Genetic Programming for Intelligent Control: Real-Time Guidance for Access to Space

Francesco Marchetti

Submitted in fulfilment of the requirements

for the degree of Doctor of Philosophy

Intelligent Computational Engineering Laboratory (ICE-Lab)
Mechanical and Aerospace Engineering Department
University of Strathclyde, Glasgow

April 22, 2024

This thesis is the result of the author's original research. It has been composed by the author and has not been previously submitted for examination which has led to the award of a degree.

Acknowledgements

I am deeply grateful to my supervisor Dr. Edmondo Minisci for his continuous guidance, support and patience throughout my research. I learned a lot thanks to his knowledge and advice.

I am also thankful to Dr. Annalisa Riccardi for the invaluable suggestions and discussions, and for involving me in extremely interesting and formative research opportunities.

I would like to acknowledge the financial support received by the University of Strathclyde and the SPECIES Society to conduct research and present my works at international conferences.

A huge thanks go to all the friends and colleagues I have met during the PhD. I am grateful to those with whom I shared my life in Glasgow and Lisbon, and who made it an unforgettable experience. I also want to thank all the colleagues who have inspired me with their example and pushed me to do my best.

I am extremely grateful to my long-time friends who supported me in the past years. They would deserve to be cited one by one, but my gratitude cannot be summarized in a few lines. Therefore, I would just say that their support was invaluable, and although I may not show it, I deeply care about them. There were hard times during this experience, and their friendship helped me greatly.

Most of all, a special thanks goes to my family. Without their support and encouragement, I would not have achieved anything. After all, a tree can grow as high as its roots are deep.

Lastly, I want to thank Alessandra, whose smile helped get through the last and most difficult part of this journey.

Abstract

Reusable Launch Vehicles (RLVs) are becoming a relevant topic within the space industry. These vehicles operate in a broad range of flight envelopes necessitating greater autonomy to compensate for uncertainties or disturbances in real time. Therefore, intelligent Guidance and Control (G&C) architectures are required. The research presented in this thesis aims at investigating the application of Genetic Programming (GP) in an Intelligent Control (IC) setting to perform the real-time guidance of a RLV. The thesis begins with a literature review of the state-of-the-art of G&C and IC methods and applications. Following this, a novel taxonomy of IC was developed, aimed at classifying the applications' levels of intelligence. The applicability of GP in an IC setting is then investigated, both as a standalone approach and when hybridized with a Neural Network (NN). The standalone GP is applied to perform the real-time guidance of a Goddard rocket. This application, the first of its kind for an aerospace vehicle, showcases the ability of GP to produce online guidance commands to successfully track a reference trajectory when external disturbances are applied. Simultaneously, a novel hybrid IC scheme named Genetically Adapted Neural Network-based Intelligent Controller (GANNIC) is introduced. GANNIC is composed of a NN, used as a nonlinear agent, whose weights are updated online by a set of differential equations found offline using GP. Applied to real-time reentry guidance of an RLV, GANNIC proves effective in generating online guidance commands to compensate for uncertainties, and its robustness is assessed through a statistical study. Lastly, as a byproduct of the research, the Inclusive Genetic Programming (IGP), a novel GP heuristic, is introduced. The IGP advances the state-of-the-art of GP algorithms by addressing the population's diversity issue while demonstrating its efficacy in G&C applications.

Contents

List of Figures

List of Figures

List of Figures

List of Tables

List of Algorithms

Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
ANFIS	Adaptive Network-based Fuzzy Inference System
BFGS	Broyden–Fletcher–Goldfarb–Shanno
BVP	Boundary Value Problem
CI	Computational Intelligence
CSG	Cubic Spline Generalization
DNN	Deep Neural Network
EA	Evolutionary Algorithm
EC	Evolutionary Computing
ES	Evolutionary Strategy
ESA	European Space Agency
FAC	Final Approach Corridor
FDIR	Fault, Detection, Isolation and Recovery
FESTIP	Future European Space Transportation Investigation Programme
FL	Fuzzy Logic

List of Algorithms

FTC Fault Tolerant Control

PID Proportional-Integral-Derivative

GA Genetic Algorithm

GANNIC Genetically Adapted Neural Network-based Intelligent Controller

G&C Guidance and Control

GP Genetic Programming

HOSMC Higher-Order Sliding Mode Control

IAE Integral of Absolute Error

IC Intelligent Control

IGP Inclusive Genetic Programming

IncC Inclusive Crossover

IncT Inclusive Tournament

ISMC Integral Sliding Mode Control

KBS Knowledge Based System

LQR Linear Quadratic Regulator

LSTM Long Short-Term Memory

MGGP Multi-Gene Genetic Programming

MIMO Multiple-Input Multiple-Output

ML Machine Learning

MOPED Multi-Objective Parzen-based Estimation of Distribution

MP-AIDEA Multi-Population Adaptive Inflationary Differential Evolution Algorithm

List of Algorithms

MPC Model Predictive Control

MRAC Model Reference Adaptive Control

MSE Mean Squared Error

NAGEP Neural-Adaptation of the Genetic Programming Control Law

NLP Nonlinear Programming

NM Nelder-Mead

NN Neural Network

NSFTSMC Non-singular Fast Terminal Sliding Mode Control

ODE Ordinary Differential Equation

PCHIP Piecewise Cubic Hermite Interpolating Polynomial

RBF Radial Basis Function

RL Reinforcement Learning

RLV Reusable Launch Vehicle

RNN Recurrent Neural Network

RMSE Root Mean Squared Error

SAV Space Access Vehicle

SGP Standard Genetic Programming

SMC Sliding Mode Control

SR Symbolic Regression

SVM Support Vector Machine

UAV Unmanned Aerial Vehicle

List of Algorithms

UP Uncertainty Profile

US Uncertainty Scenario

USSA1962 U.S. Standard Atmosphere Model 1962

VTVL Vertical Take-Off and Vertical Landing

XAI Expainable Artificial Intelligence

Nomenclature

α	Angle of attack
χ	Heading angle
$\dot{Q}$	Heat rate
γ	Flight path angle
λ	Latitude
ω	Earth's angular speed
ϕ	Neural Network activation function
σ	Bank angle
θ	Longitude
a_x	Longitudinal acceleration
a_z	Normal acceleration
c_d	Drag coefficient
c_l	Lift coefficient
D	Drag
g_0	Gravitational acceleration at sea level
h	Altitude

Nomenclature

L	Lift
l_b	Lower bound parameter used to evaluate the uncertainty bounding function $\epsilon(h)$
l_{b_S}	Lower bound parameter used to evaluate the scaler bounding function $\epsilon_S(t)$
M	Mach number
m	Vehicle empty mass
N_c	Number of constrained quantities
N_h	Number of neurons in Neural Network (NN)'s hidden layer
N_i	Number of NN's input
N_o	Number of NN's output
N_p	Number of points in the propagated trajectory
N_s	Number of state variables
N_U	Number of uncertainty scenarios used for training
N_u	Number of control variables
N_ν	Total number of NN's weights
q	Dynamic pressure
R	Earth's radius
S	Wing surface
T	Thrust
U	Uncertainty applied to the system or uncertainty scenario.
u_b	Upper bound parameter used to evaluate the uncertainty bounding function $\epsilon(h)$
u_{b_S}	Upper bound parameter used to evaluate the scaler bounding function $\epsilon_S(t)$
V	Speed

Chapter 1

Introduction

Over the preceding two decades, Artificial Intelligence (AI) has emerged as an important technological force in industrial applications, representing a dynamic and expanding field of research. The global output for AI research has witnessed a surge of more than 600% from 2000 to 2019.[1] The applications of AI can be found across diverse domains, ranging from medicine to social networks, showcasing its inherent versatility.[2]

Considerable research has been dedicated to the use of AI in space applications. Russo et al. [6] conducted a comprehensive survey on the applications of AI in earth observation, space exploration, and mission design and planning. Additionally, Meß et al. [7] focused their survey on AI applications in autonomous planning and scheduling, self-awareness, Fault, Detection, Isolation and Recovery (FDIR), on-board data analysis, as well as on-board operations and processing of earth-observation data. Another stream of research concentrates on the application of AI in Guidance and Control (G&C) systems. Izzo et al. [8] analyzed AI trends in spacecraft dynamics and control, emphasizing the potential for increased autonomy and enhanced system performance through greater integration of AI in the space domain. They also anticipate an increase in research related to the validation and explainability of AI systems.

Focusing on the application of AI to G&C systems, Intelligent Control (IC) represents a family of G&C schemes built using AI techniques and capable of adapting

[1] https://www.nature.com/articles/d41586-020-03409-8
[2] https://cloud.google.com/discover/ai-applications

or learning online. Intelligent systems exhibit an enhanced capability to handle uncertainties compared to traditional G&C approaches. They can use online-acquired data and can be applied to nonlinear systems without the need for linearization. Although the term IC was initially introduced in 1971 [9], this G&C paradigms have gained increasing popularity in the past two decades due to rapid advancements in hardware technology and AI research.

Despite notable examples of IC applications in the space industry found in the literature, such as NASA's Earth Observing One spacecraft [10], its application is still constrained by the demanding robustness and safety requirements typical of aerospace applications. As summarized by Cohen [11], this constraint primarily arises from the black-box nature of AI techniques, leading to challenges in understanding the decision-making processes of these algorithms. Furthermore, there is a lack of a well-defined framework for systematically analyzing the stability and robustness of IC approaches. As discussed in the subsequent chapter, existing approaches aim to address both of these challenges, but further research is necessary to make them applicable and widely accepted in the aerospace industry.

Among AI techniques, Genetic Programming (GP), an Evolutionary Algorithm (EA), exhibits potential for application in G&C systems. Notably, GP proved capable of generating control schemes that compete with human-designed ones, as reported by Koza et al. [12]. In GP individuals are encoded as symbolic mathematical models providing the potential to obtain interpretable solutions. This characteristic makes GP particularly valuable for understanding and analyzing the evolved control schemes. Additionally, Lyapunov stability analysis can be applied to a GP-based controller, as shown by Ali et al. [13].

Despite the potential of GP, the literature offers limited instances of its application in an IC context, as observed in two reports on IC applications produced within the scope of this thesis [14, 15]. Notably one of the reasons is its computational costs, which limits its applicability in real time. To overcome this, the hybridization with another AI approach, such as a Neural Network (NN), can be beneficial and it is investigated in this thesis.

Concerning aerospace applications, in recent times the space industry has redirected its focus towards Reusable Launch Vehicles (RLVs), a market projected to experience substantial growth from $1.94 billion in 2023 to $5.41 billion by 2030.[3] This growth in interest stems primarily from economic and environmental considerations, where the complete or partial reusability of launchers is anticipated to result in cost reductions of up to 65%, alongside a reduction in overall environmental impact.[4] Within the broader category of RLVs, the spaceplane configuration has gained significant momentum, prompting numerous new industrial developments. [5,6]

However, RLVs encounter several technical challenges. Wang et al. [16] emphasize, particularly in the context of Vertical Take-Off and Vertical Landing (VTVL) vehicles, the necessity for research in trajectory design and optimization technology, high-precision guidance and control technology for return manoeuvres, and landing support technology. The authors conclude that AI can play a pivotal role in enhancing autonomy, enabling faster responses to emergencies and off-nominal conditions without relying solely on commands from ground personnel.

Regarding RLVs configured as spaceplanes or hypersonic vehicles, Castaldi et al. [17] assert that maintaining controllability across a broad range of flight envelopes, adhering to constrained paths due to structural limitations, and managing uncertainties and external disturbances arising from extreme atmospheric conditions pose significant challenges. These challenges could be overcome by applying AI to the G&C scheme of such vehicles, to enhance autonomy and fortify robustness against uncertainties and disturbances.

1.1 Research Objectives

The main aim of the work has been the investigation of approaches to enhance the autonomy and robustness of RLVs against uncertainties and disturbances through online

[3] https://www.fortunebusinessinsights.com/reusable-launch-vehicle-market-106803

[4] https://www.kdcresource.com/insights-events/the-rise-of-reusable-rockets-transforming-the-economics-of-space-travel/

[5] https://www.space.com/polaris-spaceplanes-mira-light-flight-test-campaign-complete

[6] https://www.dawnaerospace.com/spacelaunch

adaptation. In particular, the research activity has focused on the application of GP for IC, both as a standalone method and as part of a hybrid approach, in combination with a NN.

In pursuit of this aim, the following specific objectives are pursued:

1. Investigate the current state-of-the-art of IC approaches, both within and outside the aerospace industry, and assess the level of intelligence exhibited by existing applications. This research aims to gain insights into the utilization of GP and NNs in the domain of IC.

2. Advance the state-of-the-art of GP algorithms applied to IC.

3. Develop a novel IC approach by hybridizing GP and NN algorithms. Subsequently, apply the developed IC scheme to perform real-time guidance for a RLV.

The central focus of this thesis revolves around the development of novel GP-based algorithms specifically designed for the IC of RLV. In doing so, aspects concerning the implementation, computational feasibility and overall performance were investigated. Improving the interpretability of the G&C models produced by the designed algorithms, and the development of a framework to assess their stability was initially considered but was deemed out of the scope of this first investigation.

1.2 Contributions

Some of the work presented in this thesis has been previously published in journal articles or conference proceedings and presented at international conferences and workshops. In the following, a list of the publications relevant to this thesis is listed.

1.2.1 Journal publications

1. Wilson, C.; **Marchetti, F.**; Di Carlo, M.; Riccardi, A.; Minisci, E. Classifying Intelligence in Machines: A Taxonomy of Intelligent Control. Robotics 2020, 9, 64. `https://doi.org/10.3390/robotics9030064` [1]

2. **Marchetti, F.**, Minisci, E., Riccardi, A.: Single-Stage to Orbit Ascent Trajectory Optimisation with Reliable Evolutionary Initial Guess. Optimization and Engineering (2021) [18]

3. **Marchetti, F.**; Minisci, E. Genetic Programming Guidance Control System for a Reentry Vehicle under Uncertainties. Mathematics 2021, 9, 1868. `https://doi.org/10.3390/math9161868` [4]

4. **Marchetti, F.**; Minisci, E. Genetically Adapted Neural Network-Based Intelligent Controller for Reentry Vehicle Guidance Control. [SUBMITTED TO] Soft Computing [2]

1.2.2 Peer-reviewed conferences publications

1. Wilson C., **Marchetti F.**, Di Carlo M., Riccardi A. and Minisci E., "Intelligent Control: A Taxonomy," 2019 8th International Conference on Systems and Control (ICSC), 2019, pp. 333-339, doi: 10.1109/ICSC47195.2019.8950603. [19]

2. **Marchetti F.**, Minisci E. and Riccardi A., "Towards Intelligent Control via Genetic Programming," 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1-8, doi: 10.1109/IJCNN48605.2020.9207694. [3]

3. **Marchetti F.**, Minisci E. (2020) A Hybrid Neural Network-Genetic Programming Intelligent Control Approach. In: Filipič B., Minisci E., Vasile M. (eds) Bioinspired Optimization Methods and Their Applications. BIOMA 2020. Lecture Notes in Computer Science, vol 12438. Springer, Cham. `https://doi.org/10.1007/978-3-030-63710-1_19` [20]

4. **Marchetti F.**, Minisci E. (2021) Inclusive Genetic Programming. In: Hu T., Lourenço N., Medvet E. (eds) Genetic Programming. EuroGP 2021. Lecture Notes in Computer Science, vol 12691. Springer, Cham. `https://doi.org/10.1007/978-3-030-72812-0_4` [21]

1.2.3 Workshop presentations

1. **Marchetti F.**, Minisci E., Riccardi A. . Trajectory Optimization of a Reusable Launch Vehicle. 17th Workshop on Advances in Continuous Optimization (EUROPT 2019). Glasgow, 2019. `https://pure.strath.ac.uk/ws/portalfiles/portal/103324763/Marchetti_etal_Europt_2019_Trajectory_optimization_of_a_reusable_launch_vehicle.pdf` [22]

2. **Marchetti F.**, Minisci E., Riccardi A. . Towards Intelligent Control via Genetic Programming. 5th workshop on Optimisation in Space Engineering (OSE). Ljubljana, 2019. `https://pure.strath.ac.uk/ws/portalfiles/portal/103324511/Marchetti_etal_OSE_2019_Towards_intelligent_control_via_genetic_programming.pdf` [23]

1.2.4 Technical reports

1. Riccardi A., Minisci E., Di Carlo M., Wilson C., **Marchetti F.**. Assessment of Intelligent Control Techniques for Space Applications. ESA Contract Nr. 4000124916/18/NL/CRS/hh. 2018. [14]

2. Riccardi A., Minisci E., Di Carlo M., Wilson C., **Marchetti F.**. Assessment of Intelligent Control Techniques for Space Applications - Gap Analysis. ESA Contract Nr. 4000124916/18/NL/CRS/hh. 2018. [15]

Other works, not relevant to this thesis, were produced during the PhD program: the journal paper [24], the peer reviewed conference papers [25, 26] and the workshop presentation [27].

The code developed for [3, 20, 21, 4] and for the test case in section 5.3 is publicly available at `https://github.com/strath-ace/smart-ml`.

1.3 Thesis structure

The structural arrangement of the thesis is illustrated in Figure 1.1, delineating the various components. Novel contributions are highlighted through the use of bold for-

matting.

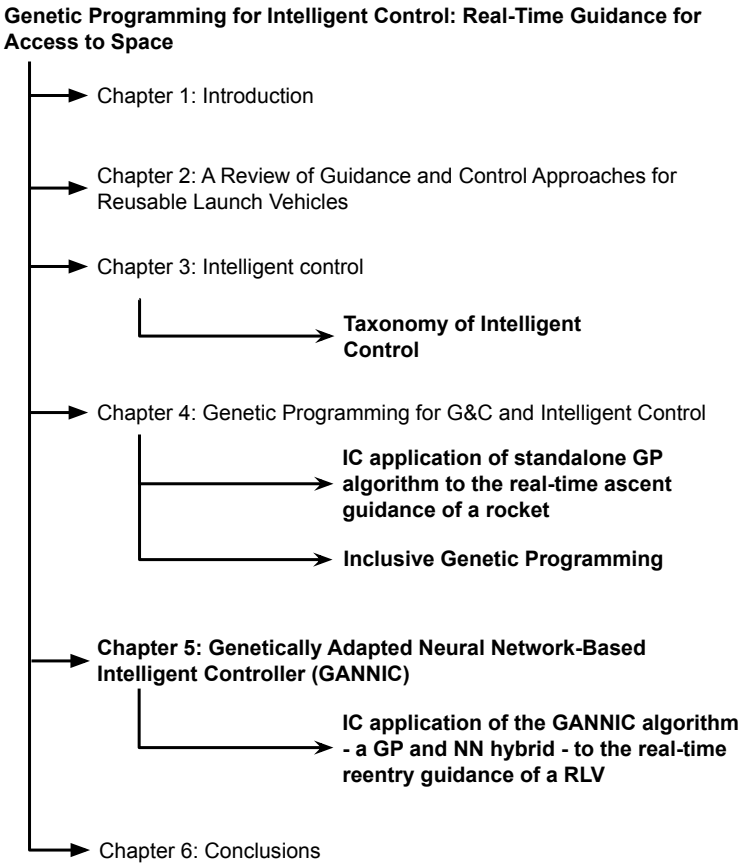


Figure 1.1: Thesis structure. The novel contributions are highlighted in bold.

Chapter 2 provides a comprehensive overview of the prominent G&C techniques employed in the aerospace industry, with a specific focus on RLVs applications. This high-level exposition serves to familiarize the reader with the relevant background information essential for contextualizing the research undertaken in this thesis and gaining insight into the current state-of-the-art in the field.

In chapter 3, the discipline of IC is described, encompassing an analysis of applications within the IC domain. A review of existing literature underscores the necessity for a more precise classification of what qualifies as IC and delves into the varying levels of intelligence exhibited by IC applications. Consequently, a novel taxonomy of IC applications is introduced and applied to enhance clarity and categorization.

Chapter 4 provides an in-depth exploration of GP and its application in the domain of IC. This is demonstrated by applying GP to perform the real-time ascent guidance of a Goddard rocket. The ascent trajectory is dynamically controlled by an online-generated GP guidance law, enabling an effective response to external disturbances. The chapter concludes with the description and test of a byproduct of the performed research. A novel GP heuristic named Inclusive Genetic Programming (IGP) was designed to address the population's diversity issue and proved efficient in solving G&C problems.

In chapter 5, the formulation of the innovative Genetically Adapted Neural Network-based Intelligent Controller (GANNIC) scheme - a GP and NN hybrid - is presented. The chapter describes the developmental process of the GANNIC scheme, addressing critical aspects that may impact its performance. The chapter concludes with the application of GANNIC for real-time reentry guidance of a RLV model. The GANNIC scheme robustness is assessed by incorporating uncertainties in the environmental models.

Chapter 6 contains a summary of the thesis, highlighting the novel contributions and acknowledging current limitations. Additionally, potential directions for future research are discussed.

Chapter 2

A Review of Guidance and Control Approaches for Reusable Launch Vehicles

2.1 Introduction

The aerospace industry is an exemplary environment for the advancement of Guidance and Control (G&C) methods, due to the stringent requirements that aerospace systems must meet and the inherent complexity of these systems. Essential criteria, including high performance, reliability, robustness, and safety, highlight the challenges faced by aerospace systems as detailed in [28]. Moreover, their dynamics are governed by highly nonlinear equations, and their performance is susceptible to various uncertainties. These uncertainties may arise from the complex nature of these systems, involving unknown interactions between subsystems, as described by Haibin et al. [29] in the context of hypersonic vehicles. Alternatively, uncertainties may arise from limited or incomplete knowledge of the operating environment, as observed in Mars entry vehicles [30] or deep space exploration missions [31].

This chapter provides the reader with an overview of the prevalent G&C schemes commonly employed in the context of Reusable Launch Vehicles (RLVs), focusing on the schemes' capability to handle uncertainties and to adapt to unforeseen situations. Based

on the performed literature review, an analysis of the following aspects is performed: 1) strengths and limitations of the state-of-the-art G&C schemes applied to RLVs, specifically focusing on their uncertainties handling capabilities; 2) how Intelligent Control (IC) could help in overcome limitations of the treated G&C schemes.

The chapter is structured as follows: section 2.2 presents a brief historical perspective on the evolution of control systems based on their capability to handle uncertainties, progressing from open-loop configurations to IC. Sections 2.3, 2.4 and 2.5 focus on optimal control, robust control and adaptive control respectively. For each of these control approaches the most widespread techniques applied to RLVs G&C are presented. For each technique, the discussion is structured with a brief introduction on its theoretical formulation and working principles, followed by examples of its application to RLVs or hypersonic vehicles, and concluded with comments on its strengths and weaknesses. Only applications involving ascent and re-entry phases of RLVs and hypersonic vehicles are considered since this is the focus of this thesis. Moreover, only works published from 2015 to the present day are taken into consideration, to assess the recent trends in the aerospace industry. In section 2.6, an introduction to IC and Artificial Intelligence (AI)-based G&C systems is presented, highlighting key AI techniques employed in RLVs G&C design. Section 2.7 presents a comparative analysis of the discussed G&C techniques addressing their strengths and limitations, and underscoring the significance of IC in overcoming the identified challenges. The chapter ends with section 2.8 comprising a summary and conclusive remarks.

2.2 Brief History of Control Systems Development [1]

To control a human-made machine, or plant, involves applying specific inputs to obtain desired responses or outputs. Over the past century, control theory has undergone significant advancements, enabling the development of controllers ranging from simple devices like home thermostats to complex systems such as the attitude control system of spacecrafts.

In the historical evolution of control systems, there has been a continuous adaptation

[1] Part of the content of this section was previously published in the journal paper [1].

to handle increasing uncertainties, enhancing reliability and robustness. The most basic architecture of a control system is the open-loop scheme illustrated in Figure 2.1, where the system's behavior is not considered, and a predefined sequence of control actions is applied.

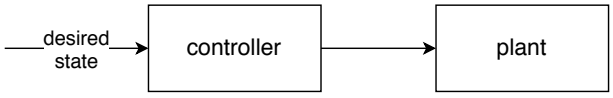


Figure 2.1: Open loop control scheme. [1]

Open-loop control systems demonstrate optimal performance only in an ideal scenario where the system and environment are perfectly understood. Acknowledging the presence of imperfections in mathematical models, the feedback control architecture was introduced by Nyquist in 1932 [32], as depicted in Figure 2.2. In a feedback control system, the current states of the plant are employed to adjust control actions, facilitating the accomplishment of specific tasks, such as following a predefined trajectory.

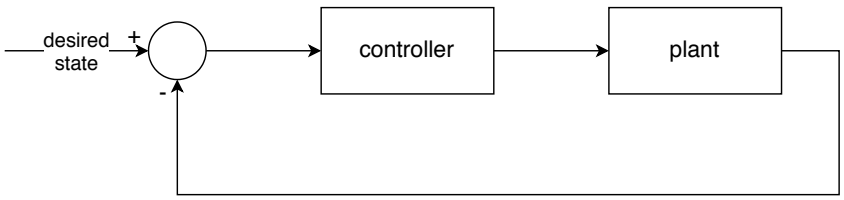


Figure 2.2: Feedback control scheme. [1]

The earliest feedback control systems were linear and designed using graphical techniques in the frequency domain. Pioneered by Bode, Ziegler, and Nichols [33], these systems incorporated the widely used Proportional-Integral-Derivative (PID) controller, which continues to be a widespread controller in industrial applications.

To enhance the performance and robustness of linear controllers, optimal control and robust control strategies were introduced. Optimal control, inspired by Bellman's foundational work [34], focuses on optimizing control system performance based on predefined metrics. This resulted in the development of linear optimal control methods such as Linear Quadratic Regulator (LQR), H_2, and Model Predictive Control (MPC), as well as nonlinear optimal control formulations. Conversely, robust control aims to

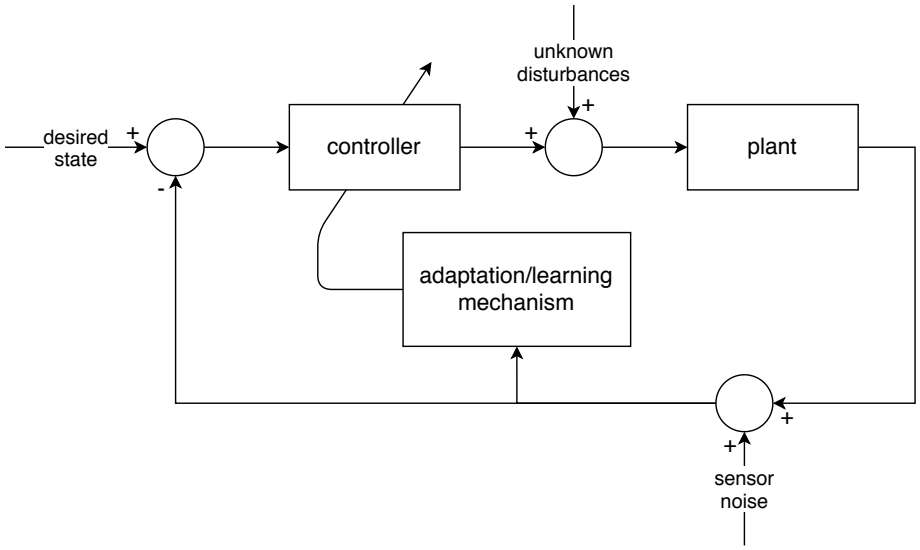


Figure 2.3: Adaptive or learning control scheme [1].

maintain controllability in the presence of uncertainties, allowing the control system to directly address them. This led to the creation of controllers such as H_∞, Sliding Mode Control (SMC), backstepping, and dynamic inversion-based controllers.

Control theory advanced further with the introduction of adaptive controllers. An adaptive controller, depicted in Figure 2.3, enhances the performance and robustness of optimal and robust controllers by incorporating an adaptive capability into the control scheme. Unlike optimal and robust controllers, which are designed offline to handle expected uncertainties at design time, an adaptive controller can dynamically adjust its parameters or objectives online in response to changes in the environment or plant conditions. This adaptation capability renders the controller nonlinear, in contrast to optimal and robust controllers explicitly designed for linear systems [35].

The next advancement in control theory is represented by intelligent controllers. As further detailed in chapter 3, a control system earns the label "intelligent" when it can dynamically adapt online and incorporates an AI technique. The integration of AI enhances the adaptation capabilities of a control system, enabling it to handle a broader spectrum of uncertainties and effectively utilize the data generated by sensors.

2.3 Optimal Control Approaches

Optimal control focuses on designing a control law for a given plant by optimizing a specified performance measure. For instance, in the context of a launch vehicle, trajectory optimization may seek to minimize fuel consumption. Rooted in the calculus of variations, the history of optimal control spans over 360 years, becoming particularly relevant from the 1960s with the advent of technological advancements, such as computers, enabling the evaluation of optimal trajectories for aerospace systems [36].

The general formulation of an optimal control problem is presented in Equation 2.1. Here, J is a defined cost functional to be minimized, $\mathbf{x}$ represents state variables, and $\mathbf{u}$ denotes control variables within the admissible control set U. The states $\mathbf{x}$ must adhere to the initial value problem defined by the system's dynamic equations, and ψ represents a set of final conditions. Additionally, S denotes a set of inequality constraints that may or may not be considered.

$$\begin{aligned} \min_{\mathbf{u}(t),\, t \in [t_0, t_f]} \quad & J(\mathbf{x}(t), \mathbf{u}(t), t) \\ \text{s.t.} \quad & \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t), \\ & \mathbf{x}(t_0) = \mathbf{x}_0, \\ & \mathbf{u} \in U, \\ & \psi(t_f, x_f) = 0, \\ & S(\mathbf{x}) \geq 0 \end{aligned} \tag{2.1}$$

Various optimal control approaches have been devised for specific applications based on the formulation and solution of the optimization problem. The most pertinent ones for controlling RLVs are discussed in the following. For an in-depth exploration of optimal control, readers are directed to [37, 38, 39, 40].

2.3.1 Linear Quadratic Regulator

The Linear Quadratic Regulator (LQR) stands out as one of the most widely adopted optimal control approaches, characterized by its simplicity and assured optimality. An

LQR controller enables the identification of the optimal trade-off between control effort and stability for the controlled system.

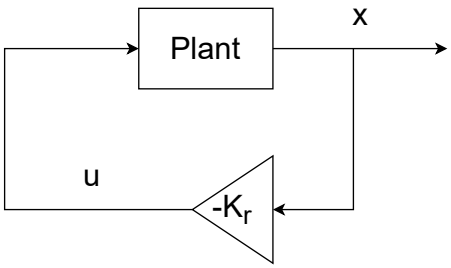


Figure 2.4: LQR generic scheme

Considering a system as illustrated in Figure 2.4, the objective is to determine the set of control gains $\mathbf{K}_r$ capable of minimising the cost function in Equation 2.2

$$J(t) = \int_{t_0}^{t_f} \mathbf{x}(t)^T \mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)^T \mathbf{R}\mathbf{u}(t) dt \tag{2.2}$$

The gains $\mathbf{K}_r$ are computed as outlined in Equation 2.3, where $\mathbf{X}$ represents the solution of the Riccati equation provided in Equation 2.4.

$$\mathbf{K}_r = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{X} \tag{2.3}$$

$$\mathbf{A}^T\mathbf{X} + \mathbf{X}\mathbf{A} - \mathbf{X}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{X} + \mathbf{Q} = \mathbf{0} \tag{2.4}$$

For more information on the LQR controller, the reader is referred to [41, 42].

Regarding some examples of the LQR controller applied to aerospace vehicles, in a study by Lu et al. [43], an LQR controller was formulated to track the reentry trajectory of a hypersonic vehicle, accounting for aerodynamic disturbances. The model underwent small perturbations linearization, and the controller's robustness was evaluated by varying the parameter set. Li et al. [44] devised an LQR controller for the air rudder of a rocket. Through hardware-in-the-loop simulation, the controller demonstrated enhanced tracking accuracy, stability margin and dynamic characteristics compared to a traditional PID controller. A last example is the work of Wang et al. [45], where

they proposed a novel G&C scheme for the reentry phase of a hypersonic vehicle. They adopted an unscented Kalman filter to estimate online the aerodynamic coefficient and reevaluated online the trajectory using a pseudospectral method. A LQR scheme is used to perform the trajectory tracking guidance and an SMC is used for attitude control. Their approach proves capable of achieving satisfactory tracking errors on the states under large aerodynamic coefficients errors.

Despite their simplicity and optimality guarantees, LQR controller's applicability is limited to linear systems. Linearization techniques are required, introducing modelling errors, and the guaranteed stability margins may not apply to real systems "due to the constraints in the selection of measurable states" as explained by Zhang et al. [46]; Additionally, LQR controllers tend to fail when uncertainties and disturbances are considered. To address this issue, Linear Quadratic Gaussian (LQG) regulators are introduced. However, LQG regulators have drawbacks, such as a lack of guaranteed stability margins, as demonstrated by Doyle [47].

2.3.2 Trajectory Optimization

In contrast with LQR control, where the considered controller is of the feedback type, trajectory optimization involves techniques that aim to find an optimal open-loop control trajectory. Trajectory optimization approaches can be classified as direct or indirect based on how the optimization problem in Equation 2.1 is formulated.

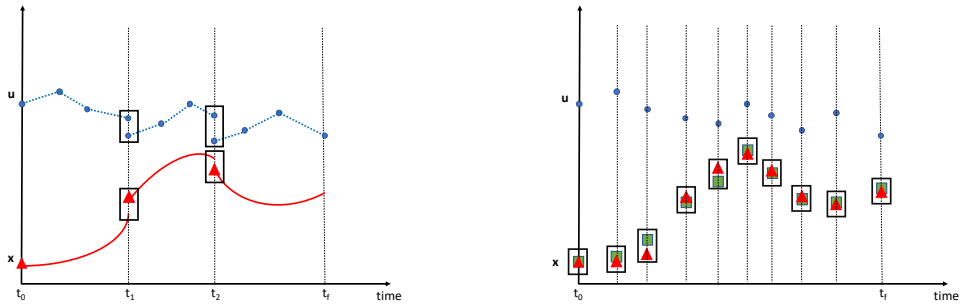
Indirect optimal control reduces the optimization problem to a Boundary Value Problem (BVP). This method requires deriving the adjoint equations of the considered system analytically along with their gradients. However, this step can be cumbersome for complex systems, and the results are highly dependent on the provided initial guess. Achieving good results without a proper initial guess is challenging. Moreover, indirect optimal control approaches struggle with nonlinear constraints, making them rarely used for real-world applications. Nevertheless, some recent applications can be found in the literature [48, 49], although their number is extremely limited.

On the other hand, if the optimization problem is reduced to a Nonlinear Programming (NLP) problem, the approach is defined as direct optimal control. The NLP

problem is then solved using a numerical optimization solver [50, 51]. This allows for consideration of nonlinear equality and inequality constraints, as well as boundary constraints on the optimization variables. Several transcription techniques can be used to reduce the original optimization problem to a NLP problem. Two major branches can be identified: shooting methods and collocation methods.

Shooting methods involve placing a defined number of control points along the trajectory, interpolating them to find a control law for each trajectory point, and using such a control law to propagate the system's dynamics equations. The optimizer aims at finding the optimal sequence of control points to satisfy the defined objective function. Additionally, the original trajectory can be split into n smaller sub-trajectories within the limits of the initial trajectory. Then n propagations are performed to reduce errors introduced by numerical propagation. This is called multiple-shooting transcription, schematically depicted in Figure 2.5a. In contrast, with collocation methods, the optimizer places a certain number of control and state points along the trajectory, as depicted in Figure 2.5b. The state points must satisfy the equality constraints with the system's dynamic equations, which are evaluated using the control points provided by the optimizer. Therefore, the optimization seeks to find the optimal sequence of control and states that minimizes the chosen objective function while satisfying the equality constraints on the states. Both approaches have pros and cons. Shooting methods are more precise, as the trajectory is obtained by propagating the system's dynamics using the optimized control points. In contrast, with collocation methods, the trajectory is obtained as a result of the equality constraints satisfaction and may not necessarily match the trajectory obtained by propagating the system's dynamics using the optimal control points. Nonetheless, collocation methods are faster, as the propagation of the system's dynamics is a time-consuming process, and with enough collocation points, accurate results can be obtained.

Trajectory optimization has a rich history of successful applications in the aerospace industry, particularly for ascent vehicles [52]. A few examples are discussed in the following. In one of the studies conducted during this thesis development [18], a trajectory optimization analysis was performed on the ascent trajectory of a RLV using a direct



(a) Multiple-Shooting Transcription [18] **(b)** Collocation Transcription

Figure 2.5: Main optimal control transcriptions: a) Multiple-shooting, the blue dots are the control points, and the red triangles are the states. b) Collocation, the blue dots are the control points, the red triangles are the states points chosen by the optimizer, and the green squares are the states evaluated with the system's dynamics. The black boxes denote where equality constraints are applied in both images.

multiple-shooting approach. The initial guess for the optimization process was discovered through a combination of two evolutionary algorithms designed by one of the authors [53, 54]. Kumar et al. [55] tackled the trajectory optimization problem for a hypersonic vehicle using a Genetic Algorithm (GA). Path and terminal constraints were imposed for the considered phase of flight, and GA successfully found the global optimum for the analyzed problem, offering greater flexibility in incorporating constraints compared to gradient-based methods. In another work by the same authors [56], trajectory optimization for a reentry vehicle was explored using various gradient-free optimization algorithms. The Aerospace Centre of Excellence at the University of Strathclyde has made significant contributions to trajectory optimization. For instance, Ricciardi et al. [57] proposed a novel method to solve multiphase, multi-objective optimal control problems by combining the Direct Finite Elements in Time transcription with a Multi-Agent Collaborative Search. This approach was robust and accurate, providing sets of Pareto optimal solutions for three test cases: ascent, reentry, and abort scenarios. Another example from Strathclyde's work is found in [58], where the abort trajectory of an ascent vehicle is optimized to maximise downrange and crossrange distances. The initial guess for the optimization process is obtained through a multi-start

analysis on the first point, followed by successive iterative generation, proving to be successful for all analyzed abort points.

Trajectory optimization, while powerful for enhancing aerospace system performance, comes with high computational costs and is typically conducted offline. This implies that an optimized trajectory is only optimal for the specific set of uncertainties or disturbances considered, lacking robustness. Furthermore, for nonlinear systems like RLVs, the optimization process becomes challenging, and the selection of the initial guess can be crucial, depending on the chosen optimization algorithm.

2.3.3 Model Predictive Control

If the optimization problem in Equation 2.1 is solved iteratively on a finite-time horizon in a closed-loop fashion, the resulting control scheme is referred to as MPC. Introduced at the end of the 1970s, MPC emerged as an alternative to classical controllers, such as PID controllers. Its fundamental characteristic lies in leveraging the knowledge of the system to be controlled to predict its future behavior and optimize it. This optimization occurs using a receding finite-time horizon, meaning that at each time step, an optimization is performed based on the information available at that time step. Consequently, a control sequence is generated from the current time step to the end of the time horizon through the optimization process. The first control action of this sequence is then employed to guide the system to the next time step, and the optimization process repeats. A schematic representation of the MPC scheme is illustrated in Figure 2.6.

An interesting aspect of MPC is its capability to consider constraints, even though this inclusion leads to an increase in computational time. This computational burden significantly limits the range of applications suitable for MPC. MPC is particularly well-suited for linear or slow varying processes, such as industrial processes, where computational time is not a critical factor, and the optimization process can be executed online while considering constraints. On the contrary, when dealing with highly nonlinear or rapidly changing systems, successfully applying MPC becomes challenging. Additionally, for linear and well-known models, mathematical proofs of robustness

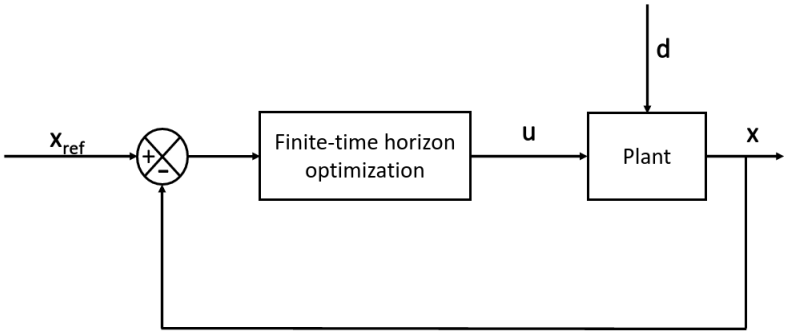


Figure 2.6: Simplified MPC control scheme

and stability for MPC can be formulated [59]. In contrast, in more general nonlinear cases, assessing the robustness and stability of MPC is often difficult, if not impossible [60, 61].

Despite its computational challenges, MPC has found some applications in the aerospace industry. Chai et al. [62] introduced an adaptive MPC scheme applied to the linearized model of a flyback booster reentry. Their MPC controller, designed for high update rates, integrated an $\mathcal{L}_1$ adaptive augmentation to compensate for matched and unmatched uncertainties. A recent study presented by Guadaganini et al. [63] explores a successive convexification MPC guidance algorithm for 6-DOF Powered Descent Guidance. Two control architectures are proposed. The first combines MPC-Guidance with a PID-Controller (MPC-G&PID-C), optimizing the trajectory during flight. The second, MPC-G&C, follows a classical MPC approach, solving the optimal control problem at a higher frequency. Sensitivity analyses show that the MPC-G&PID-C architecture is robust to dispersed parameters, reducing errors. The inclusion of a low-level PID controller aids trajectory alignment. Despite challenges, the outcomes highlight MPC guidance as promising for online autonomous guidance in achieving precise pinpoint landings.

For a more comprehensive review of MPC applications in the aerospace domain, interested readers can refer to [64].

In comparison to trajectory optimization approaches outlined in Subsection 2.3.2,

MPC emerges as a more robust technique. Its strength lies in its ability to consider unforeseen disturbances and uncertainties through online optimization. However, it's worth noting that this optimization process can be time-consuming, imposing limitations on the applicability of MPC across various systems.

2.3.4 Convex Optimization

The optimization problem described in Equation 2.1 is deemed convex if the involved functions—J, f, ψ, and S—are themselves convex. Convex problems hold significant appeal for the engineering community due to the availability of efficient numerical algorithms. These algorithms come with theoretical guarantees of finding solutions within known computational bounds. This is particularly crucial in G&C applications, where stringent requirements for precision and computational speed exist [65].

In the realm of optimal control, convex optimization plays a vital role in enhancing the efficiency of trajectory optimization algorithms or in conjunction with MPC. Convexification techniques are employed to convert the original non-convex optimization problem into a convex one. Conceptually similar to linearization methods, these transformations simplify the problem for easier treatment. However, it is important to note that the resulting models may contain less information than the original ones. Further details on convex optimization can be explored in [66].

In recent years, the popularity of convex optimization techniques has increased, notably influenced by the work of Açikmeşe, Carson, and Blackmore [67]. Their contributions have played a pivotal role in the successes of companies like SpaceX, achieving safe landings of the launcher's first stages on the ground or floating platforms post-ascent. Convex optimization is frequently integrated with MPC, as demonstrated in [68]. This study applied convex optimization with pseudospectral discretization in an MPC framework, specifically addressing the fuel-optimal rocket landing problem. The algorithm proved optimal, robust, and feasible for online computation, considering constraints and disturbances. Sagliano et al. [69] introduced an online trajectory optimization algorithm for the reentry trajectory of a Space Shuttle-like entry vehicle. Their approach utilizes a loss-less convex representation for entry guidance, leveraging

drag dynamics. The optimization problem is discretized with pseudospectral methods, incorporating constraints. Notably, their algorithm requires solving only one convex problem to generate a feasible reentry trajectory, showcasing precision and computational efficiency. Szmuk et al. [70] devised a continuous formulation for compound state-triggered constraints, applying it to the 6-DoF powered descent of a rocket within the successive convexification framework. Their approach considered velocity-triggered angle of attack constraints to mitigate aerodynamic loads and collision avoidance constraints to prevent obstacles near the landing site. While promising for powered descent guidance, the authors acknowledge the need for further work to enhance the convergence properties of state-triggered constraints.

Convex optimization for G&C in RLVs applications represents a captivating and relatively new area of research. Recent publications and industry achievements underscore the validity and potential of this technique. However, it is crucial to note a few considerations regarding its applicability. The use of convexification techniques results in "simplified" models, potentially introducing a loss of information. Additionally, convexification methods may be susceptible to poor convergence, as highlighted in works like [71, 72]. These factors underscore the need for careful consideration and further research in the application of convex optimization in this context.

2.4 Robust control Approaches

Robust control methodologies emerged in the 1980s to tackle challenges posed by uncertainties in plant and environmental models. These approaches are predominantly deterministic and model-based, relying on mathematical formulations and existing models of the considered systems. Known for their ease of implementation and backed by mathematical proofs of convergence, they are often deemed reliable. However, their effectiveness is confined to scenarios with bounded uncertainties, and being static methods, they cannot adapt online to uncertainties beyond their predefined bounds. Among the robust control approaches commonly applied to RLVs in the literature, H_∞ control and Sliding Mode Control (SMC) stand out.

2.4.1 H_∞ Control

H_∞ control theory, introduced by Zames in the '80s [73], is rooted in optimal control theory. In the generalized control scheme illustrated in Figure 2.7, both the plant G and controller K matrices are in state-space form, and they are both controllable and observable. The external input w could represent a disturbance, z is the controlled output, and y is the measured output. In this general scenario, the objective of the H_∞ controller is to determine a controller K such that the H_∞ norm of the transfer function from w to z, denoted as $||T_{wz}||_\infty$, is less than a specified value γ [74]. The system is considered stable for all admissible uncertainties $||\Delta||_\infty < 1/\gamma$. For a more in-depth exploration of the theoretical aspects of H_∞ control theory, readers are referred to [42, 75, 76].

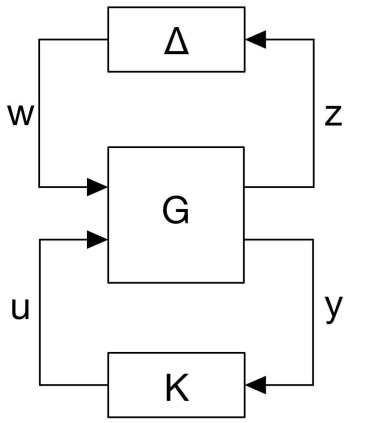


Figure 2.7: H_∞ control scheme configuration

The H_∞ controller has been utilized in a series of studies published by European Space Agency (ESA), specifically applied to the control of the VEGA launch vehicle [77, 78]. In [77], a methodology for synthesizing H_∞ controllers was presented, incorporating newly designed weighting functions to facilitate the controller design and tuning phases during the development of the control system. The approach was tested on the attitude control of the VEGA launcher, demonstrating that the same gains used in an actual flight could be recovered through their methodology. The results were further validated using a nonlinear high-fidelity simulator in [78]. The synthesis of an H_∞ con-

troller was also explored in [79], where it was employed to track the descent phase of a reusable rocket, and its performance was assessed using a nonlinear simulator. Another application of a robust H_∞ controller was presented in [80], where a controller for a hypersonic aerial vehicle was designed. The vehicle's dynamics were linearized around temporary operating points for controller development, and global asymptotic stability was established through Lyapunov analysis.

H_∞ control is a powerful technique that ensures closed-loop stability in the presence of bounded disturbances, enabling the derivation of an optimal or sub-optimal controller to optimize a specific performance index. However, its application is limited to linear systems in state-space form. Consequently, when dealing with a nonlinear system, linearization is necessary, introducing model uncertainties. Additionally, H_∞ control is effective only for bounded uncertainties, and failure may occur if these bounds are unknown.

2.4.2 Sliding Mode Control

Sliding Mode Control (SMC) originated in the late '50s in the Soviet Union as part of Variable Structure Systems research. The design process involves two steps: 1) designing a switching function to guide the system on the sliding manifold according to specifications, and 2) selecting a control law to attract the system to the sliding manifold in the presence of disturbances/uncertainties [81]. The sliding manifold, defined by points where the sliding variable equals zero, is user-designed and determines the desired closed-loop performance.

SMC possess insensitivity to internal and external disturbances, ensuring the convergence of the sliding variable to zero for a stable configuration. The inherent nonlinearity of SMC arises from the discontinuity introduced by the switching function. Despite its robust control capabilities, SMC faces challenges like chattering. Various SMC formulations were introduced to address these issues, including Terminal SMC for finite-time stability, Higher-Order Sliding Mode Control (HOSMC) to reduce chattering, and Integral Sliding Mode Control (ISMC) for chattering mitigation while maintaining finite-time stability. Further details on different SMC configurations can

be found in [71], and a more in-depth discussion of SMC theory is available in the literature [82, 83, 84].

The SMC scheme has gathered considerable attention in recent decades due to its robustness against uncertainties and mathematical proofs of convergence. Researchers have frequently employed RLVs as testbeds to assess advancements in SMC theory. This overview highlights some recent applications of SMC in the context of RLVs. Zhang et al. [85] introduced a Non-singular Fast Terminal Sliding Mode Control (NSFTSMC) for the attitude tracking of a vertical take-off and vertical landing RLV. They incorporated a fixed-time extended state observer to estimate error states and disturbances, accounting for uncertain parameters and external disturbances. Additionally, they designed a novel fast terminal sliding mode surface along with a corresponding fixed-time controller, demonstrating smaller steady-state errors in comparison to previous works. Guo et al. [86] proposed an adaptive twisting SMC for controlling the attitude of a hypersonic reentry vehicle. Their approach accommodates disturbances with unknown bounds, and the designed control law effectively mitigates the chattering phenomenon. SMC was also employed for guidance applications, as done in the works of Liao et al. [87] and Cho et al. [88]. Liao et al. [87] addressed the approach and landing guidance problem for an RLV using a finite-time ISMC. Their work showcased the stabilization of the system within a predefined finite time, expressing the system's states analytically. Cho et al. [88] presented a singular sliding mode guidance approach for a missile to intercept a target at a specific time. The guidance law was adapted to avoid singularities and included an additional component to ensure that the sliding mode remained the sole attractor.

Decades of active research on SMC have led to significant advancements and the resolution of some inherent challenges associated with this technique. Strategies such as ISMC and HOSMC have been developed to address the chattering phenomenon, while the Non-singular Fast Terminal Sliding Mode Control (NSFTSMC) has been introduced to mitigate singularity issues. However, these refinements bring forth new considerations. For instance, HOSMC poses challenges in obtaining information about higher-order derivatives of the system and demands an accurate system model. On the

other hand, ISMC exhibits a propensity for large overshoot and extended regulation time when faced with significant initial errors. Additionally, akin to H_∞ control, SMC is constrained by its ability to handle only bounded uncertainties, limiting its capacity for online adaptation.

2.5 Adaptive Control Approaches

An adaptive control system is designed to dynamically adjust its parameters in response to changes in the controlled system's parameters or environmental conditions. This real-time adjustment is facilitated by an adaptation mechanism, introducing nonlinearity into the controller [35, 89]. The schematic representation of an adaptive control scheme is depicted in Figure 2.3 in section 2.2. Essentially, any controller design can be rendered adaptive by incorporating a parameters adaptation mechanism. This mechanism, responsible for online adaptation, adjusts controller parameters, such as gains, to ensure continued plant controllability in the presence of disturbances.

Adaptive control can be categorized into open-loop, direct, and indirect schemes based on the adaptation mechanism employed. An example of an open-loop adaptive controller is gain scheduling, where controller parameters are updated online according to a predefined adaptation sequence. In a direct adaptive controller, the adaptation mechanism and a reference model are combined. The adaptation mechanism receives the error between the output predicted by the reference model and the actual plant output as input. In the indirect adaptive control scheme, a plant estimator is updated online, and its output is compared with the real output. The resulting error is then fed into the controller adaptation mechanism, updating the controller parameters. This method is termed indirect because the adaptation mechanism occurs in two steps: first, the plant parameters are estimated online, and second, the controller parameters are estimated online based on the estimated plant model.

Adaptive control shares similarities with robust control, but with a distinct focus on leveraging adaptation mechanisms to enhance resilience against uncertainties or disturbances. An adaptive controller, therefore, exhibits the ability to withstand a broader range of disturbances compared to a robust controller, where robustness is predefined

and embedded in the controller. Nevertheless, a robust controller can be enhanced with adaptive features to enhance its performance. The concept of learning has been closely associated with adaptive control [90], leading to the integration of AI techniques into adaptive controllers to augment their adaptation and learning capabilities, giving rise to intelligent controllers, as elaborated in section 2.6 and chapter 3. Despite its advantages, adaptive control faces a set of challenges outlined in [91], including impractical control objectives stemming from an incomplete plant description leading to instability, and failure in coping with instabilities due to a plant component failure.

2.5.1 Gain Scheduling

Gain scheduling represents a straightforward adaptation mechanism applicable to various controller designs. This method involves linearizing the plant around a set of operating conditions. A controller is then synthesised for these conditions and its parameters are stored in a lookup table. This table is then used to schedule the controller's parameters change during operation according to the current operating condition. To obtain the controller's parameters between operating conditions, interpolation is used. Gain scheduling is categorized as an open-loop adaptation mechanism since the lookup table is defined offline based on the predicted plant trajectory and the anticipated ranges of environmental and plant state variations. Further insights into gain scheduling theoretical aspects can be found in [92].

This technique was extensively applied for the attitude control of RLVs, as demonstrated by older works such as [93, 94]. A recent example that combines gain scheduling and an LQR controller is presented by Hameed et al. [95]. They developed a finite horizon LQR with gain scheduling for the guidance and control of the approach and landing phase of a RLV. The controller's robustness and effectiveness were evaluated by varying the flight's initial conditions and considering aerodynamic uncertainties. Their approach proves capable of minimizing the state errors to achieve touchdown at a specified downrange in the presence of variations in the initial conditions. Another common application of gain scheduling is represented by its combination with H_∞ controllers. An example of such an approach is represented by the work of De Oliveira et al. [96].

The authors investigate the design of a structured H_∞ gain-scheduled controller for the atmospheric re-entry of RLVs. The control approach is validated through linear analysis in both time and frequency domains, as well as nonlinear analyses using a 6-DoF RLV re-entry dynamics simulator. However, these validations also emphasize the requirement for increased stability and robustness in handling uncertainties.

Although straightforward to implement, gain scheduling introduces stability problems, and the robustness of the controller is questionable as pointed out by Castaldi et al. [17]. Additionally, a major drawback of this technique is that the scheduled gains cannot be updated online, posing a risk of failure if the plant operates outside the predefined set of operating conditions. This issue can be mitigated by increasing the number of operating points to cover the entire flight envelope. However, this approach poses practical challenges, limiting the inclusion of all possible operating conditions.

2.5.2 Model Reference Adaptive Control (MRAC)

In Model Reference Adaptive Control (MRAC), a reference model representing the desired plant's input-output relationships serves as a reference for the control system. The control system aims to identify a feedback control law that, when applied to the actual plant, aligns with the input-output relationships defined by the reference model. MRAC can be categorized as either direct or indirect, depending on the availability of a plant model from which the output is measured. In the case of indirect MRAC, an estimator of the plant's behavior is utilized.

A schematic representation of direct MRAC is illustrated in Figure 2.8, where the reference command r serves as input to both the controller and the reference model. The output of the reference model produces the control action u and the reference output x_{ref} corresponding to the given reference command. The control action u is then applied to the plant model, generating an output x. The tracking error e between the reference output x_{ref} and the current output is computed and fed into the adaptation mechanism. The adaptation mechanism adjusts the controller parameters based on the current plant conditions. Further theoretical details on MRAC can be explored in [97].

Applications of MRAC for RLVs were investigated by renowned authors such as

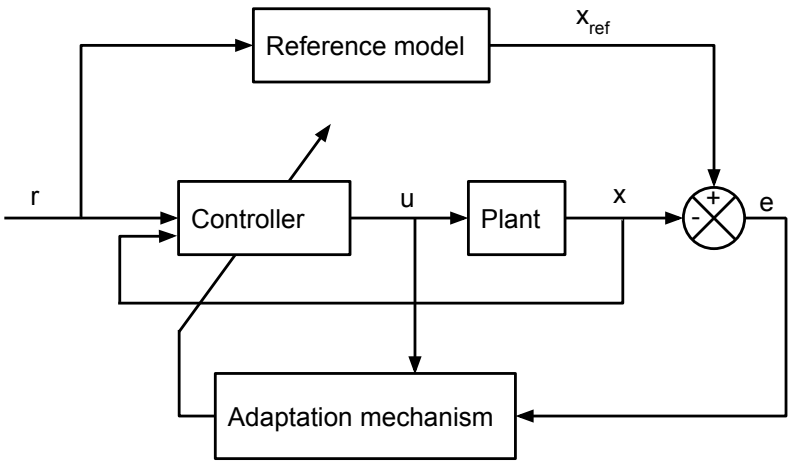


Figure 2.8: MRAC schematic configuration

Johnson et al. [98] and Orr et al. [99]. More recently, applications of MRAC focus on combining this scheme with some AI technique, such as the Deep MRAC framework presented by Joshi et al. [100]. Recent examples not involving AI, can be found in the work of Nair et al. [101] and Guo et al. [102]. In [101], a Lyapunov-based PD/PID adaptive controller was developed within the MRAC framework to govern the ascent trajectory of an RLV. The authors conducted a comparative analysis with a gain-scheduled PD/PID controller, demonstrating that their proposed approach achieved superior tracking performance and enhanced robustness against disturbances. In [102], a novel MRAC approach was devised by integrating it with a backstepping approach. The developed control system was employed for the attitude control of a near-space hypersonic vehicle. Stability analysis using Lyapunov stability theory was conducted, and simulation results showcased robustness against input constraints and parameter perturbations. The proposed scheme outperformed a standard MRAC configuration in terms of stability and control effectiveness.

MRAC offers an effective solution to perform an informed adaptation by tracking the behaviour of a reference model. As mentioned in [103], the use of a reference

model results in reduced sensitivity to environmental changes, modelling errors and non-linearities within the system. However, MRAC schemes are characterized by a great number of design variables which must be tuned appropriately to obtain good performance and this is usually done relying on the experience of the designer. Moreover, as Shekhar et al. point out [104], it is difficult to assess MRAC schemes' stability when applied to nonlinear systems.

2.5.3 Real-time or Adaptive Guidance

Real-time or adaptive guidance encompasses those schemes that generate guidance commands online, enabling the plant to track the desired reference trajectory despite external disturbances or uncertainties. These approaches are complementary to convex optimization or MPC ones where the reference trajectory is recomputed during flight to cope with disturbances and uncertainties. Adaptive guidance schemes can also be designed using AI algorithms. In that case, they are also referred to as Intelligent Guidance as done in [105], however, this terminology is not widespread and oftentimes approaches built with AI are either termed as adaptive guidance, e.g. in the works of Johnson et al. [106], or generally as IC. The latter terminology is used in this thesis, i.e. adaptive guidance approaches involving AI techniques are categorized as IC approaches. The guidance schemes developed during this doctoral work and described in chapters 4 and 5 of this thesis fall under this category.

A discussion on adaptive guidance approaches built using AI is given in chapter 5.1, while this section focuses on adaptive guidance approaches designed without using AI techniques.

Zhu et al. [107] present a robust adaptive gliding guidance strategy for a hypersonic vehicle. Their method involves the design of an analytical optimal gliding guidance law and robust compensation of aerodynamic coefficients using an Extended Kalman Filter (EKF). Aerodynamic coefficients are expressed as quadratic polynomial functions of flight states, with unknown parameters estimated using EKF. The estimated values are then utilized to calculate the required angle of attack for robust compensation. This approach accurately satisfies terminal multiple constraints over a broad range of disper-

sion. Another example is discussed by Mooij [108], who presents an adaptive guidance technique for the reentry of a hypersonic vehicle. Three approaches are tested to track the heat flux profile: an inner loop output-feedback controller, an outer loop adaptive tracker based on simple adaptive control theory, or a combination of both. The combination of inner and outer loops tracking improves the performance with respect to the nominal guidance, satisfying the heat-flux and g-load constraints. Results demonstrate stability and robustness in trajectory tracking against applied perturbations. Another example of integrated adaptive guidance and control can be found in [109]. The authors developed an integrated nonlinear adaptive guidance and control approach applied to the lading phase of a RLV. They introduced an adaptation law into the backstepping scheme to handle uncertainties, while aiming at tracking the flight path angle reference trajectory. Results obtained through numerical simulations show that their approach can effectively perform the RLV landing in the presence of uncertainties, while a Lyapunov stability analysis is used to assess the overall system's stability. A last example is proposed by Yan et al. [110]. The authors developed an adaptive guidance approach for hypersonic entry vehicles, based on a feedback control law that outputs the bank angle command based on the quasi-equilibrium glide condition. Their algorithm is composed of two parts, longitudinal and lateral profile guidance. The longitudinal guidance is responsible for generating the magnitude of the bank angle analytically in real-time, while the lateral one is used to determine the sign. The approach is tested considering a broad range of entry trajectories with applied uncertainties, and proves to be effective in guiding the vehicle towards the desired final position.

Adaptive guidance can help in relieving the pressure posed on the attitude controller and in avoiding the accumulation of control errors, by generating guidance commands to track new trajectories that already consider deviations due to uncertainties or disturbances. As discussed by Song et al. [111], this results in an increased robustness against disturbances and uncertainties. However, in adaptive guidance schemes online computations are performed and these can cause mission failure if not performed in an adequate time window.

2.6 AI-based and Intelligent Control approaches

While AI has gained substantial significance in various aspects of everyday life, ranging from social networks to healthcare and home automation, its incorporation into control applications, particularly in the aerospace field, remains limited outside the academic domain. However, the intersection of AI and control theory represents an active research area, resulting in the development and testing of numerous algorithms over the past decades. As detailed in chapter 3, the application of AI in a control system does not guarantee that the control scheme is intelligent. True intelligence emerges from the fusion of AI with online learning and adaptation capabilities. In this context, intelligent controllers share similarities with adaptive control systems as they dynamically update the controller's parameters or structures online. However, the former achieves this through the utilization of AI techniques. Compared to classical adaptation mechanisms, AI-based controllers enhance robustness against a broader range of disturbances, by leveraging the nonlinear character of AI algorithms and their ability to efficiently exploit data. This section outlines the three primary families of AI techniques employed for control - Machine Learning (ML), Evolutionary Computing (EC), and Fuzzy Logic (FL) - alongside some of the latest AI-based and IC approaches applied to RLVs. Figure 2.9 schematically illustrates how these techniques are often combined in control systems.

Within the ML family, the focus is given to Neural Networks (NNs), since it is one of the techniques used to develop the guidance scheme presented in chapter 5, while a detailed description of Genetic Programming (GP) is given in chapter 4. A comprehensive literature review of IC applications is given in chapter 3 and in the two technical reports developed as part of this thesis on behalf of ESA [14, 15].

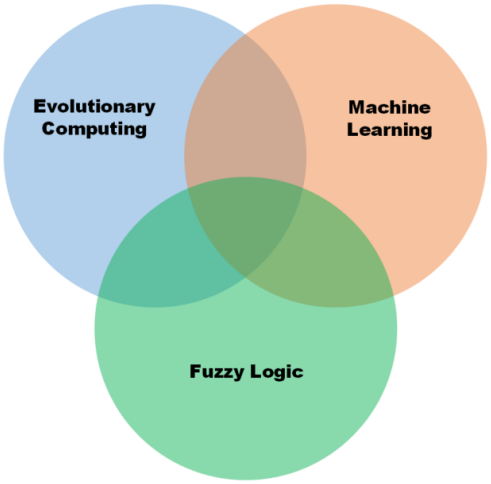


Figure 2.9: Synergies of artificial intelligence techniques used for intelligent control. [1]

2.6.1 Fuzzy Logic

Fuzzy Logic (FL) operates on the principle of reasoning by leveraging partial truths. In this context, partial truth refers to the truth values of variables, which can take any real number within the range of 0 to 1. This is in contrast with Boolean logic, where truth values are constrained to either 1 or 0 [112]. Designers define these partial truths, incorporating expert knowledge into the controller design and resulting in an interpretable control scheme. Partial truths are then used to evaluate propositions, allowing for reasoning in situations where precise numerical values are difficult to ascertain. FL can be employed both intelligently and non-intelligently. If its parameters are updated online, it is considered an intelligent application; otherwise, if the FL control law is not updated online, it cannot be deemed intelligent.

An instance of FL applied to control RLVs is presented in [113], focusing on the attitude control of a RLV during its reentry phase. The authors devised a compound adaptive fuzzy H_∞ control strategy. They tested their approach against uncertain controller parameters and disturbances, and they used the Lyapunov theory to assess the closed-loop stability of their method. The results demonstrate the effectiveness of this approach, qualifying it as an intelligent controller due to the online updating of control parameters. Further applications of FL in IC are detailed in chapter 3.

2.6.2 Machine Learning

ML techniques are particularly suitable for IC applications due to their inherent learning capabilities. Notably, Neural Networks (NNs) and Support Vector Machines (SVMs) stand out as commonly employed architectures. While these structures find widespread application in classification tasks, they can also serve as nonlinear function approximators. Their versatility extends to modelling dynamical systems, approximating uncertainties, or even being directly integrated as controllers [114].

Neural Networks

NNs have become one of the most prevalent ML techniques, finding applications across various domains for classification and regression tasks. The fundamental structure of a NN is illustrated in Figure 2.10, where its primary components, called neurons, are organized into layers.

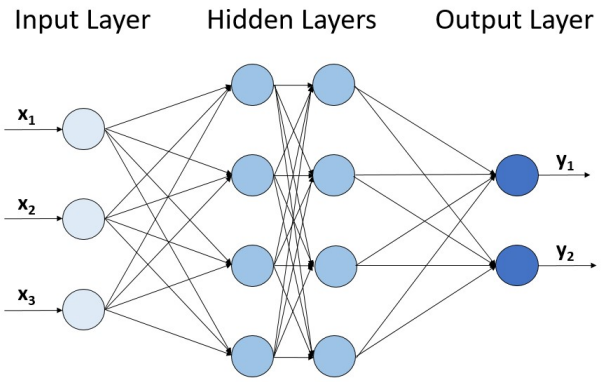


Figure 2.10: Simple Feedforward Neural Network architecture with two hidden layers

Neurons across different layers can either be fully connected, as depicted in Figure 2.10, or follow a different architecture based on specific requirements. When multiple layers are stacked consecutively, the resulting NN is termed a Deep Neural Network (DNN). Each neuron incorporates an activation function, denoted as $\sigma(z)$ in Figure 2.11.

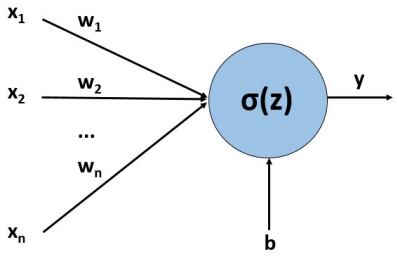


Figure 2.11: Close up of a single neuron

This function takes as input the output of the preceding layers (x_j), multiplied by a connection weight (w_j), plus a bias term (b_i) in a linear combination fashion as expressed by Equation 2.5.

$$z_i = b_i + \sum_{j=1}^{n}(w_j x_j),\ i = 1, ..., m \tag{2.5}$$

In Equation 2.5, m refers to the number of neurons in the layer, while n to the number of inputs for each neuron. The output of this linear combination, denoted as z, is then passed through the activation function to obtain the neuron's final output. Various activation functions are employed in the literature, with some common examples being *tanh*, *sigmoid* and *ReLu* as reported in [115]. The significance of the activation function lies in its ability, typically as a nonlinear function, to impart nonlinearity to the entire NN model. Specifically, a NN using solely linear activation functions would effectively be a linear combination. Conversely, employing a nonlinear activation function results in a nonlinear model capable of mapping specific inputs to their corresponding outputs. The process used to adjust the various weights and biases within the model is known as learning. Learning can take the form of supervised, unsupervised, or semi-supervised learning. Supervised learning occurs when the NN learns to predict outputs from certain inputs based on labelled data provided by the designer. In unsupervised learning, where data are not labelled, the model must learn the correlation between input and output. For a more comprehensive exploration of NNs and DNNs, please refer to the book by Goodfellow et al. [115].

Various applications of NNs in control can be found in the literature. An extensive collection of early applications is presented in [116]. These control schemes primarily involved feedforward NNs and Radial Basis Function (RBF) networks, deployed either alone, within an MPC framework, or hybridized with other AI techniques like FL. Given the absence of online learning, most of these applications do not fall under the category of IC. More recent studies have explored the applications of different NN architectures in an IC setting. Johnson et al. [106, 98] introduced a control scheme called "pseudo-control hedging" that incorporates a NN adaptive control architecture. In this approach, the NN serves as the adaptive element online, attempting to correct errors in the approximate dynamic inversion. Another instance is found in the work of Xu et al. [117], where a RBF NN is employed to approximate uncertainties and is updated online to accommodate dynamic uncertainties. In [118], a robust neuro-adaptive control system is developed for the reentry phase of a RLV. The controller, designed following the dynamic inversion methodology, is augmented with a barrier Lyapunov function-based neuro-adaptive controller for the inner loop. This intelligent approach demonstrates the capability to track guidance commands with good robustness characteristics, as the controller is updated online according to the designed update rule.

2.6.3 Evolutionary Computation

A family of AI techniques less frequently used for control applications is EC. The field of EC includes Evolutionary Algorithms (EAs), GAs, and GPs. Despite their different characteristics in design, they all rely on the concept of biological evolution, evolving a population of individuals according to the principle of the survival of the fittest to obtain a solution. These algorithms are often computationally expensive, making their online usage challenging. Nevertheless, there are some online applications, such as in [119], where EAs are used for online parameter tuning. However, instances of IC applications using an EA for a RLV were not found in the literature. This scarcity of applications contributed to the selection of GP as the technique of interest, as will be explained more thoroughly in the subsequent sections of this thesis. A more detailed description of GP is provided in chapter 4, alongside a discussion on control applications

not involving RLVs.

2.6.4 Hybrids

The presented techniques exhibit diverse strengths and weaknesses. Leveraging these characteristics, they can be synergically combined to enhance the specific advantages provided by each technique. These are denoted as "Hybrid" methods. For instance, NNs are frequently used to approximate the membership functions of a FL controller, while GP can be employed to optimize both the topology and weights of a NN controller.

2.6.5 Limitations of Intelligent and AI-based Control

AI-based and intelligent control systems hold great potential for transforming the space industry, by enhancing autonomy and robustness. Nonetheless, these systems confront a set of challenges that must be addressed for their effective application in the industry. The two most prominent ones in the context of AI applied to G&C are analyzed in the following: interpretability and explainability, and stability assessment

Interpretability and Explainability Foremost among these challenges is the issue of trustworthiness. To improve trust in AI systems, two key features must be considered: interpretability and explainability. Interpretability refers to the knowledge of the inner workings of an AI model, while explainability refers to the possibility of explaining the decisions made by the model. Expainable Artificial Intelligence (XAI), a branch of AI research, aims at addressing this concern by developing tools and procedures that clarify the reasoning behind the outputs of AI models, thereby enhancing human trust. In the aerospace industry, Sutthithatip et al. [120] provide insights into the implementation of XAI. They present a framework outlining how XAI could be integrated into the aviation sector, emphasizing its potential to enhance operational safety and support decision-making processes. While acknowledging the potential benefits, the authors caution that current explanation methods are not yet mature enough for widespread industrial use. Mandrake et al. [121] contribute to the discussion on XAI by applying a framework developed by The Aerospace Corporation to two NASA Jet

Propulsion Laboratory (JPL) projects: the Machine Learning-based Analytics for Automated Rover Systems (MAARS) and Ocean Worlds Life Surveyor (OWLS) missions. The authors report that JPL found the framework to be a practical and systematic structure that fosters valuable introspection into design principles, promoting trust and mission success. This includes specific guidance for AI-based autonomy, distinct from conventional research and development practices for non-autonomy-focused AI. Among AI techniques GP emerges as an interesting solution to tackle the interpretability and explainability challenge. Leveraging its intrinsic interpretability, GP models can be employed in safety-critical applications, offering transparent insights into the inner workings of AI models. However, for GP to be effective in this context, it must be configured to generate meaningful mathematical models that users can interpret and that are related to the specific problems being addressed. The literature extensively covers research on enhancing interpretability and explainability through GP, and interested readers are directed to the comprehensive survey conducted by Mei et al. [122]. In their analysis of nearly 300 papers, the authors highlight the considerable potential that GP holds in advancing the XAI field. This survey provides valuable insights into the various approaches and applications of GP, emphasizing its role in contributing to the transparency, interpretability and explainability of AI models.

Stability Analysis The second prominent issue with AI-based and IC approaches is the difficulty in proving their stability. Stability is an important feature of control schemes used to prove that given a bounded input, the control scheme will produce a bounded output, hence proving the predictability of the control scheme. Traditional stability analysis approaches for linear systems involve the formulation of the closed loop transfer function which can be analyzed using approaches such as the Routh–Hurwitz stability criterion and the Nyquist criterion. A description of traditional approaches can be found in [123]. For nonlinear systems, one of the most applied approaches is the Lyapunov stability analysis, as described in [124].

Traditional approaches may result in being impractical for AI-based and IC schemes, due to the complex and nonlinear nature of the models, their data-driven character, and

lack of well-established theoretical foundations. Moreover, the complex structures of some techniques such DNN architectures, coupled with their black-box nature, hinder traditional stability analysis methods designed for linear systems.

Nevertheless, substantial research has been conducted in recent years on stability analysis approaches applied to AI-based and IC schemes, primarily constructed using NN, DNN, and Reinforcement Learning (RL). Emami et al. [125] present a comprehensive review of NN-based flight control systems. Their review reveals that the majority of stability analysis approaches involve delineating a region of stability, with boundaries determined by the upper limits of a set of parameters. Unfortunately, these parameters often lack physical significance or measurability, posing a significant challenge when implementing adaptive controllers in practical applications. Determining the controllability region of the system becomes particularly problematic in model-free control systems. Thus, there is a pressing need to establish a set of tangible criteria for analyzing closed-loop stability in order to address these challenges effectively.

Regarding FL, Lam [126] conducts a thorough review of stability analysis techniques applied to continuous-time fuzzy-model-based (FMB) control systems. The author compares two distinct approaches for stability analysis in FL control systems: Membership-Function-Independent (MFI) stability analysis and Membership-Function-Dependent (MFD) stability analysis. In the MFI technique, membership function information is not considered. On the other hand, in the MFD stability analysis, stability conditions incorporate information from membership functions. Despite the widespread use of MFI stability analysis in the literature due to its simplicity (fewer stability conditions and decision variables), Lam concludes that MFD stability analysis exhibits greater potential to reduce conservativeness. The MFD approach not only offers a more relaxed stability analysis but also provides fundamental technical and theoretical support for the advancement of FMB control and its diverse applications.

In the realm of EC, specifically focusing on GP, approaches have been devised to establish stability through Lyapunov stability analysis. Grosman et al. [127] demonstrate the application of GP to automatically generate Lyapunov functions suitable for the stability analysis of nonlinear systems. Their methodology identifies the candidate

Lyapunov function with the largest connected set, defining a sub-domain of attraction that includes the origin. Consequently, this approach facilitates the estimation of domains of attraction and can be adapted for automated control system synthesis by appropriately defining the objective function to be optimized. Building upon Grosman et al.'s work, Ali et al. [13] employ a multi-objective GP to concurrently design both control and Lyapunov functions. This approach is applied to stabilize two unstable nonlinear systems, showcasing its efficacy in achieving stability. The authors emphasize the superior capability of GP to optimize not only the controller parameters but also the controller structure, distinguishing it from standard multi-objective optimization algorithms that focus solely on optimizing predefined controller parameters. Additionally, by incorporating the region of attraction into the GP fitness function, this approach provides the flexibility to control the stability margin of the controller. The authors conclude that this integration enhances the adaptability and performance of the GP-based control system synthesis.

2.7 Issues and Research Gaps

In the preceding sections, a high-level overview of four control branches was presented: Optimal Control, Robust Control, Adaptive Control, and a brief mention of AI-based and Intelligent Control, which will be more comprehensively addressed in the next chapter. The theoretical foundations of these control families were outlined, and recent applications to RLVs were discussed. The resulting landscape is summarized in Table 2.1, where the four control branches are classified based on key properties: their applicability to nonlinear systems, online adaptation capability, robustness against disturbances, optimality, ability to leverage online data, and the availability of mathematical proofs of stability.

In Table 2.1, the symbol ✓ indicates that the control family satisfies the desired property, while $\sim$ denotes fulfilment under restricted assumptions. This qualitative analysis is based on the general characteristics of each control family and does not aim to cover all specific control approaches developed in the literature. It is important to note that specific control methodologies may exist in the literature that successfully

Table 2.1: Summary of comparison between main control branches

	Nonlinear	Online Adaptation	Disturbance Rejection	Optimal	Can Exploit Data	Proof of Stability
Optimal Control	∼	∼		✓	∼	∼
Robust Control	∼		✓	✓		✓
Adaptive Control	✓	✓	✓		✓	∼
Intelligent Control	✓	✓	✓	∼	✓	∼

address challenges common to their respective control families.

Optimal control stands out as the preferred family of control schemes for designing controllers optimized to address specific tasks. It can leverage data gathered online when employed in MPC schemes. In such cases, new optimizations are carried out online, incorporating updated plant and environmental conditions, allowing for real-time adaptability. However, the extent of this adaptability is constrained by the complexity of the treated system, as reflected in the computational time required for online optimization. Despite these positive attributes, optimal control methodologies can be fully exploited, and stability proofs can be derived only when applied to linear systems, such as in the case of LQR controllers or convexified systems in the context of convex optimization. While optimal control techniques can also be extended to nonlinear systems through NLP methods, providing a mathematical proof of stability in such cases may be unfeasible.

Robust control approaches focus primarily on enhancing the resilience of the plant against disturbances. These techniques have demonstrated successful applications across various domains, and mathematical proofs of stability can be rigorously derived. Robust control methodologies, including the widely used H_∞ control, can be optimized based on specific metrics, providing a systematic way to handle disturbances. However, it's important to note that the optimality and stability proofs are typically established for linear or linearized systems. When it comes to nonlinear systems, SMC can be used within the robust control framework. It is worth highlighting that SMC demands a deep understanding of the treated plant and necessitates significant design efforts from the user. A notable limitation of robust control approaches is their lack of adaptability online to unforeseen disturbances. These approaches are typically designed offline and

are effective within the predefined range of disturbances established at the time of design. As a result, while robust control provides stability and resilience against known disturbances, its applicability to unanticipated or evolving disturbances is limited.

Adaptive control approaches stand out as the primary group of techniques designed to create controllers with the capability for online adaptation. This adaptability is crucial for systems operating across a broad range of conditions, particularly those subject to significant disturbances and uncertainties. Adaptive mechanisms can be integrated into various control schemes, allowing for flexibility in their application. For example, a robust controller can be enhanced with adaptive features. Several adaptation mechanisms have been developed over the years to address the dynamic nature of different systems. It's important to note that the design of a specific adaptation mechanism is problem-dependent, and it is difficult to mathematically assess its stability when applied to nonlinear systems. Moreover, adaptive control approaches are not inherently optimal. Despite these limitations, they represent the control approach of choice for complex and nonlinear systems, allowing them to extend their operating range and enhance resilience against disturbances. The adaptability of these controllers makes them particularly well-suited for applications characterized by varying conditions and uncertainties.

To address the limitations observed in the control groups previously described and to enhance control system capabilities in handling even more substantial uncertainties, IC was developed. IC techniques represent a progression beyond adaptive control by integrating AI methods into their design. The influence of AI techniques is evident across various domains, showcasing their exceptional capabilities in data analysis, correlation identification, classification tasks, and predictive modelling of future behaviours. These techniques find application in managing nonlinear systems and possess a certain degree of optimality, as many AI methods inherently involve optimization routines. Nevertheless, IC approaches come with notable drawbacks, primarily the challenge of comprehending the inner workings and decision processes of these algorithms. Moreover, the absence of an established framework for systematically analyzing the stability of these systems represents another significant limitation. As outlined in section 2.6.5, existing

approaches in the literature often tend to be tailored to specific applications, lacking a generalized framework that can be universally applied across diverse IC scenarios. This lack of a standardized stability analysis framework adds a layer of complexity and uncertainty, hindering the broader adoption of IC techniques, particularly in safety-critical applications where stability assurance is paramount.

Considering these factors, this thesis was developed to advance current knowledge on IC and its application to RLVs, which have been relatively underexplored in this domain. Following a comprehensive literature review on IC, resulting in the production of two technical reports [14, 15], as well as two papers [19, 1], GP was selected as the AI technique of interest. This choice is motivated by several compelling reasons. Firstly, GP stands out for its capacity to autonomously generate control laws for nonlinear systems, offering a versatile and adaptive approach suitable for the complexities often found in control applications. Secondly, its interpretability is a key feature, enabling users to comprehend and explain the generated control strategies. This transparency addresses the challenge associated with the "black box" nature of many IC techniques, contributing to a better understanding and trust in the generated control systems. Additionally, GP provides a unique advantage in that the stability of the generated control scheme can be proven through a Lyapunov stability analysis. This mathematical foundation for stability is crucial in safety-critical applications, to instill confidence in the reliability of the control system. Furthermore, the decision to focus on GP is reinforced by the observation made in section 2.6 that very few IC applications involving GP can be found in the literature, and none specifically on RLVs. This underscores the novelty and potential impact of exploring GP in the domain of IC for RLVs, offering an opportunity to contribute with new insights and solutions to this relatively unexplored area.

2.8 Summary and Comments

This chapter is meant to provide a background on control theory as applied to RLVs. The historical development of the three primary branches of control theory - Optimal Control, Robust Control, and Adaptive Control - is presented to clarify the evolving

need for increasingly sophisticated control schemes capable of handling greater uncertainties. The journey progresses from the introduction of optimal control, aimed at optimizing control laws, to the advent of robust control, addressing disturbances within defined bounds. The subsequent development of adaptive control allows for online adaptation to face unbounded or significant disturbances. The latest stage in this evolutionary process is IC, which integrates an AI techniques into control schemes to extend the range of treatable uncertainties. This control paradigm has become particularly relevant in recent years due to advancements in hardware technologies, facilitating the development and proliferation of AI techniques.

The literature review provides a high-level description and theoretical insight into the main techniques within these control branches, highlighting the issues and gaps in each control family. The discussion is enriched with a list of applications related to RLVs and hypersonic vehicles—central themes of this thesis.

The chapter concludes with a comparison of the analyzed control families, where IC emerges as a solution to handle increasingly greater uncertainties. This approach promises greater flexibility, robustness, and the ability for online learning through real-time data. However, the challenge remains in overcoming the black box nature of many AI models, emphasizing the need for enhanced explainability to foster trust and reliability in IC; and in improving the stability analysis framework for AI-based control schemes.

Chapter 3

Intelligent Control [1]

This chapter presents a comprehensive literature review on Intelligent Control (IC) to provide the reader with the necessary background to understand the rest of this thesis. The chapter concludes with a novel taxonomy to classify IC applications. It was designed during the development of the research presented in this thesis and was initially introduced in [19].

3.1 Introduction

The increasing complexity of control problems in the past century has made necessary the design of more advanced control methods. Current control systems are often required to work in challenging environments with limited preexisting available knowledge. To overcome this issue, Artificial Intelligence (AI) is introduced into control theory, and its combination with the disciplines of automatic control and operations research is defined as "Intelligent Control" [128].

IC is a topic widely discussed in the literature since the term was coined by Fu [9]. IC applications are distributed among various engineering fields and are mainly concentrated in those that seek to minimise human intervention, such as robotics. Nonetheless, the IC term was often misused, leading to increasing confusion, especially when comparing IC with similar but different concepts such as adaptive and learning control

[1] The content of this chapter was previously published in the technical reports [14, 15], in the conference paper [19] and in the journal paper [1].

[129]. Nowadays, the concepts related to IC are defined more precisely. The taxonomy presented in section 3.3 is defined according to the work of Saridis and Antsaklis, which both gave clear definitions of IC [128, 130]. Despite a more precise definition of IC, many different types of control systems with varying levels of complexity can still be classified as IC. The taxonomy presented in section 3.3 aims to quantify the level of intelligence of an IC system according to the differences and similarities among control systems. This approach allows a comparison between different control methods and applications and highlights gaps that can be filled by further research on IC systems. As Antsaklis [129] pointed out, it is not trivial to define the level of intelligence of a machine. It cannot be defined in a binary way but rather in a more detailed manner, from non-intelligent to highly intelligent. However, the problem of quantifying the level of intelligence arises.

If some aspects of a system are unknown due to stochastic behaviour or a lack of knowledge, that system is characterised by a certain degree of uncertainty. Uncertainty treatment has always been a critical challenge in control theory. In the past decades, much work has been done on control systems to make them more robust against uncertainty. Great effort was also put into increasing their autonomy in facing uncertainties, up to the point where some of them were required to learn from the environment, showing human-like behaviour. This can be considered intelligence [131]. According to this, it can be stated that the greater the level of uncertainty at design time that a control system can handle, the greater its level of intelligence.

This chapter begins by defining IC in section 3.2 with the dimensions in which an intelligent controller operates and the methods used. Section 3.3 presents the novel taxonomy produced during this thesis's development. Section 3.4 contains a list of IC examples classified according to the proposed taxonomy. Section 3.5 concludes the chapter with a summary and final remarks.

3.2 Defining Intelligent Control

After its definition, the term "Intelligent Control" became highly used and often abused by the academic community, both in and outside the control field. This widespread usage made it difficult to create a definition of IC suitable to every different IC application. Because of this, a task force was designated in 1993 by the IEEE Control Systems Society to research and define "Intelligent Control" [130]. The outcome of their investigation led to the following defining characteristics of an IC system:

"An intelligent control system is designed so that it can autonomously achieve a high level goal, while its components, control goals, plant models and control laws are not completely defined, either because they were not known at the design time or because they changed unexpectedly."

This definition shows how an intelligent controller can be defined as such not only if it is capable of dealing with system uncertainties but also if its structure and goals are not entirely defined. Before this definition, Saridis described IC as an interaction between the fields of Artificial Intelligence, Operations Research, and Automatic Control Systems (Figure 3.1) [128]. The definition given by Saridis is in line with the one initially given by Fu, who defined IC as the "intersection of artificial intelligence and automatic control" [9].

According to these definitions, it is clear how a control system can be defined intelligent if it possesses some online learning/adaptation capability and incorporates an AI technique. A common misunderstanding arises when considering adaptive or learning control systems and AI-based control systems separately. An adaptive system can be defined intelligent if it is built using an AI technique and not classical analytical formulations, while an AI-based control system can be defined intelligent if it possesses an online adaptation/learning mechanism. For example, a control system that uses AI to define its control structure offline, without updating any of its components online, cannot be considered intelligent since it cannot cope with substantial environmental uncertainties.

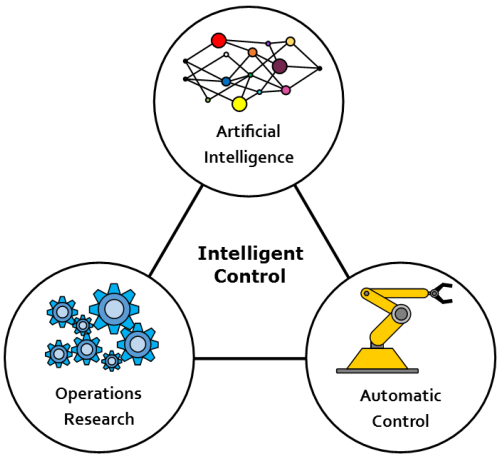


Figure 3.1: Intelligent control is the interaction of the fields of artificial intelligence, operations research, and automatic control. [1]

The AI techniques mostly used to design IC schemes belong to the three families described in section 2.6, namely Fuzzy Logic (FL), Machine Learning (ML) and Evolutionary Computing (EC). A description of these is provided in section 2.6, with a focus on Neural Networks (NNs). A detailed description of Genetic Programming (GP) can be found in chapter 4.

3.2.1 Dimensions of Intelligent Control

As previously stated in section 3.2, the main characteristic of an IC system is its ability to deal with substantial uncertainties. Therefore, it is reasonable to define the level of intelligence of an intelligent controller according to the level of uncertainty it has to face. From the task force definition of IC, it emerged that the uncertainty could be present in three dimensions: the environment (under which both plant and environment models are considered), the controller laws and components, and the control goals. From a more abstract point of view, this represents *what* is being controlled, *how* it is being controlled, and *why* it is being controlled.

Environment

An environment is considered known if a mathematical model is available. With the environment term, both the plant model and the environment model in which the plant

has to operate are considered. Such a model is necessary for the design of the control system, and the degree of knowledge of the environment models at design time affects the intelligence of the controller. A general controllable nonlinear system is shown in Equation (3.1), where y represents the output, u the input, x represents the system's state variables, and the functions f and h are mappings (linear or nonlinear) from their inputs to an appropriately dimensioned vectors.

$$\dot{x} = f(x, u) \tag{3.1a}$$

$$y = h(x) \tag{3.1b}$$

For clarity, in the rest of this chapter, only expressions for $\dot{x}$ will be considered. Also the environment model may contain time dependent parameters $A = \{a_1, a_2, \ldots, a_{na}\}$. In this case, Equation 3.1 is no longer valid, and a time dependant mapping must be used, as shown in Equation 3.2.

$$\dot{x} = f(x, u, A(t)) \tag{3.2}$$

The actual environment does not precisely match its mathematical models in real-world applications due to modelling inaccuracies or uncertainties. Considering this more realistic scenario, Equation 3.2 must be updated into Equation 3.3, where $\hat{f}$ represents an uncertain mapping.

$$\dot{x} = \hat{f}(x, u, A(t)) \tag{3.3}$$

Controller

As for the environment, also the controller can be mathematically designed according to different levels of knowledge about its components. The more intelligent the controller

is, the less precise the knowledge of its control law at design time, alongside an overall greater flexibility.

A general feedback controller can be defined as in Equation 3.4, where $e = y_d - y$ is the tracking error between the desired system output y_d, and the actual system output y.

$$u = g\,(e) \tag{3.4}$$

This type of controller has fixed parameters that are defined at design time. On the other hand, a general adaptive controller can be defined as in Equation 3.5, where $K = \{k_1, k_2, \ldots, k_{nk}\}$ are the control parameters that can vary according to variations in the operating conditions.

$$u = g\,(e, K(\cdot)) \tag{3.5}$$

So far, perfect control systems were considered, i.e. no uncertainties were taken into account. Nonetheless, significant uncertainties can affect the tracking error e, even if the environment is stationary and deterministic. These uncertainties can be due to sensor disturbances or unknown actuator dynamics. Such a controller can be defined as in Equation 3.6, where $\hat{e} = y_d - \hat{y}$ is the uncertain tracking error given the measured and uncertain system output $\hat{y}$.

$$u = g\,(\hat{e}, K(\cdot)) \tag{3.6}$$

The control parameters in Equation 3.6 will be tuned accordingly to the applied uncertainty and change in the operating conditions. In addition to the control parameters, the controller structure could also be adapted to face uncertainties. A variable structure could be achieved by considering multiple control laws simultaneously and choosing the more suited to the current observations, or new control laws could be derived online. Equation 3.7 defines a general example of such a controller.

$$u = \begin{cases} g_1\left(\hat{e}, K_1(\cdot)\right) \\ g_2\left(\hat{e}, K_2(\cdot)\right) \\ \vdots \\ g_{ng}\left(\hat{e}, K_{nk}(\cdot)\right) \end{cases} \tag{3.7}$$

Goals

Defining the dimension of the goals is a more arduous task than it was for the previous two dimensions. Goals are more abstract; hence, it is more challenging to define them mathematically. Consequently, knowledge of goals could be defined according to how well they can be described mathematically and how aware the controller is of its goals. Often the goals are determined following a stability criterion or by tracking a desired performance measure. In this scenario, the goal is defined at design time, and the controller is unaware of its purpose.

The goals can also be defined through a cost function that indirectly gives the controller information about its performance in executing a task. The controller will then try to minimise (or maximise) such a cost function according to its control policy. In this case, the controller now possesses a low level of awareness since it tries to find a way to achieve the goal by minimising (or maximising) the cost function.

A higher degree of intelligence is required in those systems where the goal cannot be defined mathematically but only in a high-level language. Such an intelligent system must understand how to act to achieve the sought high-level goals. This can be done, for example, by introducing a series of short-term goals that may change over time according to the controller's internal planning but always considering the required global goal.

3.3 Taxonomy

The taxonomy presented in this section is an original contribution of this thesis. Such a taxonomy aims to classify the level of intelligence of control systems. One of the main innovations of the proposed taxonomy is that it considers IC as a multi-dimensional dis-

cipline in the sense that a control system can show intelligence in different dimensions. Other previous works attempted to classify the levels of intelligence in IC systems. Krishnakumar [132] proposed a classification based on four levels of intelligence based on the controller's ability to self-improve, but it does not consider uncertainties treatment. Moreover, each level of intelligence is additive to the previous one. On the contrary, the taxonomy proposed in this research shows that intelligence levels can differ in each dimension. The dimensions are independent, and therefore the levels of intelligence are not additive. Another classification was presented in an industry survey from the American Institute of Aeronautics and Astronautics. Six "stages of intelligent reasoning" for spacecrafts [133] were defined in this work, which was only specific for spacecraft operations. Conversely, the taxonomy proposed in this thesis can be applied to any IC system.

As previously mentioned in section 3.2, IC approaches are employed where there is a lack of knowledge at design time. Such a lack of knowledge can be encountered in three main categories: the environment, the controller, and the goals. Any controller within these categories, also conventional ones, can possess a different level of knowledge at design time. This section presents a classification scheme for IC methods based on the level of knowledge present in the control system at design time. For all three categories, level 4, which is the highest level of uncertainty, represents a hypothetical maximum uncertainty.

3.3.1 Environment Knowledge

0. *Complete and precise environment model:* Suppose the environment is completely known, and an equation captures all the dynamics in the form of Equation 3.1. In that case, an open loop controller could be used without needing any degree of intelligence. This is an ideal scenario, and usually, this approach cannot be used in real applications due to uncertainties or unknown aspects of the considered system. Therefore, the need for more sophisticated controllers.

1. *Complete environment model subject to minor variations:* As said above, real systems can only be modelled to a certain degree of precision. For this level

of environment knowledge, only bounded uncertainties are considered, which can be tackled by a feedback controller without needing adaptation. This absent or small need for adaptation leads to controllers which are not necessarily intelligent. Nonetheless, there are still some examples of intelligent controllers in this category.

2. *Environment subject to change during operation:* As described by Equation 3.2, the environment at this level is characterized by time-varying parameters. Since it may not be possible to predict these greater changes in the environment, or they may be too complex to model, a higher level of intelligence is required. Some conventional adaptive controllers and intelligent ones can perform well at this level of uncertainty.

3. *Underlying physics of environment not well defined:* Although it represents an uncommon scenario for Earth applications, an environment could be described as an uncertain mapping from states and actions to future states as in Equation 3.3. This is the case for many space applications, such as Mars entry vehicles. In this scenario, some information about the environment is known, but an intelligent controller must account for substantial knowledge gaps.

4. *No knowledge of environment:* No model of the environment is available or exists; hence no environment knowledge is incorporated into the control system at design time. An intelligent controller is required to explore such an environment safely.

3.3.2 Controller Knowledge

0. *Stationary, globally stable controller:* Feedback controllers in the form of Equation 3.4 can maintain appropriate performances under the given assumption, and their stability can be mathematically proven. A feedback control system can perform well for simple applications without needing adaptation.

1. *Varying controller parameters:* Suppose a controller with fixed parameters at design time cannot perform efficiently in the entire operating range of the system. In that case, its parameters can be varied online to extend its operational range.

Equation 3.5 describes a controller of this kind. Many examples of intelligent and non-intelligent applications can be found at this knowledge level.

2. *Unknown sensor/actuator behaviour:* This level of controller knowledge comprehends the broader category of Fault Tolerant Control (FTC). In this context, FTC represents an uncertainty in the controller whose action might be different than the expected response and where sensors may provide erroneous measurements. A general definition of this controller is provided in Equation 3.6. This category does not comprehend FTC systems that use simple thresholds to identify faults that are defined at design time since they are known. Therefore this category is for those control systems that must deal with unknown faults.

3. *Varying controller configurations:* With the increasing level of intelligence, a controller can alter its structure online to adapt to unforeseen uncertainties. Such a controller is generally defined as in Equation 3.7. Often, techniques such as evolutionary computation are used to determine the controller structure offline. Applying these techniques online would make the control system intelligent and able to adjust its configuration while operating.

4. *No known controller structure:* The maximum level of intelligence in this category is represented by a controller that can design itself online from scratch, using, for example, control blocks, mathematical operations and intelligent architectures. To improve the online efficiency of this approach, an initial rudimentary controller can be given as a stable starting point.

3.3.3 Goal Knowledge

0. *Goals entirely predetermined by designer:* Most control systems, including intelligent ones, are designed to satisfy a defined goal. In this scenario, the control system has no awareness of its goals hence it is not able to change the current goals to better adapt to the encountered environment or uncertainties. Examples of such control systems are those where the tracking error must be reduced to zero.

1. *Goal specified implicitly, for example, as a reward function:* Classical optimal control problems pertain to this category. These control systems aim to minimize or maximize a defined objective function. This results in a high-level goal of the controller consisting of deriving an optimal control policy with respect to the objective function. This is also the case for the reinforcement learning control framework, where an agent learns a control policy by interacting with the environment. In this latter control scheme, the controller learns by observing its state and reward.

2. *Specific goals subject to change during operation with a globally defined goal:* Systems acting in highly dynamic environments require an intelligent goal planner to adapt online to significant environmental changes. For example, in a space mission, the spacecraft/rover has to wait for new instructions every time an unforeseen event occurs, or new data are available. These instructions are sent from the engineers on Earth. Hence a time delay is introduced, which could affect the mission's success. This delay could be avoided by an intelligent goal planner that defines online new goals according to the current environment.

3. *One or several abstract goals with no clear cost function:* In some cases, it might not be possible to define the goals mathematically; hence the controller must be able to understand high-level goals. For example, a high-level goal could be "capture images of scientifically interesting events" or "explore this region and collect data". The controller must understand and decide autonomously what events are scientifically interesting or which data are worth collecting.

4. *No knowledge of goals:* Controllers in this category have to deduce autonomously what actions to take when, to begin with, they have no knowledge or indication of what actions are favourable.

3.4 Classification of Relevant Examples

To illustrate the applicability of the taxonomy presented in the previous section, some examples of IC systems are described and classified, covering a broad range of methods

and levels of intelligence. The following notation is used in the classification below:

- G: Goal Knowledge
- E: Environmental Knowledge
- C: Controller Knowledge

Table 3.1 shows the classification of IC approaches found in the literature with their respective references. They are grouped according to the particular AI technique employed. For clarity, only the levels of classification found in the reviewed applications are listed in the table. Figure 3.2 presents the same classification in a parallel coordinate plot, where the applications are divided into different levels of intelligence, and the colours and line thicknesses indicate the number of applications found in the respective level of intelligence. The additional dimension regarding the publication year was added to show the spread of dates over which these were published. From Figure 3.2, it can be observed that: 1) there are no applications with a goal knowledge greater than two; 2) the most common classification is G-0, E-1, C-1 (20 applications) by a significant margin and the second most populated intelligence level is G-0, E-2, C-1 (9 applications); 3) the reviewed works were published in a well-spread manner between 1991 to 2019.

As previously discussed in section **??**, several AI techniques can be used for IC, although the majority of them can be classified either as Machine Learning (ML), Fuzzy Logic (FL), Evolutionary Computing (EC) or hybrid methods. The distribution of these methods is quantitatively depicted in Figure 3.3 where it can be observed that the majority of the reviewed applications employ NNs (51%) covering a broad range of intelligence levels as shown by Table 3.1. EC and FL are employed on a similar number of applications but with different levels of goal knowledge. Usually, EC is used for symbolic regression or optimisation applications and is often coupled with other AI techniques. While FL is used where human-like reasoning is required, thus leading to a higher concentration of FL applications in higher levels of intelligence, e.g. G-2. Hybrid methods are also widely used (24%), and among them, the most common is the combination NN and FL.

Table 3.1: Artificial intelligence techniques used for intelligent control applications. [1]

	G0							G1						G2		
	E0	E1			E2		E3	E0	E1		E2		E3	E1		E2
	C2	C1	C2	C3	C1	C2	C1	C0	C1	C4	C1	C2	C4	C0	C1	C1
FL		[134, 135]									[136]				[137]	
NN		[138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150]		[151]	[152, 153, 154, 155, 156]	[157]	[158]		[159]		[106, 160, 161]	[162]				
SVM		[163, 164]														
EC								[165]	[166, 167]	[168]			[169]			
Other														[170]	[171]	
Hybrid Methods	[172]	[173, 174]	[175]		[176, 177, 178, 179]				[180, 181]					[182]		[183]

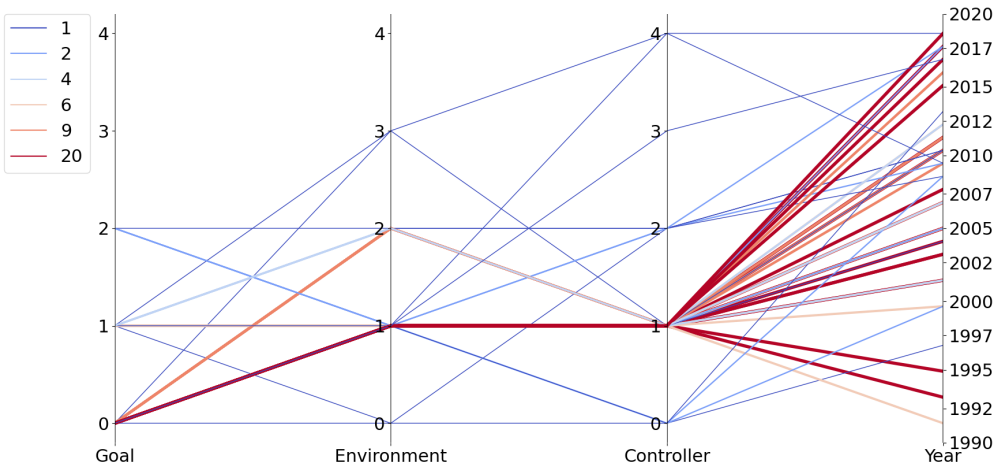


Figure 3.2: Parallel coordinate plot of the observed levels of intelligence. The colour scale and the different line thickness refer to the number of applications observed in the considered intelligence level. [1]

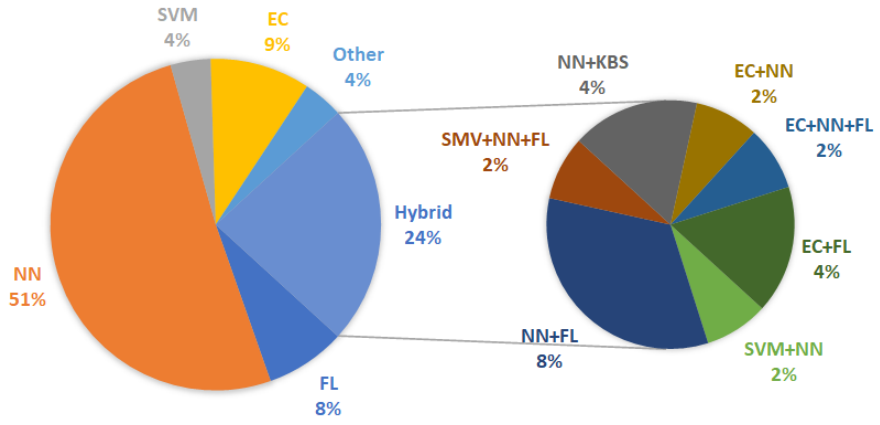


Figure 3.3: AI methods used for Intelligent Control (IC). [1]

In the following, a brief description of an application for each observed level of intelligence is provided. The reviewed applications are from different engineering domains that comprehend a robotic element, meaning that the considered systems can achieve their goal without human intervention.

- *G-0, E-0, C-2:* Kankar et al. [172] present a comparison between NN and Support Vector Machine (SVM) in predicting ball bearing failures showing how both techniques can be useful for this application. Although the presented application is not a complete controller, it is a fault detection system that can be integrated into a controller for a rotating machine.

- *G-0, E-1, C-1:* An early example of Neural Networks (NNs) being used as direct controllers is described by Ichikawa and Sawa [174]. In their work, a direct NN controller is combined with a genetic model reference adaptive control, which trains the NN considering the model of the ideal plant dynamics. This system is designed to deal with evolving environment dynamics, and the network is continually updated to optimise performance.

- *G-0, E-1, C-2:* A widespread technology for IC and in particular Fault, Detection, Isolation and Recovery (FDIR), is the Adaptive Network-based Fuzzy Inference

System (ANFIS) developed by Jang [184]. An application of ANFIS is presented by Wang et al. [185]. In their work, an adaptive backstepping sliding mode controller is augmented with an ANFIS FDIR system that can control a robotic airship. The environment states are predicted at each time step by an ANFIS observer. If these values disagree with those from the sensors, a sensor fault is declared, and the ANFIS output is used as input to the controller. The level of Goal knowledge is 0 since the goal of the control system is to minimise a tracking error following a predetermined trajectory.

- *G-0, E-1, C-3:* The NN controller designed by Wu et al. [151] is classified as C-3 since it possesses the unique ability to change the network topology and its parameters online based on the output of a learning algorithm. Such a change in the topology is not trivial and requires a trade-off between maintaining sufficient computational speed for online usage and the required precision in its output values.

- *G-0, E-2, C-1:* The neuro-fuzzy controller is among the more popular IC methods. It is designed by combining the adaptability of a NN with the ability to replicate the human-like reasoning of fuzzy controllers. In [178], the authors apply a neuro-fuzzy model reference adaptive control scheme to an electric drive system. Their results show that the controller is robust to environmental changes and adapts quickly to suppress vibrations and improve tracking accuracy.

- *G-0, E-2, C-2:* Xu et al. [157] present another example of FDIR incorporated into control systems. In particular, unknown faults are recognized using a FTC scheme based on a backstepping controller integrated with a NN, which updates its weights online using a modified back-propagation algorithm. Two networks are used to approximate unknown system faults and compensate for their effect.

- *G-0, E-3, C-1:* Vehicles that operate in uncertain environments, such as Mars entry vehicles, can benefit from having an IC system. In the paper of Li et al. [158], a NN based sliding mode variable structure controller is developed.

Such a controller is composed of a fast loop, a conventional Proportional-Integral-Derivative (PID) controller, and a slow loop containing the adaptive NN element. The user defines the goal by defining a nominal entry trajectory.

- *G-1, E-0, C-0:* The control system presented in [165] is based on a Genetic Algorithm (GA) used to optimise the temperature for ethanol fermentation online. In this application, GA is not used to adapt the controller parameters online, but rather in an optimal control fashion to find the optimal fermentation temperature online and update its goals. In contrast to classical optimal control approaches, the optimisation is performed online according to the plant states and employs an AI technique. These differences make the considered control system intelligent.

- *G-1, E-1, C-1:* The control system designed by Handelman et al. [180] pertains to the class of hybrid methods, which, as discussed in section 3.2, combine different AI techniques and exploit their strengths. The presented control system comprises a Knowledge Based System (KBS) for devising learning strategies and a NN controller, which learns the desired actions and performs these in real time. This control system is designed to mimic human learning, combining a rule-based initial learning and fine-tuning by repetitive learning. The environment and controller considered here have low levels of uncertainty, and the control goals are only implicitly defined.

- *G-1, E-1, C-4:* Despite its good performances in symbolic regression applications, GP is rarely used for IC applications due to its cumbersome computational cost. Chiang [168] presents a control system that can be classified as C-4 since the control law is created online using GP starting from predefined mathematical functions without any prior knowledge of the controller structure. Such a controller guides a mobile robot in an environment with known and unknown obstacles, and the latter introduces a slight uncertainty.

- *G-1, E-2, C-1:* Kawana and Yasunobu [136] present an intelligent controller capable of keeping the system performances at the desired level despite actuators'

failure. This example is not classified as G-2 since the introduced failure is known and defined by the user. Nonetheless, the introduced changes in the environment are significant. A particular feature of this control system is its ability to generate the environment model online through learning, and this model is then used to update the fuzzy control rules.

- *G-1, E-2, C-2:* In the work of Talebi et al. [162], two Recurrent Neural Networks (RNNs) are used to detect and isolate faults, one for sensor faults and the other for actuator faults. These NNs are also used to compensate for these faults directly without needing an additional subsystem for fault isolation.

- *G-1, E-3, C-4:* In the control system presented in section 4.3 [169] GP is used to generate online the control, similar to what was done in [168]. The controller is then tested on different failure scenarios, where the environment model is partially unknown at design time.

- *G-2, E-1, C-0:* The controller designed by Ceriotti et al. [170] can modify a planetary rover's goal during its mission. To do so, a value of "interest" is assigned to each point on the map. Such value of "interest" comes from combining navigation data with scientific data from different sources. The values of interest on the map evolve according to the observed data. The combination of data from various sensors is done using the Dezert-Smarandache Theory (DSmT) of plausible and paradoxical reasoning, which can overcome the limitations of fuzzy logic and evidence theory. The values of interest on the map evolve over time according to the observed data.

- *G-2, E-1, C-1:* One of the most advanced IC systems is the Autonomous Sciencecraft Experiment onboard NASA's Earth Observing One [171]. This system possesses a hierarchical structure, where the highest level is the CASPER planner, which plans its activities according to the information coming from the onboard science. The activities plan is then fed to the spacecraft command language, which executes it using lower-level actions. This level can also adapt to environmental

changes and make control adjustments as necessary. Conventional software is used below this level to carry out control actions as instructed by higher levels. While this system does not operate in a significantly changing environment, it can alter its controller parameters online and contains highly autonomous decision-making and goal updating.

- *G-2, E-2, C-1:* The WISDOM control system for rovers is presented in the work of Vasile et al. [183]. Such a controller is capable of adaptive control and high-level planning. It is composed of a hierarchical structure of three layers. The top layer generates plans which are fed to the adaptive controller at the lower layer. The adaptive controller deals with environmental changes and provides instructions for the lowest hierarchical level connected to the actuators. This system adapts to changing or uncertain environments and has varying parameters. The goals are also evolved over time by the system's planner.

The examples listed above are meant to show the applicability of the proposed taxonomy to a broad range of fields and applications involving IC. The taxonomy is consistent in the sense that different applications involving different AI techniques can be grouped in the same classification level according to the performed task. A few examples of this consistency are listed in the following:

- *Activity planning - G-2*: The applications involving planning and reasoning were classified as G-2. Regardless of the employed AI technique, the controller needs to choose its desired states and how to achieve them in all these cases. For example, in [171], the control system uses information from the onboard science subsystem to plan its activities. In [182], a reasoning strategy based on forward chaining is adopted to find optimal concentrations of chemicals for an electrolytic process.

- *Robotic navigation and manipulation - E-2*: Systems that operate in unknown environments or in the presence of unknown obstacles are classified as E-2. This category comprehends control systems that deal with parametric uncertainties in the plant's dynamic models as in [156, 155] and robotic exploration in an uncertain environment [183].

- *Adaptive intelligent control - C-1*: If no unknown faults are considered, then those control systems that adapt their parameters online are classified as C1. This category's most common aspect is some adaptation mechanism to update the control law parameters. Such adaptation can be performed with different AI techniques. From the comparison of two different applications [137] and [149], it can be seen that despite the different AI techniques employed (FL and NN respectively) and their differing overall goal, they are both classified as C-1 since they both adjust control law parameters online.

- *Fault Detection, Isolation, and Recovery (FDIR) - C-2*: This category comprehends those control systems that can deal with failures in their sensors or actuators. ANFIS is a technique used often in this category as in [175] for fault detection and diagnosis of an industrial steam turbine and in [185] where a controller is designed to reliably track the trajectory of a robotic airship in the presence of sensor faults.

3.5 Summary and Comments

This chapter provided a comprehensive description of Intelligent Control (IC). IC was defined alongside a high level description of the different Artificial Intelligence (AI) techniques used for IC. A more detailed description of Genetic Programming (GP) is given in chapter 4.

As a novel contribution of this thesis, a novel taxonomy of IC was proposed. The taxonomy aims to classify control systems based on their level of knowledge at design time in three dimensions: goals, environment and controller. Therefore, the classification is performed according to the level of uncertainty that the control system can handle. Several applications were studied and classified according to the proposed taxonomy to assess its applicability and consistency. From the reviewed applications, it emerges that the majority of IC systems focus on the environment and controller dimensions which tend to have a greater number of highly intelligent applications than the goal dimension. Although recent applications show an increasing development to-

wards a higher level of intelligence in each dimension, there is still work to be done to develop the level of intelligence with respect to goal knowledge.

The reviewed applications showed how IC covers several classes of systems requiring different levels of intelligence. Applications with a lower level of intelligence are the most common, but as technology develops, more autonomous machines will be required; hence the number of highly intelligent applications will increase.

Many future work directions can be pursued. Machines created to explore new or highly uncertain environments, such as disaster rescue robots, will need a controller capable of dealing with higher uncertainty in the goals knowledge. The intersection between IC and explainable AI can be an essential step towards creating trustworthy systems, allowing a human operator to interact with the reasoning process of machines effectively.

Chapter 4

Genetic Programming for G&C and Intelligent Control

4.1 Introduction

This chapter presents Genetic Programming (GP) and describes its application in a control and Intelligent Control (IC) framework. GP, an Evolutionary Algorithm (EA) devised by Koza in 1992 [186], demonstrates proficiency in generating mathematical models for achieving predefined objectives. Its applicability extends to both regression and classification scenarios. In the context of regression problems, the method is labelled as symbolic regression due to the symbolic form of the generated models. The flexibility and interpretability of models produced by GP makes it an interesting alternative for discovering control strategies. In control applications, interpretability proves useful as understanding the control equation facilitates the evaluation of system reliability and behaviour. For instance, in linear systems, the knowledge of the control law expression facilitates the construction of the closed-loop transfer function for subsequent stability analysis [187]. Furthermore, in the context of Artificial Intelligence (AI) applied to control systems, an interpretable control law promotes trust in AI-based control systems by clarifying the connection between input and output [188].

Considering GP applied to control schemes design, noteworthy non-intelligent applications are found in the literature. For instance, in [189], GP is employed to au-

tonomously generate a control Lyapunov function and the modes of a switched state feedback controller. Similarly, in [190], GP is utilized to formulate a PID-based controller resilient to noise. In the aerospace domain, only very few works about GP-based controllers can be found, such as [191] employing multi-objective GP for evolving controllers for a Unmanned Aerial Vehicle (UAV) tasked with locating and navigating around a radar source. Another instance is [192], where an adapted GP approach is employed to generate the control law for a UAV to be recovered onto a ship, accounting for real disturbances and uncertainties. However, these applications do not qualify as IC since the GP controller design was conducted offline, lacking online learning or adaptation.

To design an intelligent controller using solely GP, the control law must be generated or updated online in response to environmental or plant changes. This type of application is extremely rare in the literature, likely due to computational costs. An example is discussed by Chiang [168], who describes the online generation of the control law using GP, to allow a small robot to navigate through a predefined environment with previously unknown obstacles. The scarcity of these types of applications is one of the reasons that justified the research developed during the doctoral work. In fact, due to the rapid development of hardware technologies, issues related to computational costs will become less relevant in the future.

The remainder of this chapter is structured as follows. Section 4.2 discusses the theoretical foundations of GP while its application in an IC settings is presented in section 4.3, where GP is used to produce the real-time ascent guidance commands of a Goddard rocket. In section 4.4, the Inclusive Genetic Programming (IGP), a novel GP heuristic designed during this thesis's development is presented and the results obtained from its application on regression tasks are discussed. Section 4.5 contains a summary of the chapter along with concluding remarks.

4.2 Genetic Programming

Similar to other EAs, GP operates based on the Darwinian principle of the evolution of the fittest. Following this principle, a population of individuals evolves until a specified

goal is achieved. In GP, an individual is represented by a mathematical model, also known as a GP tree, illustrated in Figure 4.1. Here, x, y, and 5 are termed terminal nodes, with x and y representing input variables. The remaining nodes in the GP tree correspond to primitive functions defined by the user and used by the GP algorithm to autonomously construct programs.

At the beginning of the evolutionary process, an initial population of randomly generated programs is created. The user defines a fitness function, either to be minimized or maximized by the evolutionary process. Evolution proceeds through the application of crossover, mutation, and selection operators, aiming at finding the minimum (or maximum) of the specified fitness function.

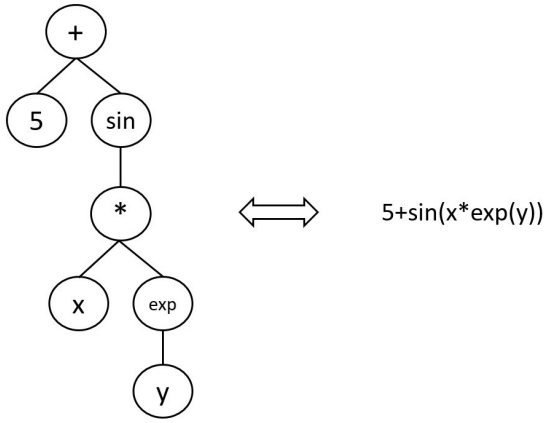


Figure 4.1: Individual structure in GP. The tree can be read as the mathematical model on the right.[2]

Concerning the evolutionary operators, crossover involves the exchange of genes between two parent individuals, resulting in the generation of two offspring individuals. In Figure 4.2, the blue and orange boxes highlight two genes that are exchanged between the parents to create the offspring. Meanwhile, the mutation operator performs the random mutation of a selected gene in an individual to generate a new child with the same structure as the parent, except for the mutated gene, as depicted in Figure 4.3. Lastly, the selection operator is employed to choose individuals from the population, which comprises parents and offspring, to form the starting population for the next generation. The selection is made based on user-defined criteria. The evolutionary

process of a generic GP algorithm is summarized in Algorithm 1.

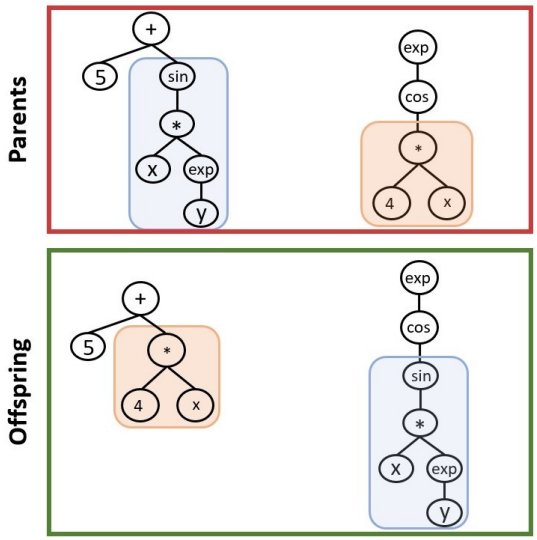


Figure 4.2: Schematic of crossover operation. From two parents, two children are generated.

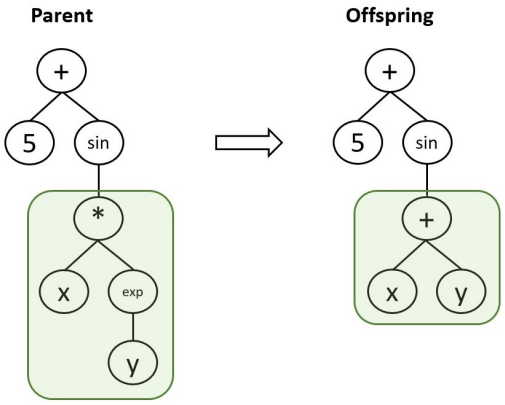


Figure 4.3: Schematic of mutation operation. A selected gene of an individual is mutated randomly.

Numerous variations of crossover, mutation, and selection operators have been developed over the past decades, and a great amount of literature exists on various flavours of GP. This thesis does not intend to offer an exhaustive description of the subtleties of the GP algorithm. Interested readers are directed to [186, 193, 194] for more comprehensive discussions on diverse GP implementations.

Algorithm 1 Generic GP evolutionary process

1: Create initial population randomly
2: Evaluate fitness of individuals in the initial population
3: Generation ← 1
4: **while** Termination criteria is not met **do**
5: Perform crossover and mutation to obtain offspring
6: Evaluate fitness of offspring
7: Select individuals from the old population and offspring to create a new population
8: Generation ← Generation + 1
9: **end while**

4.2.1 Genetic Programming for Guidance and Control

A dynamical system represented by Equation 4.1 is considered. In this equation, f denotes the system's dynamics, which may consist of both linear and nonlinear equations. The vector $\mathbf{x}$ represents the state variables, while $\mathbf{u}$ represents the vector of control inputs.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{4.1}$$

When employed for controller design, the objective of GP is to discover a mathematical model that can serve as the control law for the target dynamical system. This is expressed as $f_{GP}(\mathbf{v}) = \mathbf{u}$, where the GP model is a function of a set of user-defined variables $\mathbf{v}$. Consequently, this leads to an updated representation of the dynamical system, as given by Equation 4.2.

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), f_{GP}(\mathbf{v})) \tag{4.2}$$

Figure 4.4 illustrates a graphical representation of a GP control scheme. In this Figure, the inputs of the GP models are the tracking errors on the state variables. While this is a common approach in control scheme design, the user can define other variables for the GP to utilize. The GP will then autonomously evolve models that make use of the most influential inputs.

The methodology for designing a GP-based control system is outlined in Algorithm

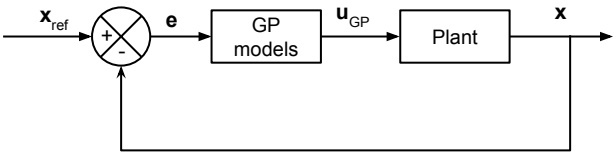


Figure 4.4: Application of GP models in a control scheme

2. When this process is executed in real-time, or online, during plant operation, it falls under the category of IC. According to the taxonomy presented in chapter 3, the described approach can be classified as G-4 since the GP generates the control law online from scratches using a set of primitives provided by the user. If the GP evolutionary process is not performed online, it is not classified as IC but it is considered AI-based control.

Algorithm 2 Generic process to obtain a GP based control system

```
Create initial population
for j = 1 → N_indPopInit do
    Propagate dynamical system in Equation 4.2 using the control law
    f_GP defined by the j-th individual
    Evaluate a fitness function according to the propagated trajectory
end for
while Termination criteria is not met do
    Perform crossover and mutation to obtain offspring
    for j = 1 → N_indOffspring do
        Propagate dynamical system in Equation 4.2 using the control law
        f_GP defined by the j-th individual
        Evaluate a fitness function according to the propagated trajectory
    end for
    Apply selection operator considering both offspring and parent populations
    Store best performing individual of current generation
end while
```

As shown in Algorithm 2, the fitness of each individual in each generation is assessed through the propagation of the dynamical system. This process is necessary for evaluating the performance of the generated control models. Consequently, the GP

algorithm learns, evolving individuals through direct interaction with the controlled plant. An advantage of this learning approach is its adaptability to systems with unknown dynamics. However, the computational cost associated with dynamical system propagation poses a significant challenge when deploying GP in an IC context. The evolutionary process must be executed within a time frame compatible with the system dynamics. This becomes especially critical when dealing with highly nonlinear equations that model the plant, as the dynamics propagation can become excessively time-consuming.

Moreover, during the initial stages of the evolutionary process, the GP may generate control laws that result in propagation failure. To address this issue, the fitness function must be formulated to identify and exclude such individuals, guiding the evolutionary process toward well-performing individuals.

4.3 Intelligent Control Application[1]

This section introduces an Intelligent Control (IC) application of GP. In this context, GP is employed at the outer loop level of the control scheme, assuming a perfect response from the actuators. GP is used to generate the values of the control signals required by the inner control loop to effectively track the desired trajectory. From this perspective, GP serves to generate real-time guidance commands for the inner loop controller. The application of GP follows the process outlined in Algorithm 2, where the fitness function is computed by propagating the system's equation of motion using the guidance laws generated by GP.

The real-time guidance generation capability of GP is assessed by subjecting the plant and environment models to external disturbances. Whenever a disturbance is detected, the GP evolutionary process is initiated to generate a guidance model in response.

The utilization of GP in an online setting introduces a specific challenge: the plant's initial conditions to propagate its trajectory using GP must be predetermined before the evolutionary process begins. Consequently, the time taken by the GP to conduct

[1]The content of this section was previously published in the conference paper [3].

the evolution must be known a priori and must be shorter than the total time of the simulated trajectory. For reliable results, this time interval must be defined conservatively, slightly exceeding the time required by the GP evolutionary process and yet remaining small enough to maintain effective control capabilities. Given the stochastic nature of GP and to test different disturbance configurations, a statistical analysis was undertaken and is presented in Subsection 4.3.3.

4.3.1 Test Case and Mission Profile

The selected test case involves tracking the ascent trajectory of a 2-dimensional version of the Goddard rocket [195]. The system is analyzed using polar coordinates, resulting in five states and two control variables. The states include the radial position or altitude r, angular position or path angle θ, radial velocity v_r, tangential velocity v_t, and mass m. The guidance commands consist of the radial thrust T_r and tangential thrust T_t. The dynamical model of the chosen plant is expressed in Equations 4.3, whose derivation can be found in [196].

$$\begin{cases} \dot{r} &= v_r \\ \dot{\theta} &= \frac{v_t}{r} \\ \dot{v}_r &= \frac{T_r}{m} - \frac{D_r}{m} - g + \frac{v_t^2}{r} \\ \dot{v}_t &= \frac{T_t}{m} - \frac{D_t}{m} + \frac{v_t v_r}{r} \\ \dot{m} &= -\dfrac{\sqrt{T_r^2 + T_t^2}}{g_0 I_{sp}} \end{cases} \tag{4.3}$$

The drag vector is defined as well with its radial and tangential components, as shown in Equation 4.4.

$$\begin{aligned} D_r &= \frac{1}{2}\rho v_r \sqrt{v_r^2 + v_t^2}\, c_d S \\ D_t &= \frac{1}{2}\rho v_t \sqrt{v_r^2 + v_t^2}\, c_d S \end{aligned} \tag{4.4}$$

The chosen density model is shown in Equation 4.5.

$$\rho = \rho_0 e^{-\beta r} \tag{4.5}$$

The gravitation acceleration is evaluated as in Equation 4.6

$$g = g_0 \Big(\frac{R}{r}\Big)^2 \tag{4.6}$$

The equations governing the dynamics of the Goddard rocket are influenced by various constant parameters, as detailed in Table 4.1. Constraints are imposed on the vehicle's mass and thrust values. Specifically, the total mass must always be greater than the vehicle's structural mass, i.e., $m > 1000\ kg$. Additionally, both tangential and radial thrust must remain within the range from zero to their respective maximum values, i.e., $0 < T_r, T_t < 1471\ kN$.

Table 4.1: Constant parameters used in the Goddard rocket test case

Symbol	Value	Unit	Meaning
R	6371000	m	Earth's radius
g_0	9.80665	m/s^2	Gravitational acceleration at sea level
m_0	100000	kg	Total initial mass
m_p	99000	kg	Propellant initial mass
S	4	m^2	Reference surface
c_d	0.6		Drag coefficient
I_{sp}	300	s	Specific impulse
ρ_0	1.225	$\frac{kg}{m^3}$	Air density at sea level
β	0.000118	m^{-1}	Scale factor

The mission profile involves reaching an altitude of 400 km while minimizing fuel consumption. An optimal trajectory for this mission was obtained through a direct optimal control study using Direct Pseudospectral Collocation discretization. The OpenGoddard[2] open-source library was employed for this purpose. The resulting optimal trajectory serves as a reference trajectory to be tracked by the GP.

The plant can operate in two conditions: nominal and off-nominal. In nominal conditions, no disturbances are present, and the open-loop controller obtained during the optimal control study is sufficient to track the reference trajectory. In off-nominal condi-

[2]https://github.com/istellartech/OpenGoddard

tions, disturbances are introduced during trajectory propagation, making the open-loop scheme incapable of tracking the reference trajectory. In such cases, the GP real-time guidance is employed. The guidance equations corresponding to the operational states of the system are summarized in Table 4.2.

Table 4.2: Open loop and closed loop control laws [3]

Open Loop	Closed Loop
$T_r(t) = T_{r_{ref}}(t)$	$T_r(t) = T_{r_{ref}}(t) + f_{GP_r}(e_r(t), e_{v_r}(t))$
$T_t(t) = T_{t_{ref}}(t)$	$T_t(t) = T_{t_{ref}}(t) + f_{GP_t}(e_\theta(t), e_{v_t}(t))$

The models found by the GP are not employed as complete guidance commands; instead, they are used as variations applied to the reference commands, as indicated in Table 4.2. This approach was adopted based on the observation that providing the GP with information about the domain in which the guidance command should be located facilitated the evolutionary process and yielded better results.

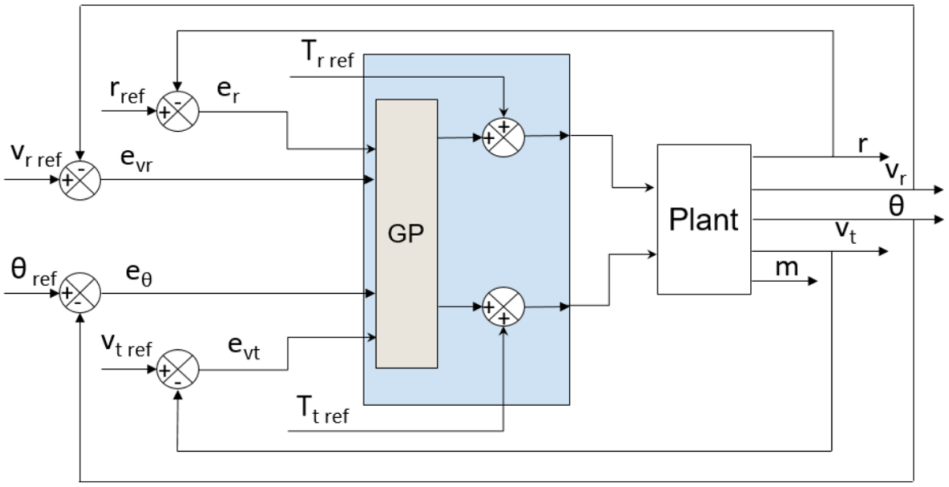


Figure 4.5: Simplified guidance scheme of modified Goddard rocket for off-nominal flight conditions [3]

The complete closed-loop guidance scheme used in off-nominal conditions is illustrated in Figure 4.5

Concerning the closed-loop models, the decision was made to design the GP equa-

tion for the radial thrust, denoted as f_{GP_r}, as a function solely dependent on the radial quantities. Similarly, the GP equation for the tangential thrust, denoted as f_{GP_t}, was designed as a function solely dependent on the tangential quantities. This choice aims to streamline the GP evolutionary process and reduce the complexity of the generated models.

4.3.2 Genetic Programming settings

The GP algorithm settings are summarized in Table 4.3, in particular: each GP individual is composed of two GP trees which are evolved simultaneously. One is used for the radial thrust f_{GP_r} and the other for the tangential thrust f_{GP_t}. As previously explained, they receive as input the respective quantities, i.e. f_{GP_r} receives the radial quantities, while f_{GP_t} the tangential ones. Two fitness measures are used, F and P. F represents the GP guidance laws' capability of tracking the reference trajectory and it is used to guide the GP evolutionary process. Specifically, it is computed as $F = \int_{t_0}^{t_f} \mid e_r(t) \mid dt$, where t_0 is the initial time of the trajectory, t_f is the final time and e_r is the tracking error on the radial position. P is employed to track the violation of the constraints. It is evaluated considering the constraints violations at all points of the trajectory. The values of the constrained quantities are grouped in the vector $\mathbf{c}$ and their values are compared against their defined maximum and minimum values, respectively $\mathbf{c}_{max}$ and $\mathbf{c}_{min}$. The overall vector of constraints violations will then be $\mathbf{c}_v = [max(0, \mathbf{c}_1 - \mathbf{c}_{1_{max}}), max(0, \mathbf{c}_{1_{min}} - \mathbf{c}_1), ..., max(0, \mathbf{c}_{N_p} - \mathbf{c}_{N_{pmax}}), max(0, \mathbf{c}_{N_{p_{min}}} - \mathbf{c}_{N_p})]$ where N_p is the number of points in the trajectory. Then, P is computed as $P = ||\mathbf{c}_v||_2$.

The mutation rate is initially set at 0.7 during the early stages of the evolutionary process to enhance exploration within the search space. Subsequently, as feasible individuals are discovered (i.e., those with $P = 0$), the mutation rate is reduced by 0.01 at each generation. Concurrently, the crossover rate is increased by 0.01 at each generation until the crossover rate reaches a value of 0.65.

The listed primitives comprise common arithmetic operations such as addition, subtraction and multiplication, but also a set of custom functions. These are *add*3, *plog*, *psqrt*, *pexp*. *add*3 is a ternary addition, while *plog*, *psqrt*, *pexp* are a modified

version of the *log*, *sqrt*, *exp* to avoid numerical errors. As an example, the *plog* function is presented in Equation 4.7.

$$plog(x) = \begin{cases} ln(x), & x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

Input features f_{GP_r}	e_r, e_{v_r}
Input features f_{GP_t}	e_θ, e_{v_t}
Population Size	500 individuals
Maximum Generations	150
Stopping criteria	Reaching the maximum number of generations or $F \leq 0.7$ and $P = 0$
Crossover probability	0.2 (+0.01 at every generation if $P = 0$) $\rightarrow$ 0.65
Mutation probability	0.7 (-0.01 at every generation if $P = 0$) $\rightarrow$ 0.25
1:1 Reproduction probability	0.1
Evolutionary strategy	$M + \Lambda$
M	Population size
Λ	Population Size
Number of Ephemeral constants	2
Limit Height	8
Limit Size	30
Selection Mechanism	Double Tournament
Double Tournament fitness size	2
Double Tournament parsimony size	1.6
Tree creation mechanism	Full
Mutation mechanisms	Uniform (50%), Shrink (25%), Insertion (15%), Mutate Ephemeral (10%)
Crossover mechanism	One point crossover
Primitives Set	$+$, $-$, $*$, *add*3, *tanh*, *psqrt* *plog*, *pexp*, *sin*, *cos*
Fitness measures	min F, min P

Table 4.3: GP setting for the guidance scheme design process.

4.3.3 Results

The GP algorithm was developed in Python 3.8 using the open source library DEAP [197]. The simulations were run on a Desktop PC with 8GB of RAM and an Intel®CoreTM i7-6700 CPU @3.40 GHz x 8 processors and multiprocessing was used.

The developed code is open source and can be found at https://github.com/strath-ace/smart-ml.

To test the effectiveness and reliability of the proposed IC approach, three different failure scenarios were simulated mimicking real world situations:

1. the unexpected change in the vehicle shape due to a component failure, which is simulated by introducing a variation in one of the plant parameters, specifically the drag coefficient c_d;

2. the unforeseen change in environmental conditions, which is simulated by introducing a wind gust, that can impact the performance of an ascent vehicle;

3. poor knowledge of the environment's physical models at design phase. This is a common scenario in space exploration applications. In such cases, engineers often have to rely on environmental models derived from observations, which may vary in accuracy. This is simulated by using a simplified air density model to find the optimal trajectory and a more detailed one during flight.

In the subsequent sections, Figures from 4.6 to 4.16 depict the trajectories of the radial distance r and the angular distance θ. These quantities are used to assess the outcomes of the conducted simulations. The success criterion for a simulation is met if the final values of r and θ deviate by no more than 1% from their respective reference values.

Hereafter, the results for each failure scenario are presented, with a comprehensive statistical analysis presented towards the end of this subsection.

1st Failure Scenario - c_d variation

To simulate a system's component failure, the drag coefficient c_d undergoes variation from its nominal value of 0.6 to a randomly selected value within the range $[0.61, 2]$. This alteration occurs at a randomly determined time within the interval $[20s, 250s]$ seconds. Refer to the end of this section for the results of the complete statistical analysis. In particular, Tables 4.4 and 4.5 list the results of the analysis conducted by

performing 1000 GP evolutionary processes considering 1000 different combinations of c_d and variation time. The values of c_d and variation time are drawn from a uniform distribution in the aforementioned ranges.

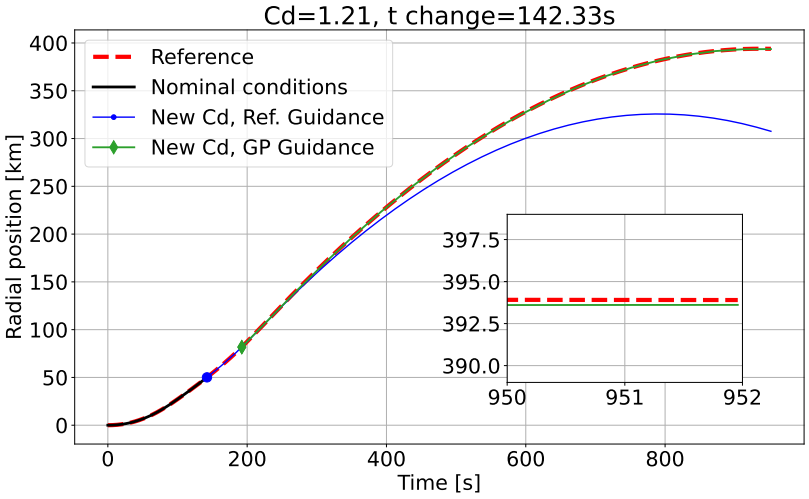


Figure 4.6: Altitude profiles for a Cd variation from 0.6 to 1.21 at 142.33 s [3]. The inset depicts a magnification of the last seconds of the trajectory.

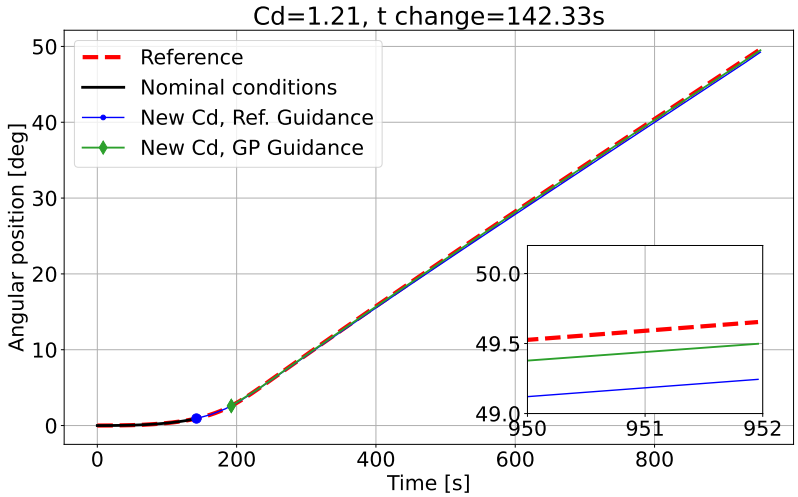


Figure 4.7: θ profile for a Cd variation from 0.6 to 1.21 at 142.33 s [3]. The inset depicts a magnification of the last seconds of the trajectory.

The trajectories depicted in Figures 4.6 and 4.7 illustrate the response when the drag coefficient (c_d) is altered from 0.6 to 1.21 at 142.33 seconds. Figure 4.6 displays the trajectory of radial distance, while Figure 4.7 presents the angular distance trajectory.

In both plots, the red dashed line represents the reference trajectory, the black line corresponds to the trajectory under nominal conditions, the blue line represents the trajectory under the influence of the new c_d with the open-loop guidance law, and the green line illustrates the trajectory under the influence of the new c_d with the closed-loop guidance law. The marker on both the blue and green lines indicates the point at which the guidance commands are applied. The inset in each plot provides a closer view of how well the trajectories obtained using the GP commands align with the reference towards the end of the simulation. The red shaded areas in the inset denote the 1% range from the reference.

As shown in Figure 4.6, the trajectory obtained using the GP models falls within the 1% range from the reference trajectory, demonstrating the effectiveness of the GP guidance law. In contrast, the trajectory obtained with the open-loop commands deviates significantly from the reference. A different behaviour is observed in Figure 4.7, where both the open and closed-loop commands result in successful trajectories. However, considering both the radial and angular distances, it can be concluded that the open-loop scheme struggles to handle a variation in the drag coefficient, while the GP guidance law successfully tracks the reference trajectory.

2nd Failure Scenario - Wind Gust

In this second failure scenario, the simulation of a change in environmental conditions is performed by introducing a wind gust. The wind gust acts within a random altitude range with a constant speed in the tangential direction. The initial altitude for the wind gust is randomly chosen within the $[0, 40]$ km range, with size within the $[10, 15]$ km interval. The gust magnitude is selected from the interval $[0, 24]$ m/s. Refer to the end of this section for the results of the complete statistical analysis. In particular, Tables 4.4 and 4.5 list the results of the analysis conducted by performing 1000 GP evolutionary processes considering 1000 different combinations of gust intensity, and range of application. The values of gust intensity, initial altitude of the gust zone and size of the gust zone are drawn from a uniform distribution in the aforementioned ranges.

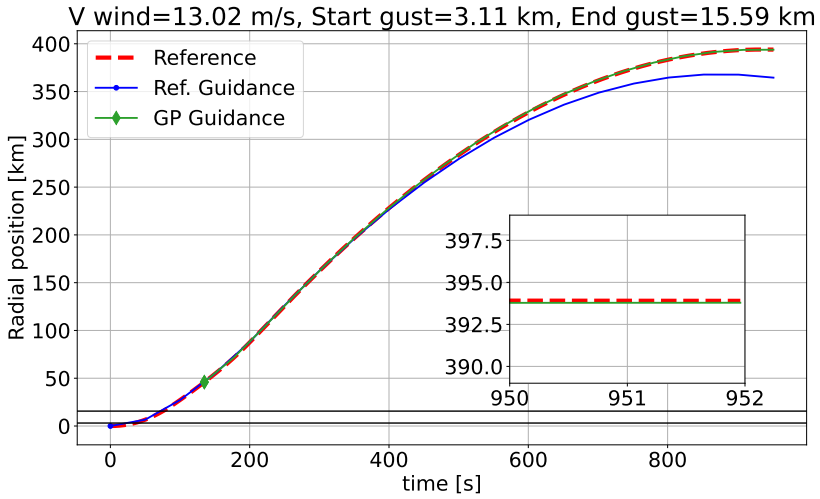


Figure 4.8: Altitude profiles for a wind gust of 13.02 m/s applied between 3.11 km and 15.59 km [3]. The inset depicts a magnification of the last seconds of the trajectory.

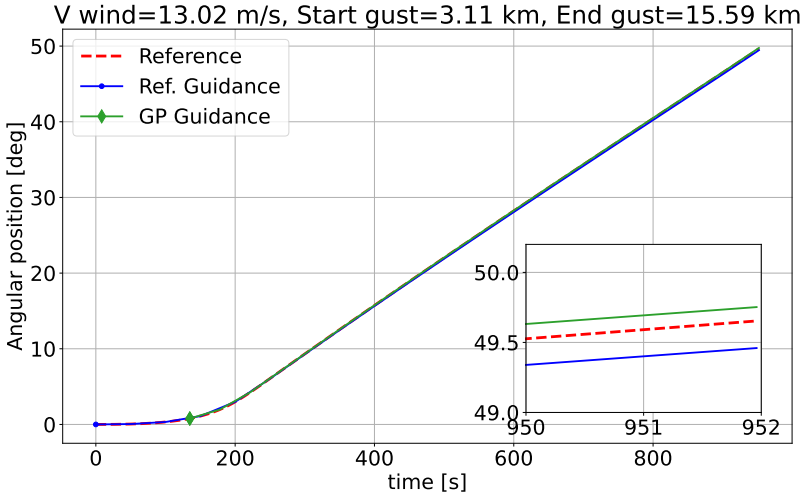


Figure 4.9: θ profile for a wind gust of 13.02 m/s applied between 3.11 km and 15.59 km[3]. The inset depicts a magnification of the last seconds of the trajectory.

The results presented in Figures 4.8 and 4.9 were obtained by introducing a wind gust with a magnitude of 13.02 m/s, acting between 3.11 km and 15.59 km.

In these plots, the reference trajectory is represented by a red dashed line, the trajectory flown with the applied gust and the open-loop guidance is depicted in blue, while the trajectory flown with the applied gust and the GP closed-loop guidance is shown in green. Once again, the markers denote the point at which the open and closed-

loop guidance commands are applied. The inset illustrates how close the trajectories obtained using the GP commands get to the reference in the final part of the trajectory, with the red shaded areas indicating the 1% range from the reference. Similar to the previous failure scenario, the open-loop guidance commands can successfully track the angular position but fail in tracking the radial position. Therefore, for this failure scenario as well, the GP guidance laws are necessary to track the desired trajectory in terms of radial and angular positions.

3rd Failure Scenario - Partially known density model

In the final failure scenario, the simulation aimed at replicating a lack of knowledge in the physical models at the design stage. For this purpose, an optimal trajectory was computed offline using the simplified density model presented in Equation 4.5, representing an incorrect atmospheric model. Conversely, during the actual simulated trajectory, the U.S. Standard Atmosphere Model 1962 (USSA1962) model was employed, reflecting the real atmospheric conditions. A direct comparison of the two density models is illustrated in Figures 4.10 and 4.11 on a semi-logarithmic scale. In both plots, the altitude is considered up to 50 km since, beyond this threshold, the density value becomes negligible.

The impact of employing different air density models is depicted in Figure 4.12. This picture showcases the altitude profiles obtained using the two different models while using the same open-loop guidance commands derived from the optimal control study performed using the density model in Equation 4.5. Consequently, the use of a different atmospheric model results in a different final altitude if the guidance commands are not updated accordingly.

To adapt to unforeseen environmental conditions, the GP guidance laws are generated in real-time by updating the current environmental models with newly acquired data during the mission. This is achieved through Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) interpolation, as depicted in Equation 4.8.

$$\rho_{new} = PCHIP(t, \rho) \tag{4.8}$$

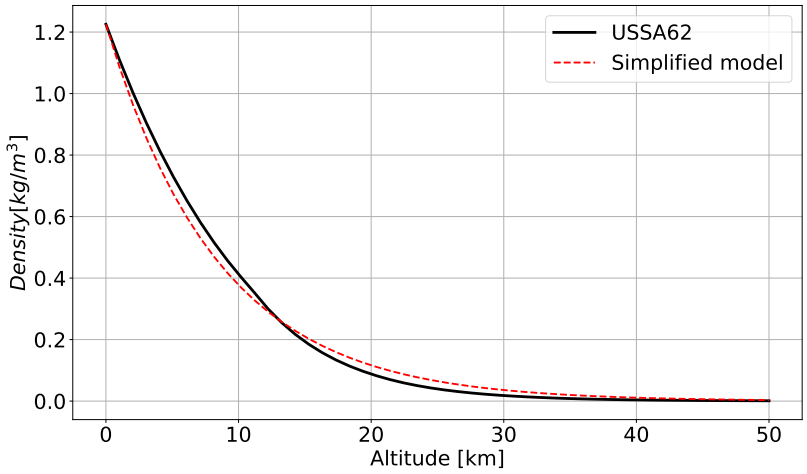


Figure 4.10: Comparison between the density models up to 50 km [3]

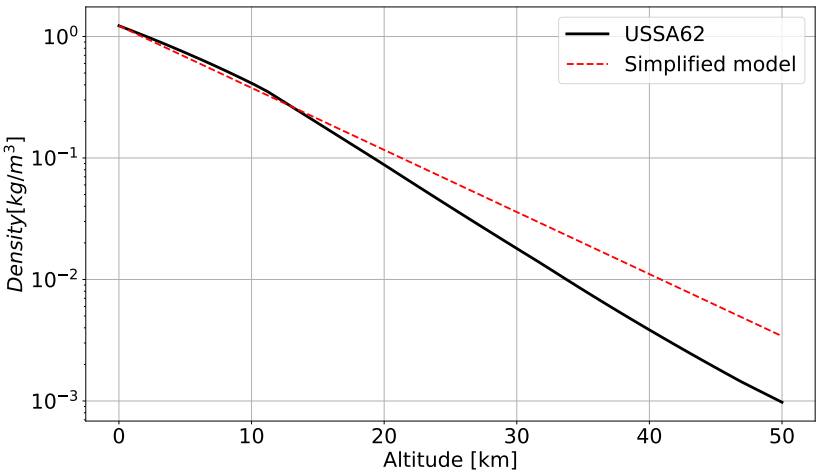


Figure 4.11: Comparison between the density models up to 50 km on a semi-logarithmic scale[3]

with ρ defined as in Equation 4.9

$$\rho = \begin{cases} \rho_{USSA1962}, & 0 \leq t \leq t_{eval} \\ \rho_{Simplified}, & t_{eval} < t \leq t_{end} \end{cases} \tag{4.9}$$

$\rho_{Simplified}$ is the density model in Equation 4.5 and t_{eval} is the time at which the

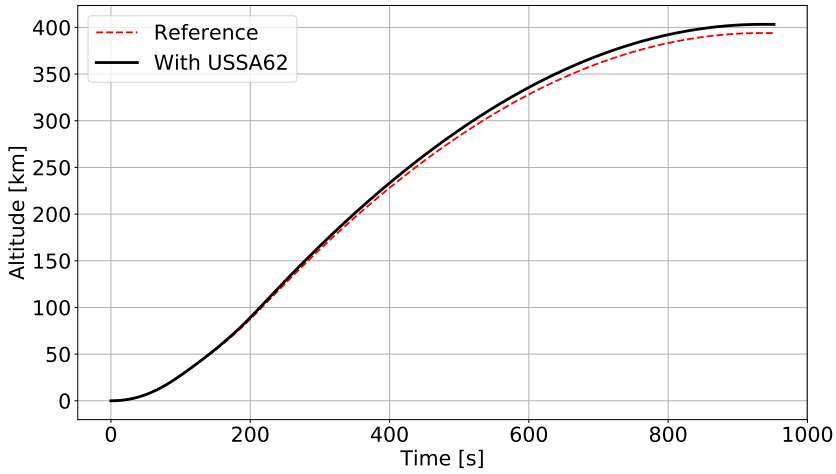


Figure 4.12: Altitude profile using the two different density models. The red dashed line is the altitude profile obtained using the simplified density model, while the continuous black line is the one obtained using the USSA1962 model. In both cases the trajectories are propagated using the reference open loop guidance commands. [3]

GP evaluation starts.

The objective of this approach is to dynamically update the density model online by interpolating newly gathered data during the mission with the anticipated data produced by the existing density model. Despite potential inaccuracies, the old model may still contain valuable information.

The results presented in Figures 4.15 to 4.16 illustrate the trajectories of the radial and tangential positions, r and θ, obtained for the specific failure scenario. In this case, the density model was updated online three times, and each obtained model is depicted in Figures 4.13 and 4.14. The density model was updated every time the tracking error on the radial position became greater than 100 m.

The notation used in Figures 4.15 and 4.16 has the following meaning:

- Red dashed line: reference trajectory, obtained with the simplified density model. Simplified model in Figures 4.13 and 4.14.
- Black continuous line: trajectory obtained with the USSA1962 density model and reference open loop guidance. USSA1962 model in Figures 4.13 and 4.14.

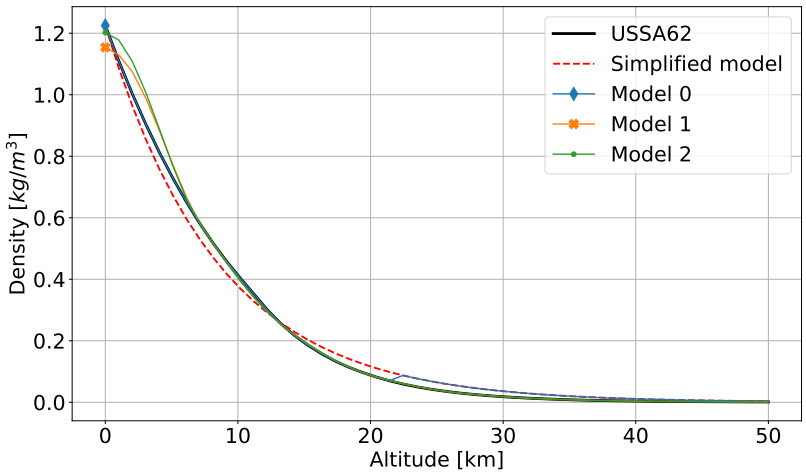


Figure 4.13: Comparison between the different density models used [3]

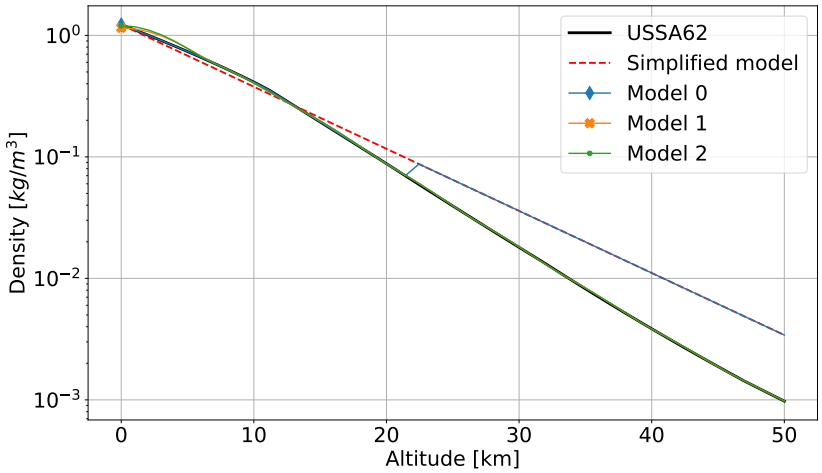


Figure 4.14: Comparison between the different density models used on a semi logarithmic scale [3]

- Light blue line: trajectory performed using density model 0 and the GP guidance. Model 0 in Figures 4.13 and 4.14.
- Orange line: trajectory performed using density model 1 and the GP guidance. Model 1 in Figures 4.13 and 4.14.
- Green line: trajectory performed using density model 2 and the GP guidance. Model 2 in Figures 4.13 and 4.14.

The insets in Figures 4.15 and 4.16 provide insight into how closely the trajectories

obtained using the GP commands align with the reference at the end of the trajectory, with the red shaded areas indicating the 1% range from the reference. Similar to the previous failure scenarios, the open-loop guidance commands can successfully track the angular position under off-nominal conditions. However, they struggle to track the radial position. Once again, the GP guidance laws are needed for successfully tracking the desired trajectory in terms of both radial and angular positions.

The results of the complete statistical analysis are listed in Tables 4.4 and 4.5. The statistical analysis was performed by repeating the trajectory propagation 1000 times. More on the statistical analysis is discussed in the following.

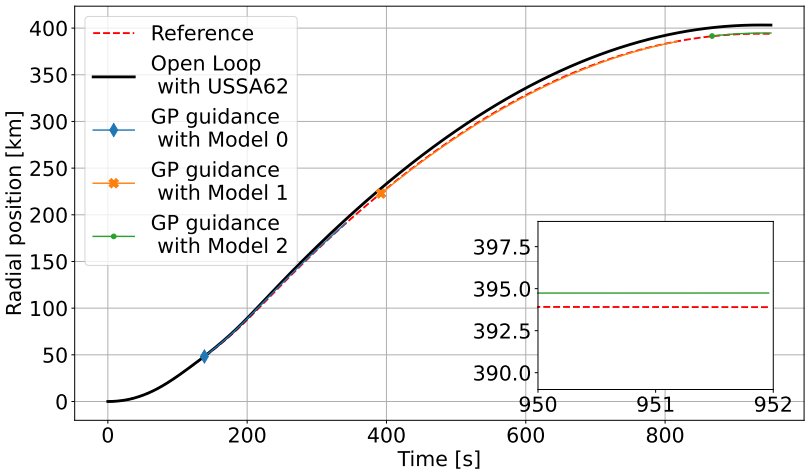


Figure 4.15: Altitude profile in the 3rd scenario. The different markers on the plot represent the trajectory performed with different density models [3]

Statistical Analysis

Given the stochastic nature of the GP algorithm, its performance can vary, requiring a robustness assessment through a statistical study. Moreover, both variations in the GP initialization and in the applied failures are considered, so to test the robustness also against a broader range of failure cases. Table 4.4 provides a summary of the results obtained from a statistical study conducted on the three previously discussed failure scenarios.

In each of the three scenarios, 1000 trajectory propagations were performed. For

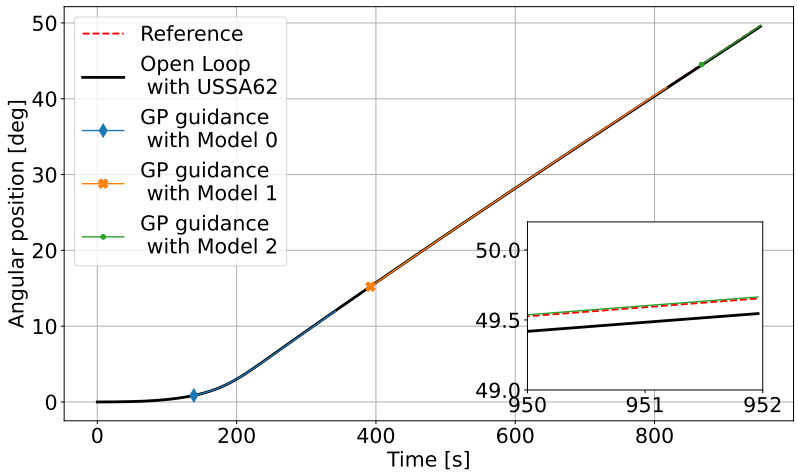


Figure 4.16: θ profile in the 3rd scenario. The different markers on the plot represent the trajectory performed with different density models [3]

each propagation, the GP evolutionary process was initialized differently and different combinations of failure scenarios were used. In particular, for the first failure scenario, each propagation was performed by using a randomly initialized GP population, a value of c_d drawn from a random uniform distribution in the $[0.61, 2]$ range, and a c_d variation time drawn from a random uniform distribution in the $[20, 250]s$ interval. In the second scenario, each simulation was performed with a different GP initialization and a different combination of wind speed and altitude range. Also in this scenario, the GP's population was initialized randomly, the gust magnitude was drawn from a random uniform distribution in the $[0, 24]m/s$ range, the initial altitude of the gust zone was picked from a random uniform distribution in the $[0, 40]km$ range and the size of the gust zone was sampled from a random uniform distribution in the $[10, 15]km$ interval. Concerning the third scenario, each time a GP evolution was performed, its population was initialized randomly.

Reviewing the results presented in Table 4.4, the Evaluation Time Success Rate indicates how frequently the GP evolution was completed within the allocated time interval. This metric is employed because the time spent by the GP to complete the evolutionary process varied in each simulation. The allocated time is reported in the "Fixed time interval" row. Min, max and median evaluation time refers to the minimum, maximum and median evaluated on the time spent by the GP to perform

Table 4.4: Statistics of GP evaluations performed on the 3 scenarios presented [3]

	Cd Variation	Wind Gust	Density Model
GP Evaluations	1000	1000	3153
Fixed time interval	40 s	100 s	40 s
Min evaluation time	4.76 s	6.46 s	3.22 s
Max evaluation time	04h58m20s	01h36m53s	54m49s
Median evaluation time	14.7 s	88.76 s	38.35 s
Evaluation Time Success Rate	61.90%	52.60%	50.77%
Range Success Rate	71.70%	99.80%	79.80%

the evolutionary process considering the 1000 simulations.

Meanwhile, the range Success Rate shows how often the GP evolution generated a guidance model capable of tracking the reference trajectory and guiding the system successfully within the 1% range from the desired trajectory. According to the obtained results, the proposed IC approach achieves the desired precision in 83.7% on average considering the three failure scenarios, with a noteworthy peak of 99.8% success in the second scenario, which is likely to be a common occurrence in real-world applications.

Learning approach

To enhance the obtained results, a learning mechanism was introduced and its effects were studied. In the learning approach, a few of the best-performing individuals from each GP evolution were retained and added to the initial population, which was randomly created at the beginning of the next simulation. The maximum size of the population composed of well-performing individuals was set at 300, while the total population for each simulation was maintained at 500 individuals. The inclusion of 200 randomly created individuals aimed at ensuring a certain degree of diversity in the population and preventing overfitting.

Table 4.5 compares the results obtained with and without the learning approach. From the comparison, it is clear that the incorporation of the learning approach resulted in improved performance across all three scenarios, resulting in higher success rates for both Range and Evaluation Time quantities.

A graphical depiction of these improvements is shown in Figures 4.17 to 4.19. These

Table 4.5: Statistics of GP evaluations performed on the 3 scenarios presented, comparing the results obtained with the learning approach against those obtained without learning. [3]

	Cd Variation		Wind Gust		Density Model	
	No Learning	Learning	No Learning	Learning	No Learning	Learning
GP Evaluations	1000	1000	1000	1000	3153	3108
Fixed time interval	40 s	40 s	100 s	100 s	40 s	40 s
Min evaluation time	4.76 s	2.73 s	6.46 s	8.34 s	3.22 s	1.45 s
Max evaluation time	04h58m20s	01h18m09s	01h36m53s	26m16s	54m49s	01h01m19s
Median evaluation time	14.7 s	5.81 s	88.76 s	22.09 s	38.35 s	28.49 s
Evaluation Time Success Rate	61.90%	83.10%	52.60%	90.80%	50.77%	54.89%
Range Success Rate	71.70%	73.50%	99.80%	100%	79.80%	79.60%

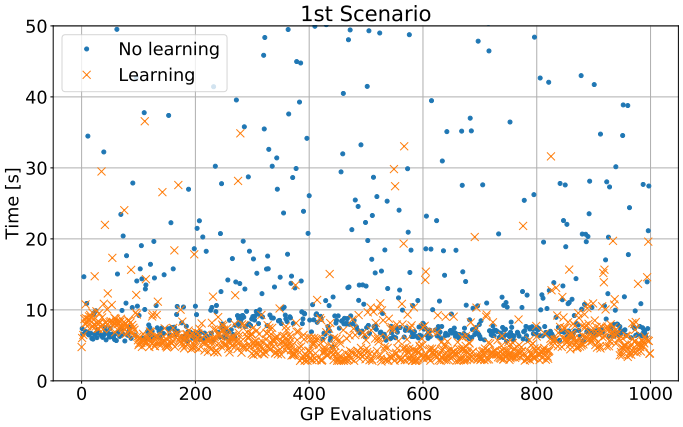


Figure 4.17: Detail of time distribution for 1st scenario between 0 and 50 seconds. For this simulation, the median evaluation time was 14.7 s without learning and 5.8 s with learning [3]

present plots of evaluation times against the number of evaluations. The results obtained with and without the learning approach are marked by orange crosses and blue dots, respectively. The evaluation time is depicted as up to 50 seconds in the first and third scenarios and up to 100 seconds in the second scenario, focusing on the region encompassing the median evaluation time. Notably, these plots reveal that points corresponding to the learning approach are tightly clustered around a smaller evaluation time in comparison to those corresponding to the approach without learning. This clearly shows how the learning approach can help in reducing the evaluation times and in improving the consistency of the GP algorithm. In this context, consistency refers to the tendency of the GP algorithm enhanced with the learning approach to

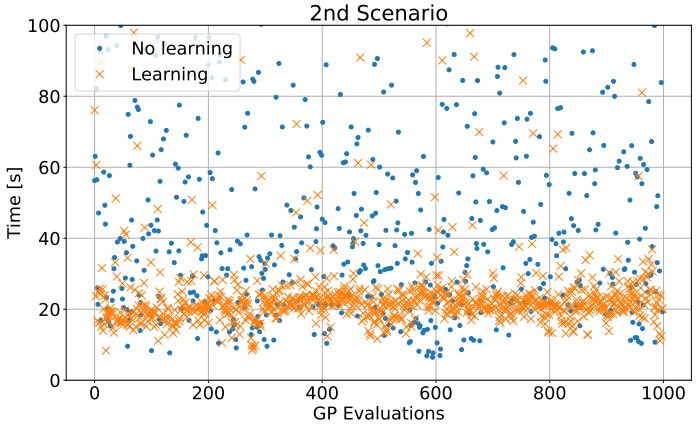


Figure 4.18: Detail of time distribution for 2nd scenario between 0 and 100 seconds. For this simulation, the median evaluation time was 88.76 s without learning and 22.09 s with learning [3]

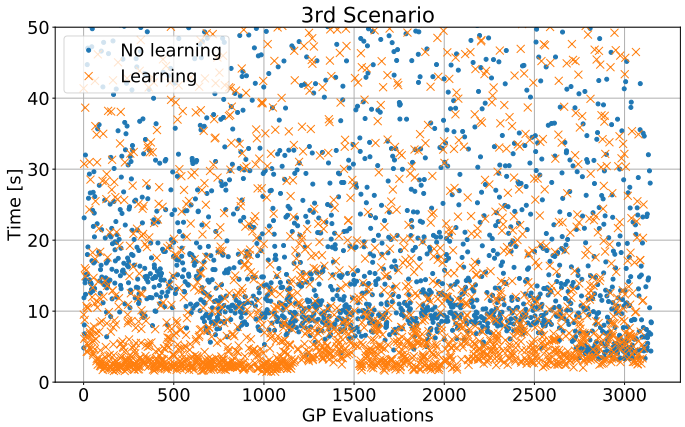


Figure 4.19: Detail of time distribution for 3rd scenario between 0 and 50 seconds. For this simulation the median evaluation time was 38.35 s without and 28.49 s with learning [3]

perform more evolutions in a time frame centred around the median, in contrast to those obtained without learning.

4.4 Inclusive Genetic Programming [3]

This section presents the Inclusive Genetic Programming (IGP). A heuristic developed during the doctoral work and initially devised to enhance the performance of a GP algorithm applied in control applications, introducing mechanisms to maintain population diversity throughout the evolutionary process. This algorithm is discussed since it is employed in the real-time guidance scheme discussed in chapter 5.

Diversity loss is a common challenge in EAs that can lead to premature convergence. This occurs when the population is filled with duplicates and variations of the same well-performing individuals, resulting in a loss of genetic diversity. Diversity loss is particularly relevant in control applications, where a control law capable of capturing nonlinearities is crucial for controlling highly nonlinear plants. GP achieves this by increasing the number of nodes in the individuals. However, if this behaviour is uncontrolled, the number of nodes may grow excessively, yielding large individuals with not necessarily improved performance. This is why bloat control operators are employed to mitigate this issue. Nevertheless, these operators can be overly restrictive, potentially discarding individuals larger than the average but containing valuable genetic information.

The significance of population diversity in EAs has been extensively explored by various authors. In [198], diversity is discussed as a method for managing exploration and exploitation during the evolutionary process, with the paper also detailing various diversity measures. The balance between exploration and exploitation is influenced by population diversity, where greater diversity favours exploration and lesser diversity enhances exploitation. Squillero et al. [199] provide a comprehensive survey of methodologies for promoting diversity in evolutionary optimization. Additionally, [200] analyzes different diversity measures, contributing to the understanding of diversity's role in evolutionary algorithms.

The approach employed in the IGP involves classifying individuals into different niches based on their genotype (i.e., their structure). Subsequently, crossover, mutation,

[3] The content of this section was previously published in the conference papers [20, 21] and the journal paper [4].

and selection operations are conducted considering all the obtained niches. Thus, IGP can be categorized as a niching technique. However, it differs from traditional niching approaches, which involve dividing the population into distinct niches and evolving them separately to discover different local optima. In contrast, IGP considers all the niches to facilitate the flow of genes between them.

The concept of combining diverse individuals, such as those from different niches, rather than focusing solely on well-performing ones, has been explored in the past. For instance, Aslam et al. [201] investigated this concept, although their approach is based on the phenotypic diversity of individuals and does not involve the subdivision of the population into niches. Instead, they classified individuals based on a measure called Binary String Fitness Characterization, initially introduced by Day and Nandi [202]. Additionally, they explored the concept of good and bad diversity.

4.4.1 Inclusive Evolutionary Process

The IGP is built on a modified version of the evolutionary strategy $M + \Lambda$ [203], the Inclusive $M + \Lambda$, developed during this doctoral work and originally introduced in [21]. The designed evolutionary strategy is depicted in Figure 4.20 and outlined in Algorithm 3. The Inclusive $M + \Lambda$ was created to incorporate the niches creation into the evolutionary process, in particular at the beginning of the evolutionary process and after the generation of new offspring. It is also structured to consider both the Inclusive Reproduction and Inclusive Tournament selection mechanisms, which were developed for the IGP and described in the following.

Niches Creation

The cornerstone of the Inclusive Evolutionary Process is the niches creation mechanism. In this context, a niche is defined as an interval between lower and upper bounds, corresponding to two values representing lengths of the GP individuals. To determine these boundaries, each time the niches are created, the individuals in the current population are examined to identify the maximum and minimum lengths in the population. Based on these values and the user-input desired number of niches, denoted as n, a set of n

Algorithm 3 Pseudocode of the Inclusive $M + \Lambda$ [3]

```
Evaluate the fitness of the individuals in the population
Update Hall of Fame of best individuals
while Termination criteria is not met do
    Create n niches according to the maximum and minimum length
    of the individuals in the population and allocate the individuals
    to their respective niche
    Perform Inclusive Reproduction to produce λ offspring
    Evaluate the fitness of the individuals in the obtained offspring
    Update Hall of Fame of best individuals
    Create n niches according to the maximum and minimum length of the
    individuals considering both the parents and the offspring and allocate
    the individuals to their respective niche
    Perform Inclusive Tournament Selection to select M new parents
end while
```

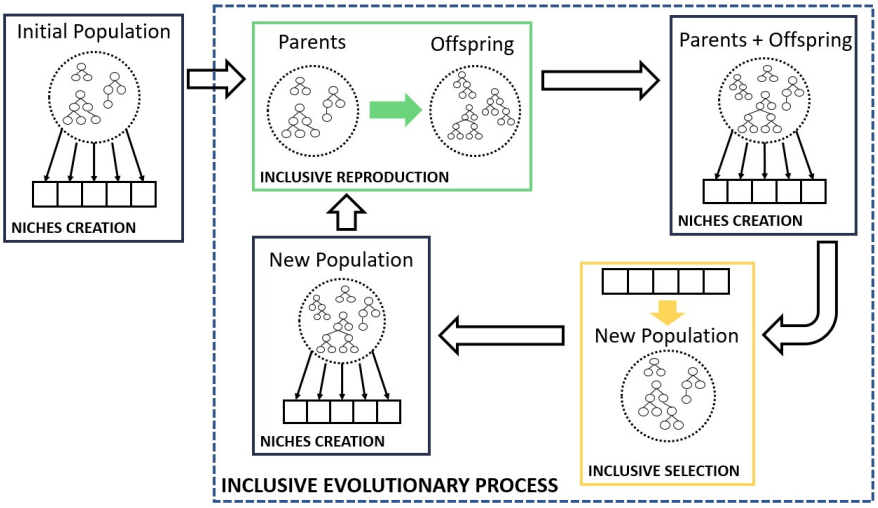


Figure 4.20: Inclusive Evolutionary Process schematic representation [4]

niches is established. The size of these niches is determined by linearly dividing the interval from the minimum to the maximum length by n. Consequently, each niche will encompass individuals with lengths falling within the boundaries of that niche. The niches creation process is illustrated in Figure 4.21. In this example, a population of ten individuals, with lengths 1, 1, 2, 2, 3, 4, 4, 5, 7, 8 (excluding the root node), is considered. Ten niches are created, spanning the lengths depicted in the figure and containing the individuals shown within them.

The overall number of niches remains constant throughout the evolutionary process.

However, the number of individuals within each niche may fluctuate in response to changes in the individuals' lengths resulting from evolutionary operators and alterations in the niches' boundaries. Specifically, if the maximum and minimum lengths in the population change, the subdivisions of the interval will be adjusted accordingly. This variation in the sizes of niches induces a migration of individuals between niches, thereby contributing to the maintenance of population diversity.

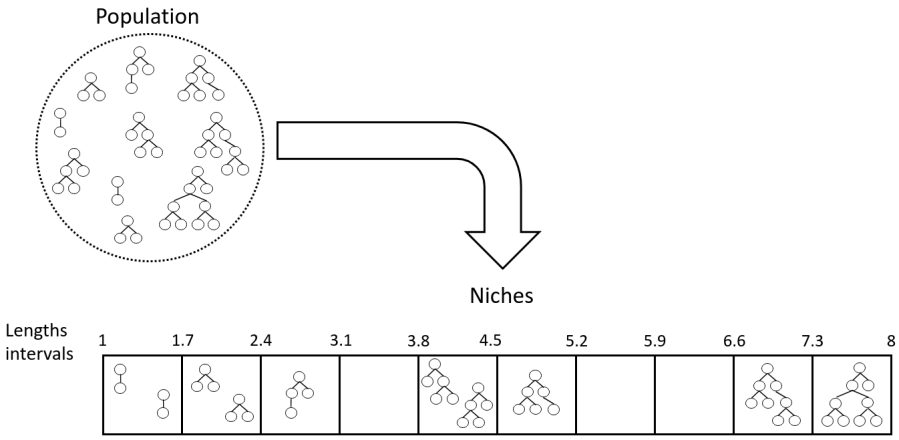


Figure 4.21: Illustration of the niches creation rationale. [3]

Inclusive Reproduction and Inclusive Tournament

To fully exploit the created niches, a modified version of the VarOr reproduction algorithm [197] and of the Double Tournament [204] selection process were developed. The Inclusive Reproduction, outlined in Algorithm 4, performs similar operations to the VarOr algorithm, including crossover, mutation, or 1:1 reproduction, but selects individuals from different niches.

When crossover is chosen, a one-point crossover is applied between two individuals from two different niches. One individual is the best from one niche, while the other is randomly selected from another niche. This approach retains well-performing genes while preserving diversity and preventing overfitting. Additionally, a mechanism is implemented to prevent breeding between identical or highly similar individuals (lines 16-20 in Algorithm 4). Here, n_l is a predefined constant indicating the maximum number of loop iterations to avoid potential infinite loops.

When mutation is selected, a mutation operator is randomly chosen from a set of options such as uniform, shrink, insertion, or mutate ephemeral, based on a user-defined selection probability. These mutation operators, as defined in the DEAP library [197], contribute to increasing randomization introduced by the mutation operation. The mutation is applied to an individual randomly chosen from the selected niche.

Lastly, when 1:1 reproduction is chosen, the best individual from the population is passed to the offspring. The niches used in all three operations (crossover, mutation, and 1:1 reproduction) are selected from a list of exploitable niches, which is continually updated to avoid consistently selecting from the same niches.

The Inclusive Tournament involves executing a Double Tournament [204] on each niche, as outlined in Algorithm 5. In the Inclusive Tournament, niches are selected sequentially. The double tournament on each niche is conducted at most t times, where t is the number of individuals within the respective niche, preventing the presence of clones in the final population.

The IGP can be configured to evolve individuals capable of satisfying a set of imposed constraints. For this purpose, the objective function of each individual is formulated as $F_{\text{ind}} = [F, P]$, where F represents the current goal of the algorithm (e.g., minimizing the prediction error), and P is a penalty function considering the constraints violation. Subsequently, the Double Tournament selection used within the Inclusive Tournament, as outlined in Algorithm 5, has been modified and is presented in Algorithms 6 and 7. The objective of this algorithm is to first select individuals that satisfy the constraints, thereby obtaining a population predominantly composed of constraints-preserving individuals. Subsequently, individuals are selected based on their fitness. The algorithm is configured for a minimization problem but can be adapted for a maximization problem as well.

4.4.2 Test Procedure and Chosen Benchmarks

To assess the efficacy of the IGP algorithm, it was compared against a standard GP implementation (Standard Genetic Programming (SGP)). The SGP employs the standard $M + \Lambda$ evolutionary strategy, the standard VarOr reproduction, and the standard

Algorithm 4 Pseudocode of Inclusive Reproduction [3]

```
1: if Crossover probability < cx_limit then
2:     Mutation probability ← Mutation probability - 0.01
3:     Crossover probability ← Crossover probability + 0.01
4: end if
5: Good Indexes ← Indexes of filled niches
6: List of exploitable niches ← Good Indexes
7: while Size offspring < λ do
8:     Choice ← Random number between [0, 1]
9:     if Choice < Crossover Probability then
10:        if List of exploitable niches is empty then
11:            List of exploitable niches ← Good Indexes
12:        end if
13:        Select randomly two different niches from List of exploitable niches
14:        Remove chosen niches from List of exploitable niches
15:        Select the best individual from the first niche and select a random
           individual from the second niche
16:        n ← 0
17:        while The selected individuals have the same fitness and n < n_l do
18:            Repeat lines 10 to 15
19:            n ← n+1
20:        end while
21:        Apply crossover to the chosen individuals
22:        Add first child to offspring
23:        if Size of offspring < λ then
24:            Add the second child to the offspring
25:        end if
26:    else
27:        if Choice < Mutation probability + Crossover Probability then
28:            Repeat lines 10 to 15 but selecting only one category
29:            Select randomly one individual from the chosen category
30:            Perform mutation of the chosen individual
31:            Add mutated individual to the offspring
32:        else
33:            Repeat lines 27, 28
34:            Add chosen individual to the offspring
35:        end if
36:    end if
37: end while
```

Algorithm 5 Pseudocode of Inclusive Tournament [3]

```
1: while Number of selected individuals < M do
2:     for i = 1 → number of niches do
3:         if Number of selected individuals from i-th niche < total number of individuals in i-th niche then
4:             Select one individual in i-th niche with modified Double Tournament selection
5:         end if
6:     end for
7: end while
```

Algorithm 6 Pseudocode of modified Double Tournament Selection - Part 1

```
1: A set of individuals I is received as input
2: chosen ← empty list
3: l_p ← random uniform ∈ [0, 1]
4: while Length of chosen < 2 do
5:     Select randomly two individuals (i_1, i_2) from I
6:     if P_{f_{i_1}} == 0 and P_{f_{i_2}} == 0 then
7:         if F_{f_{i_1}} < F_{f_{i_2}} then
8:             i_1 is appended to chosen
9:         else
10:            i_2 is appended to chosen
11:        end if
12:    else
13:        if P_{f_{i_1}} == 0 and P_{f_{i_2}} != 0 then
14:            i_1 is appended to chosen
15:        end if
16:
17:        if P_{f_{i_1}} != 0 and P_{f_{i_2}} == 0 then
18:            i_2 is appended to chosen
19:        end if
20:
21:        if P_{f_{i_1}} != 0 and P_{f_{i_2}} != 0 then
22:            if P_{f_{i_1}} < P_{f_{i_2}} then
23:                i_1 is appended to chosen
24:            else
25:                i_2 is appended to chosen
26:            end if
27:        end if
28:    end if
29: end while
```

Algorithm 7 Pseudocode of modified Double Tournament Selection - Part 2

```
n ← random uniform ∈ [0, 1]
if Length i1 == Length i2 then
    lp ← 0.5
    ind1 ← i1
    ind2 ← i2
else
    if Length i1 < Length i2 then
        ind1 ← i1
        ind2 ← i2
    else
        ind1 ← i2
        ind2 ← i1
    end if
end if
if n < lp then
    Select ind1
else
    Select ind2
end if
Output selected individual
```

double tournament selection, as implemented in the DEAP library [197].

The comparison between the two algorithms was conducted on nine regression problems outlined in [205] and summarized in Table 4.6.

Name	Definition	Features	Instances
Koza-1	$f_{K1}(x) = x^4 + x^3 + x^2 + x$	1	20(train), 100(test)
Korns-11	$f_{K11}(x, y, z, v, w) = 6.87 + 11\cos(7.23x^3)$	5	5000(train, test)
S1	$f_{S1}(x) = e^{-x}x^3\sin(x)\cos(x)\cdot$ $\cdot(\sin^2(x)\cos(x) - 1)(\sin^2(x)\cos(x) - 1)$	1	110(train), 220(test)
S2	$f_{S2}(x, y) = (y - 5)f_{S1}(x)$	2	110(train), 220(test)
UB	$f_{UB}(x_1, x_2, x_3, x_4, x_5) = \frac{10}{5+\sum_{i=1}^{5}(x_i-3)^2}$	5	1024(train), 5000(test)
ENC	[206]	8	768 (train:70%, test:30%)
ENH	[206]	8	768 (train:70%, test:30%)
CCS	[207]	8	1030 (train:70%, test:30%)
ASN	[208]	5	1503 (train:70%, test:30%)

Table 4.6: Summary of chosen benchmarks

The chosen benchmarks encompass both synthetic and real-world data, providing a diverse set of problems for evaluation. The number of samples and sampling techniques used to generate the data were consistent with those adopted in [205]. There were two

exceptions: for the benchmark *korns11*, 5000 samples were utilized instead of 10000 to decrease the computational time for both training and testing samples; and for the benchmark *S2*, the same number of training samples used for the x variable was also applied to the y variable.

To evaluate the performances of the two algorithms, three indicators are used:

1. RMSE of the final solution found by both algorithms on test and train data.

2. Evolution of the RMSE of the best individual during the evolutionary process.

3. Evolution of the entropy during the evolutionary process. Rosca demonstrated [209] that entropy reflects the diversity within a population, and the higher the entropy, the more diverse the population is. Entropy is computed using Equation 4.10, where p_k represents the proportion of the population P inside niche k. Empty niches are not taken into account in the entropy evaluation.

$$E(P) = -\sum_{k}(p_k \cdot ln(p_k)) \tag{4.10}$$

   As a clarifying example of the relationship between entropy and diversity, two populations of 10 individuals are considered, both subdivided into 10 niches. The individuals of the first population are more similar to each other filling only two niches, five in the first and five in the second niche. The individuals of the second population are more diverse with all the 10 niches filled, one individual per niche. According to this, the entropy for the first population will be $E(P_1) = -2(0.5ln(0.5)) = 0.69$, while for the second population it will be $E(P_2) = -10(0.1ln(0.1)) = 2.3$. This example clearly shows how greater diversity corresponds to greater entropy.

The mean and standard deviation for all three indicators are evaluated by performing 100 simulations.

Algorithms Settings

Both IGP and SGP are implemented in Python 3.8, using the open source library DEAP [197]. The simulations are performed on a machine with 16GB of RAM and an Intel® Core™ i7-8750H CPU @ 2.20GHz × 12 threads, and multiprocessing is used. The developed code is open source and can be found at `https://github.com/strath-ace/smart-ml`.

To perform the simulations, the algorithms are set as in Table 4.7.

	SGP	IGP
Population Size	300 individuals	
Maximum Generations	300	
Stopping criteria	Reaching maximum number of generations	
Crossover probability	0.8	$0.2 \rightarrow 0.8$
Mutation probability	0.2	$0.8 \rightarrow 0.2$
Evolutionary strategy	$M + \Lambda$	Inclusive $M + \Lambda$
M	Population size	
Λ	Population Size × 1.2	
Number of Ephemeral constants	1	
Limit Height	15	
Limit Size	30	
Selection Mechanism	Double Tournament	Inclusive Tournament
Double Tournament fitness size	2	
Double Tournament parsimony size	1.2	
Tree creation mechanism	Ramped half and half	
Mutation mechanisms	Uniform (50%), Shrink (5%), Insertion (25%), Mutate Ephemeral (20%)	
Crossover mechanism	One point crossover	
Primitives Set	$+$, $-$, $*$, $add3$, $mul3$, $tanh$ $square, plog$, $pexp$, sin, cos	
Fitness measure	RMSE	

Table 4.7: Settings for the SGP and IGP algorithms. The percentages near the mutation mechanisms refer to the probability of that mutation mechanism being chosen when the mutation is performed. [3]

The "Limit height" and "Limit size" in Table 4.7 represent parameters used by the bloat control mechanism implemented in the DEAP library. Regarding the mutation operators, they were used according to the specified probabilities in brackets; for instance, the uniform mutation was selected 50% of the time when a mutation was

performed. These mutation operators are those coded in the DEAP library.

The primitive set comprises basic arithmetic functions such as addition, subtraction, and multiplication plus a set of custom functions. These are $add3$ and $mul3$, representing ternary addition and multiplication, and $plog$ and $pexp$, which are a modified version of the logarithmic and exponential functions to prevent numerical errors. As an example, the $pexp$ function is presented in Equation 4.11.

$$pexp(x) = \begin{cases} exp(100), & x > 100 \\ exp(x), & -100 \leq x \leq 100 \\ exp(-100), & x < -100 \end{cases} \tag{4.11}$$

In the SGP, the crossover and mutation probabilities were fixed, while in the IGP, these probabilities changed dynamically. The mutation probability was decreased by 0.01, and the crossover probability was increased by the same quantity at each generation, up to a limit cx_{limit} defined by the user. This dynamic adjustment is illustrated in Algorithm 4 at lines 1-4, where cx_{limit} is the crossover limit. This approach was employed to enhance the algorithm's exploration capability at the beginning of the evolutionary process while improving exploitation towards the end. The same approach was tested in the SGP but did not result in any improvement. The simulations were terminated at 300 generations to strike a balance between result quality and reasonable computation times.

Results

As outlined in Subsection 4.4.2, the IGP and SGP were evaluated on nine distinct benchmarks representing synthetic and real-world regression problems. Their performances were evaluated using three indicators: the Root Mean Squared Error (RMSE) on the final solution, the evolution of the RMSE throughout the evolutionary process, and the entropy of the population during the evolutionary process. The RMSE was calculated using the best-performing individual in the population for each simulation. The

evolutionary process was repeated 100 times, i.e. 100 simulations, for each benchmark to ensure statistically meaningful results.

The results presented in Figure 4.22 depict the trajectory of the RMSE during the evolutionary process for both IGP and SGP across the nine benchmarks. The solid lines in the figure represent the median values, while the shaded areas denote the standard deviation intervals. By analyzing the results, it is clear that the IGP consistently outperformed or matched the SGP by converging more rapidly to the minimum.

The figures from 4.23 to 4.27 depict the RMSE median and standard deviation on the final solution for both test and train data. Once again, the IGP exhibits superior performance compared to the SGP, especially in terms of generalization capability. This trend is consistent across all benchmarks except for the koza1 benchmark, where the SGP achieves better results. This outcome might be attributed to the IGP's goal of promoting a more diverse population, leading to larger individuals, i.e. with more nodes, compared to the SGP population. These larger individuals may be overkill solutions for a straightforward test case like the koza1 benchmark, which could be more efficiently solved by smaller individuals.

As described in Subsection 4.4.2, the entropy measure was employed to assess the population's diversity. The results in Figure 4.28 illustrate the evolution of entropy throughout the evolutionary process for both the SGP and IGP. A close examination of the entropy evolution reveals that the IGP maintained an entropy value of around 2.30, which is the theoretical maximum, throughout the evolutionary process. In contrast, the entropy in the SGP tends to decrease towards the end of the evolutionary process, indicating a reduction in diversity in favour of better-performing individuals.

In summary, when analyzing both Figures 4.22 and 4.28, the presented results suggest that diversity maintenance indeed contributes to improving the performance of the GP algorithm. The IGP converges faster to a minimum, achieving better performance and generalization capabilities compared to the SGP.

As a concluding remark, Table 4.8 lists the mean and standard deviations of the computational times obtained with the SGP and IGP algorithms across the nine benchmarks. Since the IGP performs more operations than the SGP, higher computational

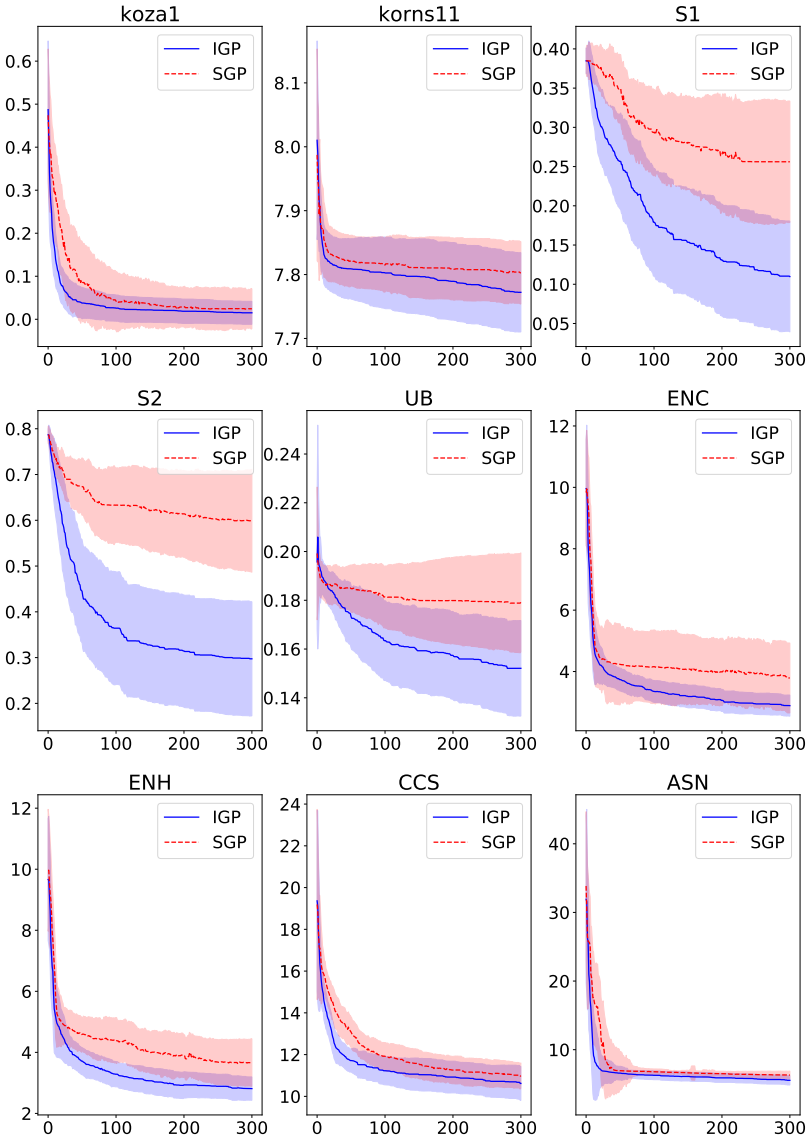
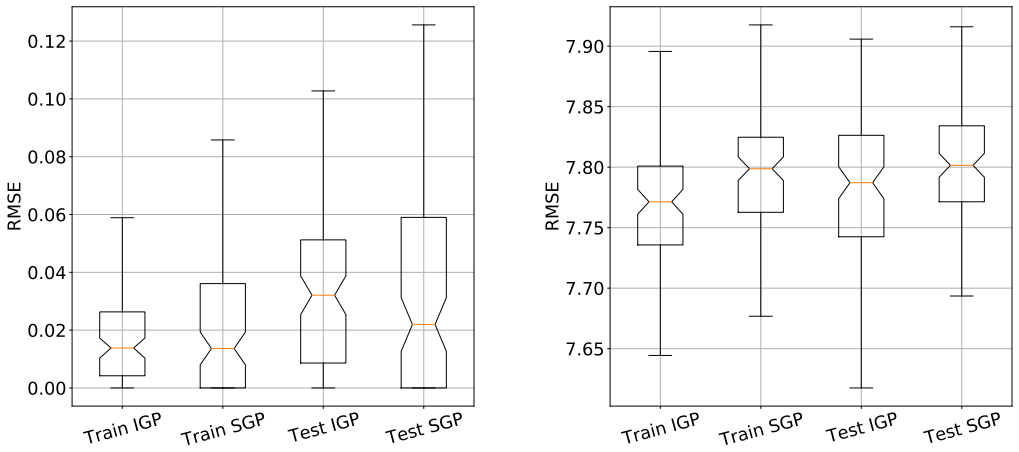
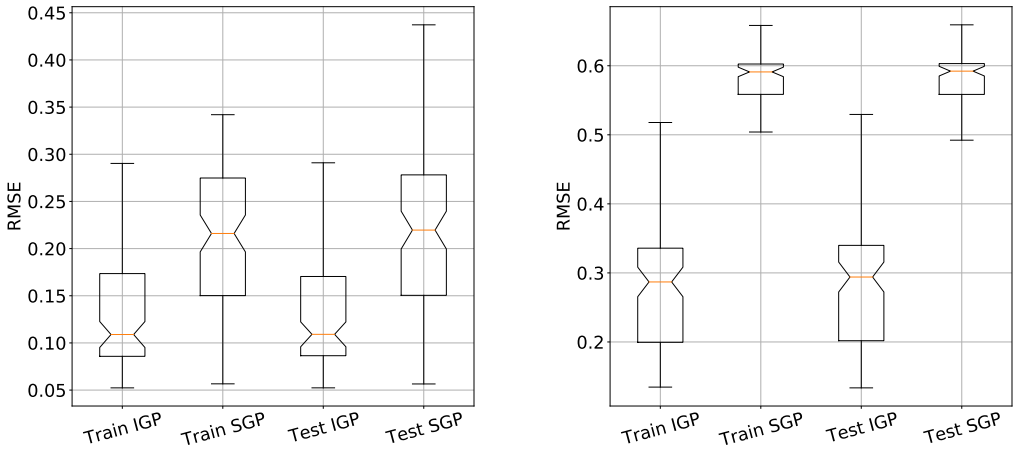


Figure 4.22: Fitness function values of the SGP and IGP algorithms on the nine different benchmarks. On the ordinate is the RMSE, while on the abscissa is the number of generations. The solid and dashed lines represent the median values for the IGP and SGP respectively, while the shaded regions are the standard deviation intervals [3]



(a) Benchmark Koza1 **(b)** Benchmark Korns11

Figure 4.23: RMSE of training and test data on synthetic benchmarks Koza1 and Korns11. The median and standard deviation values were evaluated over the results produced on 100 different runs. [3]



(a) Benchmark S1 **(b)** Benchmark S2

Figure 4.24: RMSE of training and test data on synthetic benchmarks S1 and S2. The median and standard deviation values were evaluated over the results produced on 100 different runs. [3]

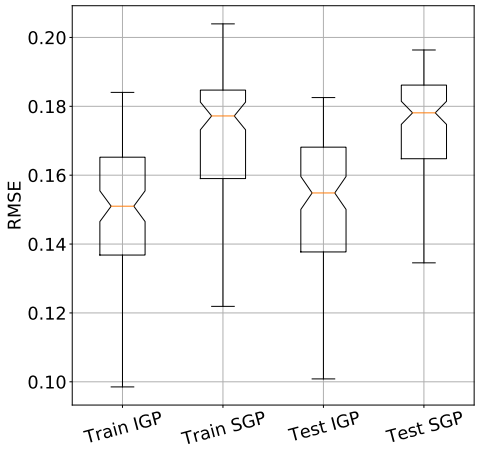
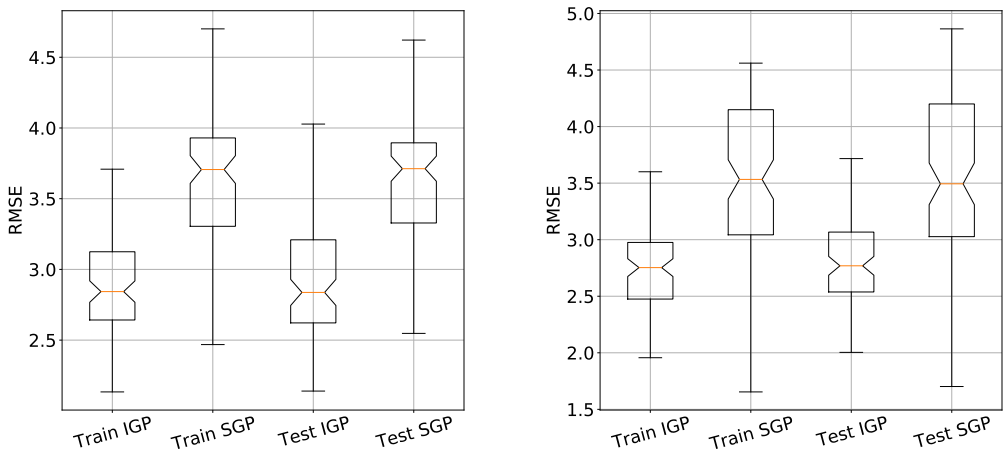
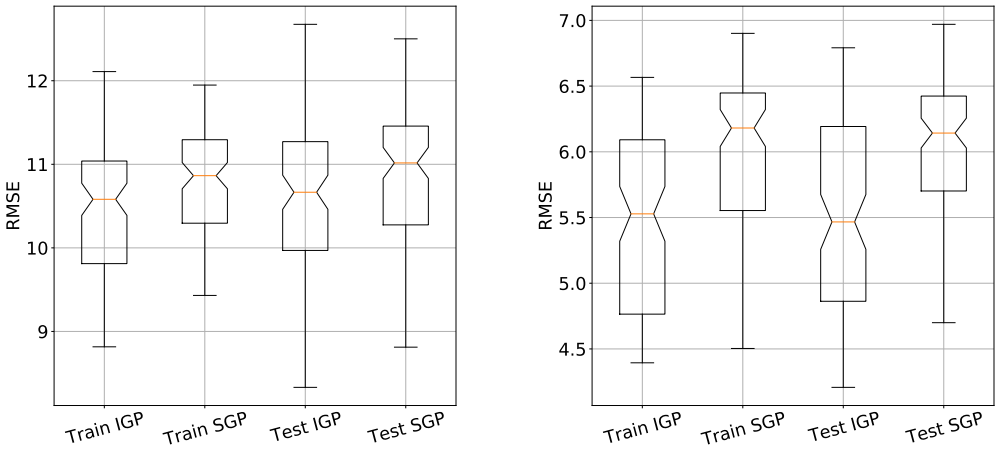


Figure 4.25: RMSE of training and test data on synthetic benchmark UB. The median and standard deviation values were evaluated over the results produced on 100 different runs. [3]



(a) Benchmark ENC **(b)** Benchmark ENH

Figure 4.26: RMSE of training and test data on real-world benchmarks ENC and ENH. The median and standard deviation values were evaluated over the results produced on 100 different runs. [3]



(a) Benchmark CCS **(b)** Benchmark ASN

Figure 4.27: RMSE of training and test data on real-world benchmarks CCS and ASN. The median and standard deviation values were evaluated over the results produced on 100 different runs. [3]

times were expected. This is true on some of the treated benchmarks, while for others the computational time is either comparable or smaller considering also the standard deviations. Therefore, by averaging on all the treated benchmarks it can be inferred that the enhancements introduced by the IGP do not come with a significant increase in computational time.

	Computational times [s]	
	SGP	IGP
Koza-1	44.83 ± 5.78	$\mathbf{17.29 \pm 1.97}$
Korns-11	$\mathbf{1079.10 \pm 745.43}$	1429.35 ± 814.94
S1	60.36 ± 17.77	$\mathbf{49.54 \pm 15.04}$
S2	$\mathbf{37.76 \pm 12.54}$	86.69 ± 18.17
UB	$\mathbf{108.96 \pm 75.18}$	198.18 ± 69.92
ENC	$\mathbf{108.60 \pm 88.69}$	147.58 ± 49.67
ENH	$\mathbf{132.43 \pm 420.11}$	141.09 ± 52.95
CCS	198.74 ± 75.022	$\mathbf{188.38 \pm 77.99}$
ASN	262.07 ± 179.45	$\mathbf{168.06 \pm 71.14}$

Table 4.8: Computational times

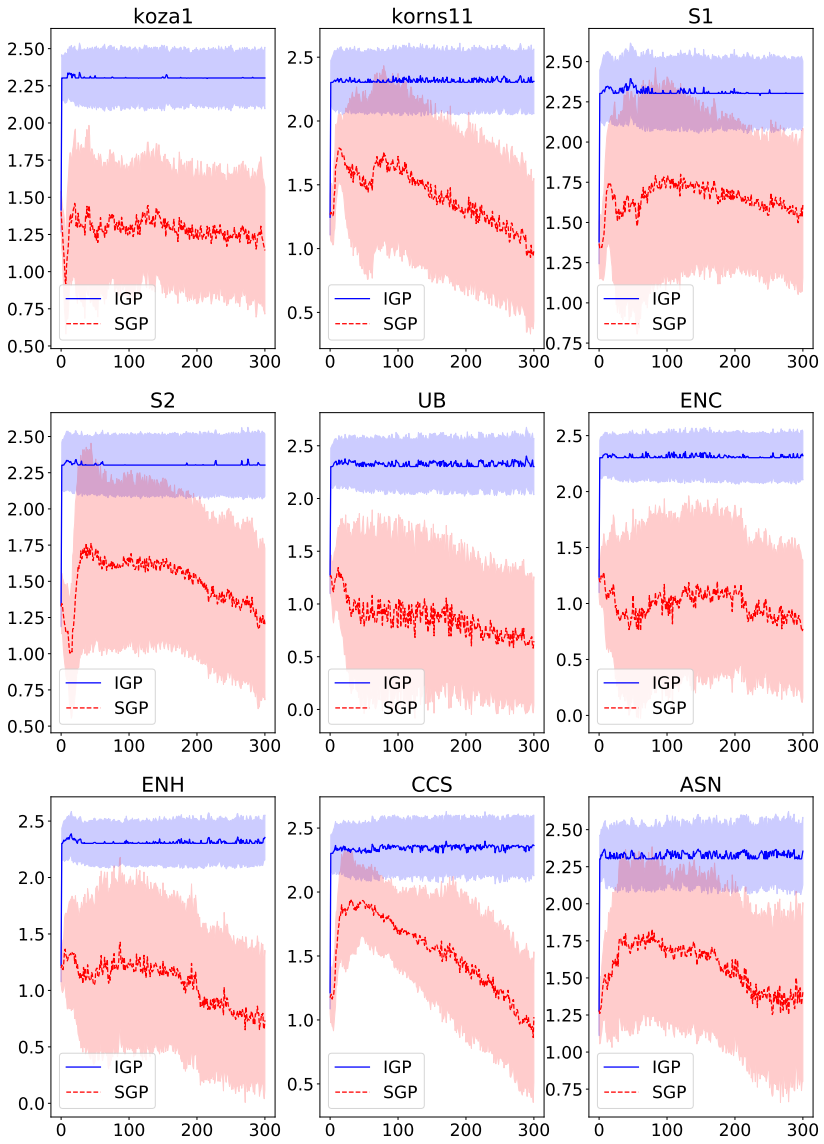


Figure 4.28: Entropy values of the SGP and IGP algorithms on the nine different benchmarks. On the ordinate is the Entropy, while on the abscissa is the number of generations. The solid and dashed lines represent the median values for the IGP and SGP respectively while the shaded regions are the standard deviation intervals [3]

4.5 Summary and Comments

This chapter provides an overview of Genetic Programming (GP) and its application in the domain of Intelligent Control (IC). It begins with an introduction to the theoretical foundations of GP and proceeds to present the necessary framework for deploying GP in the context of guidance or control tasks. Subsequently, an application of IC using GP is presented. Specifically, GP is employed in real-time to generate guidance commands for tracking the ascent trajectory of a Goddard rocket in the presence of various external disturbances. Three types of disturbances are introduced to assess the robustness of the proposed approach: a variation in the drag coefficient c_d, simulating changes in system parameters; a wind gust, emulating environmental disturbances; and a variation in the atmospheric model, representing a potential modelling error. The GP evolution is conducted online within a fixed time interval, and two success metrics are defined: the range success rate, indicating the GP's ability to track the reference trajectory, and the evaluation time success rate, denoting the frequency of successful GP evolutions within the designated time interval. The proposed approach proves its efficacy by successfully guiding the vehicle through all three disturbance scenarios, achieving an average range success rate of 83.7% and an average evaluation time success rate of 55% across these scenarios. A learning approach, where the GP algorithm learns from previous evolutions, was applied, resulting in an improvement of the evaluation time success rate by an average of 21%. These results underscore the potential of the proposed method, even when applied to a simplified problem.

Notably, for plants with a greater degree of nonlinearity, the online application of GP can be impractical due to extended evaluation times required to perform the trajectory propagation, potentially resulting in mission failure. To address this limitation and improve adaptability online, the Genetically Adapted Neural Network-based Intelligent Controller (GANNIC) approach outlined in chapter 5 was developed.

The chapter concludes with the description and test of a novel GP heuristic, the Inclusive Genetic Programming (IGP). Notably, the IGP, featured in chapter 5, is designed to enhance the performance of GP algorithms in control applications by pro-

moting and maintaining diversity within the population. To evaluate its efficacy, the IGP is tested across nine distinct benchmarks representing regression problems. Comparative analyses with a standard GP implementation reveal that the IGP achieves lower fitness values, i.e. better performance, quicker convergence, and enhanced generalization capabilities without a significant increase in the computational costs. These advantages are caused by the IGP's emphasis on promoting and maintaining the population's diversity. Diversity is monitored through the entropy measure, which is kept constant by the IGP throughout the evolutionary process, in contrast to the SGP that shows an entropy decrease, i.e. loss of diversity, towards the end of the evolutionary process.

Chapter 5

Genetically Adapted Neural Network-Based Intelligent Controller (GANNIC) [1]

5.1 Introduction

This thesis aims at investigating the applicability of Genetic Programming (GP) in an Intelligent Control (IC) setting. In doing so, the integration of GP with Neural Network (NN) is investigated, to enhance the online applicability of GP and overcome the limit posed by its computational cost. In chapter 4, GP was introduced, and its applicability in an IC setting was assessed. Subsequent experiments conducted in the course of this research revealed a significant growth of the GP's computational cost as the nonlinearity of the used models increased. This observation was particularly evident in another work produced during this thesis's development [4], where GP was employed to generate guidance commands for a Reusable Launch Vehicle (RLV), specifically angle of attack (α) and bank angle (σ). The results showcased GP's capability to produce guidance commands leading to successful mission completion under uncertainties in atmospheric and aerodynamic models. However, the computational cost, approximately five minutes per GP evolutionary process, made it impractical for online

[1] Part of the content of this chapter is submitted for publication [2]

applications with the current technology. For these reasons, the integration of GP and NN was investigated, resulting in the development of the Genetically Adapted Neural Network-based Intelligent Controller (Genetically Adapted Neural Network-based Intelligent Controller (GANNIC)).

The Genetically Adapted Neural Network-based Intelligent Controller (GANNIC) is a control framework designed with a versatile interface capable of issuing commands at both inner and outer control loop levels. Within the scope of this thesis, it was specifically formulated and implemented to operate at the outer loop level, assuming a perfect actuator response. From the perspective of the inner control loop, the GANNIC scheme functions as a real-time guidance mechanism, generating reference signals for certain desired variables to be tracked by the inner control loop. Furthermore, GANNIC was developed to guide a plant in the presence of external disturbances or uncertainties in the used models. This is achieved by leveraging the online learning capabilities provided by the combination of GP and NN. As elaborated in section 5.3.1, in the selected test case for GANNIC development and evaluation, the guidance commands comprised the angle of attack α and bank angle σ required to follow a desired trajectory in the presence of uncertainties in the environmental models. Examples of similar adaptive guidance schemes can be found in the literature, employing both conventional techniques and those augmented by Artificial Intelligence (AI). Examples not involving the use of AI are listed in section 2.5.3. When employing AI, many applications in the literature involve the use of NNs. As an example, Song et al. [210] proposed a real-time reentry guidance approach built using transformer networks trained to output the correct bank angle for a given flight state. Another real-time guidance approach relying on NNs is the one discussed in [16]. The authors developed a real-time guidance scheme for the landing phase of a reusable rocket, by combining two networks, one for classification and the other for regression. The classification network is used to classify the initial conditions of landing flights into different categories that correspond to different thrust profiles. Then the regression network is used to output the corresponding values of thrust to perform a successful landing. Recently, Reinforcement Learning (RL) has been increasingly used for adaptive guidance. Gaudet et al. applied a reinforcement

meta-learning algorithm in a series of works on planetary landing [211, 212], and more recently on the approach phase guidance of a hypersonic glider [213]. Their method involves applying reinforcement meta learning to optimize the adaptive guidance system, to output the angle of attack and bank angle rates according to the current flight conditions. Other examples of RL applied to perform the adaptive guidance of a RLV can be found in [214, 215]. Other AI techniques are more rarely used for this purpose. An example of Fuzzy Logic (FL) being applied for the adaptive guidance during the approach and landing phase of a RLV can be found in [216], while to the best of my knowledge, no applications can be found involving only Evolutionary Algorithms (EAs). In contrast EAs have been widely used to perform trajectory optimization, as done for example by [56], where the reentry trajectory of a RLV is optimized using Genetic Algorithm (GA).

The decision to employ a combination of GP and NN in designing GANNIC was influenced by the particular characteristics of each algorithm. GP is known for its exploratory capabilities, a characteristic extensively investigated in various applications, as discussed in the works of Schmidt and Lipson [217, 218]. On the other hand, NNs are universal nonlinear approximators, demonstrating the capability to learn nearly any data distribution provided an adequate number of neurons in the hidden layers [219, 220]. The combination of these two techniques is commonly observed in the domain of Neuroevolutionary control, where GP is employed to optimize the topology of a NN [221].

GANNIC introduces a novel way of combining of GP and NNs. GP is employed offline to discover a set of differential equations, which are then used to propagate the NN's weights along with the states of the considered plant. This approach can be interpreted as an adaptive Guidance and Control (G&C) scheme, where GP is employed to learn the algorithm for parameters' adaptation. According to the definition of IC provided in chapter 3, which involves the integration of AI and online learning, GANNIC can be characterized as an instance of IC. Furthermore, employing the taxonomy outlined in chapter 3, GANNIC falls under the classification of E2-C1-G1. Specifically, the controller is designed to address uncertainties linked to the plant's states (E2); the NN's

weights, representing the controller's parameters, undergo online adaptation (C1), and the trajectory tracking objective is conveyed to the GP algorithm as a fitness function to be minimized (G1).

The use of differential equations to update NN weights is not a novel concept in the literature. Notably, Chowdhary et al. [222] employed a Model Reference Adaptive Control (MRAC) control scheme featuring a single hidden layer NN as a controller. They devised two adaptation laws for the NN's weights - one for adapting the hidden layer weights and another for the output layer weights - ensuring their ultimate boundedness. These adaptation laws were formulated as differential equations. This approach traces its roots back to the foundational works of Lewis et al. [223, 224], where the stability and boundedness of NN weights were guaranteed through derived differential equations. However, the complexity of the mathematical formulation, originally developed for a simple robotic manipulator model, may render its application to more complex systems impractical, particularly those employing tabular models like most aerodynamic ones. Despite these challenges, various works have emerged over the years employing the approach presented in [223]. For instance, Zhang et al. [225] applied adaptive neural control to robotic manipulators with output constraints and uncertainties, designing a single adaptive law for all NN weights to ensure closed-loop system stability. In the work of Liu et al. [226], an adaptive neural controller for nonlinear multiple-input multiple-output systems was introduced. They used Radial Basis Function (RBF) networks to approximate unknown functions, determining weights through the backstepping technique, and demonstrating closed-loop system stability using the Lyapunov theorem. Another illustration is the work of Johnson and Calise [98], wherein they applied the adaptive laws introduced by Lewis [223] to update NN controller weights. The control system was implemented on a RLV, simulating various failure scenarios. Their approach effectively adapted to failure cases without necessitating direct knowledge of the failure, showcasing the adaptability of the technique.

Due to the inherent mathematical challenges associated with formulating adaptation laws, the application of GP offers a promising avenue for automating this complex process. This can prove invaluable for engineers treating highly nonlinear problems,

removing the need for linearization techniques and facilitating the application of a consistent design methodology across diverse plants. Consequently, the control approach outlined below represents a significant advancement towards achieving a fully automated and universally applicable controller design process.

To the best of the author's knowledge, the literature does not present the use of GP to find the adaptation laws in adaptive control or IC. There are other examples of GP being applied to find adaptation laws, such as in [227], where it is used to automatically design parameter adaptation techniques within a differential evolution algorithm.

This chapter is structured as follows. Section 5.2 provides a general description of the GANNIC framework without referring to any particular application while section 5.3 showcases the GANNIC application to the real-time guidance of a RLV during reentry. Section 5.4 presents a summary of the chapter with a discussion of the strengths and limitations of the GANNIC scheme.

5.2 GANNIC Framework Description

The GANNIC scheme design process, illustrated in Figure 5.1, comprises a series of steps that can be summarized as follows: 1) identify a reference trajectory for the considered mission; 2) define how uncertainties are applied; 3) perform the GP evolutionary process multiple times, first to optimize the NN topology (step 3.1), and then, to determine the final update laws for use with the optimal NN configuration (step 3.2); 4) test the obtained control scheme. The controller structure and the third design step are described in the following. The application of the whole design process is presented in section 5.3.1 referring to the treated test case. In the remainder of this section, GANNIC will be addressed as a control scheme and its output as control commands or control signals. This terminology is adopted for its generality, as the provided description applies to both control and real-time guidance applications.

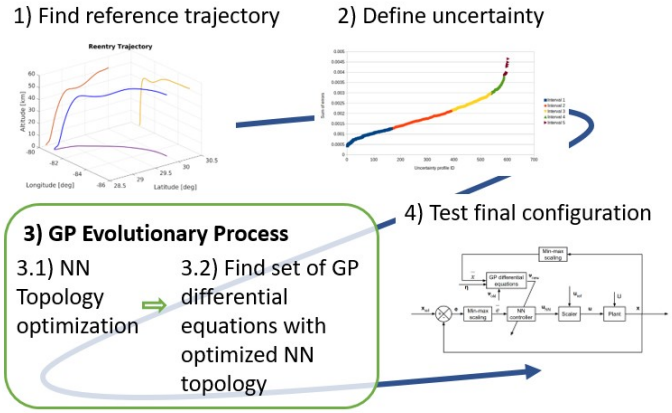


Figure 5.1: Graphical depiction of the GANNIC scheme design process

5.2.1 Scheme Structure

The generic GANNIC scheme is illustrated in Figure 5.2. The vector $\mathbf{x} \in \mathbb{R}^{N_s}$ comprises the state variables, where N_s is the number of state variables. The vector $\mathbf{u} \in \mathbb{R}^{N_u}$ encompasses the control commands, with N_u denoting the number of control commands. Control signals are obtained through the interaction between the NN output $\mathbf{u}_{NN} \in \mathbb{R}^{N_o}$, where N_o is the number of output neurons, and the reference control signal $\mathbf{u}_{ref} \in \mathbb{R}^{N_u}$, as further explained below. The vector $\mathbf{e} \in \mathbb{R}^{N_s}$ denotes the tracking errors on the states, computed as $\mathbf{e} = \mathbf{x}_{ref} - \mathbf{x}$, where $\mathbf{x}_{ref} \in \mathbb{R}^{N_s}$ is the vector of reference state variables. The vectors $\bar{\mathbf{x}} \in \mathbb{R}^{N_s}$ and $\bar{\mathbf{e}} \in \mathbb{R}^{N_s}$ represent the scaled states and tracking errors, respectively. The vector $\boldsymbol{\eta}$ encompasses environmental variables, which are problem-dependent and can be omitted. The term U represents uncertainties applied to the plant. The vectors $\boldsymbol{\nu}_{new}$ and $\boldsymbol{\nu}_{old}$ represent the NN's weights at the current and previous time steps, respectively.

A NN featuring a single hidden layer is employed, and its outputs are defined according to Equation 5.1, using the terminology adopted by Lewis [228].

$$u_{NN_i} = \sum_{j=1}^{N_h} \left[w_{ij} \phi \Big(\sum_{k=1}^{N_i} v_{jk} y_k + \theta_{v_j} \Big) \right] + \theta_{w_i}; i = 1, ..., N_o \tag{5.1}$$

In Equation 5.1 u_{NN_i} is an element of the vector $\mathbf{u}_{NN} \in \mathbb{R}^{N_o}$ containing the NN outputs. w_{ij} is an element of the matrix $\mathbf{W} \in \mathbb{R}^{N_o \times N_h}$ representing the weights con-

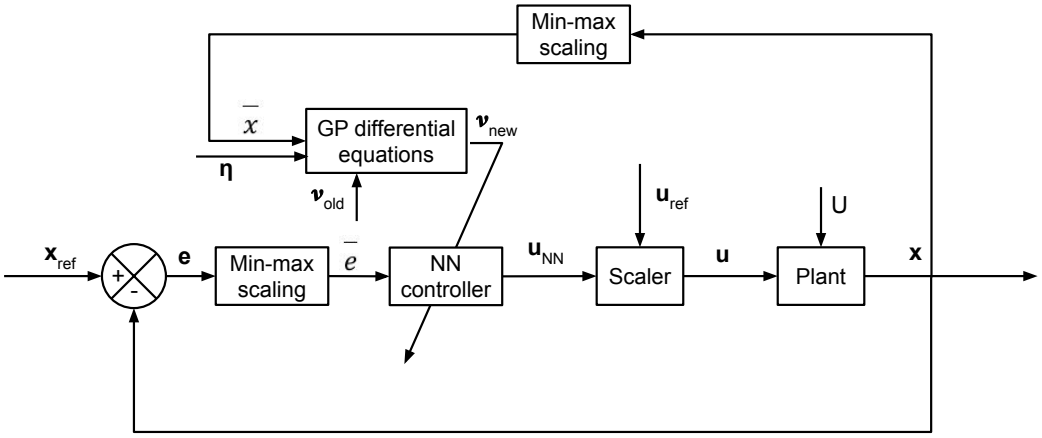


Figure 5.2: Diagram of the GANNIC scheme [2]

necting the hidden layer with the output layer. N_h represents the number of neurons in the hidden layer. ϕ is the NN's activation function. v_{jk} is an element of the matrix $\mathbf{V} \in \mathbb{R}^{N_h \times N_i}$ representing the weights connecting the input layer with the hidden layer, where N_i is the number of input variables; and $\mathbf{x} \in \mathbb{R}^{N_i}$ is the vector of input variables. $\boldsymbol{\theta}_v$ and $\boldsymbol{\theta}_w$ represent the biases applied to the hidden and output layers, respectively. In this work, no distinction has been made between the weights connecting the input layer to the hidden layer and those connecting the hidden layer to the output layer; therefore, they can be grouped in one vector $\boldsymbol{\nu}$. Also, the biases are treated equally to the weights and can be grouped in the vector $\boldsymbol{\nu} \in \mathbb{R}^{N_\nu}$ with $N_\nu = N_h(N_i + 1) + N_o(N_h + 1)$. Thus, for clarity, in the rest of this chapter both NN's weights and biases will be referred to as weights.

The input of the NN corresponds to the scaled tracking errors on the states, computed using a min-max scaling from their original range to the target range $[0, 1]$, as expressed in Equation 5.2. Here, **LB** and **UB** respectively represent the maximum and minimum permissible values for the state variables. This scaling is implemented based on literature recommendations [115], suggesting that NN performance tends to improve with a small and bounded input. For this configuration, $N_i = N_s$, $\mathbf{y} \equiv \bar{\mathbf{e}}$, and

$N_o = N_u$, where one output neuron is used for each control command.

$$\bar{\mathbf{e}} = \frac{\mathbf{x} - \mathbf{x}_{ref}}{\mathbf{UB} - \mathbf{LB}} \tag{5.2}$$

The weights of the NN, denoted by $\boldsymbol{\nu}$, are not static but are learned online by propagating them using a set of differential equations determined offline through GP. Consequently, the resulting dynamical system can be expressed in general form as presented in Equation 5.3. In this formulation, the states are propagated while considering the states-dependent uncertainty $U(\mathbf{x})$, also referred to as the uncertainty scenario. As elucidated in the subsequent subsections, a collection of uncertainty scenarios is employed during the GP evolutionary process, resulting in a vector $\mathbf{U}(\mathbf{x}) \in \mathbb{R}^{N_U}$, where N_U is the number of the considered uncertainty scenarios.

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}(t), \mathbf{u}(t), U(\mathbf{x})) \\ \dot{\boldsymbol{\nu}} &= f_{GP}(\mathbf{x}(t), \boldsymbol{\eta}(\mathbf{x}), \nu(t)) \end{aligned} \tag{5.3}$$

N_ν GP equations are evolved simultaneously therefore a single differential equation can be written as in Equation 5.4.

$$\dot{\nu}_i = f_{GP_i}(\mathbf{x}(t), \boldsymbol{\eta}(\mathbf{x}), \nu_i(t));\ i = 1, ..., N_\nu \tag{5.4}$$

The input for each GP equation encompasses the states, a set of environmental variables $\boldsymbol{\eta}$ and the corresponding weight. However, it is essential to note that the GP algorithm evolves models that consider influential inputs. Therefore, the final models might not incorporate all the input initially made available to the GP algorithm

To summarize, by propagating the dynamical system in Equation 5.3, the NN's weights are updated at each time step, leading to a control action $\mathbf{u}_{NN}$ employed to control the plant. However, this control action is not used directly, as it could assume extremely large values, especially at the start of the GP evolutionary process. This is attributed to the GP differential equations, which might produce exceedingly large

values for the weights, resulting in numerical errors in the propagation and potential failure of the evolutionary process. Consequently, the control action $\mathbf{u}_{NN}$ is passed through a scaler.

The scaler is a function employed to constrain and subsequently scale the output of the NN to the range $[\mathbf{u}_{ref}(1-\epsilon_S), \mathbf{u}_{ref}(1+\epsilon_S)]$, where $\mathbf{u}_{ref}$ denotes the reference control signals, and ϵ_S is a distribution parameter proportional to the sum of the scaled tracking errors on the state variables, assessed as presented in Equation 5.5. Here, u_{b_S} and l_{b_S} represent the maximum and minimum variations, expressed as percentages from the reference value. For instance, if $l_{b_S} = 0.01$ and $u_{b_S} = 0.5$, it signifies that the final control action will lie within the range [1%, 50%] from the reference control value.

$$\epsilon_S(t) = \sum_{i=1}^{N_s} (l_{b_S}(1 - |\bar{e}_i(t)|) + u_{b_S}|\bar{e}_i(t)|) \tag{5.5}$$

The $\epsilon_S(t)$ function allows for a broader range of variation, i.e. more significant control actions, when the tracking error is big and vice versa. This means that when the vehicle is on the optimal trajectory, i.e. $\sum \bar{\mathbf{e}} = 0$, the open loop control action $\mathbf{u}_{ref}$ is sufficient.

To perform the scaling, the NN output must be first bounded to a permissible range. To do so, the NN output is passed through the tanh function, which is modified with the parameters p_1 and p_2 to change its shape. p_1 defines the bounds of the output, while p_2 is the slope of the linear portion of the function.

The constrained output is consequently linearly scaled from [-p_1, p_1] to $[\mathbf{u}_{ref}(1-\epsilon_S), \mathbf{u}_{ref}(1+\epsilon_S)]$ using Equation 5.6.

$$\mathbf{u}_{scaled} = \mathbf{u}_{ref}(1 + \epsilon_S \tanh(p_2 \mathbf{u}_{NN})) \tag{5.6}$$

5.2.2 Genetic Programming Evolutionary Process

The GANNIC approach was developed using the Inclusive Genetic Programming (IGP) presented in section 4.4. Therefore, its usage is recommended to achieve similar performances. Nevertheless, other GP algorithms can be employed to find the NN weights'

adaptation laws, but they must possess three key features: the ability to evolve multiple GP trees simultaneously to evolve one adaptive law for each NN's weight; the capability of handling constraints; and the capacity to be applied to Multiple-Input Multiple-Output (MIMO) systems. The IGP can be easily configured to perform these tasks, thanks to the flexibility of the DEAP library [197] used to code it. Other GP architectures might fail in doing so. For example, a Multi-Gene Genetic Programming (MGGP) [229] algorithm would be impractical in a control setting. The least square optimization performed in the MGGP to evaluate each subtree is performed by minimizing the error between the target and the GP output, which is not composed of a single quantity in control applications. Multiple states are tracked simultaneously, and a way of grouping them must be employed. Moreover, the MGGP least square optimization is fast in classical regression problems, where the subtrees are evaluated on the dataset. But in a control setting, it would require the dynamical system propagation using each subtree to find their fitness value. This results in an extremely slow process.

The fitness evaluation process is summarized in Algorithm 8. Briefly, a fitness composed of two elements is assigned to each individual, $\mathbf{F}_{ind} = [F_f, P_f]$. F_f represents the true objective of the evolutionary process, i.e. reach a target final position, while P_f is a penalty parameter accounting for constraints violations. If constraints are not considered, P_f is set to 0. The IGP algorithm was configured, as described in section 4.4, to handle constraints.

F_f and P_f are computed considering the performance of the individual across N_U uncertainty scenarios. These uncertainty scenarios are previously chosen and do not change during the evolutionary process. Subsection 5.2.3 explains the process adopted to select the uncertainty scenarios. Incorporating multiple uncertainty scenarios into the fitness evaluation enhances the evolved individuals' robustness to unforeseen uncertainties. This robustness is directly proportional to the number of uncertainty scenarios considered during the evolutionary training phase. Nonetheless, it is important to acknowledge that a higher number of uncertainty scenarios implies a proportional increase in computational resources and time required for the evolutionary process.

Algorithm 8 Pseudocode of the fitness evaluation process

Load N_U uncertainty scenarios
Initialize $\mathbf{F}_s$ vector as empty
Initialize $\mathbf{P}_s$ vector as empty
for $i = 1 \rightarrow N_U$ **do**
    Selected i-th uncertainty scenario U_i
    Initialize $\mathbf{P}$ vector as empty
    Propagate system's dynamics in Equation 5.3, resulting in a trajectory of N_p points
    if $c_{v_{kj}} > 0; k = 1, ..., N_c, j = 1, ..., N_p$, where $c_{v_{kj}}$ is the constraints violation of the considered quantity and N_c is the number of constrained quantities. **then**,
        Append $c_{v_{kj}}$ to $\mathbf{P}$
    end if
    Evaluate P_u as $P_u = RMSE(\mathbf{P})$
    Evaluate Fitness function F_u as $F_u = MSE(\mathbf{F})$
    Append F_u to $\mathbf{F}_s$
    Append P_u to $\mathbf{P}_s$
end for
Evaluate Fitness F_f as in Equation 5.8 and penalty P_f as $P_f = MSE(\mathbf{P}_s)$.
Output $\mathbf{F}_{ind} = [F_f, P_f]$

A detailed description of the F_f and P_f evaluation process is provided in the following,

Objective Function F_f

The objective function F_f is designed to guide the plant toward a desired final position when uncertainties are present. To achieve this, the scaled tracking errors at the final point of the trajectory are considered, along with the integrals of the absolute scaled tracking errors on a final portion of the trajectory. The scaled tracking errors are meant to provide direct information about the ability of the GP individual to guide the plant towards the desired final position. On the other hand, the integrals computed on the final portion of the trajectory are meant to guide the GP evolutionary process towards well behaving individuals, providing information on the trajectory tracking capabilities in the proximity of the final position. As previously described, these operations are performed on N_U trajectories obtained considering N_U uncertainty scenarios.

The scaled tracking errors are computed by performing a min-max scaling of the

tracking errors, as in Equation 5.2. These are evaluated for each state, and their values on the final position are grouped in the vector $\bar{\mathbf{e}}_f \in \mathbb{R}^{N_s}$, with N_s is the number of state variables. The integrals of the absolute scaled tracking errors on the chosen portion of the trajectory are computed as in Equation 5.7. Here, $|\bar{e}_i(t)|$ is the absolute value of the i-th state scaled tracking error at time t. t_f is the final time of the trajectory, while t_{start} represents the initial time of the chosen portion. This integral aims at minimizing the discrepancy between the obtained trajectory and the reference one in their final part, to better guide the plant towards the desired final state. Therefore, t_{start} must be chosen close enough to the final time. As an example, in the application described in Section 5.3, the integrals are computed on the last 30% of the trajectory.

$$I_i = \int_{t_{start}}^{t_f} |\bar{e}_i(t)|dt, \;\; i = 1, ..., N_s \tag{5.7}$$

An integral is computed for each state variable, resulting in the vector $\mathbf{I} = [I_1, ..., I_{N_s}]$, where N_s is the number of state variables.

Using the vector of scaled tracking error on the final position $\bar{\mathbf{e}}_f$ and the vector of integrals on the final portion of trajectory $\mathbf{I}$, the vector $\mathbf{F} = [\mathbf{I}/s_f, \bar{\mathbf{e}}_f \mathbf{w}_s^T s_f] \in \mathbb{R}^{2N_s}$ can be computed. As the fitness function aims to guide the plant towards the desired final state, the vectors $\mathbf{I}$ and $\bar{\mathbf{e}}_f$ are scaled by a scalar scaling factor s_f to prioritize small tracking errors at the final position rather than small tracking errors in the last portion of the trajectory. Additionally, a vector of weights $\mathbf{w}_s \in \mathbb{R}^{N_s}$ is used to increase the value of the error on the desired states to prioritize their minimization during the evolutionary process.

The Mean Squared Error (MSE) of the vector $\mathbf{F}$ is then computed to obtain the fitness of each uncertainty scenario, defined as $F_u = \frac{1}{2N_s} \sum_{i=1}^{2N_s} F_i^2$, where $2N_s$ is the number of elements in the vector $\mathbf{F}$. Once the fitness F_u for each uncertainty scenario is evaluated, these are added to the vector $\mathbf{F}_s = [Fu_1, Fu_2, ..., F_{u_{N_U}}] \in \mathbb{R}^{N_U}$, where N_U is the number of uncertainty scenarios used for training.

The final fitness F_f for the considered individual is then evaluated as in Equation 5.8. $\mathbf{S} \in \mathbb{R}^{N_U}$ is a vector of ones and zeros used to track the uncertainty scenarios solved successfully. The success criterion is problem dependent. As an example, an

uncertainty scenario can be considered solved successfully if the tracked states are at the desired final position or within the tolerance interval from the final position and all constraints are satisfied.

$$F_f = \frac{max(\mathbf{F}_s) + median(\mathbf{F}_s)}{max(\sum \mathbf{S}, 1)} \tag{5.8}$$

If no uncertainty scenario is solved successfully, the denominator of Equation 5.8 is set to one. Equation 5.8 considers both the maximum and median of the fitnesses found in each uncertainty scenario to consider both the average performance and the behaviour in the most challenging uncertainty scenario. Moreover, the fitness is divided by the number of successes to reward those individuals who are more successful. Such a fitness function is designed to guide the evolution towards individuals who can successfully solve an increasingly greater number of uncertainty scenarios and perform well even in those scenarios where they fail.

Constraints Violation Function P_f

As summarized in Algorithm 8, a penalty vector $\mathbf{P}$ is initialized as empty for each considered uncertainty scenario and it is extended with the values of the violated constraints at each point of the trajectory. If the constrained quantities are grouped in the vector $\mathbf{c}$, and the vectors of upper and lower bounds are defined as $\mathbf{c}_{max}$ and $\mathbf{c}_{min}$, the vector of constraints violations $\mathbf{c}_v$ will be $\mathbf{c}_v = [max(0, \mathbf{c}_1 - \mathbf{c}_{1_{max}}), max(0, \mathbf{c}_{1_{min}} - \mathbf{c}_1), ..., max(0, \mathbf{c}_{N_p} - \mathbf{c}_{N_{p_{max}}}), max(0, \mathbf{c}_{N_{p_{min}}} - \mathbf{c}_{N_p})]$ where N_p is the number of points in the propagated trajectory. Only the elements of $\mathbf{c}_v$ greater than zero are added to the vector $\mathbf{P}$. Therefore, the number of elements in the vector $\mathbf{P}$ is not known a priori, but it changes for every trajectory according to the violated constraints.

At the end of each trajectory propagation performed using different uncertainty scenarios, the total penalty P_u is evaluated as the Root Mean Squared Error (RMSE) of $\mathbf{P}$, such as $P_u = \sqrt{\frac{1}{N}\sum_{i=1}^{N} P_i^2}$, where N is the number of elements in the vector $\mathbf{P}$. P_u is evaluated for each considered uncertainty scenario, and it is added to a vector $P_s \in \mathbb{R}^{N_U}$ such as $\mathbf{P}_s = [P_{u_1}, ..., P_{u_{N_U}}]$. After all the runs on the uncertainty scenarios are performed, the final penalty P_f for the considered individual is evaluated

as $P_f = \frac{1}{N_U} \sum_{i=1}^{N_U} P_{s_i}^2$.

If for each run performed on the different uncertainty scenarios, the $\mathbf{P}$ vector is empty, the constraints are always satisfied. Regarding the use of RMSE and MSE to evaluate P_u and P_f, RMSE is used to have a more direct representation of the constraints violation for each considered uncertainty scenario, while the MSE is used to average the results over different uncertainty scenarios while giving importance also to the outliers. In fact, the constraints violation function aims to have individuals who can satisfy constraints on all the considered uncertainty scenarios.

5.2.3 Training Uncertainty Scenarios Selection

To evolve a controller capable of maintaining good control capabilities in the presence of unforeseen uncertainties, it is essential to consider various uncertainty scenarios simultaneously during the training phase. The evaluation of fitness functions F_f and P_f takes into account the control system's performance across multiple uncertainty scenarios. The way uncertainty scenarios are selected and applied significantly influences the GP evolutionary process, and inappropriate choices may lead to failure. In the following discussion, the term "uncertainty scenario" is used in a general sense, referring to a user-defined formulation tailored to the specific problem under consideration. The particular uncertainty formulation employed for the test case analyzed in this chapter is detailed in Subsection 5.3.4.

The method of applying uncertainty is problem-dependent. However, the impact of uncertainty on the propagated trajectory can be systematically studied across various test cases, enabling the classification of uncertainties based on their influence on the considered plant. To achieve this, two sets of uncertainty scenarios are generated: one for training, denoted as $\mathbf{U}_{train}$, and the other for testing, denoted as $\mathbf{U}_{test}$. Both sets consist of a user-defined number of uncertainty scenarios. The scenarios in the training set are employed to propagate the dynamic system described by Equation 5.9 a total of N_{train} times, resulting in N_{train} trajectories. These trajectories are generated using the reference control actions $\mathbf{u}_{ref}$ derived from the optimal trajectory.

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}_{ref}(t), U_{train_i}(\mathbf{x})); i = 1, ..., N_{train} \tag{5.9}$$

To classify uncertainty scenarios, it is important to define a magnitude metric denoted as μ. The specific definition of the magnitude metric is contingent on the problem at hand. Once the magnitude metric μ is computed for each scenario, they can be ranked and subsequently grouped based on their magnitude metric values. The subdivision into groups is accomplished by determining the maximum and minimum values of the magnitude metrics and then linearly dividing this interval into g groups. Each group is defined by an upper and a lower bound, encompassing all the uncertainty scenarios with a μ value falling within these bounds. After creating the groups, the uncertainty scenario with the highest magnitude metric, i.e. the worst case scenarios, is selected from each uncertainty group and used during the training phase, i.e. the GP evolutionary process.

In cases akin to the one examined in this thesis, a magnitude metric that quantifies the sum of absolute scaled tracking errors on the final position can be employed, as expressed in Equation 5.10.

$$\mu = \sum_{i=1}^{N_s} |\bar{e}_i(t_f)| \tag{5.10}$$

Figure 5.3 provides an illustrative example of the ranked uncertainty scenarios, where the magnitude metric was determined using Equation 5.10 and five groups were created.

The uncertainty scenarios selection process is summarized in Algorithm 9.

5.2.4 Neural Network Settings and Topology Optimization

The NN is employed as a nonlinear agent. As such, its inputs are the scaled tracking errors evaluated with Equation 5.2, and it features one output neuron for each control command. Its topology must be defined according to the studied test case to have enough layers and neurons to capture the problem's nonlinearity but not too many, resulting in an unsuccessful GP evolutionary process due to an excess of GP differential

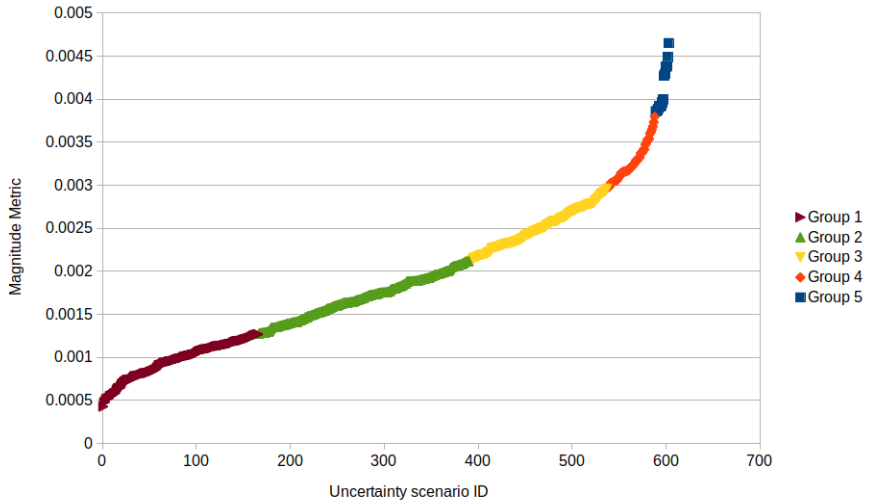


Figure 5.3: Example of ranked uncertainty scenarios divided into five groups. Each point represents an uncertainty scenario.

Algorithm 9 Pseudocode of uncertainty scenario selection process

1: Define how the uncertainty is applied and create two uncertainty sets, $\mathbf{U}_{train}$ and $\mathbf{U}_{test}$. Use $\mathbf{U}_{train}$ for the operations described hereafter.
2: Define a magnitude metric μ according to the considered test case.
3: **for** $i = 1 \rightarrow N_{train}$ **do**
4: Perform a trajectory propagation applying the i-th uncertainty scenario using reference control commands u_{ref}
5: Evaluate the magnitude metric μ
6: **end for**
7: Linearly subdivide the uncertainty scenarios into g groups according to their magnitude metric μ
8: From each uncertainty group, pick the one with max μ
9: Use the selected uncertainty scenarios in the GP evolutionary process.

equations. A heuristic designed to optimally prune the NN topology is outlined in Algorithm 10.

An example of the operations described in Algorithm 10 is illustrated hereafter. For exposition's sake, $N_{evolutions} = 1$. $N_{test} = 100$, meaning that the control system is tested on a batch of 100 uncertainty scenarios. A NN composed of six inputs, one hidden layer with four neurons and two outputs is considered; hence a total of 38 weights are used. This initial configuration is depicted in Figure 5.4.

As described in line 7 of Algorithm 10, each of the 38 weights is set to zero one by

Algorithm 10 Pseudocode of NN topology pruning heuristic

1: Define an initial NN configuration.
2: **for** $i = 1 \rightarrow N_{evolutions}$ **do**
3: Perform i-th GP evolution.
4: Select the best-performing individual
5: Perform N_{test} trajectory propagations, using N_{test} uncertainty scenarios from the uncertainty test set $\mathbf{U}_{test}$
6: **for** $j = 1 \rightarrow N_{\nu}$ **do**
7: Set to 0 the j-th weight of the NN and repeat line 5.
8: Keep track of the number of uncertainty scenarios successfully solved
9: **end for**
10: **end for**
11: Prune the NN according to the obtained results, i.e. remove those weights that, when set to 0, result in the same or a higher success ratio as the one obtained in line 5.

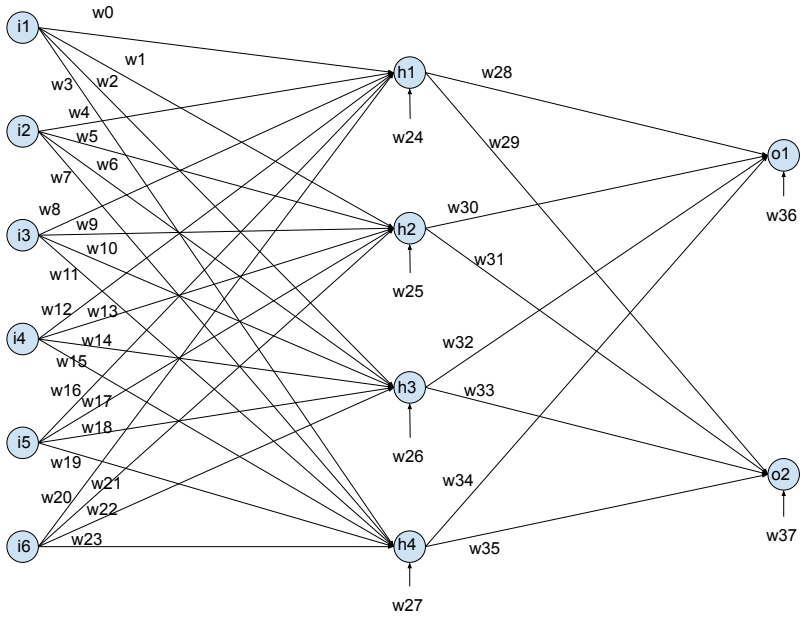


Figure 5.4: NN initial configuration

one, and N_{test} trajectory propagations are performed using the uncertainty scenarios from the uncertainty set $\mathbf{U}_{test}$. As an example, the controller with the full NN achieved a success ratio of 85% on the test uncertainty scenarios. The results obtained by setting the weights to 0 one by one are summarized in Figure 5.5. In this Figure, the critical weights, highlighted in red, are those that, when set to 0, lead to a deterioration in the

NN's performance.

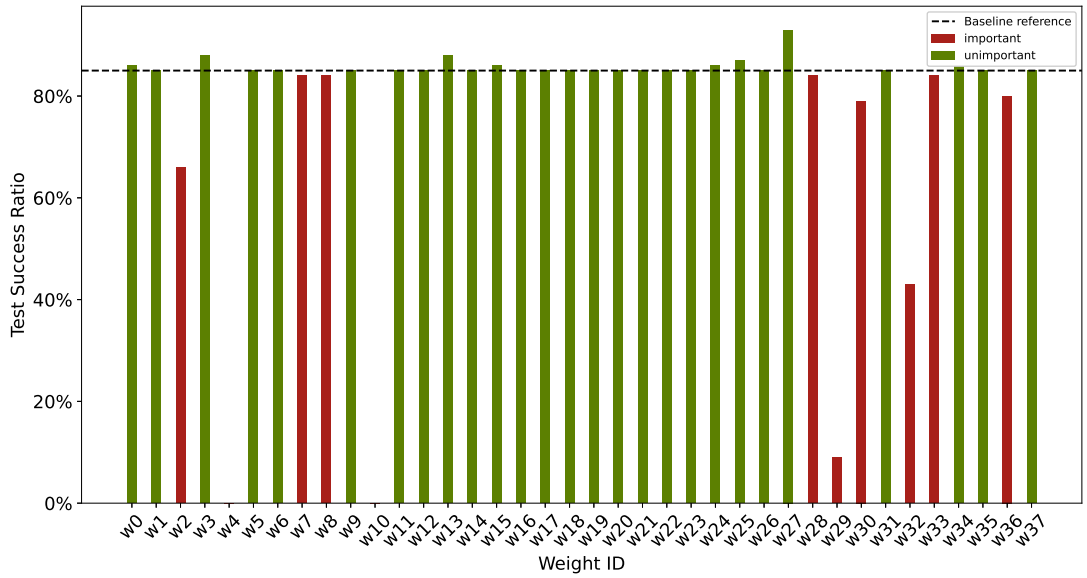


Figure 5.5: Success ratios of the control system when setting NN's weights to zero one by one

As can be seen in Figure 5.5, many of the weights can be set to 0, i.e. the green bars, and still, the controller is capable of achieving the baseline performances or better, as for the weights 0, 3, 13, 15, 24, 25, 27, 34.

Figure 5.6 illustrates the NN weights map. This Figure depicts the location of the NN weights and those non influential are not depicted. This outcome provides valuable insights since only three of the six inputs are useful, and of the four hidden neurons, only two are fully connected between the input and output. If similar patterns emerge when analyzing GP individuals obtained from multiple evolutions, it is safe to reduce the NN configuration from 6-4-2 (input-hidden-output neurons) to 3-2-2. This, in turn, will result in a faster GP evolution since fewer GP trees are simultaneously evolved, and better performances are achieved as the GP focuses on evolving only the differential equations for the valuable weights.

Algorithm 10 can be repeated multiple times by setting the initial configuration as the one obtained in the previous iteration. Using this approach, the NN will be gradually pruned until the final configuration is found. A final topology is reached when all weights are used, meaning the performances will worsen if any of them are set

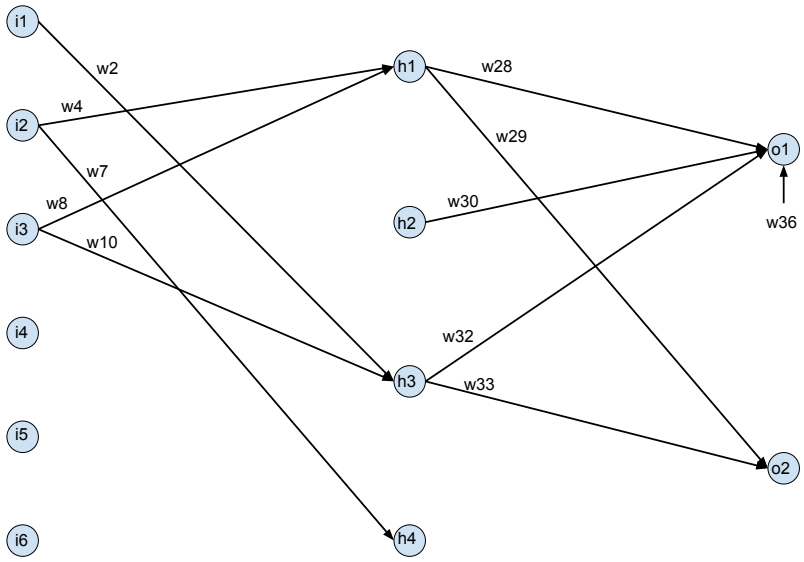


Figure 5.6: Graphical depiction of the NN weights location and their influence. In particular, the non influential weights are omitted. This kind of plot is defined as a weights map.

to zero.

5.3 Application

This section outlines the application of the GANNIC scheme for real-time guidance of a RLV during reentry. In this context, GANNIC is used to generate real-time guidance commands for the angle of attack α and bank angle σ during the flight. From this point onward, the term "guidance scheme" will be used to refer to GANNIC, and its output will be denoted as guidance commands or signals.

5.3.1 Test Case - FESTIP-FSSC5 Reentry Vehicle

The selected vehicle is the FESTIP-FSSC5 RLV, presented in Figure 5.7. This vehicle was developed by European Space Agency (ESA) as part of the Future European Space Transportation Investigation Programme (FESTIP) program, which aimed to conduct a feasibility study on various new RLV designs [230, 231]. The FESTIP-FSSC5 model

draws inspiration from the NASA Venturestar concept and its technology demonstrator, the X-33 experimental vehicle. Featuring a lifting body configuration and an aerospike engine, the FESTIP-FSSC5 was designed for vertical lift-off and horizontal landing. Notably, it is engineered to complete the entire mission, encompassing both atmospheric and space travel, without relying on multiple propulsion systems, thanks to the aerospike engine.

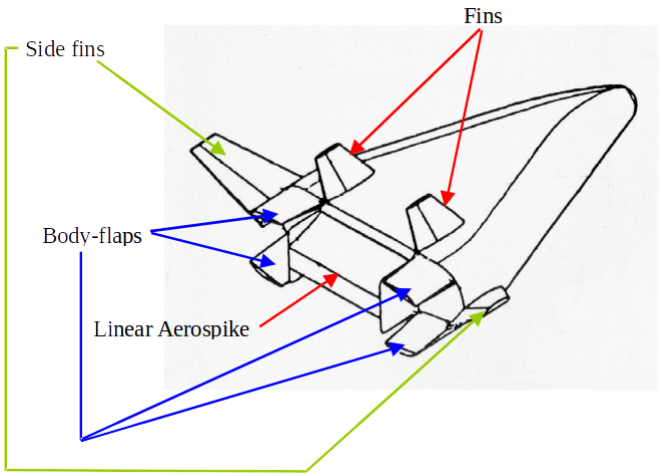


Figure 5.7: FESTIP-FSSC5 reusable launch vehicle [5]

Both the X-33 and FESTIP-FSCC5 projects were never realized due to technological and economic limitations. Concerning the FESTIP-FSSC5, "the design team was disappointed to discover that this concept, inspired by the Venturestar geometry, cannot be made both feasible and economically viable with the technology presently available or foreseeable in the near term in Europe" [230]. Several technological challenges contributed to this outcome, including the design of lightweight cryogenic tanks with the desired structural characteristics, underdeveloped technology for the aerospike engine, and difficulties in achieving accurate guidance during reentry caused by the unique shape of the vehicle, which impacted both aerodynamics and controllability [232, 233].

Given the recent interest of the aerospace industry towards reusability, RLVs with a space plane configuration [2] and aerospike engines [3], the FESTIP-FSSC5 has been

[2] https://www.space.com/polaris-spaceplanes-mira-light-flight-test-campaign-complete
[3] https://spacenews.com/pangea-aerospace-tests-aerospike-engine/

considered an appropriate candidate to test the GANNIC scheme.

The vehicle's models are taken from [5]. The vehicle is modelled as a rigid body flying over a spherical non-rotating Earth. The Earth's rotation was neglected for two reasons. First, a short duration flight is considered, as will be explained further in the text. Second, using a set of equations of motion with a lower degree of nonlinearity helps reduce the overall computational complexity without affecting the results. No wind is considered in the atmosphere, and the reentry is unpowered, i.e. the thrust is set to zero. The vehicle's flight is studied using six state variables V (speed), χ (heading angle), γ (flight path angle), θ (longitude), λ (latitude), h (altitude) and two guidance commands α (angle of attack) and σ (bank angle), therefore the vector of state variables is defined as $\mathbf{x} = [V, \chi, \gamma, \theta, \lambda, h]$, and the vector of guidance signals is defined as $\mathbf{u} = [\alpha, \sigma]$. The 3 Degrees-of-Freedom model is shown in Equation 5.11.

$$\begin{aligned}
\dot{V} =& -\frac{D}{m} - g\sin\gamma \\
\dot{\chi} =& \frac{L\sin\sigma}{mV\cos\gamma} - \cos\gamma\cos\chi\tan\lambda\frac{V}{R+h} \\
\dot{\gamma} =& \frac{L\cos\sigma}{mV} - \Big(\frac{g}{V} - \frac{V}{R+h}\Big)\cos\gamma \\
\dot{\theta} =& \cos\gamma\cos\chi\frac{V}{(R+h)\cos\lambda} \\
\dot{\lambda} =& \cos\gamma\sin\chi\frac{V}{R+h} \\
\dot{h} =& V sin\gamma
\end{aligned} \tag{5.11}$$

Table 5.1 summarises the constant parameters used in the considered models.

The employed aerodynamic models come from experimental nonlinear data obtained in [5] and result in two lookup tables that output the values of c_l and c_d as a function of the Mach and angle of attack. The chosen atmospheric model is the U.S. Standard Atmosphere Model 1962 (USSA1962), as used in [5].

Table 5.1: Constant parameters used in the application

Symbol	Value	Unit	Meaning
R	6371000	m	Earth's radius
g_0	9.80665	m/s^2	Gravitational acceleration at sea level
m	45040	kg	Vehicle empty mass
S	500	m^2	Wing surface

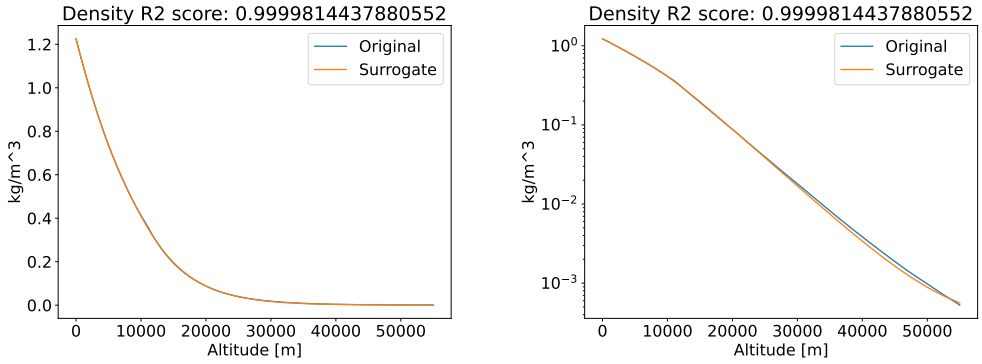
5.3.2 Surrogate Models

To facilitate the GP evolutionary process and to avoid non-differentiable models, surrogates of the atmospheric and aerodynamics models were created and used in the GANNIC design process. Surrogates were created for the following quantities: atmospheric density ρ, speed of sound c, lift coefficient c_l and drag coefficient c_d. The surrogates were created using a MGGP algorithm coded in Python 3.8 and relying on the DEAP library [197]. Training data for the atmospheric models (ρ and c) were generated using the original USSA1962 model in the altitude range from 0 to 55 km. The aerodynamic models' training data consisted of the original dataset from [5], which was expanded to increase the number of data using the Cubic Spline Generalization (CSG) approach described in [18]. The CSG approach involves employing a not-a-knot cubic spline interpolation on the original aerodynamic coefficients data to increase the database's number of data points and enhance the models' smoothness.

The obtained models are displayed in Equation 5.12 and in Figures 5.8 to 5.10.

$$
\begin{aligned}
\rho(h) =& 0.119 \cdot \tanh(1.72 \cdot 10^{-8} \cdot h^2) + \\
& - 1.34 \cdot \tanh(84.7 \cdot \tanh(\tanh(\tanh(h \cdot 10^{-6})))) + 1.22 \\
c(h) =& 109 \cdot \tanh(\exp(1.00 \cdot \tanh(h \cdot 10^{-5}) - 8.84 \cdot 10^{-5} \cdot h)) + \\
& - 16.5 \cdot \cos(\tanh(\sin(8.82 \cdot 10^{-5} \cdot h))) + \\
& - 6.11 \cdot \exp(6.35 \cdot 10^{-5} \cdot h) + \\
& - 54.2 \cdot \cos(\tanh(\sin(\sin(\sin(7.55 \cdot \cos(\sin(\tanh(6.63 \cdot 10^{-5} \cdot h))))))))) + \\
& + 731 \cdot \exp(h^2 \cdot 10^{-10}) - 407
\end{aligned}
$$

$$
\begin{aligned}
c_l(\alpha, M) =& 7.89 \cdot \alpha - 0.232 \cdot \sin(\alpha) - 2.43 \cdot \alpha \cdot \tanh(M - \tanh(\alpha)) + \\
&+ 5.78 \cdot \alpha \cdot \tanh(M - 0.922) - 3.76 \cdot \alpha \cdot \exp(\tanh(M - 0.922)) + \\
&- 0.0112 \\
c_d(\alpha, M) =& 14.9 \cdot \cos(\alpha \cdot \tanh(M - \alpha)) - 16.4 \cdot \cos(\alpha) - 2.45 \cdot \exp(\alpha - M) + \\
&+ 2.38 \cdot \exp(2.41 \cdot \alpha^2 - M) + 1.57
\end{aligned}
\tag{5.12}
$$



(a) Air density ρ surrogate

(b) Air density ρ surrogate, semi-logarithmic plot

Figure 5.8: Air density ρ surrogates. Figure 5.8b show the obtained model on a semi-logarithmic scale, highlighting the differences above 20km. [2]

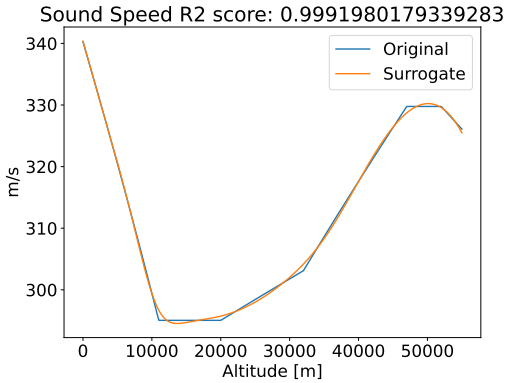
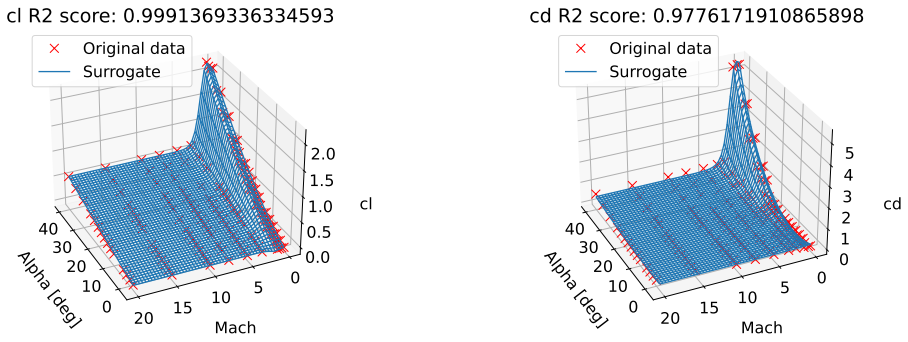


Figure 5.9: Sound speed c surrogate [2]

The accuracy of the obtained models was evaluated using the R^2 score, resulting



(a) Lift coefficient c_l surrogate **(b)** Drag coefficient c_d surrogate

Figure 5.10: Surrogate of the aerodynamic models [2]

in $R^2_\rho = 0.99998$, $R^2_c = 0.99919$, $R^2_{c_l} = 0.99913$, $R^2_{c_d} = 0.97761$. These high R^2 scores indicate that the constructed surrogates effectively capture the characteristics of the training data. Furthermore, the surrogates were constructed using only continuous differentiable functions, resulting in fully continuous differentiable models. This represents an enhancement compared to the original data, as illustrated in Figure 5.9 for the sound speed model. Specifically, the original model was piecewise differentiable, whereas the surrogate is fully differentiable.

5.3.3 Reentry Mission Description and Reference Trajectory

Currently, there are no published manuscripts addressing guidance or control of the FESTIP-FSSC5 vehicle during reentry. In contrast, numerous studies involving reentry missions of the X-33 are available in the literature. Considering the similarities between the X-33 and the FESTIP-FSSC5, the study conducted by Bollino et al. [234] was chosen as a reference. The mission involves an unpowered reentry originating from an initial point, to reach a specified Final Approach Corridor (FAC) box at the trajectory's conclusion. The Final Approach Corridor (FAC) box is defined by the altitude h, longitude θ and latitude λ, along with their respective tolerances. This defined space encompasses all permissible final points of the trajectory, enabling a horizontal landing at the Space Shuttle Landing Facility at NASA's Kennedy Space Center. Constraints are imposed on the dynamic pressure q, normal acceleration a_z and heat rate $\dot{Q}$. Regarding

the heat rate, it was modelled as Equation 5.13, with $C = 9.12 \cdot 10^{-4}\ kg^{0.5}m^{1.5}s^{-3}$, $\Lambda = 45\ deg$, V is the vehicle's speed and ρ is the atmospheric density. This model, derived from [235], was selected since it is formulated for a generic vehicle with lifting body configuration and no specific heat rate models tailored to the FESTIP-FSSC5 can be found in the literature.

$$\dot{Q} = C\sqrt{\rho}V^3(1 - 0.18\sin^2\Lambda)\cos\Lambda \tag{5.13}$$

The limit values for the constraints were taken from [5] for q and a_z and from [235] for the heat rate. The limit value on a_z was increased to 25 m/s^2 to match the limit value used in [234]. This was done since the flown trajectory is similar. The chosen reentry trajectory is summarised in Table 5.2

Initial Conditions	Final Conditions	Controls Bounds	Constraints
$V_0 = 2600$ m/s,	$V_f = 91.44$ m/s,	$-2\ deg \leq \alpha \leq 40\ deg$	$-25\ m/s^2 \leq a_z \leq 25\ m/s^2$,
$\chi_0 = 0$ deg,	χ_f = -60 deg,	$-90\ deg \leq \sigma \leq 90\ deg$	$q \leq 40\ kPa$
γ_0 = -1.3 deg,	γ_f = -6 deg,		$\dot{Q} \leq 4MW/m^2$
θ_0 = -85 deg,	$\theta_f = -80.7112 \pm 0.0014\ deg$,		
$\lambda_0 = 30$ deg,	$\lambda_f = 28.6439 \pm 0.0014\ deg$,		
$h_0 = 51$ km	$h_f = 609.6 \pm 121.92\ m$		

Table 5.2: Chosen reentry trajectory

According to the trajectory data in Table 5.2, the upper and lower bounds used to scale the errors and the states using Equation 5.2 are reported in Table 5.3.

	V	χ	γ	θ	λ	h
Max	5200 m/s	180 deg	89 deg	-80 deg	30 deg	60000 m
Min	1 m/s	-180 deg	-89 deg	-87 deg	27 deg	1 m

Table 5.3: Upper and lower bounds used to scale the states and errors variables using Equation 5.2

A reference trajectory was generated using the surrogates introduced in the previous subsection, employing the methodology developed in one of the works produced during the development of this thesis [18]. The process consists of finding an initial guess using two EAs, namely Multi-Objective Parzen-based Estimation of Distribution (MOPED) [53] and Multi-Population Adaptive Inflationary Differential Evolution

Algorithm (MP-AIDEA) [54]. This initial guess was subsequently used in an optimal control study with Multiple-Shooting transcription. The trajectory was simulated using Equations 5.11. The obtained reference trajectory is shown in Figure 5.11, while the found reference guidance commands are shown in Figure 5.12. The constraints are depicted in Figure 5.13. The found trajectory ends inside the FAC box, and all the constraints are satisfied. As can be seen from Figure 5.13c, the heat rate $\dot{Q}$ constraint is always satisfied by at least an order of magnitude, which is why it is not considered a constraint during the development and application of the GANNIC scheme.

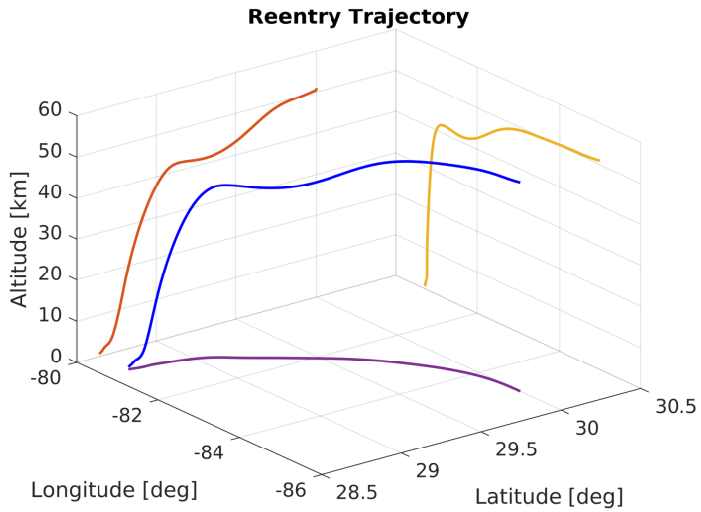
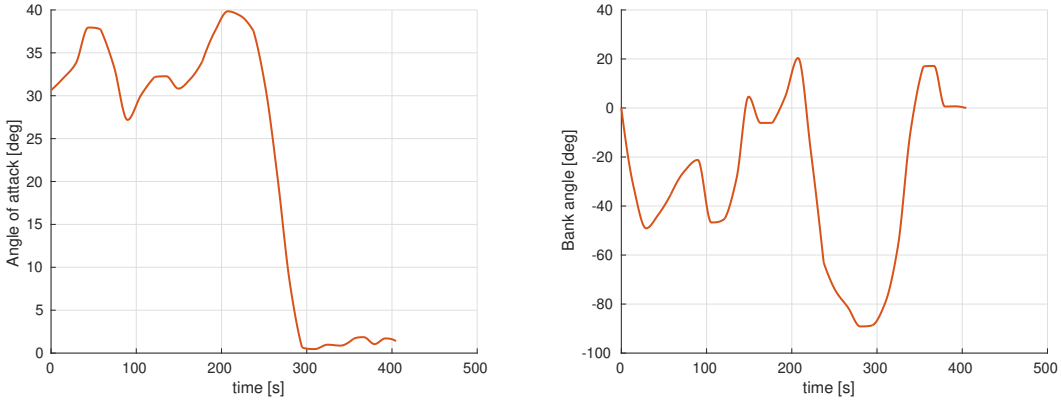


Figure 5.11: Reference trajectory [2]

A reduced portion of the optimal trajectory was used to perform the GANNIC design process, namely, the last 100 seconds. This decision is motivated by multiple factors. Firstly, the mission is deemed successful if the vehicle precisely enters the FAC box in its final position. The FAC box is defined by very strict tolerances on the latitude and longitude, necessitating the development of a precise guidance scheme. Focusing solely on the concluding portion of the trajectory contributes to refining the GANNIC precision. Secondly, a shorter trajectory results in faster trajectory propagations. This enables the performance of multiple runs to generate a statistically significant sample for testing the robustness, i.e. reproducibility, of the GP evolutionary process. Lastly, as illustrated in Figure 5.13, the considered portion of trajectory - from 303.73s to 403.73s



(a) Angle of attack α reference trajectory **(b)** Bank angle σ reference trajectory

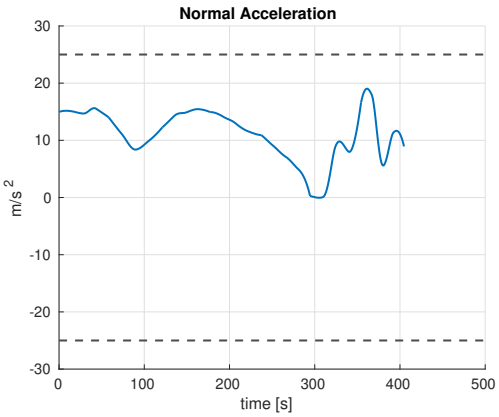
Figure 5.12: Reference trajectories of the control parameters [2]

- contains the maximum values of the a_z and q constraints, providing a comprehensive assessment of the GANNIC scheme's ability to adhere to the applied constraints.

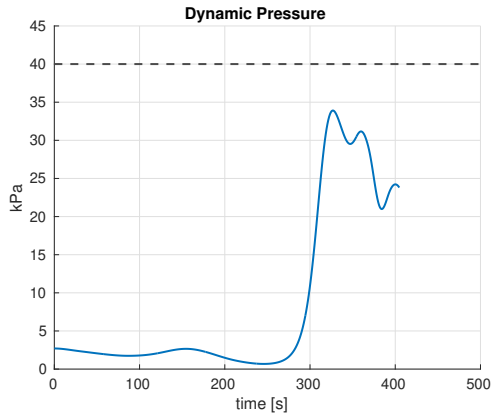
The initial conditions for the reduced trajectory are summarized in Table 5.4. The final conditions, commands bounds and constraints are consistent with those for the full trajectory, as detailed in Table 5.2. The subscript $_{100}$ denotes the initial conditions at $t_{100} = t_f - 100s = 304.73s$. The values of the states and guidance commands at t_{100} were obtained by evaluating at t_{100} the interpolating functions produced using a PCHIP interpolation scheme on the states and control variables' optimal trajectories. The reduced trajectory is depicted in Figure 5.14 and the the reduced reference guidance signals profiles are shown in Figure 5.15.

Initial Conditions at $t_f - 100s$
V_{100} = 693.28 m/s,
χ_{100} = -59.42 deg,
γ_{100} = -47.92 deg,
θ_{100} = -80.80 deg,
λ_{100} = 28.83 deg,
h_{100} = 21.83 km,
α_{100} = 0.47 deg,
σ_{100} = -83.53 deg,
t_{100} = 304.73 s

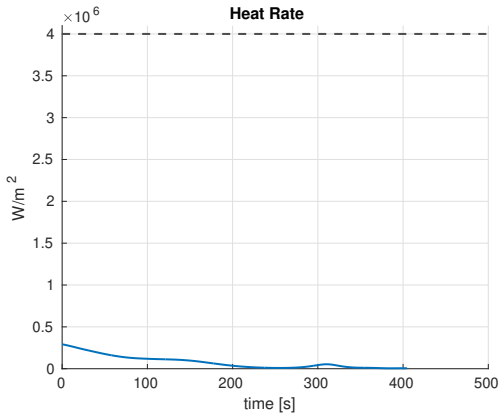
Table 5.4: Initial conditions of the reduced trajectory used for the GANNIC design process



(a) Normal acceleration a_z



(b) Dynamic pressure q



(c) Heat rate $\dot{Q}$

Figure 5.13: Variations of the constrained quantities obtained with the reference trajectory. Dashed lines denote the maximum and minimum allowed values.

The GANNIC scheme underwent testing across the entire trajectory to assess its overall performance. However, it exhibited suboptimal results using the hyperparameters detailed in this section. This outcome is attributed to the experimental determination of settings based solely on the last portion of the trajectory. Achieving effective uncertainty mitigation throughout the entire trajectory necessitates additional testing and a more sophisticated hyperparameter tuning approach. Given that the primary focus of this work was precision at the final point rather than comprehensive uncertainty mitigation, the development of an enhanced and more comprehensive version of the GANNIC scheme is left for future research.

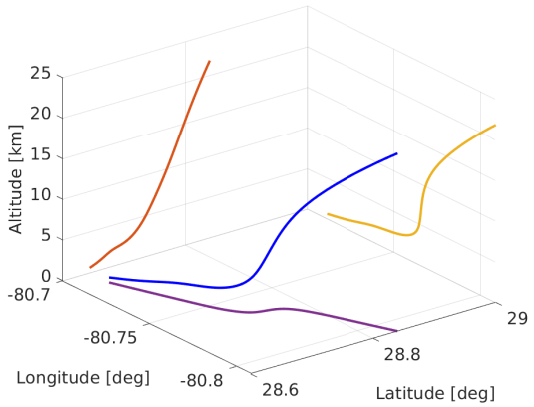
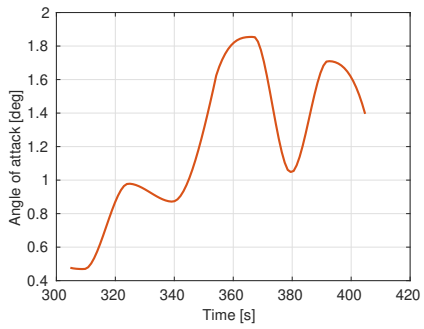
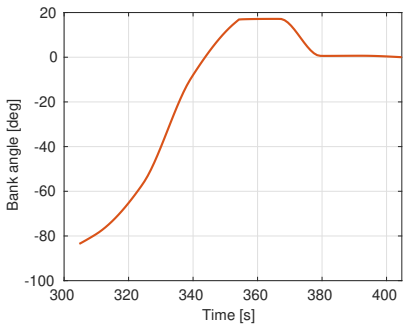


Figure 5.14: Reduced reference trajectory [2]



(a) Angle of attack α reduced reference trajectory



(b) Bank angle σ reduced reference trajectory

Figure 5.15: Reduced reference trajectories of the guidance commands [2]

5.3.4 Uncertainty Model and Training Uncertainty Profiles Selection

Uncertainties are incorporated into atmospheric models to assess the robustness of the GANNIC approach. These uncertainties are introduced as randomized perturbations applied to the surrogate models of air density (ρ) and sound speed (c) quantities. Specifically, uncertainties are modelled according to the methodology outlined in [236]. This approach is motivated by several factors. Firstly, it is a parametric model, hence the user can define the shape and magnitude of the applied uncertainties. This allows for testing the GANNIC's robustness across a broad range of uncertainties. Secondly, these uncertainties are represented as interpolating profiles, ensuring a correlation between successive points along the trajectory, thereby avoiding abrupt variations between ad-

jacent time steps. Thirdly, they are modelled to be constrained within boundaries that increase proportionally with a desired variable, while remaining confined between user-defined upper and lower bounds. In the context of this work, these boundaries are a function of the altitude (h). The described features allow for realistic modelling of the uncertainties in the atmospheric models. In fact, abrupt variations are not physically meaningful and the comparison between experimental data and atmospheric models reveals an increasing discrepancy at higher altitudes. A comprehensive description of the formulation and application of these uncertainty profiles is provided hereafter.

The interpolating profiles are generated by creating a random uniform distribution of 20 points within the [0, 1] range. A PCHIP interpolation is then applied with respect to altitude (h) to yield an Uncertainty Profile $U(h)$. An illustrative example of an uncertainty profile is presented in Figure 5.16. The 20 points are uniformly distributed across the considered altitude range, which, for the addressed problem, spans 60 km, resulting in a point positioned every 3 km. Employing this distribution ensures that even a sudden variation from one extreme to the other between successive points results in the variation of the perturbed quantity occurring over a 3 km interval. Although a variation every 3 km might still be deemed too severe for a realistic scenario, this interval is chosen to design and assess the GANNIC scheme against a more conservative scenario, i.e. with more severe uncertainties.

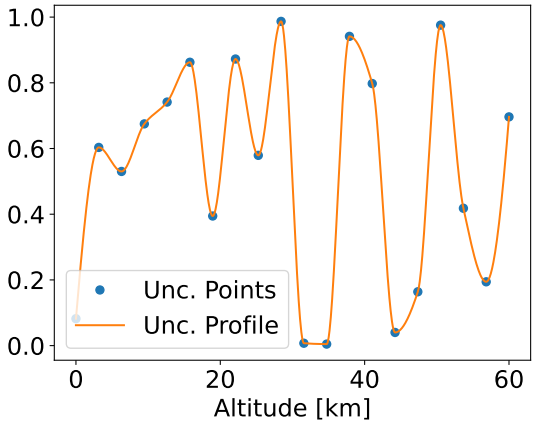


Figure 5.16: Example of an uncertainty profile $U(h)$

Once the uncertainty profile is generated, this is applied to the chosen quantities

using Equation 5.14.

$$x_{unc}(h) = x_{nom}(h)[U(h)((1+\epsilon(h)) - (1-\epsilon(h))) + (1-\epsilon(h))] \tag{5.14}$$

The uncertainty bounding parameter, denoted as ϵ, is computed using Equation 5.15 where h_c is the maximum admissible value of h. h_c is problem dependent and must be set accordingly to the considered trajectory. For this application, $h_c = 60km$. u_b and l_b are the upper and lower bounds defined by the user and can be varied to alter the magnitude of the applied uncertainty.

$$\epsilon(h) = l_b\left(1 - \frac{h}{h_c}\right) + u_b\frac{h}{h_c} \tag{5.15}$$

Equation 5.15 is formulated to output a bounding parameter ϵ which is used to define the range of variation of the perturbed quantity around its nominal value. This bounding parameter will vary according to the altitude but it is constrained by the user defined upper and lower bounds, u_b and l_b respectively, such as $\epsilon \in [u_b, l_b]$. In fact, $\epsilon = u_b$ when the altitude is at its maximum, i.e. $h = h_c$, while $\epsilon = l_b$ with $h = 0$.

In the subsequent discussion, the term "shape of the uncertainty" refers to the interpolating profiles, denoted as $U(h)$, while the term "magnitude of the uncertainty" refers to the value of u_b in Equation 5.15.

As an illustrative example of the uncertainty application, a quantity x that varies linearly with h is considered, specifically $x = h \cdot 10^{-4}$. By setting the parameters in Equation 5.15 as $l_b = 0.2$, $u_b = 0.5$, and $h_c = 60000\ m$, the perturbed quantity x_{unc} will exhibit variations between 20% and 50% of its nominal value x_{nom}. Using the uncertainty profile shown in Figure 5.16 and applying Equations 5.15 and 5.14, the perturbed quantity x_{unc} is obtained as depicted in Figure 5.17. Here, the variation of the nominal and perturbed quantities x_{nom} and x_{unc} are illustrated. Notably, x_{unc} exhibits an oscillating behaviour corresponding to the applied uncertainty profile $U(h)$ in Figure 5.16.

To develop and evaluate the GANNIC scheme, 1100 uncertainty profiles were randomly generated from a uniform distribution in the range [0, 1]. Among these, 1000

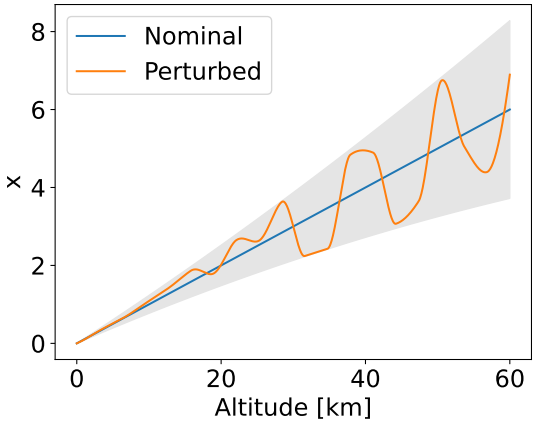
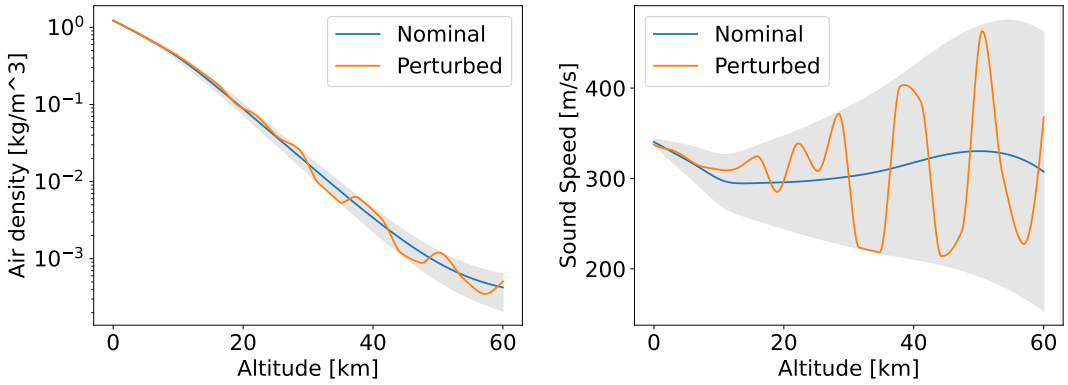


Figure 5.17: Variation of the nominal and perturbed quantities x_{nom} and x_{unc} as a function of the altitude. The shaded area represents the range of variation of the perturbed quantity defined by ϵ.

profiles constitute the set of training profiles, denoted as $\mathbf{U}_{train}$, from which the scenarios used in the training phase are selected. The remaining 100 profiles are assigned to the test uncertainty set, denoted as $\mathbf{U}_{test}$. Figure 5.18 illustrates an example of one generated uncertainty profile applied to air density ρ and sound speed c, with parameters $l_b = 0.01$, $u_b = 0.2$ and $h_c = 60km$.



(a) Air Density ρ **(b)** Sound Speed c

Figure 5.18: Example of uncertainty profile applied to the Air Density and Sound Speed quantities.

In section 5.2.3, the process of selecting uncertainty profiles to use in the GP evolutionary process involved the following steps: initially, the 1000 uncertainty profiles from $\mathbf{U}_{train}$ were used to propagate the dynamical system in Equation 5.9 by applying

open-loop guidance commands, resulting in 1000 trajectories. During this phase, ϵ was calculated with $lb = 0.01$ and $ub = 0.2$. The open-loop guidance commands were derived from the reference trajectory and are illustrated in Figure 5.12. Among the 1000 trajectories propagated with the applied uncertainties using open-loop commands, only 39.7% (397/1000) were successful. Subsequently, the uncertainty profiles that led to successful trajectories were excluded from the pool of 1000 training profiles, resulting in 603 remaining profiles for the GP training phase. As a reminder, a trajectory was considered successful if the vehicle ended up within the FAC box at the last point of the trajectory, as detailed in section 5.3.3. These 603 uncertainty profiles were then ranked from least to most severe based on the magnitude metric μ, evaluated according to Equation 5.10. This evaluation considered only errors on θ, λ, and h, the states defining trajectory success. Thus, Equation 5.16 was employed, where $\bar{e}$ represents a scaled error calculated using Equation 5.2, with the minimum and maximum values given in Table 5.3.

$$\mu = |\bar{e}_\theta(t_f)| + |\bar{e}_\lambda(t_f)| + |\bar{e}_h(t_f)| \tag{5.16}$$

These ranked uncertainty profiles were then linearly divided into five groups, as described in section 5.2.3. Within each group, the uncertainty profile with the highest metric μ was chosen to participate in the GP evolutionary process. The distribution of uncertainty profiles among the groups is illustrated in Figure 5.19.

To obtain the results presented in the following, the ϵ parameter was evaluated considering a lower bound $l_b = 0.01$ and five different values for the upper bounds u_b: 0.2, 0.3, 0.4, 0.5 and 0.6. The following subsections explain how and when these different values for the upper bounds were applied.

5.3.5 Inclusive Genetic Programming Settings

The IGP used to find the adaptation laws was set as in Table 5.5.

Within the specified primitives, *add*3 represents a ternary addition, while *psqrt*, *plog* and *pexp* represent protected versions of the *sqrt*, *log*, and *exp* functions, designed to avoid numerical issues. Examples of the *plog* and *pexp* functions are given in Equations

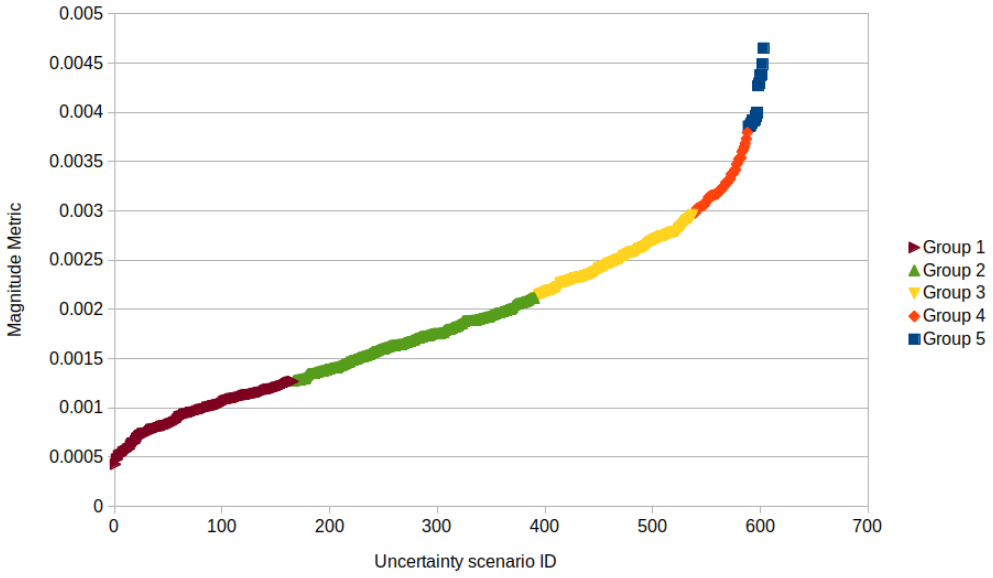


Figure 5.19: Training uncertainty profiles ranked and divided into five intervals

4.7 and 4.11. *psqrt* is defined similarly to *plog*, i.e. 0 is output for input values lower than 0. The associated mutation mechanisms feature probabilities expressed as percentages, indicating the likelihood of selecting a particular mechanism during the mutation process. The environmental parameter η, input to the GP as in Figure 5.2, comprise the air density ρ.

The fitness function is evaluated as described in Section 5.2.2. Briefly, each individual has a fitness composed of two parameters, F_f and P_f. F_f represents the true objective, i.e. reaching the FAC box; while P_f is a penalty accounting for constraints violations. These fitness parameters are evaluated considering the results on five uncertainty profiles from the training set $\mathbf{U}_{train}$, hence five trajectory propagations are performed to evaluate each GP individual.

Regarding the F_f evaluation, the necessary parameters are set as follows. The scaled tracking errors are computed by performing a min-max scaling of the tracking errors, using Equation 5.2 with the min and max values from Table 5.3. This operation results in the vector of scaled tracking errors $\bar{\mathbf{e}}$, and consequently the vector of scaled tracking errors in the final position $\bar{\mathbf{e}}_f = \bar{\mathbf{e}}(t_f)$. The vector $\bar{\mathbf{e}}$ is then used to compute the integrals of the absolute scaled tracking errors over the last 30% of the trajectory. These

Input features	V, χ, γ, θ, λ, h, ρ, ν
Population Size	500 individuals
Maximum Generations	300
Stopping criteria	Reaching the maximum number of generations
Number of niches	10
Crossover probability	0.2 (+0.01 at every generation if $P_f = 0$) $\rightarrow$ 0.65
Mutation probability	0.7 (-0.01 at every generation if $P_f = 0$) $\rightarrow$ 0.25
1:1 Reproduction probability	0.1
Crossover selection criteria	Best, random
Mutation selection criteria	Random
1:1 Reproduction selection criteria	Best
Evolutionary strategy	$M + \Lambda$
M	Population size
Λ	Population Size $\times$ 1.2
Number of Ephemeral constants	5
Limit Height	40
Limit Size	40
Selection Mechanism	Inclusive Tournament
Double Tournament fitness size	2
Double Tournament parsimony size	1.6
Tree creation mechanism	Ramped half and half
Mutation mechanisms	Uniform (55%), Shrink (5%), Insertion (25%), Mutate Ephemeral (15%)
Crossover mechanism	One point crossover
Primitives Set	$+$, $-$, $*$, $add3$, $tanh$, $psqrt$ $plog$, $pexp$, sin, cos
Fitness measures	F_f, P_f

Table 5.5: IGP setting for the GANNIC controller design process.

are computed employing Equation 5.7, with $t_{start} = 379s$ and $t_f = 404s$. One integral for each state is computed and added to the vector $\mathbf{I}$. Using the vectors $\bar{\mathbf{e}}_f$ and $\mathbf{I}$, the vector $\mathbf{F} = [\mathbf{I}/s_f, \bar{\mathbf{e}}_f \mathbf{w}_s^T s_f]$, described in section 5.2.2, is computed. The scaling factor s_f and weight vector $\mathbf{w}_s$ are set to $s_f = 1000$ and $\mathbf{w}_s = [1, 1, 1, 10, 10, 1]$, respectively. $w_s(4)$ and $w_s(5)$ are set equal to 10 to emphasize the significance of tracking errors in θ and λ in the final position. As described in Section 5.2.2, the vector $\mathbf{F}$ is then used to compute F_u and finally F_f using Equation 5.8. Similarly, the penalty P_f is evaluated by first computing the constraints violations in each point of the trajectory, considering the constraints on dynamic pressure q and normal acceleration a_z, as listed in Table 5.2. The constraint on $\dot{Q}$ is not considered following the discussion in Section

5.3.3. The constraints violations are then added to the vector P which is then used to compute P_u and finally P_f.

5.3.6 Results

The GANNIC guidance scheme was implemented in Python 3.10, using the DEAP library [197] for the GP components and Tensorflow [237] for NN creation. To test the GANNIC scheme, the final 100 seconds of the optimal trajectory outlined in Subsection 5.3.3 were considered, employing the initial conditions specified in Table 5.4. Trajectories were propagated using the Runge-Kutta 4 integration scheme with a fixed time step $dt = 5s$, chosen for a balance between computational efficiency and accuracy. Parameters for the scaler in Equations 5.5 and 5.6 were set as $p_1 = 2000$, $p_2 = 0.0015$, $u_{b_S} = 100$, and $l_{b_S} = 0$. For all NN configurations employed, weights and biases were initialized to zero, the tanh activation function was used in the hidden layer, and the linear activation function was employed in the output layer.

Throughout the subsequent experiments, the terms "train" or "training phase" denote the GP evolutionary process using uncertainty profiles from the training set $\mathbf{U}_{train}$. This phase aims to identify the optimal set of GP differential equations for updating the weights in the chosen NN configuration. On the other hand, the terms "test" or "test phase" refer to trajectory propagations using the set of GP models obtained during training. These propagations employ uncertainty profiles from the test set $\mathbf{U}_{test}$, aiming at assessing the robustness of the evolved GP models by evaluating their performance on different uncertainty profiles in terms of both shape and magnitude compared to those used during training.

Neural Network Topology Optimization

Before conducting a comprehensive statistical analysis to understand the capabilities of the GANNIC scheme, a study concerning the NN topology optimization was carried out to identify a suitable configuration for the given problem. The outlined procedure is detailed in section 5.2.4 and presented concisely in Algorithm 10. In summary, a series of runs is performed to assess the performance of a NN topology. Each run is

composed of three phases: 1) the execution of the GP evolutionary process, to find the NN adaptation laws for the given NN topology; 2) the test of the obtained scheme on the selected test uncertainty profiles; 3) the assessment of the impact of each NN weight by individually setting them to zero and observing the resulting performance on the test uncertainty profiles. Following the terminology in Algorithm 10, the used parameters were set as follows: $N_{evolutions} = 4$, indicating four GP evolutions, i.e. four runs, were conducted; $N_{test} = 100$, signifying that the best individual from each evolution underwent testing on 100 uncertainty profiles from the test set $\mathbf{U}_{test}$. The initial NN topology consisted of six inputs representing the scaled tracking errors on the states, one hidden layer with four neurons, and two neurons in the output layer, corresponding to each guidance command. This configuration is denoted as the 6-4-2 configuration hereafter. Table 5.6 lists the results of the GANNIC schemes produced in the four runs, and these are used as references to analyze the NN's weights impact. The results are shown in terms of fitness values and success rates on the test uncertainty profiles using the 6-4-2 NN topology. During the GP training phase, $u_b = 0.2$ was used.

Table 5.6: Test success rates obtained with the 6-4-2 NN configuration in the four runs performed.

Run ID	Fitness values		Test success rate
	F_f	P_f	$u_b = 0.2$
1	0.001071	0	90%
2	0.003015	0	100%
3	0.001236	0	88%
4	0.001521	0	89%

Following the evaluation on the 100 test uncertainty profiles, the NN weights were individually set to zero, and 100 trajectory propagations were executed each time using the best GP individuals identified in the four runs. The outcomes of this study are summarized in Figure 5.20. In this representation, each bar denotes the success rate on the test uncertainty profiles, with four bars presented for each weight, corresponding to each of the four runs. The green bars signify success rates equal to the reference success rate, the blue bars indicate success rates greater than the reference, and the red bars represent those with a lower success rate than the reference. The colour scheme is

interpreted as follows: green bars denote weights that are not significant, meaning their removal does not alter performance; blue bars indicate weights whose removal leads to improved NN performance, while red bars highlight influential weights, the removal of which results in degraded performance.

Analysis of Figure 5.20 reveals that weights 12 to 23 were consistently found to be non-influential across all four runs, while weights 29, 34, and 35 exhibited significant influence. Notably, some columns are missing for certain weights, indicating that, in those runs, the success rate dropped to 0% upon removing that specific weight. Additionally, it is interesting to observe that some weights in the outer layers (24 to 37) negatively impact the NN, meaning that removing them leads to improved performance. Figure 5.21 offers a more concise view of this behaviour, displaying the weights' location and their influence for each of the four runs. These diagrams will be referred to as weight maps from this point onwards. They are not to be confused with actual NN topologies, since their goal is to show only the location and the influence of each NN weight. In these diagrams, the weights are represented by arrows connecting neurons. The NN inputs, denoted as $i1$ to $i6$, correspond to the scaled tracking errors on the states. Neurons $h1$ to $h4$ represent the hidden layer, and $o1$ and $o2$ are the output neurons responsible for the guidance commands α and σ.

The weights maps illustrated in Figure 5.21 are derived from the initial NN configuration by eliminating weights that do not adversely impact performance, as indicated in Figure 5.20. Examining these maps reveals that input nodes $i4$ to $i6$, corresponding to the tracking errors on θ, λ, and h, are consistently connected by weights that are not influential. This aligns with the physical behaviour, as the equations of motion in Equation 5.11 suggest that the derivatives of the states are primarily influenced by V, χ, and γ. Additionally, the colour of each arrow signifies the weight's importance: yellow indicates a slight decrease in performance when the weight is removed, resulting in a success rate between 67% and 100%; light orange denotes a success rate between 34% and 66%, and dark orange indicates a success rate between 1% and 33%. Weights that, when removed, lead to a 0% success rate are coloured in red. According to this colour scheme, it is evident that the tracking error on γ (flight path angle) consistently

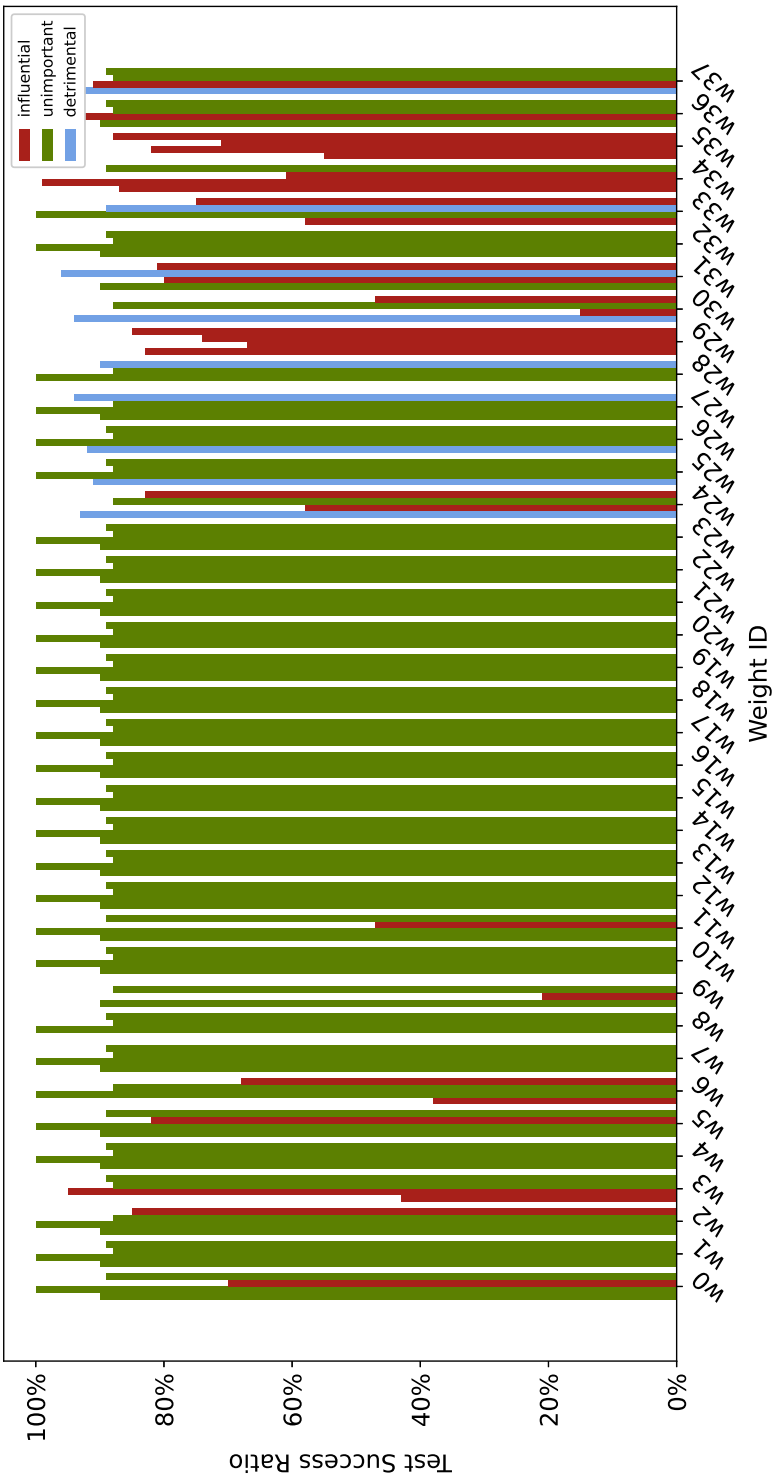
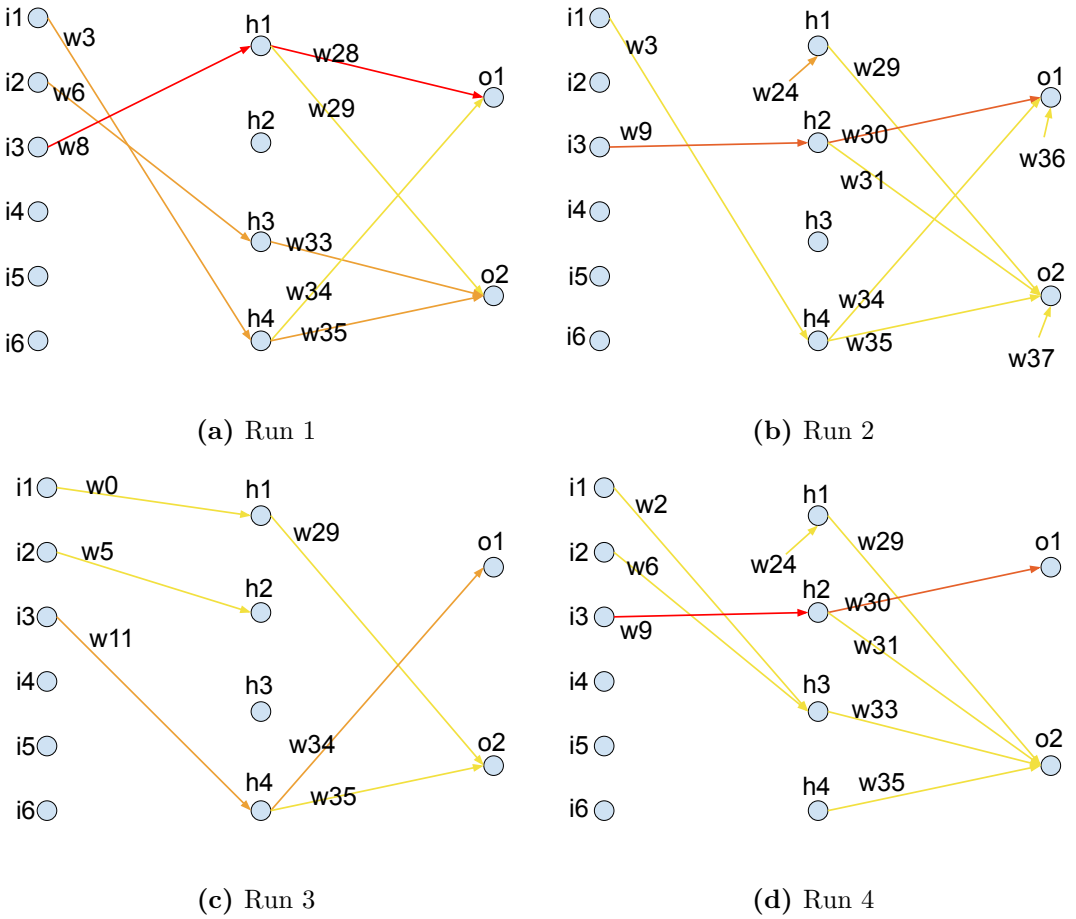


Figure 5.20: Results of the first iteration of the NN topology optimization process.



(a) Run 1 **(b)** Run 2

(c) Run 3 **(d)** Run 4

Figure 5.21: Graphical representation of the weights influence analysis. The yellow arrows represent those weights that, when removed, cause a slight decrease in performances; light and dark orange represent those weights that, when removed, cause a medium and high decrease in performances, while the red arrows represent those weights that, when removed cause the success rate to drop to zero. [2]

shows the most significant influence. This observation aligns with existing literature, where guidance schemes often focus on tracking the flight path angle and velocity, as seen in works such as [238, 239, 240]. Another noteworthy observation is that at least one of the hidden neurons is never fully connected, i.e. the weights between hidden to outer neurons are not influential. This suggests that three hidden neurons may be sufficient for the treated problem.

Based on the insight gained from this analysis, the decision was made to simplify the NN configuration from six inputs, four hidden neurons, and two output neurons

(configuration 6-4-2) to a structure with three inputs, three hidden neurons, and two output neurons (configuration 3-3-2). In this reduced configuration, the inputs consist of the scaled tracking errors $\bar{e}_v$, $\bar{e}_\chi$, and $\bar{e}_\gamma$, respectively i_1, i_2 and i_3 in Figure 5.21.

To determine whether the newly obtained topology was optimal, the topology optimization process was repeated. Using the reduced topology (3-3-2), another four runs were carried out, using an uncertainty upper bound of $u_b = 0.2$ during the GP training. The results of this second iteration are presented in Appendix A. Based on these results, the second optimization iteration did not provide clear evidence of whether further reduction in the NN configuration was beneficial. Therefore, the 3-3-2 configuration was retained as the optimal configuration for the subsequent statistical analysis.

Statistical Campaign

A comprehensive statistical campaign was conducted to evaluate the robustness and reproducibility of the GANNIC scheme. The GANNIC design process was repeated twenty times. During each run, the GP evolutionary process was performed once, the produced weights adaptation laws were applied to the NN and the resulting scheme was tested on 100 test uncertainty scenarios. For each evolutionary process, the IGP was set as in Table 5.5 and a NN with the 3-3-2 topology was employed. The NN weights were all initialized as 0. During training, the same five uncertainty profiles from the training set $\mathbf{U}_{train}$ were used for all runs.

For each run, the best individual was selected and used to produce the statistical results. The process employed to select the best performing individual is described in Appendix B. Briefly, the evolution of the F_f fitness value was monitored, and whenever a decrease was observed, the corresponding individual was picked. The selected individuals from each run underwent testing through 100 trajectory propagations, employing 100 test uncertainty profiles from $\mathbf{U}_{test}$. This process was repeated five times, varying the value of the uncertainty's upper bound u_b, from 0.2 to 0.6 in increments of 0.1, while l_b was kept fixed at $l_b = 0.01$. Subsequently, an average success rate was calculated, considering the performance under different u_b values. This average success rate reflects the individual's generalization capability across uncertainties with varying

magnitudes and shapes. This process was employed since the individual with the lowest (best) fitness might not be the one with the best generalization capabilities.

To further enhance the performance of the GANNIC scheme, i.e. increase the test success rates, the best individuals in each run underwent a process of fine-pruning. This process was applied to the NN similarly to what was done during the topology optimization process, with the exception that the aim of the fine-pruning is not to alter the number of neurons in the network but to deactivate the weights that are detrimental to the overall performance. This process is described in lines 6-9 of Algorithm 10. It consists of setting the weights of the NN to zero one by one and conducting 100 trajectory propagations with the modified topology using 100 uncertainty profiles from the uncertainty test set $\mathbf{U}_{test}$. The use of this approach is justified by the results gathered during the initial NN topology optimization study. As it was observed, some weights might be detrimental to the overall performance. Therefore, by selectively removing them, improved results can be achieved. The comprehensive results of this fine-pruning process are detailed in Appendix C.

Following the pruning for all runs, the best-pruned topologies are tested again on the 100 test uncertainty profiles. Table 5.7 lists the results on the twenty runs for both the unpruned and fine-pruned topologies. The latter are marked in bold-faced font. The initial row provides the success rate on the test uncertainty profiles obtained using the open-loop reference commands. The last four rows in Table 5.7 depict the median and standard deviation of the test success rates corresponding to different magnitudes of uncertainty. These values were evaluated based on the outcomes of the twenty runs conducted.

The results indicate that the unpruned GANNIC achieves a median success rate of 100% when tested with $u_b = 0.2$, the magnitude used during training. Furthermore, it shows satisfactory performance with $u_b = 0.6$, attaining a median success rate of 74% with a standard deviation of 8.21%. Notably, these performances greatly surpass those of the reference open-loop guidance scheme, which, when subjected to an uncertainty magnitude of $u_b = 0.2$, can successfully guide the system in only 36% of the considered test uncertainty profiles. Concerning the pruning process, an average enhancement of

2.12% in median success rate and an average reduction in the standard deviation of 1.28% are observed across the different u_b values. Therefore, it is concluded that the GANNIC scheme exhibits robustness against unforeseen uncertainties, and the fine-pruning process further enhances both the performance and statistical consistency of the GANNIC scheme.

A visual representation of the obtained results is presented in Figures 5.22 to 5.24, illustrating the outcomes of the best (run 15) and worst (run 19) runs. Each line corresponds to a trajectory propagated using a specific test uncertainty profile and magnitude. The colour bar indicates the magnitude of the applied uncertainty, with red lines representing failed trajectories. In Figures 5.22 and 5.23, the black dashed line represents the reference trajectory, while in Figure 5.24, the black horizontal lines denote the boundary values of the constrained quantities.

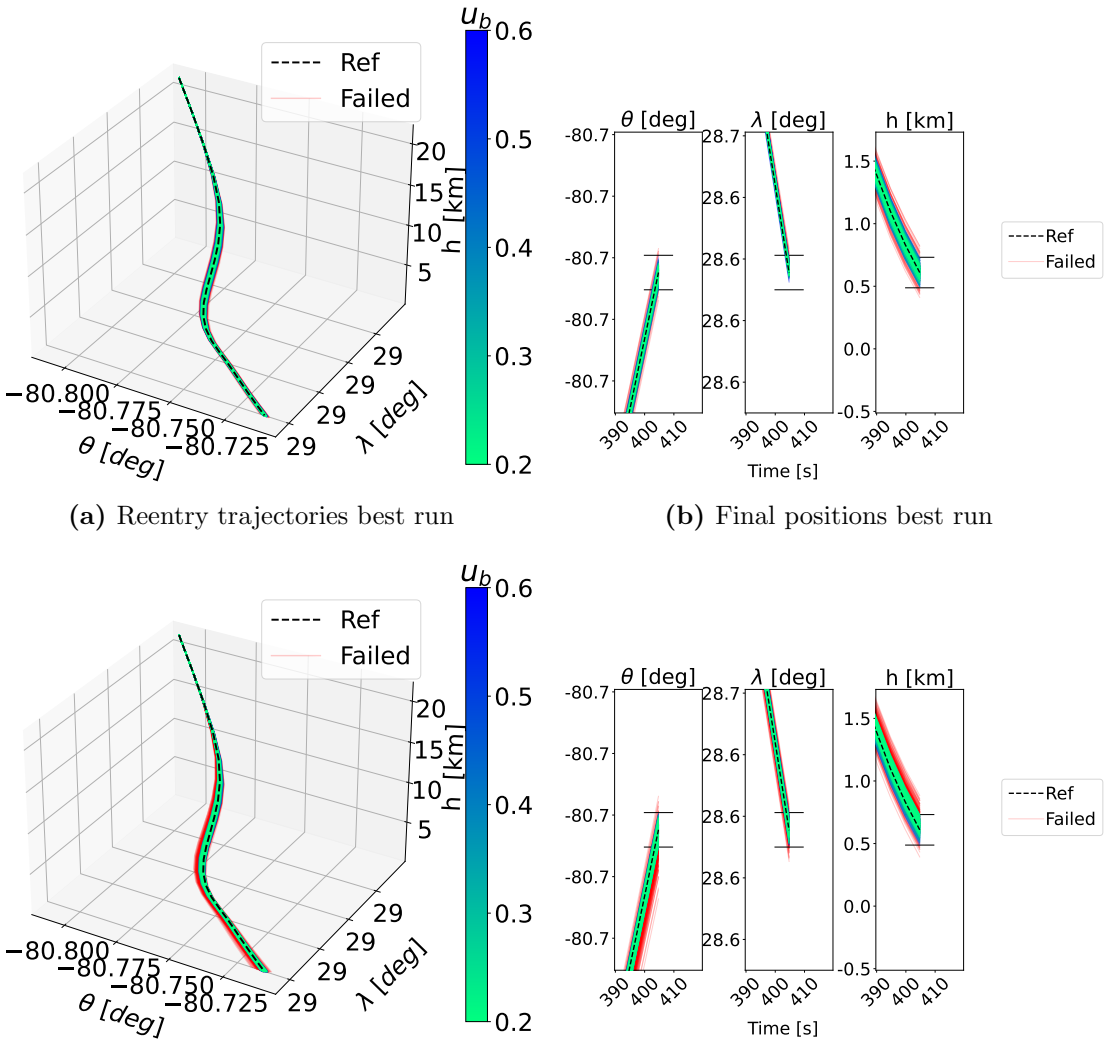
By analyzing Figure 5.22 it can be seen the GANNIC scheme demonstrates efficacy in mitigating deviations of failed trajectories from the desired final position. Additionally, GANNIC exhibits robustness to increasing magnitudes of applied uncertainty, as shown by its ability to produce trajectories showing small divergence from the reference trajectory, even under heightened magnitudes of uncertainty (blue lines in Figure 5.22).

Figure 5.23 displays the trajectories of the guidance commands α and σ—representing the angle of attack and bank angle, respectively. The GANNIC scheme effectively generates guidance commands closely aligned with the reference signals. Trajectories with larger deviations from the reference ones are associated with higher uncertainty magnitudes or failed cases. Notably, failed trajectories tend to cluster further away from the reference, suggesting that substantial deviations from the reference may contribute to failure. Therefore, reducing u_{b_S}—the scaler's upper bound—might be considered to enhance performance.

In Figure 5.24, the trajectories of the constrained quantities a_z and q are presented for both the best and worst runs. These plots clearly illustrate that the constraints are consistently satisfied, even when the desired final position is not reached.

Table 5.7: Statistical analysis results performed using the GANNIC control scheme. The data in bold correspond to the pruned topologies.

Run ID	Fitness values		Test success rate				
	F_f	P_f	$u_b = 0.2$	$u_b = 0.3$	$u_b = 0.4$	$u_b = 0.5$	$u_b = 0.6$
Open Loop	/	/	36%	20%	15%	13%	9%
1	0.00719	0	100%	100%	94%	87%	74%
			100%	**100%**	**95%**	**87%**	**80%**
2	0.00391	0	100%	94%	87%	83%	75%
			100%	**100%**	**93%**	**89%**	**79%**
3	0.00365	0	100%	96%	88%	84%	76%
			100%	**96%**	**88%**	**84%**	**76%**
4	0.0403	0	100%	99%	96%	85%	79%
			100%	**100%**	**97%**	**85%**	**78%**
5	0.0117	0	100%	98%	91%	82%	66%
			100%	**100%**	**95%**	**84%**	**70%**
6	0.00572	0	100%	96%	93%	84%	74%
			100%	**100%**	**96%**	**86%**	**80%**
7	0.00456	0	100%	100%	94%	89%	80%
			100%	**100%**	**95%**	**89%**	**81%**
8	0.00726	0	100%	97%	92%	84%	74%
			100%	**100%**	**94%**	**82%**	**78%**
9	0.0126	0	100%	99%	91%	85%	76%
			100%	**100%**	**93%**	**86%**	**79%**
10	0.0119	0	100%	98%	89%	79%	67%
			100%	**98%**	**90%**	**84%**	**72%**
11	0.00280	0	100%	95%	88%	78%	70%
			100%	**95%**	**88%**	**78%**	**70%**
12	0.00675	0	99%	97%	90%	84%	71%
			100%	**99%**	**93%**	**88%**	**80%**
13	0.00651	0	100%	100%	92%	85%	74%
			100%	**100%**	**94%**	**87%**	**79%**
14	0.00293	0	96%	90%	83%	74%	60%
			100%	**93%**	**86%**	**75%**	**69%**
15	0.00270	0	100%	98%	92%	79%	76%
			100%	**100%**	**96%**	**89%**	**80%**
16	0.00848	0	100%	97%	91%	81%	70%
			100%	**96%**	**94%**	**83%**	**70%**
17	0.00430	0	100%	100%	98%	85%	78%
			100%	**100%**	**99%**	**87%**	**78%**
18	0.00258	0	100%	100%	94%	83%	75%
			100%	**100%**	**95%**	**87%**	**79%**
19	0.0287	0	92%	78%	68%	58%	43%
			97%	**88%**	**76%**	**63%**	**53%**
20	0.0437	0	100%	100%	95%	84%	71%
			100%	**99%**	**95%**	**87%**	**75%**
Median	/	/	100.0%	98.0%	91.5%	84.0%	74.0%
			100%	**100%**	**94%**	**86%**	**78%**
Standard Deviation	/	/	1.95%	5.06%	6.27%	6.51%	8.21%
			0.67%	**3.17%**	**5.06%**	**6.10%**	**6.61%**



(a) Reentry trajectories best run

(b) Final positions best run

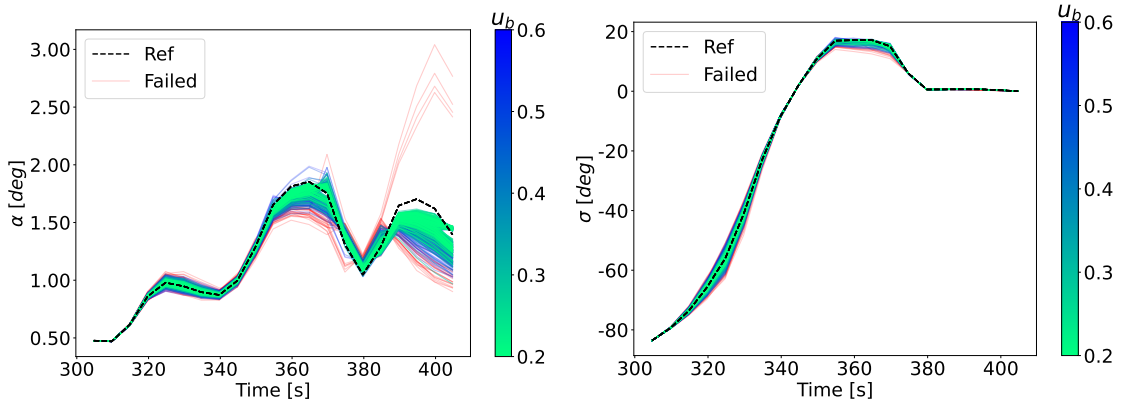
(c) Reentry trajectories worst run

(d) Final positions worst run

Figure 5.22: Reentry trajectories and final positions of the best (run 15 in Table 5.7) and worst (run 19 in Table 5.7) runs performed with the GANNIC scheme. The red lines represent the failed trajectories. The black horizontal bars in the final position plots represent the FAC box boundaries. [2]

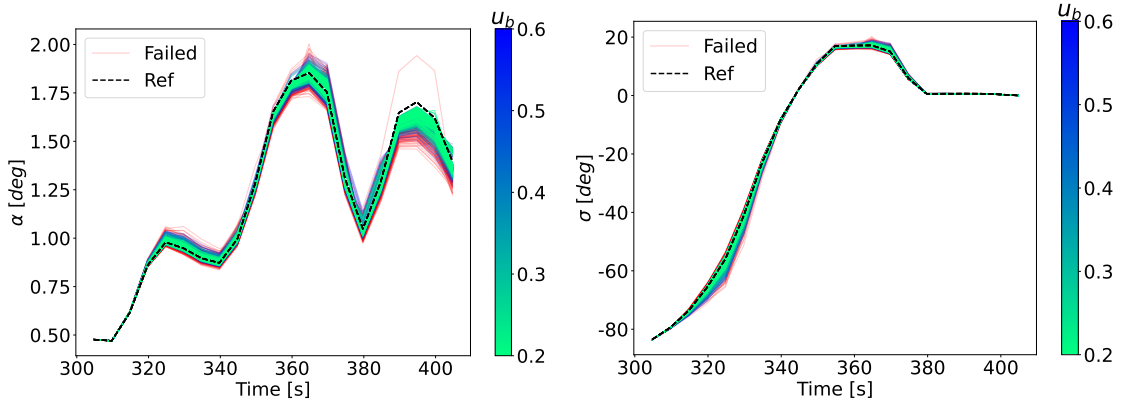
Comparison with a standalone Genetic Programming guidance scheme

To contextualize the obtained results, the performance of the GANNIC scheme was compared with a standalone GP guidance algorithm. The GP algorithm, outlined in section 4.2.1, was employed to determine guidance commands, as illustrated in Figure 4.4. For this comparative analysis, the same mission was considered as in the GANNIC



(a) α trajectory best run

(b) σ trajectory best run



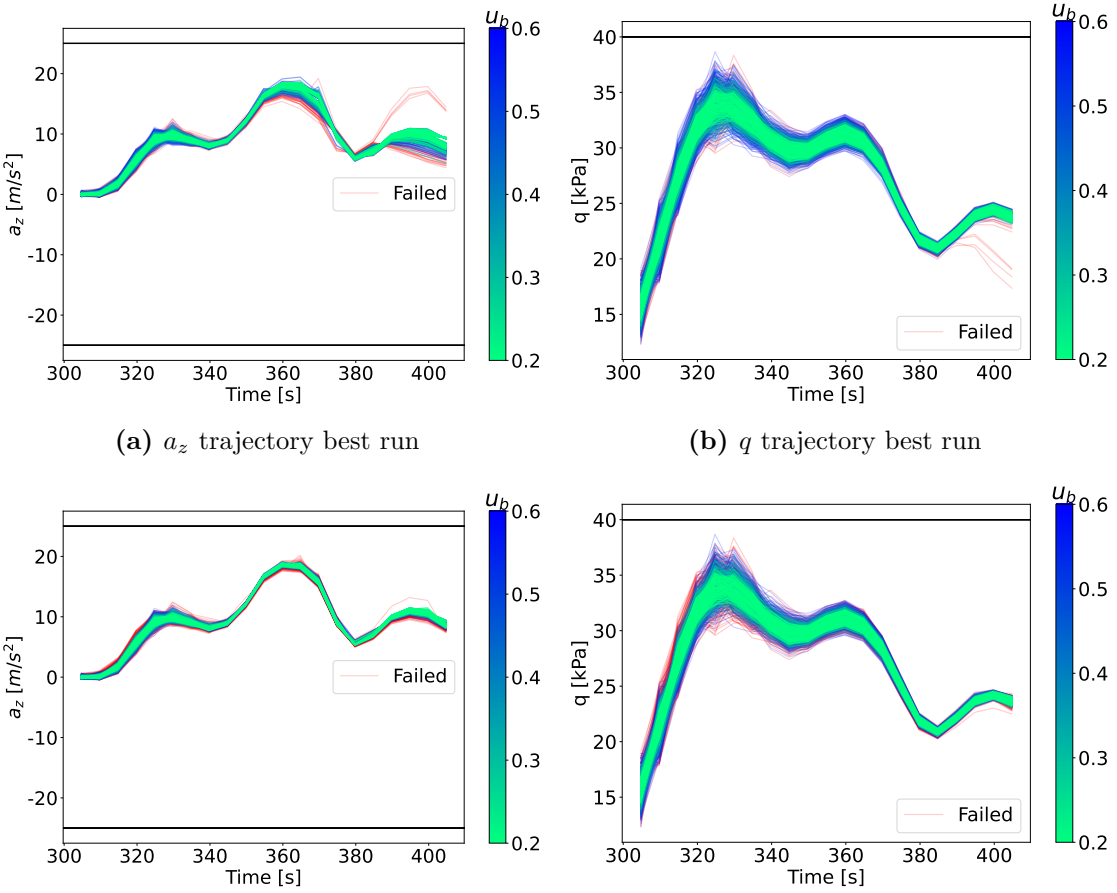
(c) α trajectory worst run

(d) σ trajectory worst run

Figure 5.23: Trajectories of the guidance commands, angle of attack α and bank angle σ for the best (run 15 in Table 5.7) and worst (run 19 in Table 5.7) runs performed with the GANNIC scheme. The red lines represent the failed trajectories. The dashed black line represents the reference trajectory.

scheme, and the GP was configured according to Table 5.5. The same fitness function and training/test uncertainty profiles as in the GANNIC scheme were employed. Furthermore, the selection of the best-performing individuals in each run was performed by following the approach outlined in Algorithm 11, as done for GANNIC. The run results using only the GP for guidance commands generation are presented in Table 5.8, with qualitative trajectory representations shown in Figure 5.25.

By comparing Figure 5.25 and Figure 5.22, it can be observed that the GANNIC and GP algorithms show similar trajectory tracking capabilities in their respective best runs. However, this is not the case for the worst runs. Examination of Figures 5.22d and



(a) a_z trajectory best run

(b) q trajectory best run

(c) a_z trajectory worst run

(d) q trajectory worst run

Figure 5.24: Trajectories of the constrained quantities, normal acceleration a_z and dynamic pressure q for the best (run 15 in Table 5.7) and worst (run 19 in Table 5.7) runs performed with the GANNIC scheme. The red lines represent the failed trajectories. The horizontal black line represents the constraints.

5.25d proves that GANNIC effectively maintains failed trajectories in the proximity of the target, whereas the trajectories generated using the GP guidance commands diverge further away from the desired final position.

Figure 5.26 presents a graphical synthesis of results obtained through GANNIC in both pruned and unpruned forms, alongside results from the standalone GP. Test success rates are depicted as box plots, showcasing the median and statistical distribution for each magnitude considered. Open-loop results are marked with black dots for reference. Notably, GANNIC exhibits statistical consistency superior to the standalone GP, an observation reinforced by the standard deviations provided in Tables 5.7 and 5.8.

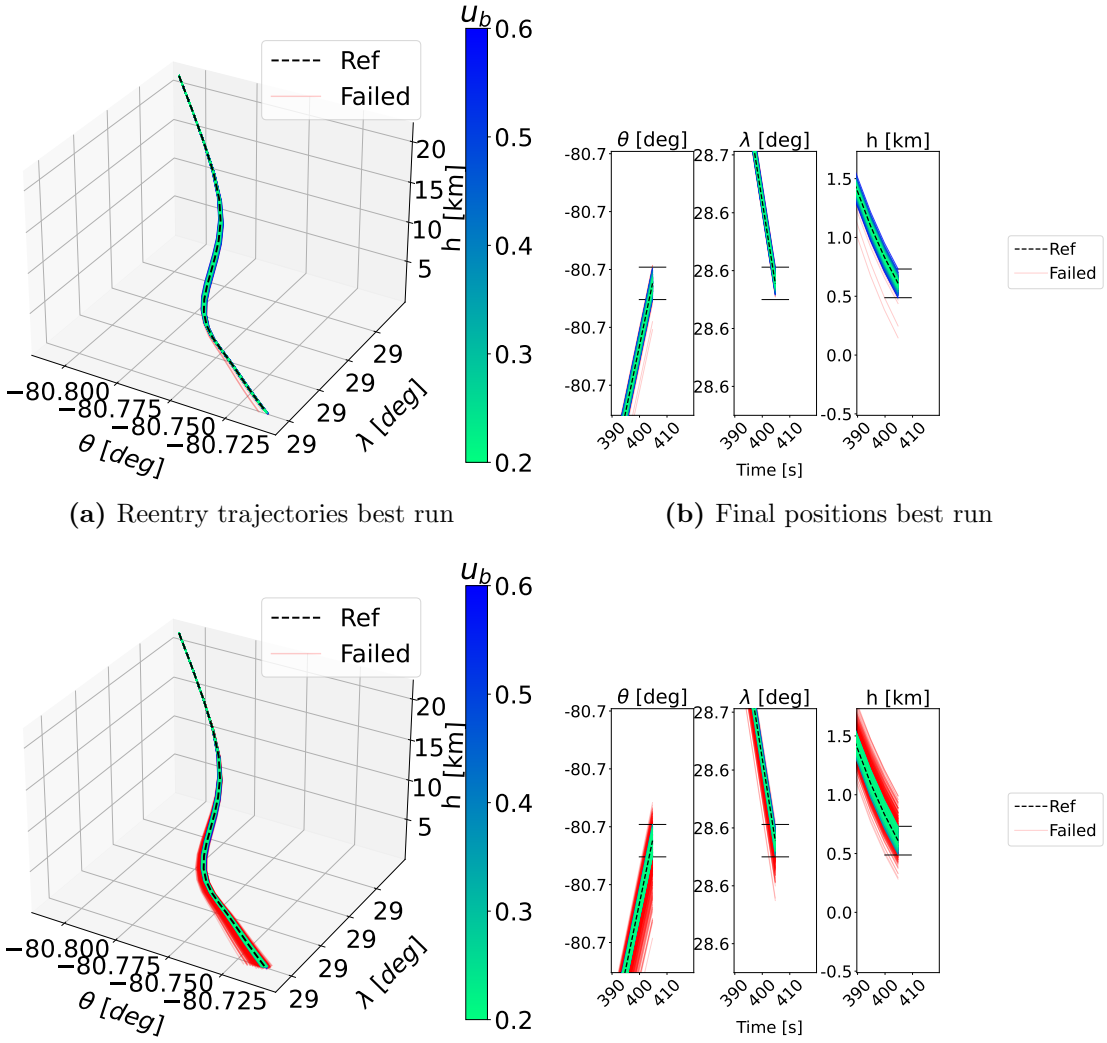
Table 5.8: Results of the statistical analysis performed using the standalone GP controller

Run ID	Fitness values		Test success rate				
	F_f	P_f	$u_b = 0.2$	$u_b = 0.3$	$u_b = 0.4$	$u_b = 0.5$	$u_b = 0.6$
1	0.00440	0	100%	91%	80%	74%	65%
2	0.00403	0	100%	99%	90%	82%	77%
3	0.116	0	81%	66%	53%	40%	26%
4	0.000731	0	100%	100%	99%	94%	88%
5	0.00214	0	100%	99%	90%	85%	68%
6	0.00192	0	100%	100%	97%	93%	85%
7	0.00315	0	100%	100%	100%	99%	90%
8	0.00139	0	100%	99%	90%	82%	76%
9	0.00244	0	99%	99%	91%	84%	78%
10	0.0405	0	96%	77%	54%	40%	28%
11	0.000770	0	100%	98%	94%	87%	81%
12	0.153	0	96%	71%	53%	37%	23%
13	0.00641	0	100%	100%	95%	88%	85%
14	0.273	0	65%	46%	32%	22%	13%
15	0.0753	0	94%	86%	75%	68%	62%
16	0.00613	0	90%	70%	61%	50%	38%
17	0.670	0	76%	52%	30%	16%	13%
18	0.00221	0	100%	100%	100%	98%	90%
19	0.00204	0	100%	100%	100%	93%	86%
20	0.02246	0	94%	80%	55%	39%	30%
Median	/	/	100%	98.5%	90%	82%	72%
Standard Deviation	/	/	9.66%	17.42%	23.46%	27.17%	28.34%

While the GP achieves a broad range of success rates, from excellent performance (e.g., $u_b = 0.4$ and $u_b = 0.5$) to success rates comparable to those obtained with open-loop commands (e.g., $u_b = 0.5$ and $u_b = 0.6$), the GANNIC approach exhibits far greater robustness. Moreover, while median values for GP results align with those of unpruned GANNIC, they are visibly inferior to those of the pruned GANNIC.

Figure 5.27 offers another illustration of the GANNIC scheme's consistency in comparison to the standalone GP. The plot depicts the evolution of the F_f fitness for the best individuals. The solid lines represent the means, and the shaded areas denote the standard deviations evaluated on the performed twenty runs. As can be observed, the GP's standard deviation is around one order of magnitude greater than that of GANNIC. Additionally, the evolutions performed with the GANNIC scheme exhibit faster convergence to lower fitness values.

These findings lead to the conclusion that, although the standalone GP guidance



(a) Reentry trajectories best run

(b) Final positions best run

(c) Reentry trajectories worst run

(d) Final positions worst run

Figure 5.25: Reentry trajectories and final positions of the best (run 7 in Table 5.8) and worst (run 14 in Table 5.8) runs performed with the standalone GP scheme. The red lines represent the failed trajectories. The black horizontal bars in the final position plots represent the FAC box boundaries. [2]

algorithm can attain peaks of exceptional performance, the pruned GANNIC scheme shows superior performance and robustness from a statistical point of view, by consistently maintaining a higher median success rate and a lower standard deviation across all considered u_b values. This is particularly visible when analyzing the results for higher values of u_b. Furthermore, the GP evolution within the GANNIC scheme converges more rapidly to a lower fitness compared to using the standalone GP guidance

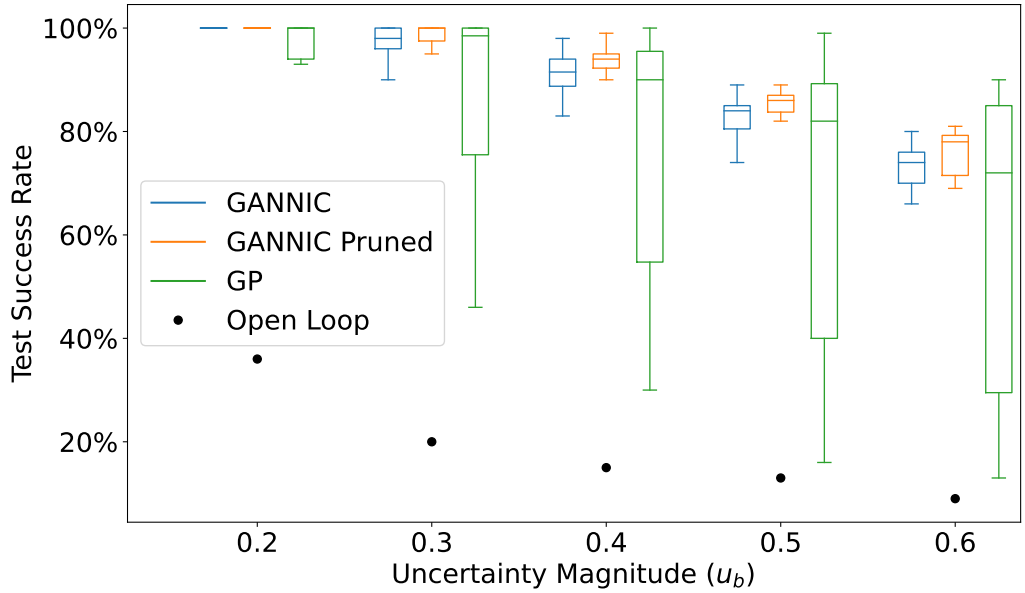


Figure 5.26: Results of the statistical study. The blue box plots are evaluated considering 20 runs using the GANNIC scheme. The orange boxplots are evaluated with the pruned GANNIC scheme, and the green boxplots using the standalone GP control scheme. [2]

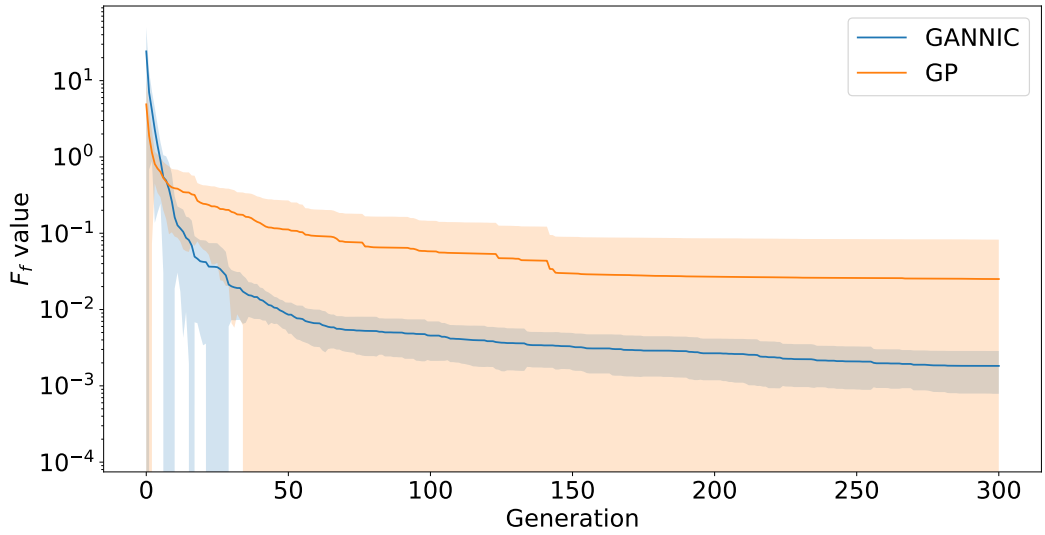


Figure 5.27: Evolution of the F_f value for the GANNIC and GP control approaches. The continuous lines are the mean, while the shaded areas represent the standard deviations. Both mean and standard deviation are evaluated by considering the best individuals in the 20 runs.

scheme. As a final remark, it is clear how both the GP and the GANNIC schemes can successfully guide the vehicle through a significantly wider range of uncertainties than what is achievable with open-loop guidance commands. This observation underscores the importance of an adaptive element in the guidance scheme to effectively manage complex, uncertain environments.

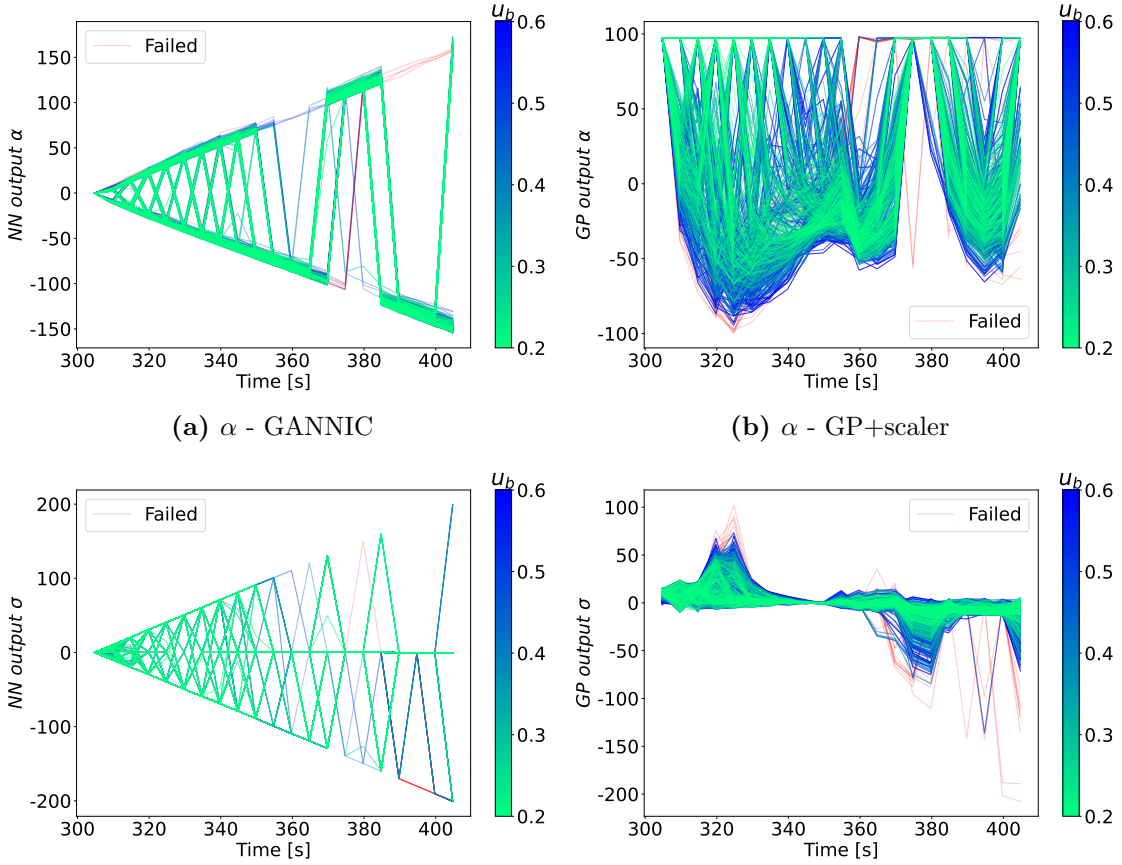
Scaler's role investigation

To investigate the heightened robustness observed in GANNIC, the identical scaler used in the GANNIC scheme was applied to the standalone GP guidance algorithm to bound its output. This configuration underwent testing through twenty runs, replicating the same settings used for GANNIC and the standalone GP cases. The complete results are provided in Appendix D. Contrary to anticipated improvements, the outcomes indicate an overall degradation in performance compared to the standalone GP controller. This behaviour is possibly attributed to the shape of the scaler's input signal as depicted in Figure 5.28, illustrating results selected from the best of the twenty runs in both the GANNIC and GP+scaler schemes. Notably, the NN employed in GANNIC generates an expanding bang-bang control signal, whereas the GP output, fed into the scaler, exhibits a more chaotic pattern. This consistent structure of the NN output, observed across runs, is likely influenced by the use of the tanh activation function in the hidden layers.

Considering the observed results, it can be inferred that the scaler alone is not capable of producing an effective bounded guidance signal. Therefore, the NN assumes a main role in ensuring overall output boundedness, likely due to the use of the tanh activation function in the hidden layer.

5.4 Summary and Comments

This chapter introduces the Genetically Adapted Neural Network-based Intelligent Controller (GANNIC), providing a description of the underlying framework and practical application showcasing its usage. Functioning as an IC scheme, GANNIC can be applied



(a) α - GANNIC

(b) α - GP+scaler

(c) σ - GANNIC

(d) σ - GP+scaler

Figure 5.28: Scaler input signals in the GANNIC and GP+scaler schemes. These results are picked from the best of the 20 runs in the GANNIC and GP+scaler schemes. [2]

both as a controller and a real-time guidance scheme. Comprising a NN responsible for generating nonlinear guidance or control commands, GANNIC adapts its weights online to address variations in environmental or system parameters. Online weight adaptation is achieved by propagating the NN's weights through time using a set of differential equations determined offline through GP. Each weight is adapted by a distinct GP equation. GANNIC offers an innovative approach to the design of adaptation laws for NN weights. Unlike conventional methods that rely on analytically derived differential equations, often cumbersome and challenging when dealing with models composed of tabular data, GANNIC autonomously generates adaptation laws for weights through interaction with the target plant. This autonomous adaptation capability eliminates

the need for explicit plant models. Engineers can focus solely on defining the GP fitness function to achieve their desired objectives.

The GANNIC scheme was applied to perform the reentry guidance of the FESTIP-FSSC5 RLV. The guidance commands, namely the angle of attack α and bank angle σ, were generated online in real-time to guide the vehicle towards the desired final position in the presence of uncertainties applied to the atmospheric models. The proposed scheme exhibits robustness to a broad range of uncertainties. A total of twenty GP evolutionary processes were conducted to evaluate the repeatability of the GP algorithm. For each run, GP adaptation laws were generated offline using a predefined set of training uncertainty profiles. Subsequently, the GP models were integrated into the GANNIC scheme, which underwent testing by propagating 100 trajectories under 100 unknown test uncertainty profiles. For each of the twenty runs, this set of 100 trajectory propagations was repeated five times, varying the magnitude of applied uncertainties. The GANNIC scheme achieved an average success rate of 91.6% across all uncertainty magnitudes, proving its robustness against unknown uncertainties. A comparative analysis with a standalone GP guidance algorithm was performed. GANNIC exhibited superior statistical performance, indicated by higher median values, along with enhanced statistical robustness underscored by a lower standard deviation. In particular, the average median success rate, evaluated considering the median success rates on different magnitudes of uncertainties, is 91.6% for GANNIC and 88.5% for the standalone GP. The average standard deviation is 4.34% for GANNIC and 21.21% for the standalone GP.

The conducted experiment shed light on some noteworthy characteristics of both the GANNIC and the treated test case. The NN topology optimization process highlighted the great importance of input variables associated with quantities possessing physical relevance to the treated problem. Specifically, the optimization revealed that variables θ, λ, and h exhibit negligible influence, as can also be observed through an analysis of the equations of motion. The generated commands for the angle of attack (α) and bank angle (σ) displayed an observable pattern, suggesting that substantial deviations from their reference values may lead to failure. This information can be leveraged to

further tune the hyperparameters of the scaler.

Furthermore, an analysis of the importance of a bounded guidance command was conducted by analyzing the scaler's impact. A comparison between the GANNIC and the GP guidance schemes, both augmented with the same scaler, highlighted the negligible influence of the scaler itself. However, the comparison emphasized the importance of the bounding mechanism embedded in the NN. Specifically, the tanh activation function in the hidden layer generates a bounded signal that is more effectively scaled by the scaler. The bounded nature conferred by the NN contributes to a heightened robustness of the overall guidance scheme.

The GANNIC scheme is currently in the developmental stage, and as such, several issues impact its performance and design process. These challenges are outlined in the following discussion.

GANNIC Design Process

The GANNIC scheme involves several components, each of them necessitating an individual analysis to understand their critical aspects. For instance, the decision on how to select and apply the uncertainties during GP training is of great importance. While the presented solution is effective for the considered problem, a comparative study across diverse test cases could offer a more profound understanding of the impact of uncertainties during training. Another crucial component is the NN. A feedforward network was chosen for its simplicity, but it might not be the optimal configuration. Alternative NN structures my lead to better performance. A comprehensive understanding of the NN's influence on GANNIC outcomes is essential. As highlighted in section 5.3.6, it is evident that the use of the NN yields more consistent results than the standalone GP controller, and this behaviour is not solely attributed to the scaler. However, the ultimate results stem from complex interactions involving the propagated NN weights, the NN topology, and the scaler.

Scaler's Hyperparameters

The GANNIC system is currently suboptimally configured, with hyperparameters p_1, p_2, u_{b_S}, and l_{b_S} defined in section 5.2. While the existing parameter values have shown satisfactory results, a thorough investigation is necessary to optimize the system's performance.

Statistical Sample

Using a personal laptop, the computational time required for a single GP evolutionary process, both for the standalone GP and the GANNIC controllers, was substantial. Specifically, the average computational time for one run during standalone GP evolution was approximately 4 hours and 20 minutes, while for the GANNIC scheme, it was around 19 hours and 40 minutes. It is crucial to note that this time pertains to the offline GP evolution. The online learning in the GANNIC scheme occurs almost instantaneously using equations determined offline. Consequently, to generate results within a reasonable timeframe, only 20 runs per test case were conducted to produce the statistical data.

Comparison with Other Guidance Schemes

GANNIC was exclusively compared with a GP-based guidance scheme for two primary reasons: 1) the GP scheme applied to the FESTIP-FSSC5 had been analyzed in a prior study, and the models were already accessible; 2) no other guidance scheme applied to the FESTIP-FSSC5 could be found in the existing literature. Particularly, the latter point introduced uncertainty regarding what could be considered as the reference for this vehicle. Consequently, it was impractical to test and optimize numerous algorithms to identify the best-performing one and designate it as the reference. To facilitate further research and comparison of G&C schemes for this vehicle, the used models are publicly available at `https://github.com/strath-ace/smart-ml/tree/master/GP`.

A more detailed discussion on these aspects is provided in Section 6.2.

Chapter 6

Conclusions

This project was carried out to investigate the domain of Intelligent Control (IC) and assess the applicability of Genetic Programming (GP), both alone and combined with a Neural Network (NN), within an IC framework for the real-time guidance of a Reusable Launch Vehicle (RLV). This chapter presents an overview of the findings derived from the research, describing the present status of the developed methodologies while discussing their inherent constraints. The last section presents a discussion on directions for future research, aiming at refining the proposed methodologies and enriching the academic literature on IC systems.

6.1 Summary and Contributions

The carried out research was guided by three primary objectives: 1) investigate the state-of-the-art of IC to understand how GP and NNs are applied in this context. 2) Advance the state-of-the-art of GP applied to IC. 3) Develop an innovative IC scheme by hybridizing GP and NNs for the real-time guidance of a RLV.

The first objective was addressed in chapters 2 and 3. Chapter 2 presents a comprehensive literature review of control approaches applied to RLVs, identifying four main families: optimal control, robust control, adaptive control, and Artificial Intelligence (AI)-based and IC. The most frequently employed methods for RLVs within these families were described, accompanied by illustrative examples of their applica-

tions. Following this analysis, a comparative examination of these control families was conducted to underscore how IC could enhance the RLVs industry. This comparison considered the general characteristics of the discussed control families and did not intend to analyze every specific control approach developed in the literature. Consequently, certain control approaches may exist that are not afflicted by the common issues of their respective control families. The comparison revealed that IC has the potential to augment the robustness of existing control approaches against disturbances and uncertainties by leveraging the nonlinearity introduced by AI techniques, the capability to adapt online, and the efficient use of real-time data. However, it was noted that many AI-based and IC approaches often lack a stability analysis framework and are frequently characterized as black-box models, compromising their explainability.

Chapter 3 presents a comprehensive literature review of IC applications. This review, conducted on behalf of the European Space Agency (ESA), resulted in the production of two technical reports [14, 15]. Two major remarks stemmed from the review: 1) the term IC is oftentimes misused; 2) not all IC approaches possess the same level of intelligence. To address these findings, a novel taxonomy for IC applications was developed. It was initially introduced in [19] and subsequently refined in [1]. This taxonomy served a dual purpose: firstly, as a tool to assess the legitimacy of an application being classified as IC; secondly, as a classification scheme to delineate the intelligence levels inherent in different applications, clarifying the factors that make one application more intelligent than another. The proposed taxonomy revolves around the control system's ability to manage uncertainties based on the knowledge available at the time of design, encompassing three key aspects: goal knowledge, environment knowledge, and control system knowledge. Each of these aspects is categorized into five levels, ranging from 0 (non-intelligent) to 4 (highly intelligent). The taxonomy was subsequently applied to the analyzed applications to evaluate its consistency and applicability.

Chapter 4 details the development of the second objective of this thesis. It begins with the description of the theoretical aspects of GP and its application within an IC and, more broadly, a Guidance and Control (G&C) framework.

Subsequently, the application of GP in an IC setting is illustrated, specifically for real-time ascent guidance of a Goddard rocket, initially presented in [169]. This application serves as an illustrative example of the potential of GP within an IC framework. In this context, GP proved its capability to generate online a guidance law for tracking a reference trajectory in the presence of applied disturbances. These include a variation in the drag coefficient (c_d) representing changes in plant geometry during flight; wind gusts representing environmental disturbances; and variations in the adopted air density model during flight, simulating a partially known environmental model updated online during the mission. In all three scenarios, GP effectively generated a guidance law online within a limited time interval, highlighting the advantages of its online usage. This feature is currently limited to plants with a low degree of nonlinearity or slow-varying, due to the cumbersome computational cost of GP. Nonetheless, it will be extended to more complex systems with future technological advancements.

The chapter concludes with the description of the Inclusive Genetic Programming (IGP) algorithm. Initially presented in [20] and further explored in [21], IGP is an innovative GP heuristic designed to promote and maintain diversity in a GP population. Specifically tailored for G&C applications, IGP addresses the criticality of population's diversity in these contexts, as larger (i.e., more complex) individuals showed greater efficiency in capturing plant nonlinearities and, traditionally, GP algorithms tend to evolve smaller individuals for improved interpretability. Through testing on a set of regression problems and comparison with a standard GP implementation, IGP exhibited superior performance due to its diversity maintenance and promotion mechanisms.

Chapter 5 introduces the concluding contribution of this dissertation, the Genetically Adapted Neural Network-based Intelligent Controller (GANNIC) scheme. This control paradigm was formulated leveraging insights gained from prior experiments with GP to enable its online application within an IC framework. As discussed in this thesis, the use of GP alone in an IC setting is suitable only for systems with a low degree of nonlinearity or slowly varying. To overcome this limitation, a NN is employed as

a nonlinear controller, and GP is applied to generate a set of differential equations to dynamically adjust the NN's weights online. This approach allows for the online updating of NN parameters in response to environmental or system variations.

The chapter concludes with the application of the GANNIC scheme in the real-time reentry guidance of the FESTIP-FSSC5 RLV. The test case involved the online adaptation of guidance commands to track a reference trajectory in the presence of uncertainties in environmental models. The experiments demonstrated the superiority of the GANNIC scheme in comparison to the reference guidance open-loop commands, allowing to successfully guide the vehicle over a wide range of uncertainty with varying magnitudes. Additionally, a comparative analysis with a standalone GP guidance scheme shows the superior statistical robustness and performances of GANNIC.

In summary, the research and experiments presented in this doctoral dissertation lead to the following conclusions:

- There is a general misconception of what IC is, leading to the improper use of the term by many researchers. Moreover, there was no structured framework to assess the level of intelligence of IC applications. To address these issues, the taxonomy described in chapter 3 was developed.

- The developed research showed that GP can be successfully employed in an IC setting. It can be applied to nonlinear systems without requiring linearization and it can handle constraints. For the considered real-time guidance application, GP showed the capability of generating online a guidance law to adapt to unforeseen disturbances. It is capable of doing so only by interacting with the plant and environment, without relying on previous knowledge of them. Nevertheless, the application of GP to highly nonlinear systems is currently limited by its computational costs. While this poses a challenge at present time, it is anticipated that the issue will become increasingly negligible as hardware technologies continue to advance rapidly.

- The novel GP heuristic developed for this project, the IGP, shows superior per-

formance than the standard GP and can be successfully applied in a G&C framework.

- The presented research showed that GP can be successfully employed to design the NN's weights adaptation algorithm. The resulting G&C scheme, named GANNIC, was applied to perform the real-time guidance of a RLV. It proved robust against a broad range of unforeseen uncertainties showing superior performance w.r.t. the open-loop reference commands and a GP-based guidance scheme.
- These experiments conducted with both GP and GANNIC highlighted the importance of incorporating an adaptive element into the guidance scheme to adapt to complex and uncertain environments. This is particularly true for RLVs, which operates in a large flight envelope and could greatly benefit from the online adaptation of the guidance commands to cope online with unforeseen situations.

6.2 Current Limitations and Future Work Directions

The conducted research aimed at developing novel IC algorithms involving GP. Because of the foundational work being performed, the focus was placed on the algorithms development and testing. An investigation to improve the explainability of the obtained schemes and to perform a stability analysis were not conducted, since they represent different research branches in themselves. Considering these aspects, and the issues pertaining to the GP and GANNIC G&C schemes, the following limitations and areas for improvement can be identified.

6.2.1 Interpretability and Explainability

The selection of GP for designing an innovative IC scheme was motivated, in part, by its inherent interpretability. However, having interpretable models is not enough to convey useful information to the user. To this end, future research should focus on developing a GP algorithm capable of producing models based on the physical features of the treated problem. To do so, several strategies could be pursued, such as introducing appropriately formulated constraints or developing fitness functions that

explicitly encourage the evolution of models with greater physical relevance. This could be done similarly to Physics Informed Neural Networks (PINNs) [241], where the physics of the treated problem is embedded into the NN and the loss function contains a physics-informed regularization term.

Concerning the GANNIC scheme, interpretability and explainability of the overall scheme may pose a challenge due to the predominant role played by the NN, which inherently functions as a black-box model. Nonetheless, the experiments detailed in chapter 5 highlight that a compact NN configuration suffices for the real-time guidance task under consideration. In such instances, it becomes feasible to comprehensively analyze the complete analytical model. Therefore, future research could focus on improving interpretability and explainability of the GANNIC scheme by developing a framework to analyze both the NN and the GP models simultaneously and their interaction.

6.2.2 Stability Analysis

Stability analysis represents a central element in control systems. Although stability analysis methods have been developed for AI-based or IC schemes, gaps exist in the literature regarding the discussion of stability properties, and a common framework for AI-based control schemes is lacking. Notably, Lyapunov stability analysis can be applied to the G&C schemes developed in this thesis. Existing works in the literature, such as [127, 13, 242], have explored embedding the search for a suitable Lyapunov function into the GP fitness function. This innovative approach simultaneously produces both the desired Lyapunov function and a suitable control scheme. However, the widespread adoption and applicability of this approach across different plants require further investigation.

In the context of the GANNIC scheme, the stability analysis is made more challenging by the interaction between the GP and the NN. Therefore, both the stability of NN's weights adaptation laws and the overall scheme should be assessed.

Future research endeavours should focus on examining the incorporation of Lyapunov stability analysis into the GP algorithm and testing its applicability to complex nonlinear systems. Additionally, investigations into the characteristics of the resulting

region of attraction derived from the designed Lyapunov function are required. Understanding whether the obtained region of attraction holds physical significance is crucial for assessing the practical implications of the stability analysis and refining the design process. This exploration will contribute to the advancement of stability-assured AI-based and IC control schemes, fostering their applicability across diverse systems and enhancing their robustness in real-world applications.

6.2.3 Comparison with established approaches

The comparison of the developed algorithms against state-of-the-art methodologies presents a significant challenge, particularly in the context of the selected test case. The literature currently lacks research pertaining to the guidance of the FESTIP-FSSC5 vehicle, indicating a necessity for comprehensive preliminary studies to identify the state-of-the-art methodologies applicable to this vehicle and its specific mission.

Moreover, within the broader RLV domain, there is an absence of a standardized approach to real-time guidance, complicating the identification of a common state-of-the-art. Recent years have seen the emergence of various methodologies deemed state-of-the-art for distinct applications. For instance, convex optimization techniques have been effectively applied to Vertical Take-Off and Vertical Landing (VTVL) vehicles. An alternative comparative analysis could involve Reinforcement Learning (RL) techniques which, alongside other AI-based methods, are yet to be recognized as state-of-the-art in real-world applications. Nonetheless, such a comparison could elucidate the learning mechanisms of NNs within the GANNIC and RL frameworks, highlighting their respective advantages and limitations.

Although such a comparison work could potentially give highly valuable insights, its execution would have required time and computational resources beyond those available for this project, and it is left for future work.

6.2.4 GANNIC Design Process

The Genetically Adapted Neural Network-based Intelligent Controller (GANNIC) scheme involves several key steps that require individual investigation to comprehend their crit-

ical aspects. For instance, the method to select and apply the uncertainties during the Genetic Programming (GP) training phase is crucial. While the approach presented in this thesis has proven effective for the considered problem, a comparative study involving different test cases could provide deeper insights into the influence of uncertainties during training.

Another critical aspect deserving further investigation is the NN agent. The use of a feedforward network, while straightforward, may not represent the optimal configuration. For instance, architectures like Recurrent Neural Network (RNN) or Transformer Networks, which possess the ability to recognize temporal patterns, could offer significant advantages for guidance applications. A comprehensive exploration of various NN structures is required to identify the most suitable architecture for the GANNIC scheme, considering performance, complexity, adaptability and interpretability.

Overall, a more in-depth understanding of the NN's impact on GANNIC results would be beneficial. As highlighted in section 5.3, it is evident that the use of the NN yields more consistent results than the standalone GP scheme, and this improvement is not solely attributable to the scaler. However, the final outcomes result from complex interactions involving the propagated NN weights, the NN topology, and the scaler. Investigating these interactions and their implications on the overall system performance will contribute to refining the GANNIC scheme for broader applicability and effectiveness across various scenarios.

GANNIC Hyperparameters

The existing configuration of the Genetically Adapted Neural Network-based Intelligent Controller (GANNIC) scheme is acknowledged to be suboptimal, relying on a set of hyperparameters, namely the tanh shaping parameters p_1 and p_2, and the scaler bounds u_{b_s}, and l_{b_s} as described in Subsection 5.2.1. While the current values of these parameters have yielded favourable results, a thorough investigation is needed to further enhance the system's performance.

One potential avenue for improvement involves exploring an approach that eliminates the use of the specified hyperparameters and, consequently, the scaler. This ad-

justment aims at increasing the GANNIC system's adaptability to different plants. The proposed approach involves constraining the GP to generate algebraic convex models. The use of convex models will facilitate the discovery of the maximum and minimum values of the generated functions, consequently providing insight into the maximum and minimum output of the NN controller, particularly when using a tanh activation function.

Further investigation and experimentation are required to validate the effectiveness of this approach, assessing its impact on the GANNIC system's performance and applicability across a broader spectrum of test cases.

Statistical Sample

The computational time required for simulations using both the GP and GANNIC schemes on a personal laptop was considerable. Because of that, a limited number of simulations were conducted to generate statistical data. A more extensive set of simulations would enhance the statistical reliability of the controllers' performance evaluation.

While the limitations in computational resources influenced the scope of the simulations, this acknowledgement underscores the potential for future work to delve deeper into the statistical aspects of controller performance. Expanding the number of runs would not only contribute to a more robust understanding of the controllers' capabilities but also provide additional insights into their behaviour under varied conditions and scenarios.

Appendix A

Results of second Neural Network Topology Optimization

This appendix contains the results of the second iteration of the NN topology optimization process for the GANNIC scheme. The results are presented in Table A.1, which reports the success rate on the test uncertainty scenarios for the best individuals of each run.

Table A.1: Test success rates obtained with the 3-3-2 NN configuration in the four runs performed.

run ID	Fitness values		Test success rate
	F_f	P_f	$u_b = 0.2$
1	0.001838	0	100%
2	0.001906	0	100%
3	0.001135	0	100%
4	0.002859	0	94%

By comparing the results in Table A.1 with those in Table 5.6, it can be seen how by pruning the Neural Network (NN) and therefore allowing the Genetic Programming (GP) to focus on evolving only the necessary weights adaptation laws, led to a performance increase. A reduced network configuration also led to a faster convergence, as shown in Figure A.1. In this Figure, the evolution of the mean F_f values for the full and reduced NN configurations are plotted as a continuous line. The shaded areas represent the standard deviation of the F_f fitness values. The mean and standard devi-

ations are evaluated considering the four runs with the complete net configuration and the four runs with the reduced net configuration. The full network configuration can reach a lower fitness value, but this is not representative of the generalization capabilities shown by Tables 5.6 and A.1. In fact, a lower fitness value can also be a symptom of overfitting.

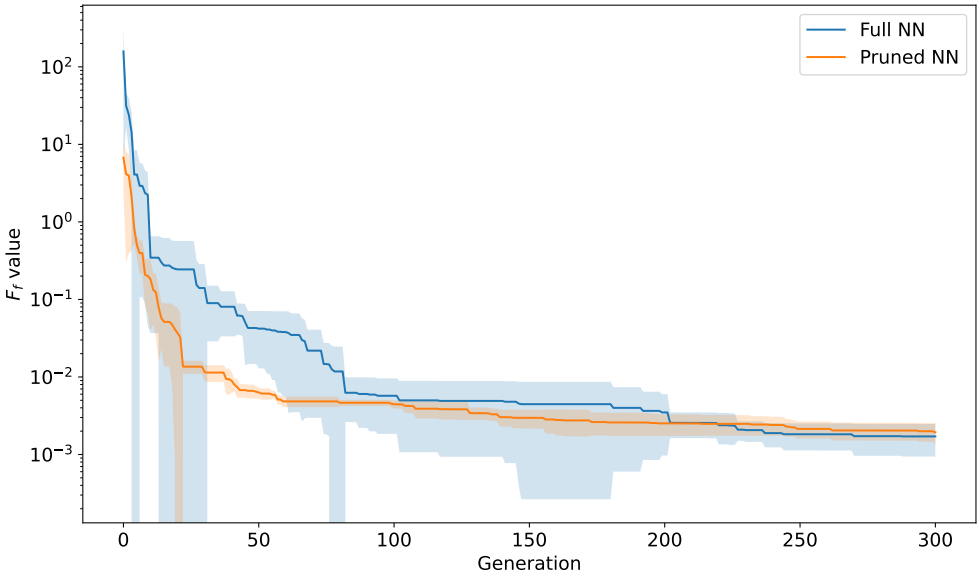
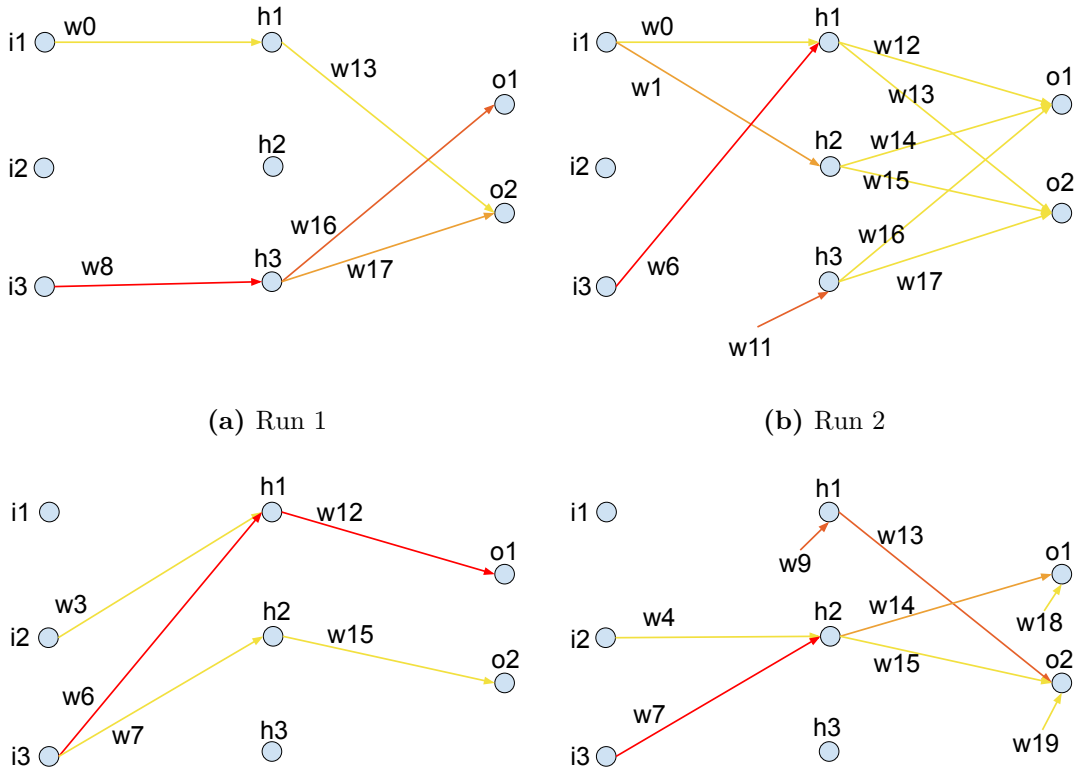


Figure A.1: Evolution of fitness F_f for the full and reduced network configuration. The continuous lines represent the mean fitness value for each generation considering the four performed runs. The shaded areas represent the standard deviation.

The results obtained from the second iteration of the NN topology optimization process are summarized in Figure A.3. A less clear image emerges than Figure 5.20. As expected, more weights are influential compared to the entire network configuration: 18/38 (47%) in the whole network, against 17/20 (85%) for the reduced network. These percentages are evaluated considering a weight influential if it is such in at least one out of four runs. Also, many weights were found to be detrimental in run 4, but this might be due to an evolution that did not reach satisfactory results in 300 generations.

The weights maps are plotted in Figure A.2. Contrarily to the first iteration, now it is unclear whether some weights could be removed. Biases seem unimportant in two of the four runs, but they are used in the other two, and when removed, the performances drop significantly. Also, the input $\bar{e}_v$ and $\bar{e}_\chi$ are never used together, but it is unclear

if one should be preferred over the other. Regarding the hidden neurons, only two of them are simultaneously used in three runs, but in run 2, they are all used. Instead, as for the previous iteration, $\bar{e}_\gamma$ is the most influential, leading to a 0% success rate when removed in all four runs.



(a) Run 1

(b) Run 2

(c) Run 3

(d) Run 4

Figure A.2: Graphical representation of the second iteration of the topology optimization process. The yellow arrow represents those weights that, when removed, cause a slight decrease in performances; light and dark orange represent those weights that, when removed, cause a medium and high decrease in performances, while the red arrow represents those weights that, when removed, cause the success rate to drop to zero.

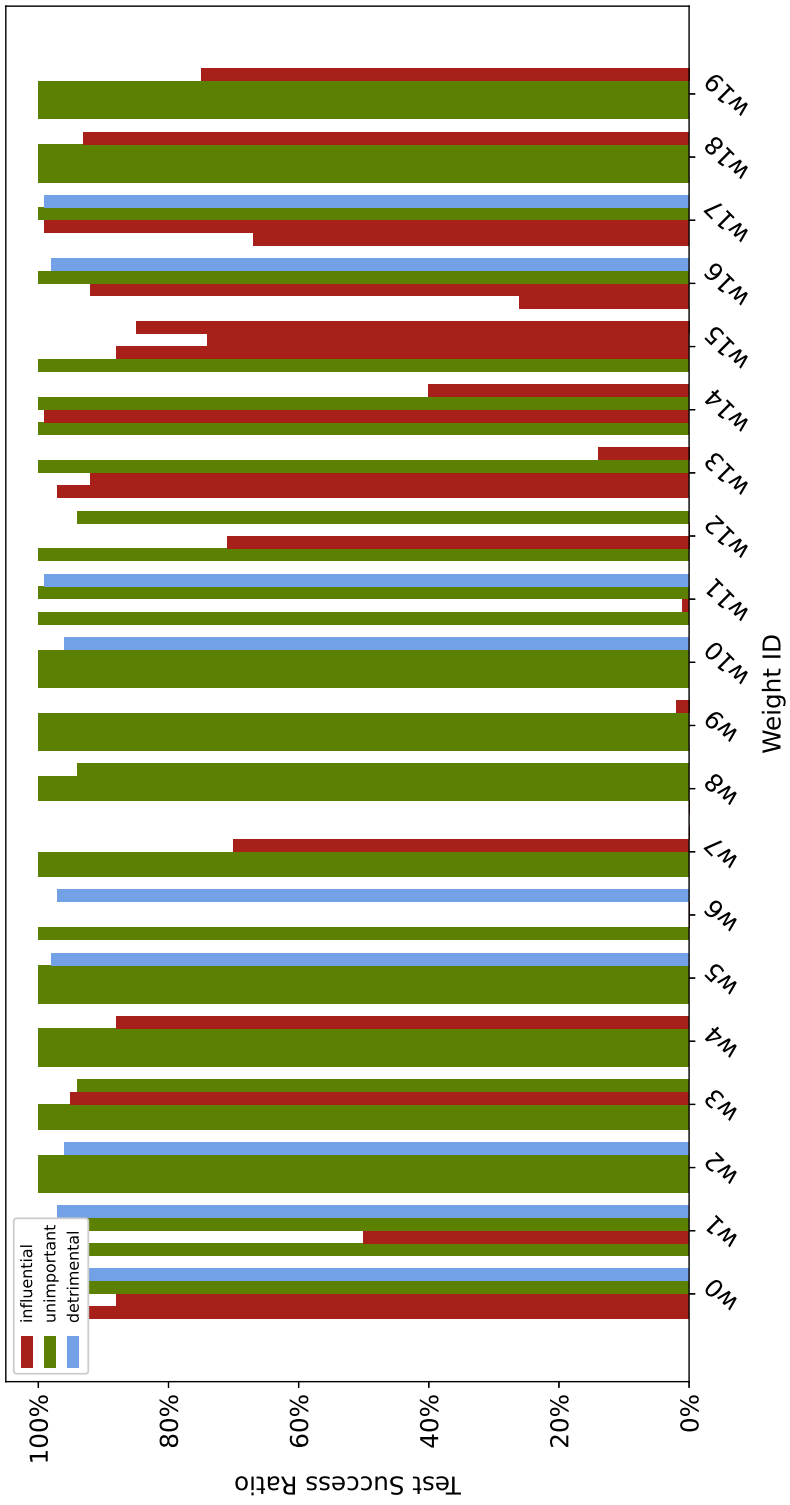


Figure A.3: Results of the second iteration of the NN topology optimization process.

Appendix B

Best Individual Selection Process

This Appendix contains the description of the best individual selection process for each of the 20 runs performed with the GANNIC scheme. This is done to avoid overfitting. The approach summarized in Algorithm 11 was used. Briefly, the evolution of the F_f fitness value was monitored for each run, and every time a decrease was observed, the corresponding individual was picked. These individuals were tested by performing 100 trajectory propagations using 100 test uncertainty scenarios from $\mathbf{U}_{test}$. This process was repeated N_{u_b} times by increasing the value of u_b - i.e. the upper bound of the applied uncertainty - by 0.1 at every iteration, starting from $u_b = 0.2$ up to $u_b = 0.6$. Therefore, N_{u_b} in Algorithm 11 was set as $N_{u_b} = 5$. This was done to assess the generalization capabilities of the individual when considering uncertainties with different shapes and magnitudes. For each u_b considered, a success rate S_r on the 100 performed runs was evaluated. Having obtained N_{u_b} success rates, the mean was evaluated to obtain the average success rate $\bar{S}_r$ for that individual. Therefore, at the end of each of the 20 runs, the individual with the highest average success rate was picked as the best one for that run and used for the statistical analysis discussed in the following.

This research for the best-performing individual was performed since the final individual produced by the Genetic Programming (GP) evolution, i.e. the one with the lowest fitness value, might not be the one with the best generalization capabilities since it might overfit the training uncertainty scenarios. This behaviour is depicted in Figure

Algorithm 11 Pseudocode of the algorithm adopted to select the best individual from each run

for $i = 1 \rightarrow N_{sim}$ **do**
  for $j = 1 \rightarrow N_{gen}$ **do**
    if $F_{f_j} < F_{f_{j-1}}$ **then**
      for $k = 1 \rightarrow N_{u_b}$ **do**
        Test j-th individual on N_{test} uncertainty scenarios using $u_{b_k} = 0.2 + 0.1(k-1)$
        Save the success rate $S_{r_{jk}}$
      end for
      Find the average success rate $\bar{S}_{r_j}$ for the j-th individual
    end if
  end for
  Best individual of the i-th run $\leftarrow$ individual with $\max_{1 \leq j \leq N_{gen}} (\bar{\mathbf{S}}_{r_j})$
end for

B.1, where the average test success rate is plotted against the number of generations. These data are obtained from the individuals in run 2 listed in Table 5.7. Each dot in the Figure corresponds to a different individual who caused a decrease in the F_f value at the corresponding generation. The red dot represents the best-performing individual. From this Figure, it is clear how the best-performing individual is found at generation 57, and then overfit happens, leading to a decrease in the average success rate.

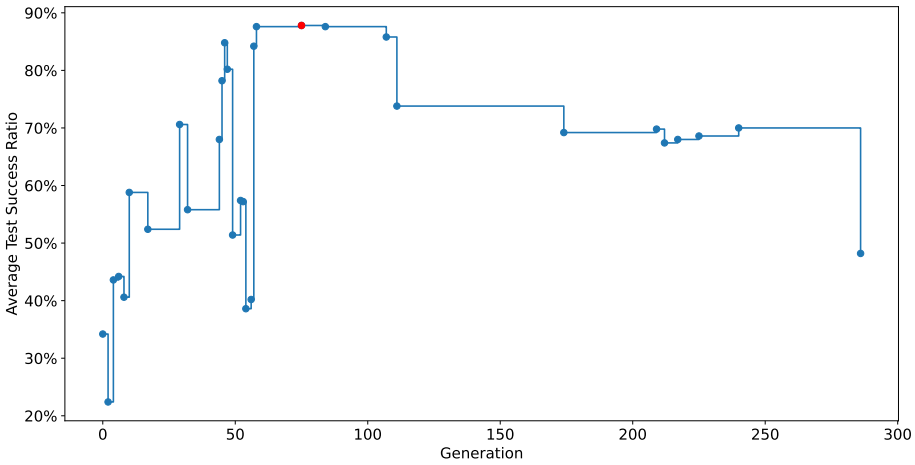


Figure B.1: Average test success rate evolution in run 2. The best-performing individual is the red dot.

Appendix C

Complete results of the Neural Network fine-pruning before deployment

The results of the NN fine-pruning before deployment are shown in Figures C.1 to C.5. These pictures show a set of tables containing the success rates obtained by removing the weights one by one and performing 100 trajectory propagations using the uncertainty scenarios from $\mathbf{U}_{test}$. In these tables, the rows list the results for the different weights, and the columns refer to the uncertainty magnitudes considered. An exception is made by the first row, which contains the reference success rates from the unpruned topology and the last column, which contains the row-wise average of the success rates obtained on all the magnitude of uncertainties by setting the considered weight to 0. So, for example, by looking at Figure C.1a, the data in the second row and first column cell says that by setting the weight w_0 to 0, the success rate is 100% using an uncertainty magnitude with $u_b = 0.2$. The adopted colour scheme has the following meaning: the green cells represent a success rate greater than the reference; the yellow cells a success rate lower than the reference and greater than 66%; the dark yellow cells a success rate lower than the reference and between 66% and 33%; the orange cells a success rate lower than the reference and between 33% and 1%; the red cells a success rate of 0%. This colour scheme is adopted to highlight the severity

of the performance decrease when removing the considered weight. The performance improvement obtained with pruning is measured by comparing the obtained average success rate with the reference one, highlighted in green if greater than the reference. In most cases, the performances can be improved with pruning. Still, it can also happen that pruning does not contribute to performance improvement, as in the case of Run 3 Figure C.1c, and Run 11 in Figure C.1a. On the other hand, in some cases, many weights could be removed resulting in a performance increase, as for Run 12 shown in Figure C.3d. In most cases, the pruned topology with the highest success rate is picked as the new topology. Still, it was observed that combining two good pruned topologies can lead to even higher average success rates, as in the case of Run 1 shown in Figure C.1a. Here, it was observed that by removing both weights w_{18} and w_{19}, the average success rate becomes 92.4%, which is greater than those obtained by removing either w_{18} or w_{19}. On average, an increase of the average success rate of 2.12% is observed, with a maximum increase of 7.6% observed in Run 19 shown in Figure C.5c.

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**1.000**	**0.940**	**0.870**	**0.740**	**0.910**
w0	1.000	1.000	0.940	0.870	0.740	0.910
w1	0.900	0.780	0.670	0.620	0.500	0.694
w2	1.000	1.000	0.940	0.870	0.740	0.910
w3	1.000	1.000	0.940	0.870	0.740	0.910
w4	1.000	1.000	0.940	0.870	0.740	0.910
w5	1.000	1.000	0.940	0.870	0.740	0.910
w6	1.000	0.950	0.900	0.760	0.630	0.848
w7	1.000	1.000	0.940	0.870	0.740	0.910
w8	0.000	0.000	0.000	0.000	0.000	0.000
w9	1.000	1.000	0.940	0.870	0.750	0.912
w10	1.000	1.000	0.940	0.870	0.760	0.914
w11	1.000	1.000	0.940	0.870	0.740	0.910
w12	1.000	1.000	0.940	0.860	0.740	0.908
w13	1.000	0.980	0.900	0.760	0.630	0.854
w14	1.000	1.000	0.940	0.860	0.760	0.912
w15	0.940	0.830	0.740	0.610	0.540	0.732
w16	0.710	0.580	0.410	0.270	0.190	0.432
w17	0.970	0.870	0.750	0.590	0.520	0.740
w18	1.000	1.000	0.940	0.870	0.770	0.916
w19	1.000	1.000	0.950	0.870	0.770	0.918

(a) Run 1

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**0.940**	**0.870**	**0.830**	**0.750**	**0.878**
w0	1.000	0.940	0.870	0.830	0.750	0.878
w1	1.000	0.940	0.870	0.830	0.750	0.878
w2	1.000	0.940	0.870	0.830	0.750	0.878
w3	0.130	0.170	0.180	0.170	0.170	0.164
w4	1.000	0.940	0.870	0.830	0.750	0.878
w5	1.000	0.940	0.870	0.830	0.750	0.878
w6	1.000	0.940	0.870	0.830	0.750	0.878
w7	0.000	0.000	0.010	0.010	0.020	0.008
w8	1.000	0.940	0.870	0.830	0.750	0.878
w9	1.000	0.940	0.870	0.830	0.750	0.878
w10	1.000	0.940	0.870	0.830	0.750	0.878
w11	0.970	0.940	0.890	0.810	0.730	0.868
w12	0.760	0.610	0.490	0.400	0.350	0.522
w13	1.000	1.000	0.930	0.890	0.790	0.922
w14	0.000	0.000	0.000	0.000	0.000	0.000
w15	0.970	0.840	0.730	0.660	0.570	0.754
w16	0.980	0.930	0.890	0.830	0.730	0.872
w17	1.000	0.950	0.860	0.830	0.750	0.878
w18	0.770	0.630	0.560	0.490	0.430	0.576
w19	1.000	0.940	0.870	0.830	0.750	0.878

(b) Run 2

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**0.960**	**0.880**	**0.840**	**0.760**	**0.888**
w0	0.390	0.310	0.250	0.240	0.230	0.284
w1	1.000	0.960	0.880	0.840	0.760	0.888
w2	1.000	0.960	0.880	0.840	0.760	0.888
w3	1.000	0.960	0.880	0.840	0.760	0.888
w4	1.000	0.960	0.880	0.840	0.760	0.888
w5	0.870	0.760	0.740	0.690	0.580	0.728
w6	1.000	0.960	0.880	0.840	0.760	0.888
w7	0.000	0.000	0.000	0.000	0.000	0.000
w8	1.000	0.960	0.880	0.840	0.760	0.888
w9	1.000	0.960	0.880	0.840	0.760	0.888
w10	1.000	0.960	0.880	0.840	0.760	0.888
w11	1.000	0.960	0.880	0.840	0.760	0.888
w12	1.000	0.890	0.800	0.700	0.610	0.800
w13	0.820	0.670	0.550	0.470	0.440	0.590
w14	0.000	0.000	0.000	0.000	0.000	0.000
w15	1.000	0.940	0.820	0.700	0.570	0.806
w16	1.000	0.960	0.880	0.840	0.760	0.888
w17	1.000	0.960	0.860	0.810	0.670	0.860
w18	1.000	0.960	0.880	0.830	0.740	0.882
w19	1.000	0.960	0.880	0.840	0.760	0.888

(c) Run 3

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**0.990**	**0.960**	**0.850**	**0.790**	**0.918**
w0	1.000	0.990	0.960	0.850	0.790	0.918
w1	1.000	0.990	0.960	0.850	0.790	0.918
w2	0.480	0.420	0.350	0.290	0.250	0.358
w3	1.000	0.990	0.960	0.850	0.790	0.918
w4	0.850	0.800	0.750	0.720	0.690	0.762
w5	1.000	0.990	0.960	0.850	0.790	0.918
w6	0.000	0.000	0.000	0.000	0.000	0.000
w7	1.000	0.990	0.960	0.850	0.790	0.918
w8	1.000	0.990	0.960	0.850	0.790	0.918
w9	1.000	1.000	0.940	0.850	0.790	0.916
w10	1.000	0.990	0.960	0.850	0.790	0.918
w11	1.000	0.990	0.960	0.850	0.790	0.918
w12	0.710	0.490	0.370	0.250	0.190	0.402
w13	0.930	0.730	0.600	0.500	0.440	0.640
w14	1.000	0.990	0.960	0.850	0.790	0.918
w15	1.000	1.000	0.950	0.840	0.700	0.898
w16	1.000	0.990	0.910	0.840	0.770	0.902
w17	0.810	0.650	0.520	0.500	0.380	0.572
w18	1.000	0.970	0.950	0.850	0.780	0.910
w19	1.000	1.000	0.970	0.850	0.780	0.920

(d) Run 4

Figure C.1: Results of pruning performed on runs 1, 2, 3 and 4

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**0.980**	**0.910**	**0.820**	**0.660**	**0.874**
w0	1.000	0.980	0.910	0.820	0.660	0.874
w1	0.760	0.640	0.540	0.500	0.360	0.560
w2	1.000	0.980	0.910	0.820	0.660	0.874
w3	1.000	0.980	0.910	0.820	0.660	0.874
w4	1.000	0.980	0.910	0.820	0.660	0.874
w5	1.000	0.980	0.910	0.820	0.660	0.874
w6	1.000	0.980	0.910	0.820	0.660	0.874
w7	1.000	0.980	0.910	0.820	0.660	0.874
w8	0.000	0.000	0.000	0.000	0.000	0.000
w9	0.000	0.000	0.000	0.000	0.010	0.002
w10	1.000	0.980	0.910	0.820	0.660	0.874
w11	1.000	0.980	0.910	0.840	0.670	0.880
w12	0.140	0.100	0.080	0.060	0.050	0.086
w13	0.970	0.890	0.790	0.720	0.650	0.804
w14	1.000	1.000	0.950	0.820	0.700	0.894
w15	0.800	0.620	0.520	0.450	0.370	0.552
w16	0.440	0.310	0.210	0.140	0.110	0.242
w17	0.820	0.690	0.540	0.440	0.360	0.570
w18	0.460	0.400	0.330	0.290	0.280	0.352
w19	1.000	0.970	0.840	0.770	0.710	0.858

(a) Run 5

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**0.960**	**0.930**	**0.840**	**0.740**	**0.894**
w0	0.860	0.720	0.630	0.540	0.430	0.636
w1	1.000	0.960	0.930	0.840	0.740	0.894
w2	1.000	0.960	0.930	0.840	0.740	0.894
w3	1.000	0.960	0.930	0.840	0.740	0.894
w4	1.000	0.960	0.930	0.840	0.740	0.894
w5	1.000	0.960	0.930	0.840	0.740	0.894
w6	1.000	0.960	0.930	0.840	0.740	0.894
w7	1.000	0.960	0.930	0.840	0.740	0.894
w8	0.000	0.000	0.000	0.000	0.000	0.000
w9	1.000	0.960	0.930	0.840	0.740	0.894
w10	0.940	0.860	0.730	0.690	0.580	0.760
w11	1.000	0.960	0.930	0.840	0.740	0.894
w12	1.000	0.960	0.920	0.820	0.740	0.888
w13	0.870	0.760	0.620	0.550	0.510	0.662
w14	1.000	1.000	0.960	0.860	0.800	0.924
w15	0.980	0.960	0.900	0.790	0.700	0.866
w16	0.210	0.180	0.170	0.130	0.100	0.158
w17	0.930	0.770	0.640	0.520	0.420	0.656
w18	1.000	0.960	0.930	0.840	0.740	0.894
w19	0.860	0.790	0.670	0.640	0.550	0.702

(b) Run 6

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**1.000**	**0.940**	**0.890**	**0.800**	**0.926**
w0	0.660	0.510	0.400	0.290	0.240	0.420
w1	1.000	1.000	0.940	0.890	0.800	0.926
w2	1.000	1.000	0.940	0.890	0.800	0.926
w3	1.000	1.000	0.940	0.890	0.790	0.924
w4	0.720	0.580	0.490	0.410	0.370	0.514
w5	1.000	1.000	0.940	0.890	0.800	0.926
w6	1.000	1.000	0.940	0.890	0.800	0.926
w7	1.000	1.000	0.940	0.890	0.800	0.926
w8	0.000	0.000	0.000	0.000	0.000	0.000
w9	1.000	1.000	0.940	0.890	0.800	0.926
w10	1.000	1.000	0.940	0.890	0.800	0.926
w11	1.000	1.000	0.940	0.890	0.800	0.926
w12	1.000	1.000	0.940	0.890	0.800	0.926
w13	0.870	0.740	0.650	0.560	0.510	0.666
w14	1.000	1.000	0.950	0.890	0.810	0.930
w15	0.790	0.660	0.540	0.500	0.440	0.586
w16	0.790	0.610	0.460	0.330	0.240	0.486
w17	0.940	0.850	0.760	0.590	0.520	0.732
w18	1.000	0.990	0.930	0.880	0.780	0.916
w19	1.000	1.000	0.940	0.890	0.800	0.926

(c) Run 7

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**0.970**	**0.920**	**0.840**	**0.740**	**0.894**
w0	0.140	0.170	0.150	0.140	0.140	0.148
w1	1.000	0.970	0.920	0.840	0.740	0.894
w2	1.000	0.970	0.920	0.840	0.740	0.894
w3	1.000	0.970	0.920	0.840	0.740	0.894
w4	1.000	0.970	0.920	0.840	0.740	0.894
w5	1.000	0.970	0.920	0.840	0.740	0.894
w6	1.000	0.970	0.920	0.840	0.740	0.894
w7	0.000	0.000	0.000	0.000	0.000	0.000
w8	1.000	0.970	0.920	0.840	0.740	0.894
w9	1.000	0.970	0.920	0.840	0.740	0.894
w10	1.000	0.970	0.930	0.850	0.770	0.904
w11	0.710	0.700	0.620	0.530	0.480	0.608
w12	0.830	0.700	0.560	0.470	0.400	0.592
w13	0.880	0.770	0.680	0.550	0.480	0.672
w14	0.000	0.000	0.000	0.000	0.000	0.000
w15	0.940	0.810	0.730	0.640	0.530	0.730
w16	1.000	1.000	0.940	0.800	0.750	0.898
w17	0.720	0.650	0.610	0.540	0.480	0.600
w18	1.000	0.990	0.920	0.840	0.710	0.892
w19	0.560	0.580	0.520	0.440	0.390	0.498

(d) Run 8

Figure C.2: Results of pruning performed on runs 5, 6, 7 and 8

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**0.990**	**0.910**	**0.850**	**0.760**	**0.902**
w0	0.420	0.360	0.370	0.270	0.240	0.332
w1	1.000	0.980	0.950	0.850	0.780	0.912
w2	1.000	0.970	0.950	0.870	0.760	0.910
w3	1.000	0.970	0.940	0.850	0.760	0.904
w4	1.000	1.000	0.910	0.850	0.760	0.904
w5	1.000	1.000	0.930	0.860	0.790	0.916
w6	1.000	1.000	0.920	0.870	0.780	0.914
w7	0.000	0.000	0.000	0.000	0.000	0.000
w8	0.630	0.460	0.410	0.310	0.250	0.412
w9	1.000	0.980	0.940	0.830	0.780	0.906
w10	1.000	1.000	0.910	0.850	0.760	0.904
w11	1.000	1.000	0.930	0.840	0.800	0.914
w12	1.000	0.970	0.960	0.820	0.750	0.900
w13	0.860	0.780	0.670	0.620	0.520	0.690
w14	0.000	0.000	0.000	0.000	0.000	0.000
w15	0.790	0.610	0.490	0.410	0.350	0.530
w16	0.050	0.010	0.040	0.010	0.000	0.022
w17	0.990	0.920	0.860	0.710	0.590	0.814
w18	1.000	1.000	0.930	0.860	0.770	0.912
w19	1.000	1.000	0.910	0.850	0.760	0.904

(a) Run 9

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**0.980**	**0.890**	**0.790**	**0.670**	**0.866**
w0	1.000	0.980	0.890	0.790	0.670	0.866
w1	1.000	0.980	0.890	0.790	0.670	0.866
w2	0.880	0.850	0.720	0.660	0.590	0.740
w3	1.000	0.980	0.890	0.790	0.670	0.866
w4	1.000	0.980	0.890	0.790	0.670	0.866
w5	1.000	0.980	0.890	0.790	0.670	0.866
w6	1.000	0.980	0.890	0.790	0.670	0.866
w7	0.000	0.000	0.000	0.000	0.000	0.000
w8	1.000	0.980	0.890	0.790	0.670	0.866
w9	0.990	0.870	0.750	0.570	0.450	0.726
w10	1.000	0.980	0.900	0.820	0.700	0.880
w11	1.000	0.980	0.890	0.790	0.670	0.866
w12	1.000	0.960	0.890	0.760	0.670	0.856
w13	1.000	0.950	0.860	0.790	0.600	0.840
w14	0.430	0.330	0.280	0.180	0.140	0.272
w15	0.800	0.600	0.490	0.460	0.370	0.544
w16	1.000	0.970	0.880	0.740	0.740	0.866
w17	0.810	0.720	0.620	0.540	0.470	0.632
w18	1.000	0.980	0.890	0.790	0.670	0.866
w19	1.000	0.980	0.900	0.830	0.720	0.886

(b) Run 10

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**0.950**	**0.880**	**0.780**	**0.700**	**0.862**
w0	1.000	0.950	0.880	0.780	0.700	0.862
w1	0.960	0.860	0.780	0.690	0.620	0.782
w2	1.000	0.950	0.880	0.780	0.700	0.862
w3	1.000	0.950	0.880	0.780	0.700	0.862
w4	1.000	0.950	0.880	0.780	0.700	0.862
w5	0.090	0.100	0.070	0.070	0.060	0.078
w6	0.000	0.000	0.000	0.000	0.000	0.000
w7	1.000	0.950	0.880	0.780	0.700	0.862
w8	1.000	0.950	0.880	0.780	0.700	0.862
w9	1.000	0.950	0.880	0.780	0.700	0.862
w10	1.000	0.950	0.880	0.780	0.700	0.862
w11	1.000	0.950	0.860	0.750	0.710	0.854
w12	0.080	0.030	0.060	0.020	0.020	0.042
w13	0.920	0.790	0.590	0.500	0.420	0.644
w14	1.000	0.950	0.880	0.780	0.700	0.862
w15	0.880	0.770	0.680	0.570	0.500	0.680
w16	0.990	0.900	0.820	0.650	0.640	0.800
w17	0.970	0.860	0.780	0.690	0.640	0.788
w18	1.000	0.930	0.850	0.770	0.690	0.848
w19	1.000	0.950	0.880	0.780	0.700	0.862

(c) Run 11

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**0.990**	**0.970**	**0.900**	**0.840**	**0.710**	**0.882**
w0	0.990	0.980	0.900	0.840	0.710	0.884
w1	1.000	0.970	0.900	0.840	0.760	0.894
w2	0.820	0.610	0.500	0.450	0.400	0.556
w3	0.990	0.980	0.900	0.840	0.710	0.884
w4	1.000	0.980	0.900	0.850	0.740	0.894
w5	1.000	0.980	0.900	0.850	0.730	0.892
w6	0.990	0.980	0.900	0.840	0.710	0.884
w7	0.000	0.000	0.000	0.000	0.000	0.000
w8	0.990	0.970	0.900	0.830	0.730	0.884
w9	1.000	0.930	0.820	0.780	0.710	0.848
w10	1.000	0.960	0.910	0.850	0.740	0.892
w11	1.000	0.980	0.880	0.840	0.770	0.894
w12	1.000	0.990	0.900	0.820	0.740	0.890
w13	1.000	0.930	0.860	0.770	0.680	0.848
w14	0.050	0.010	0.010	0.010	0.000	0.016
w15	0.920	0.780	0.660	0.530	0.460	0.670
w16	1.000	0.990	0.930	0.880	0.800	0.920
w17	0.850	0.710	0.610	0.520	0.470	0.632
w18	1.000	0.960	0.900	0.850	0.730	0.888
w19	0.990	0.970	0.870	0.820	0.720	0.874

(d) Run 12

Figure C.3: Results of pruning performed on runs 9, 10, 11 and 12

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**1.000**	**0.920**	**0.850**	**0.740**	**0.902**
w0	1.000	1.000	0.920	0.850	0.740	0.902
w1	0.870	0.740	0.590	0.510	0.440	0.630
w2	1.000	1.000	0.910	0.850	0.740	0.900
w3	1.000	1.000	0.920	0.850	0.740	0.902
w4	1.000	1.000	0.920	0.850	0.740	0.902
w5	1.000	1.000	0.920	0.850	0.740	0.902
w6	1.000	1.000	0.920	0.850	0.740	0.902
w7	1.000	1.000	0.920	0.850	0.740	0.902
w8	0.060	0.090	0.070	0.060	0.070	0.070
w9	1.000	1.000	0.920	0.850	0.740	0.902
w10	1.000	1.000	0.920	0.850	0.740	0.902
w11	1.000	1.000	0.910	0.840	0.740	0.898
w12	1.000	1.000	0.920	0.850	0.740	0.902
w13	1.000	1.000	0.920	0.850	0.740	0.902
w14	1.000	1.000	0.940	0.870	0.790	0.920
w15	0.850	0.750	0.590	0.520	0.450	0.632
w16	0.480	0.340	0.210	0.180	0.140	0.270
w17	0.850	0.630	0.550	0.460	0.400	0.578
w18	1.000	0.980	0.930	0.850	0.700	0.892
w19	1.000	1.000	0.920	0.850	0.740	0.902

(a) Run 13

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**0.960**	**0.900**	**0.830**	**0.740**	**0.600**	**0.806**
w0	0.960	0.880	0.830	0.750	0.610	0.806
w1	0.960	0.890	0.860	0.730	0.620	0.812
w2	0.380	0.260	0.210	0.160	0.120	0.226
w3	0.960	0.900	0.830	0.740	0.600	0.806
w4	0.730	0.670	0.540	0.430	0.270	0.528
w5	0.970	0.900	0.830	0.740	0.590	0.806
w6	0.000	0.000	0.000	0.000	0.000	0.000
w7	1.000	0.930	0.860	0.750	0.690	0.846
w8	0.960	0.900	0.830	0.740	0.600	0.806
w9	0.970	0.900	0.840	0.730	0.600	0.808
w10	0.960	0.900	0.830	0.740	0.600	0.806
w11	0.960	0.900	0.850	0.740	0.630	0.816
w12	0.100	0.040	0.040	0.030	0.010	0.044
w13	0.950	0.860	0.810	0.680	0.540	0.768
w14	0.950	0.890	0.750	0.640	0.530	0.752
w15	0.960	0.910	0.770	0.700	0.640	0.796
w16	0.960	0.900	0.830	0.740	0.600	0.806
w17	0.770	0.580	0.410	0.360	0.320	0.488
w18	0.950	0.880	0.820	0.740	0.580	0.794
w19	0.990	0.900	0.810	0.750	0.640	0.818

(b) Run 14

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**0.980**	**0.920**	**0.790**	**0.760**	**0.890**
w0	0.740	0.650	0.530	0.430	0.340	0.538
w1	1.000	0.990	0.920	0.770	0.720	0.880
w2	1.000	0.970	0.890	0.780	0.760	0.880
w3	1.000	0.970	0.920	0.810	0.730	0.886
w4	1.000	0.960	0.900	0.810	0.740	0.882
w5	1.000	0.970	0.900	0.770	0.760	0.880
w6	1.000	0.970	0.910	0.790	0.740	0.882
w7	0.800	0.580	0.450	0.370	0.300	0.500
w8	0.000	0.000	0.000	0.000	0.020	0.004
w9	1.000	0.960	0.900	0.780	0.760	0.880
w10	1.000	0.980	0.920	0.790	0.760	0.890
w11	1.000	0.970	0.910	0.810	0.740	0.886
w12	1.000	1.000	0.960	0.890	0.800	0.930
w13	0.870	0.730	0.630	0.540	0.450	0.644
w14	0.980	0.890	0.830	0.680	0.600	0.796
w15	0.920	0.810	0.660	0.550	0.430	0.674
w16	0.630	0.450	0.440	0.240	0.220	0.396
w17	1.000	0.980	0.920	0.790	0.760	0.890
w18	1.000	0.960	0.900	0.810	0.720	0.878
w19	1.000	0.970	0.890	0.800	0.730	0.878

(c) Run 15

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**0.970**	**0.910**	**0.810**	**0.700**	**0.878**
w0	1.000	0.970	0.910	0.810	0.700	0.878
w1	0.800	0.650	0.540	0.450	0.440	0.576
w2	1.000	0.970	0.910	0.810	0.700	0.878
w3	1.000	0.970	0.910	0.810	0.700	0.878
w4	1.000	0.970	0.910	0.810	0.700	0.878
w5	1.000	0.970	0.910	0.810	0.700	0.878
w6	1.000	0.970	0.910	0.810	0.700	0.878
w7	1.000	0.970	0.910	0.810	0.700	0.878
w8	0.000	0.000	0.000	0.000	0.000	0.000
w9	0.720	0.630	0.550	0.470	0.390	0.552
w10	1.000	0.970	0.910	0.810	0.700	0.878
w11	1.000	0.970	0.910	0.830	0.700	0.882
w12	0.660	0.500	0.470	0.400	0.380	0.482
w13	1.000	0.970	0.910	0.760	0.730	0.874
w14	1.000	0.960	0.940	0.830	0.700	0.886
w15	0.840	0.700	0.560	0.500	0.450	0.610
w16	0.000	0.000	0.000	0.000	0.000	0.000
w17	0.910	0.730	0.610	0.500	0.420	0.634
w18	0.770	0.660	0.600	0.530	0.470	0.606
w19	0.950	0.880	0.780	0.720	0.620	0.790

(d) Run 16

Figure C.4: Results of pruning performed on runs 13, 14, 15 and 16

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**1.000**	**0.980**	**0.850**	**0.780**	**0.922**
w0	1.000	1.000	0.990	0.870	0.780	0.928
w1	1.000	1.000	0.980	0.850	0.780	0.922
w2	0.870	0.770	0.710	0.640	0.540	0.706
w3	1.000	1.000	0.980	0.840	0.800	0.924
w4	1.000	1.000	0.980	0.850	0.780	0.922
w5	1.000	1.000	0.980	0.850	0.780	0.922
w6	0.000	0.000	0.000	0.000	0.000	0.000
w7	0.980	0.940	0.890	0.830	0.720	0.872
w8	1.000	1.000	0.980	0.850	0.780	0.922
w9	1.000	1.000	0.980	0.860	0.790	0.926
w10	1.000	1.000	0.980	0.850	0.780	0.922
w11	1.000	1.000	0.970	0.870	0.760	0.920
w12	0.140	0.090	0.050	0.050	0.020	0.070
w13	1.000	0.950	0.870	0.750	0.600	0.834
w14	1.000	1.000	0.970	0.850	0.770	0.918
w15	1.000	0.990	0.960	0.830	0.790	0.914
w16	1.000	1.000	0.980	0.850	0.780	0.922
w17	0.890	0.800	0.730	0.640	0.630	0.738
w18	1.000	1.000	0.980	0.850	0.770	0.920
w19	1.000	1.000	0.980	0.850	0.780	0.922

(a) Run 17

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**1.000**	**0.940**	**0.830**	**0.750**	**0.904**
w0	1.000	1.000	0.940	0.830	0.740	0.902
w1	1.000	0.990	0.950	0.840	0.790	0.914
w2	0.670	0.570	0.460	0.380	0.310	0.478
w3	1.000	1.000	0.940	0.820	0.750	0.902
w4	1.000	1.000	0.940	0.820	0.750	0.902
w5	1.000	0.990	0.940	0.780	0.750	0.892
w6	0.000	0.030	0.020	0.020	0.010	0.016
w7	0.000	0.000	0.000	0.000	0.000	0.000
w8	1.000	0.990	0.940	0.810	0.750	0.898
w9	1.000	0.960	0.940	0.820	0.770	0.898
w10	1.000	0.990	0.940	0.830	0.730	0.898
w11	1.000	1.000	0.940	0.860	0.750	0.910
w12	1.000	0.920	0.860	0.740	0.620	0.828
w13	0.960	0.850	0.610	0.480	0.380	0.656
w14	0.970	0.880	0.770	0.670	0.580	0.774
w15	0.690	0.590	0.470	0.360	0.310	0.484
w16	1.000	1.000	0.940	0.860	0.770	0.914
w17	0.890	0.750	0.680	0.590	0.470	0.676
w18	1.000	0.990	0.920	0.810	0.760	0.896
w19	1.000	1.000	0.940	0.830	0.750	0.904

(b) Run 18

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**0.920**	**0.780**	**0.680**	**0.580**	**0.430**	**0.678**
w0	0.920	0.780	0.680	0.580	0.430	0.678
w1	0.920	0.790	0.670	0.600	0.420	0.680
w2	0.920	0.780	0.680	0.580	0.430	0.678
w3	0.920	0.780	0.680	0.580	0.430	0.678
w4	0.920	0.790	0.690	0.580	0.440	0.684
w5	0.920	0.780	0.680	0.580	0.430	0.678
w6	0.920	0.780	0.680	0.580	0.430	0.678
w7	0.810	0.680	0.540	0.450	0.370	0.570
w8	0.000	0.000	0.000	0.000	0.000	0.000
w9	0.920	0.840	0.720	0.540	0.450	0.694
w10	0.920	0.730	0.610	0.490	0.430	0.636
w11	0.900	0.790	0.680	0.600	0.430	0.680
w12	0.910	0.840	0.720	0.550	0.410	0.686
w13	0.920	0.780	0.680	0.580	0.430	0.678
w14	0.920	0.780	0.680	0.580	0.430	0.678
w15	0.970	0.880	0.760	0.630	0.530	0.754
w16	0.070	0.050	0.060	0.040	0.030	0.050
w17	0.820	0.670	0.510	0.450	0.350	0.560
w18	0.960	0.860	0.750	0.530	0.420	0.704
w19	0.720	0.590	0.580	0.440	0.400	0.546

(c) Run 19

	ub = 0.2	ub = 0.3	ub = 0.4	ub = 0.5	ub = 0.6	Average
Ref	**1.000**	**1.000**	**0.950**	**0.840**	**0.710**	**0.900**
w0	0.910	0.790	0.700	0.570	0.530	0.700
w1	1.000	1.000	0.950	0.840	0.710	0.900
w2	1.000	1.000	0.950	0.840	0.710	0.900
w3	1.000	1.000	0.950	0.840	0.710	0.900
w4	1.000	1.000	0.950	0.840	0.710	0.900
w5	1.000	1.000	0.950	0.840	0.720	0.902
w6	1.000	1.000	0.950	0.840	0.710	0.900
w7	1.000	1.000	0.950	0.840	0.710	0.900
w8	0.230	0.150	0.080	0.050	0.040	0.110
w9	1.000	1.000	0.950	0.840	0.710	0.900
w10	1.000	0.990	0.880	0.770	0.680	0.864
w11	1.000	1.000	0.950	0.840	0.710	0.900
w12	1.000	1.000	0.950	0.840	0.710	0.900
w13	0.910	0.780	0.730	0.650	0.530	0.720
w14	1.000	0.990	0.850	0.720	0.670	0.846
w15	1.000	1.000	0.950	0.850	0.720	0.904
w16	0.330	0.250	0.160	0.120	0.100	0.192
w17	0.660	0.540	0.390	0.320	0.270	0.436
w18	1.000	0.990	0.950	0.870	0.750	0.912
w19	1.000	0.970	0.940	0.830	0.680	0.884

(d) Run 20

Figure C.5: Results of pruning performed on runs 17, 18, 19 and 20

Appendix D

Results of the scaler application to the standalone Genetic Programming (GP) guidance scheme

The results presented in this Appendix were obtained by performing the GP evolutionary process as described at the end of Subsection 5.3.6. They are meant to show the effect of the scaler application to a standalone GP guidance scheme resulting in the framework in Figure D.1. The scaler is the same as described in Subsection 5.2.1 with the hyperparameters set as in Subsection 5.3.6. The complete results are shown in Table D.1 while a depiction of the performed trajectories is shown in Figure D.2

Appendix D. Results of the scaler application to the standalone Genetic Programming (GP) guidance scheme

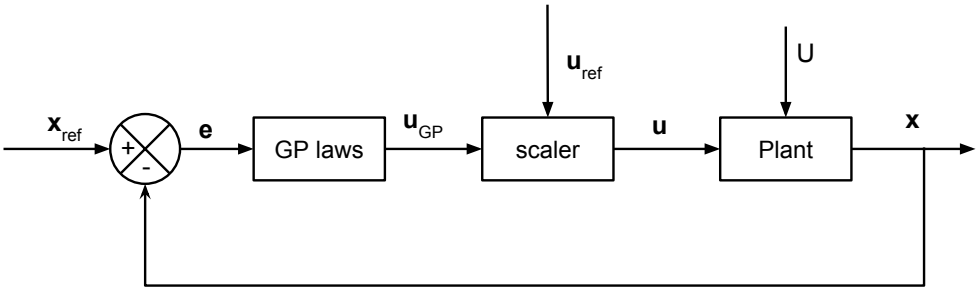
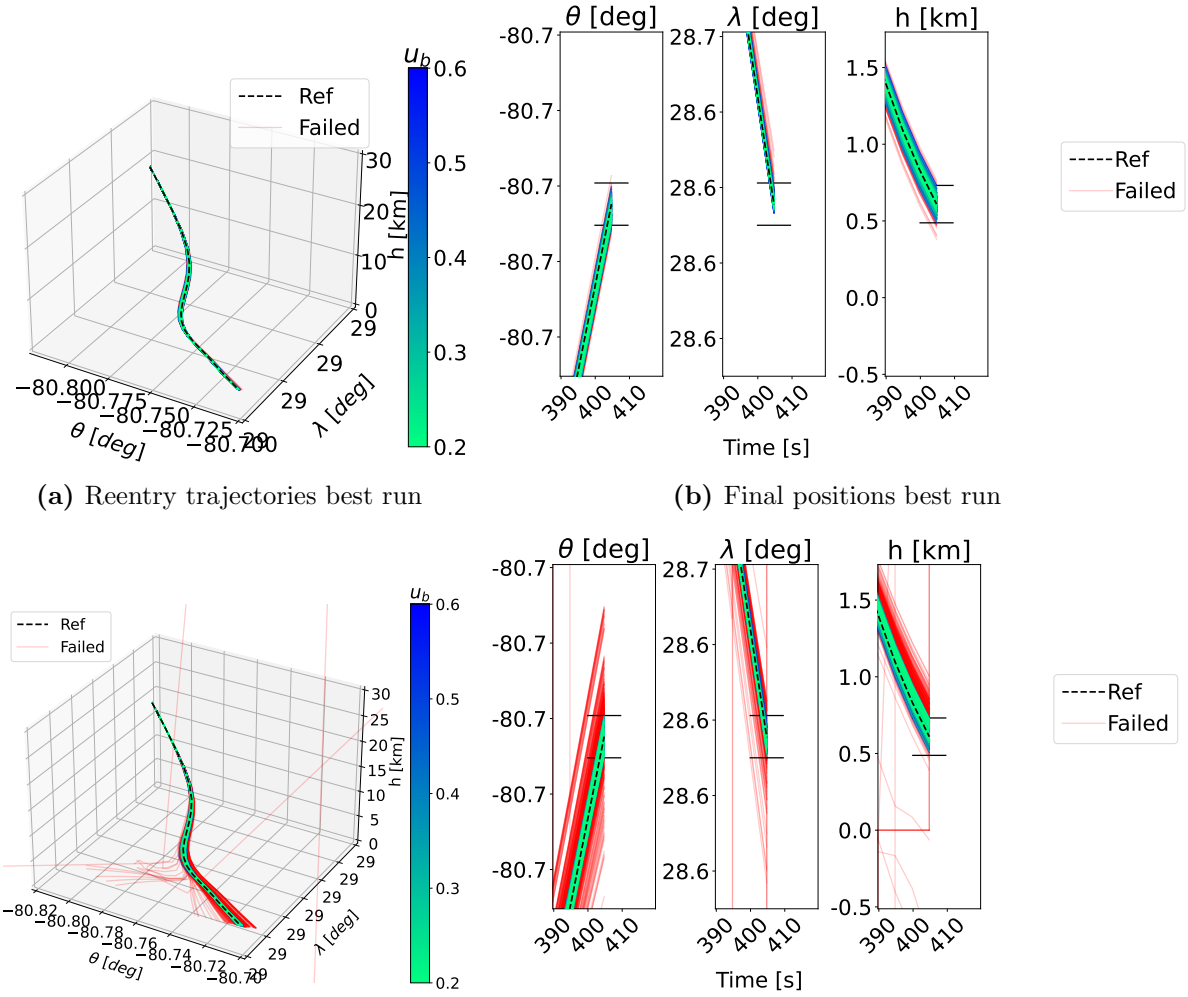


Figure D.1: Control scheme of the standalone GP controller with the scaler

Table D.1: Results of the statistical analysis performed using the standalone GP with the scaler

Run ID	Test success ratio				
	$u_b = 0.2$	$u_b = 0.3$	$u_b = 0.4$	$u_b = 0.5$	$u_b = 0.6$
1	83%	59%	40%	31%	27%
2	88%	71%	55%	38%	30%
3	89%	76%	60%	50%	41%
4	100%	90%	81%	70%	65%
5	71%	45%	26%	14%	13%
6	100%	95%	91%	80%	74%
7	70%	47%	34%	25%	17%
8	97%	72%	56%	46%	39%
9	100%	100%	98%	88%	79%
10	82%	63%	40%	24%	16%
11	90%	82%	69%	57%	53%
12	94%	69%	53%	34%	28%
13	94%	83%	71%	62%	50%
14	91%	84%	65%	54%	43%
15	100%	99%	89%	81%	71%
16	92%	80%	74%	65%	52%
17	98%	88%	74%	62%	52%
18	98%	92%	74%	62%	52%
19	91%	88%	77%	66%	54%
20	53%	35%	22%	18%	14%
Median	91.5%	81%	67%	55.5%	46.5%
Standard Deviation	12.24%	18.36%	21.78%	22.19%	20.47%



(a) Reentry trajectories best run

(b) Final positions best run

(c) Reentry trajectories worst run

(d) Final positions worst run

Figure D.2: Reentry trajectories and final positions of the best (run 9) and worst (run 20) runs performed with the GP+scaler scheme. The red lines represent the failed trajectories. The black horizontal bars in the final position plots represent the Final Approach Corridor (FAC) box boundaries. [2]

Bibliography

[1] C. Wilson, F. Marchetti, M. Di Carlo, A. Riccardi, and E. Minisci, "Classifying intelligence in machines: A taxonomy of intelligent control," *Robotics*, vol. 9, no. 3, p. 64, 2020.

[2] F. Marchetti and E. Minisci, "Genetically Adapted Neural Network-Based Intelligent Controller for Reentry Vehicle Real-Time Guidance," *[SUBMITTED TO] Soft Computing*, 2024.

[3] F. Marchetti, E. Minisci, and A. Riccardi, "Towards Intelligent Control via Genetic Programming," *Proceedings of the International Joint Conference on Neural Networks*, 2020.

[4] F. Marchetti and E. Minisci, "Genetic programming guidance control system for a reentry vehicle under uncertainties," *Mathematics*, vol. 9, no. 16, pp. 1–17, 2021.

[5] S. D'Angelo, E. Minisci, D. Di Bona, and L. Guerra, "Optimization Methodology for Ascent Trajectories of Lifting-Body Reusable Launchers," *Journal of Spacecraft and Rockets*, vol. 37, no. 6, 2000.

[6] A. Russo and G. Lax, "Using Artificial Intelligence for Space Challenges: A Survey," *Applied Sciences*, vol. 12, p. 5106, may 2022.

[7] J.-G. Meß, F. Dannemann, and F. Greif, "Techniques of Artificial Intelligence for Space Applications-A Survey," tech. rep., 2019.

[8] D. Izzo, M. Märtens, and B. Pan, "A survey on artificial intelligence trends in spacecraft guidance dynamics and control," *Astrodynamics*, vol. 3, no. 4, pp. 287–299, 2019.

[9] K. Fu, "Learning control systems and intelligent control systems: An intersection of artifical intelligence and automatic control," *IEEE Transactions on Automatic Control*, vol. 16, no. 1, pp. 70–72, 1971.

[10] S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using iterative repair to improve the responsiveness of planning and scheduling," *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*, pp. 300–307, 2000.

[11] K. Cohen, "Grand challenges in intelligent aerospace systems," *Frontiers in Aerospace Engineering*, vol. 2, sep 2023.

[12] J. Koza, M. Keane, J. Yu, F. Bennett III, and W. Mydlowec, "Automatic Creation of Human-Competitive Programs and Controllers by Means of Genetic Programming," *Genetic Programming and Evolvable Machines*, vol. 1, no. 1/2, pp. 121–164, 2000.

[13] M. M. A. Ali, A. Jamali, A. Asgharnia, R. Ansari, and R. Mallipeddi, "Multi-objective Lyapunov-based controller design for nonlinear systems via genetic programming," *Neural Computing and Applications*, vol. 34, no. 2, pp. 1345–1357, 2022.

[14] A. Riccardi, E. Minisci, M. Di Carlo, C. Wilson, and F. Marchetti, "Assessment of Intelligent Control Techniques for Space Applications," tech. rep., 2018.

[15] A. Riccardi, E. Minisci, M. D. Carlo, C. Wilson, and F. Marchetti, "Assessment of Intelligent Control Techniques for Space Applications - Gap Analysis," tech. rep., 2018.

[16] X. Wang, F. Zhang, D. Hu, R. Chen, and Z. Song, "Review, Prospect and Technical Challenge of Launch Vehicle," pp. 1–31, 2023.

[17] P. Castaldi, S. Emami, S. Mallipeddi, and S. Simani, "Resilient Composite Learning Neuro-Adaptive Integrated Guidance and Control for reusable Launch Vehicle," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 6030–6036, 2023.

[18] F. Marchetti, E. Minisci, and A. Riccardi, "Single-stage to orbit ascent trajectory optimisation with reliable evolutionary initial guess," *Optimization and Engineering*, no. 0123456789, 2021.

[19] C. Wilson, F. Marchetti, M. Di Carlo, A. Riccardi, and E. Minisci, "Intelligent Control: A Taxonomy," in *2019 8th International Conference on Systems and Control (ICSC'2019)*, 2019.

[20] F. Marchetti and E. Minisci, "A Hybrid Neural Network-Genetic Programming Intelligent Control Approach," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (B. Filipič, E. Minisci, and M. Vasile, eds.), vol. 12438 LNCS, (Brussels), pp. 240–254, Springer, Cham, 2020.

[21] F. Marchetti and E. Minisci, "Inclusive Genetic Programming," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (T. Hu, N. Lourenço, and E. Medvet, eds.), vol. 12691 LNCS, (Cham), pp. 51–65, Springer International Publishing, 2021.

[22] F. Marchetti, E. Minisci, and A. Riccardi, "Trajectory Optimization of a Reusable Launch Vehicle," in *17th EUROPT Workshop on Advances in Continuous Optimization*, (Glasgow), 2019.

[23] F. Marchetti, E. Minisci, and A. Riccardi, "Towards Intelligent Control via Genetic Programming," in *5th workshop on Optimisation in Space Engineering (OSE)*, (Ljubljana), 2019.

[24] F. Marchetti, G. Pietropolli, F. J. Camerota Verdù, M. Castelli, and E. Minisci, "Control Law Automatic Design Through Parametrized Genetic Programming with Adjoint State Method Gradient Evaluation," *[SUBMITTED TO] Applied Soft Computing*, 2023.

[25] F. Marchetti, C. Wilson, C. Powell, E. Minisci, and A. Riccardi, "Convolutional Generative Adversarial Network, via Transfer Learning, for Traditional Scottish Music Generation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12693 LNCS, pp. 187–202, 2021.

[26] F. Marchetti, M. Castelli, I. Bakurov, and L. Vanneschi, "Full Inclusive Genetic Programming," in *[SUBMITTED TO] 2024 IEEE Congress on Evolutionary Computation (CEC)*, 2024.

[27] F. Marchetti and E. Minisci, "Adjoint state method for the trajectory optimization of a reentry vehicle," in *19th EUROPT Workshop on Advances in Continuous Optimization*, 2022.

[28] A. Zolghadri, "Modern Control Theory and Real-World Aerospace Applications: I love You, Nor do I ?," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6446–6451, 2017.

[29] D. Haibin and L. Pei, "Progress in control approaches for hypersonic vehicle," *Science China Technological Sciences*, vol. 55, no. 10, pp. 2965–2970, 2012.

[30] S. Li and X. Jiang, "Review and prospect of guidance and control for Mars atmospheric entry," 2014.

[31] Z. Sun, *Guidance, Navigation and Control Technology*, vol. 46. 2021.

[32] H. Nyquist, "Regeneration Theory," *Bell System Technical Journal*, vol. 11, pp. 126–147, jan 1932.

[33] J. G. Ziegler and N. B. Nichols, "Process Lags in Automatic Control Circuits," 1943.

[34] R. Bellman, "The Theory of Dynamic Programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503–515, 1954.

[35] K. J. Astrom and B. Wittenmark, *Adaptive Control*. 2nd ed., 2008.

[36] F. L. Lewis, "Optimal control," *The Control Systems Handbook: Control System Advanced Methods, Second Edition*, vol. 124, pp. 577–612, 2010.

[37] J. M. Longuski, J. J. Guzmàn, and J. E. Prussing, *Optimal Control with Aerospace Applications*. Springer, New York, NY, 2010.

[38] W. Levine, "Optimal control theory: An introduction," 1972.

[39] J. T. Betts, *Practical Methods for Optimal Control and Estimation using Nonlinear Programming*. SIAM, second ed., 2010.

[40] D. G. Hull, *Optimal Control Theory for Applications*, vol. 28 of *Mechanical Engineering Series*. New York, NY: Springer New York, 2003.

[41] S. L. Brunton and J. N. Kutz, *Data-Drive Science and Engineering: Machine Learning, Dynamical Systems and Control*, vol. 53. 2019.

[42] Z. Kemin, J. C. Doyle, and K. Glover, *Robust and optimal control*. Prentice Hall, 1996.

[43] Q. Lu and J. Zhou, "LQR tracking guidance law for hypersonic vehicle," in *2017 29th Chinese Control And Decision Conference (CCDC)*, pp. 7090–7094, IEEE, may 2017.

[44] Q. Li, M. Zhang, D. Chang, W. Zhou, H. Li, Z. Li, C. Shao, and C. Heng, "Research on control of solid rocket electric actuator based on LQR," *2021 IEEE International Conference on Artificial Intelligence and Computer Applications, ICAICA 2021*, pp. 569–573, 2021.

[45] X. Wang, Y. Li, and J. Zhang, "A Novel IGC Scheme for RHV with the Capabilities of Online Aerodynamic Coefficient Estimation and Trajectory Generation," *Mathematics*, vol. 9, p. 172, jan 2021.

[46] Cishen Zhang and Minyue Fu, "A revisit to the gain and phase margins of linear quadratic regulators," *IEEE Transactions on Automatic Control*, vol. 41, no. 10, pp. 1527–1530, 1996.

[47] J. Doyle, "Guaranteed margins for LQG regulators," *IEEE Transactions on Automatic Control*, vol. 23, pp. 756–757, aug 1978.

[48] R. Bonalli, B. Herisse, and E. Trelat, "Optimal control of endoatmospheric launch vehicle systems: Geometric and computational issues," *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2418–2433, 2020.

[49] A. Banerjee and M. Nabi, "Re-entry trajectory optimization for space shuttle using Sine-Cosine Algorithm," *Proceedings of 8th International Conference on Recent Advances in Space Technologies, RAST 2017*, pp. 73–77, 2017.

[50] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, New York, NY, second ed., 2006.

[51] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[52] I. M. Ross, C. D'Souza, F. Fahroo, and J. B. Ross, "A fast approach to multi-stage launch vehicle trajectory optimization," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, no. August, pp. 1–9, 2003.

[53] M. Costa and E. Minisci, "MOPED: A multi-objective Parzen-based estimation of distribution algorithm for continuous problems," in *EMO 2003: Evolutionary Multi-Criterion Optimization*, vol. 2632, pp. 282–294, Springer, Berlin, Heidelberg, 2003.

[54] M. Di Carlo, M. Vasile, and E. Minisci, "Adaptive multi-population inflationary differential evolution," *Soft Computing*, vol. 24, no. 5, pp. 3861–3891, 2019.

[55] G. N. Kumar, D. Penchalaiah, A. K. Sarkar, and S. E. Talole, "Hypersonic Boost Glide Vehicle Trajectory Optimization Using Genetic Algorithm," *IFAC-PapersOnLine*, vol. 51, no. 1, pp. 118–123, 2018.

[56] G. N. Kumar, A. K. Sarkar, M. S. Ahmed, and S. E. Talole, "Reentry Trajectory Optimization using Gradient Free Algorithms," *IFAC-PapersOnLine*, vol. 51, no. 1, pp. 650–655, 2018.

[57] L. A. Ricciardi, C. A. Maddock, and M. Vasile, "Direct Solution of Multi-Objective Optimal Control Problems Applied to Spaceplane Mission Design," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 1, pp. 30–46, 2019.

[58] F. Toso and C. A. Maddock, "Launch abort trajectory optimisation for reusable launch vehicles," *21st AIAA International Space Planes and Hypersonics Technologies Conference, Hypersonics 2017*, no. March, pp. 1–11, 2017.

[59] A. Zheng and M. Morari, "Stability of model predictive control with mixed constraints," *IEEE Transactions on Automatic Control*, vol. 40, no. 10, pp. 1818–1823, 1995.

[60] E. F. Camacho and C. Bordons, *Model predictive control*. London: Springer London, 2007.

[61] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: an engineering perspective," *International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5-6, pp. 1327–1349, 2021.

[62] J. Chai, E. Medagoda, and E. Kayacan, "Adaptive and efficient model predictive control for booster reentry," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 12, pp. 2372–2382, 2020.

[63] J. Guadagnini, M. Lavagna, and P. Rosa, "Model predictive control for reusable space launcher guidance improvement," *Acta Astronautica*, vol. 193, no. February 2021, pp. 767–778, 2022.

[64] U. Eren, A. Prach, B. B. Koçer, S. V. Rakovic, E. Kayacan, and B. Açikmese, "Model predictive control in aerospace systems: Current state and opportunities," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 7, pp. 1541–1566, 2017.

[65] X. Liu, P. Lu, and B. Pan, "Survey of convex optimization for aerospace applications," *Astrodynamics*, vol. 1, no. 1, pp. 23–40, 2017.

[66] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, mar 2004.

[67] B. Acikmese, J. M. Carson, and L. Blackmore, "Lossless convexification of nonconvex control bound and pointing constraints of the soft landing optimal control problem," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2104–2113, 2013.

[68] J. Wang, N. Cui, and C. Wei, "Optimal Rocket Landing Guidance Using Convex Optimization and Model Predictive Control," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 5, pp. 1078–1092, 2019.

[69] M. Sagliano and E. Mooij, "Optimal drag-energy entry guidance via pseudospectral convex optimization," *Aerospace Science and Technology*, vol. 117, p. 106946, 2021.

[70] M. Szmuk, T. P. Reynolds, B. Açıkmeşe, M. Mesbahi, and J. M. Carson, "Successive convexification for 6-dof powered descent guidance with compound state-triggered constraints," *AIAA Scitech 2019 Forum*, no. January, pp. 1–16, 2019.

[71] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. Philip Chen, "Review of advanced guidance and control algorithms for space/aerospace vehicles," *Progress in Aerospace Sciences*, vol. 122, p. 100696, apr 2021.

[72] J. Wang, H. Li, and H. Chen, "An Iterative Convex Programming Method for Rocket Landing Trajectory Optimization," *Journal of the Astronautical Sciences*, vol. 67, no. 4, pp. 1553–1574, 2020.

[73] G. Zames, "Feedback and Optimal Sensitivity: Model Reference Transformations, Multiplicative Seminorms, and Approximate Inverses," *IEEE Transactions on Automatic Control*, vol. 26, no. 2, pp. 301–320, 1981.

[74] Y.-c. Xie, H. Huang, Y. Hu, and G.-q. Zhang, "Applications of advanced control methods in spacecrafts: progress, challenges, and future prospects," *Frontiers of Information Technology and Electronic Engineering*, vol. 17, no. 9, pp. 841–861, 2016.

[75] G. E. Dullerud and F. G. Paganini, *A Course in Robust Control Theory: A Convex Approach*, vol. 46. 2000.

[76] B. M. Chen, *Robust and H Control*. Communications and Control Engineering, London: Springer London, 2000.

[77] D. Navarro-Tapia, A. Marcos, S. Bennani, and C. Roux, "Structured H-infinity control based on classical control parameters for the VEGA launch vehicle," *2016 IEEE Conference on Control Applications, CCA 2016*, no. February 2013, pp. 33–38, 2016.

[78] D. Navarro-Tapia, A. Marcos, S. Bennani, and C. Roux, "Structured H-infinity Control Design for the VEGA Launch Vehicle: Recovery of the Legacy Control Behaviour," in *Gns*, vol. 23, pp. 1–7, 2017.

[79] M. Sagliano, T. Tsukamoto, A. Heidecker, J. A. Maces Hernandez, S. Fari, M. Schlotterer, S. Woicke, D. Seelbinder, S. Ishimoto, and E. Dumont, "Robust Control for Reusable Rockets via Structured H-infinity Synthesis," in *11th International ESA Conference on Guidance, Navigation & Control Systems*, jun 2021.

[80] G. Rigatos, P. Wira, M. Abbaszadeh, K. Busawon, and L. Dala, "Nonlinear optimal control for autonomous hypersonic vehicles," *Aerospace Systems*, vol. 2, no. 2, pp. 197–213, 2019.

[81] Y. Shtessel, C. Edwards, and L. Fridman, *Sliding Mode Control and Observation, Series: Control Engineering*, vol. 10. 2016.

[82] C. Edwards and S. K. Spurgeon, *Sliding Mode Control - Theory and Applications*. Taylor & Francis, 1998.

[83] V. I. Utkin, *Sliding Modes in Control and Optimization*. 1992.

[84] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. 1991.

[85] L. Zhang, C. Wei, R. Wu, and N. Cui, "Fixed-time extended state observer based non-singular fast terminal sliding mode control for a VTVL reusable launch vehicle," *Aerospace Science and Technology*, vol. 82-83, pp. 70–79, 2018.

[86] Z. Guo, J. Chang, J. Guo, and J. Zhou, "Adaptive twisting sliding mode algorithm for hypersonic reentry vehicle attitude control based on finite-time observer," *ISA Transactions*, vol. 77, pp. 20–29, 2018.

[87] X. Liao and L. Chen, "Integral sliding mode control based approach and landing guidance law with finite-time convergence," *Optik*, vol. 127, no. 20, pp. 8215–8230, 2016.

[88] D. Cho, H. J. Kim, and M. J. Tahk, "Nonsingular sliding mode guidance for impact time control," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 1, pp. 61–68, 2016.

[89] I. D. Landau, R. Lozano, M. M'Saad, and A. Karimi, *Adaptive Control: Algorithms, Analysis and Applications*. Springer London, 2nd ed., 2011.

[90] M. Benosman, *Learning-Based Adaptive Control*. Elsevier, 2016.

[91] B. D. Anderson and A. Dehghani, "Challenges of adaptive control-past, permanent and future," *Annual Reviews in Control*, vol. 32, no. 2, pp. 123–135, 2008.

[92] D. Rotondo, "Advances in Gain-Scheduling and Fault Tolerant Control Techniques," *Universitat Politecnica de Catalunya*, vol. 49, p. 6221, 2016.

[93] M. Kerr, A. Marcos, L. Peñin, and E. Bornschlegl, "Gain Scheduled FDI for a Re-Entry Vehicle," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, (Reston, Virigina), American Institute of Aeronautics and Astronautics, aug 2008.

[94] S. McNamara, C. Restrepo, E. Medina, R. Whitley, R. Proud, and J. Madsen, "Gain Scheduling for the Orion Launch Abort Vehicle Controller," in *AIAA Guidance, Navigation, and Control Conference*, (Reston, Virigina), American Institute of Aeronautics and Astronautics, aug 2011.

[95] A. S. Hameed and G. R. Bindu, "Gain Scheduled Finite Horizon LQR for Approach and Landing Phase of a Reusable Launch Vehicle," *Journal of The Institution of Engineers (India): Series C*, 2022.

[96] A. De Oliveira and M. Lavagna, "Robust Control Design via Structured H-infinity for the Atmospheric Re-entry of Reusable Launchers," in *Papers of ESA GNC-ICATT 2023*, ESA, jul 2023.

[97] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Prentice Hall, Inc, 1996.

[98] E. N. Johnson and A. J. Calise, "Limited Authority Adaptive Flight Control for Reusable Launch Vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 6, pp. 906–913, 2003.

[99] J. S. Orr and T. S. VanZwieten, "Robust, practical adaptive control for launch vehicles," *AIAA Guidance, Navigation, and Control Conference 2012*, no. August, pp. 1–20, 2012.

[100] G. Joshi and G. Chowdhary, "Deep Model Reference Adaptive Control," *Proceedings of the IEEE Conference on Decision and Control*, vol. 2019-Decem, no. CoRL, pp. 4601–4608, 2019.

[101] A. P. Nair, N. Selvaganesan, and V. R. Lalithambika, "Lyapunov based PD/PID in model reference adaptive control for satellite launch vehicle systems," *Aerospace Science and Technology*, vol. 51, pp. 70–77, 2016.

[102] J. Guo, T. Zhang, C. Cheng, and J. Zhou, "Model reference adaptive attitude control for near space hypersonic vehicle with mismatched uncertainties," *Transactions of the Institute of Measurement and Control*, vol. 41, no. 5, pp. 1301–1312, 2019.

[103] E. Mooij, "Simple Adaptive Control System Design Trades," in *AIAA Guidance, Navigation, and Control Conference*, (Reston, Virginia), American Institute of Aeronautics and Astronautics, jan 2017.

[104] A. Shekhar and A. Sharma, "Review of Model Reference Adaptive Control," *2018 International Conference on Information, Communication, Engineering and Technology, ICICET 2018*, pp. 1–5, 2018.

[105] Z. Wang, P. Cai, Z. Gong, C. Zhang, S. Zhao, J. Wu, and P. Dong, "Review on guidance and control of aerospace vehicles: recent progress and prospect," *Aerospace Systems*, feb 2024.

[106] E. Johnson, A. Calise, and J. E. Corban, "Reusable launch vehicle adaptive guidance and control using neural networks," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2001.

[107] J. Zhu, L. Liu, G. Tang, and W. Bao, "Robust adaptive gliding guidance for hypersonic vehicles," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 232, no. 7, pp. 1272–1282, 2018.

[108] E. Mooij, "Simple Adaptive Re-entry Guidance for Path-Constraint Tracking," *2018 AIAA Guidance, Navigation, and Control Conference*, no. January, 2018.

[109] H. Yan and Y. He, "Adaptive Integrated Guidance and Control Based on Backstepping for the Landing of Reusable Launch Vehicles.," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 496–501, 2015.

[110] X. Yan, P. Wang, S. Xu, S. Wang, and H. Jiang, "Adaptive Entry Guidance for Hypersonic Gliding Vehicles Using Analytic Feedback Control," *International Journal of Aerospace Engineering*, vol. 2020, 2020.

[111] Z. Song, C. Wang, and Y. He, "Autonomous Guidance Control for Ascent Flight," pp. 33–74, 2023.

[112] K. M. Passino and S. Yurkovich, *Fuzzy Control*, vol. 42. Menlo Park, CA: Addison-wesley, 1998.

[113] Q. Mao, L. Dou, Q. Zong, and Z. Ding, "Attitude controller design for reusable launch vehicles during reentry phase via compound adaptive fuzzy H-infinity control," *Aerospace Science and Technology*, vol. 72, pp. 36–48, 2018.

[114] K. J. Hunt, D. Sbarbaro, R. Zbikowski, and P. J. Gawthrop, "Neural networks for control systems-A survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.

[115] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," 2016.

[116] G. W. Irwin, K. Warwick, and K. J. Hunt, *Neural Network Applications in Control*. 1995.

[117] B. Xu, "Robust adaptive neural control of flexible hypersonic flight vehicle with dead-zone input nonlinearity," *Nonlinear Dynamics*, vol. 80, no. 3, pp. 1509–1520, 2015.

[118] S. Mathavaraj and R. Padhi, "Optimally allocated nonlinear robust control of a reusable launch vehicle during re-entry," *Unmanned Systems*, vol. 8, no. 1, pp. 33–48, 2020.

[119] P. J. Fleming and R. C. Purshouse, "Evolutionary algorithms in control systems engineering: a survey," *Control Engineering Practice*, vol. 10, no. 11, pp. 1223–1241, 2002.

[120] S. Sutthithatip, S. Perinpanayagam, S. Aslam, and A. Wileman, "Explainable AI in Aerospace for Enhanced System Performance," in *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, vol. 2021-Octob, Institute of Electrical and Electronics Engineers Inc., 2021.

[121] L. Mandrake, G. Doran, A. Goel, H. Ono, R. Amini, M. S. Feather, L. Fesq, P. Slingerland, L. Perry, B. Bycroft, and J. Kaufman, "Space Applications of a Trusted AI Framework: Experiences and Lessons Learned," in *IEEE Aerospace Conference Proceedings*, vol. 2022-March, IEEE Computer Society, 2022.

[122] Y. Mei, Q. Chen, A. Lensen, B. Xue, and M. Zhang, "Explainable Artificial Intelligence by Genetic Programming: A Survey," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 3, pp. 621–641, 2023.

[123] W. S. Levine, ed., *The Control Handbook - Control System Fundamentals*. CRC Press, 2nd ed., 2011.

[124] W. S. Levine, ed., *The control Handbook - Control System Advanced Methods*. CRC Press, 2011.

[125] S. A. Emami, P. Castaldi, and A. Banazadeh, "Neural network-based flight control systems: Present and future," *Annual Reviews in Control*, vol. 53, no. May, pp. 97–137, 2022.

[126] H. K. Lam, "A review on stability analysis of continuous-time fuzzy-model-based control systems: From membership-function-independent to membership-function-dependent analysis," *Engineering Applications of Artificial Intelligence*, vol. 67, pp. 390–408, jan 2018.

[127] B. Grosman and D. R. Lewin, "Lyapunov-based stability analysis automated by genetic programming," *Automatica*, vol. 45, pp. 252–256, jan 2009.

[128] G. N. Saridis, "Toward the Realization of Intelligent Controls.," *Proceedings of the IEEE*, vol. 67, no. 8, pp. 1115–1133, 1979.

[129] P. J. Antsaklis, "Intelligent Learning Control," *IEEE Control Systems*, vol. 15, no. 3, pp. 5–7, 1995.

[130] P. J. Antsaklis, "Defining Intelligent Control. Report to the Task Force on Intelligent Control," *IEEE Control Systems Society*, pp. 1–31, 1993.

[131] D. Linkens, "Learning systems in intelligent control: an appraisal of fuzzy, neural and genetic algorithm control applications," *Control Theory and Applications,*, pp. 367–386, 1996.

[132] K. Krishn and N. Kulkarni, "Inverse adaptive neuro-control of a turbo-fan engine," in *Guidance, Navigation, and Control Conference and Exhibit*, (Reston, Virigina), pp. 354–364, American Institute of Aeronautics and Astronautics, aug 1999.

[133] D. B. Lavallee, C. Olsen, J. Jacobsohn, and J. Reilly, "Intelligent Control For Spacecraft Autonomy – An Industry Survey," in *Space 2006, AIAA Space Forum*, (San Jose, California), 2006.

[134] P. Guan, X. J. Liu, and J. Z. Liu, "Adaptive fuzzy sliding mode control for flexible satellite," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 4, pp. 451–459, 2005.

[135] B. G. Elkilany, A. A. Abouelsoud, A. M. R. Fathelbab, and H. Ishii, "Potential field method parameters tuning using fuzzy inference system for adaptive formation control of multi-mobile robots," *Robotics*, vol. 9, no. 1, 2020.

[136] E. Kawana and S. Yasunobu, "An intelligent control system using object model by real-time learning," *Proceedings of the SICE Annual Conference*, pp. 2792–2797, 2007.

[137] Y. Zhenghong, "Research on intelligent fuzzy control algorithm for moving path of handling robot," *Proceedings - 2019 International Conference on Robots and Intelligent System, ICRIS 2019*, pp. 50–54, 2019.

[138] Y. Gu, W. Zhao, and Z. Wu, "Online adaptive least squares support vector machine and its application in utility boiler combustion optimization systems," *Journal of Process Control*, vol. 21, pp. 1040–1048, aug 2011.

[139] T. Lee and Y. Kim, "Nonlinear Adaptive Flight Control Using Backstepping and Neural Networks Controller," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 4, pp. 675–682, 2001.

[140] J. S. Brinker and K. A. Wise, "Flight Testing of Reconfigurable Control Law on the X-36 Tailless Aircraft," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 5, pp. 903–909, 2001.

[141] E. N. Johnson and S. K. Kannan, "Adaptive Trajectory Control for Autonomous Helicopters," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 3, pp. 524–538, 2005.

[142] P. Williams-Hayes, "Flight Test Implementation of a Second Generation Intelligent Flight Control System," Tech. Rep. November 2005, NASA Dryden Flight Research Center, 2005.

[143] K. Krishnakumar, "Adaptive Neuro-Control for Spacecraft Attitude Control," in *1994 Proceedings of IEEE International Conference on Control and Applications*, 1994.

[144] K. Sabahi, M. A. Nekoui, M. Teshnehlab, M. Aliyari, and M. Mansouri, "Load frequency control in interconnected power system using modified dynamic neural networks," *2007 Mediterranean Conference on Control and Automation, MED*, 2007.

[145] Y. Becerikli, A. F. Konar, and T. Samad, "Intelligent optimal control with dynamic neural networks," *Neural Networks*, vol. 16, no. 2, pp. 251–259, 2003.

[146] O. Kuljaca, N. Swamy, F. L. Lewis, and C. M. Kwan, "Design and implementation of industrial neural network controller using backstepping," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 1, pp. 193–201, 2003.

[147] P. P. San, B. Ren, S. S. Ge, T. H. Lee, and J. K. Liu, "Adaptive neural network control of hard disk drives with hysteresis friction nonlinearity," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 2, pp. 351–358, 2011.

[148] V. T. Yen, W. Y. Nan, and P. Van Cuong, "Robust Adaptive Sliding Mode Neural Networks Control for Industrial Robot Manipulators," *International Journal of Control, Automation and Systems*, vol. 17, no. 3, pp. 783–792, 2019.

[149] M. Hamid, M. Jamil, and S. I. Butt, "Intelligent control of industrial robotic three degree of freedom crane using Artificial Neural Network," *Proceedings of 2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference, ITNEC 2016*, pp. 113–117, 2016.

[150] D. D. Ligutan, A. C. Abad, and E. P. Dadios, "Adaptive robotic arm control using artificial neural network," *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, HNICEM 2018*, 2019.

[151] Q. Wu, C. M. Lin, W. Fang, F. Chao, L. Yang, C. Shang, and C. Zhou, "Self-organizing brain emotional learning controller network for intelligent control system of mobile robots," *IEEE Access*, vol. 6, pp. 59096–59108, 2018.

[152] S. L. Dai, C. Wang, and F. Luo, "Identification and learning control of ocean surface ship using neural networks," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 4, pp. 801–810, 2012.

[153] C. Nicol, C. J. MacNab, and A. Ramirez-Serrano, "Robust adaptive control of a quadrotor helicopter," *Mechatronics*, vol. 21, pp. 927–938, sep 2011.

[154] B. V. E. How, S. S. Ge, and Y. S. Choo, "Dynamic load positioning for subsea installation via adaptive neural control," *IEEE Journal of Oceanic Engineering*, vol. 35, no. 2, pp. 366–375, 2010.

[155] W. He, Y. Chen, and Z. Yin, "Adaptive Neural Network Control of an Uncertain Robot with Full-State Constraints," *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp. 620–629, 2016.

[156] S. Klecker, B. Hichri, and P. Plapper, "Neuro-inspired reward-based tracking control for robotic manipulators with unknown dynamics," *2017 2nd International Conference on Robotics and Automation Engineering, ICRAE 2017*, vol. 2017-Decem, pp. 21–25, 2018.

[157] Y. Xu, B. Jiang, G. Tao, and Z. Gao, "Fault tolerant control for a class of nonlinear systems with application to near space vehicle," *Circuits, Systems, and Signal Processing*, vol. 30, no. 3, pp. 655–672, 2011.

[158] S. Li and Y. M. Peng, "Neural network-based sliding mode variable structure control for Mars entry," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 226, no. 11, pp. 1373–1386, 2011.

[159] C. Yang, Z. Li, and J. Li, "Trajectory planning and optimized adaptive control for a class of wheeled inverted pendulum vehicle models," *IEEE Transactions on Cybernetics*, vol. 43, no. 1, pp. 24–36, 2013.

[160] M.-u.-d. Qazi, H. Linshu, and T. Elhabian, "Rapid Trajectory Optimization Using Computational Intelligence for Guidance and Conceptual Design of Multistage Space Launch Vehicles," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pp. 1–18, 2005.

[161] C. M. Wen and M. Y. Cheng, "Development of a recurrent fuzzy CMAC with adjustable input space quantization and self-tuning learning rate for control of a dual-axis piezoelectric actuated micromotion stage," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 11, pp. 5105–5115, 2013.

[162] H. A. Talebi, K. Khorasani, and S. Tafazoli, "A recurrent neural-network-based sensor and actuator fault detection and isolation for nonlinear systems with application to the satellite's attitude control subsystem," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 45–60, 2009.

[163] X. Zhang, D. Xu, and Y. Liu, "Intelligent control for large-scale variable speed variable pitch wind turbines," *Journal of Control Theory and Applications*, vol. 2, no. 3, pp. 305–311, 2004.

[164] P. K. Wong, Q. Xu, C. M. Vong, and H. C. Wong, "Rate-dependent hysteresis modeling and control of a piezostage using online support vector machine and relevance vector machine," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 4, pp. 1988–2001, 2012.

[165] H. Moriyama and K. Shimizu, "On-line optimisation of culture temperature for ethanol fermentation using a genetic algorithm," *Journal of Chemical Technology and Biotechnology*, vol. 66, no. 3, pp. 217–222, 1996.

[166] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, and M. Sumner, "A fast adaptive memetic algorithm for online and offline control design of PMSM drives," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 1, pp. 28–41, 2007.

[167] H. Ponce and P. V. C. Souza, "Intelligent control navigation emerging on multiple mobile robots applying social wound treatment," *Proceedings - 2019 IEEE*

33rd International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2019, pp. 559–564, 2019.

[168] C. H. Chiang, "A genetic programming based rule generation approach for intelligent control systems," *3CA 2010 - 2010 International Symposium on Computer, Communication, Control and Automation*, vol. 1, pp. 104–107, 2010.

[169] F. Marchetti, E. Minisci, and A. Riccardi, "Towards Intelligent Control via Genetic Programming," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020.

[170] M. Ceriotti, M. Vasile, G. Giardini, and M. Massari, "An approach to model interest for planetary rover through Dezert- Smarandache Theory," *Journal of Aerospace Computing, Information, and Communication*, vol. 6, no. 2, pp. 92–108, 2009.

[171] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davis, D. Mandl, S. Frye, B. Trout, S. Shulman, and D. Boyer, "Using Autonomy Flight Software to Improve Science Return on Earth Observing One," *Journal of Aerospace Computing, Information, and Communication*, vol. 2, pp. 196–216, 2005.

[172] P. K. Kankar, S. C. Sharma, and S. P. Harsha, "Fault diagnosis of ball bearings using machine learning methods," *Expert Systems with Applications*, vol. 38, pp. 1876–1886, mar 2011.

[173] K. K. Ahn and N. B. Kha, "Modeling and control of shape memory alloy actuators using Preisach model, genetic algorithm and fuzzy logic," *Mechatronics*, vol. 18, no. 3, pp. 141–152, 2008.

[174] Y. Ichikawa and T. Sawa, "Neural Network Application for Direct Feedback Controllers," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 224–231, 1992.

[175] K. Salahshoor, M. Kordestani, and M. S. Khoshro, "Fault detection and diagnosis of an industrial steam turbine using fusion of SVM (support vector machine)

and ANFIS (adaptive neuro-fuzzy inference system) classifiers," *Energy*, vol. 35, pp. 5472–5482, dec 2010.

[176] H. A. Gabbar, A. Sharaf, A. M. Othman, A. S. Eldessouky, and A. A. Abdelsalam, "Intelligent control systems and applications on smart grids," in *Intelligent Systems Reference Library*, vol. 107, pp. 135–163, Springer, 2016.

[177] M. M. A. Al-isawi and J. Z. Sasiadek, "Guidance and Control of Autonomous , Flexible Wing UAV with Advanced Vision System," *2018 23rd International Conference on Methods & Models in Automation & Robotics (MMAR)*, pp. 441–448, 2018.

[178] T. Orlowska-Kowalska and K. Szabat, "Control of the drive system with stiff and elastic couplings using adaptive neuro-fuzzy approach," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 1, pp. 228–240, 2007.

[179] S. Kaitwanidvilai and M. Parnichkun, "Force control in a pneumatic system using hybrid adaptive neuro-fuzzy model reference control," *Mechatronics*, vol. 15, pp. 23–41, feb 2005.

[180] D. A. Handelman, S. H. Lane, and J. J. Gelfand, "Integrating Neural Networks and Knowledge-Based Systems for Intelligent Robotic Control," *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 77–87, 1990.

[181] W. K. Lennon and K. M. Passino, "Intelligent control for brake systems," *IEEE Transactions on Control Systems Technology*, vol. 7, no. 2, pp. 188–202, 1999.

[182] M. Wu, M. Nakano, and J.-H. She, "An expert control strategy using neural networks for the electrolytic process in zinc hydrometallurgy," *Proceedings of the 1999 lEEE International Conference on Control Applications*, vol. 16, no. 2, pp. 135–143, 1999.

[183] M. Vasile, M. Massari, and G. Giardini, "Wisdom - An Advanced Intelligent, Fault-tolerant System for Autonomy in Risky Environments," tech. rep., ESA ITI Contract 18693/04/NL/MV, 2004.

[184] J. S. R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[185] Y. Wang, W. Zhou, J. Luo, H. Yan, H. Pu, and Y. Peng, "Reliable Intelligent Path Following Control for a Robotic Airship Against Sensor Faults," *IEEE/ASME TRANSACTIONS ON MECHATRONICS*, vol. 24, no. 6, pp. 2572–2582, 2019.

[186] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, 1992.

[187] K. J. Åström and R. M. Murray, *Feedback Systems*. Princeton University Press, apr 2010.

[188] C. Utama, B. Karg, C. Meske, and S. Lucia, "Explainable artificial intelligence for deep learning-based model predictive controllers," *2022 26th International Conference on System Theory, Control and Computing, ICSTCC 2022 - Proceedings*, pp. 464–471, 2022.

[189] C. F. Verdier and M. Mazo, Jr., "Formal Controller Synthesis via Genetic Programming," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 7205–7210, 2017.

[190] K. Łapa, K. Cpałka, and A. Przybył, "Genetic programming algorithm for designing of control systems," *Information Technology and Control*, vol. 47, no. 4, pp. 668–683, 2018.

[191] C. K. Oh and G. J. Barlow, "Autonomous controller design for unmanned aerial vehicles using multi-objective genetic programming," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, pp. 1538–1545 Vol.2, 2004.

[192] A. Bourmistrova and S. Khantsis, "Genetic Programming in Application to Flight Control System Design Optimisation," *New Achievements in Evolutionary Computation*, 2010.

[193] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza, *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. No. December 2015, 2003.

[194] R. Poli, W. Langdon, and N. McPhee, *A Field Guide to Genetic Programing*. No. March, 2008.

[195] R. H. Goddard, "A Method of Reaching Extreme Altitudes," *Smithsonian Miscellaneous Collections*, vol. 71, no. 2, 1919.

[196] J. R. Rea, *A Legendre Pseudospectral Method for rapid optimization of launch vehicle trajectories*. PhD thesis, Massachusetts Institute of Technology, 1975.

[197] F. A. Fortin, F. M. De Rainville, M. A. Gardner, M. Parizeau, and C. Gagńe, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, 2012.

[198] M. Crepinsek, S. H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Computing Surveys*, vol. 45, no. 3, pp. 1–33, 2013.

[199] G. Squillero and A. Tonda, "Divergence of character and premature convergence: A survey of methodologies for promoting diversity in evolutionary optimization," *Information Sciences*, vol. 329, pp. 782–799, 2016.

[200] E. Burke, S. Gustafson, G. Kendall, and N. Krasnogor, "Advanced population diversity measures in genetic programming," in *Parallel Problem Solving from Nature — PPSN VII* (J. J. M. Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, and J.-L. Fernández-Villacañas, eds.), (Berlin, Heidelberg), pp. 341–350, Springer Berlin Heidelberg, 2002.

[201] M. W. Aslam, Z. Zhu, and A. K. Nandi, "Diverse partner selection with brood recombination in genetic programming," *Applied Soft Computing Journal*, vol. 67, pp. 558–566, 2018.

[202] P. Day and A. K. Nandi, "Binary string fitness characterization and comparative partner selection in genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 724–735, 2008.

[203] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – A comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[204] S. Luke and L. Panait, "Fighting bloat with nonparametric parsimony pressure," in *Parallel Problem Solving from Nature — PPSN VII* (J. J. M. Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, and J.-L. Fernández-Villacañas, eds.), (Berlin, Heidelberg), pp. 411–421, Springer Berlin Heidelberg, 2002.

[205] J. Žegklitz and P. Pošík, "Benchmarking state-of-the-art symbolic regression algorithms," *Genetic Programming and Evolvable Machines*, 2020.

[206] A. Tsanas and A. Xifara, "Energy efficiency." UCI Machine Learning Repository, 2012.

[207] I.-C. Yeh, "Concrete Compressive Strength." UCI Machine Learning Repository, 2007.

[208] P. D. Brooks Thomas and M. Marcolini, "Airfoil Self-Noise." UCI Machine Learning Repository, 2014.

[209] J. P. Rosca, "Entropy-Driven Adaptive Representation," *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pp. 23–32, 1995.

[210] J. Song, X. Tong, X. Xu, and K. Zhao, "A Real-Time Reentry Guidance Method for Hypersonic Vehicles Based on a Time2vec and Transformer Network," *Aerospace*, vol. 9, no. 8, 2022.

[211] B. Gaudet, R. Linares, and R. Furfaro, "Deep reinforcement learning for six degree-of-freedom planetary landing," *Advances in Space Research*, vol. 65, no. 7, pp. 1723–1741, 2020.

[212] B. Gaudet, R. Linares, and R. Furfaro, "Adaptive guidance and integrated navigation with reinforcement meta-learning," *Acta Astronautica*, vol. 169, no. January, pp. 180–190, 2020.

[213] B. Gaudet, K. Drozd, and R. Furfaro, "Adaptive Approach Phase Guidance for a Hypersonic Glider via Reinforcement Meta Learning," in *AIAA SCITECH 2022 Forum*, (Reston, Virginia), American Institute of Aeronautics and Astronautics, jan 2022.

[214] B. A. Costa, F. L. Parente, J. Belfo, N. Somma, P. Rosa, J. M. Igreja, J. Belhadj, and J. M. Lemos, "A reinforcement learning approach for adaptive tracking control of a reusable rocket model in a landing scenario," *Neurocomputing*, vol. 577, no. November 2023, p. 127377, 2024.

[215] S. Xu, Y. Guan, C. Wei, Y. Li, and L. Xu, "Reinforcement-Learning-Based Tracking Control with Fixed-Time Prescribed Performance for Reusable Launch Vehicle under Input Constraints," *Applied Sciences (Switzerland)*, vol. 12, no. 15, 2022.

[216] C. Schmitt and B. Burchett, "Fuzzy IPPD Guidance for Approach and Landing of a Reusable Launch Vehicle," in *AIAA 1st Intelligent Systems Technical Conference*, vol. 1, (Reston, Virigina), pp. 276–281, American Institute of Aeronautics and Astronautics, sep 2004.

[217] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, 2009.

[218] M. Schmidt and H. Lipson, "Symbolic Regression of Implicit Equations," in *Genetic Programming Theory and Practice VII* (R. Riolo, U.-M. O'Reilly, and T. McConaghy, eds.), pp. 73–85, Boston, MA: Springer US, 2010.

[219] A. Kratsios and E. Bilokopytov, "Non-Euclidean universal approximation," *Advances in Neural Information Processing Systems*, vol. 2020-Decem, no. 1, pp. 1–12, 2020.

[220] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[221] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: From architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[222] G. V. Chowdhary and E. N. Johnson, "Theory and flight-Test validation of a concurrent-learning adaptive controller," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 2, pp. 592–607, 2011.

[223] F. Lewis, A. Yesildirek, and Kai Liu, "Multilayer neural-net robot controller with guaranteed tracking performance," *IEEE Transactions on Neural Networks*, vol. 7, pp. 388–399, mar 1996.

[224] Y. H. Kim and F. L. Lewis, *High-Level Feedback Control with Neural Networks*, vol. 21 of *World Scientific Series in Robotics and Intelligent Systems*. WORLD SCIENTIFIC, sep 1998.

[225] S. Zhang, Y. Dong, Y. Ouyang, Z. Yin, and K. Peng, "Adaptive Neural Control for Robotic Manipulators with Output Constraints and Uncertainties," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5554–5564, 2018.

[226] Y. J. Liu, S. Tong, C. L. Chen, and D. J. Li, "Neural controller design-based adaptive control for nonlinear MIMO systems with unknown hysteresis inputs," *IEEE Transactions on Cybernetics*, vol. 46, no. 1, pp. 9–19, 2016.

[227] V. Stanovov, S. Akhmedova, and E. Semenkin, "The automatic design of parameter adaptation techniques for differential evolution with genetic programming," *Knowledge-Based Systems*, vol. 239, p. 108070, 2022.

[228] F. Lewis, "Nonlinear Network Structures for Feedback Control," *Asian Journal of Control*, vol. 1, no. 4, pp. 205–228, 1999.

[229] D. P. Searson, D. E. Leahy, and M. J. Willis, "GPTIPS:An open source genetic programming toolbox for multigene symbolic regression," *Proceedings of the In-*

ternational MultiConference of Engineers and Computer Scientists 2010, IMECS 2010, no. December, pp. 77–80, 2010.

[230] C. Dujarric, "Possible Future European Launchers – A Process of Convergence," *ESA Bulletin 97*, 1999.

[231] D. Burleson, "The European Space Agency's FESTIP initiative," in *AIP Conference Proceedings*, vol. 420, pp. 921–925, AIP, 1998.

[232] C. Dujarric, M. Caporicci, H. Kuczera, and P. Sacher, "Conceptual studies and technology requirements for a new generation of European launchers," *Acta Astronautica*, vol. 41, no. 4-10, pp. 219–228, 1997.

[233] P. González, "Influence of the abort capability in reusable systems reliability. Festip results overview," *9th International Space Planes and Hypersonic Systems and Technologies Conference*, no. c, pp. 1–9, 1999.

[234] K. P. Bollino, I. M. Ross, and D. D. Doman, "Optimal nonlinear feedback guidance for reentry vehicles," *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference 2006*, vol. 1, pp. 563–582, 2006.

[235] S. T. u. I. Rizvi, L. shu He, and D. jun Xu, "Optimal trajectory and heat load analysis of different shape lifting reentry vehicles for medium range application," *Defence Technology*, vol. 11, no. 4, pp. 350–361, 2015.

[236] F. Pescetelli, E. Minisci, C. Maddock, I. Taylor, and R. E. Brown, "Ascent trajectory optimisation for a single-stage-to-orbit vehicle with hybrid propulsion," in *18th AIAA/3AF International Space Planes and Hypersonic Systems and Technologies Conference 2012*, pp. 1–18, 2012.

[237] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vijay, F. Viégas, O. Vinyals,

P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," 2015.

[238] F. Gavilan, R. Vazquez, and J. Á. Acosta, "Output-feedback control of the longitudinal flight dynamics using adaptative backstepping," *Proceedings of the IEEE Conference on Decision and Control*, pp. 6858–6863, 2011.

[239] W. A. Butt, L. Yan, and K. Amezquita S., "Adaptive integral dynamic surface control of a hypersonic flight vehicle," *International Journal of Systems Science*, vol. 46, no. 10, pp. 1717–1728, 2015.

[240] Y. Wang, T. Chao, S. Wang, and M. Yang, "Byrnes-Isidori-based dynamic sliding-mode control for nonminimum phase hypersonic vehicles," *Aerospace Science and Technology*, vol. 95, p. 105478, 2019.

[241] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next," *Journal of Scientific Computing*, vol. 92, no. 3, pp. 1–62, 2022.

[242] A. Y. Amte and P. S. Kate, "Automatic generation of Lyapunov function using Genetic programming approach," in *2015 International Conference on Energy Systems and Applications*, no. Icesa, pp. 771–775, IEEE, oct 2015.