)^2\right), \\ k_\alpha(\alpha(t)) &= 2.8 \left(1 - 22\,\alpha(t) + 1316\,\alpha(t)^2 \right. \\ &\qquad \left. -8580\,\alpha(t)^3 + 17290\,\alpha(t)^4\right) \end{aligned} \tag{3}$$

The actuator dynamics associated with the flap are modeled as a first-order system:

$$\dot{\beta}(t) = k_\beta \left(u_{\text{flap}}(t) - \beta(t)\right), \tag{4}$$

where $k_\beta > 0$ is the actuator bandwidth and $u_{\text{flap}}(t)$ denotes the commanded flap deflection input.

In addition to the control input, external disturbances such as vertical gusts influence the system behavior by modify-

ing the effective angle of attack and thereby affecting the aerodynamic forces and moments.

Combining the structural, aerodynamic, actuator, and disturbance effects results in a continuous-time nonlinear state-space system of the general form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), u(t), w(t)\right), \tag{5}$$

where $\boldsymbol{f} : \mathbb{R}^5 \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^5$ encapsulates the full physical behavior of the flexible aircraft model, including nonlinear stiffness, actuator dynamics, and disturbance influence.

2. Turbulence and System Coupling

This study focuses only on vertical gusts, as they have the most significant effect on the angle of attack and the pitch motion. Although horizontal gusts can also affect the aircraft, their influence is less severe and therefore are not included. The vertical gust w enters the system through the effective angle of attack, which becomes:

$$\alpha^{\text{eff}}(t) = \tan^{-1}\left(\frac{U\sin(\alpha(t)) - w(t)}{U\cos(\alpha(t))}\right) + \frac{\dot{h}(t)}{U} + a_b b(t)\cdot\frac{\dot{\alpha}(t)}{U} \tag{6}$$

This establishes a nonlinear, state-dependent coupling between the turbulence and the aircraft dynamics. Which in turn determines the aerodynamic lift and moment:

$$L^{\text{aero}} = \rho U^2 b\left(c_{L\alpha}\alpha^{\text{eff}}(t) + c_{L\beta}\beta(t)\right), \tag{7}$$

$$M^{\text{aero}} = \rho U^2 b^2\left(c_{m\alpha}\alpha^{\text{eff}}(t) + c_{m\beta}\beta(t)\right). \tag{8}$$

These forces affect the dynamics of the system as shown in 2 thus embedding the gust signal into the state update through both aerodynamic and structural channels.

To ensure that the control architecture can reject gusts effectively, the controller must operate fast enough to capture their dominant frequencies. In this study, w is generated using a Dryden turbulence model augmented with stochastic variations to reflect extreme flight conditions [37]. A Fourier analysis of representative gust signals shows that most energy lies between 0.2 Hz and 2 Hz, with negligible energy above 5 Hz. Given the 1000 Hz * control rate and the low-pass nature of the plant, the LPV-MPC framework remains valid under these disturbance conditions.

The turbulence $w(t)$ is generated using a Dryden turbulence model filtered to reflect stochastic but physically realistic flight conditions. As constructed, $w(t)$ is bounded in both amplitude and frequency, with over 90% of its spectral energy concentrated below 3 Hz and negligible content above 10 Hz. This low-frequency structure ensures that the disturbance evolves smoothly relative to the 1 kHz control rate. Although $\frac{\partial \alpha}{\partial w} \neq 0$ introduces a nonlinear coupling between gusts and the aircraft dynamics, the overall system remains continuous and well-posed for LPV linearization. These properties validate the key design assumptions underlying both the LPV-MPC and RL-MPC controllers, namely, that the system is subject to bounded, slowly varying yet unpredictable disturbances.

3. Modeling Setup and Assumptions

This work studies a discretized version of a nonlinear flexible aircraft model. The model is derived from prior identification based on wind tunnel. The original dynamics are continuous, but analysis is performed in discrete

*Lower control frequencies such as 200–500 Hz were tested during initial simulations but led to instability in states that were expected to remain stable. Stability was only consistently achieved at a 1000 Hz simulation frequency, which was then matched by the control frequency for consistency.

time to enable tractable development while preserving key behaviors. The nonlinear stiffness terms $k_h(h(t))$ and $k_a(\alpha(t))$ are modeled as smooth polynomials as shown in (3), justifying the use of Taylor expansions. The control input modifies flap deflection β through a first-order filter, after which aerodynamic forces are applied.

Several assumptions are made to frame the theoretical analysis. The flap input u^{flap} and the turbulence $w(t)$ are bounded within known limits, which is met by physical actuators and atmospheric phenomena. For simulation, gusts are treated as fixed external signals injected independently of the state evolution. All derivations are conducted in discrete time, and stability, boundedness, and monotonicity are established for the discrete-time system. While the continuous-time system is the physical reference, it is assumed that properties shown in the discrete domain approximately carry over to the continuous case within the limits of numerical fidelity. This is a common approximation in digital control where sufficiently small time steps are used. During training and rollout, the control input is held constant for two time steps to capture actuator dynamics accurately. The actuator transport delay itself is not explicitly treated in this section, as the goal here is to verify that the trained state trajectories remain within safe bounds; delay compensation will be addressed later in Section II.C using a Lipschitz-based deployment-time safety filter. These assumptions form the foundation for the rest of the theoretical and numerical developments.

4. Discrete-Time State Update and Approximation

The analysis from this point onward is based on a discretized version of the continuous-time system. While the true dynamics follow a differential equation given in Equation (5), a discrete-time approximation is adopted for tractable prediction, analysis, and control design. This allows the nonlinear behavior of the system to be captured over finite time steps while enabling theoretical guarantees to be derived in a step-wise manner.

To discretize the system, a Taylor expansion is applied. For a small time step Δt, the state after n steps is given by

$$\boldsymbol{x}_{k+n} \approx \boldsymbol{x}_k + \sum_{i=1}^{n} \frac{\Delta t^i}{i!} \cdot \frac{d^{(i-1)}}{dt^{(i-1)}} \boldsymbol{f}(\boldsymbol{x}_k, u_k, w_k). \quad (9)$$

In this work, a two-step horizon is considered ($n = 2$). This choice is motivated by the actuator structure: the control input u_k influences the actuator state β, which only begins to affect the lift and moment terms in the second step. Using only one step fails to capture this effect, while using more than two would no longer reflect the real-time control update pattern, where a new input is selected at each time step. The two step expression is derived as follows:

$$\boldsymbol{x}_{k+2} \approx \boldsymbol{x}_k + \Delta t \cdot \boldsymbol{f}(\boldsymbol{x}_k, u_k, w_k) + \Delta t \cdot \boldsymbol{f}(\boldsymbol{x}_{k+1}, u_{k+1}, w_{k+1}).$$

$$\boldsymbol{f}(\boldsymbol{x}_{k+1}, u_{k+1}, w_{k+1}) \approx \boldsymbol{f}_k + \Delta t \cdot \nabla_{\boldsymbol{x}} \boldsymbol{f}_k \cdot \boldsymbol{f}_k,$$

$$\boldsymbol{x}_{k+2} \approx \boldsymbol{x}_k + 2\Delta t \cdot \boldsymbol{f}_k + \Delta t^2 \cdot \nabla_{\boldsymbol{x}} \boldsymbol{f}_k \cdot \boldsymbol{f}_k + O(\Delta t^3) \quad (10)$$

The term $2\Delta t \cdot \boldsymbol{f}_k$ represents the accumulated first-order motion over two steps. The second-order term $\Delta t^2 \cdot \nabla_{\boldsymbol{x}} \boldsymbol{f}_k \cdot \boldsymbol{f}_k$ captures the nonlinear curvature of the trajectory as the states evolve. Higher-order effects, such as those involving the Hessian $\nabla^2_{\boldsymbol{x}} \boldsymbol{f}$, are of order $O(\Delta t^3)$ and are neglected. This is justified given the small time step used in simulation, $\Delta t = 0.001$, making such terms numerically insignificant. As will be shown later, the system response under this approximation behaves nearly linearly over short rollouts,

even in the presence of nonlinearities. This supports the use of second-order discretization as a basis for controller design and safety certification. The full dynamics include actuator filtering, nonlinear stiffness, and aerodynamic forces evaluated at each step of the expansion.

B. Numerical System Validation

This section establishes the system-specific mathematical groundwork required to support the formal proofs presented later in this report. The goal is to ensure that key assumptions, such as the validity of the linear parameter-varying (LPV) model and the use of second-order approximations, hold under the dynamic conditions of the flexible aircraft system considered here. While the analytical structure of the arguments is general, the numerical results and thresholds derived in this section apply specifically to the current system configuration. All detailed calculations, including the bounding of nonlinear terms and justification for higher-order term neglect, need to be replicated for other systems. This phase is essential: the formal guarantees in later sections rely on the numerical validity of these assumptions for the specific model under consideration.

1. Numerical Validation of LPV Modeling

The MPC controller used in this work relies on an LPV (Linear Parameter-Varying) approximation of the nonlinear aircraft model to generate two reference trajectories. These trajectories are later interpolated by a reinforcement learning (RL) policy during deployment. For this control architecture to remain safe and reliable, it is necessary to ensure that the LPV-generated trajectories are valid approximations of the true nonlinear dynamics.

The system's nonlinear behavior arises primarily from the state-dependent stiffness terms and aerodynamic coupling. The plunge stiffness k_h introduces a quadratic dependence on vertical displacement h, while the pitch stiffness k_a follows a fourth-order polynomial in α. Additional nonlinearities enter through the effective angle of attack, which is influenced by both $\dot{h}$ and $\dot{\alpha}$, thereby affecting the lift and aerodynamic moment equations.

To construct the LPV model, the dynamics are linearized at each timestep around the current state $\boldsymbol{x}_k = [h_k, \alpha_k, v_k^h, v_k^\alpha, \beta_k]$, resulting in an expression of the form $\boldsymbol{x}_{k+1} = A(h_k, \alpha_k)\boldsymbol{x}_k + Bu_k$. Since the control input affects only the flap through $\beta_{k+1} = \beta_k + \Delta t \cdot k_\beta(u_{\text{flap},k} - \beta_k)$, the input matrix B is constant with respect to the system state and independent of the structural and aerodynamic nonlinearities.

The validity of the LPV approximation hinges on the assumption that the local linearization error remains bounded. Specifically, it must hold that

$$\|\boldsymbol{f}(\boldsymbol{x}_k, u_k) - A(h_k, \alpha_k)\boldsymbol{x}_k - Bu_k\| \leq \epsilon \quad \text{for all } \boldsymbol{x} \in \mathcal{X} \tag{11}$$

where $\mathcal{X}$ is a neighborhood around the linearization point. The following sections provide frequency-domain and numerical validation of this approximation.

In this setup, the system matrices $A(h, \alpha)$ are updated at every control step based on real-time evaluations of h and α, capturing the instantaneous stiffness and aerodynamic behavior. Because the timestep Δt is sufficiently small, the nonlinear evolution of the system remains close to that predicted by the LPV approximation, ensuring control accuracy.

This approach is consistent with theoretical studies showing that LPV approximations offer bounded error for systems

with smoothly varying parameters. When linearizations are performed frequently and within a well-characterized local domain, the LPV model accurately reflects the underlying nonlinear dynamics. This supports its use as a foundation for safe control in the presence of state-dependent stiffness and aerodynamic coupling. A mathematical justification of this approximation is provided in the following subsections.

As stated in Equation (11), the validity of the LPV approximation depends on the boundedness of the linearization error across the relevant state space. Before quantifying this error directly, it is instructive to examine the frequency content of the system's response to determine whether the dynamics evolve slowly enough to justify frequent linearizations. Specifically, if the dominant system modes lie well below the LPV update rate (1000 Hz), then the linear approximation at each timestep is likely to remain locally accurate throughout the horizon. This serves as a practical first check for whether the model's dynamics are compatible with a high-frequency LPV-based controller. To further support this, a comparison is made between the open-loop response of the continuous system and that of the LPV system updated at 1000 Hz. The similarity between the two trajectories provides additional evidence that the LPV approximation remains valid at this update rate over the relevant dynamic range.

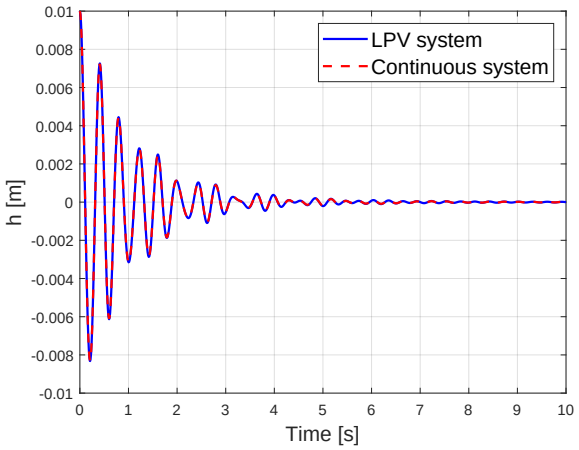


Fig. 2 Figure comparing the output of the LPV system and the continuous system

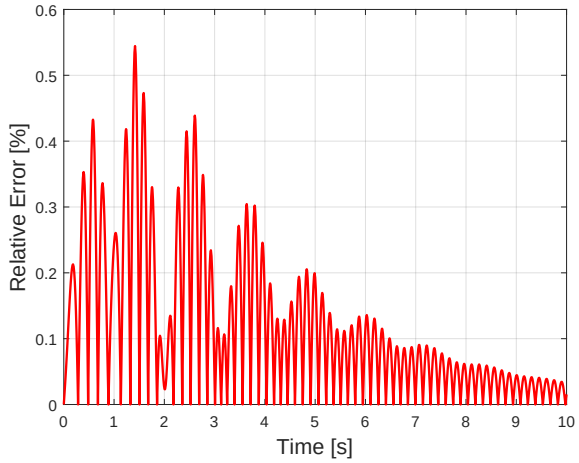


Fig. 3 Figure showing the relative difference between the LPV and continuous systems

A targeted frequency analysis is also conducted on the nonlinear model to characterize the timescales of all five states in response to random low-amplitude actuation. The FFT analysis confirmed that the dominant frequencies for all five states lie between (0.5-5 Hz), well below the 1000 Hz LPV update rate. The actuator's bandwidth of 9.5 Hz is approximately double that of the fastest component, further supporting the feasibility of this control frequency. Additionally, this supports the use of MPC during RL

training, as the prediction horizons required to resolve such low-frequency dynamics would be too long for real-time deployment. However, during training, computational time is not a constraint, enabling MPC to generate long-horizon rollouts that further justify its use as an offline training oracle rather than an online controller.

To support the numerical validation, this section provides a formal justification for why the LPV approximation remains accurate under the model's nonlinear dynamics. Given that the system is linearized around the current state $\boldsymbol{x}_k$ at every control step, the resulting LPV model corresponds to a first-order Taylor expansion of the true dynamics:

$$\boldsymbol{f}(\boldsymbol{x}, u) = \boldsymbol{f}(\boldsymbol{x}_k, u) + \nabla_{\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) + \boldsymbol{R}(\boldsymbol{x}) \tag{12}$$

with the approximation error $\boldsymbol{R}(\boldsymbol{x})$ bounded as:

$$\|\boldsymbol{R}(\boldsymbol{x})\| \leq \frac{1}{2} \sup_{\boldsymbol{\xi} \in \mathcal{X}} \|D^2 \boldsymbol{f}(\boldsymbol{\xi})\| \cdot \|\boldsymbol{x} - \boldsymbol{x}_k\|^2 \tag{13}$$

This expression confirms that the LPV model offers a second-order accurate approximation in the vicinity of $\boldsymbol{x}_k$, provided the dynamics are sufficiently smooth. In this application, the nonlinear stiffness and aerodynamic terms are differentiable with bounded second derivatives, implying the LPV error remains locally controlled.

This theoretical guarantee justifies the use of frequent linearizations in the control loop and supports the safety of trajectory generation using LPV models. The perturbation values used to assess the LPV approximation error are derived from 500 simulations of the nonlinear aircraft model and represent the 99.9^{th} percentile of observed single-timestep deviations. While the specific numerical results are tied to the system considered in this work, the approach itself is applicable to other systems, however any attempt to replicate or adapt the LPV approach to a different model should be accompanied by a similar system-specific validation. In this case, the maximum deviation in any state matrix entry was 0.27%, and the maximum deviation in the resulting state trajectory was 0.156%, both are sufficiently small to justify the use of the LPV approximation.

2. Validation of Second-Order Model

To justify the use of a second-order approximation in the state propagation, the next term in the Taylor expansion is examined to demonstrate that its contribution is negligible for small time steps. The full Taylor expansion of the discrete-time state update is given by:

$$\begin{aligned} \boldsymbol{x}_{k+2} = \boldsymbol{x}_k + 2\Delta t \cdot f_k + \Delta t^2 \cdot \nabla_x f_k f_k \\ + \frac{\Delta t^3}{6} \cdot D^2 f_k[f_k, f_k] + O(\Delta t^4) \end{aligned} \tag{14}$$

In this expression, the third term represents the second-order curvature (Jacobian), while the fourth term captures the third-order behavior via the Hessian $D^2 f_k$ applied twice to the vector f_k. Specifically, the notation $D^2 f_k[f_k, f_k]$ refers to the application of the Hessian tensor to the direction f_k twice. For each component i, this results in:

$$\left[D^2 f_k[f_k, f_k]\right]_i = \sum_{j=1}^{n} \sum_{l=1}^{n} \frac{\partial^2 f_i}{\partial x_j \partial x_l} f_{k,j} f_{k,l} \tag{15}$$

This represents the second-order directional derivative of the i-th component of f along the direction f_k. The objective is to demonstrate that the magnitude of this third-order term is significantly smaller than that of the second-order term.

To evaluate the relative importance of the third-order term,

State	h_k	v_k^h	α_k	v_k^α	β_k
3rd / 2nd Order (%)	0.153%	0.000126%	0.291%	0.04326%	2.0%

Table 2 Relative contribution of third-order term to second-order update for each state.

the ratio between the third- and second-order terms in Equation 14 is considered:

$$\text{Ratio} = \frac{\left\| \frac{\Delta t^3}{6} D^2 f_k[f_k, f_k] \right\|}{\left\| \Delta t^2 \nabla_x f_k f_k \right\|} = \frac{\Delta t}{6} \cdot \frac{\|D^2 f_k\| \cdot \|f_k\|}{\|\nabla_x f_k\|} \quad (16)$$

Let us define the following bounds:

- $\|f_k\| \leq M$: Maximum expected value of the state derivative
- $\|\nabla_x f_k\| \leq K^{\mathrm{J}}$: Maximum norm of the Jacobian
- $\|D^2 f_k\| \leq H$: Maximum norm of the Hessian (curvature)

Substituting these into the ratio gives:

$$\text{Ratio} \leq \frac{\Delta t}{6} \cdot \frac{H \cdot M}{J} \quad (17)$$

This expression provides an upper bound on the third-order term's contribution relative to the second-order term. To represent this ratio quantitatively, values of H, M, and K^{J} must be evaluated based on the system's dynamics. Representative values for the system states and their derivatives are therefore selected using a numerical analysis based on the 99.9^{th} percentile of state and derivative magnitudes, extracted from an extensive set of simulations of the current aircraft model. This approach ensures that the conclusions remain valid even under extreme, yet realistic, operating conditions. It is important to note that the results presented here are specific to the system under consideration and may not be generalizable. For other systems or configurations, a similar analysis should be performed using appropriately tailored simulation data. The approximation is more likely to transfer well to smooth, time-invariant, and fully actuated systems with moderate state rates and curvature. In contrast, systems with strong nonlinearity, under-actuation, or significant time variation may require tighter bounds or result in less favorable ratios. This conclusion is supported by a quantitative evaluation showing that third-order term effects are negligible as is summarized in Table 2. The table reports the ratio of the third-order Taylor expansion term to the second-order term, expressed as a percentage. These ratios quantify the relative magnitude of the third-order contribution for each state under the worst-case conditions used in the analysis.

According to this numerical analysis, which is specific to the aircraft system studied here and based on conservative high-percentile values from extensive simulation data. The third-order term remains consistently small relative to the second-order term for all state updates. This validates the use of a second-order Taylor approximation for this system, and higher-order contributions can be safely neglected in subsequent derivations and controller implementations.

The analysis demonstrated that the third-order term remains small for the default time step $\Delta t = 0.001$. Table 2 summarizes the relative third-to-second-order contributions for each state under worst-case conditions. All ratios are below 2%, with most STATES significantly lower (e.g., v_k^h at 0.000126%), confirming that the second-order approximation is accurate at this time step. Since this ratio scales linearly with Δt, the existing results can be extrapolated to

identify the maximum step size that preserves a desired accuracy threshold. Table 3 reports the maximum allowable time step for each state such that the third-order contribution remains below 5% of the second-order term. Notably, the flap state β_k imposes the most restrictive bound.

State	h_k	α_k	v_k^h	v_k^α	β_k
Max Δt [s]	0.032	0.033	3.97	0.115	0.0025

Table 3 Maximum Δt per state for limiting third-order terms to under 5% of second-order.

These results confirm that the second-order Taylor expansion remains valid at $\Delta t = 0.001$ across all states, and significantly larger steps are possible in most cases. The flap state β_k poses the most restrictive condition. However, this bound depends on the value of k_β. In systems with faster flap dynamics, i.e., with larger k_β the third-order term becomes less significant, allowing larger time steps without violating the second-order accuracy assumption.

C. Theoretical Guarantees

This section provides the theoretical guarantees that support the safe deployment of the learned control policy. Building on the assumptions and observations established in the numerical analysis II.B, it is first shown that the training procedure yields control inputs that produce bounded and constraint-satisfying state updates. These guarantees are then extended to the deployment phase, where the controller must operate on previously unseen states. To ensure safety in such cases, a Lipschitz-based filtering mechanism is introduced to certify the interpolated control inputs by bounding their deviation from known-safe trajectories.

1. Monotonicity and Bounding of the Full State Update

This section investigates whether the second-order state update lies between the outcomes obtained by applying the extreme control inputs $u_{\min}$ and $u_{\max}$. Specifically, it will be shown that, under the assumptions established in Section II.B, applying any intermediate control input $u \in [u_{\min}, u_{\max}]$ may result in a state $\boldsymbol{x}_{k+2}(u)$ that remains bounded between $\boldsymbol{x}_{k+2}(u_{\min})$ and $\boldsymbol{x}_{k+2}(u_{\max})$. If such boundedness can be established, it enables constraint satisfaction to be inferred by verifying only the two bounding trajectories. Since these cases correspond to the outputs of LPV-MPC controllers that have already been proven to satisfy the full system constraints, the intermediate state resulting from any RL-selected input can also be certified as safe. This property is critical, as it forms the theoretical foundation for enabling safe action selection by the reinforcement learning agent. Using the second-order Taylor expansion, the state update is expressed as:

$$\boldsymbol{x}_{k+2} = \boldsymbol{x}_k + \Delta t \cdot \boldsymbol{f}(\boldsymbol{x}_k, u_k) + \frac{\Delta t^2}{2} \cdot \boldsymbol{K}^{\mathrm{J}}(\boldsymbol{x}_k) \cdot \boldsymbol{f}(\boldsymbol{x}_k, u_k)$$

$$\boldsymbol{x}_{k+2} = \boldsymbol{x}_k + \boldsymbol{S}(\boldsymbol{x}_k) \cdot \boldsymbol{f}(\boldsymbol{x}_k, u_k) \tag{18}$$

where $S(\boldsymbol{x}_k) = \Delta t \cdot I + \frac{\Delta t^2}{2} \cdot \boldsymbol{K}^{\mathrm{J}}(\boldsymbol{x}_k)$. While it might initially appear that if the function $\boldsymbol{f}(\boldsymbol{x}_k, u)$ lies between $\boldsymbol{f}(\boldsymbol{x}_k, u_{\min})$ and $\boldsymbol{f}(\boldsymbol{x}_k, u_{\max})$, then the same would be true for the updated state $\boldsymbol{x}_{k+2}(u)$, this implication is not guaranteed. To formally guarantee that the updated state $\boldsymbol{x}_{k+2}(u)$ lies between the trajectories generated by $u_{\min}$ and $u_{\max}$, certain conditions must hold. Specifically, it must be assumed that the mapping $u \mapsto \boldsymbol{f}(\boldsymbol{x}_k, u)$ is monotonic for each individual state component, so the effect of control

inputs does not reverse direction. Additionally, the dynamic update structure, captured by the matrix $\boldsymbol{A}(\boldsymbol{x}_k)$, must preserve both the sign and relative magnitude of variations in $\boldsymbol{f}(\boldsymbol{x}_k, u)$ across the state evolution. Under these assumptions, it would follow that bounding $\boldsymbol{f}(\boldsymbol{x}_k, u)$ between its values at the control extremes is sufficient to bound the future state as well. However, this implication cannot be relied upon unless those structural properties are satisfied.

These conditions are not uniformly satisfied across the studied system, particularly for states influenced by nonlinear aerodynamic forces and stiffness terms that depend indirectly on the control input, such as through the effective angle of attack. As a result, it cannot be generally assumed that the full state update lies between those generated by $u_{\min}$ and $u_{\max}$. Guaranteeing this would require global monotonicity of $\boldsymbol{f}(\boldsymbol{x}_k, u)$ with respect to u, or that the second-order update matrix $\boldsymbol{A}(\boldsymbol{x}_k)$ preserves such ordering across all state directions—neither of which holds in this setting. Therefore, each state component must be analyzed individually to rigorously establish boundedness between the two control extremes.

2. Bounding Analysis for h_{k+2} and α_{k+2}

This section verifies whether the second-order updates of the plunge and pitch states, h_{k+2} and α_{k+2}, remain bounded between the outcomes resulting from the extreme control inputs $u_{\min}$ and $u_{\max}$. Establishing this property ensures that applying any intermediate control $u \in [u_{\min}, u_{\max}]$ yields a state trajectory within a known safe region, assuming the bounding cases are verified to satisfy constraints.

Step 1: Second-Order Expansion. The second-order updates are given by:

$$\begin{aligned} h_{k+2} &= h_k + \Delta t \cdot v_k^h + \frac{\Delta t^2}{2} \cdot a_k^h, \\ \alpha_{k+2} &= \alpha_k + \Delta t \cdot v_k^\alpha + \frac{\Delta t^2}{2} \cdot a_k^\alpha \end{aligned} \tag{19}$$

As h_k, v_k^h, α_k, and v_k^α are fixed at the current timestep, the only input-dependent variation arises from the acceleration terms a_k^h and a_k^α.

Step 2: Expressions for Accelerations. The coupled dynamics yield the following expressions:

$$\begin{aligned} a_k^h &= \left(\frac{I_a}{\det M^{\text{mass}}}\right)(-L_k^{\text{aero}} - c_h v_k^h - k_h h_k) \\ &\quad + \left(-\frac{m_w x_a b}{\det M^{\text{mass}}}\right)(-M_k^{\text{aero}} - c_a v_k^\alpha - k_a \alpha_k) \\ a_k^\alpha &= \left(\frac{m_t}{\det M^{\text{mass}}}\right)(-M_k^{\text{aero}} - c_a v_k^\alpha - k_a \alpha_k) \\ &\quad + \left(-\frac{m_w x_a b}{\det M^{\text{mass}}}\right)(-L_k^{\text{aero}} - c_h v_k^h - k_h h_k) \end{aligned} \tag{20}$$

The terms affected by the control input u_k are the aerodynamic force L_k^{aero} and moment M_k^{aero}, both of which depend on the flap deflection β_k. The flap evolves according to the first-order discrete-time system:

$$v_k^\beta = \dot{\beta}_k = k_\beta(u_k - \beta_k) \tag{21}$$

This structure ensures that β_k is a smooth and monotonic function of the control input u_k over short time intervals.

Step 3: Dependence of Aerodynamic Terms on Input. The aerodynamic terms are given by:

$$\begin{aligned} L_k^{\text{aero}} &= \rho U^2 b \left(c_{L\alpha} \cdot \alpha_{\text{eff},k} + c_{L\beta} \cdot \beta_k\right), \\ M_k^{\text{aero}} &= \rho U^2 b^2 \left(c_{m\alpha} \cdot \alpha_{\text{eff},k} + c_{m\beta} \cdot \beta_k\right) \end{aligned} \tag{22}$$

with

$$\alpha_{\text{eff},k} = \alpha_k + \frac{v_k^h}{U} + a_b b \cdot \frac{v_k^\alpha}{U} \tag{23}$$

Since $\alpha_{\text{eff},k}$ is independent of the control input u_k, both L_k^{aero} and M_k^{aero} depend on u_k only through the term $\beta_k(u_k)$. As β_k is a monotonic function of u_k, the mappings $u_k \mapsto L_k^{\text{aero}}$ and $u_k \mapsto M_k^{\text{aero}}$ are also monotonic. Consequently, the accelerations $a_k^h(u_k)$ and $a_k^\alpha(u_k)$, which are affine in L_k^{aero} and M_k^{aero}, are monotonic with respect to the control input. It follows that:

$$\begin{aligned} & a_k^h(u_k) \in \left[a_k^h(u_{\min}),\ a_k^h(u_{\max})\right], \\ & a_k^\alpha(u_k) \in \left[a_k^\alpha(u_{\min}),\ a_k^\alpha(u_{\max})\right], \\ \Rightarrow \quad & h_{k+2}(u_k) \in \left[h_{k+2}(u_{\min}),\ h_{k+2}(u_{\max})\right], \\ & \alpha_{k+2}(u_k) \in \left[\alpha_{k+2}(u_{\min}),\ \alpha_{k+2}(u_{\max})\right] \end{aligned} \tag{24}$$

This confirms that the second-order updates of both h and α remain bounded between the two trajectories corresponding to the extreme control inputs.

3. Bounding Analysis for v_{k+2}^h and v_{k+2}^α

This section analyzes whether the velocity states v_{k+2}^h and v_{k+2}^α remain bounded between the outcomes corresponding to the extreme control inputs $u_{\min}$ and $u_{\max}$, assuming an intermediate control value $u_k \in [u_{\min}, u_{\max}]$.

Step 1: Approximate Update Expressions. The discrete-time velocity updates are approximated using the second-order expansion:

$$v_{k+2}^h = v_k^h + \Delta t \cdot a_k^h + \frac{\Delta t^2}{2} \cdot j_k^h, \tag{25}$$

$$v_{k+2}^\alpha = v_k^\alpha + \Delta t \cdot a_k^\alpha + \frac{\Delta t^2}{2} \cdot j_{\alpha,k} \tag{26}$$

As established in the third-order truncation analysis, the jerk terms j_k^h and j_k^α are negligibly for small time steps ($\Delta t = 0.001$). The updates are therefore simplified as:

$$v_{k+2}^h \approx v_k^h + \Delta t \cdot a_k^h, \qquad v_{k+2}^\alpha \approx v_k^\alpha + \Delta t \cdot a_k^\alpha \tag{27}$$

Step 2: Bounding Behavior. The accelerations a_k^h and a_k^α are monotonic in control input u_k, due to their linear dependence on the aerodynamic terms L_k^{aero} and M_k^{aero}, which are affine in the flap deflection $\beta_k(u_k)$. Since $\beta_k(u_k)$ evolves monotonically with u_k, the accelerations are kept within the minimum and maximum values as is shown in equation 24. Because the timestep is fixed and the system follows a discrete-time kinematic relation, the resulting velocities also remain within predefined bounds, as expressed in:

$$\begin{aligned} v_k^h(u_k) &\in \left[v_k^h(u_{\min}),\ v_k^h(u_{\max})\right] \\ v_k^\alpha(u_k) &\in \left[v_k^\alpha(u_{\min}),\ v_k^\alpha(u_{\max})\right] \end{aligned} \tag{28}$$

Substituting Equation 24 and Equation 28 into velocity update rule yields:

$$\begin{aligned} v_{k+2}^h(u_k) &\in \left[v_{k+2}^h(u_{\min}),\ v_{k+2}^h(u_{\max})\right] \\ v_{k+2}^\alpha(u_k) &\in \left[v_{k+2}^\alpha(u_{\min}),\ v_{k+2}^\alpha(u_{\max})\right] \end{aligned} \tag{29}$$

This confirms that the second-order updates of the velocity states remain bounded within the range defined by the extreme control inputs.

4. Bounding Analysis for β_{k+2}

The flap state β_{k+2} evolves according to a first-order discrete-time equation and remains bounded between the trajectories generated by the control inputs $u_{\min}$ and $u_{\max}$. This is particularly important, as β directly influences the aerodynamic terms in the coupled dynamics.

Step 1: Discrete Update Rule. The continuous-time

evolution is given by:

$$\dot{\beta} = k_\beta(u_k - \beta_k) \tag{30}$$

Applying forward Euler integration for two steps yields:

$$\begin{aligned} \beta_{k+1} &= \beta_k + \Delta t \cdot k_\beta(u_k - \beta_k), \\ \beta_{k+2} &= \beta_{k+1} + \Delta t \cdot k_\beta(u_{k+1} - \beta_{k+1}) \end{aligned} \tag{31}$$

Both updates are affine in the control inputs and represent convex combinations of prior values, ensuring that β_k evolves smoothly and monotonically in u_k and u_{k+1}.

Step 2: Monotonicity and Bounding. Because each step depends monotonically on its respective control input, the full two-step update is also monotonic with respect to both u_k and u_{k+1}. As a result:

$$\beta_{k+2}(u) \in [\beta_{k+2}(u_{\min}), \beta_{k+2}(u_{\max})] \tag{32}$$

This guarantees that the discrete-time update of the flap state β remains bounded within the safe envelope defined by the extremal control inputs. While the system is nonlinear due to stiffness terms like $k_h(h)$, $k_a(\alpha)$, and aerodynamic forces that depend on α^{eff}, the key dependencies on the control input u occur through the flap deflection β, which evolves smoothly according to a first-order stable system. As a result, $\beta(u)$ is monotonic in u, and so are the aerodynamic terms $L(u)$ and $MM(u)$, which are affine in β. This monotonicity carries through to all acceleration terms and thereby to the full state update expressions.

5. Lipschitz-Based Deployment-Time Generalization Guarantee

While earlier sections verified that the trained policy produces safe trajectories, this section addresses deployment-time safety, specifically when the controller encounters a state $\boldsymbol{x}_{\text{curr}}$ not seen during training. To ensure safety in such cases, a runtime filter is introduced that selects a control input by interpolating from a set of nearby data points, each associated with a known-safe rollout. This approach draws on ideas from local control synthesis and data-driven safety enforcement. It assumes access to a local database:

$$\mathcal{D} = \left\{ \left(\boldsymbol{x}_k^{(j)},\ \boldsymbol{u}_k^{(j)},\ \boldsymbol{x}_{k+1}^{(j)} \right) \right\}_{j=1}^{N} \tag{33}$$

where each state-action pair $(\boldsymbol{x}_k^{(j)}, \boldsymbol{u}_k^{(j)})$ produces a one-step outcome $\boldsymbol{x}_{k+1}^{(j)}$ that is known to satisfy all constraints. Here, the superscript (j) denotes the index of the database entry, indicating the jth verified rollout sample, while k refers to the time index within that rollout. These verified outcomes are used to construct and certify safe control inputs at deployment.

The core objective is to show that, under mild regularity assumptions on the dynamics, the one-step state resulting from the interpolated input $\boldsymbol{u}^*$ remains close to the verified outcomes $\boldsymbol{x}_{k+1}^{(j)}$. This deviation is bounded using a Lipschitz expression, and if the bound remains within the safety margin to the constraint boundary, the interpolated state is guaranteed to be safe. This filtering scheme is applied at every timestep during deployment, providing formal, simulation-free safety certification. It generalizes safety to unseen states using only locally available data, without requiring retraining or model revalidation.

6. Interpolated Control and One-Step Deviation Bound

Given a current state $\boldsymbol{x}_{\text{curr}}$, a control input is computed by interpolating between the control inputs $\boldsymbol{u}_k^{(j)}$ associated with its nearest neighbors $\boldsymbol{x}_k^{(j)}$ in the local database $\mathcal{D}$.

The interpolated input is given by:

$$\boldsymbol{u}^* = \sum_{j=1}^{N} w^{(j)} \cdot \boldsymbol{u}_k^{(j)}, \quad \text{where } w^{(j)} \geq 0, \quad \sum_{j=1}^{N} w^{(j)} = 1, \tag{34}$$

$$\text{and } w^{(j)} \propto \frac{1}{\|\boldsymbol{x}_{\text{curr}} - \boldsymbol{x}_k^{(j)}\| + \epsilon} \tag{35}$$

Here, $w^{(j)}$ denotes the interpolation weight associated with the j-th entry in the database $\mathcal{D}$, matching the pair $(\boldsymbol{x}_k^{(j)}, \boldsymbol{u}_k^{(j)})$. A small regularization constant $\epsilon > 0$ is included for numerical stability. This weighting prioritizes neighbors closer to $\boldsymbol{x}_{\text{curr}}$, under the assumption that similar states require similar safe actions. The key idea is that if each $\boldsymbol{u}_k^{(j)}$ is known to produce a safe transition from $\boldsymbol{x}_k^{(j)}$, and if $\boldsymbol{x}_{\text{curr}}$ is sufficiently close to all $\boldsymbol{x}_k^{(j)}$, then the interpolated action $\boldsymbol{u}^*$ is also likely to produce a safe outcome. Unlike rollout-based approaches that rely on forward simulation to assess safety, this method guarantees safety through static local bounds, avoiding any need for multi-step prediction. In the following section, this idea is formalized by deriving a bound on the deviation between the interpolated one-step state and the verified safe outcomes.

To certify the safety of $\boldsymbol{u}^*$, the resulting state $\boldsymbol{x}_{\text{curr+1}}(\boldsymbol{u}^*)$, obtained by applying the interpolated control at $\boldsymbol{x}_{\text{curr}}$, is compared to the safe outcomes $\boldsymbol{x}_{k+1}^{(j)} = \boldsymbol{x}_{k+1}(\boldsymbol{u}_k^{(j)})$, each resulting from applying $\boldsymbol{u}_k^{(j)}$ at the corresponding neighbor state $\boldsymbol{x}_k^{(j)}$. Assuming Lipschitz continuity of the dynamics $f(\boldsymbol{x}, \boldsymbol{u})$, there exist constants L_x and L_u such that:

$$\left\| f(\boldsymbol{x}_{\text{curr}}, \boldsymbol{u}^*) - f(\boldsymbol{x}_k^{(j)}, \boldsymbol{u}_k^{(j)}) \right\| \leq L_x \left\| \boldsymbol{x}_{\text{curr}} - \boldsymbol{x}_k^{(j)} \right\| + L_u \left\| \boldsymbol{u}^* - \boldsymbol{u}_k^{(j)} \right\| \tag{36}$$

Using forward Euler integration, the one-step updates are:

$$\boldsymbol{x}_{\text{curr+1}} = \boldsymbol{x}_{\text{curr}} + \Delta t \cdot f(\boldsymbol{x}_{\text{curr}}, \boldsymbol{u}^*) \tag{37}$$

$$\boldsymbol{x}_{k+1}^{(j)} = \boldsymbol{x}_k^{(j)} + \Delta t \cdot f(\boldsymbol{x}_k^{(j)}, \boldsymbol{u}_k^{(j)}) \tag{38}$$

The deviation between $\boldsymbol{x}_{\text{curr+1}}$ and each safe outcome $\boldsymbol{x}_{k+1}^{(j)}$ is bounded using the triangle inequality and Equation 36:

$$\left\| \boldsymbol{x}_{\text{curr+1}} - \boldsymbol{x}_{k+1}^{(j)} \right\| \leq \left\| \boldsymbol{x}_{\text{curr}} - \boldsymbol{x}_k^{(j)} \right\| + \Delta t \cdot L_x \left\| \boldsymbol{x}_{\text{curr}} - \boldsymbol{x}_k^{(j)} \right\| + \Delta t \cdot L_u \left\| \boldsymbol{u}^* - \boldsymbol{u}_k^{(j)} \right\| \tag{39}$$

This bound establishes that the one-step deviation from any known-safe neighbor is determined by the distance in state space $\left\| \boldsymbol{x}_{\text{curr}} - \boldsymbol{x}_k^{(j)} \right\|$ and the difference in control inputs $\left\| \boldsymbol{u}^* - \boldsymbol{u}_k^{(j)} \right\|$. Rather than considering the entire database, only a subset of the closest neighbors is selected, where the number of selected neighbors is a variable that depends on the proximity of the states $\boldsymbol{x}_k^{(j)}$ to the current state $\boldsymbol{x}_{\text{curr}}$. Since both quantities are minimized through the nearest-neighbor interpolation process, the interpolated successor $\boldsymbol{x}_{\text{curr+1}}(\boldsymbol{u}^*)$ remains close to the verified safe outcome $\boldsymbol{x}_{k+1}^{(j)}$. This deviation bound provides the foundation for a deployment-time safety condition that certifies the interpolated state remains strictly within the constraint-satisfying region.

In reality, control inputs are often applied with a delay due to factors such as actuator response time or signal propagation. To account for such delays during deployment, the safety condition can be reformulated to cover multiple timesteps under a held input. Specifically, if the control input $\boldsymbol{u}^*$ is applied over d consecutive steps, the bound in Equation 39 can be conservatively adjusted by replacing

Δt with $d \cdot \Delta t$. In addition, the Lipschitz constants L_x and L_u used in the deviation bound are scaled to reflect a worst-case growth over the extended horizon. This results in a more conservative estimate of the potential deviation, effectively introducing a safety margin that ensures constraint satisfaction even in the presence of known, fixed delays. By incorporating these modifications, the filtering scheme maintains deployment-time safety guarantees without requiring explicit multi-step simulation.

7. Deployment-Time Safety Condition

To certify safety of the interpolated input $\boldsymbol{u}^*$, the resulting state $\boldsymbol{x}_{\text{curr+1}}(\boldsymbol{u}^*)$ must remain within the constraint-satisfying region. For each neighbor $\boldsymbol{x}_k^{(j)}$, let the associated successor $\boldsymbol{x}_{k+1}^{(j)}$ be strictly safe, with a corresponding safety margin:

$$\left(d^{\text{safe}}\right)^{(j)} = \text{distance}\left(\boldsymbol{x}_{k+1}^{(j)},\ \partial\mathcal{X}^{\text{safe}}\right) \tag{40}$$

This margin is computed component-wise for each of the five state variables, yielding a vector $\left(d^{\text{safe}}\right)^{(j)} \in \mathbb{R}^5$, where $\left(d^{\text{safe}}\right)_i^{(j)}$ denotes the safe distance to the constraint boundary in the i-th state dimension for the j-th neighbor. Safety is guaranteed if the predicted deviation remains within this margin:

$$(1 + \Delta t L_x) \cdot \left\|\boldsymbol{x}_{\text{curr}} - \boldsymbol{x}_k^{(j)}\right\| + \Delta t L_u \cdot \left\|\boldsymbol{u}^* - \boldsymbol{u}_k^{(j)}\right\| < \left(d^{\text{safe}}\right)^{(j)} \tag{41}$$

If the condition holds for all neighbors, then $\boldsymbol{x}_{\text{curr+1}}(\boldsymbol{u}^*) \in \mathcal{X}_{\text{safe}}$. If the condition is violated for any neighbor, it suggests that $\boldsymbol{x}_{\text{curr}}$ lies in a sparse region of the database, where nearby verified trajectories are insufficiently close. In such cases, the controller can revert to a known-safe input or trigger database expansion to improve coverage. This mechanism enforces safety with minimal runtime overhead and acts as a data-driven safeguard during deployment.

8. Runtime Formulation

At runtime, the safety filter checks whether the interpolated control input $\boldsymbol{u}^*$, computed via nearest-neighbor interpolation, yields a safe next state. This is evaluated for each neighbor $\boldsymbol{x}_k^{(j)}$ involved in the interpolation. For each pair $(\boldsymbol{x}_k^{(j)}, \boldsymbol{u}_k^{(j)})$, the following deviation bound is computed:

$$\delta_k^{(j)} = (1 + \Delta t \cdot L_x) \left\|\boldsymbol{x}_{\text{curr}} - \boldsymbol{x}_k^{(j)}\right\| + \Delta t \cdot L_u \left\|\boldsymbol{u}^* - \boldsymbol{u}_k^{(j)}\right\| \tag{42}$$

This quantity bounds the deviation between the interpolated next state $\boldsymbol{x}_{\text{curr+1}}(\boldsymbol{u}^*)$ and the verified safe outcome $\boldsymbol{x}_{k+1}^{(j)}$, based on differences in state and control. The bound is compared to the available safety margin:

$$\delta_k^{(j)} < \left(d^{\text{safe}}\right)^{(j)}. \tag{43}$$

If this inequality holds for all selected nearest neighbors $j = 1, \ldots, N_{\text{NN}}$ used in the interpolation, the input $\boldsymbol{u}^*$ is deemed safe and applied. If not, the controller expands the database with new safe rollouts from the current state. This ensures safety during deployment while supporting generalization to previously unseen states.

9. Estimating Lipschitz Constants for Safety Bounds

To evaluate deployment-time safety, Lipschitz constants are required to characterize how sensitively the system responds to small changes in the control input and the state. These constants are used to bound the deviation of the interpolated next state $\boldsymbol{x}_{k+1}(\boldsymbol{u}^*)$ from its nearest certified neighbor. This section defines the estimation process.

A scalar constant L_u captures the sensitivity to perturbations in the control input:

$$L_u \approx \sup_{\Delta \boldsymbol{u}} \frac{\|\boldsymbol{f}(\boldsymbol{x}_{\text{curr}}, \boldsymbol{u}^* + \Delta \boldsymbol{u}) - \boldsymbol{f}(\boldsymbol{x}_{\text{curr}}, \boldsymbol{u}^*)\|^{\text{scaled}}}{\|\Delta \boldsymbol{u}\|} \tag{44}$$

Here, $\boldsymbol{f}(\boldsymbol{x}_{\text{curr}}, \boldsymbol{u}^*) \in \mathbb{R}^5$ is the discrete-time state update evaluated at the current state and interpolated control input. The norm is computed using a scaled Euclidean form:

$$\|\boldsymbol{f}\|^{\text{scaled}} = \sqrt{\left(\frac{v_k^h}{c_1}\right)^2 + \left(\frac{v_k^\alpha}{c_2}\right)^2 + \left(\frac{a_k^h}{c_3}\right)^2 + \left(\frac{a_k^\alpha}{c_4}\right)^2 + \left(\frac{v_k^\beta}{c_5}\right)^2} \tag{45}$$

The constants $c_1, \ldots$ are nominal scaling factors that reflect typical magnitudes of each component. These values are based on simulation data or heuristic expectations to prevent any term, such as a_k^α from dominating the norm.

Unlike the scalar L_u, state sensitivity defines one Lipschitz constant per output dimension, making $\boldsymbol{L}_x$ a vector:

$$L_{x_i} = \sup_{\Delta \boldsymbol{x}_{\text{curr}}} \frac{|f(\boldsymbol{x}_{\text{curr}} + \Delta \boldsymbol{x}_{\text{curr}}, \boldsymbol{u}^*) - f(\boldsymbol{x}_{\text{curr}}, \boldsymbol{u}^*)|}{\|\Delta \boldsymbol{x}_{\text{curr}}\|} \tag{46}$$

For $i = 1, \ldots, n$, this produces $\boldsymbol{L}_x = [L_{x_1}, \ldots, L_{x_n}]$. The control input $\boldsymbol{u}^*$ is held fixed, so the estimate isolates the system's sensitivity to each individual state variable. In practice, each component L_{x_i} is approximated using:

$$L_{x_i} \approx \left\| \frac{\boldsymbol{f}(\boldsymbol{x}_{\text{curr}} + \delta \boldsymbol{e}_i, \boldsymbol{u}^*) - \boldsymbol{f}(\boldsymbol{x}_{\text{curr}}, \boldsymbol{u}^*)}{\delta} \right\| \tag{47}$$

Here, $\boldsymbol{e}_i$ is the unit vector in the direction of the i-th state variable, and δ is a small scalar perturbation. This approach yields a conservative estimate of the directional sensitivity in each state dimension.

Because the system is nonlinear and state-dependent, the Lipschitz constants L_u and $\boldsymbol{L}_x$ are recomputed online at each deployment step using the current state $\boldsymbol{x}_{\text{curr}}$ and interpolated input $\boldsymbol{u}^*$, ensuring the bound remains valid in the local region around the interpolated trajectory.

The predicted deviation of the interpolated next state is computed as:

$$\begin{aligned} \boldsymbol{\delta}^{(j)} = &\left\| \boldsymbol{x}_{\text{curr}} - \boldsymbol{x}_k^{(j)} \right\| + \Delta t \cdot \boldsymbol{L}_x \left\| \boldsymbol{x}_{\text{curr}} - \boldsymbol{x}_k^{(j)} \right\| \\ &+ \Delta t \cdot L_u \cdot \left| u^* - u_k^{(j)} \right| \end{aligned} \tag{48}$$

This yields a component-wise deviation vector, where each element corresponds to a specific state variable. The result is compared to the component-wise safety margin $(d^{\text{safe}})_i^{(j)}$ for each dimension i. The safety condition is satisfied if:

$$\delta_i^{(j)} < (d^{\text{safe}})_i^{(j)} \quad \text{for all } i = 1, \ldots, 5 \tag{49}$$

10. Summary and Conclusions

This section presented a Lipschitz-based safety filter for runtime deployment of learned controllers. By interpolating control actions from a local database of verified safe transitions, and bounding deviations using locally estimated Lipschitz constants L_u and L_x, the filter certifies that the resulting one-step state remains within the safe set. The method is lightweight, modular, and simulation-free, in contrast to many rollout-based safety filters that require online forward simulation to predict outcomes. By relying solely on local state-control data and deviation bounds, the approach enables safety generalization to previously unseen states without incurring simulation overhead. The formulation can also be extended to account for known actuation delays by conservatively adjusting the deviation bound over multiple timesteps. This allows the filter to

maintain guarantees even when control inputs are applied with a fixed delay. If a violation is detected, the controller triggers database expansion to recover coverage. This enables reliable policy deployment with formal safety guarantees and minimal runtime overhead in nonlinear systems.

III. Results

A. Simulation Context

This section compares the three controllers, LPV-MPC, RL, and a coupled RL-MPC design under a representative turbulence scenario to assess their relative effectiveness in suppressing transient aeroelastic responses in a simulation. The model is a 2D aeroelastic wing section and does not correspond to a specific aircraft. As such, absolute performance figures cannot be directly interpreted. Instead, RL-MPC is taken as the baseline, as it represents the proposed hybrid approach. Using it as a reference highlights the specific advantages and limitations of individual pure RL and LPV-MPC controllers when evaluated under identical conditions. This framing allows the analysis to focus on how much the RL-MPC design improves over standalone controllers. Each simulation spans 10 s. A lateral gust, synthesized using the Dryden turbulence model, is applied during the first 5 s, followed by a 5 s recovery phase. To emulate real-world conditions, all controllers operate with a fixed actuation delay of 8 ms (eight discrete steps at $\Delta t = 0.001$ s).

To illustrate typical behavior, the time-domain response of each controller is first examined in a representative run. The analysis is then extended to 1000 Monte Carlo simulations, each using a unique gust realization and initial condition. Turbulence is generated using the Dryden model with intensity $\sigma = 1\,\mathrm{m\,s^{-1}}$, correlation length $L = 5\,\mathrm{m}$, and freestream velocity $V = 12\,\mathrm{m\,s^{-1}}$. Each controller uses a sampling time of $T_s = 0.001$ s, with tailored parameters such as prediction horizon, exploration policy, and tuning weights. These were determined trough simulations and empirical adjustment to balance real-time feasibility with reliable performance across the tested gust scenarios.

While no universal tolerance thresholds for plunge or angle of attack are defined, the aim is to show how consistently each controller limits excursions and accelerates recovery under uncertainty. These findings, while not conclusive, provide insights into how such control strategies might generalize to more realistic aeroelastic systems.

B. Representative Run: Time-Domain Behavior

Before presenting statistical results, this subsection highlights the time-domain behavior of each controller under the same gust, with the open-loop response included for reference. Figures 4, 5, and 6 show the angle of attack, plunge displacement, and control input respectively.

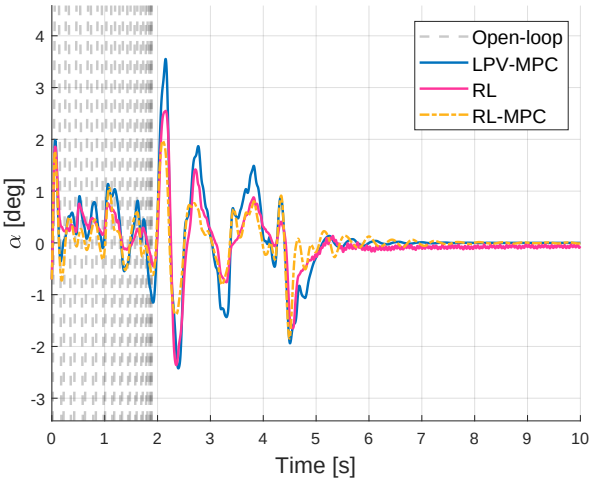


Fig. 4 Angle of attack response.

Table 4 Description of performance metrics used in Section III.

Metric	Description
O_h, O_α	Maximum overshoot in plunge (mm) and angle of attack (rad) following a disturbance.
$T_{s,h}$, $T_{s,\alpha}$	Time for plunge and angle of attack to settle within a tolerance band around the final value.
Excursion count (h, α)	Number of times the plunge or angle of attack exceeds defined safe bounds.
RMS $\dot{h}$, RMS $\dot{\alpha}$	Root-mean-square of plunge and AoA velocities, indicating system smoothness.
Median $\lvert \Delta u_k \rvert$	Change in control input per time step, as a percentage of the maximum allowed input.
E^h_{Low}, E^h_{Mid}, E^h_{High}	plunge error energy in 0.1–1, 1–5, 5–50 Hz bands ($\text{mm}^2\cdot\text{s}$)
E^α_{Low}, E^α_{Mid}, E^α_{High}	AoA error energy in 0.1–1, 1–5, 5–50 Hz bands ($\text{deg}^2\cdot\text{s}$)

Figure 4 shows that the RL-MPC design provides the strongest suppression of the gust-induced spike in angle of attack, with the lowest peak deviation (0.034 rad). The RL controller follows with a slightly higher peak (0.044 rad), while LPV-MPC exhibits the weakest gust rejection, allowing the highest overshoot (0.062 rad). In terms of recovery, LPV-MPC settles fastest (5.73 s), followed by RL-MPC (7.06 s) and RL (10.0 s). Since large excursions in angle of attack are a primary driver of unsteady lift and stall risk, limiting peak values is often more critical than achieving perfectly smooth convergence. The RL-MPC trade-off, smaller peaks with tolerable ripple is considered favorable. Notably, the open-loop response diverges rapidly under the same gust, with unstable oscillations that grow unbounded. This outcome occurs in nearly all trials, illustrating the severity of the aeroelastic instability. As a result, all three controllers are effectively performing both flutter suppression and GLA simultaneously. The ability to stabilize this highly nonlinear and unstable system, while also reducing gust-induced load excursions, highlights the difficulty of the control task and the effectiveness of each method.

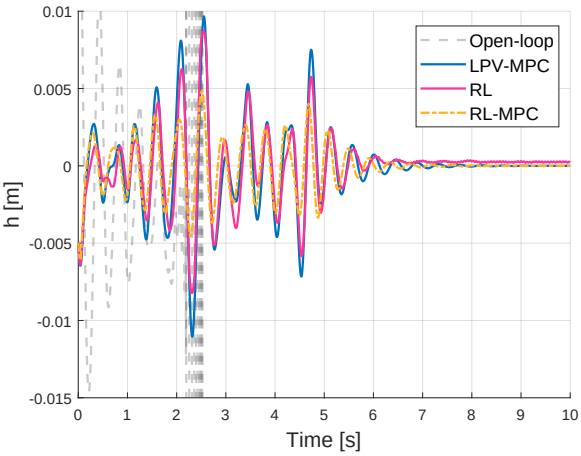


Fig. 5 Plunge displacement response.

Figure 5 illustrates the superior disturbance rejection of the RL-MPC controller in the structural channel. It shows the smallest plunge overshoot (0.0061 m) and fastest recovery (1.99 s), outperforming both LPV-MPC (0.0111 m, 2.36 s) and RL (0.0087 m, 4.90 s). This quick and clean response is particularly valuable in aeroelastic systems where vertical displacement contributes to fatigue and long-term structural wear. By limiting wear, the longevity of vital structural elements is preserved. LPV-MPC, while smooth, is noticeably slower to settle, while RL reacts quickly but lacks the damping to return efficiently to steady state. As in the angle of attack case, the open-loop response diverges rapidly, confirming that without active control the plunge dynamics become unstable and unbounded.

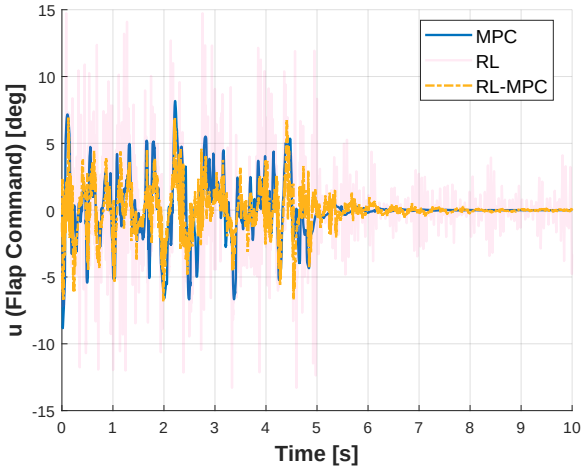


Fig. 6 Control input (flap command) over time.

Figure 6 reveals the trade-offs in control activity. The LPV-MPC controller provides the smoothest input, minimizing actuator workload through conservative but steady actuation. The RL controller, issues nearly saturated commands at every timestep during the gust phase, reflecting its lack of internal smoothness constraints and strong preference for rapid correction. The RL-MPC controller sits between these two extremes, it applies more dynamic inputs than LPV-MPC, but less aggressive than RL, suggesting a more measured yet responsive control strategy. Once the gust subsides, LPV-MPC reduces its control input cleanly to near zero, indicating strong convergence. RL and RL-MPC retain small residual oscillations in their post-gust input signals, likely due to their higher reactivity and learning-based policies. This reflects a trade-off: while RL-based controllers developed in this work offer faster suppression, they do so at the cost of post-disturbance smoothness. From a purely actuator-efficiency standpoint, LPV-MPC retains a clear advantage.

C. Monte Carlo Analysis

To evaluate controller performance under turbulence, a Monte Carlo campaign was conducted using 1 000 randomized gust realizations. Each metric reflects the controller's behavior over the full ensemble. For error-based quantities, box plots are constructed from all individual time-step values across all runs, capturing the full distribution of point wise tracking performance rather than aggregated summaries. This enables a more detailed view of instantaneous accuracy and variability throughout the simulation horizon.

Overshoot and settling times

Controller	O_h [mm]	$T_{s,h}$ [s]	O_α [rad]	$T_{s,\alpha}$ [s]
LPV-MPC	0.0080	2.47	0.0398	0.83
RL	0.0084	4.78	0.0416	4.42
RL-MPC	0.0073	2.07	0.0318	1.09

Table 5 Time-domain metrics (mean over 1 000 runs).

Table 5 summarizes peak overshoot (O) and settling time (T_s). For plunge, the RL-MPC controller records an average overshoot of 7.3 µm, improving on LPV-MPC by 9%, and settles 16% faster with a mean time of 2.07 s. The pure RL controller is also by RL-MPC in this metric posting a 15% higher overshoot and settling 130% slower than RL-MPC. These results highlight RL-MPC's ability to rapidly damp structural motion with minimal vertical displacement, outperforming both alternatives on both metrics.

For angle of attack, RL-MPC achieves the lowest average overshoot at 0.0318 rad. Compared to this, LPV-MPC allows 25% higher peak excursions, while RL exceeds it by 31%. In terms of settling time, LPV-MPC converges

24% faster than RL-MPC, while RL is 306% slower. These results reflect a trade-off: RL-MPC prioritizes immediate suppression of gust-induced deviations, while LPV-MPC favors smoother but slower recovery. Overall, RL-MPC demonstrates the best capability for limiting peak aerodynamic loads, even if it does not converge as quickly as LPV-MPC in the post-gust phase.

Angle-of-attack dynamics

This section presents a focused evaluation of the angle of attack dynamics across all Monte Carlo simulations for the three control strategies. First, box plots of the angle of attack histories are shown for the full simulation duration and the post-gust window to highlight differences in response across runs. This is followed by a summary table reporting the number of angle of attack limit excursions and pitch rate statistics aggregated over all simulations.

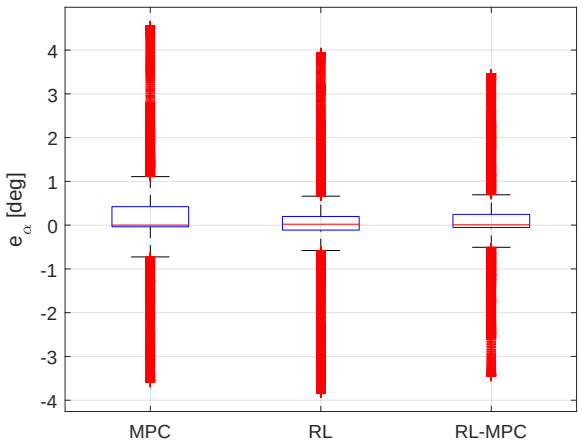


Fig. 7 Angle of attack error (e_α) distribution over the full simulation duration. Central mark = median; boxes = Q1–Q3; whiskers = adjacent values

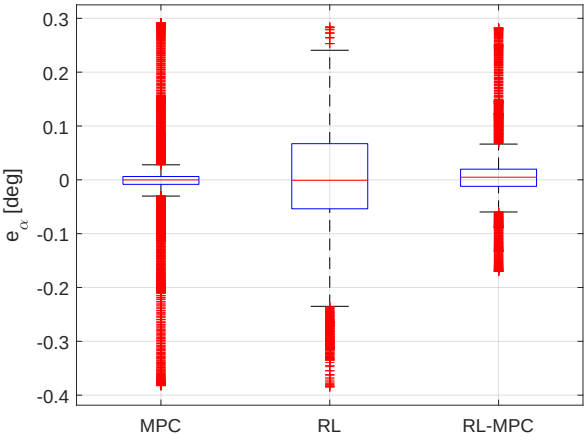


Fig. 8 Angle of attack error (e_α) distribution in the post-gust window

Controller	Avg. excursions	RMS $\dot{\alpha}$ $\left[\frac{\text{rad}}{\text{s}}\right]$(full)	RMS $\dot{\alpha}$ $\left[\frac{\text{rad}}{\text{s}}\right]$(post-gust)
LPV-MPC	5.1973	0.1699	0.0108
RL	3.6255	0.1654	0.0479
RL-MPC	3.1479	0.1473	0.0243

Table 6 Angle-of-attack metrics over 1 000 runs. Excursion threshold: 20% of max gust-induced α.

The angle of attack error box plots (Figures 7 and 8) reveal that the RL-MPC controller exhibits the narrowest spread over the full 10-second duration, with an interquartile range (IQR) of approximately 0.30°, compared to 0.31° for RL and 0.46° for MPC. This indicates that RL-MPC is most effective at rejecting large gusts, resulting in the most consistent performance throughout the run. However, in the post-gust window, LPV-MPC achieves the smallest IQR of about 0.0146°, outperforming RL-MPC (0.0319°) and RL (0.1209°). These results suggest that while RL-MPC offers the most stable tracking overall, MPC demonstrates superior convergence after the gust subsides.

Table 6 presents two additional metrics, the number of excursions and angle of attack velocity. RL-MPC shows

the fewest excursions above the 20% threshold, reducing them by 39% compared to LPV-MPC and 13% compared to RL, demonstrating its predictive capability in rejecting large gusts. It also achieves the lowest RMS pitch rate, indicating smallest plunge variations. While overall performance on this metric is comparable across controllers, RL-based strategies perform better during the gust, whereas LPV-MPC shows improved stabilization after the gust.

In summary, the RL-MPC controller offers the most stable angle-of-attack behavior overall by minimizing both the spread of tracking error and the number of moderate excursions. Its ability to anticipate gusts results in smoother and more consistent corrective motion across the entire simulation. However, once the gust subsides, the LPV-MPC controller exhibits the best convergence, as indicated by its tighter post-gust tracking and reduced pitch activity.

Plunge dynamics

This section presents a focused evaluation of the plunge dynamics across all Monte Carlo simulations for the three control strategies.

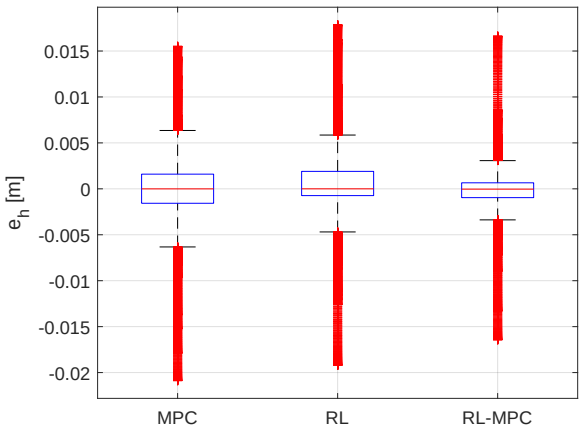


Fig. 9 Plunge error (e_h) distribution full run

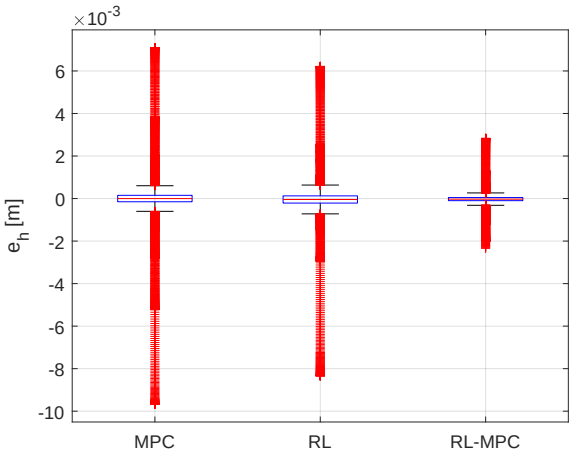


Fig. 10 Plunge error (e_h) distribution post-gust

Controller	Avg. excursions	RMS $\dot{h}$ $\left[\frac{\text{m}}{\text{s}}\right]$ (full)	RMS $\dot{h}$ $\left[\frac{\text{m}}{\text{s}}\right]$ (post-gust)
LPV-MPC	4.7553	0.04216	0.0095
RL	4.4891	0.03958	0.0101
RL-MPC	3.1257	0.03512	0.0093

Table 7 Plunge metrics over 1 000 runs. Excursion threshold: 20% of max open-loop gust response.

The plunge error box plots (Figures 9 and 10) show that the RL-MPC controller achieves the narrowest spread over the full duration, with an interquartile range (IQR) of approximately 1.61×10^{-3} m, compared to 2.64×10^{-3} m for RL and 3.17×10^{-3} m for MPC. This indicates that RL-MPC is most effective at maintaining plunge stability throughout the simulation. In the post-gust window, RL-MPC again achieves the tightest distribution with an IQR of about 1.46×10^{-4} m, outperforming both MPC and RL. These results suggest that RL-MPC consistently delivers the most stable plunge control.

Table 7 summarizes the number of significant plunge excursions and the average vertical velocity across all simulations. RL-MPC shows the strongest performance on both metrics,

with a 52% reduction in excursion events compared to LPV-MPC and 44% compared to RL, indicating enhanced suppression of vertical deflections under gusts. It also achieves the lowest plunge velocity, suggesting improved structural stability and comfort. While all controllers converge to similar behavior post-gust, RL-MPC retains a slight advantage.

Altogether, RL-MPC provides the most consistent plunge control by minimizing both displacement variability and vertical motion across all phases of the run. This reinforces its effectiveness in mitigating structural loads and maintaining altitude stability under turbulent conditions.

Control Energy Response

This section analyzes the control effort required by each controller by examining the per-step variation in elevator deflection, providing insight into actuator usage and overall control smoothness. This is particularly important because excessive actuator activity can lead to increased wear, energy consumption, and reduced system longevity.

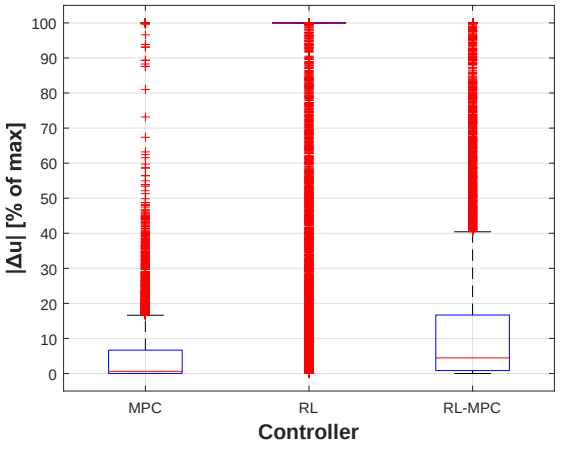


Fig. 11 Box-plot of per-step elevator increment $|\Delta u_k| = |u_k - u_{k-1}|$ for the three controllers.

Controller	LowerAdj	Q1	Median	Q3	UpperAdj
LPV-MPC	0	0.0218	0.6224	6.6645	16.625
RL	100	100	100	100	100
RL-MPC	0	0.8424	4.4483	16.681	40.426

Table 8 Five-number summary of $|\Delta u_k|$ (% of $\Delta u_{\max}$).

Table 8 and Fig. 11 compare the smoothness of control input by showing the distribution of instantaneous changes $|\Delta u_k|$. The RL controller, lacking any internal mechanism for penalizing actuator variation, saturates the servo rate limit and issues nearly constant full-rate steps, resulting in a collapsed box-plot where all summary statistics coincide at 100% of the maximum input variation.

In contrast, the LPV-MPC and RL-MPC controllers exhibit much smoother control. The RL-MPC achieves a median $|\Delta u_k|$ at 4.45% of the maximum, while LPV-MPC performs significantly better with a median of just 0.62%, reflecting an 86% reduction in actuator activity. While RL-MPC blends some of the responsiveness of RL with the moderation of LPV-MPC, it is the pure LPV-MPC controller that most effectively limits actuator movement. This makes LPV-MPC the clear choice when actuator wear, energy usage, or long-term durability are primary concerns.

Frequency Domain Control Performance

Table 9 Average error energy for three octave-like bands (Low 0.1–1 Hz, Mid 1–5 Hz, High 5–50 Hz).

(a) Plunge error energy E^h ($\text{mm}^2\cdot\text{s}$)

Controller	Low	Mid	High
LPV-MPC	2.50×10^{-4}	4.60×10^{-3}	2.50×10^{-5}
RL	4.35×10^{-4}	5.46×10^{-3}	2.79×10^{-5}
RL-MPC	4.12×10^{-5}	2.59×10^{-3}	1.19×10^{-5}

(b) AoA error energy E^{α} (deg^2·s)

Controller	Low	Mid	High
LPV-MPC	0.04781	0.05221	0.00264
RL	0.05608	0.05201	0.00238
RL-MPC	0.02018	0.08440	0.00337

For the Plunge, RL-MPC achieves the lowest error energy across all three frequency bands, offering markedly improved performance over both LPV-MPC and RL. Its most significant advantage lies in the low-frequency band, where suppression is crucial to avoid build-up of large structural motions that can lead to fatigue or instability.

In the AoA channel, performance is more nuanced. RL-MPC clearly excels in the low-frequency band, halving the error energy relative to LPV-MPC, which is important for maintaining long-term orientation and flight stability. However, this comes at the cost of increased energy in the mid- and high-frequency bands. Specifically, RL-MPC's mid-band energy is 62% higher than LPV-MPC and RL, while in the high band it trails LPV-MPC and RL by 28% and 42%, respectively. RL yields the best high-frequency performance overall, which translates to smoother angular motion and reduced actuator strain during rapid maneuvers.

Overall, RL-MPC delivers the most consistent plunge attenuation and the best low-frequency AoA tracking, while LPV-MPC remains superior in mid-frequency AoA suppression and RL leads in high-frequency noise rejection.

IV. Discussion

A. General Discussion

Justification for Truncating the Taylor Expansion

The safety guarantees in this work are based on bounding the second-order error in the Taylor expansion of state deviations. The third-order term is not analyzed, but for the system considered, it was negligible relative to the second-order term due to smooth dynamics and small time step Δt. Specifically, numerical analysis showed its contribution was at most 0.291% for the aeroelastic states, with a maximum of 2.0% for the actuator state, as detailed in Table 2. However, this assumption is system-specific. Designers must verify the negligibility of higher-order terms for systems under consideration, as system dynamics become faster or the time step Δt increases, the minimum required order for accurate analysis tends to increase. In other settings, third-order effects may be non-negligible, and ignoring them could undermine the guarantees. While future work could aim to mathematically bound the third-order term, a more practical alternative is to define conditions based on system smoothness and Δt under which the third-order contribution can be safely neglected.

Limitations of Discrete-Time Guarantees

While the real-world system evolves in continuous time, all safety guarantees in this work are derived in a discrete-time setting. The analysis assumes a fixed time step Δt and provides bounds on the discrete evolution of the system states. This discretized model approximates the continuous dynamics under the assumption of sufficiently small Δt and smooth system behavior. However, formal guarantees in continuous time are not provided, and the discrete-time

results do not automatically generalize. Bridging this gap remains an open direction, where future work could investigate whether the bounds established here converge to valid continuous-time guarantees as $\Delta t \to 0$.

Applicability to Other Systems

The bounding analysis in this work relies on identifying monotonic relationships between the control input and the state updates through structured intermediate variables, such as the flap deflection. This approach is likely to transfer well to systems with few state variables, smooth nonlinearities, and control effects that propagate through stable, monotonic channels. In particular, systems with single-input, single-path actuation and affine dependencies on intermediate variables are well-suited to this method. However, for higher-dimensional systems with multiple interacting control channels, strong state coupling, or non-monotonic input effects, the assumptions underpinning the bounding procedure may no longer hold. In such cases, more sophisticated analysis would be required to ensure that intermediate control values produce bounded trajectories.

Actuator Delay Compensation

A fixed actuator delay is included to reflect real-world latency in control signal execution. All controllers handle this delay similarly: they extrapolate the current state from state history, moving it forward by the measured delay and apply their policies to this delay-free estimate. While the RL-based designs only need to predict a few delayed steps, LPV-MPC must forecast both the delay and its full prediction horizon. This gives RL a slight edge in state reconstruction, but overall the shared compensation strategy ensures a fair comparison.

The role of MPC in RL-MPC Architecture

In the RL-MPC architecture, MPC plays a different role than in stand-alone use. The standalone LPV-MPC serves as the onboard controller, operating under tight real-time constraints that limit its prediction horizon to short durations ($N^{\mathrm{P}} \approx 20$). Inside the RL-MPC, however, MPC is used only during offline policy generation, where computation time is no longer critical and the horizon can be extended until the slowest aero-elastic modes are fully captured. Additionally, because the policy generation is performed offline, the LPV-MPC can be constructed in a more adaptable and iterative manner. Unlike real-time operation where the entire prediction horizon must be computed before an output is given, the offline nature allows for the step-by-step construction of specific trajectories. This means that not only can we achieve super accurate LPV-MPC with super long prediction horizons, but we can also leverage our prior knowledge of the gusts. Since we are in a training environment, we choose the gust profiles that the controller will experience. This controlled environment allows us to build incredibly accurate trajectories because we know exactly what external disturbances the system will face, enabling precise refinement of the control actions. The result is a set of long-horizon, constraint-satisfying trajectories that the RL agent can learn from, something an online LPV-MPC of practical length could never see. This further validates the superior effectiveness of the RL-MPC architecture, especially in handling complex gust profiles, which is consistently demonstrated in the simulation results. The RL-MPC controller is also inherently more robust to gust uncertainty than the on-board LPV-MPC alone. During training it is exposed not just to the nominal gust prediction but to worst and best case envelopes, so the

learned policy internalizes margins that tube-MPC would normally enforce. Achieving similar guarantees is possible with tube-MPC, however, it would require long prediction horizons and large invariant sets, making it impractical for online deployment. By off-loading that burden to the training phase, the RL–MPC design preserves long-horizon performance guarantees while remaining suitable for real-time execution on embedded hardware.

Computational Cost and Runtime Comparison

Computation times were measured on a standard desktop PC (AMD Ryzen 5 5600X, 6 cores @ 3.70 GHz, 16.0 GB RAM) to give a rough sense of relative complexity. Average runtimes were 11.1 s for LPV-MPC, 7.9 s for RL, and 10.7 s for RL-MPC. These values are not definitive, but illustrate typical performance under the current setup. RL-MPC is slightly slower than RL due to storing value estimates over a 7-dimensional input space instead of 5, reflecting its inclusion of MPC state information. Replacing the Q-table with a neural network, as is common in deep RL, could reduce evaluation time and memory use, though this would add training complexity.

B. Results Based Discussion

1. Flutter Suppression

As is seen in the results in Section III, all three controllers are tasked with achieving both flutter suppression and GLA simultaneously. Each controller must first stabilize the flexible aeroelastic system to suppress flutter, and then, while maintaining that stability work to minimize structural loads caused by gusts. The results assess how well each controller performs GLA, since flutter suppression is achieved in nearly all cases and does not serve as a point of comparison. All controllers maintain stability under nominal conditions, so the evaluation focuses on their ability to mitigate structural loads during gusts.

However, it is important to note that in rare cases (0.6%), the LPV-MPC controller failed to suppress flutter under strong gust excitation, resulting in divergent states. An example of such a failure is shown in Figure 12. These cases were excluded from the statistical analysis for consistency, but they highlight the importance of robust stability when designing for both flutter suppression and GLA. RL-based controllers remained stable across all runs for the considered gust setup with $\sigma = 1$ m/s, reinforcing their reliability under this level of turbulence. However, for stronger gust intensities ($\sigma \geq 1.5$ m/s), all controllers can fail, depending on the specific gust realization. On average, the system fails to suppress roughly half of the $\sigma = 1.5$ m/s cases, and failure becomes nearly certain at higher intensities. These failures are not due to controller design limitations, but rather stem from the physical constraints of the system and actuators, which are unable to counteract the magnitude of such disturbances.

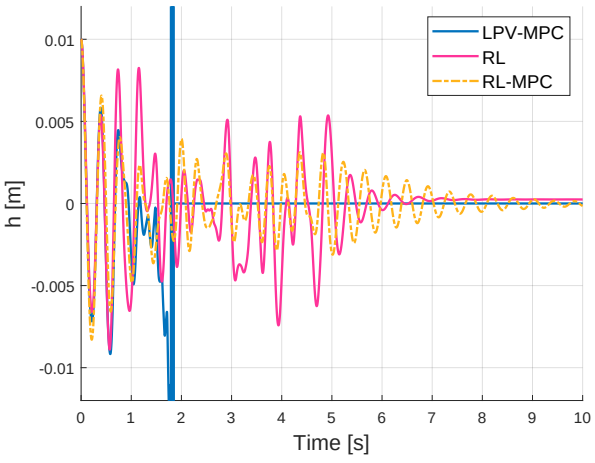


Fig. 12 Plunge response showing MPC failure

2. Gust Rejection and Post Gust Recovery Trade-off

The RL-MPC controller outperforms others by combining RL's ability to handle turbulence with MPC's precision in enforcing constraints once the disturbance passes. While RL and RL-MPC outperform LPV-MPC during gusts, LPV-MPC excels in post-gust conditions due to its close match between model and plant. This highlights a key limitation of standalone LPV-MPC: its prediction horizon is often too short to capture entire disturbances, causing degraded performance during gusts. However, in calm conditions, the model-plant match allows LPV-MPC to restore equilibrium. RL-MPC leverages the strengths of both approaches, learning disturbance rejection from RL and recovery behavior from MPC. As the RL policy used here is model-free, its post gust convergence is slower than the model-aligned LPV-MPC once the gust subsides.

Overshoot and settling-time metrics reinforce this pattern. RL–MPC clips the peaks most effectively in plunge and AoA because its training exposed it to a wide envelope of gust magnitudes, prompting aggressive, constraint-aware corrections. For plunge, it also settles fastest, yet in AoA the stand-alone LPV-MPC reaches the tolerance band sooner. This is likely due to biases in the RL training process, normalization may have overemphasized plunge, causing the RL-MPC controller to focus more on plunge errors than pitch convergence. Additionally, because the RL policies were trained primarily on gust scenarios, they tend to anticipate continued disturbances even after the gust has ended. This leads to unnecessarily aggressive corrections in the recovery phase. Additional training on no-gust scenarios could help reduce residual oscillations and improve post-gust settling. While RL performs well during the gust phase, its post-gust behavior suffers from actuator mismatches. Because it did not explicitly learn the actuator rate limits, the policy repeatedly issues infeasible deflection commands. This leads to clamping, which induces sustained input oscillations and increases the settling time.

3. Control Energy

The control-energy statistics reinforce earlier findings. Pure RL drives the elevator to its rate limit nearly every sample, with constraints enforced only by the hardware clamp; without it, the deflections would exceed physical limits. The RL-MPC policy inherits the LPV-MPC's move-rate penalty, resulting in smoother actuation, though still less restrained than the stand-alone LPV-MPC, which yields the smallest median and quartile increments. Time-traces show that after gusts, LPV-MPC damps its command to near-zero quickly, while RL–MPC continues to exhibit small oscillations and the standalone RL still issues near-maximum deflections, with the hardware clamp actively limiting the input even in the quiet phase.

4. Post Gust oscillations

The residual oscillations seen in RL-based controllers near equilibrium stem largely from the reward structure. Since accelerations $\ddot{h}$ and $\ddot{\alpha}$ are not included in the state vector, they are absent from the cost, and the agent lacks incentive to fully suppress minor oscillations. Including accelerations in the reward could improve this, but doing so would require estimating them at deployment, introducing additional uncertainty. Moreover, the current reward prioritizes immediate next-state performance. Near equilibrium, a longer reward rollout could help the policy anticipate and reduce future deviations, leading to smoother convergence.

Additionally separate training on no-gust scenarios could reduce these oscillatory effects, such exhaustive training is not scalable in practice. Real-world systems demand generalizable policies, and training with random gusts, as done here, provides broader applicability. RL-MPC performs better post gust, as the embedded MPC offers domain knowledge about settling behavior. However, since zero-gust conditions are underrepresented during training, the controller sometimes over corrects during the quiet phase, leading to small persistent oscillations.

5. Frequency-Domain Performace

The frequency-domain results confirm and nuance the time-domain story. For plunge, the RL-MPC controller is emphatically superior at every scale: an order-of-magnitude drop in slow 0.1–1 Hz energy eliminates long heave drifts, while halving the 1–5 Hz content and further reducing the 5–50 Hz tail suppress both structural bending and high-frequency vibration. AoA presents a trade-off: the RL-MPC again delivers the cleanest low-frequency response, vital for lift and trim. But its bias toward aggressive gust rejection permits more mid-band and high-band activity than pure LPV-MPC, which benefits from its perfectly matched linear model once the disturbance subsides. Pure RL, meanwhile, happens to be quietest above 5 Hz because its rate-clamped input effectively filters the fastest dynamics. Thus no single scheme dominates every AoA band, yet the RL-MPC offers the broadest attenuation in the plunge axis and the most important low-frequency benefit in pitch, while LPV-MPC remains the benchmark for fine, high-band AoA clean-up.

V. Conclusion

Modern flexible wings demand controllers that can suppress flutter and alleviate loads in real time, despite strong nonlinearities, actuator limits, and onboard computational constraints. Classical methods often fail under turbulence, while nonlinear approaches like (I)NDI or backstepping are unreliable in underactuated, uncertain systems such as aeroelastic wings. This paper proposes a hybrid RL–MPC architecture that uses offline LPV-MPC rollouts to train a constraint-aware RL policy for online use. The result is a lightweight controller that delivers effective flutter suppression and load alleviation while respecting constraints in nonlinear aeroelastic systems.

All tested controllers exhibit strong flutter suppression, with RL-based methods outperforming LPV-MPC in consistency under gusts. The RL–MPC controller combines the gust-phase agility of RL with the post-gust precision of LPV-MPC. It delivers the tightest plunge tracking, smallest overshoot in both states, and broadest low-frequency error attenuation, while retaining actuator smoothness much closer to LPV-MPC than pure RL. Stand-alone LPV-MPC still converges quickest in angle of attack once the disturbance vanishes, illustrating the value of an exact model in the linear regime, yet the RL-MPC offers the best overall compromise across all metrics examined.

While demonstrated on a nonlinear aeroelastic wing, the proposed RL–MPC framework is broadly applicable. Its core design, offline long-horizon constrained learning with lightweight online deployment suits a wide range of nonlinear systems with input and state constraints, especially where real-time computation is limited. This includes morphing aircraft, active aero surfaces, and even domains like robotics and automotive industry. More generally, it

provides a practical route for applying high-performance, constraint-aware RL controllers to nonlinear systems subject to disturbances and constraints where classical methods struggle.

Future work could focus on refining the reward or training on no-gust scenarios to reduce residual oscillations near equilibrium. Beyond this, two theoretical directions remain open: (i) bounding higher-order contributions to formally include them in the stability proof, and (ii) extending the analysis to guarantee convergence in addition to constraint satisfaction. addition to these, a practical next step would be to evaluate the controller in real-world environments, such as wind tunnel testing, to validate its effectiveness beyond simulation.

In summary, combining an offline, long-horizon LPV-MPC with an online, data-driven RL agent yields yields a controller that is both constraint-respecting and highly effective against nonlinear turbulence, providing a pragmatic path toward reliable active flutter suppression with simultaneous gust-load alleviation on resource-limited aircraft.

References

[1] Q. Guo, X. He, Z. Wang, and J. Wang, "Effects of wing flexibility on aerodynamic performance of an aircraft model," *Chinese Journal of Aeronautics*, vol. 34, no. 9, pp. 133–142, 2021. [Online]. Available: https://doi.org/10.1016/j.cja.2021.01.012

[2] T. F. Wunderlich, S. Dähne, L. Reimer, and A. Schuster, "Global aerostructural design optimization of more flexible wings for commercial aircraft," *Journal of Aircraft*, vol. 58, no. 849, 2021. [Online]. Available: https://doi.org/10.2514/1.C036301

[3] E. Livne, "Aircraft active flutter suppression: State of the art and technology maturation needs," U.S. Department of Transportation, Federal Aviation Administration, William J. Hughes Technical Center, Atlantic City, NJ, Tech. Rep. DOT/FAA/TC-18/47, 2019.

[4] W. L. Garrard and B. S. Liebst, "Active flutter suppression using eigenspace and linear quadratic design techniques," *Journal of Guidance, Control, and Dynamics*, vol. 8, no. 3, pp. 304–311, 1985. [Online]. Available: https://doi.org/10.2514/3.19980

[5] L. Chrif and Z. M. Kadda, "Aircraft control system using lqg and lqr controller with optimal estimation–kalman filter design," *Procedia Engineering*, vol. 80, pp. 245–257, 2014. [Online]. Available: https://doi.org/10.1016/j.proeng.2014.09.084

[6] A. Schoon and S. Theodoulis, "Review of h∞ static output feedback controller synthesis methods for fighter aircraft control," in *Proceedings of the AIAA SCITECH 2025 Forum*. AIAA, 2025, p. 25, iSBN: 978-1-62410-723-8. [Online]. Available: https://doi.org/10.2514/6.2025-2241

[7] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Dover Publications, 2014, [Edition unavailable]. [Online]. Available: https://www.perlego.com/book/1444508/optimal-control-linear-quadratic-methods-pdf

[8] G. Puyou and C. Berard, "Gain-scheduled flight control law for flexible aircraft: A practical approach," in *IFAC Proceedings Volumes*, vol. 40, no. 7, 2007, pp. 497–502, 17th IFAC Symposium on Automatic Control in Aerospace. [Online]. Available: https://doi.org/10.3182/20070625-5-FR-2916.00085

[9] A. Hjartarson, P. Seiler, and G. J. Balas, "Lpv analysis of a gain scheduled control for an aeroelastic aircraft," in *2014 American Control Conference*, 2014, pp. 3778–3783.

[10] R. Steffensen, A. Steinert, and E. J. J. Smeur, "Non-linear dynamic inversion with actuator dynamics: An incremental control perspective," *arXiv preprint arXiv:2201.09805*, 2022, version 2, last revised 25 Nov 2022. [Online]. Available: https://arxiv.org/abs/2201.09805

[11] A. Steinert, S. Raab, S. Hafner, F. Holzapfel, and H. Hong, "From fundamentals to applications of incremental nonlinear dynamic inversion: A survey on indi – part i," *Chinese Journal of Aeronautics*, 2025, available online 25 April 2025, In Press, Journal Pre-proof. [Online]. Available: https://doi.org/10.1016/j.cja.2025.103553

[12] M. Lungu, *Backstepping Control Method in Aerospace Engineering*. Academica Greifswald, January 2022.

[13] E. F. Camacho and C. Bordons Alba, *Model Predictive Control*, 2nd ed. Springer, 2004.

[14] T. He and W. Su, "Robust control of gust-induced vibration of highly flexible aircraft," *Aerospace Science and Technology*, vol. 143, 2023.

[15] M. D. V. Pereira, I. Kolmanovsky, C. E. Cesnik, and F. Vetrano, "Model predictive control architectures for maneuver load alleviation

in very flexible aircraft," in *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics (AIAA), 2019, p. 1591. [Online]. Available: https://doi.org/10.2514/6.2019-1591

[16] W. Gao, Y. Liu, Q. Li, and B. Lu, "Gust load alleviation of a flexible flying wing with linear parameter-varying modeling and model predictive control," *Aerospace Science and Technology*, vol. 155, p. 109671, 2024.

[17] T. He and W. Su, "Gust alleviation of highly flexible aircraft with model predictive control," in *AIAA SCITECH 2023 Forum*. American Institute of Aeronautics and Astronautics (AIAA), 2023, p. 586. [Online]. Available: https://doi.org/10.2514/6.2023-0586

[18] Y. Wang, A. Wynn, and R. Palacios, "Model-predictive control of flexible aircraft dynamics using nonlinear reduced-order models," *AIAA SciTech*, 2016.

[19] R. Bittner, H. Joos, T. Lombaerts, and Q. Chu, "Model predictive control for maneuver load alleviation," in *IFAC Proceedings Volumes*, vol. 45, 2012, pp. 199–204. [Online]. Available: https://doi.org/10.3182/20120823-5-NL-3013.00049

[20] J. M. Maciejowski and C. N. Jones, "Mpc fault-tolerant flight control case study: Flight 1862," *IFAC Proceedings Volumes*, vol. 36, no. 5, pp. 119–124, June 2003. [Online]. Available: https://doi.org/10.1016/S1474-6670(17)36480-7

[21] M. de F. V. Pereira, I. Kolmanovsky, and C. E. S. Cesnik, "Model predictive control with constraint aggregation applied to conventional and very flexible aircraft," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 431–437. [Online]. Available: https://doi.org/10.1109/CDC40024.2019.9029769

[22] A. R. Darwich Ajjour, "Gust and manoeuvre loads alleviation using upper and lower surface spoilers," Ph.D. dissertation, University of Bristol, 2022. [Online]. Available: https://research-information.bris.ac.uk/en/studentTheses/gust-and-manoeuvre-loads-alleviation-using-upper-and-lower-surfac

[23] T. Chaffre, J. Moras, A. Chan-Hon-Tong, J. Marzat, K. Sammut, G. Le Chenadec, and B. Clement, "Learning-based vs model-free adaptive control of a mav under wind gust," *arXiv preprint arXiv:2101.12501*, 2021.

[24] N. Schaff, "Online aircraft system identification using a novel parameter informed reinforcement learning method," *Engineering Applications of Artificial Intelligence*, 2023. [Online]. Available: https://commons.erau.edu/edt/779/

[25] R. Konatala, D. Milz, C. Weiser, G. Looye, and E. van Kampen, "Flight testing reinforcement-learning-based online adaptive flight control laws on cs-25-class aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 47, no. 11, November 2024. [Online]. Available: https://doi.org/10.2514/1.G008321

[26] K. P. Haughn, C. Harvey, and D. Inman, "Deep reinforcement learning reveals fewer sensors are needed for autonomous gust alleviation," *arXiv preprint arXiv:2304.03133*, 2023.

[27] K. Dally and E.-J. van Kampen, "Soft actor-critic deep reinforcement learning for fault tolerant flight control," in *AIAA SCITECH 2022 Forum*. San Diego, CA, USA: American Institute of Aeronautics and Astronautics (AIAA), January 2022, p. 2078, published Online: 29 Dec 2021, Session: Autonomy for Space and Surface In-Situ Assembly II. [Online]. Available: https://doi.org/10.2514/6.2022-2078

[28] M. Zahmatkesh, S. A. Emami, A. Banazadeh, and P. Castaldi, "Attitude control of highly maneuverable aircraft using an improved q-learning," *arXiv preprint arXiv:2210.12317*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.12317

[29] C. F. O. da Silva, A. Dabiri, and B. D. Schutter, "Integrating reinforcement learning and model predictive control with applications to microgrids," *arXiv preprint*, vol. arXiv:2409.11267, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2409.11267

[30] D. Sun, A. Jamshidnejad, and B. D. Schutter, "A novel framework combining mpc and deep reinforcement learning with application to freeway traffic control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 7, pp. 6756–6769, 2024, published under a Creative Commons License.

[31] M. Zanon and S. Gros, "Safe reinforcement learning using robust mpc," *IEEE Transactions on Automatic Control*, 2021, originally submitted to arXiv as 1906.04005v2 on 10 Jun 2019, last revised 17 Aug 2020. [Online]. Available: https://doi.org/10.1109/TAC.2020.3024161

[32] A. B. Kordabad, D. Reinhardt, A. S. Anand, and S. Gros, "Reinforcement learning for mpc: Fundamentals and current challenges," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 5773–5780, 2023, published in January 2023. [Online]. Available: https://doi.org/10.1016/j.ifacol.2023.10.548

[33] K. Seel, A. Bemporad, S. Gros, and J. T. Gravdahl, "Variance-based exploration for learning model predictive control," *IEEE Access*, vol. 11, pp. 60 724–60 736, 2023, published under Creative Commons License. [Online]. Available: https://doi.org/10.1109/ACCESS.2023.3282842

[34] S. Adhau, S. Gros, and S. Skogestad, "Reinforcement learning based mpc with neural dynamical models," *European Journal of*

Control, vol. 101, 2024, open Access under Creative Commons license. [Online]. Available: https://doi.org/10.1016/j.ejcon.2024.101048

[35] S. Gros and M. Zanon, "Data-driven economic nmpc using reinforcement learning," *arXiv preprint arXiv:1904.04152*, 2019, published in IEEE Transactions on Automatic Control, vol. 65, no. 2, pp. 636-648, Feb. 2020, arXiv:1904.04152v1 [cs.SY]. [Online]. Available: https://doi.org/10.48550/arXiv.1904.04152

[36] F. Airaldi, B. D. Schutter, and A. Dabiri, "Reinforcement learning with model predictive control for highway ramp metering," *arXiv preprint arXiv:2311.08820*, 2023, submitted to IEEE Transactions on Intelligent Transportation Systems. [Online]. Available: https://doi.org/10.48550/arXiv.2311.08820

[37] T. R. Beal, "Digital simulation of atmospheric turbulence for dryden and von kármán models," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 1, pp. 143–145, 1993. [Online]. Available: https://doi.org/10.2514/3.11437

Part III

Additional Results

6

More Results

6.1. Second-Order Discretization of Nonlinear Dynamics

The system is analyzed by performing **two sequential Euler steps**, each with step size Δt, to evaluate how the state evolves over two time steps. This procedure captures the **delayed response of the input** (due to actuator filtering) and the **nonlinear effects** introduced by gusts and aerodynamic couplings. The first Euler step evolves the system as:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta t \cdot \boldsymbol{f}(\boldsymbol{x}_k, u_k, w_k) \tag{6.1}$$

where the state and corresponding dynamics are:

$$\boldsymbol{x}_k = \begin{bmatrix} h_k \\ \alpha_k \\ v_k^h \\ v_k^\alpha \\ \beta_k \end{bmatrix}, \quad \boldsymbol{f}(\boldsymbol{x}_k, u_k, w_k) = \begin{bmatrix} v_k^h \\ v_k^\alpha \\ a_k^h(M^{\text{mass}}, L_k^{\text{aero}}, v_k^h, h_k) \\ a_k^\alpha(M^{\text{mass}}, M_k^{\text{aero}}, v_k^\alpha, \alpha_k) \\ k_\beta(u^{\text{flap}} - \beta_k) \end{bmatrix} \tag{6.2}$$

The second Euler step follows from the expression derived in Equation

$$\boldsymbol{x}_{k+2} \approx \boldsymbol{x}_k + 2\Delta t \cdot \boldsymbol{f}_k + \Delta t^2 \cdot \nabla_{\boldsymbol{x}} \boldsymbol{f}_k \cdot \boldsymbol{f}_k + \mathcal{O}(\Delta t^3) \tag{6.3}$$

The explicit matrix form of this expansion is:

$$\boldsymbol{x}_{k+2} = \boldsymbol{x}_k + \Delta t \cdot \begin{bmatrix} x_3 \\ x_4 \\ a_k^h(M^{\text{mass}}, L_k^{\text{aero}}, x_3, x_1) \\ a_k^\alpha(M^{\text{mass}}, M_k^{\text{aero}}, x_4, x_2) \\ k_\beta(u^{\text{flap}} - x_5) \end{bmatrix} + \frac{\Delta t^2}{2} \cdot \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{\partial a_k^h}{\partial x_1} & \frac{\partial a_k^h}{\partial x_2} & \frac{\partial a_k^h}{\partial x_3} & \frac{\partial a_k^h}{\partial x_4} & \frac{\partial a_k^h}{\partial x_5} \\ \frac{\partial a_k^\alpha}{\partial x_1} & \frac{\partial a_k^\alpha}{\partial x_2} & \frac{\partial a_k^\alpha}{\partial x_3} & \frac{\partial a_k^\alpha}{\partial x_4} & \frac{\partial a_k^\alpha}{\partial x_5} \\ 0 & 0 & 0 & 0 & -k_\beta \end{bmatrix} \cdot \boldsymbol{f}(\boldsymbol{x}_{k+1}, u_{k+1}, w_{k+1}) \tag{6.4}$$

Each Jacobian term in this matrix depends on the derivatives of the nonlinear stiffness terms $k_h(x_1)$ and $k_a(x_2)$, as well as the aerodynamic forces $L^{\text{aero}}(\alpha_{\text{eff}}, x_5(u))$ and $M^{\text{aero}}(\alpha_{\text{eff}}, x_5(u))$, where x_5 is the flap state influenced by the control input. Additionally, gust influence appears through the chain rule in the derivatives of α^{eff} with respect to the state variables $x_2, x_3,$ and x_4, due to its implicit effect on angle of attack and aerodynamic coupling.

The second derivatives are computed from the flexible aircraft dynamics using the coupled system:

Constants and Parameters:

$$b = 0.135\ \text{m}, \quad \rho = 1.225\ \text{kg/m}^3, \quad a_b = 0.5 - 0.6847, \quad x_a = \frac{0.0873 - (b - 0.6847b)}{b}$$
$$m_t = 12.387\ \text{kg}, \quad m_w = 2.049\ \text{kg}, \quad I_a = m_w x_a^2 b^2 + 0.0517\ \text{kg} \cdot \text{m}^2$$
$$c_h = 27.43\ \text{kg/s}, \quad c_a = 0.036\ \text{kg} \cdot \text{m}^2/\text{s}, \quad c_{l\alpha} = 6.28, \quad c_{l\beta} = 3.358$$
$$c_{m\alpha} = (0.5 + 0.6847) \cdot c_{l\alpha}, \quad c_{m\beta} = -1.94$$
$$k_a = 2.82 \cdot \left(1 - 22.1\alpha + 1315.5\alpha^2 - 8580\alpha^3 + 17289.7\alpha^4\right)$$
$$k_h = 2844 \cdot \left(1 + 0.9h^2\right)$$

Matrix Definitions:

$$M^{\text{mass}} = \begin{bmatrix} m_t & m_w x_a b \\ m_w x_a b & I_a \end{bmatrix}, \quad C = \begin{bmatrix} c_h & 0 \\ 0 & c_a \end{bmatrix}, \quad K = \begin{bmatrix} k_h & 0 \\ 0 & k_a \end{bmatrix}$$

Aerodynamic Terms:

$$\alpha^{\text{eff}} = \alpha + \frac{\dot{h}}{U} + a_b b \cdot \frac{\dot{\alpha}}{U}, \quad L^{\text{aero}} = \rho U^2 b \left(c_{l\alpha}\alpha_{\text{eff}} + c_{l\beta}\beta\right), \quad M^{\text{aero}} = \rho U^2 b^2 \left(c_{m\alpha}\alpha_{\text{eff}} + c_{m\beta}\beta\right)$$

Second Derivatives:

$$\begin{bmatrix} \ddot{h} \\ \ddot{\alpha} \end{bmatrix} = (M^{\text{mass}})^{-1} \left(- \begin{bmatrix} L^{\text{aero}} \\ M^{\text{aero}} \end{bmatrix} - C \begin{bmatrix} \dot{h} \\ \dot{\alpha} \end{bmatrix} - K \begin{bmatrix} h \\ \alpha \end{bmatrix} \right)$$

Partial Derivatives of $\ddot{h}$

To find the partial derivatives of $\ddot{h}$ the first row from the continuous time equation 6.5 is extracted and the matrix definitions above are used. The first row is then explicitly written as equation 6.6 in the discrete time format.

$$\begin{bmatrix} \ddot{h}(t) \\ \ddot{\alpha}(t) \end{bmatrix} = M_{\text{mass}}^{-1} \left(- \begin{bmatrix} L_{\text{aero}} \\ M_{\text{aero}} \end{bmatrix} - C \begin{bmatrix} \dot{h}(t) \\ \dot{\alpha}(t) \end{bmatrix} - K \begin{bmatrix} h(t) \\ \alpha(t) \end{bmatrix} \right), \tag{6.5}$$

$$a_k^h = \left(\frac{I_a}{\det \boldsymbol{M}^{\text{mass}}} \right) (-\boldsymbol{L}_k^{\text{aero}} - c_h \boldsymbol{v}_k^h - k_h h_k) + \left(-\frac{m_w x_a b}{\det \boldsymbol{M}^{\text{mass}}} \right) (-\boldsymbol{M}_k^{\text{aero}} - c_a \boldsymbol{v}_k^\alpha - k_a \alpha_k) \tag{6.6}$$

The partial derivatives are computed in the following:

Derivative w.r.t. $x_1 = h_k$:

$$\frac{\partial a_k^h}{\partial h_k} = -\left(\frac{I_a}{\det \boldsymbol{M}^{\text{mass}}} \right) \left(\frac{\partial k_h}{\partial h_k} \cdot h_k + k_h \right), \quad \text{where} \quad \frac{\partial k_h}{\partial h_k} = 5119.2 \cdot h_k$$

Derivative w.r.t. $x_2 = \alpha_k$:

$$\frac{\partial a_k^h}{\partial \alpha_k} = -\left(\frac{m_w x_a b}{\det \boldsymbol{M}^{\text{mass}}} \right) \left(\rho U^2 b \cdot c_{l\alpha} + k_a \right)$$

Derivative w.r.t. $x_3 = \boldsymbol{v}_k^h$:

$$\frac{\partial a_k^h}{\partial \boldsymbol{v}_k^h} = -\left(\frac{I_a}{\det \boldsymbol{M}^{\text{mass}}} \right) \left(\rho U b \cdot c_{l\alpha} + c_h \right)$$

Derivative w.r.t. $x_4 = \boldsymbol{v}_k^\alpha$:

$$\frac{\partial a_k^h}{\partial \boldsymbol{v}_k^\alpha} = -\left(\frac{m_w x_a b}{\det \boldsymbol{M}^{\text{mass}}} \right) \left(\rho U b a_b b \cdot c_{l\alpha} + c_a \right)$$

Derivative w.r.t. $x_5 = \beta_k$:

$$\frac{\partial a_k^h}{\partial \beta_k} = -\left(\frac{I_a}{\det \boldsymbol{M}^{\text{mass}}} \right) \rho U^2 b \cdot c_{l\beta} - \left(\frac{m_w x_a b}{\det \boldsymbol{M}^{\text{mass}}} \right) \rho U^2 b^2 \cdot c_{m\beta}$$

Partial Derivatives of $\ddot{\alpha}$

To compute the partial derivatives of $\ddot{\alpha}$, the second row from the continuous Equation 6.5 is extracted and expressed explicitly in discrete time using the previously defined system matrices:

$$a_k^\alpha = \left(\frac{m_t}{\det(M^{\text{mass}})}\right)(-M_k^{\text{aero}} - c_a v_k^\alpha - k_a \alpha_k) + \left(-\frac{m_w x_a b}{\det(M^{\text{mass}})}\right)(-L_k^{\text{aero}} - c_h v_k^h - k_h h_k) \tag{6.7}$$

The partial derivatives are now computed as following:

Derivative w.r.t. $x_1 = h_k$:

$$\frac{\partial a_k^\alpha}{\partial h_k} = -\left(\frac{m_w x_a b}{\det \boldsymbol{M}^{\text{mass}}}\right)\left(\rho U^2 b \cdot c_{l\alpha} \cdot \frac{\partial \alpha^{\text{eff}}}{\partial h_k} + \frac{\partial k_h}{\partial h_k} \cdot h_k + k_h\right)$$

Derivative w.r.t. $x_2 = \alpha_k$:

$$\frac{\partial a_k^\alpha}{\partial \alpha_k} = -\left(\frac{m_t}{\det \boldsymbol{M}^{\text{mass}}}\right)\left(\rho U^2 b^2 \cdot c_{m\alpha} + k_a\right)$$

Derivative w.r.t. $x_3 = \boldsymbol{v}_k^h$:

$$\frac{\partial a_k^\alpha}{\partial \boldsymbol{v}_k^h} = -\left(\frac{m_w x_a b}{\det \boldsymbol{M}^{\text{mass}}}\right)(\rho U b \cdot c_{l\alpha} + c_h)$$

Derivative w.r.t. $x_4 = \boldsymbol{v}_k^\alpha$:

$$\frac{\partial a_k^\alpha}{\partial \boldsymbol{v}_k^\alpha} = -\left(\frac{m_t}{\det \boldsymbol{M}^{\text{mass}}}\right)\left(\rho U b^2 a_b \cdot c_{m\alpha} + c_a\right)$$

Derivative w.r.t. $x_5 = \beta_k$:

$$\frac{\partial a_k^\alpha}{\partial \beta_k} = -\left(\frac{m_t}{\det \boldsymbol{M}^{\text{mass}}}\right)\rho U^2 b^2 \cdot c_{m\beta}$$

Evaluation of $\boldsymbol{f}(\boldsymbol{x}_{k+1}, u_{k+1}, w_{k+1})$

The final term in the second-order expansion is the direction vector $\boldsymbol{f}(\boldsymbol{x}_{k+1}, u_{k+1}, w_{k+1})$, evaluated at the updated state after the first Euler step. It is computed using the same system model and follows the structure:

$$\boldsymbol{f}(\boldsymbol{x}_{k+1}, u_{k+1}, w_{k+1}) = \begin{bmatrix} \boldsymbol{v}_{k+1}^h \\ \boldsymbol{v}_{k+1}^\alpha \\ a_{k+1}^h \\ a_{k+1}^\alpha \\ k_\beta(\boldsymbol{u}_{k+1}^{\text{flap}} - \beta_{k+1}) \end{bmatrix} \tag{6.8}$$

The values of v_{k+1}^h and v_{k+1}^α are obtained from the first-order updates:

$$v_{k+1}^h = v_k^h + \Delta t \cdot a_k^h, \quad v_{k+1}^\alpha = v_k^\alpha + \Delta t \cdot a_k^\alpha \tag{6.9}$$

Similarly, $\beta_{k+1} = \beta_k + \Delta t \cdot k_\beta(\boldsymbol{u}_k^{\text{flap}} - \beta_k)$. These values are then used to recompute the aerodynamic terms and accelerations a_{k+1}^h and a_{k+1}^α using equations for aerodynamic and second derivatives defined above.

This ensures that the second-order correction term is fully consistent with the discretized physical model, and includes the delayed propagation of control inputs through β, gust, and aerodynamic couplings.

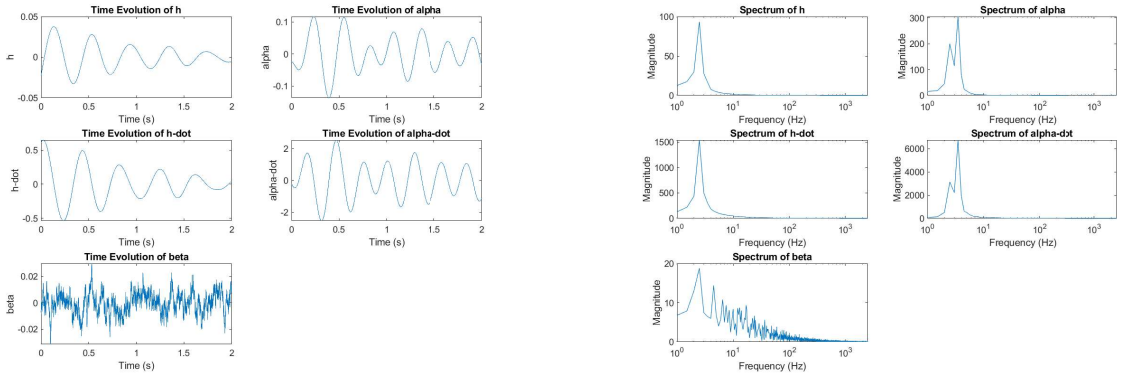
6.2. Dynamic Feasibility of LPV-MPC Assumptions

System Dynamics Bandwidth

To justify the use of an LPV model updated at every timestep, it is important to verify that the system's nonlinear dynamics evolve slowly enough for linearizations to remain accurate. This relies on frequency separation: the dominant modes of the system must lie well below the LPV update rate.

If the system's frequency content is below the Nyquist frequency defined by $f_s = 1/\Delta t$, the linearization at each step remains locally valid. In this setup, the LPV model is updated every 0.001 seconds (1000 Hz), under the assumption that the dynamics evolve slowly enough for this resolution to track their behavior effectively.

To test this, the full nonlinear model was simulated for 2 seconds at a rate of 5000 Hz using a small random input to the flap actuator. All five discrete-time states h_k, α_k, v_k^h, v_k^α, and β_k were recorded, and an FFT was applied to extract their frequency content.



(a) Time-domain evolution of all five states

(b) Frequency spectra of each state

Figure 6.1: Visualization of system behavior in time and frequency domains

As expected, h_k and $v_{h,k}$ exhibit low-frequency motion, while α_k and $v_{\alpha,k}$ show higher variability but remain bounded. The state β_k, directly driven by the input, exhibits broader frequency content.

The FFT confirms that the dominant dynamics are concentrated below 50 Hz, with negligible energy beyond 100 Hz. Even the fastest state, v_k^α, peaks around 40–50 Hz. Since the LPV update rate is 1000 Hz, the system is sampled well above its Nyquist limit, ensuring accurate tracking at each linearization point.

These findings confirm sufficient frequency separation: the system evolves slowly enough for an LPV update every 0.001 seconds to remain valid throughout the prediction horizon.

Actuator vs. Plant Bandwidth Separation

In addition to ensuring the system evolves slowly enough for LPV updates to be valid, it is also necessary to verify that the actuator dynamics are sufficiently fast relative to the plant. This ensures that control inputs can be applied effectively within each control interval. A standard guideline is that actuator bandwidth should exceed plant bandwidth by a factor of 2–5 to ensure responsiveness. In this system, the control flap is modeled as a first-order actuator:

$$\beta_{k+1} = \beta_k + \Delta t \cdot k_\beta (u_k^{\text{flap}} - \beta_k)$$

where $k_\beta = 60$ implies a time constant $\tau = 1/k_\beta$ and a bandwidth $f_\beta \approx 9.55\,\text{Hz}$.

To compare this with the plant, the LPV model is simulated at 5000 Hz under small random inputs and extracted the dominant frequency for each state via FFT. The simulation was repeated ten times, and each trial produced identical frequency characteristics, confirming the consistency of the response. Results are shown in Table 6.1.

State	h	α	$\dot{h}$	$\dot{\alpha}$	β
Dominant Frequency (Hz)	2.50	5.00	2.50	5.00	0.50

Table 6.1: Dominant frequency for each state obtained via Fourier analysis.

All plant states exhibit dominant frequencies well below 10 Hz. The fastest, $\dot{\alpha}(t)$, peaks at 5 Hz, while the actuator bandwidth is approximately 9.55 Hz — slightly below the typical factor-of-2 guideline but still within an acceptable range. The state $\beta(t)$, which reflects actuator dynamics, varies more slowly due to limited excitation. Overall, the actuator is sufficiently fast to track control inputs without introducing significant delay, supporting the use of LPV-based control at 1000 Hz.

Prediction Horizon and System Dynamics Resolution

An essential consideration in validating the LPV-based LPV-MPC framework is whether the chosen prediction horizon is long enough to capture the system's dominant dynamics. Even if the LPV model provides an accurate local approximation, a short horizon may fail to resolve slower modes of the system, reducing the effectiveness of the predictions and compromising constraint anticipation.

In standard MPC theory, the prediction horizon duration $T_H = N^{\mathsf{p}}\Delta t$ should ideally be large compared to the inverse of the dominant frequencies in the system:

$$T_H \gg \frac{1}{f_{\mathsf{dom}}}.$$

Given the characteristics of the current setup, the control update rate is $\Delta t = 0.001$ s (i.e., 1000 Hz), and the prediction horizon is set to $N_p = 20$, yielding a total lookahead time of $T_H = 0.02$ s. The dominant frequencies observed in the plant dynamics lie between 0.5 Hz and 5 Hz, which correspond to characteristic time scales ranging from 0.2 s to 2 s.

This means that the current prediction horizon is significantly shorter than the periods of the system's dominant modes. From a theoretical perspective, this implies that MPC operating with such a short horizon cannot fully resolve the system's dynamics, especially for slower-evolving states such as h, α, and β.

Consequently, a standalone online MPC implementation under this configuration would be unable to predict sufficiently far ahead to guarantee effective gust rejection. While increasing N_p could theoretically address this limitation by extending the prediction window, doing so would incur significant computational cost, rendering real-time deployment impractical.

However, this observation does not invalidate the standalone MPC control architecture. Instead, it reframes the role of MPC: the controller is used as a local trajectory generator that provides short-horizon predictions based on accurate LPV linearizations. These predictions are sufficient to ensure constraint satisfaction in the immediate future, even if they cannot anticipate the full system evolution.

Importantly, this analysis further supports the use of MPC as an offline training tool rather than an online controller. During training, MPC can be run with arbitrarily large prediction horizons (i.e., much larger N_p) since it is not constrained by real-time computational limits. This enables the generation of long-horizon, constraint-satisfying rollouts that are then used to train a RL policy. The RL agent, once trained, can be deployed online with minimal computation while inheriting the long-term awareness learned from the MPC trajectories.

In summary, although the short prediction horizon used in the MPC configuration is insufficient to fully capture the dominant dynamics of the system, this limitation is inherently resolved by the chosen architecture. While such a constraint would typically pose a significant challenge for online MPC, the current approach leverages MPC primarily as a training oracle for an offline RL policy. In this setting, longer prediction horizons can be used during training without concern for real-time feasibility, allowing the RL agent to inherit control behavior informed by long-term dynamics. As a result, the architecture enables effective long-term planning and constraint satisfaction without requiring computationally expensive online optimization, reinforcing the suitability of this RL-MPC approach.

6.3. Numerical Validation of LPV Accuracy

To validate the local accuracy of the LPV approximation, a numerical analysis was conducted comparing two LPV systems linearized at nearby operating points. The key objective is to verify that the linearization error remains sufficiently small over one control step, which is relevant in the current setup since the LPV system is recomputed at each timestep. Unlike traditional LPV methods where the linearization remains fixed over the prediction horizon, here the system matrices are updated continuously not only during training and deployment, but also internally within the MPC itself using predicted values of h and α over the prediction horizon. Therefore, it is sufficient to analyze whether the approximation holds locally for a single time step of size $\Delta t = 0.001$ s.

The experiment begins with the generation of a random discrete-time state $x_0 = [h_k, \alpha_k, v_k^h, v_k^\alpha, \beta_k]$, using the following bounded ranges: $h_k \in [-0.05, 0.05]$, $\alpha_k \in [-0.3, 0.3]$, $v_k^h, v_k^\alpha \in [-1, 1]$, and $\beta_k \in [-0.1, 0.1]$. A perturbed state x_1 is constructed by modifying h_k and α_k by $\delta h = 0.02$ and $\delta\alpha = 0.04$, corresponding to the maximum deviations induced by the rate limits $v_{h,max} = 0.07$ and $v_{\alpha,+max} = 1.19$ over one timestep $\Delta t = 0.001$ s. These perturbation values were chosen based on the 99.9$^{\text{th}}$ percentile of observed single-timestep variations across 1000 nonlinear simulations of the aircraft model, each with randomized initial states and random turbulences. As such, the analysis is conservatively designed to reflect worst-case deviations within the operational envelope. In practice, the actual deviations and hence the LPV approximation error are typically much smaller under nominal conditions.

The matrix difference is given by:

$$A_1 - A_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -0.0009 & 0.0096 & 0 & 0 & 0 \\ 0.0015 & -1.2944 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and the corresponding percentage difference is:

$$\frac{|A_1 - A_0|}{|A_0|} \cdot 100 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.0004 & 0.1110 & 0 & 0 & 0 \\ 0.0004 & 0.2780 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \ \%$$

To evaluate the effect of these matrix differences on system trajectories, both LPV systems were simulated from the same initial condition x_0 using the same random control inputs with $\Delta t = 0.001$. The final deviation in state trajectories was computed as:

$$x_1 - x_0 = \begin{bmatrix} 0 & \\ -3 \times 10^{-6} & rad \\ 3.3 \times 10^{-5} & m/s \\ -5.2 \times 10^{-4} & rad/s \\ 0 & \end{bmatrix} \quad , \quad \frac{|x_1 - x_0|}{\text{RMS}} \times 100\% = \begin{bmatrix} 0 \\ -0.003\% \\ 0.0099\% \\ -0.156\% \\ 0 \end{bmatrix}$$

These results confirm that the LPV approximation remains accurate over one timestep even under maximum expected variations in state values. Because the LPV model is updated at every timestep using current or predicted states, this local validity guarantees that the approximation error remains bounded across the full prediction horizon. This conclusion aligns with the theoretical justification previously established, where the remainder term was shown to be quadratically bounded in the state deviation. The numerical results support this theory, demonstrating that both the system matrix deviations and the resulting state trajectory differences remain small under realistic conditions. The LPV system thus provides a reliable representation of the nonlinear dynamics for control and safety analysis.

6.4. Justification of Second-Order Truncation in State Update

This section provides a detailed quantitative comparison of second- and third-order contributions to the update of each state, based on the Taylor expansion introduced in Equation (6.10). The constants used in this analysis are defined in Section 6.1.

$$\boldsymbol{x}_{k+2} = \boldsymbol{x}_k + 2\Delta t \cdot \boldsymbol{f}_k + \Delta t^2 \cdot \nabla_x \boldsymbol{f}_k \boldsymbol{f}_k + \frac{\Delta t^3}{6} \cdot D^2 \boldsymbol{f}_k[\boldsymbol{f}_k, \boldsymbol{f}_k] + \mathcal{O}(\Delta t^4) \tag{6.10}$$

The following state values are selected to reflect the 99.9$^{\text{th}}$ percentile of magnitudes observed across an extensive set of simulations. These values serve as conservative upper bounds, ensuring that the results remain valid even under extreme, yet realistic, operating conditions:

$$h = 0.01, \quad \alpha = 0.17, \quad v_{h,k} = 0.07, \quad v_{\alpha,k} = 1.19, \quad \beta = 0, \quad U = 12 \text{ m/s}, \quad \Delta t = 0.001$$

Quantitative Comparison of Second- and Third-Order Terms for h

Stiffness and Aerodynamic Terms:

$$k_h = 2844 \cdot (1 + 0.9 \cdot h^2) \approx 2844, \quad k_a \approx 21.29 \quad (\text{evaluated at } \alpha = 0.17)$$
$$\alpha^{\text{eff}} = \alpha + \frac{v_k^h}{U} + a_b b \cdot \frac{v_k^\alpha}{U} \approx 0.1734$$
$$L^{\text{aero}} = \rho U^2 b \left(c_{l\alpha} \alpha_{\text{eff}} + c_{l\beta} \beta\right) \approx 25.92, \quad M^{\text{aero}} = \rho U^2 b^2 \left(c_{m\alpha} \alpha^{\text{eff}} + c_{m\beta} \beta\right) \approx 4.147$$

System Matrix Evaluation:

$$\det M^{\text{mass}} \approx 0.6828, \quad \frac{I_a}{\det M^{\text{mass}}} \approx 0.0817, \quad \frac{m_w x_a b}{\det M^{\text{mass}}} \approx 0.1342, \quad \frac{m_t}{\det M^{\text{mass}}} \approx 18.14$$

Acceleration Term:

$$\begin{aligned} a_k^h &= \left(\frac{I_a}{\det M^{\text{mass}}}\right)(-L^{\text{aero}} - c_h v_k^h - k_h h_k) + \left(-\frac{m_w x_a b}{\det M^{\text{mass}}}\right)(-M^{\text{aero}} - c_a v_k^\alpha - k_a \alpha_k) \\ &= 0.0817 \cdot (-56.29) + (-0.1342) \cdot (-7.808) \\ &\approx -4.599 + 1.048 = -3.551 \quad \text{m/s}^2 \end{aligned}$$

Third-Order Estimate: The third derivative of h, denoted as the plunge jerk j_k^h, is approximated using a first-order finite difference between consecutive acceleration values. These values are computed from the flexible aircraft model at representative perturbed states. The discrete-time approximation is given by:

$$j_k^h \approx \frac{a_{k+1}^h - a_k^h}{\Delta t} = -16.27 \quad \text{m/s}^3$$

Fourth-Order Estimate: The fourth derivative of h, denoted as the plunge snap s_k^h, is approximated using a second-order central difference over a sequence of acceleration values. This quantifies the rate of change of the jerk, and is expressed as:

$$s_k^h \approx \frac{a_{k+2}^h - 2a_{k+1}^h + a_k^h}{\Delta t^2} = -0.0616 \quad \text{m/s}^4$$

Ratio of Third-Order to Second-Order Contribution (for h): The second-order update of the plunge state h_k is computed as:

$$\frac{\Delta t^2}{2} \cdot a_k^h = \frac{1 \times 10^{-6}}{2} \cdot (-3.551) = -1.776 \times 10^{-6} \quad \text{m}$$

The corresponding third-order term is:

$$\frac{\Delta t^3}{6} \cdot j_k^h = \frac{1 \times 10^{-9}}{6} \cdot (-16.27) = -2.712 \times 10^{-9} \quad \text{m}$$

The ratio of third- to second-order contributions is:

$$\frac{\text{Third-order}}{\text{Second-order}} = \frac{2.712 \times 10^{-9}}{1.776 \times 10^{-6}} \approx 0.00153 \quad (0.153\%)$$

Ratio of Third-Order to Second-Order Contribution (for v_k^h): When the velocity v_k^h is considered as the state variable, its second-order update involves the acceleration a_k^h, and the third-order update involves the jerk j_k^h. In this case, the fourth derivative s_k^h serves as the third-order correction term in the Taylor expansion:

$$\frac{\Delta t^2}{2} \cdot j_k^h = \frac{1 \times 10^{-6}}{2} \cdot (-16.27) = -8.137 \times 10^{-6} \quad \text{m/s}$$

$$\frac{\Delta t^3}{6} \cdot s_k^h = \frac{1 \times 10^{-9}}{6} \cdot (-0.0616) = -1.027 \times 10^{-11} \quad \text{m/s}$$

The ratio of the fourth-order term to the second-order update (for $v_{h,k}$) is:

$$\frac{\text{Third-order}}{\text{Second-order}} = \frac{1.027 \times 10^{-11}}{8.137 \times 10^{-6}} \approx 1.26 \times 10^{-6} \quad (0.000126\%)$$

These results confirm that the third-order contributions to both the plunge state h_k and the plunge velocity v_k^h are negligible under the considered conditions.

Quantitative Comparison of Second- and Third-Order Terms for α

Acceleration Terms:

$$\begin{aligned} a_k^\alpha &= \left(\frac{m_t}{\det M^{mass}}\right)(-M_k^{aero} - c_a v_k^\alpha - k_a \alpha_k) + \left(-\frac{m_w x_a b}{\det M^{mass}}\right)(-L_k^{aero} - c_h v_k^h - k_h h_k) \\ &= 18.14 \cdot (-7.808) + (-0.1342) \cdot (-56.29) \\ &\approx -141.6 + 7.554 = -134.0 \quad \text{rad/s}^2 \end{aligned}$$

Third-Order Estimate: The third derivative of α, denoted as the pitch jerk j_k^α, is approximated using a first-order finite difference between consecutive acceleration values. These values are computed from the flexible aircraft model at representative perturbed states. The discrete-time approximation is given by:

$$j_k^\alpha \approx \frac{a_{k+1}^\alpha - a_k^\alpha}{\Delta t} = -1170 \quad \text{rad/s}^3$$

Fourth-Order Estimate: The fourth derivative of α, denoted as the pitch snap s_k^α, is approximated using a second-order central difference over a sequence of acceleration values. This quantifies the rate of change of the jerk, and is expressed as:

$$s_k^\alpha \approx \frac{a_{k+2}^\alpha - 2a_{k+1}^\alpha + a_k^\alpha}{\Delta t^2} = -1520 \quad \text{rad/s}^4$$

Ratio of Third-Order to Second-Order Contribution (for α): The second-order update of the pitch angle α_k is computed as:

$$\frac{\Delta t^2}{2} \cdot a_k^\alpha = \frac{1 \times 10^{-6}}{2} \cdot (-134.39) = -6.70 \times 10^{-5} \quad \text{rad}$$

The corresponding third-order term is:

$$\frac{\Delta t^3}{6} \cdot j_k^\alpha = \frac{1 \times 10^{-9}}{6} \cdot (-1170) = -1.95 \times 10^{-7} \quad \text{rad}$$

The ratio of third- to second-order contributions is:

$$\frac{\text{Third-order}}{\text{Second-order}} = \frac{1.95 \times 10^{-7}}{6.70 \times 10^{-5}} \approx 0.00291 \quad (0.291\%)$$

Ratio of Third-Order to Second-Order Contribution (for v_k^α): When the velocity v_k^α is considered as the state variable, its second-order update involves the acceleration a_k^α, and the third-order update involves the jerk j_k^α. In this case, the fourth derivative s_k^α serves as the third-order correction term in the Taylor expansion:

$$\frac{\Delta t^2}{2} \cdot j_k^\alpha = \frac{1 \times 10^{-6}}{2} \cdot (-1170) = -5.850 \times 10^{-4} \quad \text{rad/s}$$

$$\frac{\Delta t^3}{6} \cdot s_k^\alpha = \frac{1 \times 10^{-9}}{6} \cdot (-1520) = -2.533 \times 10^{-7} \quad \text{rad/s}$$

The ratio of the third-order term (snap) to the second-order update (jerk) for v_k^α is:

$$\frac{\text{Third-order}}{\text{Second-order}} = \frac{-2.533 \times 10^{-7}}{-5.850 \times 10^{-4}} \approx 4.33 \times 10^{-4} \quad (0.0433\%)$$

These results confirm that the third-order contributions to both the pitch angle α_k and pitch velocity v_k^α are negligible under the considered conditions.

Quantitative Comparison of Second- and Third-Order Terms for β

The flap state β evolves according to a first-order differential equation. In discrete-time, its second and third derivatives can be approximated using known dynamics:

$$v_k^\beta = k_\beta(u_k^{\text{flap}} - \beta_k) \quad \Rightarrow \quad a_k^\beta = -k_\beta v_k^\beta, \quad j_k^\beta = k_\beta^2 v_k^\beta$$

The second- and third-order contributions in the Taylor expansion of the discrete-time state update are:

$$\frac{\Delta t^2}{2} \cdot a_k^\beta = -\frac{\Delta t^2}{2} \cdot k_\beta v_k^\beta, \quad \frac{\Delta t^3}{6} \cdot j_k^\beta = \frac{\Delta t^3}{6} \cdot k_\beta^2 v_k^\beta$$

Taking the ratio of the third- to second-order terms:

$$\left| \frac{\text{Third-order}}{\text{Second-order}} \right| = \left| \frac{\frac{\Delta t^3}{6} \cdot k_\beta^2 v_k^\beta}{\frac{\Delta t^2}{2} \cdot (-k_\beta v_k^\beta)} \right| = \frac{\Delta t}{3} \cdot k_\beta$$

Substituting $\Delta t = 0.001$ and $k_\beta = 60$, the ratio becomes:

$$\left| \frac{\text{Third-order}}{\text{Second-order}} \right| = \frac{0.001}{3} \cdot 60 = 0.02$$

This confirms that for $k_\beta = 60$, the third-order term is only 2% of the second-order term and can safely be neglected.

To determine the maximum k_β for which the third-order term remains within acceptable bounds, a threshold of 10% relative to the second-order term is used. This represents a practical engineering margin beyond which the higher-order contributions may no longer be negligible. Setting:

$$\frac{k_\beta}{3000} = 0.1 \quad \Rightarrow \quad k_\beta = 300$$

Therefore, as long as $k_\beta \leq 300$, the third-order contribution to the update of β remains sufficiently small.

6.5. Controller Configurations

6.5.1. MPC Controller parameters

Sampling and horizons The controller operates at a fixed sampling time of $T_s = 0.001\,\text{s}$. A prediction horizon of $N^{\text{P}} = 20$ and a control horizon of $N_c = 15$ are used. Command–actuator latency is accommodated by extending the internal model with $N^{\text{delay}} =$ delay dummy steps.

State estimation Separate third-order Kalman filters estimate $\{\alpha, \dot{\alpha}, \ddot{\alpha}\}$ and $\{U, \dot{U}, \ddot{U}\}$. Process and measurement covariances are $Q_\alpha = \text{diag}(1,\, 0.1,\, 0.01)$, $R_\alpha = 10$; $Q_U = \text{diag}(1,\, 0.1,\, 0.01)$, $R_U = 50$ in order to remove as much noise as possible.

Plant model A five-state aeroelastic wing model $(h,\ \alpha,\ \dot{h},\ \dot{\alpha},\ \beta)$ is linearized online at each time step and for each prediction step, starting at N^{delay} and ending at $N^{\text{delay}} + N^{\text{P}}$ steps. Both A and B matrices are re-computed from the predicted flight speed U and the AoA, ensuring consistency with the non-linear stiffness and aerodynamic terms.

Cost function The stage cost is $J = \sum_{i=0}^{N^{\text{P}}-1} \left(x_{k+i|k}^\top Q_{k+i} x_{k+i|k} + u_{k+i|k}^\top R u_{k+i|k}\right)$, with $R = I$. Three time-varying state weights are toggled:

- *Gust mode*: $Q_{\text{gust}} = \text{diag}(10,\, 1,\, 10,\, 10,\, 1)$, activated when $|\alpha| > 0.2\,\text{rad}$.
- *Away from equilibrium*: $Q_{\text{away}} = \text{diag}(20,\, 5,\, 1,\, 1,\, 1)$, when $\text{sign}(\alpha) = \text{sign}(\dot{\alpha})$.
- *Toward equilibrium*: $Q_{\text{tow}} = \text{diag}(5,\, 1,\, 10,\, 1,\, 1)$ otherwise.

Optional economic penalties for total movement (w_m) and smoothness (w_s) are set to one.

Constraints

$$-0.10 \le h \le 0.10\ \text{m}, \qquad -30^\circ \le \alpha \le 30^\circ,$$
$$-20^\circ \le u \le 20^\circ, \qquad |\Delta u| \le 0.0157\ \text{rad}.$$

Solver The quadratic programme is solved each step with `quadprog` (interior-point). If the solver fails (`exitflag`<0), the previous input u_{k-1} is re-issued.

Tuning rationale The prediction horizon is fixed at $N^{\text{P}} = 20$, giving a look-ahead of $T_H = 0.02\,\text{s}$, which is shorter than the dominant structural periods (0.2–2 s). Although this window cannot fully capture the states, it is the longest horizon that can be re-optimized at the 1 kHz update rate on the present processor. The setting therefore represents a trade-off between dynamic coverage and real-time feasibility; N^{P} may be increased whenever more on-board computational power becomes available.

6.5.2. RL Controller parameters

Sampling and delay compensation The RL policy executes at the same rate as the MPC, $T_s = 0.001\,\text{s}$. Constant–acceleration Kalman filters predict h and α (and their derivatives) forward by the commanded actuator delay so that the policy evaluates a delay-free estimate $x_k = [h,\ \alpha,\ \dot{h},\ \dot{\alpha},\ \beta]^{\text{T}}$.

Policy representation The controller is a k-nearest-neighbour table lookup with $k = 5$. For a query state x_k it retrieves the k closest stored samples, weights each action inversely to its distance, and returns their weighted average. If the nearest sample differs by more than 1 % of the full state range in *any* coordinate, a local refinement is triggered (see below) and the newly found $[x_k, u_k]$ pair is appended to an auxiliary table, enabling lifelong learning.

Local refinement A dense grid of 4 000 candidate elevator deflections in $[-0.35, 0.35]$ rad is rolled out for two time-steps; the action maximizing the shaped reward is cached and the highest reward is picked.

Reward design The instantaneous reward is

$$r = -w_\alpha \alpha^2 - w_{\dot\alpha}(\dot\alpha - \dot\alpha^\star)^2 - w_h h^2 - w_{\dot h}(\dot h - \dot h^\star)^2 - w_\beta \beta^2 - 0.01\, u^2,$$

with weights that adapt to the initial error ($w_h : w_\alpha = 3 : 1$ at large excursions, velocity weights quintuple when the state is already moving toward the origin ensuring minimized overshoot).

Offline training The initial table is populated by sampling the starting points drawn from truncated-normal distributions within $h \in [-0.05, 0.05]$ m, $\alpha \in [-0.5, 0.5]$ rad, $\dot h \in [-0.1, 0.1]$, $\dot\alpha \in [-1, 1]$, $\beta \in [-0.05, 0.05]$ rad. For each state the best action is selected from $[-0.35, 0.35]$ rad using the reward above and stored as a single $[x, u^*]$ row. These ranges *are used only for training* and do not constrain the policy during deployment, where actuator limits are enforced separately. The RL controller is trained on 1,000,000 initial states. For each state the control input is held for 2 time steps to capture the effect on all states and closely match the deployment scenarios.

Tuning rationale The k-NN regressor offers constant-time inference while retaining the ability to grow incrementally when the flight envelope ventures into new regions. Choosing the five closest stored states provides enough local samples to average out outliers and noise without diluting the action with points that lie too far from the current operating region. The 1 % coverage threshold minimizes unnecessary refinements, and the reward weighting prioritizes plunge suppression, which is more critical than AoA tracking for the flexible-wing configuration. The range of control inputs was selected to cover the entire actuation range with a sufficient precision.

6.5.3. RL-MPC Controller parameters

Sampling and delay compensation The RL-MPC policy shares the global control rate $T_s = 0.001$ s. Constant-acceleration Kalman filters predict h and α (plus their first derivatives) ahead by the measured actuator delay, yielding the delay-free vector $x_k = [h,\ \alpha,\ \dot h,\ \dot\alpha,\ \beta]^\mathsf{T}$. Central differences over the last three filtered samples add instantaneous accelerations,

$$\ddot h_k = \frac{\dot h_k - \dot h_{k-2}}{2T_s}, \qquad \ddot\alpha_k = \frac{\dot\alpha_k - \dot\alpha_{k-2}}{2T_s},$$

so the policy really operates on the augmented state $x_k^{\text{aug}} \in \mathbb{R}^7$.

Policy representation A k-nearest-neighbour regressor ($k = 5$) is queried with x_k^{aug}. If the nearest stored sample differs by more than 1 % of the full range in *any* of the seven co-ordinates, the point is judged *out-of-coverage* and triggers an on-line refinement (below). The deployed table starts with $\sim 1\,000\,000$ rows and grows slowly during flight.

Local refinement If coverage fails, a two-stage search refines the action:

1. **Candidate grid** 201 points uniformly spaced inside $[u_{\min}, u_{max}] \cap [u_{k-1} \pm \Delta u_{max}]$ with $\Delta u_{max} = 15.7 \times 10^{-3}$ rad (servo rate limit).

2. **Roll-out scoring** Each candidate is frozen for $N_h = 2$ steps and evaluated with the shaped reward

$$r = -\tfrac{1}{2} x^\top Q x - \tfrac{1}{2}(u^2) + \text{cross-terms},$$

where the cross-terms penalize excessive plunge-rate when the craft is already descending toward the trim. The top-K $(= 3)$ actions are stored; the best becomes the control.

Reward design Each position state is referenced to the origin, while the velocity states are driven toward the first-order decay targets $\dot h^\star = -h/0.5$ s and $\dot\alpha^\star = -\alpha/0.5$ s, which would settle the motion in roughly 0.5 s. The quadratic weights are biased toward velocity damping, $q_{\dot h} = q_{\dot\alpha} = 10$ versus $q_h = q_\alpha = 5$, and a modest control-effort penalty $r_u = 0.8$ keeps elevator deflections smooth. All coefficients can be re-tuned to trade off energy consumption against settling time.

Offline training with MPC–derived bounds For each of the $1e6$ training samples we first draw the five observable states $(h, \alpha, \dot{h}, \dot{\alpha}, \beta)$ from truncated-normal ranges $h \in [-0.05, 0.05]$ m, $\alpha \in [-0.5, 0.5]$ rad, $\dot{h} \in [-0.1, 0.1]$, $\dot{\alpha} \in [-1, 1]$, $\beta \in [-0.05, 0.05]$ rad. A least-squares regressor then estimates the gust speed that would make those states consistent with the nonlinear model; with that gust we analytically recover the accelerations $\ddot{h}, \ddot{\alpha}$ and form the 7-D augmented state x^{aug}. Next, a lightweight MPC controller is queried three times: with the inferred gust, with a $+25$% stronger gust, and with a -25% weaker gust. The smallest and largest of the three MPC outputs define a provably safe search interval $[u_{\text{min}}, u_{max}]$ that encloses the true-gust solution while respecting all constraints. Inside this interval we evaluate a uniform grid of 1 000 candidate actions; each action's reward is the weighted average of the scores obtained under the weaker, nominal, and stronger gusts, making the selection robust to estimation error. Only the three highest-reward actions are kept, so every stored table row consists of the 7-D state followed by its best-, second-, and third-best control inputs $[x^{\text{aug}}, u_1, u_2, u_3]$.

Tuning rationale * Five neighbours give enough resolution for local accuracy while keeping the lookup $O(k)$ and real-time safe. * Accelerations enrich the feature vector just enough to discriminate between "moving toward" and "moving away" cases without exploding table size. * MPC bounds embed hard constraints, letting the RL layer focus on performance rather than safety.

Tuning rationale

- **k-NN design.** Five neighbours strike a balance between local accuracy and $O(k)$ lookup time; inverse-distance weighting averages out outliers without diluting the action with points that lie far from the current operating region. The 1 % coverage threshold prevents unnecessary refinements while still allowing the table to grow incrementally whenever the flight envelope explores new territory.
- **Feature choice.** Adding the accelerations $\ddot{h}$ and $\ddot{\alpha}$ gives just enough information to distinguish whether the aircraft is moving *toward* or *away* from trim, without exploding table size.
- **MPC–based safety envelope.** For every training and on-line query an MPC routine evaluates the same state under nominal, $+25$%, and -25% gust levels; the minimum and maximum of the three outputs define a hard envelope $[u_{\text{min}}, u_{max}]$. This embeds all state and rate constraints, so the RL layer can focus purely on performance.
- **Control-grid resolution.** For each query we draw 201 candidate actions *inside* the MPC-derived envelope $[u_{\text{min}}, u_{max}]$ rather than over the entire $[-0.35, 0.35]$ rad span. Because the envelope length is typically ≤ 0.1 rad, the resulting average spacing is about 0.0002 rad—still more than fifteen times finer than the physical rate limit $\Delta u_{max} = 15.7 \times 10^{-3}$ rad—ensuring the optimum is not missed while keeping the evaluation cost low.
- **Reward priorities.** Weights are biased toward plunge suppression $(q_{\dot{h}} = q_{\dot{\alpha}} = 10,\ q_h = q_\alpha = 5)$, reflecting the tighter mm-level tolerance on h compared with α; a modest control-effort term $r_u = 0.8$ smooths the elevator motion.
- **Long-horizon MPC in training.** Offline, the MPC horizon is extended far beyond the real-time limit, allowing it to capture the full 0.2–2 s structural dynamics. The resulting long-range predictions yield tighter, yet still constraint-satisfying, action envelopes that the RL policy inherits without incurring any run-time cost.

7

Conclusions & Recommendations

7.1. Conclusions

This thesis investigated the control of flexible, aeroelastic aircraft structures under gust disturbances, where conventional control methods such as LQR, LQG or $\mathcal{H}_\infty$ are often inadequate due to their limited ability to handle nonlinear dynamics, structural coupling, and uncertainty. Linear Parameter Varying Model Predictive Control (LPV-MPC) has proven effective for flexible aircraft control, offering structured constraint handling, multivariable coordination, and robustness across flight regimes. However, its performance can degrade under turbulence due to model mismatch and actuator delay. It also faces practical limitations from high computational demands. In contrast, Reinforcement Learning (RL) provides a lightweight, data-driven approach capable of adapting to nonlinear dynamics and uncertainty, but typically lacks guarantees for constraint satisfaction. These contrasting properties reveal a natural synergy: by combining the structure and safety of LPV-MPC with the adaptability of RL, this thesis proposes a hybrid framework that balances robustness and flexibility for gust-affected aeroelastic systems.

The main contribution of this thesis is a novel controller architecture that combines LPV-MPC with RL for the control of aeroelastic aircraft. Two long-horizon LPV-MPC controllers compute safe control bounds under best and worst case disturbance assumptions, while a RL agent selects actions within these bounds based on gust predictions and current states. To ensure safe operation in previously unseen conditions, a weighted least-squares interpolation method is introduced, with constraint satisfaction guaranteed through a Lipschitz-based criterion. This structure ensures robust, adaptive, and constraint aware control of nonlinear aeroelastic aircraft under turbulent conditions. Although developed for aeroelastic aircraft, the proposed controller architecture can be extended to other nonlinear systems with constraints, including applications in robotics and the automotive industry, provided they exhibit similar structure and smoothness properties.

7.2. Research Questions

The aim of this thesis was to investigate the use of LPV-MPC and RL for the control of flexible aircraft, with a particular focus on the potential benefits of integrating the two approaches. This investigation was guided by the following research questions:

Research Question 1

How do standalone MPC and RL controllers perform for aeroelastic aircraft under gust disturbances, and what are their respective strengths and limitations?

MPC, particularly in its LPV formulation, has been widely applied to flexible aircraft due to its ability to manage constraints and coordinate multivariable dynamics. However, existing research often neglects fast transients caused by turbulence, leading to performance degradation in realistic gust scenarios. In such conditions, LPV-MPC suffers from model mismatch because it cannot anticipate rapid disturbances and relies heavily on accurate parameter scheduling. RL, by contrast, has been explored less frequently in this context but shows promise as a data-driven method that can adapt to nonlinearities and unmodeled dynamics. Still, most RL approaches lack formal guarantees for constraint satisfaction and stability, making them difficult to apply directly in safety-critical systems.

The simulation results confirmed the trends observed in the literature review. Both controllers demonstrated effective flutter suppression and gust load alleviation. RL was able to handle slightly stronger gusts in some scenarios, showing more robustness during high intensity disturbances, while LPV-MPC provided smoother recovery and more stable performance under moderate conditions. LPV-MPC outperformed RL during times when the disturbance had subsided and the plant model matched the controller's internal model. In these conditions, it achieved stable trajectories and smooth recovery, as expected. However, during active gust disturbances, its performance degraded due to model mismatch caused by unmodeled turbulence. Despite this, LPV-MPC required the least actuator movement, particularly in the recovery phase following the gusts. It was also the most computationally expensive method, with longer runtimes than RL. In contrast, RL showed strong suppression of large deviations during the gust but had less consistent convergence afterward and could not match the precision of LPV-MPC during recovery. It frequently issued control inputs beyond the actuator limits, which were externally clamped in simulation, indicating that the learned policy did not account for the system constraints. These results reinforce the known trade-offs: LPV-MPC provides structure, smoothness, and constraint handling but suffers under turbulence, while RL is responsive to disturbances but lacks constraint awareness and consistent post-disturbance recovery.

Research Question 2

What are the most effective strategies for combining MPC and RL for aeroelastic control, considering safety and operational constraints, fast aeroelastic responses, and the computational demands of online implementation?

Several integration strategies have been proposed in the literature to combine MPC and RL for systems requiring both adaptability and constraint satisfaction. One group embeds MPC as a structured policy inside the RL loop, replacing neural networks with optimization-based decisions that preserve safety and structure. However, this increases computational complexity and depends on accurate models, limiting scalability. Another group uses RL to tune MPC parameters such as weights or constraints online, improving adaptability, but often at the cost of predictability and guaranteed feasibility. A third category refines MPC-generated actions using RL to compensate for disturbances or modeling errors, though this assumes MPC's output is close enough to optimal for refinement to be effective. Supervisor architectures reverse this flow, using MPC to filter unsafe RL actions—but may become overly conservative and reduce performance. Overall, while these methods address some aspects of the control challenge, they often face trade-offs between performance, safety, and real-time applicability.

The proposed hybrid controller addresses the competing demands of safety, responsiveness, and computational feasibility through a structured design. For safety and constraint satisfaction, the RL agent is trained exclusively on control actions generated by long-horizon LPV-MPC, ensuring that all observed behaviors lie within safe operational bounds. Additionally, during deployment, a Lipschitz-based runtime check guarantees constraint satisfaction even in previously unseen states. For fast aeroelastic response, LPV-MPC is used during training to provide precise control under known disturbances, enabling the RL agent to learn high-quality responses across a wide range of gust scenarios. By leveraging state history information, the agent learns to anticipate disturbances and learn effective compensatory actions. Finally, to meet computational demands, the trained RL policy is deployed using a lightweight Q-learning-based implementation that encodes the complex dynamics learned during training into a fast, reactive policy that remains computationally efficient during deployment.

Simulation results confirm the effectiveness of this approach. In terms of safety and constraint handling, the hybrid controller satisfied all constraints in every test case, avoiding the violations observed under standalone RL. It also showed improved actuator usage, more conservative than RL but slightly more active than LPV-MPC, striking a balance between safety and responsiveness. For aeroelastic response, the hybrid controller demonstrated the fastest recovery times and the lowest excursion counts and overshoot values, consistently outperforming both standalone methods in plunge suppression, angle-of-attack stabilization, and frequency-domain error attenuation. Regarding computational demands, the final RL-based policy allowed for fast online execution with efficiency comparable to standalone RL and lower runtime overhead than LPV-MPC. These findings demonstrate that the proposed hybrid controller effectively balances robustness, responsiveness, and real-time feasibility in demanding aeroelastic environments.

Research Question 3

How can the proposed hybrid MPC-RL approach be theoretically justified for safety and empirically validated through computer-based simulations for both safety and performance, and does it provide a measurable improvement over standalone controllers in aeroelastic control scenarios?

Theoretical verification of the hybrid controller was established through formal safety analysis. A second-order Taylor expansion was used to approximate the nonlinear system dynamics, and it was shown that the higher-order terms were negligible compared to the dominant terms. This was verified even for the largest state and input excursions encountered during testing, ensuring the approximation holds across all relevant conditions. Using this approximation, it was proven that if the minimum and maximum control inputs generated by the LPV-MPC controller satisfy all state and input constraints, then any control action selected within this range will also respect those constraints. Since the reinforcement learning agent is trained only using actions within this safe set, the resulting policy inherits this safety guarantee. To support safe deployment, a second result was derived using Lipschitz continuity. It showed that interpolating between nearby safe training actions remains safe, provided the current state lies within a neighborhood of previously encountered states. With sufficiently dense training coverage, it can be guaranteed that the entire safe region is covered by such neighborhoods. Together, these results verify that the hybrid controller maintains constraint satisfaction both during learning and at runtime.

Empirical validation was conducted through a Monte Carlo simulation of 1000 randomized gust scenarios, each including five seconds of turbulence followed by five seconds of recovery. The hybrid controller maintained strict constraint satisfaction across all cases, confirming that the theoretical safety guarantees hold in practice, even under severe disturbances. In terms of performance, it consistently outperformed both standalone methods by achieving stronger suppression of plunge and angle-of-attack deviations during gusts, along with smoother and faster recovery afterward. While the standalone LPV-MPC achieved slightly lower control effort and faster convergence in ideal conditions, the hybrid controller provided the most robust overall performance, effectively balancing adaptability, safety, and constraint handling in challenging aeroelastic environment.

7.3. Recommendations

Based on the findings and limitations of this study, several directions are proposed to strengthen and extend the current work. These recommendations aim to improve the theoretical guarantees, expand the scope of applicability, and move the approach closer to real-world deployment.

This study demonstrated that higher-order terms in the system dynamics can be neglected, as their influence is negligible compared to the dominant second-order effects. However, future work could aim to derive explicit bounds on these higher-order contributions to formally ensure that the control input remains within the safety envelope under broader dynamic conditions. This would provide an additional layer of robustness and help extend the theoretical guarantees to more complex systems involving greater dimensionality and structural complexity than the current model.

Another recommendation concerns the convergence properties of the controller. While this study demonstrated that the proposed hybrid controller can maintain constraint satisfaction under a wide range of disturbances, it did not provide formal guarantees on convergence to the equilibrium. Future work could investigate conditions under which convergence can be ensured, providing a more complete foundation for real-world deployment.

The current framework operates entirely in discrete time, with fixed-interval control updates and system evolution. Extending the approach to continuous-time systems would be a valuable direction for future research. This would involve adapting the training and deployment framework to handle continuous dynamics, potentially improving accuracy and applicability to real-world aeroelastic systems with faster dynamics.

This study was conducted on a simplified two-dimensional simple model of a flexible wing, entirely within a simulation environment. Future work should extend this approach to larger, more realistic aircraft models and consider experimental validation beyond simulation to assess the real-world performance and practical feasibility fo the proposed control framework.