ROBUST NEUROCONTROL FOR AUTONOMOUS SOARING

NEUROCONTRÔLE ROBUSTE POUR LE VOL EN SOARING AUTONOME

A Thesis Submitted to the Division of Graduate Studies
of the Royal Military College of Canada
by

Eric Jihun Kim, B.Eng.
Second Lieutenant

In Partial Fulfillment of the Requirements for the Degree of
Master of Applied Science in Aeronautical Engineering

April, 2022

A labour of lockdowns.

Acknowledgments

I would like to express my gratitude to my supervisor, Dr. Ruben E. Perez. His sharing of insight and knowledge on multiple subjects, the attitude of excitement he imparted towards research, and his constant guidance throughout my graduate studies allowed for the completion of this work. I am extremely thankful for the many opportunities he provided me and for continually raising the bar to ensure the highest standards, which made all the effort worthwhile. I would also like to thank the other members of the Advanced Aircraft Design Lab for their help and company at the weekly virtual gatherings. Lastly, a thank you to my parents for supporting me throughout these interesting times.

Abstract

Kim, Eric Jihun. M.A.Sc. Royal Military College of Canada, April 2022. *Robust Neurocontrol for Autonomous Soaring.* Supervised by Ruben E. Perez, B.Eng., M.A.Sc., Ph.D., P.Eng., Associate Professor.

Currently, the flight endurance of small unmanned aerial vehicles is limited by their onboard energy capacity. Improving flight duration can increase the utility of such vehicles and consequently benefit multiple domains from scientific data acquisition to military surveillance. Biologically-inspired techniques such as dynamic, thermal, and ridge soaring offer a method of achieving this goal by allowing autonomous aircraft to exploit naturally occurring wind gradients and engage in thrustless flight, reducing the expenditure of energy reserves. Due to the nonlinear dynamics of aerial systems and the non-ideal, stochastic nature of real-world environments, recent studies in the field of autonomous soaring have explored the use of artificial neural networks to generate the soaring trajectory and the subsequent guidance controls. Nevertheless, the computational limits onboard SUAVs and the necessity for robust operation make practical implementation a challenge. As a response, this study explores the use of the Neuroevolution of Augmenting Topologies algorithm to train efficient, effective, and robust neurocontrollers that can control a simulated aircraft along sustainable soaring trajectories in the presence of varying initial states and environmental conditions, stochastic disturbances, and system noise. The proposed approach evolves neural networks in a way that preserves simplicity while maximizing performance, which allows the resulting policies to be understood, implemented, and operated onboard real-time SUAV platforms. Application of the evolutionary method resulted in neural networks capable of conducting various soaring techniques in a range of environments, with robust neurocontrollers successfully performing autonomous soaring up to 4.4 times as often as deterministic neural networks. This research introduces the novel neuroevolutionary control method whose applicability is not limited to aerial systems, a method of quantifying and measuring robustness for the comparison of different control solutions, a set of metrics to identify the topological characteristics that encode robustness, and validation for the presented approach in the form of a software-in-the-loop implementation.

Keywords: autonomous soaring, neurocontrol, artificial neural network, neuroevolution, unmanned aerial vehicle, aircraft control, trajectory optimization, dynamic soaring, thermal soaring, ridge soaring

Résumé

Kim, Eric Jihun. M.A.Sc. Collège militaire royale du Canada, Avril 2022. *Neurocontrôle Robuste pour le Vol en Soaring Autonome.* Thèse dirigée par Ruben E. Perez, B.Eng., M.A.Sc., Ph.D., P.Eng., Professeur adjoint.

Actuellement, l'endurance de vol des drones aériens miniatures est limitée par leur capacité d'énergie à bord. L'amélioration de l'endurance de vol peut augmenter l'utilité de ces véhicules et ensuite bénéficier à plusieurs domaines comme l'acquisition des données scientifiques, ou la surveillance militaire. Les techniques inspirées par la biologie comme le vol de gradient, le vol thermique et le vol d'arête fournissent une façon d'atteindre ce but en permettant à des aéronefs autonomes d'exploiter des phénomènes météorologiques et d'engager un vol plané, réduisant conséquemment la dépense des réserves d'énergie. À cause des dynamiques non-linéaires des systèmes aériens et la nature non-idéale et stochastique des environnements réels, des études récentes dans le domaine du vol plané autonome ont exploré l'usage des réseaux de neurones artificiels pour produire la trajectoire de vol et les commandes de guidage. Néanmoins, les limites informatiques à bord des drones miniatures et la nécessité pour l'opération robuste rendent difficile la mise en oeuvre pratique. Comme réponse, cette étude explore l'usage de l'algorithme «Neuroevolution of Augmenting Topologies» pour former les neurocontrôleurs efficaces et robustes, qui peuvent contrôler un aéronef simulé sur des trajectoires de vol plané viables en présence des états initiaux et environnements variables, des perturbations stochastiques et du bruit du système. L'approche proposée évolue des réseaux de neurones artificiels d'une façon avec laquelle la simplicité est conservée et la performance est maximisée, ce qui permet les systèmes qui en découlent d'être compris, mis en oeuvre, et faire fonctionner à bord des plateformes de drone en temps réel. L'application de la méthode évolutive a donné lieu à des réseaux capables de mener une variété de techniques de vol plané dans une plage d'environnements, où les neurocontrôleurs robustes ont réalisé le vol plané autonome avec succès jusqu'à 4,4 fois plus souvent que les neurocontrôleurs entraînés en environnements déterministes. Cette recherche présente la nouvelle méthode de contrôle neuro-évolutif, dont l'applicabilité n'est pas limitée aux systèmes aériens, une façon de quantifier et mesurer la robustesse pour la comparaison des solutions différentes de contrôle, un ensemble des indicateurs pour identifier les caractéristiques topologiques qui encodent pour la robustesse et la validation pour l'approche proposée avec une simulation à logiciel dans la boucle.

Mots-clés: vol plané autonome, neurocontrôle, réseau de neurones artificiels, neuro-évolution, drone, contrôle d'avion, optimisation de trajectoire, vol de gradient, vol thermique, vol d'arête

Contents

List of Tables

List of Figures

Nomenclature

Roman Symbols

$\hat{h}$	Ridge shear transition height [ft]
D	Drag [N]
e_K	Kinetic energy [J]
e_P	Potential energy [J]
e_T	Total energy [J]
g	Standard gravity [ft/s^2]
h	Height [ft]
h_{tr}	Shear wind transition height [ft]
h_t	Thermal bubble height [ft]
$h_{th_{max}}$	Maximum thermal persistence duration [s]
$h_{tr_{max}}$	Thermal ceiling height [ft]
L	Lift [N]
l	Ridge length [ft]
m	Mass [slug]
R	Ridge radius [ft]
r	Distance to thermal centre, projected on horizontal plane [ft]
r_z	Thermal bubble radius along z-axis [ft]
r_{xy}	Thermal bubble radius along horizontal plane [ft]
S	Wing reference area [s^2]
V	Airspeed [ft/s]
W	Wind strength [ft/s]
w_core	Maximum thermal core updraft wind speed [ft/s]

$W_{\hat{h}}$	Wind speed at ridge shear transition height [ft/s]
w_∞	Ridge shear wind speed at infinite distance from ridge [ft]
x	X-axis position [ft]
y	Y-axis position [ft]
z_0	Ridge shear height closest to terrain [ft]

Greek Symbols

β	Wind profile gradient [ft/s/ft]
δ	Discrete time step [s]
γ	Pitch [deg]
μ	Roll [deg]
π	Artificial intelligence policy
ψ	Heading [deg]
ρ	Local air density [slug/ft^3]
θ	Aircraft elevation from centre of ridge [deg]
θ'	Greatest-lift aircraft elevation from centre of ridge [deg]

Dimensionless Groups

A	Shear wind shape parameter [-]
C_D	Drag coefficient [-]
C_L	Lift coefficient [-]
C_{D0}	Zero-lift drag coefficient [-]
E_{max}	Maximum lift-to-drag ratio [-]
K	Induced drag factor [-]
n	Load factor [-]

Roman Subscripts

x	X-axis
y	Y-axis
z	Z-axis

Abbreviations

$3DOF$	Three degree of freedom
ANN	Artificial neural network
DNN	Deep neural network
$NEAT$	Neuroevolution of Augmenting Topologies
NEC	Neuroevolutionary control
NLP	Nonlinear programming
RL	Reinforcement learning
$SNEC$	Stochastic neuroevolutionary control
$SUAV$	Small unmanned aerial vehicle

1 Introduction

1.1 Motivation for the Research

Interest in small unmanned aerial vehicles (SUAVs) has steadily been increasing due to their utility in a variety of applications ranging from scientific data acquisition to military surveillance. Unmanned, lightweight aircraft fitted with sensor arrays could be rapidly deployed to map terrain, record meteorological conditions, conduct aerial patrols, and gather intelligence, all with fewer resources and support infrastructure than manned vehicles. However, the ability of current SUAVs to perform operations for extended periods of time is currently limited, as the power storage limits onboard small-scale aircraft greatly restrict their maximum flight times. In addition, the remote operation of such vehicles is both impractical for long-duration missions due to the costly commitment of human resources, and unreliable for long-distance flights as a result of the reduced availability and quality of wireless communication links in distant environments. These considerations have motivated research in biologically-inspired autonomous soaring systems. In nature, certain species of birds such as the albatross and the frigatebird have evolved methods of extracting energy from the wind by soaring in cyclical patterns that exploit certain atmospheric features. One such technique known as dynamic soaring can be observed in albatross, who perform cyclic manoeuvres using horizontal winds to travel across large, transoceanic distances without flapping their wings [1]. Similarly, thermal soaring allows many birds such as eagles, hawks, and seabirds to continuously loiter over oceans using columns of rising air, or thermals [2]. Since the discovery of these energy-efficient techniques, autonomous soaring research for SUAV applications has been a growing field pursuing a multidimensional challenge that involves wind field mapping, trajectory planning, and aircraft control.

The overall problem begins with estimating the local wind field around an aircraft to identify the presence of an environment suitable for soaring manoeuvres. Past efforts have largely achieved this task through various approximation and filtering methods [3, 4]. Once the wind profile has been identified, an energy-efficient soaring trajectory that utilizes the local atmospheric features while taking into account any mission objectives must be rapidly computed before the start of the next soaring cycle for sustained soaring and high-level navigation. However, this process is complicated by the limited computational resources onboard SUAVs, making suboptimal, simplified solutions more suitable than numerically optimal but calculation-intensive approaches. Assuming the timely generation of a path, control commands must then be actuated through a control scheme to steer the aircraft along the trajectory. Nevertheless, tracking flight paths is also difficult, considering the uncertain and stochastic disturbances present in real-world environments, which are likely to cause tracking errors where the vehicle is perturbed or drifted away from the reference trajectory. Correcting such deviations requires additional energy expenditure that can compromise the vehicle's abil-

ity to continue soaring or reduce its maximum flight time, which is a consequence of the lack of flexibility in the control approaches that have been explored in the field thus far. Therefore, the planning and control problems remain as significant challenges that have yet to be both solved and applied through a simple and robust solution capable of autonomously soaring in a variety of conditions. Robustness in this context is the degree to which an autonomous system can sustain soaring manoeuvres in the presence of uncertainties, disturbances, and noise. To avoid the limitations of the tracking approach, and motivated by the need for simple and robust controllers capable of generalized behaviour, the current research investigates a neural network-based evolutionary training approach. Mapping states to actions through the nonlinear nature of neural networks may allow for control behaviours that can adapt and remain effective in different aircraft and environmental parameters. Furthermore, an evolutionary method of training such controllers can result in sparse, efficient networks that can be implemented on computationally-limited hardware platforms.

1.2 Research Objectives

The aim of this research is to investigate the applicability of a neuroevolutionary control approach to autonomous soaring. Through simulation testing, the work analyzes the ability of the neuroevolutionary method's resulting controllers to serve as simple, robust, and combined trajectory planning and control solutions for generating and actualizing the paths of various autonomous soaring techniques. This study is concerned with the real-time onboard control task of generating the higher-level paths that enable energy-efficient soaring, instead of the lower-level systems that manage aircraft stabilization. Specifically, the objectives consist of the following:

i. Develop a neuroevolutionary method of generating neurocontrollers that can be applied to various forms of autonomous soaring;
ii. Develop a method of measuring the robustness of neurocontrollers to various uncertainties and disturbances;
iii. Develop a method of analyzing and understanding the structural features of irregular neural networks;
iv. Assess the performance of the neurocontrollers against canonical soaring trajectories;
v. Assess the neuroevolutionary method's ability to instill in its neurocontrollers robustness against different sources of uncertainties and disturbances;
vi. Explore the implementability of the real-time robust control strategy.

1.3 Thesis Layout

The remainder of the thesis dissertation is organized into seven chapters. Chapter 2 presents a literature review examining the existing body of autonomous soaring research and identifying the knowledge gaps in the field. Chapter 3 introduces the flight characteristics and dynamic equations of the representative SUAV used in the research, as well as mathematical descriptions of the various canonical wind models that simulate the environments necessary for real-world soaring. The section also describes the optimal control problem and its relevance to the research, and includes a canonical example of an optimal soaring trajectory. Chapter 4 presents the novel neuroevolutionary control process developed to generate soaring-capable neurocontrollers for deterministic environments, including the corresponding case studies of

the evolved neural networks for the various types of autonomous soaring. Chapter 5 reveals the stochastic neuroevolutionary process that was subsequently developed to tackle the robustness problem, as well as the descriptions of the stochastic simulation environments and the Monte-Carlo estimator designed to measure relative robustness. The section also includes case studies of different neural network groups trained for the uncertain environments to assess the validity and applicability of the stochastic neuroevolutionary method to physical systems. Chapter 6 provides software-in-the-loop flight test results of a soaring neurocontroller that was evolved and implemented in an autopilot framework for real-time aircraft control. Lastly, Chapter 7 concludes the dissertation with final remarks and recommendations for future research on the topic.

1.4 Contributions

The contributions of this study include the development of:

i. A novel approach to generate simple, interpretable neurocontrollers for autonomous soaring;
ii. A novel method to generate robust neurocontrollers that is not limited to aeronautical applications;
iii. A way to measure the robustness of a controller to different sources of uncertainties and disturbances;
iv. A tool to analyze the structural characteristics and parameters of irregular neural networks and their effects on network behaviour.
v. A software-in-the-loop test of an in silico implementation of the autonomous soaring control approach.

2 Literature Review

Autonomous soaring for small unmanned aerial vehicles is an actively researched field that has largely been inspired by biologically-evolved flight techniques observed in nature. Two such methods on which most of the research has been focused are dynamic and thermal soaring. Enabling an SUAV to perform unique soaring manoeuvres by extracting energy from the local wind profile using onboard sensors and limited computational power is a yet unrealized goal that is made particularly difficult by the stochastic, unpredictable nature of real world environments. This chapter presents a summarized analysis of the existing body of autonomous soaring literature organized into the subfields of wind detection, trajectory planning, and aircraft control, before defining the direction of this work.

2.1 Wind Detection

Whether exploiting horizontal winds for dynamic soaring or climbing rising air columns in thermal soaring, one of the first requirements for the autonomous application of such techniques is the detection of a suitable wind environment. Although environments conducive for these types of flight can often be expected in certain geographical and meteorological conditions, unmanned aerial vehicles must be able to measure and identify the local wind profile in real time as it continually changes position. Specifically, in the case of dynamic soaring, a region with a sufficiently large vertical wind gradient must be found, and one primary method of identifying such environments is through probabilistic models. Lawrance and Sukkarieh developed a wind estimation technique based on onboard pitot static airspeed sensors, GPS data, and sideslip wind vane measurements [3]. The method estimated the horizontal components of wind around the aircraft by taking the mean of all directional measurements to obtain the wind bearing, while using Gaussian process regression to estimate the altitude-dependent wind speed variation. This method provided a mean function as well as confidence bounds for the wind gradient, and since a minimum wind gradient is required for a flight agent to conduct dynamic soaring manoeuvres, it was possible to estimate the maximum and minimum altitude limits beyond which soaring would not be feasible. Estimates of the wind profile were based on the assumption that the bearing is constant over large, unobstructed surfaces such as oceans, which are where albatross birds can be most often observed performing dynamic soaring. Simulated flight results showed that the proposed wind estimation method was capable of accurately modelling the local wind profile while being inexpensive enough to achieve on an SUAV's on-board computer.

Instead of using sensor measurements to directly approximate a wind profile, Langelaan *et al.* developed a parameterized approach of representing the wind field using polynomials [4]. The parameters of polynomial functions that described typical wind profiles for dynamic

soaring were estimated through sensors available onboard SUAVs to generate the complete wind model. The method used a Kalman filter to predict the wind velocity by estimating the set of polynomial parameters, assumed to be affected by Gaussian measurement noise. Additionally, in the likely event that an *a priori* estimate of the wind profile from external meteorological data were unavailable, the Kalman filter was initialized using a series of wind measurements taken by the aircraft at different altitudes. Due to the iterative nature of the Kalman update equations and the low order of the matrix operations, this method could be performed rapidly without extensive computational resources. Although the estimated wind profile parameters had difficulty converging to those of ground truth measurements for more complex profile shapes, the presented approach performed well in approximating linear wind shears such as those commonly found in the lee of ridges [4].

While wind estimation for dynamic soaring involves the identification of the horizontal wind strength at different altitudes, thermal soaring is concerned with detecting the wind strengths and radii of updraft regions. The primary form of wind approximation involves experimentally detecting thermal columns, such as the method presented by Allen and Lin [5]. During flight, measurements of the SUAV's position and rate of change of energy were recorded and placed in a queue, before being used to estimate the thermal centre, a region in which the aircraft's energy would have been relatively high. The thermal radius was calculated through a gradient descent optimization that minimized the difference between a thermal radius distribution constructed from the positional readings and a radius estimate generated through the energy history. Since thermals are known to change position over time, the queue was continually updated to place a greater emphasis on more recent entries. This allowed the proposed wind estimation approach to successfully track moving columns and exploit thermals for longer durations, as supported by the experimental results.

In expanding the problem of thermal soaring to include the detecting and tracking of updrafts for interthermal flight, Depenbusch *et al.* more recently developed a method of generating an atmospheric mapping of the wind profile [6]. The approach continually estimated the local updraft strength at the aircraft's location by computing the difference between the vehicle's estimated energy rate of change, obtained through an unscented Kalman filter applied to various sensor measurements readily available on typical SUAVs, and the anticipated sink rate, defined by aerodynamic equations. For instance, if the aircraft rose more quickly than expected, it could be assumed that it was in an updraft region with a vertical wind velocity proportional to the difference between the estimated and anticipated energy rates. This series of mean wind estimates and the variances was superimposed over a square grid of the environment in which the aircraft was expected to remain, where a single wind value was used to represent the region over each discrete cell of the grid. In addition to wind strength, a thermal column must be of sufficient radius so that an aircraft stays within its turn limits when circling the updraft region. To account for this aspect, a two-dimensional mask of grid cells in the shape of a doughnut, with the inner and outer circles representing the vehicle's minimum and maximum typical orbit radii respectively, was created. A convolution of this mask with the previous map of the estimated wind velocities resulted in a map of probabilities that indicated how likely circling manoeuvres centred around a given grid cell would yield sufficient climb rates. Instead of explicitly obtaining the wind speed and radius of a thermal, the atmospheric mapping approach generated a feasibility map from the wind speeds estimated at various locations in the grid. Such a method allows for higher-level path planning and trajectory generation algorithms that can use and reuse multiple thermals to

extend flight times, which holds a significant advantage over completely relying on chance encounters to remain soaring.

These various successful approaches of detecting wind for both dynamic and thermal soaring shows that estimating the wind profile is currently feasible and is not necessarily the primary obstacle that lies in the way of implementing autonomous soaring.

2.2 Trajectory Planning

After an aircraft has detected a sufficient wind environment that contains the necessary characteristics for soaring, the vehicle must be placed on a path, or trajectory, that can take advantage of the wind. In thermal soaring, such paths, illustrated in Fig. 2.1, are quite simple once an updraft region has been found, typically defined by only a constant bank angle that dictates the turn radius of the aircraft as it circles the updraft column.

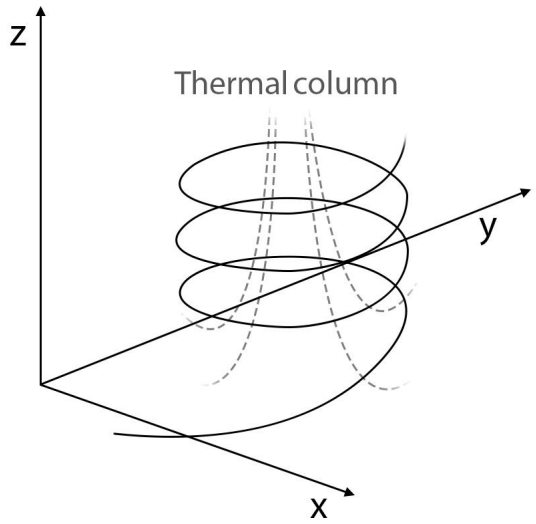
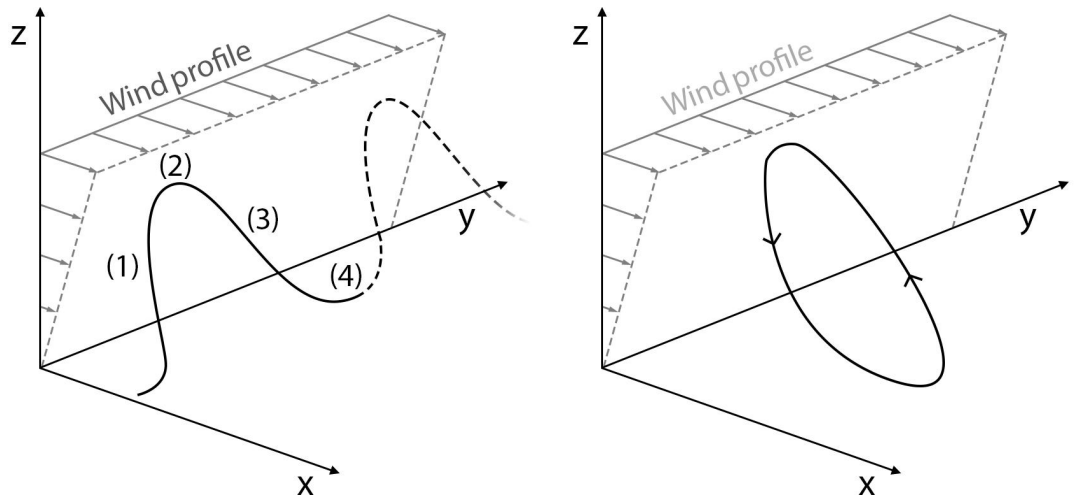


Figure 2.1: Thermal soaring trajectory with a thermal column.

Consequently, research in thermal soaring has not often separated the trajectory planning and aircraft control tasks into two distinct problems, resulting in the forgoing of any planning systems by directly commanding aircraft along soaring manoeuvres through heuristic-based control schemes. On the contrary, dynamic soaring is an area in which trajectory planning is a significant challenge. Rayleigh first made observations on the soaring patterns of seabirds, identifying the key elements of a travelling dynamic soaring trajectory [7]:

1. Upwind climb
2. High altitude turn
3. Downward sink
4. Low altitude turn

In examining such trajectories, where energy management must be strictly regulated in the presence of sharp flight manoeuvres, the field has seen a significant amount of research



(a) Travelling (b) Loitering

Figure 2.2: Travelling dynamic soaring trajectory with a linear wind profile.

involving the use of numerical trajectory optimization. Zhao converted the optimal control problem of computing dynamic soaring trajectories with minimum cycle times into a nonlinear programming (NLP) problem, which was solved using a numerical optimizer [8]. The process required a model of a glider defined by aerodynamic equations of motion, a cost function that directed the optimization goal, and boundary constraints to ensure the resulting flight path remained within the limits of the model and environment. Various trajectories were obtained through a sequential quadratic programming-based NLP solver by applying different objectives, such as minimizing cycle time, maximizing altitude gain, and minimizing the wind gradient that still led to successful soaring. The resulting loitering and travelling flight paths were characteristic of the behaviour observed in albatross birds and allowed for an analysis of the environmental conditions and control objectives conducive to dynamic soaring. Another instance of numerical trajectory optimization can be found in the research conducted by Sachs [1]. A commercial optimizer was used in generating optimal trajectories to determine the minimum strength of shear wind required for dynamic soaring manoeuvres. The findings coincided with data on the wind environments in which albatrosses had been found to soar, noting that the calculated minimum wind strength can often be found or exceeded in such locations. As these studies show, generating optimal trajectories by numerically solving a nonlinear programming problem was an early method of examining the characteristics and general feasibility of dynamic soaring. Although such techniques soon became the basis of a variety of approaches in supplying optimal control trajectories for real-world soaring, the extremely lengthy calculation times and large computational power requirements of numerical optimization methods greatly hindered their applicability as practical solutions. Furthermore, the resulting single-cycle trajectories were insufficient for the repeating nature of dynamic soaring, particularly as they often consisted of harsh state and control transitions while not accounting for aircraft behaviour and energy accumulation between cycles [1].

To take advantage of the numerically optimal trajectories as well as the greater availability of data storage in contrast to the limited computational power available on SUAVs, other strategies have relied on pre-computed paths. Bird *et al.* proposed a trajectory planning approach based on a database of locally optimal loitering trajectories computed for a range of

aircraft and environmental conditions, which was loaded ahead of time onto an SUAV computer [9]. The *a priori* trajectories varied in terms of the aircraft airspeed at the lowest point of the trajectory, the altitude range of the shear layer, and the maximum wind speed of the shear. A nearest neighbour interpolation method selected the flight path that best matched the conditions measured by the aircraft's sensors, which was then loaded to be tracked by the attitude controller. However, even if a representative trajectory could be computed and stored for every meaningful discrete value of a set of parameters, the system would be susceptible to any unexpected deviations or gradual drift from the reference flight path. The authors remarked that optimal trajectories are simply not realistic, due to the imperfect and stochastic nature of real-world systems. For instance, any uncertainties or disturbances that lead to tracking errors would require the aircraft to expend energy in correcting its trajectory, which is a critical problem when considering the strict energy management necessary for dynamic soaring. In summary, systems that depend on tracking a reference trajectory are at constant risk of depleting the minimal energy gains accumulated over dynamic soaring cycles, eventually becoming incapable of continuing soaring manoeuvres [9].

Another class of planning methods that has been explored in the field is the parameterization approach. In a more recent work, Li and Langelaan explored robustness through a geometric parametrization strategy that used splines and sinusoidal functions to model and optimize dynamic soaring flight paths [10]. Instead of optimizing for every point of a soaring trajectory, which has proven to be a lengthy, computationally expensive process, the proposed method simplified the way in which suitable soaring cycles were defined, allowing for a more rapid and lightweight optimization. Cubic polynomials and sine functions formed the basic structures of the trajectories, where the coefficients, frequencies, and biases formed the set of parameters that were optimized to maximize the total energy experienced by the aircraft over individual soaring cycles. Simulation results showed that the parameterization approach required fewer function evaluations than classic point-wise optimization by several orders of magnitude, allowing for real-time trajectory generation. Furthermore, since the optimization output trajectories are not defined as functions of time, it was found that the trajectory for one cycle could be reused, as although gains in energy caused simulated aircraft to slowly deviate from the original reference trajectory, the system would eventually convergence to an equilibrium. This behaviour was also observed when the simulation was initialized with varying initial aircraft conditions, indicating a certain level of robustness. To further generalize the soaring behaviour, a deep neural network (DNN) was trained through an actor critic reinforcement learning (RL) method in a continuous action space. The trained neural network was able to output the sinusoidal parameters for loitering dynamic soaring cycles, and since training occurred off-line, generating each trajectory through a forward pass of the neural network required minimal computational effort. Finally, although the trajectory planning approach could compensate for varying initial conditions, it was ultimately reliant on a tracking controller that would be sensitive to stochastic disturbances and more significant uncertainties. Nevertheless this use of deep reinforcement learning exemplifies the latest trend in the field, which is the application of artificial neural networks (ANNs) to achieve robust, generalized behaviour.

2.3 Aircraft Control

Research in autonomous soaring control systems has largely been concerned with higher level trajectory tracking techniques, and can be classified into three main categories, the first of which is the heuristic approach. Many of the heuristic control strategies in thermal soaring have been based on the experiential findings of Reichmann [11]. The approach lays out simple rules for setting the bank angle of an aircraft based on its climb rate in a region of updraft, and suggests increasing, decreasing, or maintaining constant the bank angle as the climb rate deteriorates, improves, or remains constant, respectively. The main advantages of such an approach is that its simplicity only requires the consideration of the control surfaces related to an aircraft's roll, an estimate of the wind profile is not required, implementation does not require extensive computational power and storage requirements, and it is generalizable to different aircraft and thermal conditions. Andersson *et al.* [12] applied the heuristic approach as a foundation for a turn rate controller that used the acceleration in specific energy, where the total energy of the aircraft measured using onboard sensors was normalized with respect to its mass, and the rate of change in this specific energy was interpreted as the climb rate. A series of logic operations was designed to trigger the circling controller if the energy rate achieved a threshold, after which the rate value was used as an input to the proportional-derivative controller dictating the aircraft's turn rate.

Similarly, Wharington initially explored the applicability of heuristics for dynamic soaring control [13]. Due to the more complex manoeuvres required for dynamic soaring, the proposed method investigated the use of separate pitch and bank control schemes. Assuming a net gain in kinetic energy over a soaring cycle, a pitch controller was designed to govern the aircraft's glide slope to achieve constant soaring velocity while exchanging the excess speed with height. As a note, while the primary method of energy extraction in thermal soaring is to increase and decrease wing loading by banking the aircraft, in dynamic soaring, energy is gained by changing the aircraft's heading relative to the horizontal wind. As such, the roll heuristic was implemented to bank the vehicle so that it would turn away from the incoming wind. These rules formed the basis of multiple controller designs, which were created to address different types of wind profiles. From testing the heuristic-based controllers in simulation, it was found that the resulting trajectories consisted of significant variations in airspeed, clearly showing the ineffectiveness of the pitch controller and consequently invalidating the corresponding heuristic. It was also concluded that the heuristic-based design approach resulted in controllers that yielded suboptimal trajectories when compared to those of numerical optimization methods.

A semi-heuristic strategy by Lawrance and Sukkarieh [3] proposed a piecewise dynamic soaring control method based on optimized segments of the soaring trajectory illustrated earlier in Fig. 2.2a. The piecewise segments were optimized offline, or prior to flight, through numerical optimization before being uploaded to the control system as a reference trajectory, which the aircraft was made to track through simple PID controllers commanding the rudder, ailerons, and elevator. The method continuously looped through each part of the cycle, transitioning from one phase to the next when certain conditions were met. Specifically, the controller logic would switch from the upwind climb phase to high altitude turn once the maximum altitude, based on wind profile estimates, was reached, to the downward sink after the aircraft was reoriented to point downwind, and to the low altitude turn when the minimum altitude was reached. An additional travelling component was added between cycles to expend

any additional energy gained from soaring beyond the amount necessary to continue cycling. This phase also allowed for directional travelling soaring, as the vehicle could be made to soar towards mission waypoints. Lastly, once all the excess energy had been expended, the control cycle would return to the upwind climb. Simulation testing showed that the controller could perform directional soaring over multiple cycles, and that the low complexity of the approach would make implementation possible. However, the primary disadvantage of the method was its limited applicability, because the reference trajectories had been optimized within a single environment with specific parameters that would not necessarily be suitable for the range of conditions encountered by a real system.

The second category of soaring control systems is characterized by the use of linear models to approximate the nonlinearities of real-world soaring. In one application of this approach, Akhtar *et al.* used a PID-based tracking controller to have an aircraft follow a pre-optimized reference trajectory [14]. While the previous study used PID control as a low-level controller for the aircraft's actuators, this work designed both its low-level stability augmentation controller and the high-level autopilot system using PID control to minimize errors between the vehicle's true position and the commanded position references. However, due to time delays in the control system, the optimal trajectory was found to require manoeuvres that were too aggressive, and as such, a suboptimal solution was used instead as the reference path, further illustrating the entangled nature of trajectory planning and aircraft control. Moreover, the simple approach of exclusively using PID controllers is likely to present additional challenges in the stochastic, uncertain, and nonlinear nature of real-world environments for which PID systems are unable to compensate due to their limited operational range around a single linearization point.

Alternatively, Deittert *et al.* employed a linear quadratic regulator (LQR) optimal controller to a disturbance-injected nonlinear flight model to investigate the controller's ability to reject atmospheric turbulence and modelling errors [15]. First, state and control sequences for an optimal trajectory were passed through a nonlinear flight model containing simulated disturbances in the form of atmospheric turbulence. Then, the resulting state values were fed back and compared with the nominal trajectory from the optimizer to compute the flight path errors, which were sent through an LQR feedback controller to produce an updated control sequence corresponding to a periodic soaring solution. The LQR, based on a linearized time invariant system of the nonlinear plant, served to cancel out the trajectory tracking errors caused by the turbulence. Results of simulation testing showed that for dynamic soaring, the controller was capable of achieving mean times between failures, or time until the aircraft model's impact with the surface, in the order of dozens of minutes. However, the findings also showed that the proposed controller was unable to account for persistent stochastic disturbances, which eventually would cause incorrigible displacement from the optimal trajectory. It was reported that control schemes that track predefined trajectories are not necessarily correlated with sufficient energy management, and that future controller designs should focus on maximizing energy instead of minimizing tracking errors. The design considerations also highlighted an important dilemma of tracking control: effectuating control actions for trajectory correction due to disturbances causes additional drag and loss of energy, while restricting corrective actions makes the system less robust to the same disturbances.

The approaches examined thus far illustrate this primary disadvantage of decoupling the trajectory planning and control problems. The simplicity and convenience of trajectory track-

ing solutions are offset by their increased dependence on the path planning algorithm, and consequent sensitivity to tracking errors. Considering the practical need to account for disturbances and uncertainties, the tracking problem becomes an issue of achieving either a limited number of robust cycles or a greater number of sustainable cycles at the cost of the other. Furthermore, although systems based on *a priori* solutions take advantage of the greater availability of computational storage over computational power onboard SUAVs, such control systems are nonetheless more vulnerable to changes in the aircraft or environment, as previously discussed in the context of trajectory planning. These conclusions demonstrate the need for robust controllers that can perform over a range of conditions and environments while being capable of adjusting the flight trajectory if error correction is anticipated to irrevocably compromise the soaring cycle. It is this central requirement of robustness that has motivated research in the third category of soaring control techniques, comprised of nonlinear methods.

Contrasting the PID controller's limited operating region or the LQR's fixed time horizon, Liu *et al.* investigated a nonlinear model predictive control (NMPC) strategy for UAV thermal soaring [16]. The scheme initially generated optimal trajectories using a 3DOF flight model to reduce the large computational costs of NMPC optimization, which is a repetitive process that occurs along a receding time window. A generalized regression neural network was used to estimate the local wind field around the aircraft through onboard updraft velocity measurements, and a heuristic search technique dictated whether the aircraft would search for or circle a thermal column by taking into account the estimated thermal updraft velocity and the vehicle's altitude. In addition, a model interaction strategy allowed the resulting NMPC control inputs to the 3DOF aircraft model to be used by a more accurate 6DOF model. In all, the controller used a prediction horizon of 120 seconds to estimate the future state of the vehicle given certain control inputs, which were obtained with the goal of maximizing the amount of energy extracted at the end of the horizon. Results of a 6DOF simulation showed that the proposed strategy could sufficiently estimate the thermal wind regions in the environment using the neural network and optimize trajectories to chase moving thermals with an average computational time of less than 2 seconds. This research serves as an example of an on-line computational strategy in contrast to the *a priori*-based systems described thus far, and shows the potential applicability of neural networks in the field.

2.4 Intelligent Control

In applying the nonlinear approximation abilities of artificial neural networks trained through machine and reinforcement learning algorithms, research in autonomous soaring control has recently taken an interest in the collection of such tools and techniques known as intelligent control. These methods have become increasingly attractive in addressing the complex dynamics of nonlinear processes, for which accurate models may be difficult to create, or for creating systems with generalized behaviours, which is difficult to achieve using explicitly defined and tuned control laws. This section will examine the existing efforts in applying intelligent control schemes to dynamic soaring.

Fuzzy controllers use continuous inputs that represent degrees of truth, known as fuzzy logic, to form control protocols defined by conditional statements. Since the control rules are derived from the designer's knowledge about the desired system behaviour, this method by-

passes the need to explicitly and accurately model the plant. Taking advantage of this unique characteristic, Barate *et al.* designed a fuzzy controller inspired by the observations of albatross birds [17]. Two sets of rules were written for the ascending and descending portions of the travelling dynamic soaring cycle using aircraft altitude as the variable. The consequences of the rules, also known as the actions, were defined as linear combinations of the aircraft states, and directly correlated to control commands for the rudder, elevators, and ailerons. These rule parameters were initially empirically determined before being optimized through an evolutionary algorithm to increase robustness and efficiency, where each rule parameter was encoded in the genome of each species, or controller, of the algorithm. Simulations showed that while the resulting controller's ability to fly consecutive soaring cycles was sensitive to small changes in the rule parameters, it was able to generate travelling trajectories for narrow ranges of initial states, wind turbulence, and sensor measurement noise with the simple fuzzy rule set. Nevertheless, the findings also demonstrated the fuzzy system's high dependence on the educated initial guess and lack of generalization ability in the absence of explicitly formulated rules for robust behaviour.

In exploring generalization ability, Montella and Spletzer investigated a reinforcement learning control approach to loitering soaring [18]. The strategy was based on the model-free Q-learning algorithm, which was modified to accelerate and reduce the computational complexity of the learning process. In reinforcement learning, training data is typically collected by a learning agent that repeatedly explores the state-action space. However, since finding a successful dynamic soaring cycle by chance through random actions is both unlikely and impractical, a two stage learning process was implemented. After optimal trajectories were generated using a collocation-based nonlinear optimization technique, a teaching controller was made to steer an aircraft model along the optimal trajectories. By observing the teaching controller's actions, a learning controller passively generated a Q-function, which was approximated using a K nearest neighbours selection. In the second stage, control of the aircraft was given to the learning controller, which used the previously constructed Q-function to determine which actions from a given state allowed for the greatest reward, defined by the total kinetic and potential energies gained from completing a single loitering cycle. Results of point mass model simulations showed that a learned controller was able to fly trajectories with greater net energy gain than those of the teaching controller after just ten training flights. In addition, its ability to find novel trajectories indicated that the learned controller was not simply recreating the training trajectories. Further testing in a higher fidelity 6DOF simulator revealed that even without retraining, when not starting along the planned trajectory, the controller was able to generate new flight paths that closely resembled those of the optimal training sets, instead of sacrificing energy to steer the aircraft back onto the initially planned trajectory. This result directly addresses the primary issue with tracking controllers, showcasing the generalization ability of certain neurocontrollers.

Another subset of intelligent control is the field of neurocontrol, which is defined as the direct application of artificial neural networks in control strategies. Kim *et al.* explored the use of deep neural networks for directional dynamic soaring, where a series of DNNs were trained on a point mass UAV model [19]. To obtain the large training sets required for generating deep neural networks, 1000 optimal trajectories were first computed, each of which had different wind gradients and initial conditions. This library of trajectories was generated by solving optimal control problems that maximized the final energy, altitude, and directional velocity of a UAV travelling along the trajectory, and the dataset was used in a machine

learning algorithm to train three controller networks regulating the angle of attack, the bank angle, and a wing morphing parameter for backwards-swept configurations. These networks were then integrated into a single feedback controller. Finally, the trained neurocontrollers were tested and compared to the optimal trajectories of a test set. Simulation results for various wind and initial conditions showed that the DNN-based control scheme was able to generate commands that mimicked those of the corresponding optimal trajectories even in various wind fields and with different initial conditions, albeit sub-optimally. One significant advantage of neural network-based controllers is that once trained, obtaining the control commands is a near-instantaneous process that simply involves a feedforward pass through the network. However, one aspect of deep neural networks that may be of concern are their size. Each of the three networks of the DNN control strategy consisted of 80 neurons with over 1200 neural weights. Simpler networks would reduce training times, provide greater efficiency in calculating control commands, and allow for the interpretation of the network's topology and weights with respect to its outputs. Additionally, machine learning methods require many training samples that are representative of the environment in which the real system will fly. It is yet unclear how the cumbersome process of generating these numerically optimized training sets can be applied to account for stochastic disturbances and time-varying uncertainties, which is another crucial aspect of real-world implementation that must be addressed.

Using a different form of neurocontrol, Perez *et al.* applied the NeuroEvolution of Augmenting Topologies (NEAT) algorithm in generating dynamic soaring-capable neural networks [20]. The dynamic soaring problem was formulated as an optimization task, where the goal was to optimize the topology and weights of various neurocontrollers to eventually select the neural network that exhibited the best soaring performance. The networks accepted states of a simulated aircraft model as inputs and produced control commands as outputs to steer the vehicle. The trained neural networks were not only extremely simple, consisting of just six or seven neurons, but were also capable of providing control commands for optimal trajectories that extracted more energy per single cycle than those of numerically optimized paths. The study serves as the initial work that is expanded on in this research.

An important commonality between these neurocontrol approaches is that the trajectory planning and aircraft control problems are no longer distinct. Earlier methods of addressing these challenges relied on separate planning and control schemes, where the path planner would explicitly pass a trajectory to a high-level controller, which would then actuate the aircraft accordingly using low-level control techniques. On the other hand, the aforementioned intelligent control systems simply conveyed commands directly to low-level controllers, bypassing the trajectory planning phase as well as the disadvantages associated with tracking control.

Knowledge Gaps

Autonomous soaring research has largely focused on exploring different solutions to the problems of trajectory planning and aircraft control. Existing efforts in addressing these interconnected tasks can be categorized into trajectory-dependent approaches, which focus on simplifying the path planning system and relegating aircraft control to tracking methods, and control-dependent approaches, which instead forgo explicit trajectory planning by consign-

ing the task to the controller. Regardless of the approach, the actualization of autonomous soaring systems necessitates simple and robust solutions that can be implemented using the limited hardware onboard SUAVs while accounting for the stochastic and uncertain nature of real-world environments. While the works presented in the previous sections have explored techniques of either achieving efficient computation through various simplification methods, or of addressing the problem of generalizability using intelligent control schemes, there is a relative lack of research that directly examines robustness, and an even greater scarcity of attempts to achieve both. Based on the overview of autonomous soaring research presented in this literature review, this thesis dissertation aims to fill the identified knowledge gaps through the development of a neuroevolutionary control solution.

3 Modelling and Trajectory Optimization

This chapter presents the models used to define and characterize flight behaviour in a windy environment. The equations of motion and formulations of the aircraft and wind models shown in this section are integrated in the flight simulation that was used throughout this research. Furthermore, this chapter describes the optimal control problem framework and its specific application to the dynamic soaring case, while highlighting the challenges associated with the use of numerical optimization methods for autonomous applications to validate the case for a robust neurocontrol approach that can rapidly generate energy-neutral trajectories in various conditions. Finally, in the scope of this research, the optimal control problem is formulated as a trajectory optimization problem, and therefore, the two terms are used interchangeably.

3.1 Aircraft Characterization

The design of an autonomous soaring control system first requires a characterization of flight dynamics. In particular, both trajectory optimization methods and neural-network-based controllers require a simulated training environment with which they can find feasible flight solutions or interact under an iterative learning algorithm. Therefore, this research used a three-degree-of-freedom (3DOF) point mass model of a typical commercial SUAV, along with wind models common in the field. The 3DOF framework was selected, because its control variables were more representative of the higher-level commands that would be conveyed to an autopilot system for trajectory generation, compared to the lower-level stabilization variables that would need to be managed in a higher-fidelity six-degree-of-freedom model. This work assumes the controllability and observability of the aircraft and therefore its stability. In addition, the simpler 3DOF model allows for more rapid calculations when simulating trajectories.

3.1.1 Equations of Motion

The equations of motion of the 3DOF point-mass aircraft model are formulated for a Cartesian coordinate system, where the aircraft's motion is considered relative to an inertial frame located on the Earth's surface. Figure 3.1 depicts a free body diagram of the model which indicates, along with the forces acting on the aircraft, the positive directions for roll and pitch. The equations account for wind in the x, y, and z directions for various types of soaring, and are defined by the airspeed V, lift L, drag D, mass m, standard gravity g, pitch γ, heading ψ, roll μ, coordinate positions x, y, and h, the various wind components W, as well as their rates of change $\dot{W}$ [8]:

$$\dot{V} = -\frac{D}{m} - g\sin(\gamma) - \dot{W}_x\cos(\gamma)\sin(\psi) - \dot{W}_y\cos(\gamma)\cos(\psi) \tag{3.1}$$

$$\dot{\psi} = \frac{L\sin(\mu)}{Vm\cos(\gamma)} - \frac{\dot{W}_x\cos(\psi)}{V\cos(\gamma)} - \frac{\dot{W}_y\sin(\psi)}{V\cos(\gamma)} \tag{3.2}$$

$$\dot{\gamma} = \frac{1}{V}\left(\frac{L\cos(\mu)}{m} - g\cos(\gamma) + \dot{W}_x\sin(\gamma)\sin(\psi) + \dot{W}_y\sin(\gamma)\cos(\psi)\right) \tag{3.3}$$

$$\dot{x} = V\cos(\gamma)\sin(\psi) + W_x \tag{3.4}$$

$$\dot{y} = V\cos(\gamma)\cos(\psi) + W_y \tag{3.5}$$

$$\dot{h} = V\sin(\gamma) \tag{3.6}$$

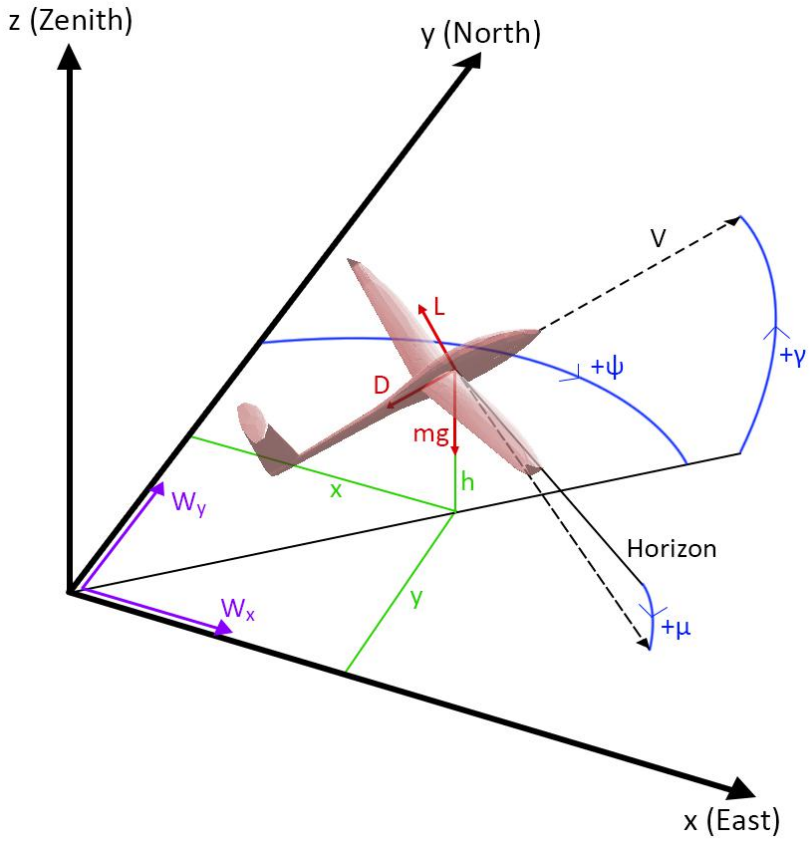


Figure 3.1: Free body diagram of SUAV model.

The equations for lift, drag, and drag coefficient C_D are shown below. The drag coefficient is modelled as the sum of both parasitic and lift-induced drag, with local air density ρ, wing reference area S, lift coefficient C_L, zero-lift drag coefficient C_{D0}, induced drag factor K, and maximum lift-to-drag ratio E_{max}:

$$L = \frac{1}{2}\rho V^2 S C_L \tag{3.7}$$

$$D = \frac{1}{2}\rho V^2 S C_D \tag{3.8}$$

$$C_D = C_{D0} + K C_L^2 \tag{3.9}$$

$$K = \frac{1}{4 C_{D0} E_{max}^2} \tag{3.10}$$

Additionally, the total energy e_T of the aircraft is comprised of both the kinetic e_K and potential e_P energies:

$$e_K = \frac{1}{2} m V^2 \tag{3.11}$$

$$e_P = mgh \tag{3.12}$$

$$e_T = e_K + e_P \tag{3.13}$$

Another potential consideration would be the internal energy of the aircraft, such as the energy stored in onboard batteries and used for control actuation. However, the target platform for implementing autonomous soaring primarily consists of SUAVs, whose control surfaces are actuated by servo motors that are governed by constant-voltage, continuous pulse-width-modulated signals. Therefore, there is no appreciable increase in internal energy expenditure associated with a greater frequency of control actions, and as such, the calculation of control effort is not considered in this work.

Regardless, this mathematical model of the aircraft describes the dynamics of the simulation, which can be accessed by a control agent through the following state $\mathbf{x}$ and control $\mathbf{u}$ variables:

$$\mathbf{x} = \begin{bmatrix} V \\ \psi \\ \gamma \\ h \\ \dot{h} \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} C_L \\ \mu \end{bmatrix} \tag{3.14}$$

The airspeed can be estimated through a pitot-static system, the heading and pitch angles calculated from gyroscopes, and the aircraft's height and its rate of change can be obtained through an altimeter and variometer, respectively. These inputs were selected specifically because their values are typically accessible using onboard sensors, which is necessary for the high-frequency control required to perform soaring. Additionally, the lift coefficient and roll were used as the control variables due to their higher, navigation-level nature, which is more appropriate for tasks that focus on vehicle attitude control rather than the precise, but lower-level actuators commonly associated with aircraft stabilization. In particular, the lift coefficient can be related to the angle of attack through the aircraft's lift curve polar. Furthermore, the lift coefficient is used over the angle of attack, because the 3DOF system does not model the complete dynamics of the vehicle, and as a result, the former is more appropriate as an instantaneously-changing state. Directly commanding a lift coefficient through the presumed actuating of an aircraft's control surfaces is more accurate than an instantaneous change in its angle of attack.

3.1.2 3DOF Aircraft Model

The specific SUAV model used in the 3DOF simulator was based off RnR Products' SBXC cross country sailplane and the model used in Zhao's seminal work on SUAV dynamic soaring [8]. However, computing a flight trajectory either explicitly through trajectory optimization or implicitly using neural network training algorithms requires limiting the states and controls of the vehicle. As a result, the minimum airspeed was calculated to be equal to the stall velocity. From the definition of lift (Eqn. 3.7):

$$L = \frac{1}{2}\rho V^2 S C_L \tag{3.15}$$

$$V = \sqrt{\frac{2L}{\rho S C_L}} \tag{3.16}$$

In level flight, the lift force balances the weight that is acting downwards on the aircraft. Therefore, to obtain the minimum speed at which an aircraft of a fixed weight and size at a given altitude can remain in level flight, the lift coefficient must be maximized:

$$V = \sqrt{\frac{2mg}{\rho S C_L}} \tag{3.17}$$

$$\begin{aligned} V_{min} &= \sqrt{\frac{2mg}{\rho S C_{L_{max}}}} \\ &= V_{stall} \end{aligned} \tag{3.18}$$

Additionally, instead of an arbitrary initial airspeed for soaring trajectories, the best glide speed was used. This value is the airspeed at which the lift-to-drag ratio is maximized, and can be computed using the definitions of the drag coefficient (Eqn. 3.9):

$$\frac{L}{D} = \frac{C_L}{C_D} \tag{3.19}$$

$$= \frac{C_L}{C_{D0} + K C_L^2} \tag{3.20}$$

The maximum of this ratio is obtained by taking the root of the first derivative with respect to C_L:

$$\begin{aligned} \frac{d}{dC_L}\left(\frac{C_L}{C_D}\right) &= \frac{C_{D0} + K C_L^2 - C_L(2KC_L)}{(C_{D0} + K C_L^2)^2} \\ &= 0 \end{aligned} \tag{3.21}$$

$$C_{D0} + K C_L^2 - C_L(2KC_L) = 0 \tag{3.22}$$

$$C_{D0} - K C_L^2 = 0 \tag{3.23}$$

$$C_L = \sqrt{\frac{C_{D0}}{K}} \tag{3.24}$$

Combining Equations 3.17 and 3.24 yields the following, where the airspeed that maximizes the lift-to-drag ratio is also known as the minimum drag speed:

$$V_{md} = \sqrt{\frac{2mg}{\rho S\sqrt{\frac{C_{D0}}{K}}}} \tag{3.25}$$

Table 3.1 details the specifications of the glider. This model is representative of the small, lightweight, manually-launched aircraft for which the implementation of autonomous soaring technology is intended.

Table 3.1: SUAV characteristics.

Parameter	Symbol	Value	Units
Wing area	S	10.7	sq ft
Mass	M	0.295	slug
Maximum lift-to-drag ratio	E_{max}	20.0	-
Maximum lift coefficient	$C_{L_{max}}$	1.5	-
Minimum lift coefficient	$C_{L_{min}}$	-0.2	-
Zero-lift drag coefficient	C_{D_0}	0.025	-
Maximum load factor	n_{max}	5	-
Maximum absolute bank angle	$\|\mu\|_{max}$	60	deg
Stall speed	V_{stall}	22.3	ft/s
Minimum drag speed	V_{md}	27.3	ft/s

3.2 Surface Wind Shear Model

Dynamic soaring requires the presence of a vertical wind gradient, or wind shear. One of the most common models, initially formulated by Zhao [8], assumes a horizontal wind whose strength is a function of altitude and whose shape is dependent on several shape parameters. As shown in Figure 3.2, the model takes a logarithmic shape according to the shape parameter A when $0 \leq A < 1$, a linear profile when $A = 1$, and an exponential form when $1 < A \leq 2$. Additionally, the transition height h_{tr} stretches or compresses the altitude range within which the gradient exists, and the maximum wind strength W_{max} determines the overall magnitude of the profile. The wind strength and bearing are assumed to be uniform over the ground surface.

$$W_x = \begin{cases} \beta\left[Ah + \frac{1-A}{h_{tr}}h^2\right] & h < h_{tr} \\ W_{max} & h \geq h_{tr} \end{cases} \tag{3.26}$$

$$\beta = \frac{W_{max}}{h_{tr}} \tag{3.27}$$

The rates of change of the wind profile used in the aircraft model's equations of motion are dependent on the existence of wind components in the respective axes. For instance, a purely eastward wind would have the following gradient:

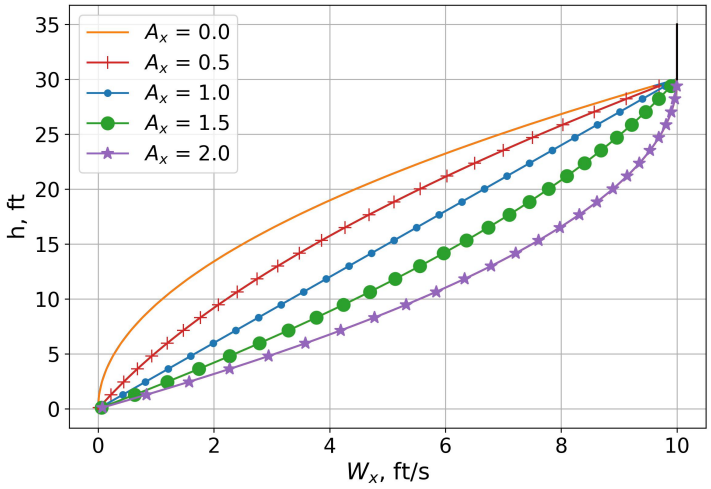


Figure 3.2: Wind profile shapes.

$$\dot{W}_x = \beta_x \left[A_x + 2\frac{1 - A_x}{h_{tr_x}} h \right] \dot{h} \tag{3.28}$$

$$\dot{W}_y = 0.0 \tag{3.29}$$

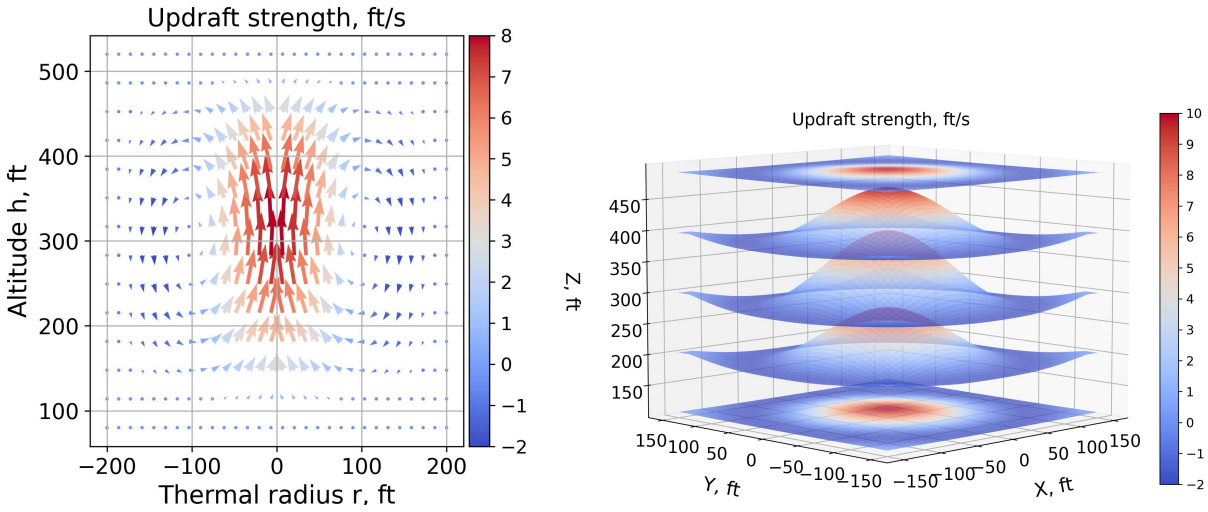
$$\dot{W}_z = 0.0 \tag{3.30}$$

3.3 Thermal Models

3.3.1 Thermal Bubble

Thermal soaring involves circling an updraft region to gain altitude before eventually exchanging the potential energy for kinetic energy by soaring out of the thermal once the updraft fades. A common model for thermals is the toroidal bubble model described by Lawrance and Sukkarieh [21], which is characterized by a core updraft region surrounded by sinking air, disconnected from the ground and moving upwards over time. This representation is visualized in Figure 3.3.

The model, minimally altered from the original formulation to fit the coordinate system presented in Section 3.1.1, is defined by the agent's distance from the thermal centre r, the maximum core updraft wind speed w_{core}, the thermal's height h_t, the thermal radius over the xy plane r_{xy}, the radius over the z axis r_z, the bubble eccentricity factor k, and the rising



(a) Vector field

(b) Vertical wind speed cross-sections

Figure 3.3: Toroidal thermal bubble model.

velocity $\dot{h_t}$, where x, y, and h are the agent's location in the inertial frame:

$$r = \sqrt{x^2 + y^2}, \qquad k = \frac{r_z}{r_{xy}} \tag{3.31}$$

$$W_z = \begin{cases} w_{core} & r = 0,\ h \in [h_t - r_z, h_t + r_z] \\ \cos(\frac{2\pi}{4r_z}(h - h_t))\frac{r_{xy} w_{core}}{\pi r} \sin(\frac{\pi r}{r_{xy}}) & r \in (0, 2r_{xy}],\ h \in [h_t - r_z, h_t + r_z] \end{cases} \tag{3.32}$$

$$W_x = -Wz\frac{h - h_t}{(r - r_{xy})k^2}\frac{x}{r} \tag{3.33}$$

$$W_y = -Wz\frac{h - h_t}{(r - r_{xy})k^2}\frac{y}{r} \tag{3.34}$$

The rates of change of the wind components were calculated using the finite difference approximation method for simplicity. In this model, the bubble rises from an initial height h_{t_0} at a speed $\dot{h_t}$, which allows a flight agent to continually gain potential energy as long as it remains within the thermal region.

3.3.2 Thermal Column

Contrarily, studies on boundary layer meteorology suggest that thermals are less like bubbles and instead better represented as finite-length-and-duration columns [22]. These empirical observations have prompted the design of a column-based thermal model, which exists below a maximum altitude $h_{tr_{max}}$ and persists for a maximum duration $t_{th_{max}}$:

$$r = \sqrt{x^2 + y^2} \tag{3.35}$$

$$Wz = \begin{cases} w_{core} & r = 0,\ h < h_{tr_{max}},\ t < t_{th_{max}} \\ \frac{r_{xy} w_{core}}{\pi r} \sin(\frac{\pi r}{r_{xy}}) & r \in (0, 2r_{xy}],\ h < h_{tr_{max}},\ t < t_{th_{max}} \\ 0.0 & \text{otherwise} \end{cases} \tag{3.36}$$

$$W_x = 0.0 \tag{3.37}$$

$$W_y = 0.0 \tag{3.38}$$

This model, illustrated in Fig. 3.4, is a simpler representation of thermal winds that does not require a flight agent to time its entrance into the thermal region as would be the case with the bubble model.

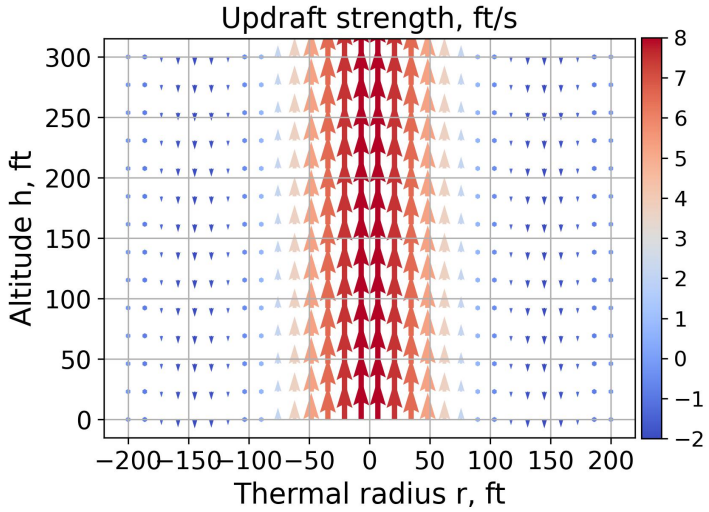


Figure 3.4: Column thermal model.

3.4 Ridge Wind Shear Model

Ridge or orographic soaring involves using the winds around geographic features such as ridges and hills. Airflow becomes compressed as it approaches a hill to form a region of higher pressure, which results in a wind velocity gradient that increases in magnitude proportionally to the altitude. This wind field can then be exploited by a flight agent to continuously loiter on the windward side of the hill.

The model for the ridge wind shear [23] is based on a semi-cylindrical feature spanning east to west with a radius R and length l, and is a function of the infinitely-far horizontal wind speed w_∞ also referred to as the freestream wind, the distance from the centre of the hill to the aircraft r, and the angle between the horizontal plane and the aircraft θ. The freestream wind is represented by a logarithmic profile as a function of the transition height $\hat{h}$ of the boundary layer $W_{\hat{h}}$, the vehicle's altitude h, and the shear height nearest to the terrain z_0. The flow vectors are illustrated in Fig. 3.5.

$$W_\infty = \frac{W_{\hat{h}} ln(\frac{h}{z_0})}{ln(\frac{\hat{h}}{z_0})} \tag{3.39}$$

$$W_y = w_\infty + w_\infty \frac{R^2}{r^2}(\sin^2\theta - \cos^2\theta) \tag{3.40}$$

$$W_z = 2w_\infty \frac{R^2}{r^2} \sin\theta\cos\theta \tag{3.41}$$

If the aircraft flew past the length of the ridge, it would experience an exponentially decaying wind profile, where the wind components W_y and W_z would be computed as per Eqn. 3.39-3.41 before being multiplied by a decay factor k_d:

$$W_y = k_d W_y \tag{3.42}$$

$$W_z = k_d W_z \tag{3.43}$$

$$k_d = 1.05^{-\frac{(|x|-l)}{2}} \tag{3.44}$$

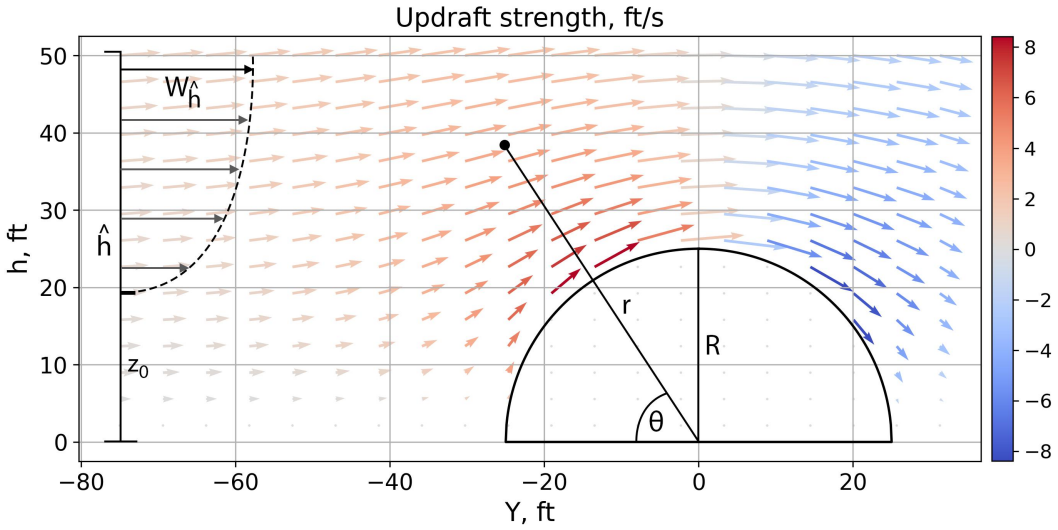


Figure 3.5: Ridge shear wind model.

3.5 Formulating the Optimal Control Problem

One of the most common methods of solving optimal control problems is by transforming it into a nonlinear programming problem, which involves finding a set of values that optimize an objective function under a set of constraints. The problem of soaring is an energy extraction task, in which the objective is to maximize the kinetic and potential energies obtained through environmental wind phenomena. This quantifiable goal allows for the formulation of a trajectory optimization problem for the solving of energy-optimal soaring trajectories. In the general case, trajectory optimization involves determining the control inputs $\mathbf{u}(t)$, the initial time t_0, and the final time t_f that minimizes the cost function J along the states $\mathbf{x}(t)$, with boundary conditions Φ and Lagrange term performance index $\mathcal{L}$, all summarized as [24]:

$$J = \Phi\left[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f\right] + \int_{t_0}^{t_f} \mathcal{L}\left[\mathbf{x}(t), \mathbf{u}(t), t\right] dt \tag{3.45}$$

The cost function is minimized with respect to the control inputs within the system defined by a set of ordinary differential equations in the form of the state equations. Furthermore, while the cost function directs the optimization process, it is also necessary to impose certain rules, such that the solution is within the domain of physically realizable solutions. Therefore, the formulation of an optimal control problem requires a set of differential equations $\dot{\mathbf{x}}$ that represent the dynamic constraints of the system, path constraints C that enforce restrictions along the trajectory, and boundary constraints ϕ, which limit the initial and final system states:

$$\dot{\mathbf{x}}(t) = \mathrm{f}\left[\mathbf{x}(t), \mathbf{u}(t), t\right] \tag{3.46}$$
$$C\left[\mathbf{x}(t), \mathbf{u}(t)\right] \leq 0 \tag{3.47}$$
$$\phi_{min} \leq \phi\left[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f\right] \leq \phi_{max} \tag{3.48}$$

Path constraints, or inequalities, ensure that states and controls stay within defined limits in a way that reflects a physical system. For soaring, constraints would include restrictions on altitude h, structural limits such as the airframe load factor n, and aircraft control limits for any control variables such as the lift coefficient C_L and the roll angle μ:

$$h(t) \geq 0 \tag{3.49}$$
$$n(t) = \frac{L}{mg} \leq n_{max} \tag{3.50}$$
$$C_{L_{min}} \leq C_L(t) \leq C_{L_{max}} \tag{3.51}$$
$$-\mu_{max} \leq \mu(t) \leq \mu_{max} \tag{3.52}$$

Boundary conditions, or equality constraints, are also specified to define the initial and final states of the trajectory at initial and final times t_0 and t_f, and are necessary to distinguish between different soaring techniques. For instance, the different set of boundary conditions for loitering and travelling soaring are shown below, where the former requires the aircraft to return to its initial spatial position, while the latter requires a displacement along a certain direction:

$$\begin{array}{lll} \text{Loitering:} & V(t_0) = V_0 & V(t_f) = V_f \\ & \psi(t_0) = \psi_0 & \psi(t_f) = \psi_f \\ & \gamma(t_0) = \gamma_0 & \gamma(t_f) = \gamma_f \\ & \vdots & \vdots \\ & x(t_0) = x_0 & x(t_f) = x_0 \\ & y(t_0) = y_0 & y(t_f) = y_0 \end{array}$$

$$
\begin{aligned}
\text{Travelling:} \qquad & V(t_0) = V_0 V(t_f) = V_f \\
& \psi(t_0) = \psi_0 \psi(t_f) = \psi_f \\
& \gamma(t_0) = \gamma_0 \ \gamma(t_f) = \gamma_f \\
& \vdots \qquad\qquad \vdots \\
& x(t_0) = x_0 \ x(t_f) = x_f \\
& y(t_0) = y_0 \ y(t_f) = y_f
\end{aligned}
$$

Together, these elements define the trajectory optimization problem, which can be solved through various techniques.

3.6 Solving the Optimal Control Problem

The two main categories of solving optimal control problems consist of direct and indirect methods. Direct methods use a sequence of NLP points $y_1, y_2, \ldots, y_f$ that minimize the objective function, where the NLP variable $\mathbf{y} = (\mathbf{u}_1, \mathbf{x}_2, \mathbf{u}_2, \mathbf{x}_3, \ldots, \mathbf{x}_M, \mathbf{u}_M)$. Contrarily, indirect methods find the point y that satisfies the necessary condition $F'(y) = 0$, where the derivative of the objective function is near zero [25]. Therefore, the direct method's relative simplicity of only requiring evaluations of the objective function and not its derivative makes it more practical for the dynamic soaring optimal control problem.

This section describes the trapezoidal collocation approach. The continuous-time soaring task is transcribed into a nonlinear programming problem by dividing the fixed time interval $[t_0, t_f]$ of the expected solution into N discrete subintervals and $M = N+1$ nodes or collocation points [25]:

$$t_0 < t_1 < t_2 < ... < t_k < ... < t_M \tag{3.53}$$

$$t_M = t_f \tag{3.54}$$

The direct collocation approach parameterizes the state and control trajectories using functions, which are usually select-order polynomials. In order to ensure continuity in the states between collocation points, a set of defect constraints ξ must be satisfied, which for the trapezoidal method are as follows, with state at node k as x_k, duration of subinterval h_k, and system dynamics f_k:

$$\xi = x_{k+1} - x_k - \frac{h_k}{2}(f_k + f_{k+1}) = 0 \tag{3.55}$$

$$h_k = t_{k+1} - t_k \tag{3.56}$$

Therefore, the trapezoidal collocation process involves finding a trajectory that ensures that defects are zero. The optimal control problem is reformulated as a nonlinear programming problem, which is summarized as follows, dependent also on path and boundary constraints C and Φ:

$$\text{minimize} \quad J = \int_{t_0}^{t_k} \mathcal{L}\left[\mathbf{x}(t_k), \mathbf{u}(t_k), t_k\right] dt \tag{3.57}$$

$$\text{with respect to} \quad \mathbf{x}(t_k), \mathbf{u}(t_k), k = 0, 1, ...N \tag{3.58}$$

$$\text{subject to} \quad \zeta_k = y_{k+1} - y_k = \frac{h_k}{2}\left[f_{k+1} + f_k\right] = 0 \tag{3.59}$$

$$C[\mathbf{x}(t_k), \mathbf{u}(t_k), t_k] \leq 0 \tag{3.60}$$

$$\Phi[\mathbf{x}(t_0), t_0, \mathbf{x}(t_N), t_N] = 0 \tag{3.61}$$

3.7 Trajectory Optimization for Dynamic Soaring

To understand the nature of trajectory-optimization-based path solutions, this section presents a canonical example of an optimal control solution for dynamic soaring. Since dynamic soaring is the more challenging technique when compared to thermal soaring due to the rapid and precise control manoeuvres required for the former, its study in this section better showcases the weaknesses of numerical trajectory optimization for path planning.

Sachs used commercial optimization software to compute the optimal energy-neutral trajectory requiring minimum shear wind strength, where the lift coefficient was kept constant throughout the entirety of the flight [1]. Therefore, this configuration implies a constant angle of attack, which was used to simplify the overall control problem. The resulting flight path is shown in Fig. 3.6, along with its trace in Fig. 3.7.

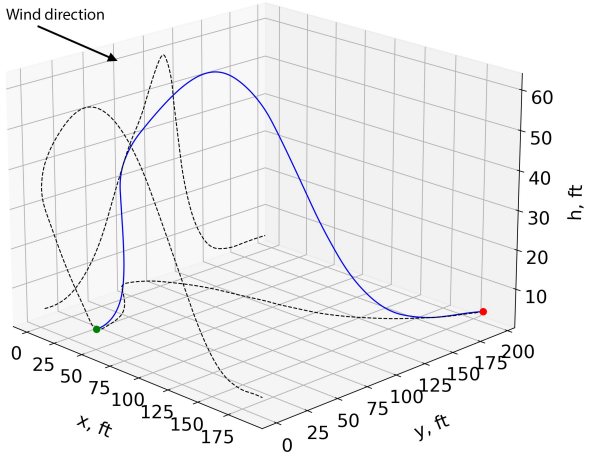


Figure 3.6: Flight path of dynamic soaring trajectory optimization, with start marker (green), end marker (red), path (blue), and axis projections (dashed) [1].

The trajectory reveals that the aircraft, whose parameters remain unknown, heads into the easterly (+X) wind to gain potential energy before conducting the high-altitude turn and descending the wind gradient to regain airspeed. Furthermore, the airspeed and height plots indicate that the vehicle did not lose any significant amount of total energy. However, one of the main challenges with tracking such a trajectory on a real aircraft is that the optimized solution typically only accounts for a single cycle. Therefore, it is necessary that the boundary constraints at the start and end of the path are equal, if the solution were to be reused for subsequent cycles. Even so, the more significant issue is that the vehicle may deviate from the

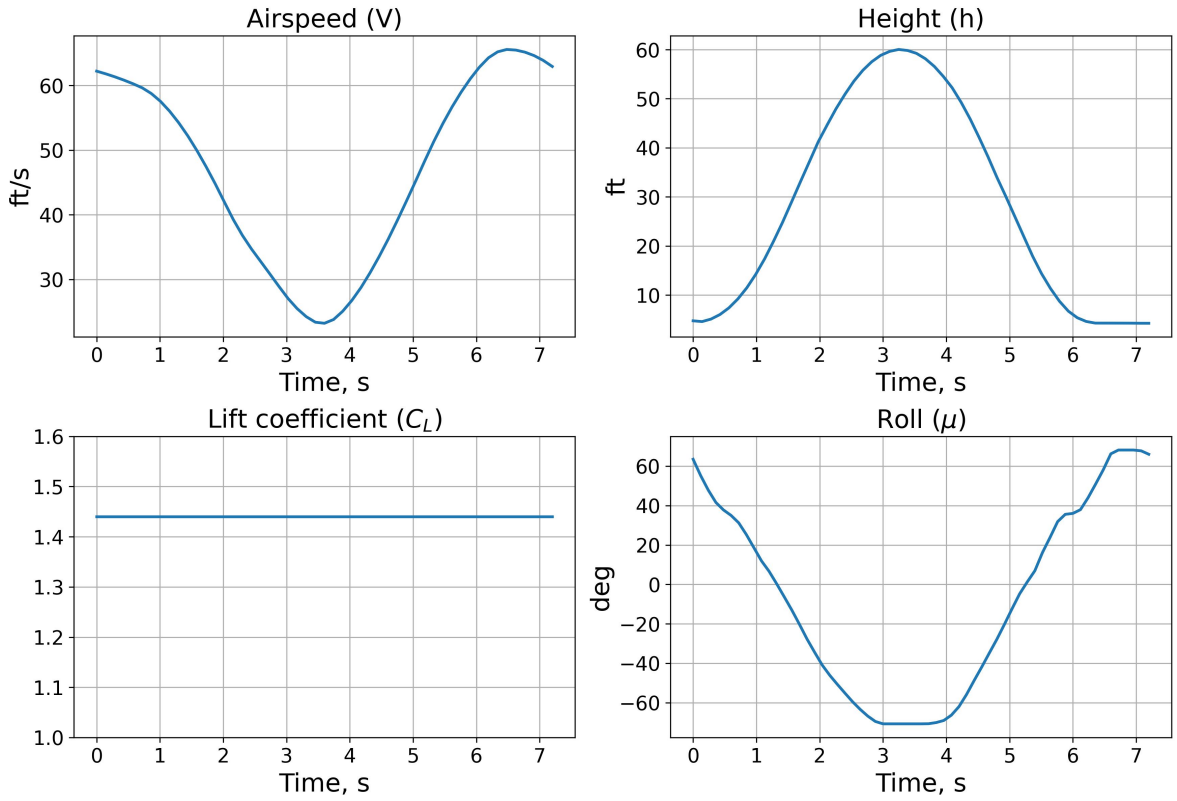


Figure 3.7: Trace of dynamic soaring trajectory optimization [1].

trajectory. One consequence of this event would be that the same trajectory is no longer able to be recycled as the aircraft's states become misaligned with those of the reference path. Optimal trajectories are directly dependent on the initial guess provided to the optimizer, and therefore, are local solutions that can become obsolete if the vehicle sufficiently diverges from the path. Furthermore, calculating optimal trajectories is a computationally-intensive process that must be completed prior to flight, while computing longer trajectories comprised of multiple soaring cycles is significantly more costly in terms of computation time and more difficult to achieve, even with specifically-engineered objective functions.

Another scenario is that while correcting the tracking errors, either the energy state of the vehicle decreases as it takes a less efficient path, or the vehicle is forced to expend internal energy reserves by using thrust. In any case, consistently drifting from the flight path would result in a gradual loss of soaring ability as additional energy is consumed for course correction, which is necessary despite the consequent impact to the effectiveness of soaring manoeuvres, because new optimal trajectories cannot be computed in real time. As a result, the local nature of optimal control trajectories along with their computational requirements present obstacles to the implementation of such schemes for real-world environments that contain uncertainties and stochastic disturbances. Since it is impractical to preemptively calculate and store a trajectory for every combination of potential initial conditions or time-varying parameters, the design of a successful autonomous soaring system requires the consideration of robust control schemes.

4 Neurocontrol

This chapter presents the neural-network-and-artificial-intelligence-based methods that have recently gained interest in the field of autonomous soaring control, before introducing the novel neuroevolutionary approach that represents the main effort of this research. The applicability of popular machine learning techniques to the soaring problem are discussed, and the NEAT algorithm is described in detail, along with the advantages that allow for its use in the neurocontroller-generation scheme. Multiple case studies demonstrate the applicability of the developed method for control problems and its advantages over other techniques. This section also compares the trajectories resulting from evolved neural networks against canonical, numerically-optimized flight paths to validate the neurocontrol behaviour.

4.1 Machine Learning

Although trajectory optimization techniques provide numerically optimal solutions, the resulting flight paths are not always practical in suboptimal real-world environments. Modern advancements in artificial intelligence research have encouraged the exploration of controllers that are potentially capable of exhibiting more generalized behaviour without the computational burden of storing and searching through enormous datasets of pre-optimized trajectories. For instance, the field of dynamic soaring has already seen the application [18, 19, 26, 27] of one of the most common disciplines of artificial intelligence, known as reinforcement learning.

4.1.1 Reinforcement Learning

Reinforcement learning involves three major components: an agent, an environment, and rewards. The agent is the decision-maker that produces the action after receiving information on the situation, or the state of a system, similar to a control scheme. The way in which the agent selects actions is through a set of rules known as the policy π, which estimates the value V of taking a specific action from a certain state. Learning the values of all the possible state-action pairs allows the comparison of multiple actions so that the agent can select the action with the greatest value. The process of machine learning involves updating the policy through an algorithm to produce the optimal policy π^*, which maximizes the value across all states.

$$\pi : S \rightarrow A \tag{4.1}$$

The environment is the dynamical system that is used to generate the data used for learning. It accepts actions as inputs and outputs state values, which are fed-back to the agent

so that it can select the next action. Repeatedly interacting with the environment allows an agent to learn its policy. For the case of autonomous soaring, the costly nature of learning on physical systems through trial and error requires a simulation-based training approach, so the environment takes the form of a virtual simulation, which in this work is defined by the dynamics model described in Section 3.1.1.

Rewards are the feedback signals that indicate how good it was for the agent to take a specific action from a particular state. The numerical reward value of every decision is computed through a reward function, which is analogous to the cost function of optimal control theory. Learning algorithms use the reward history of an actor's sequence of actions to calculate state-action values and subsequently update the policy. Therefore, the reward function is designed to encourage and discourage certain behaviours.

These elements are components in all reinforcement learning algorithms. However, one important consideration of applied machine learning is the way in which the policy is encoded. To observe relatively high-level, generalized behaviour, mapping a near-infinite number of state-action pairs and storing their values would be impractical. Therefore, the policy is often in the form of an artificial neural network, a type of function approximator. Specifically, deep neural networks are favoured for their improved approximation abilities, which make them more appropriate for learning relatively complex behaviours.

4.1.2 Artificial Neural Networks

The typical feedforward artificial neural network as shown by Fig. 4.1 is defined by an input layer, a hidden layer, and an output layer, all consisting of nodes and connections defined by numerical values. The feedforward process multiplies the values x_i at each input node or unit with the weights of all outbound connections w_ij, adds the node's bias value b_j, and passes the result h_j for hidden layers or y_k for the output layer through a activation function such as the sigmoid function $\sigma(z)$ of Fig. 4.2, the nonlinear nature of which allows ANNs to approximate functions:

$$h_j = \sigma\left(\left[\sum_{i=1}^{5} x_i w_{ij}\right] + b_j\right) \tag{4.2}$$

$$y_k = \sigma\left(\left[\sum_{j=1}^{3} h_j w_{jk}\right] + b_k\right) \tag{4.3}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{4.4}$$

Modern computational advancements have improved the accessibility of ANNs with multiple hidden layers, referred to as deep neural networks. The additional depth allows for more complex policies, but also extends the duration of the learning process. Regardless, the use of DNNs in autonomous soaring control systems suffers from two main problems. First, neural networks with many hidden layers each with a large number of nodes can pose computational challenges, particularly when considering that SUAVs are often significantly hardware-limited. Assuming that such networks can be stored in onboard memory, there will be an inevitable delay in exciting the network that is proportional to its depth. This time lag can reduce the

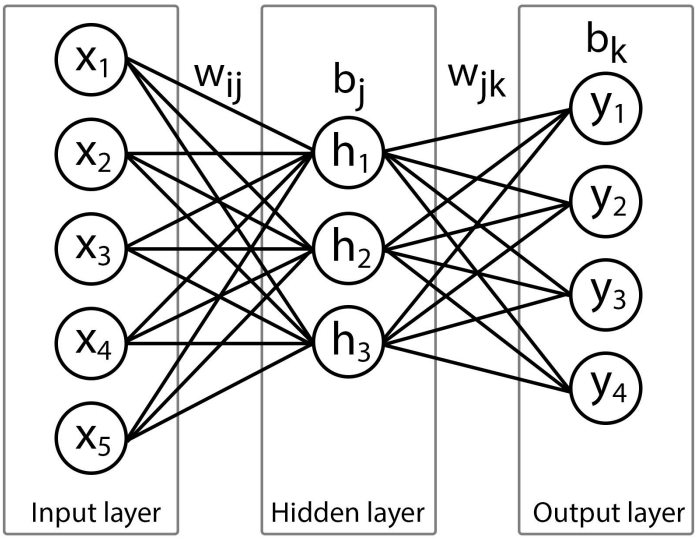


Figure 4.1: Structure of a feedforward ANN.

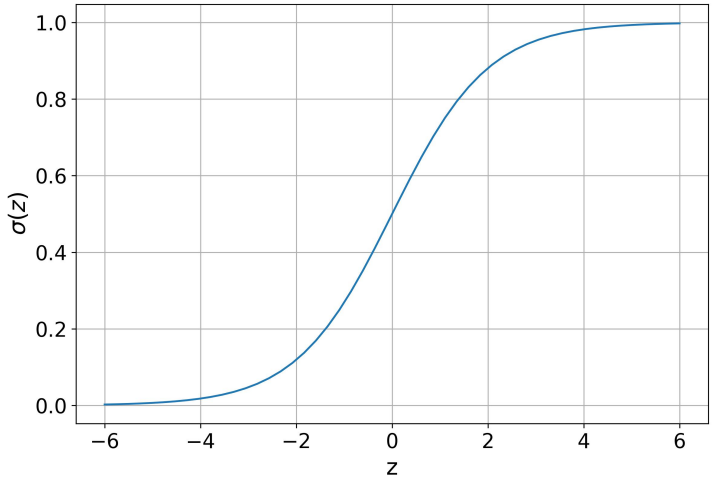


Figure 4.2: Sigmoid activation function.

control action frequency to levels insufficient for the rapid and precise commands required for an energy-sensitive task such as soaring. However, even if computational challenges are overcome, the second issue is that the black-box nature of deep neural networks hinders their interpretability. Their innate complexity and the consequent lack of transparency in the feed-forward process make it difficult to trace and predict the signals that dictate the emergent behaviour, which presents obstacles in the testing, regulation, and certification of aerial vehicles. For these reasons, this research develops and uses an evolutionary-algorithm-based approach to generate simple neural networks that can exhibit autonomous soaring behaviour.

4.2 NeuroEvolution of Augmenting Topologies

4.2.1 NEAT Algorithm

The NeuroEvolution of Augmenting Topologies algorithm developed by Stanley and Miikkulainen [28] is an evolutionary algorithm that operates on the principle of Darwinian fitness, a measure of how well an individual is able to propagate itself through successive generations. It relies on an encoding scheme that allows neural networks to be represented in a way that is analogous to genetic data. Specifically, the weights and connections of a given network are stored as a linear sequence of data, the order of which corresponds to the point in time in the evolutionary process when the feature was created. This encoding of the genome is shown in Fig. 4.3.

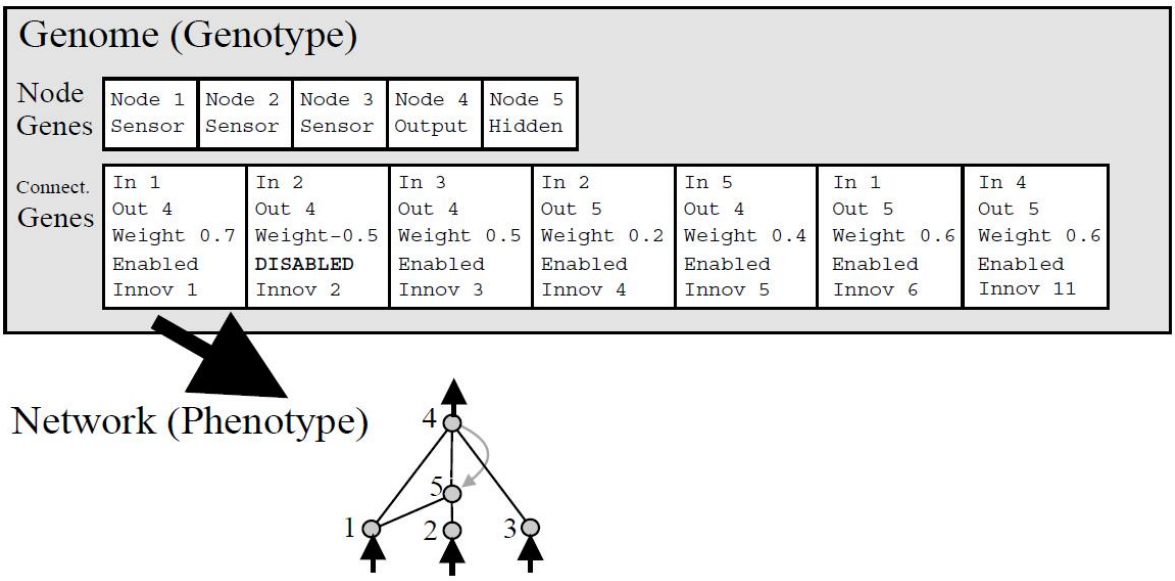


Figure 4.3: NEAT encoding [28].

The nodes, or neural network units, are identified by numbers and are categorized as sensor or input units, hidden units, and output units. Similarly, each encoding of the network connections indicates which two nodes are connected, the connection weight, whether the link is enabled or disabled, and the innovation number, which marks the point in the evolutionary sequence when the connection was formed. This representation of a neural network allows for their evolution into more complex structures that exhibit novel behaviours. Nevertheless, understanding the NEAT algorithm is important to realize its applicability to the autonomous soaring problem.

NEAT first creates an initial population of members, or neurocontrollers, each with randomly generated features consisting of nodes and connections. Members that share similar network shapes are grouped together into subcategories classified as species, which allows for the nondestructive optimizing of the node and connection weights. For instance, it is highly unlikely that a random mutation that results in a new connection would immediately increase a member's fitness, since the weight associated with the connection would not have yet undergone tuning, or the optimization process. By only comparing a member against other members within its own species through this mechanism known as speciation, innova-

tive mutations have a chance to mature. Nevertheless, the features of each initial network are dictated by random number generators to promote a diverse starting population.

After initialization, the performance of every member with respect to the objective is evaluated by allowing the neural network to interact with the environment. The output of this evaluation step is a fitness value that is assigned to the member, which is used to rank the networks by how successful each member was at the task. Members with the lowest fitness scores are eliminated from the gene pool, and those that remain undergo reproduction or are mutated. Offspring arise when two genotypes, or parent networks, are spliced and concatenated in a process known as crossover at randomly determined points along each genotype, with any genes that do not undergo crossover being inherited from the parent with the higher fitness value. This operation merges two different neural networks to create a potentially better-performing network that contains characteristics from both of its parents. Once offspring are generated, mutations have a chance to occur. Mutation processes encompass various different operations, including the random addition and subtraction of nodes and connections, all of which are made possible by the genomic history encoded in each genotype through the innovation numbers. The crossover and mutation operations allow for the creation of new, unique neural networks while persevering the genes or characteristics that contribute to high performance. Lastly, competition between species is addressed through stagnation, which is when the highest fitness achieved within a specie has not improved after a certain number of generations, where a generation is defined as one cycle of the evaluation and reproduction steps. Such species are marked as stagnant and are prohibited from reproducing further, eventually becoming extinct through the generational culling process. Yet, to prevent the complete extinction of all species, an elitism mechanism preserves a small number of the highest-performing species, regardless of whether they have stagnated. This entire process is managed by the NEAT algorithm, which populates each generation, evaluates every member, and outputs the best-performing neural network after a fitness threshold is reached or a certain number of generations has elapsed.

4.2.2 NEAT Control

To initially verify the feasibility and potential of applying NEAT-evolved neural networks to physical control problems, the algorithm was tested on the canonical cart-pole balancing task and compared with the results of the actor-critic reinforcement learning algorithm. Initially, a simulation environment was created for an inverted pendulum system on a track, which approximates the cart-pole system and is depicted in Fig. 4.4, where the objective of the control agent is to keep the pole balanced at the unstable equilibrium position by applying horizontal forces.

The system was modelled using the following equations of motion [29], which included the mass of the cart M, the mass of the pole m, half of the pole length l, the cart's lateral position x, the pendulum's angle θ with respect to the vertical axis, the force f applied to the cart, the pendulum's mass moment of inertia I, and standard gravity g:

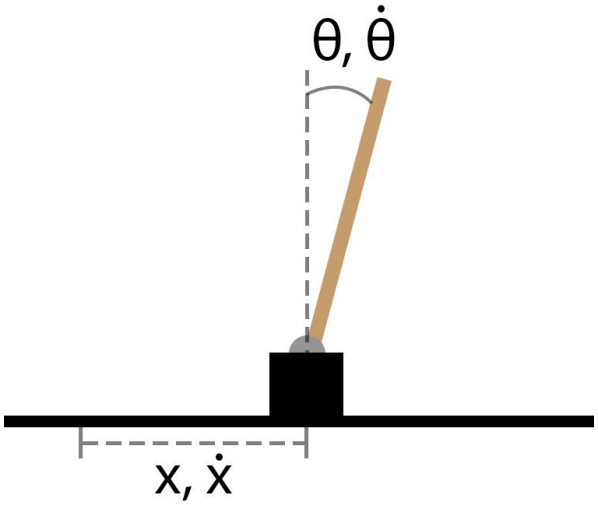


Figure 4.4: Cart-pole balancing model.

$$\ddot{\theta} = \frac{g \sin\theta + \cos\theta \left(\frac{-f - ml\dot{\theta}^2 \sin\theta}{M+m} \right)}{l \left(\frac{4}{3} - \frac{m\cos^2\theta}{M+m} \right)} \tag{4.5}$$

$$\ddot{x} = \frac{F + ml(\dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta)}{M+m} \tag{4.6}$$

The system's states $\mathbf{x}$ and controls $\mathbf{u}$ are indicated below, where at every timestep of the simulation, the control agent outputs the external force that is exerted on the cart:

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} f \end{bmatrix} \tag{4.7}$$

For both the RL and NEAT implementations, each simulation or training episode terminated when the cart and pole exceeded position and angle limits, and the algorithm stopped after encountering the first simulation in which the system remained within bounds for the full simulation time of 30.0s. Furthermore, continuous state and action spaces were used, and the reward was computed to be the duration of each simulation, thereby encouraging behaviour that maintained the pole's upright position.

After only 13 generations of evolution, taking a CPU time of 1.91s, the NEAT algorithm produced the extremely sparse network shown in Fig. 4.6a, consisting of two hidden units and six connections. The algorithm was able to rapidly evolve a successful policy that was capable of keeping the pole balanced for the full 30.0s.

Contrarily, the actor-critic reinforcement learning algorithm was used to train the model of Fig. 4.6b, which consists of a hidden layer of 32 units. This topology was experimentally determined through a hyperparameter tuning, elaborated in Appendix A, that tested the performances of networks with various numbers of hidden units and layers. However, although the policy encoded by the 32-unit network was found to perform the best among the other topologies that were tested, the neural network of Fig. 4.6b that trained through 1000 simulations taking a CPU time of 1.50hrs only kept the pole balanced for 5.5s. The states and control actions in the last 5.5 seconds of the NEAT and RL simulations are shown in Fig. 4.5, revealing how the RL policy exceeded the cart position limit in its best episode while the NEAT policy successfully maintained balance.

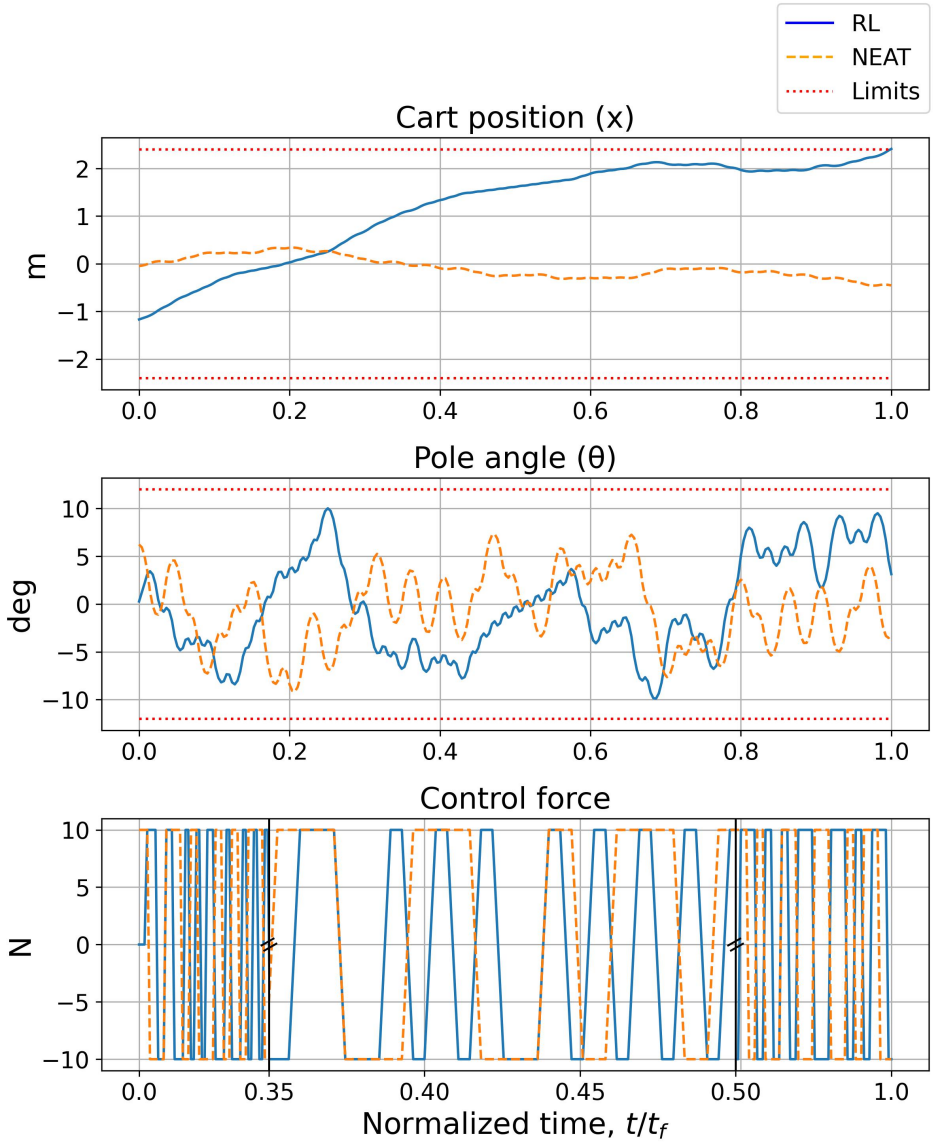
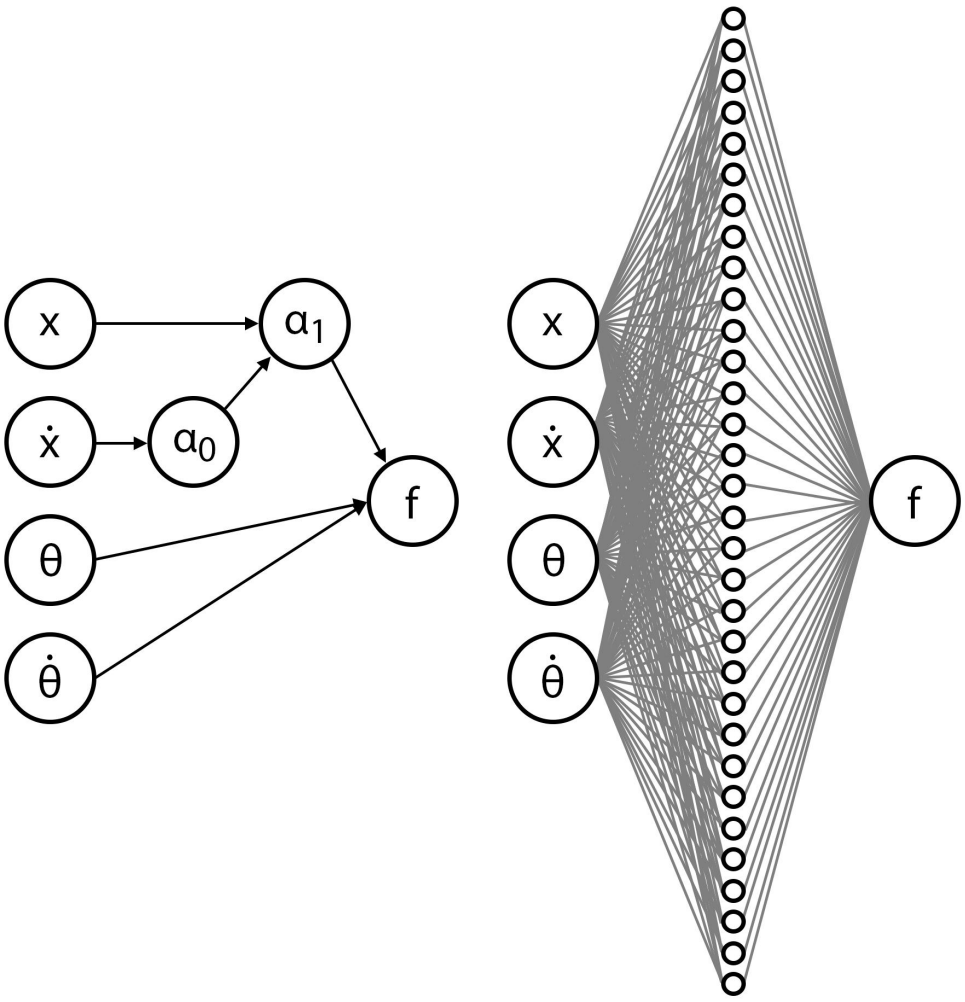


Figure 4.5: Simulation traces of NEAT and RL policies.

Apart from the difference in performance, which can be attributed to the extremely large number of training episodes required for the RL algorithm to adequately map the continuous

state space, the total number of connection weights and biases for the NEAT-based neural network is 9 while the RL-based policy is characterized by 161 values. Although the RL network is relatively small compared to reinforcement learning standards, the number of connections grows exponentially with the number of hidden nodes and layers. Therefore, for more complex control tasks with a larger continuous action space, the feedforward process may become a computational bottleneck within the control scheme. Another benefit of NEAT is that the topology is automatically tuned by the algorithm, while typical RL methods require the network's shape to be predefined prior to training. This is advantageous when the region in an algorithm's hyperparameter domain that would result in a successful policy solution is unknown.



(a) NEAT (b) Actor-critic RL

Figure 4.6: ANN topologies of policies from different training algorithms.

While the inverted pendulum task has smaller action and state spaces compared to autonomous soaring control, it was hypothesized that the repeatable, periodic nature of the behaviours required in both problems and the progressively-evolving nature of NEAT policies would yield similarly sparse networks for the soaring problem. The primary focus of the test

case was on validating the simplicity of the evolutionary networks when trained for physics-based control tasks. Ultimately, it was clear that the NEAT algorithm exhibited training and implementation advantages, thereby demonstrating its potential applicability to more challenging control problems.

4.3 Neuroevolutionary Control Approach

When applied to autonomous soaring, each member or neural network produced by NEAT represents a neurocontroller that dictates the attitude of an aircraft model in different soaring environments. The neuroevolutionary control (NEC) approach developed in this research is depicted by Fig. 4.7, which represents a method of evolving sparse neural network controllers. In the outer loop, the NEAT algorithm provides an initial population of neurocontrollers, with each member entering an inner evaluation loop so that its fitness, or soaring performance, can be determined. Every neural network is subjected to a single flight simulation, where the controller receives the states of the aircraft model and outputs control commands that are used to propagate the simulation and obtain the vehicle's future states. The loop continues for every discrete time interval from t_0 until the aircraft crashes or exceeds flight limits at t_f. Once the entire state and control trajectories experienced and commanded by the neural network are collected, they are used to calculate the member's fitness score, which is based on how well the network succeeded at the soaring task and by how much it exceeded constraints. The value is then passed back to the NEAT algorithm, which ranks the neurocontroller relative to others in its species and either eliminates the member or reuses its genes in subsequent generations. The entire process terminates once a fitness threshold has been reached, or a maximum number of generations has elapsed.

4.3.1 Fitness Function

The fitness function used to assess the relative performance of neurocontrollers is critical to the overall NEC process. Although adjustments must be made for different types of soaring, the core formulation that remains consistent across flight modes is defined by a penalty function.

The penalty function was designed to penalize behaviours that result in the destruction of the aircraft model, caused by the exceeding of aerodynamic and control constraints. Using the state and control trajectories collected during the flight simulation, the function outputs a value that is proportional to the degree to which the aircraft exceeded the various limits presented below:

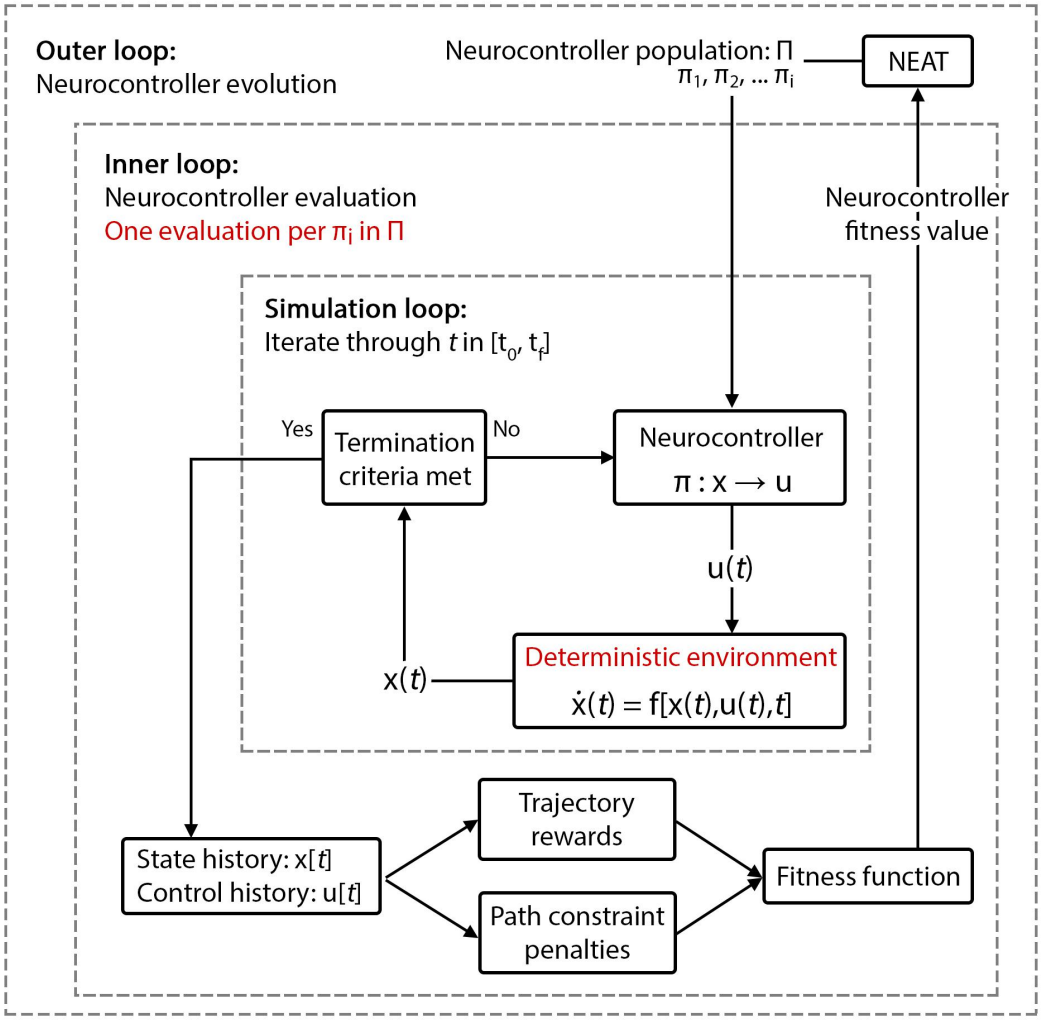


Figure 4.7: Neurocontroller evolution scheme.

$$
\begin{aligned}
\text{Airspeed:} \quad & V_{min} \leq V \leq V_{max} \\
\text{Height:} \quad & h_{min} \leq h \leq h_{max} \\
\text{Pitch angle:} \quad & \gamma_{min} \leq \gamma \leq \gamma_{max} \\
\text{Load factor:} \quad & n \leq n_{max}
\end{aligned}
$$

$$
\begin{aligned}
\text{Pitch rate of change:} \quad & \dot{\gamma}_{min} \leq \dot{\gamma} \leq \dot{\gamma}_{max} \\
\text{Heading rate of change:} \quad & \dot{\psi}_{min} \leq \dot{\psi} \leq \dot{\psi}_{max} \\
\text{Lift coefficient rate of change:} \quad & \dot{C}_{L_{min}} \leq \dot{C}_L \leq \dot{C}_{L_{max}} \\
\text{Roll rate of change:} \quad & \dot{\mu}_{min} \leq \dot{\mu} \leq \dot{\mu}_{max}
\end{aligned}
$$

The penalty s_{pen} for exceeding the constraints of one of these variables s is taken as the sum across all timesteps of the amount by which the limits s_{min} and s_{max} were exceeded.

$$s_{pen} = \sum_{t=0}^{t_f} [min(0, s_t - s_{min}) + max(0, s_t - s_{max})] \tag{4.8}$$

The total penalty of the neurocontroller p_π is calculated as the sum of the square of each penalty, divided by the total flight time t_f to ensure that the evolutionary process, in early generations, favours longer trajectories with correspondingly greater penalties over shorter flights with smaller penalties. Furthermore, a large penalty constant was added in the event of the aircraft crashing.

$$p_\pi = \frac{(V_{pen}^2 + h_{pen}^2 + \gamma_{pen}^2 + n_{pen}^2 + \dot{\gamma}_{pen}^2 + \dot{\psi}_{pen}^2 + \dot{C_L}_{pen}^2 + \dot{\mu}_{pen}^2)}{t_f} + p_{crash} \tag{4.9}$$

This penalty value is computed for each neurocontroller at every generation, and is consistent regardless of the type of soaring. However, there must also be an incentive for the agent to remain soaring for prolonged durations, which motivated the design of reward functions unique to the dynamic, thermal, and ridge soaring cases.

For travelling dynamic soaring, the objective is to maximize flight distance while consuming the minimal amount of energy reserves. As such, the reward r_π was calculated as the square of the aircraft's total displacement at the end of the flight, where x and y represent the vehicle's position over the horizontal plane:

$$r_\pi = (x_f - x_i)^2 + (y_f - y_i)^2 \tag{4.10}$$

Contrarily, thermal soaring involves maximizing the flight duration by remaining in updraft regions. Therefore, the reward was computed as the square of the vehicle's change in potential energy e_P over the flight interval $[t_i, t_f]$:

$$r_\pi = (e_{Pf} - e_{Pi})^2 \tag{4.11}$$

Finally, while the objective of ridge soaring is to also extend flight duration, a pacing reward was formulated to direct the agent towards learning the figure-eight pattern that allows for sustained soaring. Members were rewarded based on each cycle's lateral displacement, summed over the total number of cycles C in the entire trajectory, where x is the vehicle's position along the x-axis:

$$r_\pi = \sum_{i=1}^{C} max(x[i]) - min(x[i]) \tag{4.12}$$

Therefore, the fitness value f_π of a single neurocontroller after evaluation was calculated as the sum of the type-dependent reward and negated boundary penalty, with additional scaling constraints k that were experimentally tuned to prevent either value from dominating the other:

$$f_\pi = k_1 r_\pi - k_2 p_\pi \tag{4.13}$$

4.3.2 Simulation Environment

To evolve neurocontrollers, an explicit time simulation environment was created. Using the equations of motion for the flight agent along with the wind models described in Section 3, the simulation time was incremented by a discrete timestep $\delta > 0$, and future state values were computed through a first-order approximation. The rapid and simple to implement forward Euler method used to advance the simulation is detailed below, where a state x_k at timestep k is updated to x_{k+1}:

$$x_{k+1} = x_k + \delta \dot{x_k} \tag{4.14}$$

A neurocontroller undergoing evolution influences the states of the flight agent through the control commands that it outputs, consequently dictating its own trajectory. Throughout the simulation, the state and control histories are recorded for the fitness calculation after the simulation terminates either due to the member crashing the vehicle or sustaining flight for the maximum simulation duration. Therefore, the system dynamics are calculated separately from the fitness function called at the end of each simulation. The inner loop is summarized by Algorithm 1.

Algorithm 1 Flight simulation.

```
for neural network π_i in population Π do
    while t < t_f do
        states ← get_states()                          ▷ fetch aircraft states
        if states > constraints then
            break
        actions ← π_i(states)                          ▷ obtain control commands from ANN
        W_x, W_y, W_z ← get_wind()                     ▷ calculate local wind profile
        x_{k+1} ← x_k + δẋ_k                           ▷ apply dynamics model
        t += δ
    f_π ← get_fitness()                                ▷ compute rewards and penalties
```

Each run of the NEC process presented in this paper consisted of a population size of 250 members, repeated over a maximum of 100 generations. To accelerate the process of testing neural networks in the simulated flight environment, the 250 simulations conducted at every generation were run in parallel, with each simulation ending after a maximum flight time of 600s. The simulation environment, advanced at discrete intervals of 0.1s, was integrated into a NEAT implementation built in the Python programming language [30]. After the fitness of a neural network was evaluated, a new instance of the simulation environment was initialized so that all members of the population would experience the same training environment.

4.4 Deterministic Dynamic Soaring

This section presents the neurocontroller that was evolved for dynamic soaring in a deterministic, unidirectional wind environment, where parameters such as the initial aircraft position or wind profile remained unchanged between simulations. To validate the neural network's behaviour, its trajectory is compared with the canonical optimal solution presented in Section 3.7. Therefore, the flight agent's parameters, wind profile, and initial conditions listed

in Table 4.1 were selected to match those found in Sachs' work [1] as closely as possible.

Table 4.1: Dynamic soaring 3DOF flight agent and simulation parameters.

SUAV parameters	Value	Wind parameters	Value	Initial conditions	Value
S [sq ft]	7.0	A_x [-]	1.0	V_0 [ft/s]	41.3
M [slug]	0.582	h_{trx} [ft]	60.0	ψ_0 [deg]	0
E_{max} [-]	20.0	W_{max_x} [ft/s]	33.5	γ_0 [deg]	0
$C_{L_{max}}$ [-]	1.5			h_0 [ft]	60
$C_{L_{min}}$ [-]	-0.2	A_y [-]	1.0	x_0 [ft]	0
C_{D_0} [-]	0.033	h_{try} [ft]	60.0	y_0 [ft]	0
n_{max} [-]	5	W_{max_x} [ft/s]	15.7		
$\|\mu\|_{max}$ [deg]	60				
V_{stall} [ft/s]	38.7				
V_{md} [ft/s]	41.3				

Figure 4.8 shows the sparse neurocontroller that was successfully evolved in this environment after a CPU time of merely 3.38s. Effectively, the NEC process indirectly encoded the characteristics of the wind profile that the neurocontroller was subjected to during evolution into the neural network's topology and weights, and this knowledge of the environment allowed the agent to execute pitch and roll manoeuvres that led to dynamic soaring trajectories.

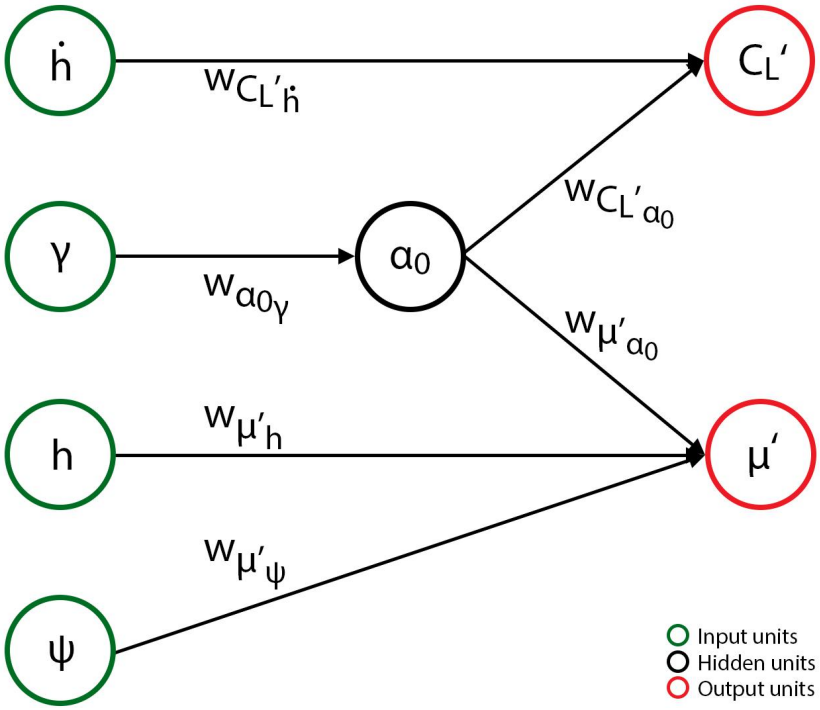


Figure 4.8: Topology of the deterministically-evolved dynamic soaring neurocontroller.

The output node biases $b_{C_L'}$ and $b_{\mu'}$, the hidden node's bias b_{α_0}, as well as the connection

weights w that were all evolved by the NEAT algorithm take the values:

$$b_{C_L{}'} = 0.979 \qquad w_{C_L{}'_{\dot{h}}} = -0.0189$$
$$b_{\mu'} = -0.728 \qquad w_{\alpha 0_\gamma} = 2.07$$
$$b_{\alpha_0} = 0.225 \qquad w_{C_L{}'_{\alpha_0}} = 0.0655$$
$$w_{\mu'_{\alpha_0}} = -1.04$$
$$w_{\mu'_h} = 1.78$$
$$w_{\mu'_\psi} = -2.05$$

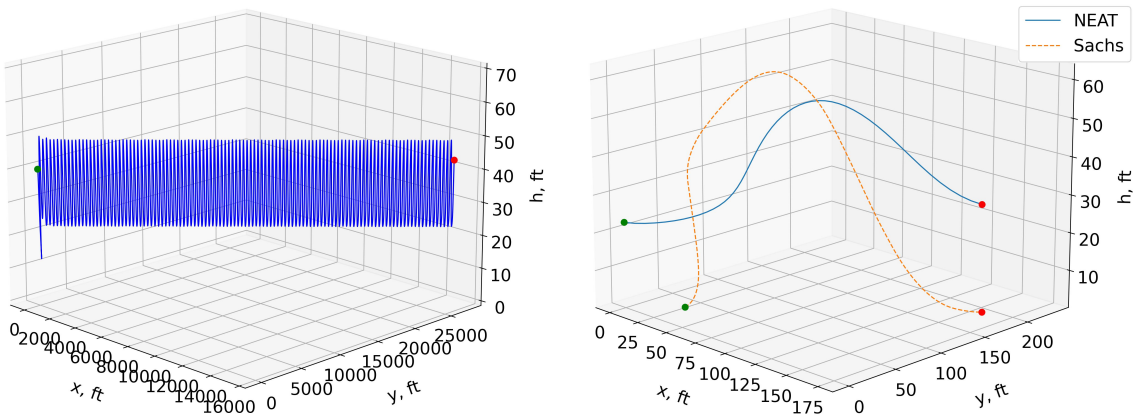
Mathematically, the feedforward network can be described as shown below, where Equation (4.19) converts the normalized outputs $C_L{}'$ and μ' of the sigmoid function $\sigma(x)$ to values within the control limits:

$$\alpha_0 = \sigma(w_{\alpha_{0_\gamma}}\gamma + b_{\alpha_0}) \tag{4.15}$$
$$C'_L = \sigma(w_{C_L{}'_{\dot{h}}}\dot{h} + w_{C_L{}'_{\alpha_0}}\alpha_0 + b_{C'_L}) \tag{4.16}$$
$$\mu' = \sigma(w_{\mu'_{\alpha_0}}\alpha_0 + w_{\mu'_h}h + w_{\mu'_\psi}\psi + b_{\mu'}) \tag{4.17}$$
$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{4.18}$$

$$\begin{bmatrix} C_L \\ \mu \end{bmatrix} = \begin{bmatrix} C'_L & 0 \\ 0 & \mu' \end{bmatrix} \begin{bmatrix} C_{L_{max}} - C_{L_{min}} \\ 2\mu_{max} \end{bmatrix} + \begin{bmatrix} C_{Lmin} \\ -\mu_{max} \end{bmatrix} \tag{4.19}$$

These input-output relationships produced the soaring cycles illustrated below, where Fig. 4.9a shows the simulation results of a 600s flight, and Fig. 4.9b shows a single period at the midpoint of the same trajectory, superimposed with the reference optimal trajectory of Fig. 3.6. In addition, the trace of the same cycle is displayed in Fig. 4.10, normalized in time for ease of comparison. Since the two paths were obtained through different objective and fitness functions, the flight paths vary in shape. However, the similarities between the airspeed, height, and roll plots reveal that the neuroevolutionary control approach can train neural networks that can control a flight agent along trajectories similar to those obtained through numerical optimization. The constant lift coefficient also shows that an agent can perform soaring with a fixed angle of attack. Lastly, the energy histories and specifically the peak total energy values at the start and end of the cycle indicate that after the initial transient period of energy gain shown in Fig. 4.9a, the neurocontroller consistently maintains potential and kinetic energies to produce sustained and repeatable dynamic soaring trajectories. Regarding the neural network itself, the use of different activation functions in the feedforward process had no noticeable effect on soaring performance, since the outputs were scaled to bounded control values as shown in Eqn. 4.19.

To better understand the differences in flight path, Fig. 4.11 shows the total specific energy of the first three cycles of the NEAT trajectory and the estimated energy of Sachs' optimal trajectory. The overall energy of the latter is lower due to the path's proximity to the surface. Regardless, the flight agent initially flies into the wind to climb the gradient before rapidly accelerating downwards. The NEAT-evolved trajectory behaves similarly in the first



(a) 79 cycles (600.0s) (b) 1 cycle (NEAT: 5.1s, Sachs: 7.2s)

Figure 4.9: Simulated trajectory of the deterministically-evolved dynamic soaring neurocontroller, with start marker (green), end marker (red), and path (blue).

cycle, where the agent, starting from a low-energy state, exploits the shear wind by sharply turning into the wind vector. However, after quickly gaining sufficient energy, the flight agent no longer needs to climb the wind profile as aggressively, and instead uses its momentum to cover distance. This change in priority from energy gain to soaring distance is a unique phenomenon that demonstrates the more complex, emergent behaviours of NEAT-evolved trajectories.

Apart from soaring performance, to demonstrate the close relation between the states and controls, Fig. 4.12 shows the input signals of an arbitrarily-selected slice of the trajectory plotted against the output control commands. While the lift coefficient was nearly constant throughout the flight, it was rescaled for this illustration. Even without the effect of the sigmoid function and the subsequent nonlinear shaping of the normalized controls, the outputs can be seen to track their respective inputs. While the traces of both the height RoC and the hidden node are similar and add to define the lift coefficient command, it is the height input in particular whose signal is closest to that of the roll angle, indicating its importance in determining the output. The proximity of the network's inputs to its outputs in terms of the number of intermediary nodes and connections enables the observing and analytical tracking of the network's feedforward process, and this topological traceability allows for an intuitive understanding of the neurocontroller. This interpretability is an important aspect for validating neurocontrol schemes that is difficult to achieve with the abstract, black-box nature of densely interconnected deep neural networks.

4.5 Deterministic Thermal Soaring

For the thermal soaring case, the input space was reduced to four variables by excluding the aircraft's heading. Without heading information, the neurocontroller is forced to evolve soaring behaviour that is dependent not on the geographic position of the thermal learned through trial and error, but instead on an identification of whether the aircraft is inside an updraft region, ultimately resulting in a more generalized policy. The thermal column parameters

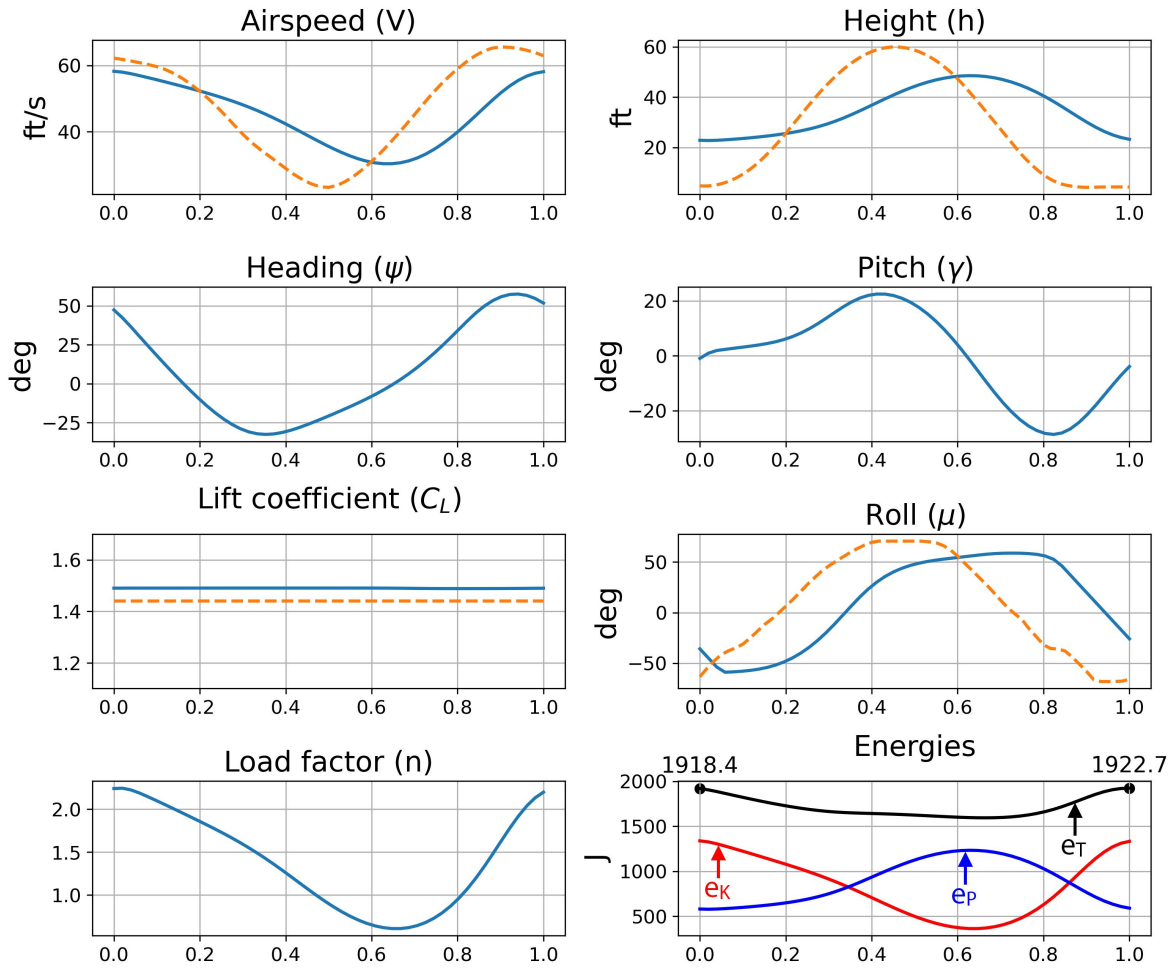
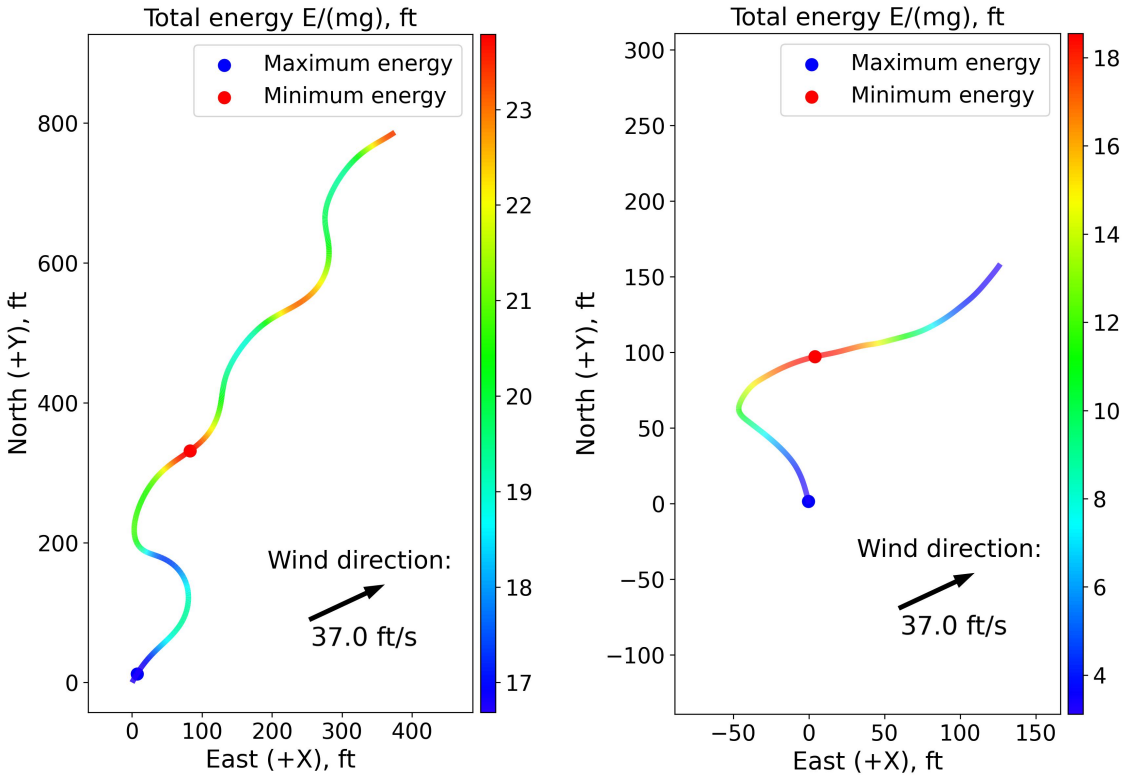


Figure 4.10: Trace of the deterministically-evolved dynamic soaring neurocontroller, NEAT (solid), Sachs (dashed) with respect to normalized cycle time, including kinetic e_K, potential e_P, and total e_T energies.

obtained from meteorological data [22] as well as the aircraft's initial conditions are described in Table 4.2, where the column's maximum height and maximum duration were extended to infinity to observe strictly intra-thermal soaring performance and not thermal mapping and searching behaviour. In addition, the evolutionary cases presented in the remainder of this chapter used the aircraft parameters listed in Table 3.1.

The evolved neurocontroller is depicted in Fig. 4.13, with its parameters described below. This particular member, trained in 5.81s of CPU time, formed a hidden node between the lift coefficient and roll angle outputs, such that the roll angle is partially dependent on the lift



(a) NEAT (3 cycles) (b) Sachs (1 cycle)

Figure 4.11: Total energies of NEAT and optimal trajectories.

Table 4.2: Thermal soaring 3DOF simulation parameters.

Wind parameters	Value	Initial conditions	Value
w_{core} [ft/s]	8.0	V_0 [ft/s]	27.3
r_{xy} [ft]	160.0	ψ_0 [deg]	0.0
$h_{tr_{max}}$ [ft]	∞	γ_0 [deg]	0.0
$t_{th_{max}}$ [s]	∞	h_0 [ft]	100
		x_0 [ft]	0
		y_0 [ft]	-160

coefficient.

$$b_{C_L'} = -0.991 \qquad w_{\alpha_{0_h}} = 1.28$$
$$b_{\mu'} = 1.56 \qquad w_{\mu'_{\alpha_0}} = -1.34$$
$$b_{\alpha_0} = 0.296 \qquad w_{\mu'_{\alpha_1}} = -1.43$$
$$b_{\alpha_1} = 0.200 \qquad w_{\alpha_{1_\gamma}} = -0.169$$
$$w_{C_L{}'_{\alpha_1}} = 1.42$$

The policy resulted in the trajectory of Fig. 4.14. The neurocontroller climbs the thermal column for the full simulation time without leaving the updraft core. The regular, patterned

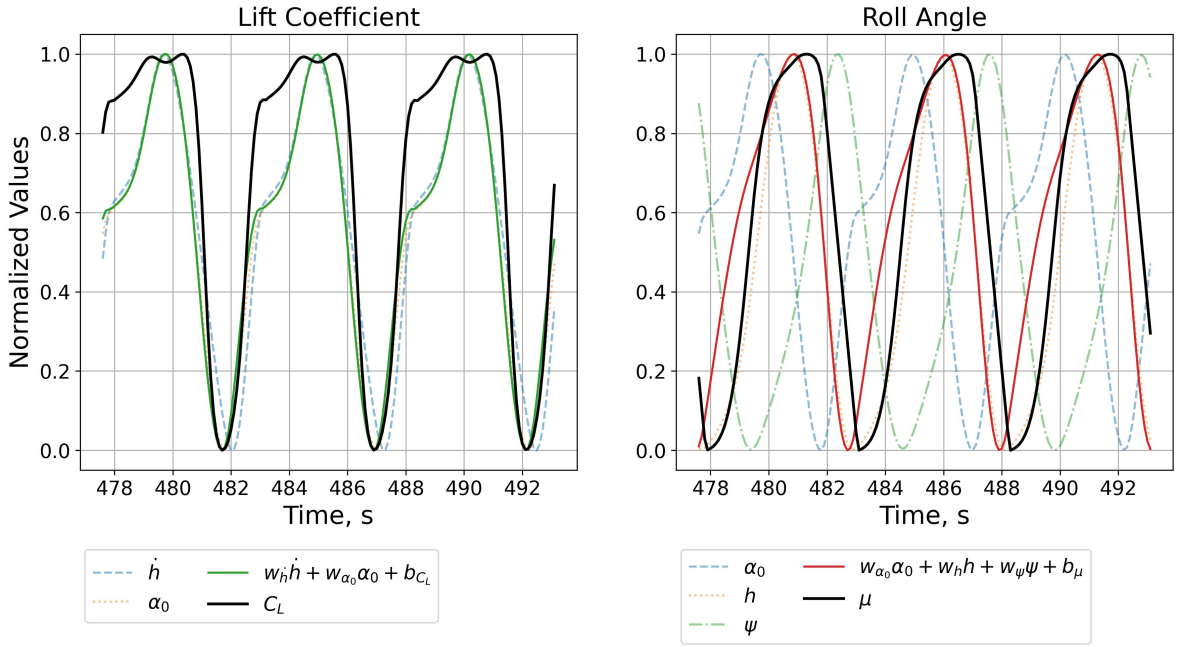


Figure 4.12: Network signal tracing of the deterministically-evolved dynamic soaring neurocontroller.

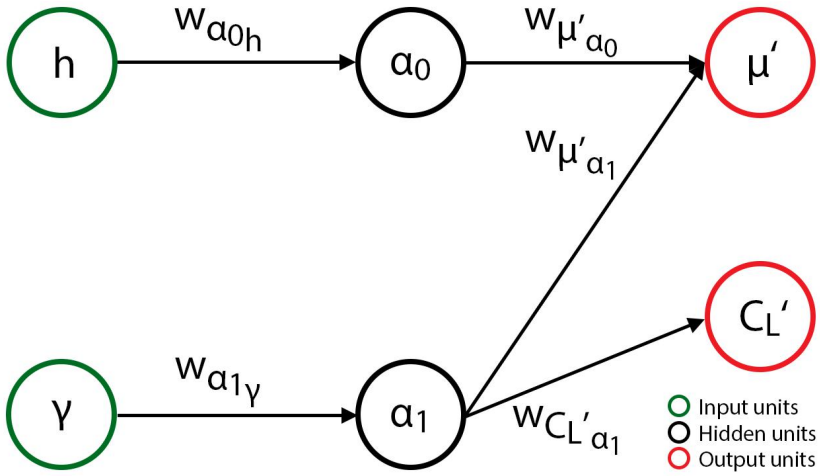
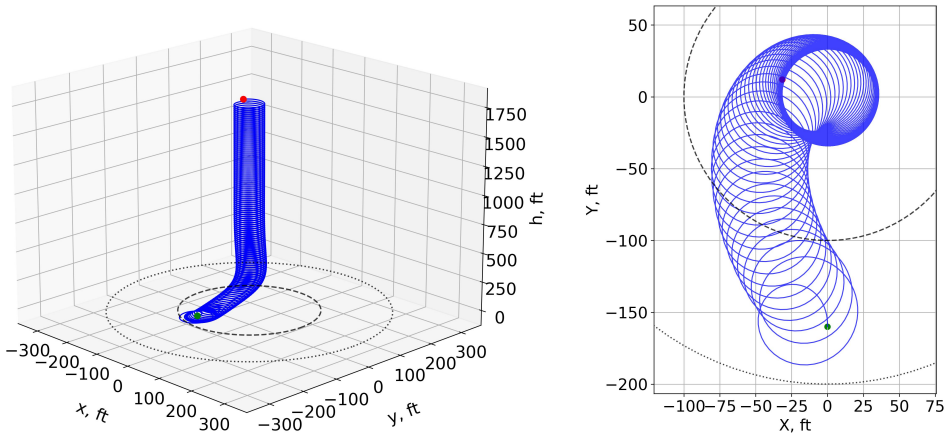


Figure 4.13: Topology of the deterministically-evolved thermal soaring neurocontroller.

trajectory as shown in Fig. 4.14a is a product of the evolutionary tuning of the network's connections and biases, since ensuring that the vehicle remains within the core would have maximized the flight-duration-based fitness function. The gradual drift towards the centre of the thermal indicates that the neural network evolved to exploit the higher updraft strengths at the core to maximize the potential energy gained during the flight. Furthermore, the states of Fig. 4.15 show that the constant lift coefficient and roll angle controls in the latter half of the trajectory allowed the aircraft to climb the updraft.



(a) 99 cycles (600.0s) (b) Bird's eye view

Figure 4.14: Simulated trajectory of the deterministically-evolved thermal soaring neurocontroller, with flight path (solid), inner updraft core (dashed), and outer downdraft region (dotted).

4.6 Deterministic Ridge Soaring

Evolving neurocontrollers becomes more difficult as the domain of potential action sequences increases. For instance, training a neural network to perform orographic soaring when starting at a high altitude near a cylindrical ridge model with a large radius is less likely to result in a successful neurocontroller after N generations when compared to an environment in which the vehicle is situated close to the ground and near a smaller ridge. The space of potential action sequences is much greater in the first case, since the aircraft is physically further from the ground and ridge, meaning the agent can perform a greater number of actions before crashing. Therefore, the search for a solution becomes a slow process. Contrarily, agents in the second case are more physically constrained and so is the action space, which increases the ratio of sequences resulting in successful soaring to action sequences that would cause a crash, assuming a successful trajectory is possible. This problem, illustrated Fig. 4.16, is one of the weaknesses of NEAT that was encountered for the autonomous ridge soaring case.

It was necessary to model ridges that had relatively large radii to produce sufficient updraft winds, such that a flight agent could remain soaring for extended periods of time. However, this also increased the aircraft's starting altitude, since the vehicle needed to clear the geographic feature to prevent an immediate crash and remain in the region of optimal soaring at 60 degrees elevation, noted in Section 3.4. To account for the extended action space, additional constraints were imposed on the aircraft, where breaking any of the following conditions terminated the flight and imposed the crash penalty:

- $y < y_0$ - discourages agent from moving backwards away from the ridge
- $|x| > l$ - discourages agent from moving past the edges of the ridge, lengthwise
- $y > R$ - discourages agent from flying in the leeward side of the ridge

The neurocontroller members were also provided information on the aircraft's ground track, with variables x and y being additional inputs. While this extended the search space of topologies and subsequently prolonged the evolutionary process to require 500 generations

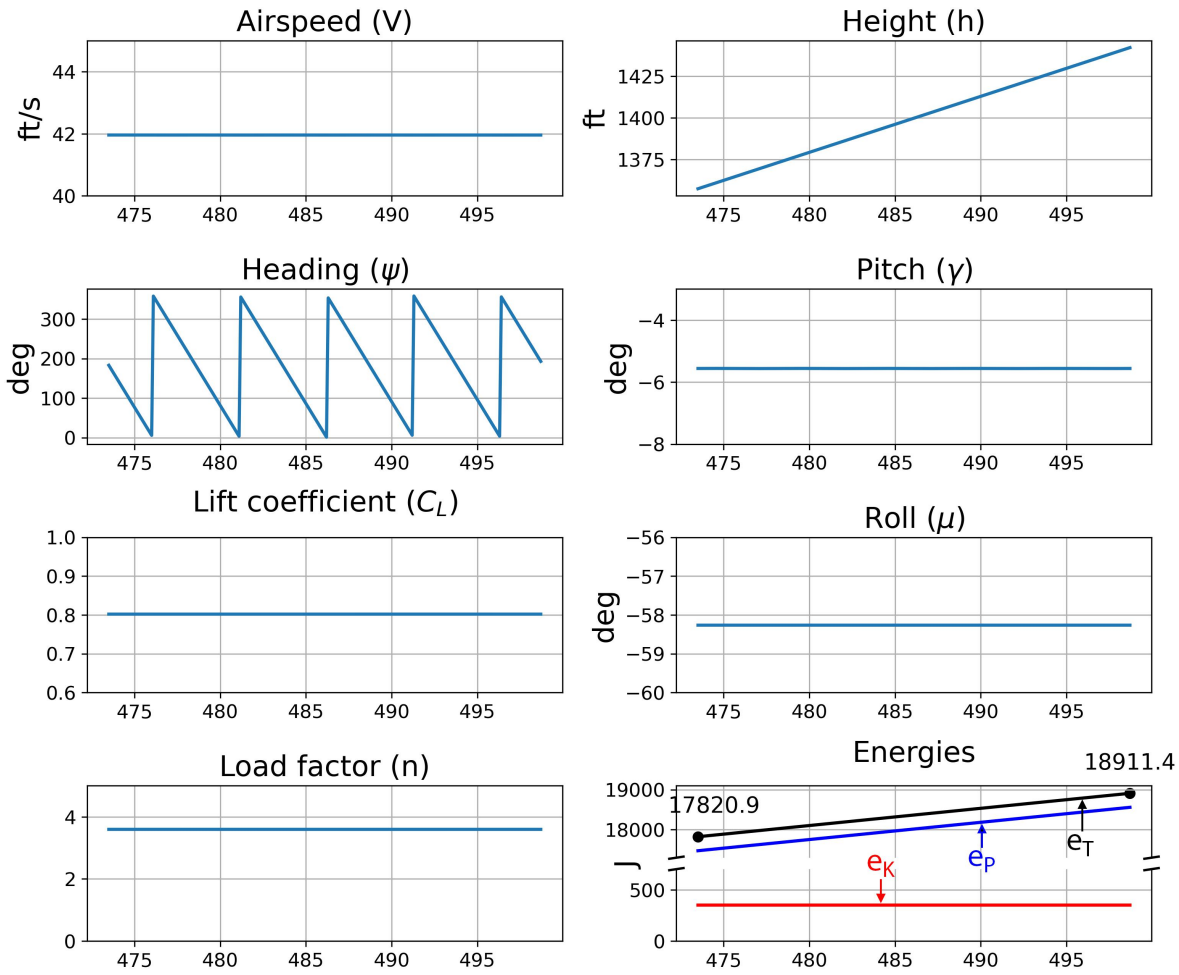


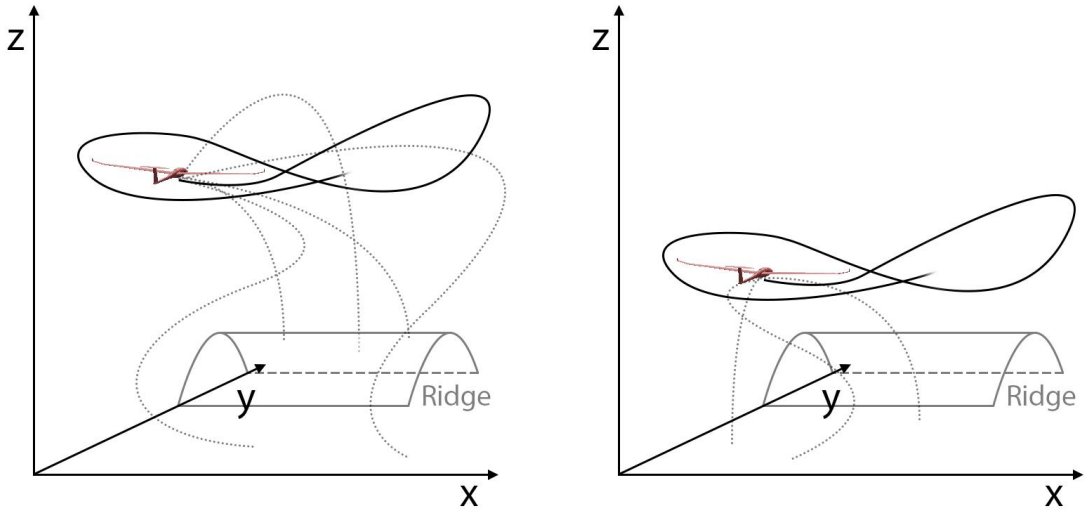
Figure 4.15: Trace of the deterministically-evolved thermal soaring neurocontroller, with kinetic e_K, potential e_P, and total e_T energies.

and a CPU time of 70.53s, the final network was able to use the supplementary data in generating the characteristic figure-eight trajectories of ridge soaring. These restrictions and additions allowed for the evolution of the network shown in Fig. 4.17 in the environment containing the ridge feature and local wind model listed in Table 4.3.

Table 4.3: Ridge soaring 3DOF simulation parameters.

Wind parameters	Value	Initial conditions	Value
$W_{\hat{h}}$ [ft/s]	10.0	V_0 [ft/s]	27.3
z_0 [ft]	2.13	ψ_0 [deg]	0.0
$\hat{h}$ [ft]	19.7	γ_0 [deg]	0.0
R [ft]	300	h_0 [ft]	600
l [ft]	200	x_0 [ft]	0
		y_0 [ft]	-1000

The network parameters detailed below describe a more complex ANN than previously



(a) Case 1 (large action space) (b) Case 2 (small action space)

Figure 4.16: Illustration of action space sizes for ridge soaring, with soaring flight path (solid) and potential flight paths (dotted).

seen, comprised of four hidden nodes and eleven connections. While the aircraft's position along the y axis remained unused, its x position proved to be a necessary input.

$$
\begin{aligned}
b_{C_L'} &= 0.139 \\
b_{\mu'} &= 6.33 \\
b_{\alpha_0} &= 0.793 \\
b_{\alpha_1} &= -0.325 \\
b_{\alpha_2} &= -0.547 \\
b_{\alpha_3} &= -1.91
\end{aligned}
\qquad
\begin{aligned}
w_{\alpha_{2_x}} &= 1.81 \\
w_{\alpha_{0_h}} &= 0.300 \\
w_{\alpha_{1_{\alpha_0}}} &= -5.73 \\
w_{\alpha_{2_{\alpha_1}}} &= -0.956 \\
w_{\alpha_{1_{\dot{h}}}} &= -0.411 \\
w_{\alpha_{2_{\dot{h}}}} &= -0.808 \\
w_{\mu'_{\alpha_2}} &= 4.26 \\
w_{\mu'_{\psi}} &= -7.16 \\
w_{\alpha_{3_\gamma}} &= 0.433 \\
w_{C_L'_{\alpha_3}} &= 1.57 \\
w_{C_L'_{\gamma}} &= -0.758
\end{aligned}
$$

The simulated trajectory and trace of the neural network are shown in Fig. 4.18 and 4.19, which show the characteristic figure-eight manoeuvre of ridge soaring on the windward side of the ridge. Although the rapid control actions shown in the trace are undesirable, the rate-of-change limits for the state and control variables can be adjusted to better filter the high-frequency signals. Nevertheless, the trajectory demonstrates how the NEC process is capable of generating more complex trajectories, albeit with additional constraints and inputs that guide evolution in the enlarged action and state spaces.

These cases show that through the developed NEC approach, neural networks can be evolved for a variety of soaring techniques, and may be further extended to other control

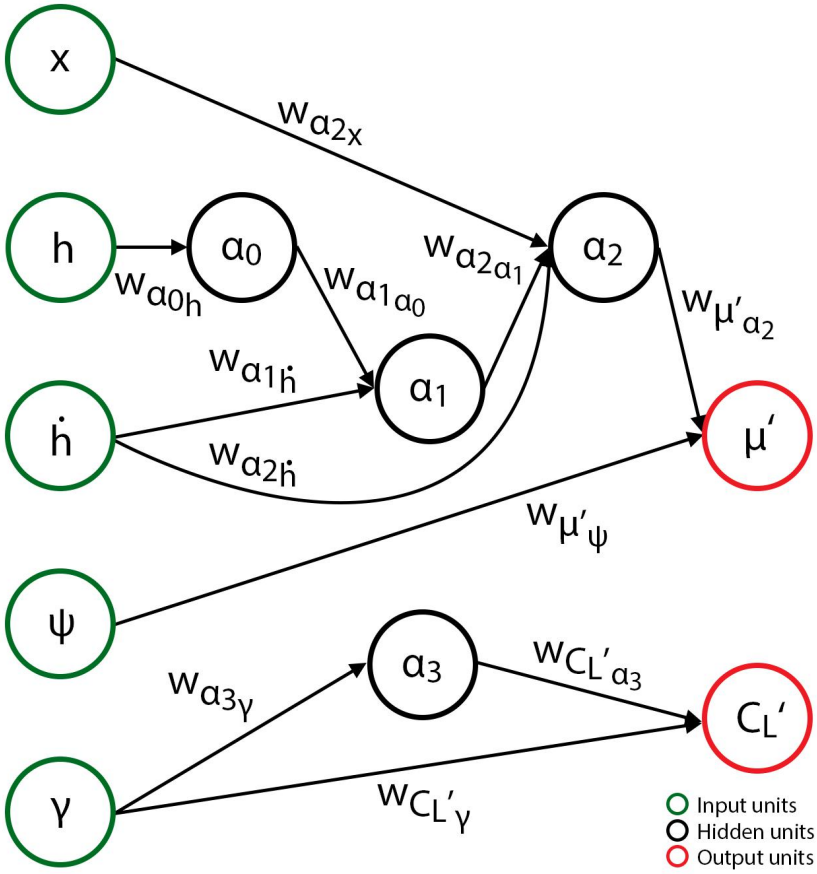
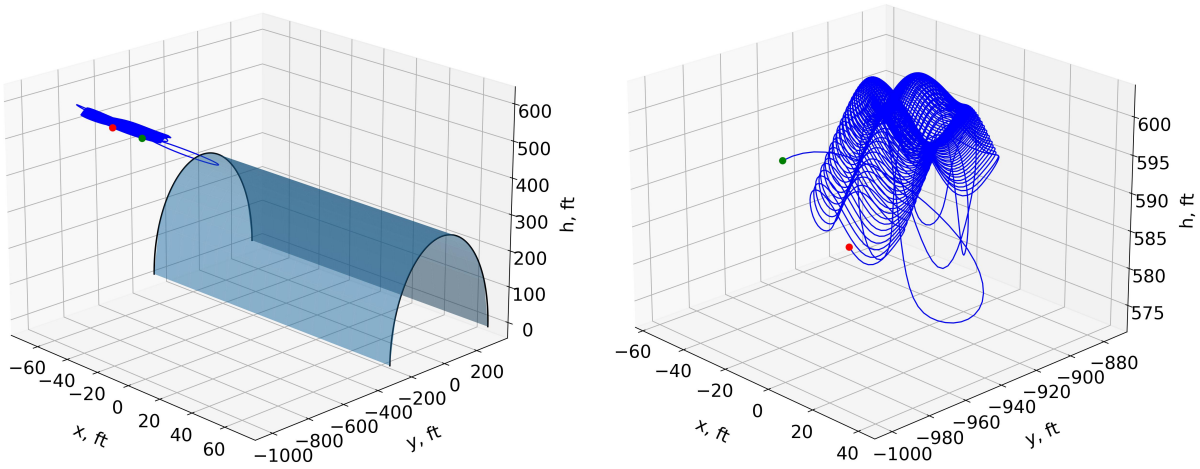


Figure 4.17: Topology of the deterministically-evolved ridge soaring neurocontroller.

problems. The primary advantage of these networks is their simplicity over the deep neural networks most commonly used alongside reinforcement learning algorithms. As a consequence of the neurocontrollers being defined by extremely few parameters, their resulting trajectories can be traced to each network's structural topology. Additionally, the evolution of the presented dynamic, thermal, and ridge soaring neurocontrollers took 3.38, 5.81, and 70.53 CPU seconds respectively, which correspond to wall times of 1.18m, 2.10m, and 15.45m on an Intel i5 4-core CPU. These relatively rapid training phases would be far exceeded by the backpropagation cycles of training DNNs using typical reinforcement learning algorithms. The neuroevolutionary control approach offers advantages in the training, interpretation, and implementation of neural networks for autonomous aircraft control.



(a) Complete trajectory with ridge (600.0s)

(b) Expanded view of trajectory (600.0s)

Figure 4.18: Simulated trajectory of the deterministically-evolved ridge soaring neurocontroller.

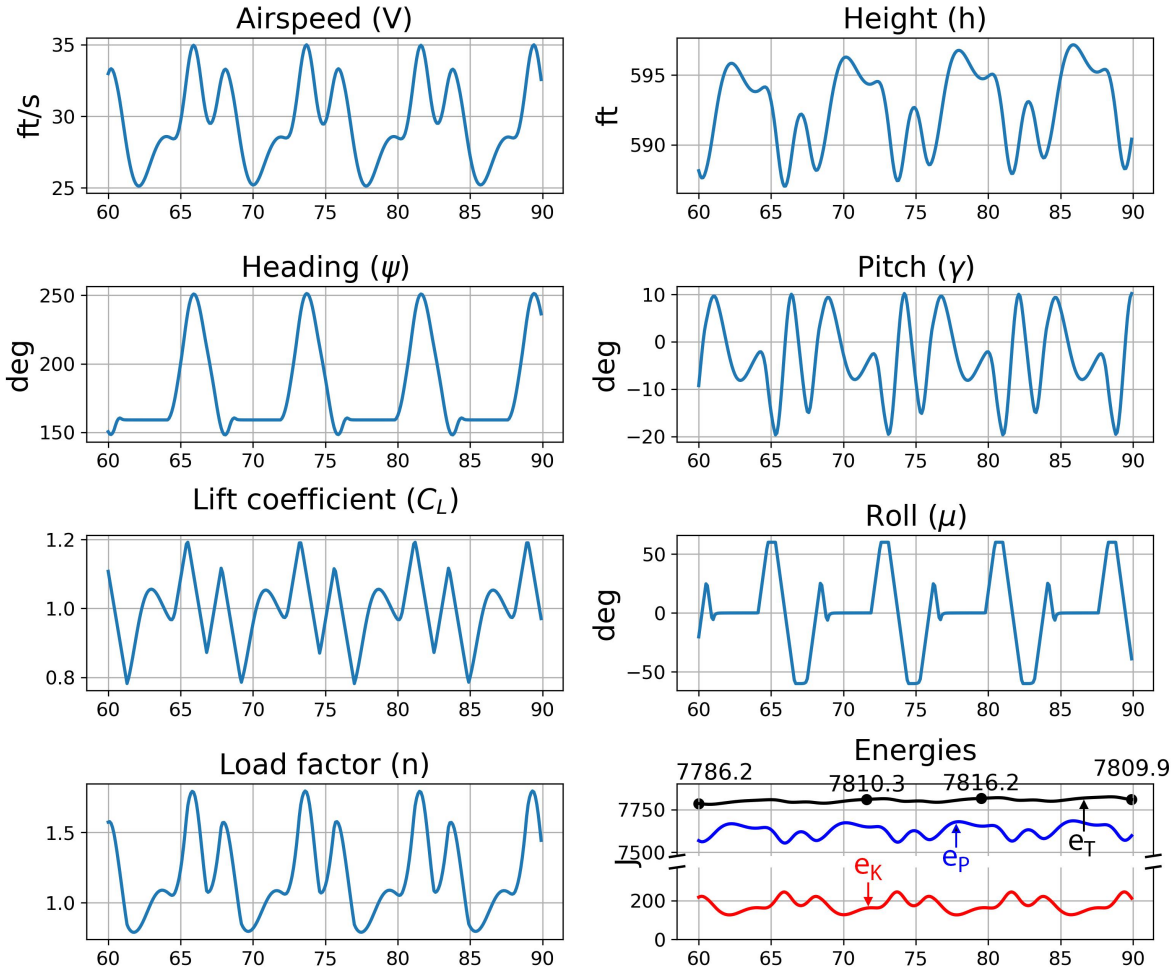


Figure 4.19: Trace of the deterministically-evolved ridge soaring neurocontroller, with kinetic e_K, potential e_P, and total e_T energies.

5 Robust Neurocontrol

This chapter presents the stochastic neuroevolutionary control (SNEC) approach that was developed as a continuation of the deterministic NEC method. The section describes the models of various uncertainties, as well as a way of quantifying the property of robustness from observations of neural network behaviour. Groups of dynamic soaring neurocontrollers are evolved for each type of uncertainty, producing an unbiased sample set of neural networks, whose topology and performance are analyzed to validate the SNEC method of evolving robust neural network controllers.

5.1 Stochastic Neuroevolutionary Control

The neurocontroller evolution scheme depicted previously in Fig. 4.7 describes the process of evolving neural networks capable of soaring in deterministic environments. Similar to trajectory optimization solutions, the resulting neurocontrollers are only successful for the specific set of conditions in which they were trained, and introducing any variations results in failure to exhibit soaring behaviour. Therefore, while the NEC approach was appropriate for demonstrating the applicability of neuroevolutionary control to the autonomous soaring problem, the local nature of its solutions is inadequate for real systems.

The next objective was to design a method of evolving robust neurocontrollers that could perform soaring in uncertain environments, where the aircraft and wind parameters are randomized. Thus, instead of evaluating each network once, it was deemed necessary to perform multiple flights per neurocontroller, where the fitness of each member is equivalent to some quantifiable measure of robustness. In consideration of these aspects, the stochastic neuroevolutionary control method illustrated in Fig. 5.1 was designed.

In the inner loop, each member of the neurocontroller population undergoes N_s evaluations, where the environment parameters are randomly sampled. With a sufficiently large number of unique evaluation loops, a given member's experience will span the domain of parameter values for which robustness is being evolved. Correspondingly, the network's final fitness value F_π after completing N_s simulations is computed by taking the sum of all the single-simulation fitnesses, whose formulation was shown earlier in Eqn. 4.13.

$$F_\pi = \sum_{n=0}^{N_s} f_{\pi_n} \tag{5.1}$$

$$f_\pi = k_1 r_\pi - k_2 p_\pi \tag{5.2}$$

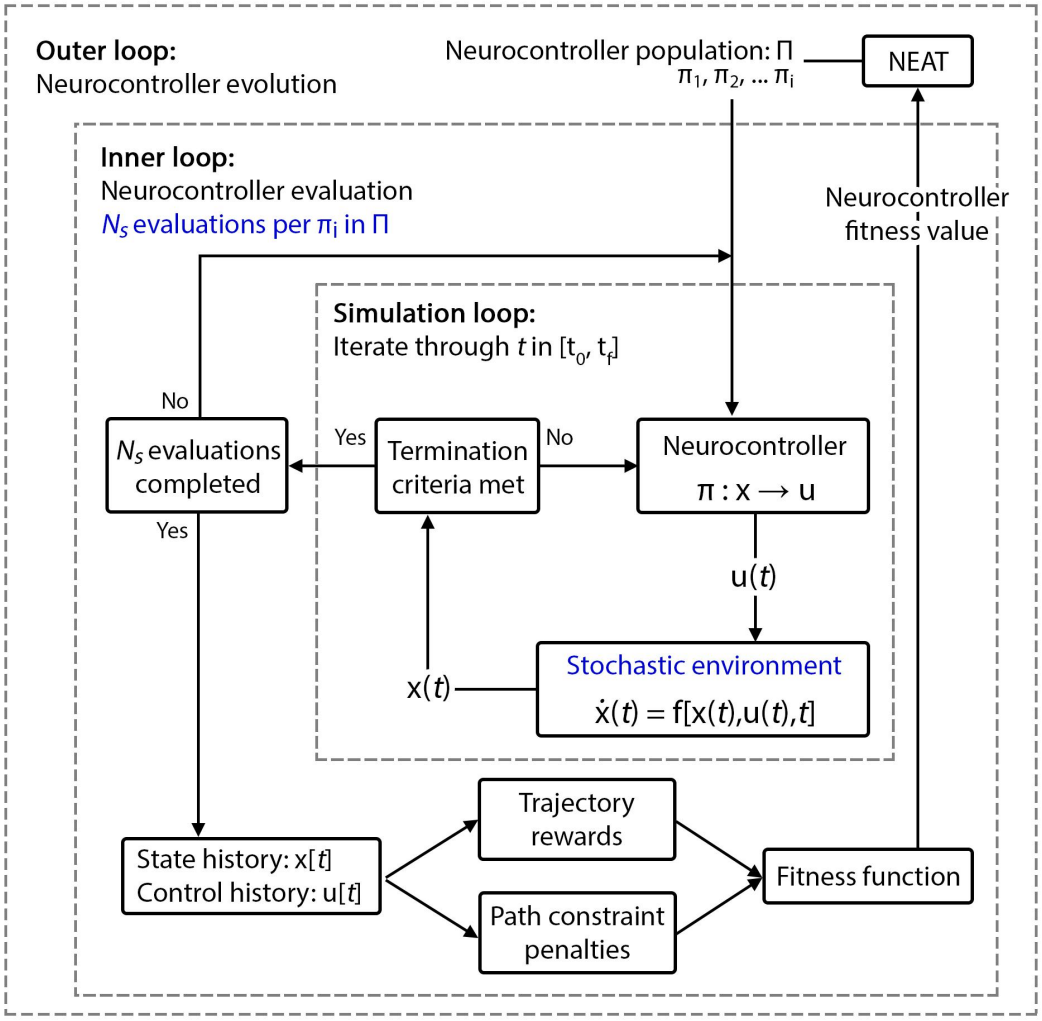


Figure 5.1: Robust neurocontroller evolution scheme.

This random sampling of stochastic environments in which neural networks are tested allows all the networks in the population to experience a variety of conditions, with the NEAT algorithm optimizing for the member that exhibits the best performance across all of its stochastic runs. Additionally, this effectively random sampling of neurocontroller performances can also be used to define a metric for robustness.

5.1.1 Robustness Quantification

In stochastic environments, a controller's performance during a single simulation is insufficient in describing its overall robustness to the uncertainty. Instead, a more accurate measure of robustness would be based on simulation results that encompass all combinations of initial parameters or random events within the range of values for which the network was trained. For instance, a neurocontroller evolved in a random initial pitch angle environment would need to be tested for each meaningful discrete pitch value in order to assess its resilience against the uncertainty. Multiple evaluations would also be needed for the initial heading angle, airspeed, altitude, and wind profile parameters. However, the extreme number of tests

that would be required to account for all combinations of these discrete values, in addition to the fact that time-related stochastic events that occur within a single simulation would further extend this number to infinity, make such exhaustive attempts to measure robustness impractical.

In light of these considerations, a Monte Carlo estimator was used to quantify and compare the relative robustness of neurocontrollers. The resilience of a given neurocontroller to an uncertainty was based on the accumulated results of a large number of nondeterministic flight simulations, and a robustness score r_b was defined as the number of simulations n_s for which a neurocontroller exhibited successful soaring trajectories without crashing up to the maximum simulation time t_f, divided over the total number of simulations performed n_T, taken as a percentage. This value, also referred to as the success rate and shown in Eqn. 5.3, provides a sense of how often a neurocontroller is able to conduct continuous autonomous soaring.

$$r_b = \frac{n_s}{n_T} \times 100\% \tag{5.3}$$

In applying Monte Carlo estimation, the central limit theorem states that the distribution of sample means of a sequence of independent and identically distributed (i.i.d.) random variables approximates a normal distribution. Defining $\hat{\mu}_n$ as the mean of n samples of an i.i.d. random variable with a true mean μ and standard deviation σ, the theorem can be expressed as the probability of a sample mean being within a standard score z tending towards the cumulative distribution function $\Phi(z)$ for a standard normal distribution, assuming the number of samples tends towards infinity [31]:

$$P\left(\sqrt{n}\frac{\hat{\mu}_n - \mu}{\sigma} \leq z | n \to \infty\right) \to \Phi(z) \tag{5.4}$$

A controller's robustness score can be considered as an i.i.d. random variable given that simulations are mutually independent and the probability distributions for the sources of uncertainty present across all simulations remain constant. By repeatedly sampling a neurocontroller's performance, or maximum achieved flight time, over numerous tests, it is possible to obtain the mean time of successful soaring for a certain confidence value.

For instance, the standard score z of a 95% confidence interval $c = 0.95$ can be obtained as shown below, where $\Phi^{-1}(\frac{1+c}{2})$ is the inverse of the cumulative distribution function $\Phi(z)$.

$$z = \Phi^{-1}(0.975) = 1.96 \tag{5.5}$$

Substituting z into Eqn. 5.4, the probability that the sample mean exceeds a value of $z\frac{S_n}{\sqrt{n}}$ units on either side of the true mean, where S_n is an unbiased estimate of the sample variance, can be expressed as:

$$P\left(|\hat{\mu}_n - \mu| \geq z\frac{S_n}{\sqrt{n}}\right) = 1 - c = 0.05 \tag{5.6}$$

$$S_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2 \tag{5.7}$$

Therefore, to obtain an estimate of the sample mean with a 95% confidence that the interval defined by $\pm x$ units contains the true mean, the Monte Carlo process can be set to continue until:

$$1.96\frac{S_n}{\sqrt{n}} < x \tag{5.8}$$

As long as n is large, the sample variance estimate can be used to terminate the evaluations and obtain an accurate mean. Therefore, in evaluating the neurocontrollers presented in this work, the Monte Carlo evaluations were set to end according to Eqn. 5.8, along with a minimum sample size $n_{min} = 1000$, with $x = 0.1$. In other words, the evaluation of a neurocontroller continued until the sample average of the fraction of successful simulations with respect to the total number of simulations stabilized within 0.1 percentage points, since the random variable, or the robustness score, is expressed as a percentage.

Another aspect of robustness is that for any given stochastic environment containing bounds on the values of its variations, it is not always guaranteed that a success rate of 100% is achievable, even for a hypothetical, optimally robust controller. That is, there may be regions of the stochastic parameter space in which dynamic soaring manoeuvres would be physically impossible. Therefore, a neurocontroller's score must be interpreted as a relative measure of performance in relation to other control schemes tested in the same environment.

5.1.2 Uncertainty Modelling

To test the SNEC approach through the unique trajectories of the travelling dynamic soaring case, various uncertainties were modelled, including environments with variable initial states, variable initial wind profiles, time-varying wind profiles, stochastic wind gusts, and sensor measurement noise.

Variable Initial States

There is a practical need for autonomous systems to initiate soaring manoeuvres from a range of initial aircraft states. As a result, neurocontrollers that were evolved to be robust against variable initial states were subjected, during the NEAT process' evaluation simulations, to environments in which the initial flight agent's states were randomly determined. The airspeed and altitude were initialized according to uniform distributions that spanned a certain multiple $m \in [0, 1]$ of a nominal value. The heading angle was assigned a value within a 360 degree arc, and the pitch angle was set to be between 45 degrees above and below the horizontal plane. The altitude's rate of change was except from any stochastic mechanisms under the assumption that soaring manoeuvres would typically commence from level flight. The uniform distribution ranges are depicted by Fig. 5.2.

Variable Initial Wind Profile

In real-world soaring, flight agents will be encountering a variety of wind profiles. Therefore, the parameters that characterize the winds necessary for dynamic soaring were randomly

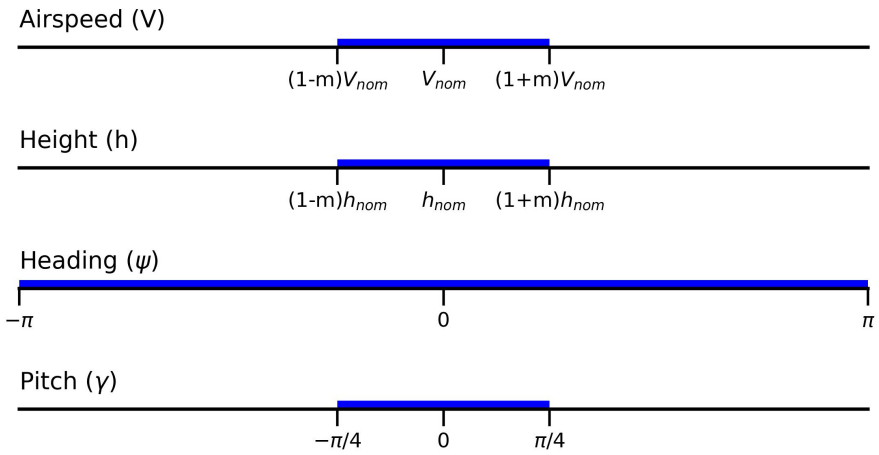


Figure 5.2: Uncertainty modelling: uniform distribution range for variable initial states.

initialized similar to the way in which the initial aircraft states were set. These variables include the shape parameter, the transition height, and the maximum wind strength.

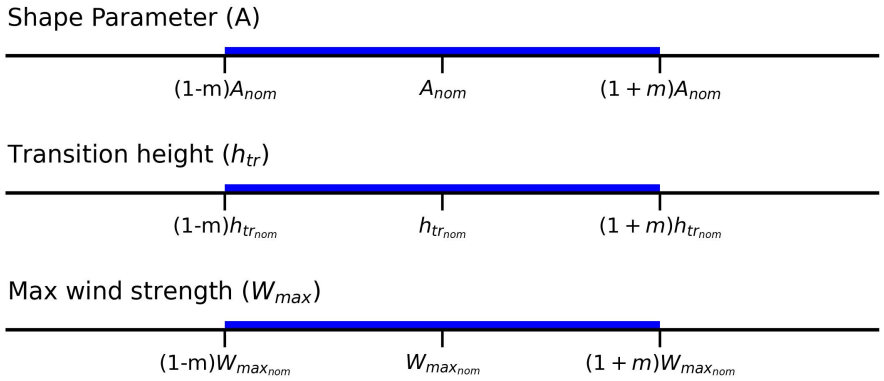


Figure 5.3: Uncertainty modelling: uniform distribution range for variable initial wind profiles.

Dynamic Wind Profile

In addition to stochastic initial parameters, wind profiles can be expected to change during flight. In the dynamic or time-varying wind profile environment, one of the three wind parameters (A, h_{tr}, W_{max}) was selected to be varied sinusoidally throughout the simulation. The frequency and amplitude of change were set during the simulation's initialization process based on a uniform distribution to oscillate the parameters in the timescale of typical soaring cycles. This effect is illustrated in Fig. 5.4, which shows a sinusoidal variance in the maximum wind strength W_{max}, indirectly inducing a similar variation in the wind profile at every altitude.

Wind Gusts

Another phenomenon of real-world environments is gusts, which are modelled as stochastic disturbances of various strengths and durations. At every simulation timestep, there is a probability p of a gust occurring in any direction in the three-dimensional Cartesian coordinate

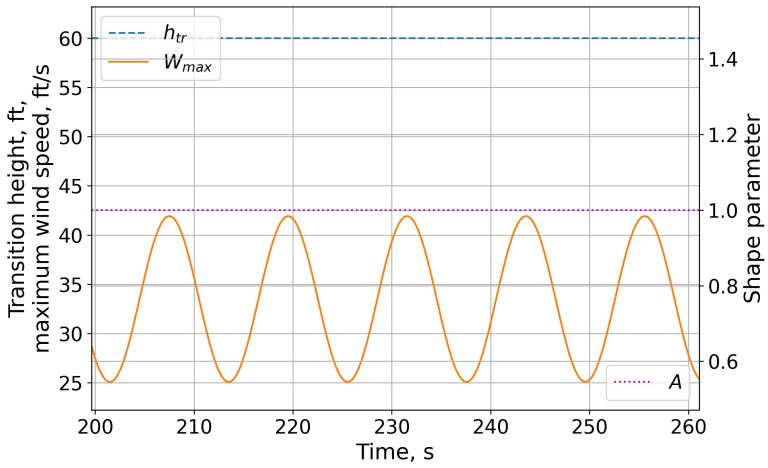


Figure 5.4: Uncertainty modelling: dynamically varying wind profiles for dynamic soaring.

system. If a gust is triggered, an initial gust magnitude up to a maximum value, a wind strength decay rate between 10% and 50%, and a maximum gust duration up to 1s are randomly generated based on uniform distributions. The gust magnitude is added to the preexisting local wind experienced by the flight agent at a given timestep, and this magnitude undergoes an exponential attenuation according to the decay rate until either the magnitude decreases to zero or the gust expires according to the maximum duration parameter. Finally, since gusts are treated as independent random variables, it is possible for multiple occurrences to overlap. The complete mechanism is illustrated in Fig. 5.5, which shows the magnitude, direction, and decaying of four separate gust events in various directions.

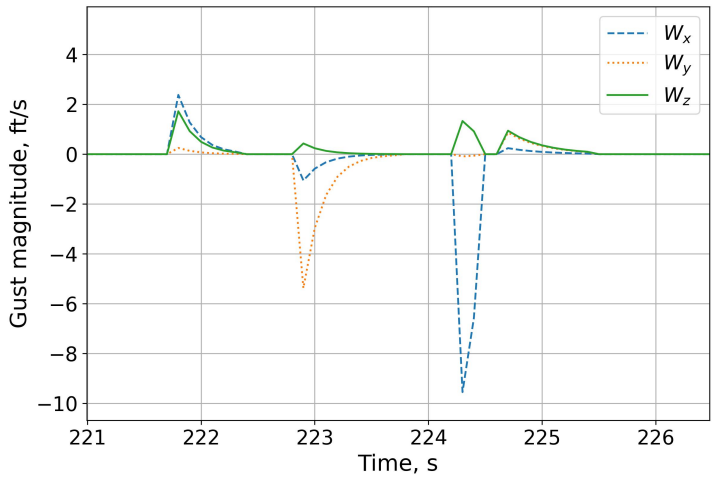


Figure 5.5: Uncertainty modelling: horizontal wind gusts.

Measurement Noise

Aside from parameter variations, environmental factors, and external disturbances, another form of uncertainty common to real-world systems is sensor measurement noise. Measurement noise is the stochastic fluctuations that perturb the states of a control system independently of the state values themselves. Therefore, the noisy environment was modelled by zero-mean

Gaussian distributions and unique standard deviations for each state. At every timestep, a random sample of the normal distribution is added to the state value, which is fed back into the neurocontroller as an input. Figure 5.6 shows the airspeed state that is experienced by the aircraft (solid) along with the signal that is inputted into the neural network (dashed).

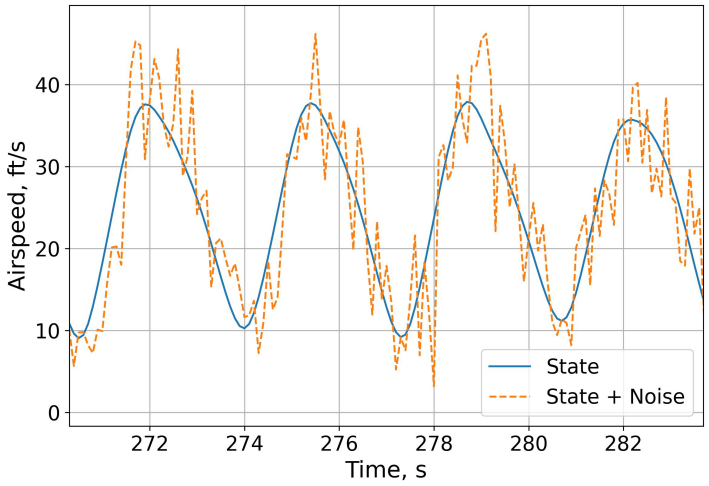


Figure 5.6: Uncertainty modelling: sensor measurement noise.

5.1.3 Stochastic Simulation Environment

Algorithm 2 outlines the process that was used to train and evaluate neurocontroller robustness. The pseudocode consists of, at simulation interval k: the aircraft states $\mathbf{x_k}$, the simulation number n, the standard deviation of the robustness score S_n, the confidence threshold v, and the robustness score sample mean $\bar{m}$.

5.2 Robust Dynamic Soaring Case Studies

This section presents an analysis of the various neurocontrollers that were evolved through the stochastic neuroevolutionary control approach. Their dynamic soaring performance in uncertain environments and the topological characteristics that enabled the robust behaviour are examined. However, the stochastic mechanisms of the NEAT algorithm make unlikely the generation of identical neural networks, resulting in variations in performance and structure even among neurocontrollers trained in the same environment. Conclusions from investigating a single neural network are therefore biased by the unique aspects of the particular controller. To account for these biases and to also demonstrate the approach's consistency in producing control solutions, a group of ten neural networks are consecutively evolved in each of the stochastic environments described in Section 5.1.2. Furthermore, instead of examining the Monte Carlo robustness scores of each neurocontroller in the group independently, a single score is computed for the entire group of networks by taking the weighted average of the ten members' individual success rates, based on the respective number of simulation runs that were required for convergence. The neural networks presented in this section were evolved for a maximum of 50 generations, each consisting of 100 evaluations. The time elapsed during evolution was dependent on the difficulty of the stochastic environment, but a training cycle lasting all 50 generations took an average CPU time of 6.77s, or a wall time of 2.78hrs. The

Algorithm 2 Robustness quantification.

for neural network π_i in population Π **do**
  $n \leftarrow 1$
  while $1.96\frac{S_n}{\sqrt{n}} > v$ and $n < n_{min}$ **do**
    $x_0 \sim U(0, \sigma_x^2)$ ▷ initialize states from uniform distribution
    $w_0 \sim U(0, \sigma_w^2)$ ▷ initialize wind parameters from uniform distribution
    while $t < t_f$ and $x_k < constraints$ **do**
      $states \leftarrow get\_states()$ ▷ fetch aircraft states
      $noise \leftarrow normal\_dist(\sigma)$ ▷ sample from normal distribution
      $x_k \leftarrow x_k + noise$ ▷ add measurement noise
      $actions \leftarrow \pi_i(x_k)$ ▷ obtain control commands
      $W_x, W_y, W_z \leftarrow get\_wind()$ ▷ calculate dynamic wind profile
      if $X \sim U_{[a,b]} > p$ **then** ▷ if gust is toggled
        $W_x, W_y, W_z \leftarrow add\_gusts()$ ▷ apply stochastic wind gusts
      $x_{k+1} \leftarrow x_k + \delta \dot{x_k}$ ▷ apply dynamics model to update states
      $t \leftarrow t + \delta$
      $\bar{m}_k \leftarrow \bar{m}_{k-1} + \frac{m_k - \bar{m}_{k-1}}{n}$ ▷ update robustness score sample mean
      $S_n \leftarrow S_n + (m_k - \bar{m}_{k-1})(m_k - \bar{m}_k)$ ▷ update robustness score std dev.
      $\bar{m}_{k-1} \leftarrow \bar{m}_k$
      $n \leftarrow n + 1$
  $f_\pi \leftarrow get\_fitness()$ ▷ compute rewards and penalties if evolving networks
  $r_\pi \leftarrow \bar{m}_k$ ▷ get robustness score if testing networks

aircraft and environment parameters of the SUAV model and unidirectional wind profile have been specified in Tables 3.1 and 5.1 respectively.

Table 5.1: Dynamic soaring 3DOF simulation parameters.

Wind parameters	Value	Initial conditions	Value
A_x [-]	1.0	V_0 [ft/s]	27.3
h_{tr_x} [ft]	60.0	ψ_0 [deg]	0
W_{max_x} [ft/s]	30.0	γ_0 [deg]	0
		h_0 [ft]	40
A_y [-]	N/A	x_0 [ft]	0
h_{tr_y} [ft]	N/A	y_0 [ft]	0
W_{max_x} [ft/s]	N/A		

Unlike the black-box nature of deep neural networks whose signals and behaviours are difficult to trace and intuitively comprehend, the sparse networks generated by the NEAT algorithm are more easily interpretable. In evolving the various neurocontroller groups, it was found that neural networks trained in the same environment consisted of similar connections. Therefore, a set of metrics were developed to examine the similarities and differences in the topologies of different groups by reducing the networks into their fundamental input-output connections.

First, the prevalence metric measures the frequency of all input-to-output connections in a group of networks, ignoring any intermediary nodes and duplicate connections. For instance, a prevalence score of 50% for a particular input-output pair would indicate that half of the neurocontrollers in a group contained a connection between the two nodes, regardless of the number of hidden nodes separating the pair.

In addition, an influence metric is computed as the product of all the connection weights between each input and output to provide a sense for an input-output relation's significance, taking into account hidden nodes and redundant connections. Since the connection weights that are linked to the input variables are dependent on the magnitude of the input value, the aircraft states being fed to the network are first normalized to a common order of magnitude between 0 and 1. Influence scores themselves are also normalized such that the sum of all scores of each input that head into a single output sum to 100. The connection in a network with only a single connection would have an influence score of 100 regardless of its weight, and a path between input-output nodes consisting of multiple hidden nodes would have a relatively small score, being the product of numerous intermediary connection weights whose magnitudes are naturally between 0 and 1. Taking the product of the prevalence and influence scores provides an importance score that quantifies each connection's overall value within the network.

Lastly, a complexity ratio was formulated to allow for the comparison of neural network topologies in terms of the relative number of interconnected pathways and neurons. While a single ratio is calculated by dividing the total number of connections and neurons of one network π by that of another, the complexity ratio c of a neurocontroller group A with n_A members with respect to another group B with n_B members is taken to be the mean ratio across all of its members:

$$c = \frac{\frac{1}{n_A}\sum_{\pi=1}^{n_A}(\text{number of biases}_\pi + \text{number of connection weights}_\pi)}{\frac{1}{n_B}\sum_{\pi=1}^{n_B}(\text{number of biases}_\pi + \text{number of connection weights}_\pi)} \tag{5.9}$$

The use of these metrics, for the analysis of unique neural network topologies, presents a quantifiable method of determining a connection's necessity in performing the control task and its impact in calculating the network's control output, while providing a sense of network complexity. This section presents an analysis on the robustness and topology of the neurocontroller groups that were evolved for each stochastic environment.

5.2.1 Deterministic Neurocontrollers

To assess the relative robustness of neurocontroller groups trained through the SNEC method, it was necessary to first train a collection of baseline neural networks in a deterministic environment. The ten networks were evolved with constant initial conditions and without disturbances or noise through the procedure presented in a previous work [32]. The simulation performance of this group is used for comparison of the robust neurocontrollers, and the deterministically-evolved group's topology provides a sense of the fundamental connections required for soaring flight.

Figure 5.7 presents the simplified network of all the neurocontrollers trained in the deterministic environment. The leftmost number of each pair of values above every input variable refers to the connection between the input and the lift coefficient, and the rightmost number relates the input to the roll angle. Furthermore, the first row indicates the prevalence as a percentage of the total number of neural networks in the group, and the second row represents the influence, with both metrics also visualized through the thickness and darkness of the connections, respectively. The third row displays the overall importance score. Lastly, the deterministically-evolved group is the baseline relative to which the complexity of the other network groups are compared. As a result, its complexity ratio is shown to be 1.0, and groups with more complex networks that consist of a greater number of connections would have a larger ratio.

The reduced network shows that every neurocontroller in the deterministically-evolved group had evolved a connection between the heading angle and the roll command. Additionally, the height and airspeed were shown to be the most common inputs for the lift coefficient. These topological traits reveal the most significant correlations between input-output pairs, which can be explained by examining the nature of dynamic soaring. For instance, the ground track trajectory of cycles on the xy plane is not linear and instead takes the shape of a curve, making the heading angle an important variable in determining the aircraft's roll. Specifically, the vehicle must roll towards the wind profile during the low altitude turn and away from the wind in the high altitude turn segment. For the lift coefficient, however, the relative lack of a single prominent connection reflects the high variance in the topologies within the group. The evolutionary process did not find any particular input-to-lift-coefficient relation that substantially improved fitness, so the different connections manifested in nearly equal proportions. The training of successfully soaring neurocontrollers that only had access to the roll control in flight simulations where the lift coefficient had been fixed to an experimentally-tuned constant demonstrates the lower importance and therefore higher connection variance of the lift coefficient control. Regardless, the deterministic topology is used as a baseline to compare and examine the specific attributes that encode robust properties in the stochastically-evolved neurocontrollers.

5.2.2 Initial-State-Robust Neurocontroller

The deterministically-evolved neurocontroller would be ineffective and impractical in real-world environments, where it is not guaranteed that the initial aircraft states would precisely match those seen during the controller's training. Therefore, for the evolution of the neurocontroller group that was evolved to be resilient to randomized initial states, the values of the state variables were sampled from uniform distributions, which, for the airspeed and height, spanned a range of $\pm 25\%$ of the nominal value.

The collective results of the Monte Carlo simulations for the deterministic and initial state-robust neurocontroller groups tested in the variable initial state environment are plotted in Fig. 5.8. The height of each bar represents the percentage of the total number of tests where the aircraft soared for the amount of time reflected in the horizontal axis, and was computed through a weighted average of the simulation results of every controller in the group. The group success rate is simply the height of the bar at the maximum simulation time of 600.0s.

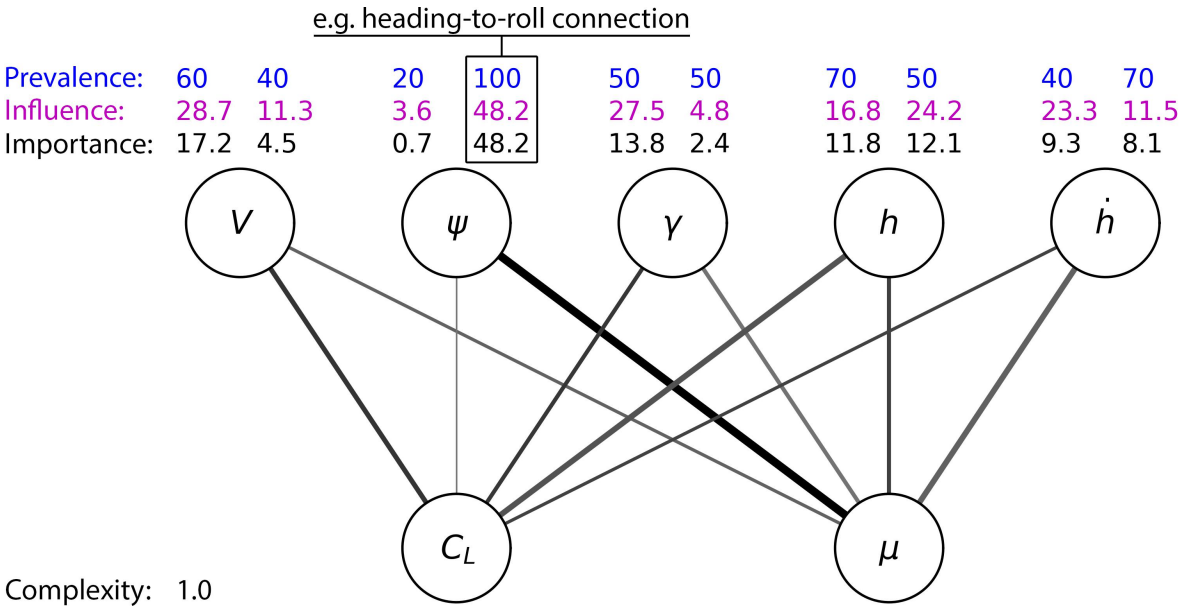
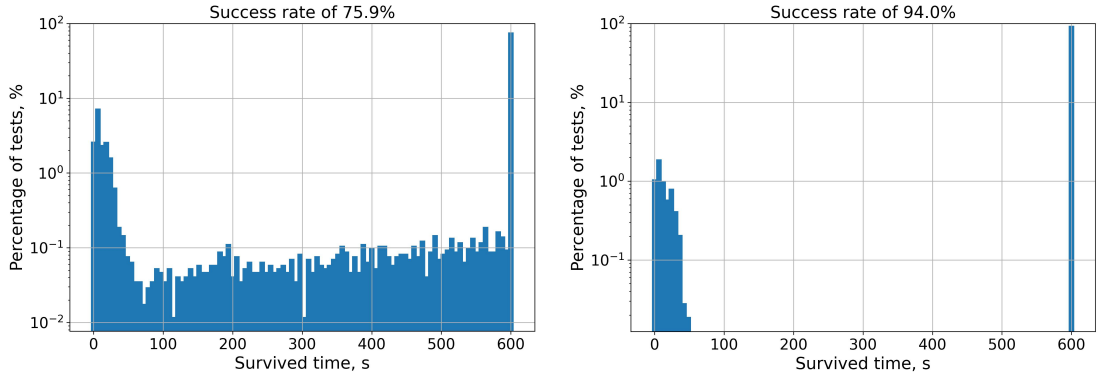


Figure 5.7: Simplified topology of deterministically-evolved neurocontrollers.



(a) Deterministic group (b) Initial state-robust group

Figure 5.8: Monte Carlo robustness for deterministic and initial-state-robust neurocontrollers.

The results are polarized towards immediate failure and sustained soaring, since a controller would either fail to initially extract sufficient energy and soon crash, or quickly enter and maintain a sustainable dynamic soaring trajectory for the duration of the simulation. The success rates show that neurocontrollers evolved in the stochastic environment are much less affected by randomized initial states than the neural networks of the deterministic environment. Considering that each network in the robust group had been, during evolution, the member of the population pool with the highest fitness, this network by definition also had been the most robust. Therefore, the SNEC process explicitly optimizes for robustness against the range of parameters that are varied during evolution. Moreover, the difference in the averaged success rates of both the deterministic and initial-state-robust groups, when calculated over multiple consecutively-evolved members, shows that the robustness training method consistently produced resilient controllers.

To better understand the individual sensitivities of the deterministic and initial-state-

robust neurocontrollers, Figures 5.9 and 5.10 show the mean survival times of the Monte Carlo runs against each of the initial states that were varied in the stochastic test environment. Every data point represents the average time for which the aircraft remained soaring at various discrete values of the initial parameters, calculated once again by averaging all the simulations from every network in the group. The results of the deterministically-evolved group show that low airspeeds, low altitudes, and extreme pitch angles were unfavourable for dynamic soaring, while the stochastically-evolved networks were only particularly susceptible to high pitch angles, which point the aircraft towards the ground. Regardless, although the performance of a nonexistent optimally robust controller is unavailable, the stochastically-evolved group is clearly less sensitive to the aircraft's initial orientation. Furthermore, the robustness of a neurocontroller is a function of the number of situations it encounters during training, and therefore, longer evolutionary cycles with a greater number of randomized domains can further reduce the performance loss in certain parts of the stochastic parameter space.

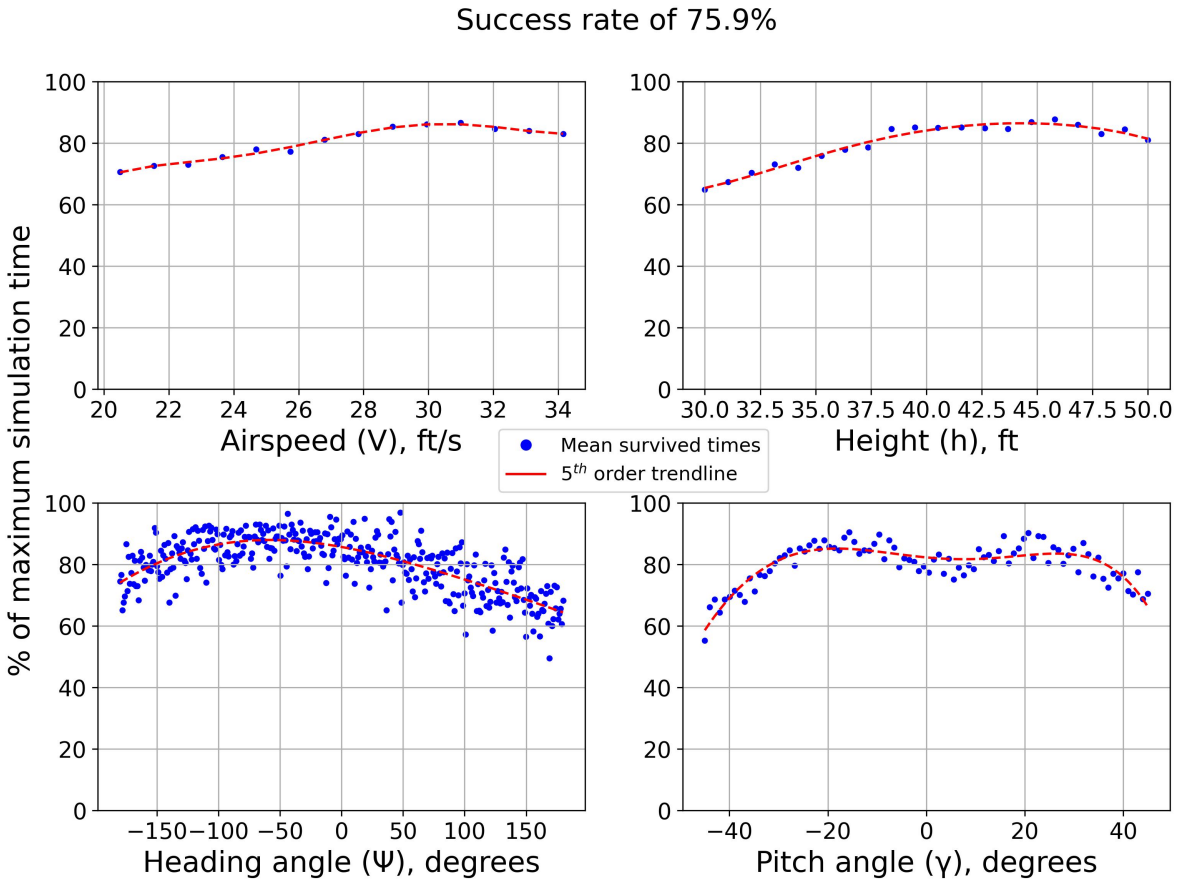


Figure 5.9: Initial state robustness for deterministically-evolved neurocontroller.

Figure 5.11 presents the simplified topology of the initial-state-robust neurocontrollers, where the fourth row indicates the change in importance scores relative to those of the deterministically-evolved group. The pitch angle became completely decoupled from the roll angle, which suggests that the connection was found to be detrimental to soaring. Instead, the heading-to-roll connection became slightly more emphasized in controlling the vehicle, while

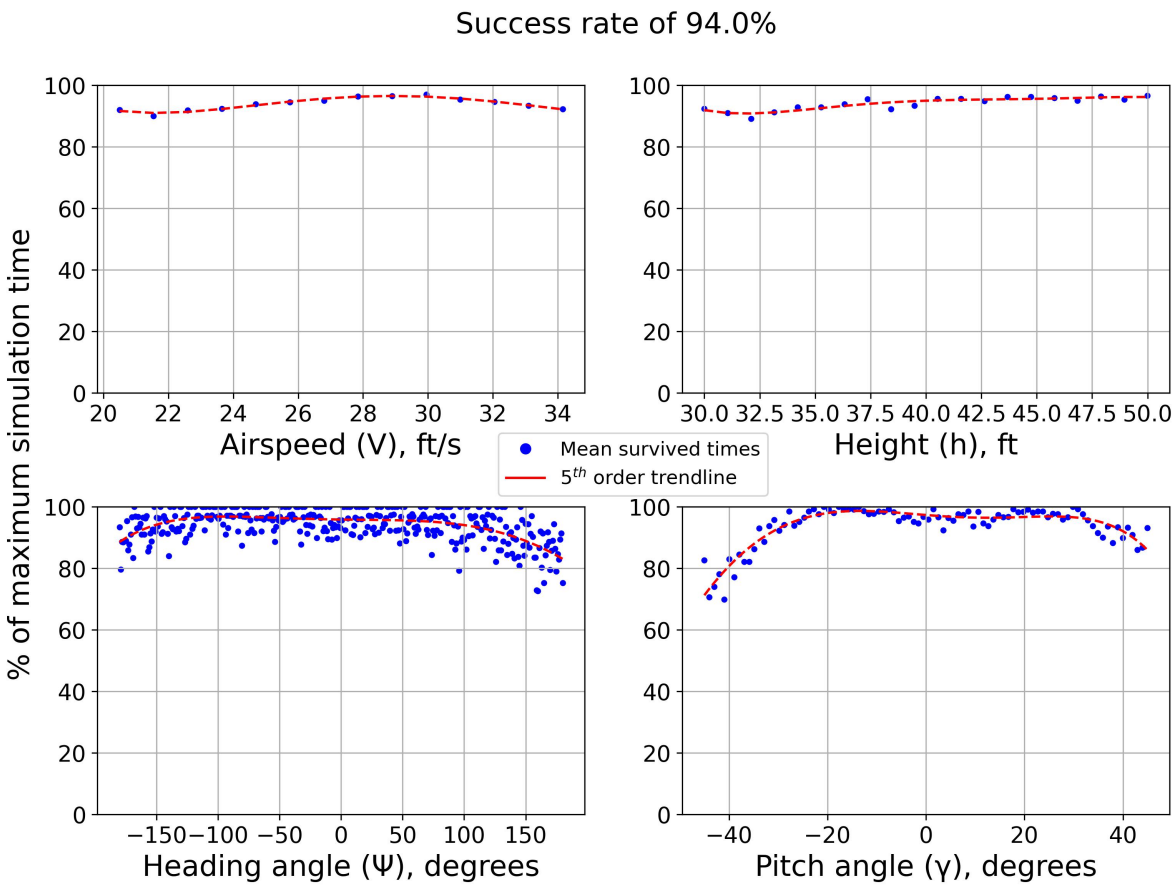


Figure 5.10: Initial state robustness for initial state-robust neurocontroller.

the prevalence of all the other input variables decreased for both controls. In the initial-state-varying environment, the controller must reorient the aircraft in a specific direction from which dynamic soaring is possible. Since the heading cannot be derived as a function of the other inputs, the heading input's importance in steering the vehicle is amplified. The simplified topology also reveals a greater variance in the importance of each input-to-lift-coefficient connection, again marked by the decrease in the prevalence scores for all inputs. The already weak relation between successful soaring performance and the lift coefficient is emphasized by the uncertain nature of the environment. Lastly, the complexity ratio of 0.80 indicates that the average network in the initial-state-robust group consists of 20% fewer parameters than the average deterministically-evolved neurocontroller, showing that robustness can be achieved with sparse topologies.

5.2.3 Initial-Wind-Robust Neurocontroller

To produce the initial-wind-robust neurocontroller group, the parameters of the wind profile, A, h_{tr}, and W_{max}, were obtained from uniform distributions spanning ±50% of the nominal values for each parameter.

Figures 5.12 and 5.13 show the sensitivities of the deterministic and initial wind-robust

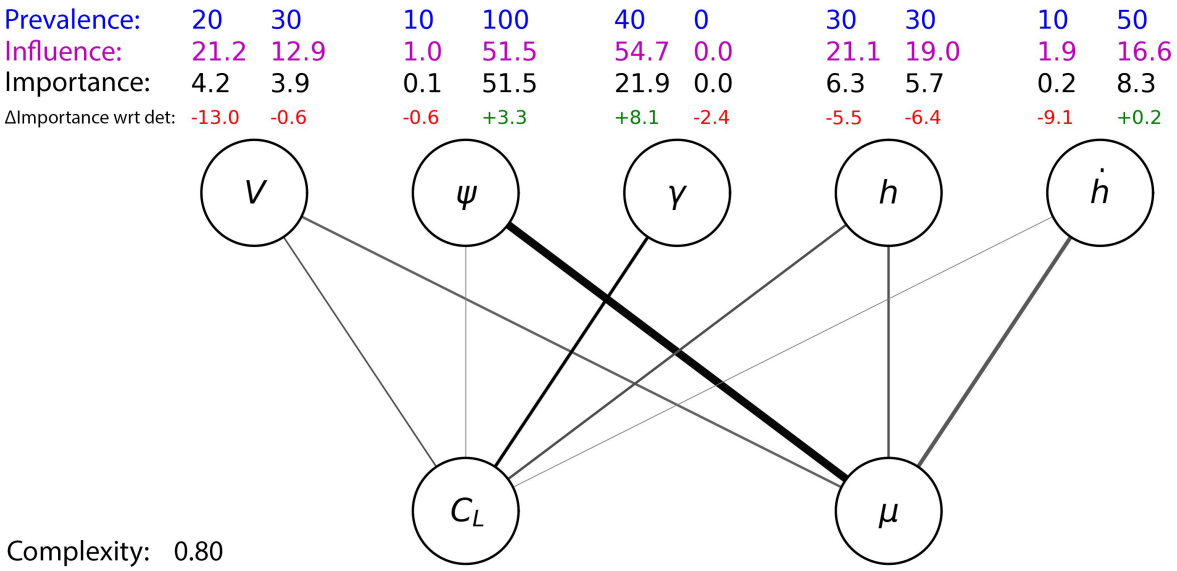


Figure 5.11: Simplified topology of initial-state-robust neurocontrollers.

neurocontrollers with respect to the wind parameters. Although the average success rate of the robust controller group is over 1.6 times that of the deterministically-evolved group, the objectively low value of the former indicates that it is difficult to conduct dynamic soaring for a significant portion of the randomized parameter space. In particular, logarithmic profile shapes, extreme transition heights, and low wind strengths were shown to pose significant challenges for the neural networks. However, both groups exhibited greater performance in exponential wind profiles, where the wind strength at all altitudes is either equal to or greater than the corresponding wind strength experienced in logarithmic profiles. Comparison of each group's transition height sensitivity also shows that the robust neurocontrollers evolved to take advantage of the lower transition heights' stronger gradients. The decrease in average flight times at the lowest values indicates that there exists an optimal altitude range that is best suited for dynamic soaring regardless of the gradient strength, due to the control boundaries and subsequent limits to how quickly an aircraft can climb the wind profile to sufficiently extract energy. Lastly, the monotonically increasing trendline of the robust group's maximum wind strength sensitivity shows that greater values resulted in longer soaring trajectories. This is expected, since the amount of energy that an aircraft can extract from the wind is proportional to the magnitude of the wind's vertical gradient, which is proportional to the maximum wind strength. In this case, the wind strength was also not large enough to completely overwhelm the aircraft. Regardless, the analysis shows how the stochastic simulations and Monte Carlo robustness scores can lead to an improved understanding of not only the neurocontroller, but also the environments that are best suited for certain flight techniques.

The most notable difference between the initial-wind-robust networks, summarized in Fig. 5.14, and the deterministic baseline is the higher prevalence and influence of the height input with respect to the roll angle. At every simulation instance, although all three wind parameters are randomly sampled from uniform distributions, only the transition height affects the behaviour of the controller with respect to the vehicle's height. The shape parameter and

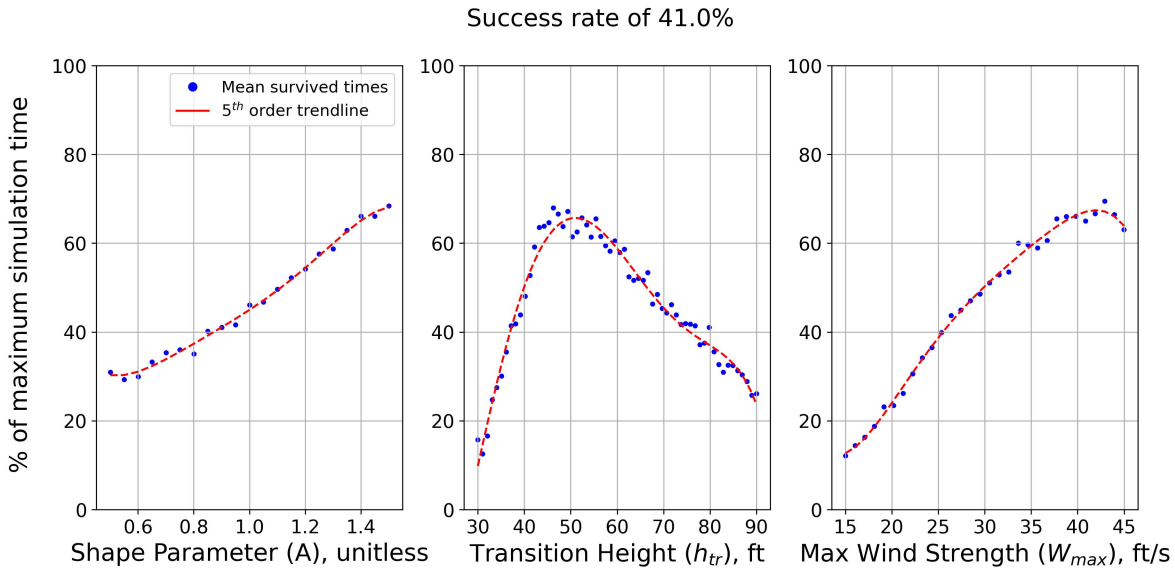


Figure 5.12: Initial wind robustness for deterministically-evolved neurocontroller.

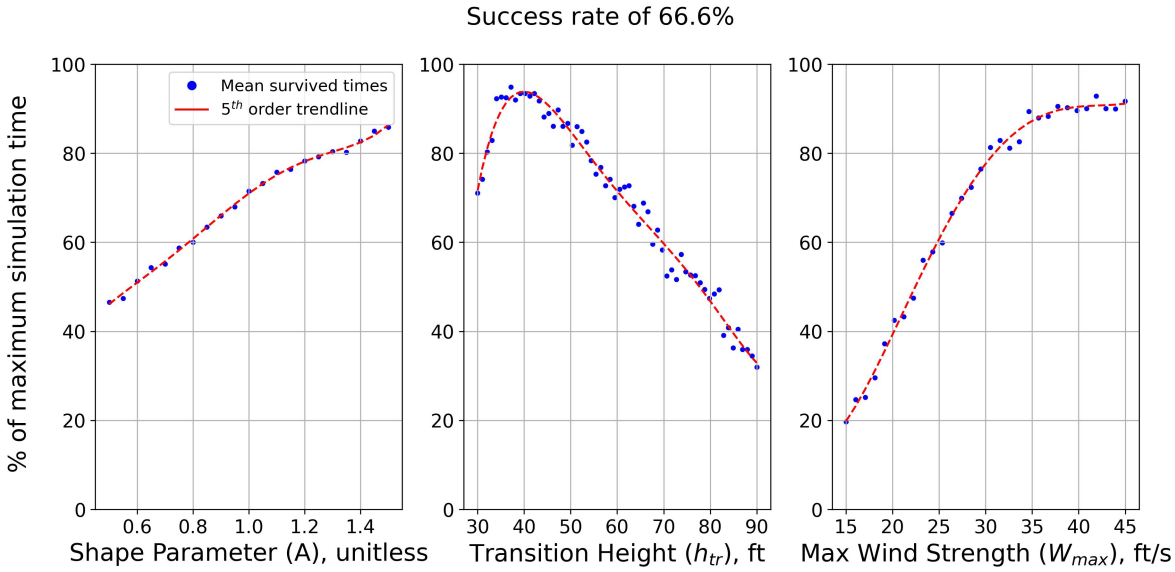


Figure 5.13: Initial wind robustness for initial wind-robust neurocontroller.

maximum wind strength influence the wind magnitude at every point in the altitude range, but it is the transition height that dictates the altitude at which the profile saturates and the gradient terminates. The simplified topology reveals that the majority of robust networks depend on not only the heading angle but also the height to roll the vehicle. The equations of motion show that the vehicle's height is independent of the horizontal wind profile, and therefore, the controllers in the robust group had evolved to soar in an altitude region that guarantees the greatest number of successful flights, regardless of the specific wind environment. Once again, the prevalence of all inputs feeding the lift coefficient command decreased, further supporting the idea that in stochastic environments particularly, there are no specific connections that singly characterize successful soaring. Notably, the initial-wind-robust

group's complexity ratio was 1.03, showing that the average neurocontroller was defined by a 3% greater number of weights and biases with respect to the deterministically-evolved group. Therefore, marginally larger networks were required to encode the properties that characterized the resilience against randomly-initialized wind parameters.

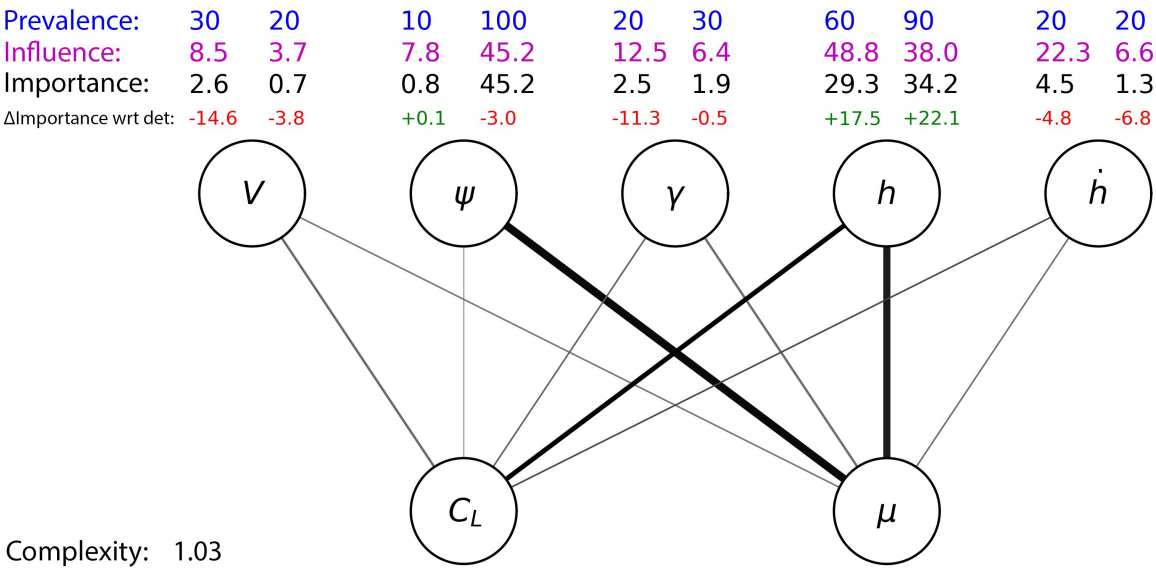


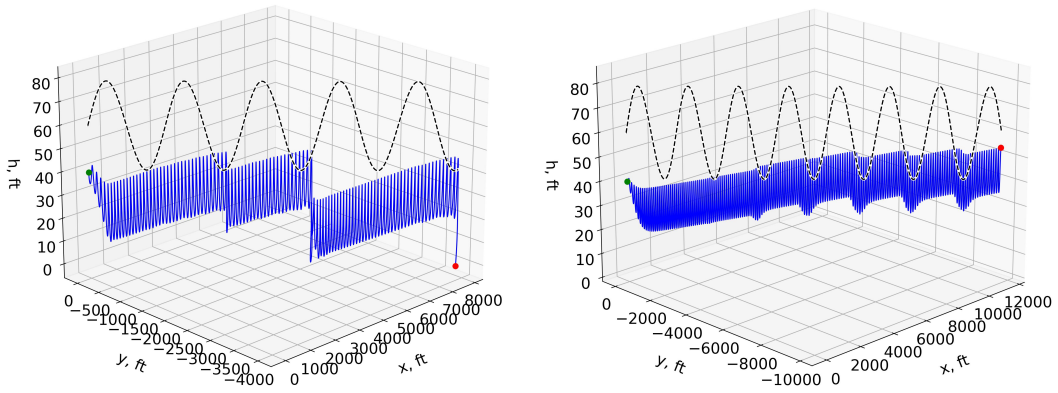
Figure 5.14: Simplified topology of initial-wind-robust neurocontrollers.

5.2.4 Dynamic-Wind-Robust Neurocontroller

For the dynamic-wind-robust group, one of the three wind profile parameters was sinusoidally varied by a randomly sampled amplitude, obtained from a uniform distribution between 0 and 1. Since this factor is multiplied with the varying parameter's nominal value, an amplitude factor of 0 would result in no fluctuation, and a factor of 1 would cause the parameter to swing between zero and double its nominal value. For instance, with a nominal shape parameter of 1.0 and an amplitude factor of 1, the wind profile would periodically transition from a completely exponential shape to a fully logarithmic profile. For the frequency of oscillation, another random value was sampled from a uniform distribution between 0 and 1 Hz.

Unlike the previous environments, the effects of the time-varying wind profile can be easily visualized through the resulting trajectories. Figure 5.15a shows a sample trajectory of one of the deterministically-evolved neurocontrollers along with the environment's sinusoidally changing transition height, whose value is shown by the vertical axis. Figure 5.15b displays a sample trajectory of one of the robust neurocontrollers in an identical environment. The deterministic controller's irregular trajectory contrasts the robust network's much more regular altitude. This difference, corroborated in the more general sense with the success rates of the deterministic and robust groups at 74.2% and 93.4% respectively, explains the effect of the dynamic wind profile and the way in which the robust neurocontroller had evolved to compensate for the variations. As the transition height increases beyond the maximum cycling height, the gradient weakens and the aircraft gains energy at a decreased rate. The trajectory is consequently affected, since the aircraft dynamics are functions of the wind

strength. Similarly, when the transition height falls below a certain threshold and the wind profile is condensed, the vertical gradient changes such that the aircraft extracts more energy per cycle, also influencing the trajectory. It should be stressed that this behaviour is induced by the wind, and is not a deliberate pattern that is actively pursued by the controller. This exemplifies how the evolutionary process produces networks that have evolved to handle such conditions through trial and error. Nevertheless, the nontrivial performance difference between the two groups represents a definite improvement in flight behaviour.



(a) Deterministic (454.2s) (b) Robust (600.0s)

Figure 5.15: Simulated trajectories of deterministic and dynamic-wind-robust neurocontrollers in dynamic wind environment, with trajectory (solid, blue) and transition height (dashed, black).

Both neurocontroller groups attempt to perform standard dynamic soaring, but are affected in different ways by the wind profile due to the robust networks' better-tuned parameters. This is supported by a comparison of the robust group's simplified topology of Fig. 5.16 to that of the deterministic controllers, which shows that the topologies are extremely similar. Once again, all networks relied on the heading-to-roll connection. Simulated flight tests revealed that every network of the dynamic-wind-robust group exhibited trajectories similar to that of Fig. 5.15. Therefore, it is likely that the resulting successful trajectories originate from the only trait common to all networks, which is the heading-to-roll connection. This topological observation, coupled with the complexity ratio of 0.84, suggests that the performance difference between the neurocontroller groups is not necessarily due to added network complexity, but is instead a product of the robust networks' better-tuned weights and biases that encode for dynamic soaring.

5.2.5 Gust-Robust Neurocontroller

The gusty environment consisted of stochastically triggered wind disturbances that occurred in any direction in the three-dimensional Cartesian space, with a maximum horizontal gust component strength of 10 ft/s, a maximum vertical gust strength of 2 ft/s, a maximum duration of 1s, and a 10% probability of occurrence at any given second. Monte Carlo evaluations showcased the significant improvement in the robust group's success rate of 92.8% in contrast to the deterministic networks' 26.4%.

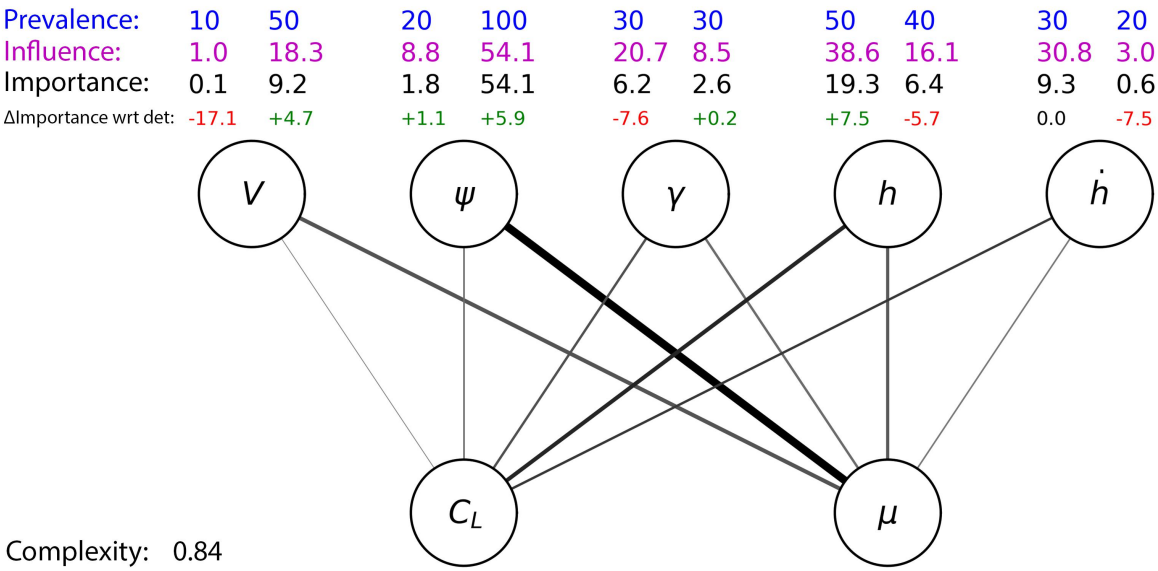
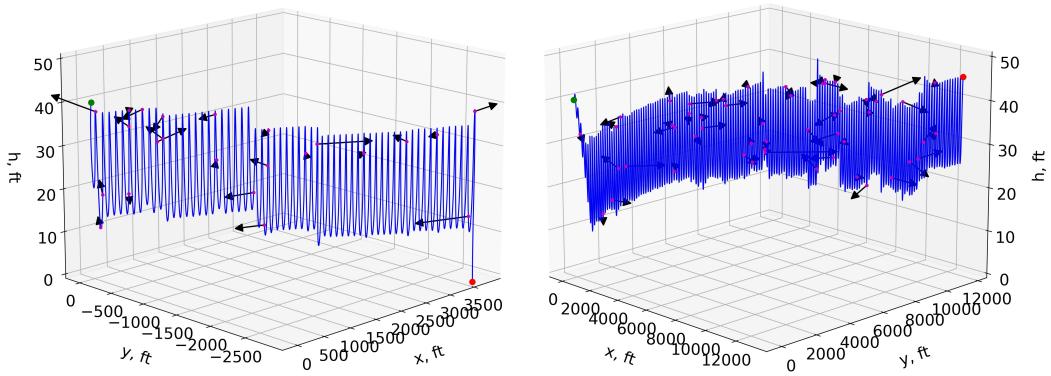


Figure 5.16: Simplified topology of dynamic-wind-robust neurocontrollers.

The effect of the stochastically-occurring gusts can be visualized through sample trajectories of deterministic and gust-robust neurocontrollers, depicted in Fig. 5.17. It can be seen that gusts, whose magnitude and direction are indicated by the length and orientation of the black vectors respectively, decrease the cycling altitude of the trajectory by disturbing the aircraft's energy state. Figure 5.17a shows that after every significant gust event, the aircraft loses potential energy, until the final gust irrevocably perturbs the vehicle and causes it to crash. Contrarily, the robust network is able to withstand the disturbances and remain soaring for the full simulation time.



(a) Deterministic (219.5s, 23 gusts) (b) Robust (600.0s, 53 gusts)

Figure 5.17: Simulated trajectories of deterministic and gust-robust neurocontrollers in gusty environment.

In explaining this difference in performance, the most significant change in the topology of the gust-robust group illustrated in Fig. 5.18 is the lift coefficient's increased reliance on the heading angle and height RoC inputs, which opposes the decreased importance of all other input-to-lift-coefficient connections. The equations of motion presented in Section 3.1.1 show

that unlike the airspeed or pitch angle, the heading angle is independent from the rate of change of the vertical wind gust component $\dot{W}_z$. In addition, the height input is a function of the vertical gust component W_z, and the prevalence of these inputs as variables for the lift coefficient is explained by the limits on the horizontal and vertical gust components. The gusts' horizontal magnitude was sampled from a distribution with an upper limit of 10 ft/s, and the vertical magnitude was assigned a value up to 2 ft/s. Therefore, it was more advantageous for networks to use the heading angle and height RoC variables to compute the lift coefficient, because these inputs were generally less perturbed by each gust. The heading angle dynamics were only dependent on two of the three wind components, and the height RoC was a function of only the vertical wind, which fluctuates less severely than the horizontal components. For the roll command, although the airspeed's value suffered from abrupt fluctuations, its prevalence as an input variable increased when compared to the deterministically-evolved group. Since each state is affected differently by gusts, the network evolved to combine multiple inputs as a way of filtering out the disturbances.

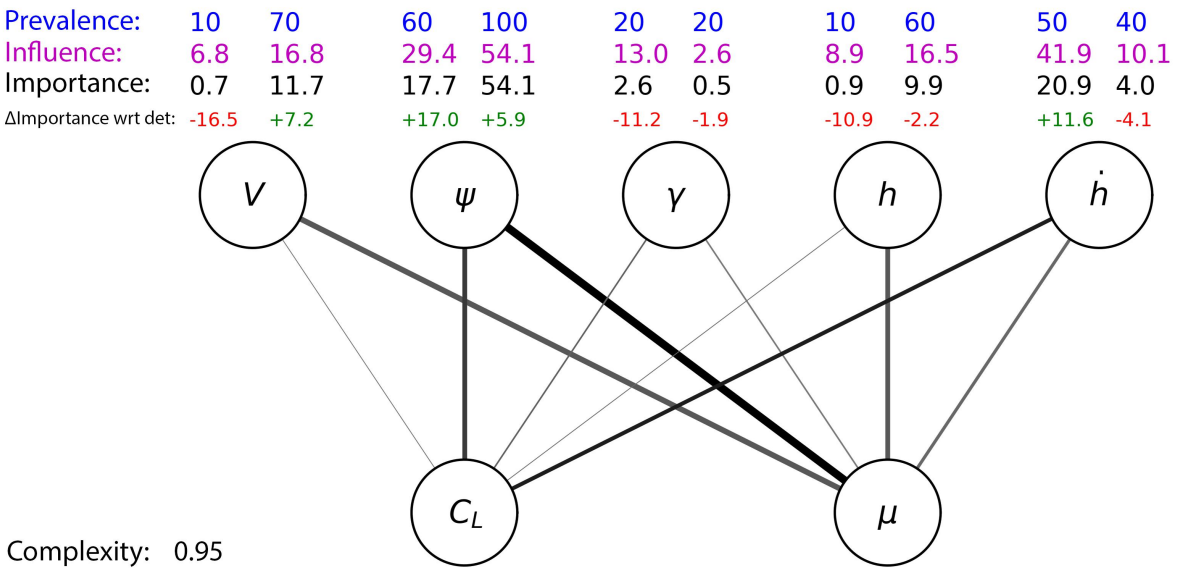


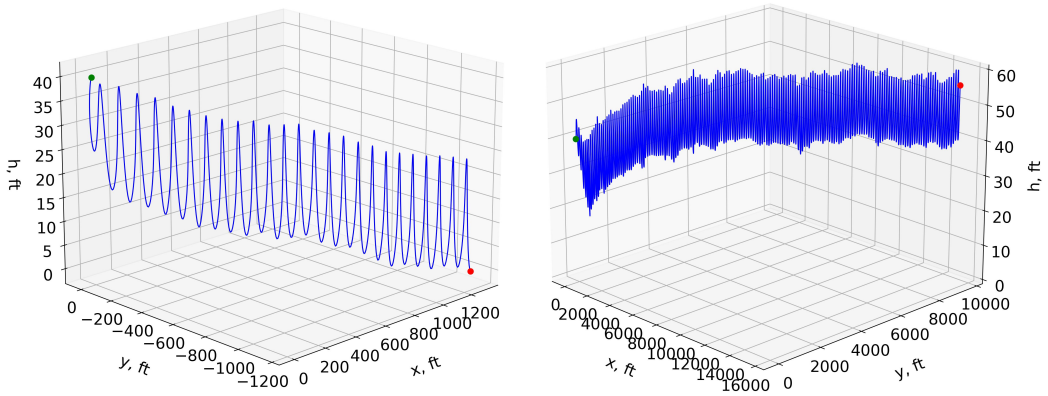
Figure 5.18: Simplified topology of gust-robust neurocontrollers.

5.2.6 Sensor-Noise-Robust Neurocontroller

The sensor-noise-robust neurocontroller group was trained by adding Gaussian noise to the neural network inputs at each timestep. The standard deviations σ for each input variable are listed below.

$$\begin{aligned}\sigma_V &= 5.0 \text{ ft/s}\\ \sigma_\psi &= 5.0 \text{ deg}\\ \sigma_\gamma &= 5.0 \text{ deg}\\ \sigma_h &= 5.0 \text{ ft}\\ \sigma_{\dot{h}} &= 1.0 \text{ ft/s}\end{aligned}$$

Testing showed that the robust group significantly outperformed the deterministically-evolved group in noise-injected environments, resulting in success rates of 94.9% and 27.7% respectively. The effect of measurement noise on the trajectory is exemplified by Fig. 5.19, where it is clear that a sample deterministically-evolved neurocontroller was unable to sustain flight, while one of the robust neurocontrollers performed dynamic soaring for the full simulation time.



(a) Deterministic (90.0s) (b) Robust (600.0s)

Figure 5.19: Simulated trajectories of deterministic and noise-robust neurocontrollers in noisy environment.

The simplified topology of Fig. 5.20 suggests that the decrease in the use of the pitch angle and height RoC inputs is a result of their values periodically approaching zero. As the aircraft levels out mid-cycle or begins to descend after gaining altitude and vice versa, the two states take on extremely small values, which causes any noise to induce greater errors in the states as a percentage of the true value. Therefore, networks that heavily rely on the pitch and height RoC variables would have more difficulty in characterizing the dynamic soaring cycle, causing lower performance.

In addition, the success rate difference between the two groups is likely a consequence of better-tuned network parameters that allow members of the robust group to perform soaring in the presence of measurement noise. This can be seen when comparing the topologies of individual networks in both groups, such as the pair illustrated in Fig. 5.21. Although both neural networks use extremely similar inputs without any hidden layers for the roll angle, the deterministic network's success rate of 0.0% indicates that it is completely unable to remain flying, in contrast to the robust controller's rate of 99.9%, suggesting that the connection weights are the significant factors that affect flight time in the noisy environment. This comparison of topologies and the sensor-noise-robust group's complexity ratio of 0.72 suggest that robustness is first a product of the specific connections that filter for the least-affected input variables, and secondly the result of extensive parameter tuning. While certain neuronal links are important for particular environments, a robust network's weights and biases are also significantly optimized during evolution.

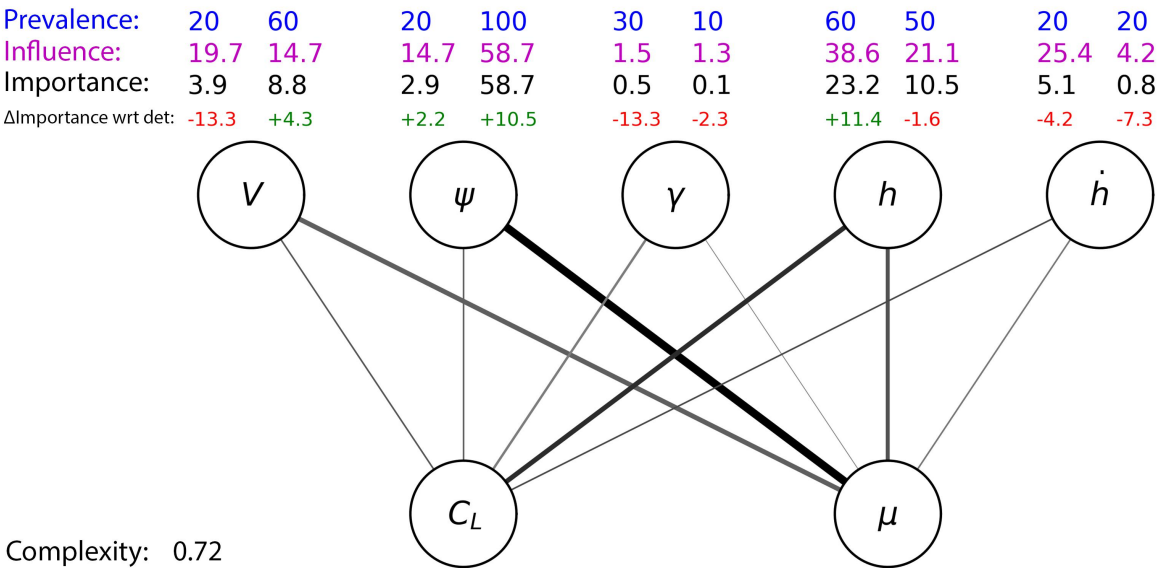
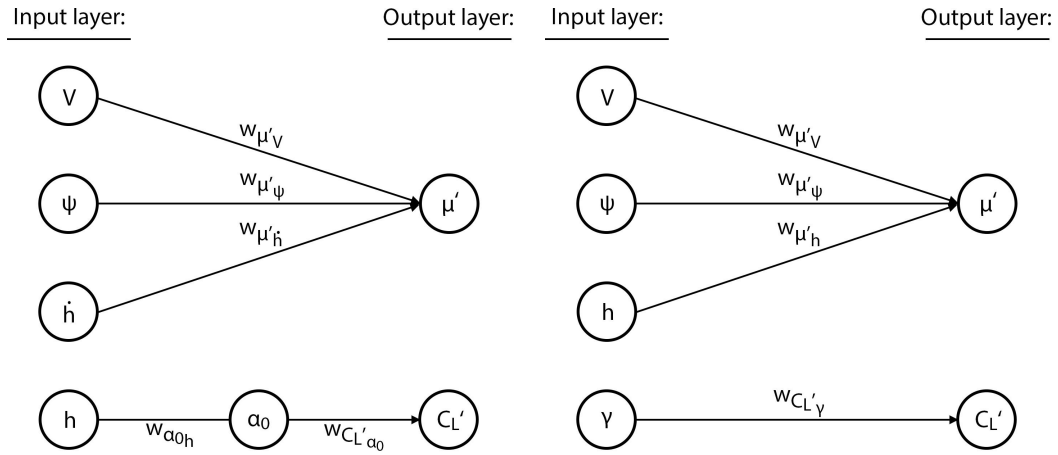


Figure 5.20: Simplified topology of noise-robust neurocontrollers.



(a) Deterministic (success rate of 0.0%) (b) Robust (success rate of 99.9%)

Figure 5.21: Sample deterministic and noise-robust neurocontroller topologies.

5.2.7 Multi-Robust Environment

While the previous sections demonstrate the SNEC method's applicability to individual random environments, real-world implementation of autonomous systems must account for combinations of uncertainties and disturbances. Therefore, this section presents a multi-robust neurocontroller group that was evolved in an environment containing all of the previously examined stochastic elements, which serves to demonstrate the evolutionary approach's ability in instilling robustness to multiple sources of uncertainty in severely varying and disturbance-prone environments.

Monte Carlo evaluations of the multi-robust group demonstrated that while the chaotic environment caused large variances in the topologies between networks, the most notable change was an even greater emphasis on the heading-to-roll connection, supported by a 21%

increase in its importance factor. The necessity of the heading input, which acts as an anchor in characterizing successful dynamic soaring cycles, is further reinforced in the multi-robust environment. Regardless, the weighted average success rates of the deterministic and robust groups were 4.7% and 20.8%, respectively. This marks a significant improvement, particularly when considering again that the highest achievable success rate in the multi-robust environment is unknown and likely less than 100%.

To provide a summarized analysis of the various neurocontroller groups examined in this work, Fig. 5.22 presents the complete matrix of all Monte Carlo evaluations for each neurocontroller-environment combination. Every cell contains the weighted average success rate for the neurocontroller group indicated by the row tested in the environment shown by the column, with the colour value of the cell corresponding also to the success rate. The leftmost column references the results of the deterministically-evolved group (italicized, red), the diagonal elements (blue) represent the native pairings where the neurocontroller group was tested in the stochastic environment for which it was evolved, and the values in bold indicate the best-performing neurocontroller group for each environment.

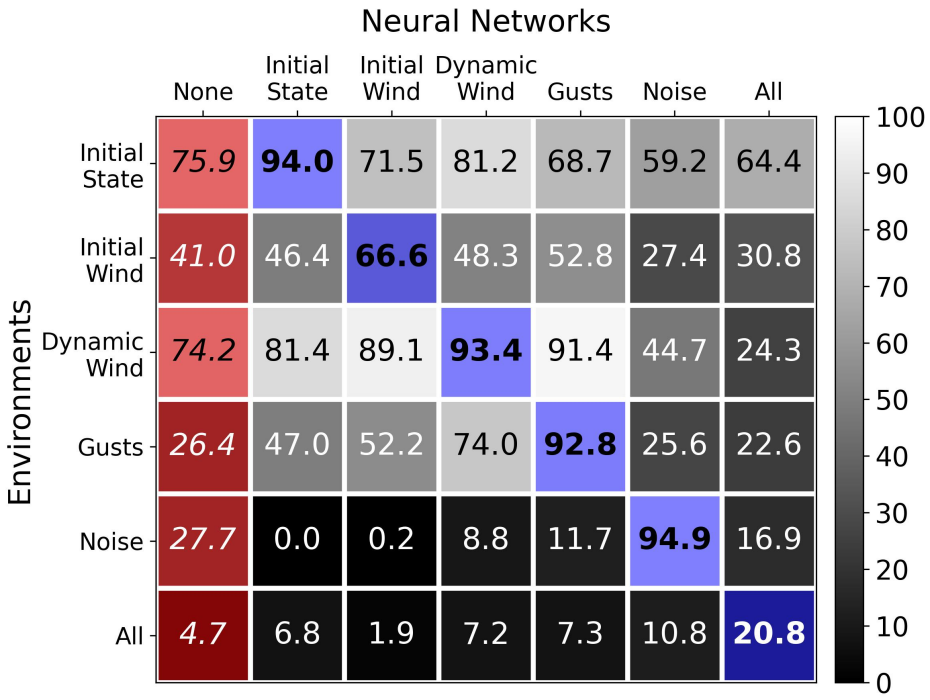


Figure 5.22: Success rates for all neurocontroller-environment combinations.

Despite the multi-robust group's relatively high performance in the chaotic environment, its subpar success rates when evaluated in each individual environment shows that generalizing backwards from the significantly chaotic environment is a challenge. These results suggest that the overwhelming number of variations necessitates more extensive training. Furthermore, the highly specialized success of the noise-robust controllers in their native environment indicates an overfitting of the neural networks in that group, and the colour value of rows provide a sense of each environment's relative difficulty, with the noise and multi-robust environments being the most challenging. Nevertheless, despite these potential shortcomings,

the positioning of the best performing groups on the diagonal shows that the highest success rates were achieved by the neurocontrollers that were tested in the environment for which they were trained. The stochastic neuroevolutionary method is conclusively capable of targeting resilience against specific variations.

In practice, however, a robust network with only a 20.8% success rate in the harshest environments will likely not be the most appropriate neurocontroller for flight in typical conditions. A more nuanced approach would be to evolve a neurocontroller in an environment that mirrors the expected real-world conditions, where the number and severity of uncertainties are reduced to only include those to which the vehicle is most susceptible.

6 Autopilot Implementation

This chapter examines the transferability of evolved knowledge as a precursor to implementing robust SNEC networks on a commercially-available SUAV platform. An open source autopilot software suite for UAVs was used in conjunction with a flight simulator to emulate the performance of a thermal soaring neurocontroller evolved through the neuroevolutionary control process. This initial software implementation considers the thermal soaring problem due to the relative simplicity of the control sequences required to climb thermals. Lastly, a software-in-the-loop (SITL) simulation was conducted to test the software implementation and observe the differences in neurocontroller flight behaviour between the 3DOF training environment and the higher-fidelity flight simulator.

6.1 SITL Framework

The SITL architecture, depicted in Fig. 6.1, consisted of a custom ArduPilot [33] SITL build, the commercial flight simulator X-Plane 10 [34], and the popular ground control station (GCS) Mission Planner [35], all of which communicated through various network protocols.

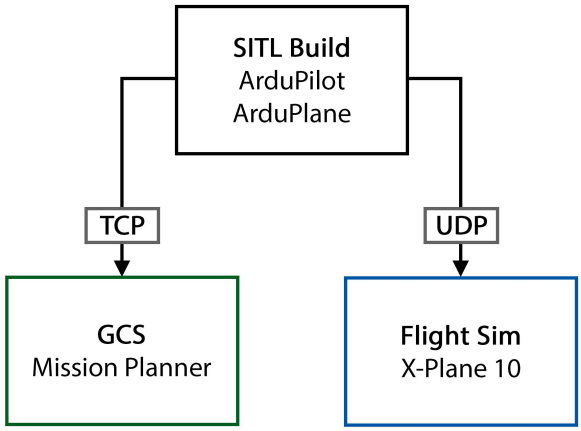


Figure 6.1: SITL software architecture.

To demonstrate specifically the intrathermal behaviour of an SNEC neural network, the preexisting thermalling mode within ArduPilot's base autopilot software was modified such that the neural network logic would only replace the aircraft's roll behaviour when inside a thermal column. This way, the neurocontroller could be tested without disrupting the takeoff, waypoint navigation, and thermal detection routines.

The original autopilot's thermalling mode uses an extended Kalman filter to estimate the vehicle's sink rate, which is then compared to the expected sink rate, obtained from airspeed data and the aircraft's sink polar. The estimated and expected values are used to compute a thermalability factor that dictates whether the autopilot engages the thermalling logic. Upon deciding that soaring is feasible, a loitering radius is calculated, from which the roll angle required to achieve the radius is commanded to the attitude controller.

For the SITL neurocontroller test, the functions for determining the thermalling radius and roll angle were replaced by a neural network generated from the SNEC method. Using aircraft states available in the autopilot software, the target roll angle is sent to the low-level actuator controller directly.

6.2 SITL Models

The aircraft used in the flight simulator was a V-tail RC plane, whose parameters are listed in Table 6.1. To facilitate the transferability of the behaviour trained through the 3DOF simulation to the higher fidelity flight simulator, it was necessary to match the aircraft parameters of the model used for evolution to the vehicle flown in X-Plane. For simplicity, the parameters of the training model were set nearly identically to that used in the previous sections, apart from the wing area and mass, which were calculated from its 3D model.

Table 6.1: X-Plane V-tail SUAV characteristics.

Parameter	Symbol	Value	Units
Wing area	S	4.65	sq ft
Mass	M	0.124	slug
Maximum lift-to-drag ratio	E_{max}	20.0	-
Zero-lift drag coefficient	C_{D_0}	0.025	-
Maximum load factor	n_{max}	5	-
Maximum absolute bank angle	$\|\mu\|_{max}$	60	deg
Stall speed	V_{stall}	22.0	ft/s
Minimum drag speed	V_{md}	26.9	ft/s

Although the thermal wind modelling in X-Plane 10 is not explicitly known, test flights suggested that thermals exist as vertical columns with a single, uniform updraft speed. The columns seemed to persist either until a maximum altitude or until a maximum time. Additionally, while the range or probability distribution of column radii remained unknown, the thermal coverage percent, which is proportional to the number of thermals over a ground surface area, was a user-defined variable that could be set to a value between 0% and 20%. The thermal climb-rate was also configurable to values from 0 to 25 ft/s.

6.3 SITL Neurocontroller Evolution

The neurocontroller implemented in the autopilot was evolved in the 3DOF environment through the SNEC process, where the initial aircraft conditions and the thermal wind parameters were randomized. For the aircraft states, the airspeed and pitch angle were initialized using the uniform distribution detailed in Section 5.1.2. Furthermore, to match the flight simulator's thermal model, which lacked the outer downdraft region of the mathematical model described in Section 3.3.2, the vehicle's initial position on the horizontal plane was set to a point in the half-ring defined by the thermal column's updraft region that has a thickness of one-half the column radius, with the heading angle set to point the aircraft towards the other half of the disk. This way, the vehicle would enter different parts of the thermal column, and successfully soaring agents would remain in the updraft region without experiencing any downward wind. The initialization scheme is depicted in Fig. 6.2, and the stochastic parameter space is detailed in Table 6.2.

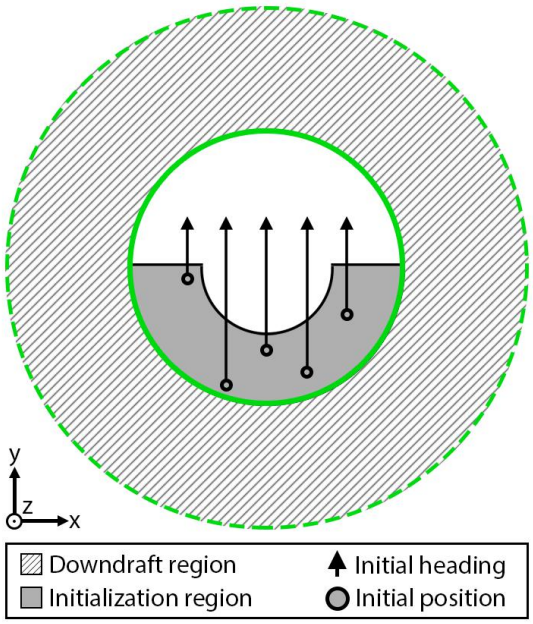


Figure 6.2: Initialization domain of the SITL random-initial-state environment.

Table 6.2: Thermal soaring SITL simulation parameters.

Wind parameters	Range	Initial conditions	Range
w_{core} [ft/s]	[5.0, 15.0.]	V_0 [ft/s]	[20.2, 33.6]
r_{xy} [ft]	[50.0, 150.0]	ψ_0 [deg]	0.0
$h_{tr_{max}}$ [ft]	∞	γ_0 [deg]	[-45.0, 45.0]
$t_{th_{max}}$ [s]	∞	h_0 [ft]	100
		x_0 [ft]	(Inside core updraft disk)
		y_0 [ft]	(Inside core updraft disk)

The inputs were reduced to only include the airspeed, pitch angle, and climb rate, as their values were readily available in the ArduPilot source code. The control output was also limited to a single dimension by fixing the lift coefficient to a constant due to ambiguity in the X-Plane aircraft model's lift polar. The fitness function was identical to the formulation described in Section 4.3.1, where the reward is proportional to the change in potential energy. Due to the reduced state and action spaces, the resulting neural network of Fig. 6.3 defined entirely by the six parameters listed below was evolved with only 20 generations, each with 10 stochastic flights.

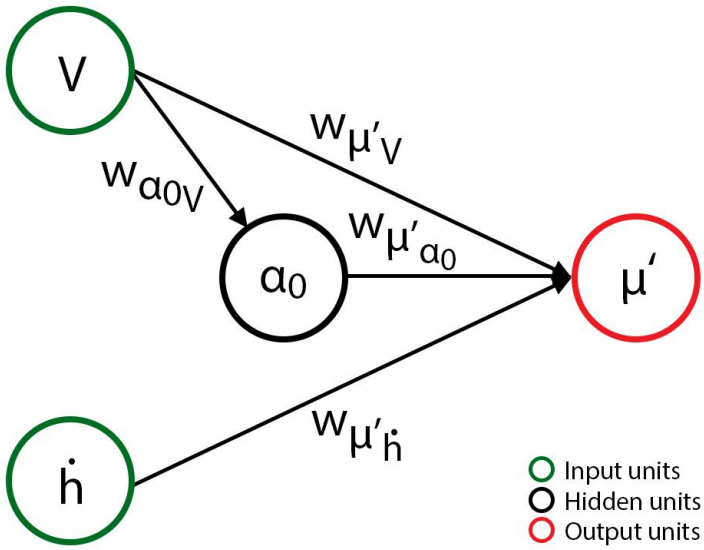


Figure 6.3: Topology of the robust thermal soaring neurocontroller.

The node biases and connection weights are:

$$b_{\mu'} = -1.16 \qquad w_{\mu'_V} = 0.262$$
$$b_{\alpha_0} = -0.845 \qquad w_{\alpha_{0V}} = -0.197$$
$$w_{\mu'_{\alpha_0}} = 0.430$$
$$w_{\mu'_{\dot{h}}} = 0.164$$

While experiencing only 10 randomly-initialized environments may seem insufficient in evaluating successful soaring performance for the full range of varied parameters, the rapid training process took only 0.06 CPU seconds and provided consistent thermal soaring trajectories. Five flights, each with a constant core radius of 100ft but randomly sampled updraft strength, are shown in Fig. 6.4. The simulations lasted for 600s, and demonstrate how the neurocontroller tends to increase the aircraft's roll angle as the wind speed increases. This behaviour can be deduced from the network's topology and weights, where there is a proportional relationship between the height RoC and the roll command.

A typical thermalling technique that is well-known to glider pilots is the Reichmann method [11], where the aircraft's bank angle is set to be inversely proportional to the climb rate. For instance, when experiencing a low rate of climb, a steeper roll angle is favoured so that the aircraft remains near the thermal centre, where the updraft is strongest. However,

SUAVs are typically much smaller than manned gliders, which allows for greater manoeuvrability within thermal columns and less of a concern of exiting the updraft region. Contrary to the Reichmann heuristics, the neurocontroller evolved to increase its roll angle proportionally to the height RoC, because the fitness function rewarded gains in potential energy. The aircraft can maximize its altitude by flying with a greater velocity at a steeper roll angle than with a lower velocity at a shallower roll angle, since the climb rate is a function of the wind velocity, which is strongest at the thermal centre.

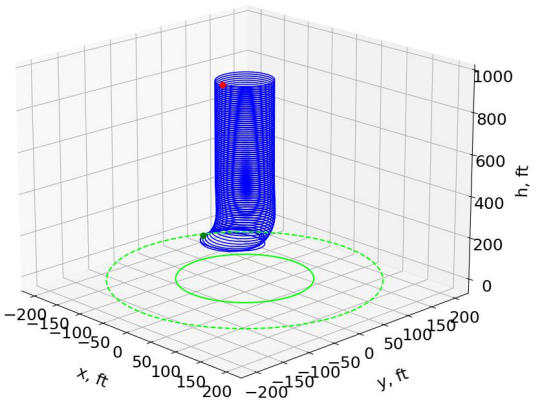
Interestingly, the output's dependence on the airspeed and the negative connection weight between the airspeed input and the hidden node accounts for the spiralling of the trajectory at the highest updraft magnitude, which is shown Fig. 6.4e. Beyond a certain airspeed, the hidden node's effect on the roll angle becomes greater than the direct relationship between the updraft strength and roll command, which results in regular fluctuations in the bank angle during a single cycle and causes translational motion. Regardless, the neurocontroller's robustness score in the hybrid stochastic environment was 88.14% after 1609 Monte Carlo simulations, which was anticipated to be sufficient for the SITL test.

Since the ArduPilot software's existing autopilot program would manage the interthermal wind mapping and thermal mode activation logic, and because the goal of the SITL simulation was to validate the ability to implement a SNEC network in a higher fidelity simulation, the rapidly-trained, minimal network of Fig. 6.3 was deemed sufficient.

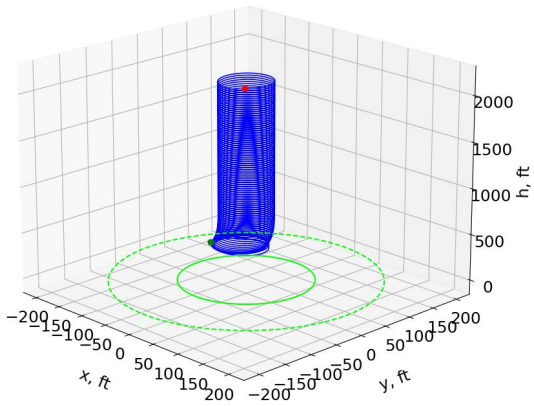
6.3.1 SITL Flight Simulation

The neurocontroller trained in the 3DOF environment was then directly integrated without further tuning into the autopilot system of the X-Plane simulator's V-tail aircraft SITL flight testing. The aircraft model was winch-launched from a runway and allowed to climb to a minimum altitude before the vehicle's motor was disabled and the neurocontroller thermalling mode was engaged. Therefore, without exploiting thermal updrafts, the SUAV, with no means of generating thrust, would be unable to remain flying. A series of waypoints guided the aircraft's interthermal heading, and while the thermal column coverage was set to the maximum value of 25% for the entirety of the simulation, the updraft wind speed was varied for each leg of the flight path. Figure 6.5 shows the resulting 50-minute trajectory overlaid on a satellite map of the region, with numbered legs that correspond to the wind speeds listed in Table 6.3, which match the parameters used in Fig. 6.4. Additionally, Figures 6.6 and 6.7 provide three-dimensional views of the same trajectory. The aircraft's high-level mission objective is to reach each waypoint, but upon detection of a thermal column through the Kalman filtering mechanism described in Section 6.1, the vehicle enters the thermalling logic defined by the neurocontroller. After the wind estimation system determines the loss of the thermal region, the aircraft's heading is set by the mission controller such that the vehicle glides towards the next waypoint.

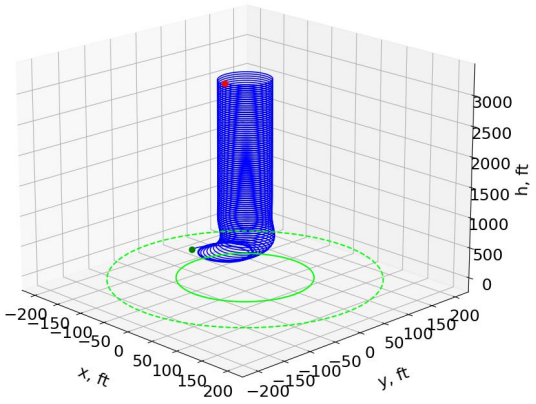
The trajectory shows that the neural network was able to continuously soar for 50 minutes by controlling the vehicle's roll angle. Telemetry and data logs extracted from the simulator and the GCS showed that when inside a thermal region, the neurocontroller provided commands at a frequency of approximately 50 Hz, or every 0.02s of simulation time, providing also a benchmark for the 3DOF training simulator. Furthermore, the extracted flight trace shown in Fig. 6.8 shows the aircraft gaining potential energy during every soaring event,



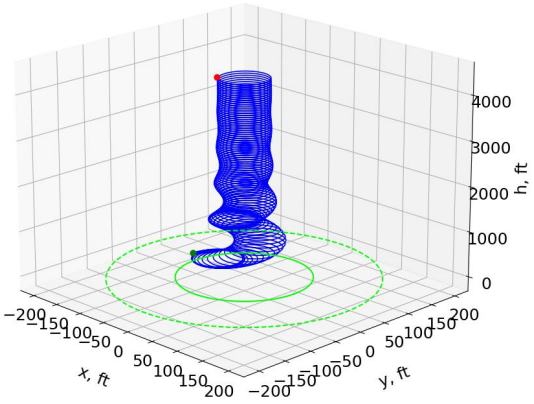
(a) $W_{core} = 5.0$, $\bar{V} = 34.7$, $\bar{\mu} = 41.5$



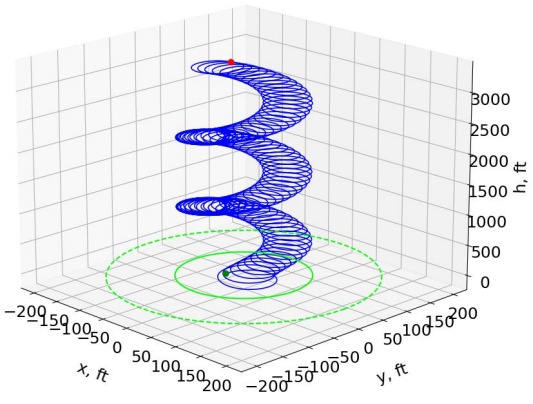
(b) $W_{core} = 7.5$, $\bar{V} = 35.3$, $\bar{\mu} = 43.6$



(c) $W_{core} = 10.0$, $\bar{V} = 35.9$, $\bar{\mu} = 45.5$



(d) $W_{core} = 12.5$, $\bar{V} = 36.4$, $\bar{\mu} = 47.2$



(e) $W_{core} = 15.0$, $\bar{V} = 36.7$, $\bar{\mu} = 45.8$

Figure 6.4: Simulated trajectories of the robust thermal soaring neurocontroller, with core updraft strength (ft/s), mean airspeed (ft/s), and mean roll (deg).

which is exchanged for kinetic energy immediately after leaving the updraft as the aircraft pitches downwards and gains airspeed.

To numerically verify and interpret the neurocontroller's behaviour, Fig. 6.9 shows the input and output traces of the neural network. The topmost numbers correspond to each leg of the flight, which are inferred from the heading plot and delineated by the dotted, black,

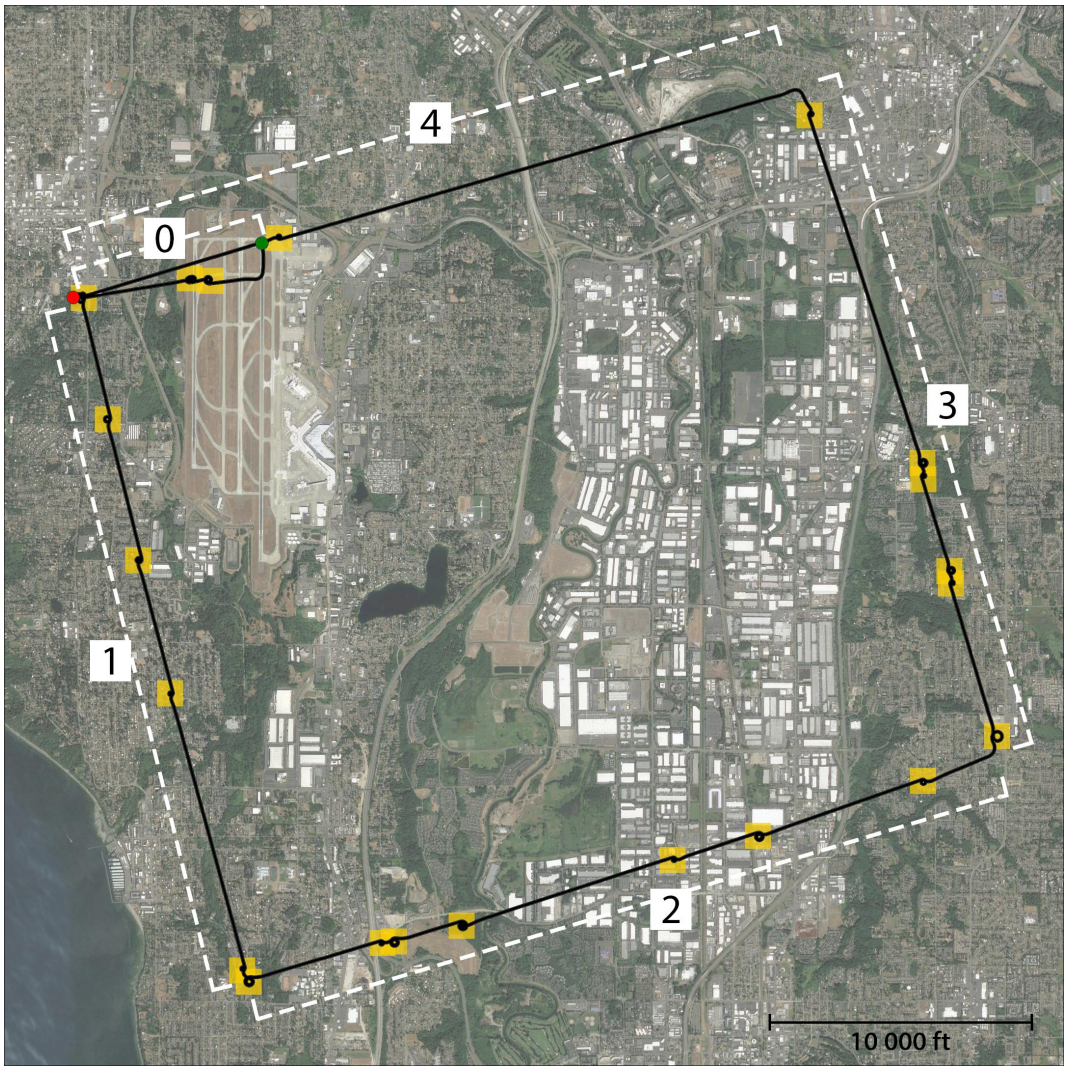


Figure 6.5: Thermal soaring SITL flight trajectory, with start marker (green), end marker (red), highlights of soaring events (yellow), and legs (white).

vertical lines. The airspeed, pitch angle, and climb rate traces for the entire trajectory are also illustrated, where the segments corresponding to soaring events are emphasized and the maximum values of the states across each thermal occurrence within individual legs are marked by the dashed red lines. Finally, the last two plots contain the traces of the roll commands sent to the low-level attitude controller and the vehicle's actual, or experienced, roll angle. While the heading and experienced roll plots were obtained from telemetry logs, the other states were extracted from the onboard autopilot program's data logs, and therefore show the values as seen by the neurocontroller.

The traces indicate a significant correlation between the airspeed and the commanded roll, which shows how the neural network implemented within the autopilot system was functioning as intended. However, the range of commanded roll values was much smaller than when the neurocontroller was tested in the 3DOF model. This is due to the large airspeed values of the V-tail aircraft that were not experienced during evolution, which biased the network

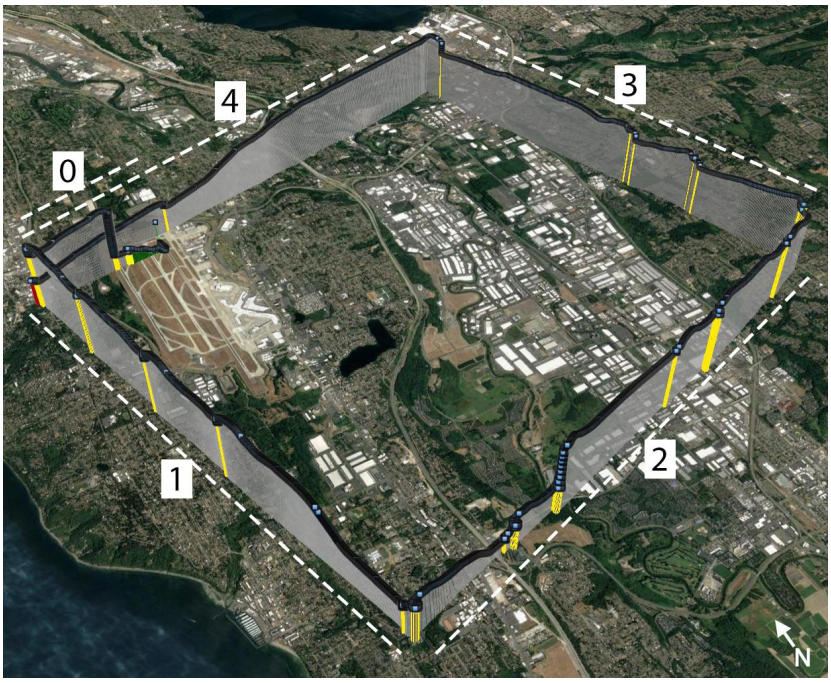


Figure 6.6: Thermal soaring SITL flight trajectory - 3D overview.

Table 6.3: Thermal soaring SITL wind speeds.

Leg	Updraft strength (ft/s)
0	10.0
1	12.5
2	15.0
3	7.5
4	5.0

outputs. The increase in the values entering the neural network's output node saturated the nonlinear activation function, resulting in roll commands that approached the maximum threshold of 60.0 degrees. This phenomenon can be best observed in leg 2 of the trajectory, where despite the various airspeed and climb values seen by the neurocontroller, the network's output consistently reached its predefined limit.

In addition, there exists a discrepancy between the commanded and experienced roll angles, which can be more easily seen in Fig. 6.10. Only the values when the aircraft was in the thermalling manoeuvre are shown, which was the only time that commands were sent to the attitude controller. The plot reveals that the vehicle's bank angle, which lags behind the commanded value, overshoots the desired bank angle before oscillating near the set point. Although this behaviour is not ideal, it can be remedied by tuning the autopilot's actuator controller. Nevertheless, the general trend of the experienced roll follows that of the neurocontroller's outputs, supporting the implementation of SNEC-based neural networks on real-time systems.

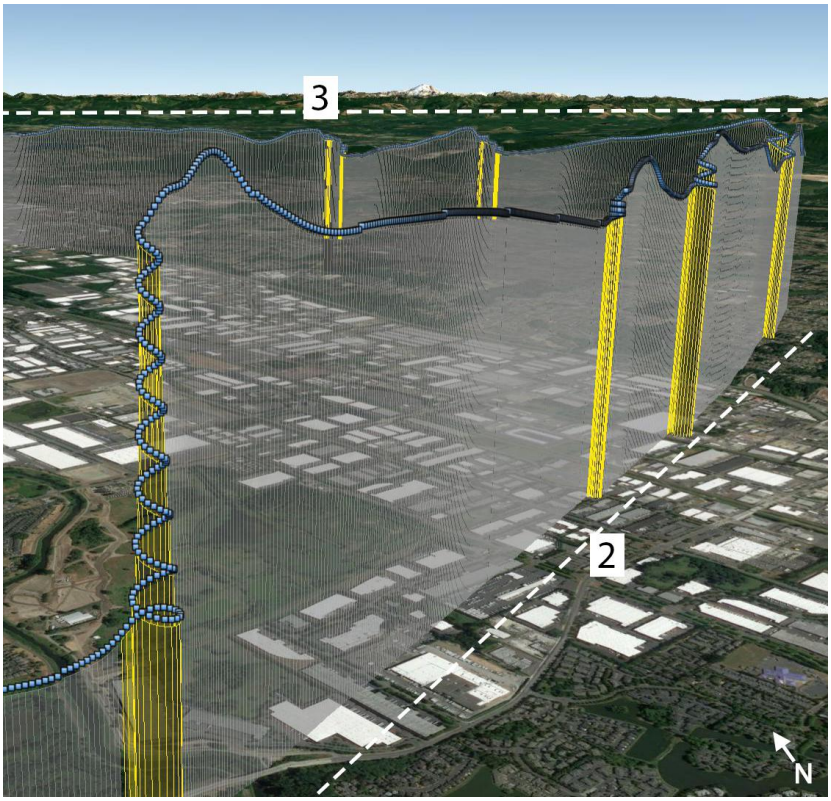


Figure 6.7: Thermal soaring SITL flight trajectory - 3D climbing path.

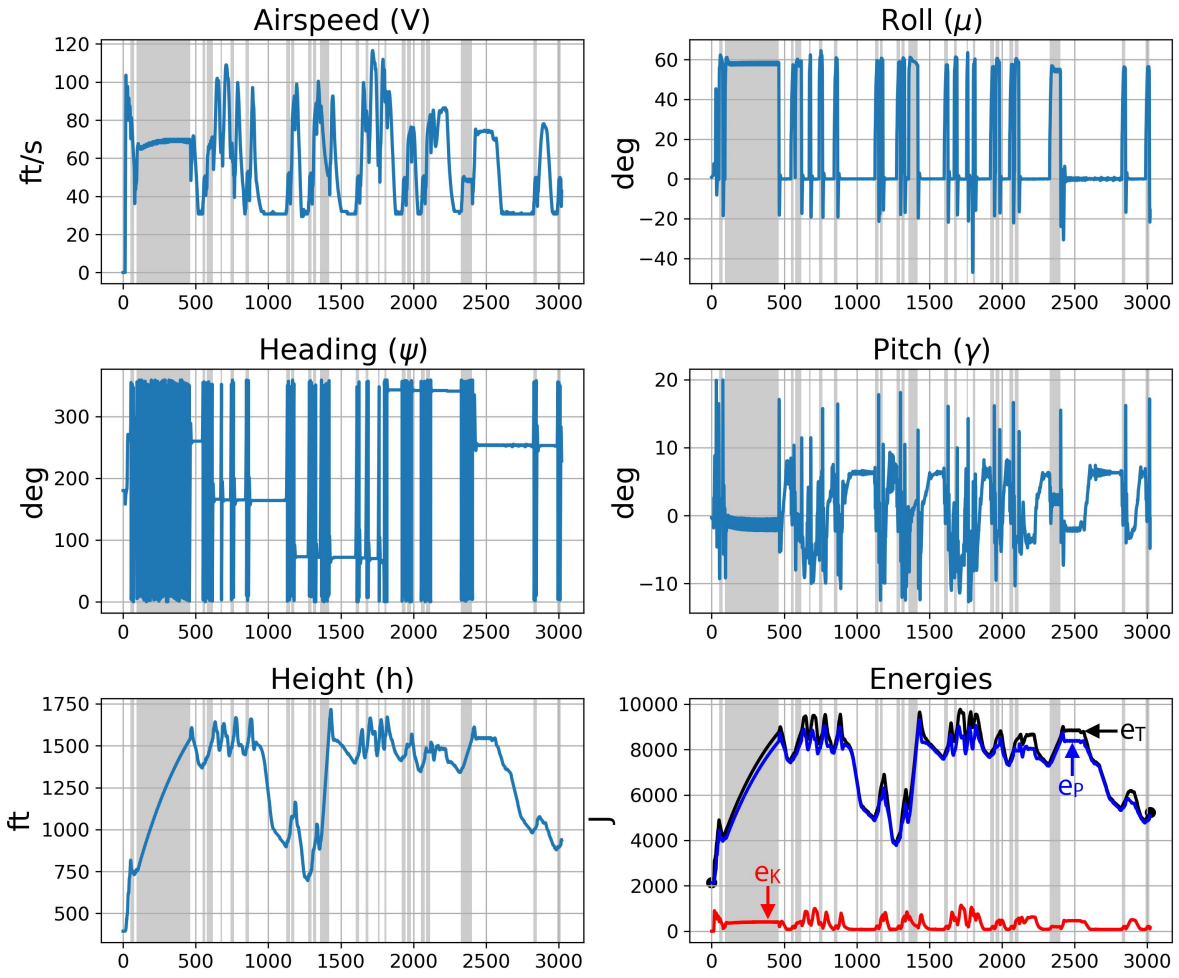


Figure 6.8: Thermal soaring SITL flight trace with respect to time (s), including kinetic e_K, potential e_P, and total e_T energies.

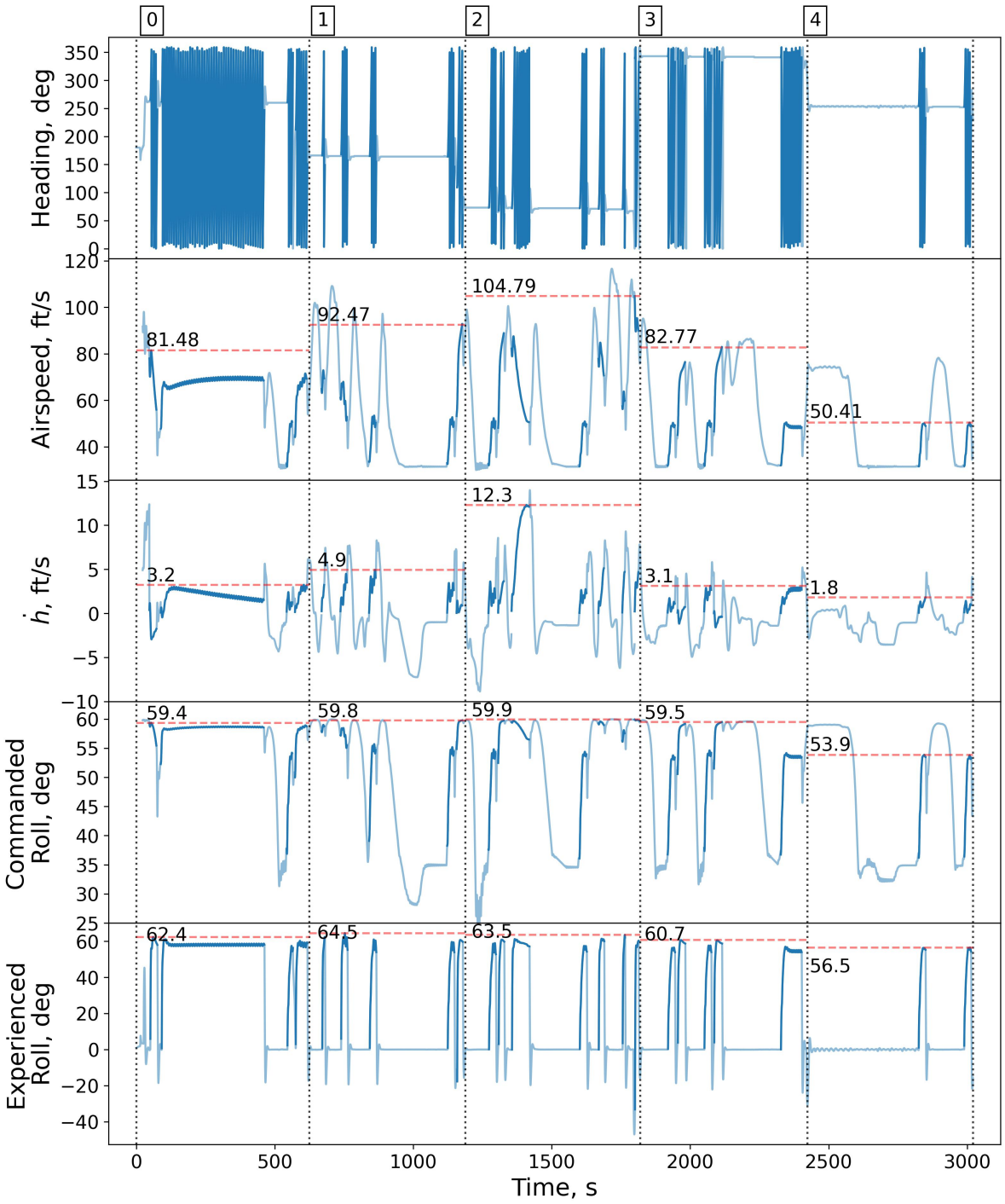


Figure 6.9: Thermal soaring SITL ANN signal trace.

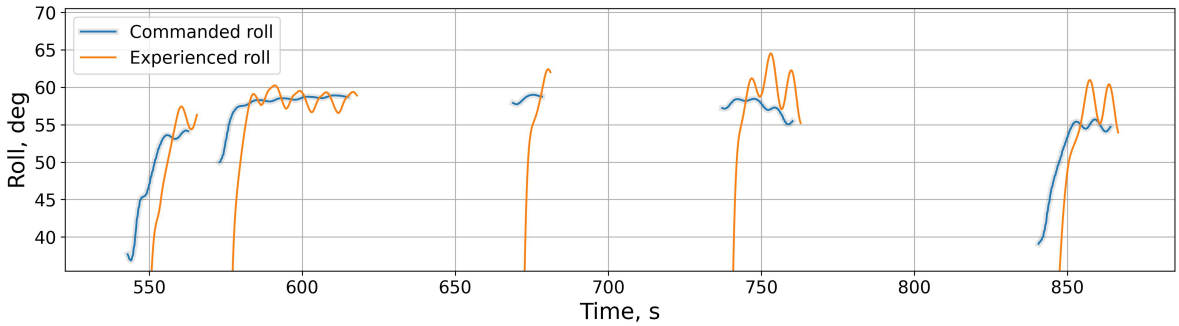


Figure 6.10: Thermal soaring SITL roll angles.

7 Conclusions and Recommendations

The aim of this research was to investigate, design, and validate an artificial-neural-network-based control approach for autonomous soaring that can be implemented on small unmanned aerial vehicles for a variety of stochastic environments. This final chapter will provide concluding remarks, suggestions for improvement, and recommendations for future research on this topic.

7.1 Conclusions

This research is the first study in the current body of knowledge on autonomous aircraft control that examines a neuroevolutionary method of generating simple, effective, and robust neurocontrollers. Initially, the cart pole balancing task was used to explore the feasibility of applying the Neuroevolution of Augmenting Topologies algorithm to control tasks. The extremely simple yet effective neural networks that resulted from the experiment suggested that a similar approach could be applied to autonomous soaring, whose trajectories are also characterized by repetitive signals. Application of a reinforcement learning algorithm to the same problem revealed the advantages of evolved, irregular neural networks over large, fixed-topology networks. Despite requiring a vastly greater amount of computational power, the reinforcement learning technique unsuccessfully trained a 32-unit network, exemplifying the method's inefficiency in mapping continuous action and state spaces, its increased complexity, and also its reduced interpretability.

With this initial experiment validating the potential applicability of evolutionary neural networks, the neuroevolutionary control approach was developed. The fitness function that guides the training process was designed to discourage flight agents from exceeding boundary constraints that would otherwise be infeasible or catastrophic to the vehicle, while encouraging certain behaviours by maximizing rewards that were proxies for successful soaring. The goal was to loosely direct the evolution of soaring manoeuvres without an extensive artificial restricting of the search space so that the process ultimately discovers a general solution that is not explicitly engineered by the designer. The creation of the simulation environment along with the aircraft and wind models allowed for its integration with the NEAT algorithm, defining the NEC process.

The travelling dynamic soaring case was then used to compare the evolved trajectories to that of numerical optimization. The policy that was encoded in the neurocontroller was found to be similar to the canonical reference, where the flight agent learned to initially orient itself into the wind profile to gain potential energy before exchanging it for kinetic energy by descending the gradient. However, the neurocontroller also learned to change its behaviour

from maximizing energy gain to maximizing flight distance after accumulating sufficient total energy for continued soaring. This demonstrated the NEC method's ability to achieve the higher-level objective of sustained soaring, which contrasts the limited-time-horizon trajectories of numerical optimization techniques. The NEC approach also required minimal computational time to generate simple dynamic, thermal, and ridge soaring neurocontrollers whose outputs could be traced backwards to reconstruct the internal network signals and determine the relative importance of certain inputs. However, it was also discovered that the process suffers from large action spaces, where the greater number of possible state and control trajectories necessitates longer evolutionary periods and more restrictive conditions. In the ridge soaring case, this was overcome by reducing the space of allowable states. Regardless, the various soaring cases demonstrated the NEC process' capacity to find solutions to different soaring problems.

Subsequently, the necessity for robust behaviour prompted the design and development of the stochastic neuroevolutionary approach. Various models defined environments with random initial conditions, time-varying wind profiles, unexpected wind gusts, and Gaussian sensor noise for the travelling dynamic soaring problem. The method of randomizing the training domain during evolution resulted in neural networks that indirectly encoded the experiences of its ancestors to ultimately exhibit resilience to the random environment. The Monte Carlo estimation process also enabled the quantification of robustness so that neurocontrollers could be compared relative to each other. To account for the individual variances between neural networks trained in identical environments, the analysis presented groups of multiple neurocontrollers, whose performances were averaged to obtain a more representative assessment of the SNEC technique's efficacy. Interestingly, not only did the performances of the stochastically-evolved networks show clear evidence of robustness, where in some instances, robust neurocontroller groups were capable of sustained soaring at a rate 4.4 times that of deterministically-evolved neural networks, but their topologies also revealed consistent patterns both between and within neurocontroller groups. The analytical metrics and network reduction method that were developed to examine the role of network structure on robustness indicated how the SNEC process, after evolving the fundamental connections that are required to define the soaring manoeuvres, tuned the network's weights and biases to achieve resilience instead of continuing to add topological complexity. This suggests that network depth, a characteristic of deep neural networks, may not necessarily be required to encode robust flight behaviour. Additionally, the topological analysis allowed for the explanation of specific, recurring connections, leading to an intuitive understanding of the neural networks that may eventually be valuable for industrial or commercial certification, and further supporting their applicability over fixed-topology ANNs.

Finally, the simplicity of the SNEC-trained networks enabled the software implementation of a thermal soaring neurocontroller for a software-in-the-loop experiment using an open source autopilot platform. Despite the modelling differences between the commercial flight simulator and the 3DOF environment used for training, the general behaviours that were learned in the lower-fidelity simulation transferred to the more advanced simulator.

7.2 Recommendations for Future Work

Throughout this research, a number of potential areas for future development were identified that, if implemented, would contribute to the overall objective of operational autonomous flight.

While this work explored the trajectories of travelling dynamic soaring, intrathermal soaring, and ridge soaring, there also exist other techniques such as loitering dynamic soaring and interthermal soaring that can offer different use cases for autonomous aircraft. For instance, loitering dynamic soaring trajectories would require a reformulation of the fitness function's reward mechanism to prioritize geographically local, energy-neutral flight. On the contrary, interthermal soaring is a higher-level task that necessitates either the identification and tracking of multiple thermals or an exploratory method of seeking updraft regions. To evolve more complex behaviours beyond the predictable, repetitive trajectories demonstrated in this research, it is recommended that the input space accessed by the neurocontroller is sufficiently large. As an example, for the travelling dynamic soaring case, the sparse set of attitude and inertial data consisting of the airspeed, heading, pitch, height, and height RoC was found to be adequate for the neural network to learn successful trajectories. Similarly, the ridge soaring neurocontroller used additional positional data to characterize its trajectory. However, one significant dimension that remains to be explored is time. Expanding the input space by simply adding a clock variable that allows for time-dependent behaviour is the evident approach, but the more interesting method of incorporating memory through recurrent or long-short-term-memory neural networks is recommended. While the input space used in this work did not explicitly provide the neurocontrollers any temporal knowledge of the vehicle or environment, such topologies and mechanisms may produce solutions that are encoded in time, opening another dimension along which evolution can occur.

Another recommendation is the exploration of other variations and uncertainties in the evolutionary environment. The presence of signal noise in a neurocontroller's output commands that are sent to the actuator controller is a possibility in real systems, and can be simulated through Gaussian distributions. Robustness experiments of neural networks trained in such an environment may reveal the degree of attitude control accuracy that is required for sustained soaring. In addition, control delay and low control frequencies may also be obstacles in SUAV platforms, which are defined by numerous subsystems. The outputs of neurocontrollers will inevitably be delayed as the signals are passed between autopilot software modules and eventually to the attitude controller. Similarly, while the neural networks presented in this work were evolved to output controls at a frequency of 10 Hz, which were successfully interpolated at 50 Hz in the SITL thermal soaring experiment, an examination of the control sequence's extrapolation to lower frequencies is recommended as well. Another area of robustness that remains to be explored is system failures, comprised of, for instance, variations in sensor availability and changes in physical aircraft characteristics. An environment could be modelled to simulate the sudden loss of the neurocontroller inputs to evolve a system that is resilient against internal signal corruption or failure. An ablation study on the input space would also be useful in determining the most necessary connections used in characterizing different types of soaring trajectories. An abrupt change in the physical state of the aircraft is another scenario for which robustness may be trained by altering the system's dynamics in the evolutionary environment.

The final recommendation for future research is the continued reduction of the gap between simulation training and physical implementation. Further experiments in the SITL environment for dynamic or interthermal soaring are required prior to integrating neurocontrollers on real-world systems. The SITL experiment revealed that there exists discrepancies when transferring learned behaviours from the 3DOF environment to higher-fidelity models, due to differences in dynamics as well as aircraft and wind parameters. As a result, future work on the tuning of training environments to better align with the expected reality is recommended, where the domain randomization technique of the SNEC approach can also be applied to bridge the sim-to-real gap. For instance, marginally varying the control limits and physical properties of the training vehicle between evolutionary episodes would effectively decouple the desired flight behaviour from the specific vehicle, and facilitate the real-world implementation of autonomous soaring systems.

Bibliography

[1] Gottfried Sachs. Minimum shear wind strength required for dynamic soaring of albatrosses. *Ibis*, 147(1):1–10, 2005.

[2] C. J. Pennycuick. Thermal Soaring Compared in Three Dissimilar Tropical Bird Species, Fregata Magnificens, Pelecanus Occidentals and Coragyps Atratus. *Journal of Experimental Biology*, 102(1):307–325, January 1983.

[3] Nicholas R. J. Lawrance and Salah Sukkarieh. A guidance and control strategy for dynamic soaring with a gliding UAV. In *2009 IEEE International Conference on Robotics and Automation*, pages 3632–3637, Kobe, Japan, May 2009.

[4] Jack W. Langelaan, John Spletzer, Corey Montella, and Joachim Grenestedt. Wind field estimation for autonomous dynamic soaring. In *2012 IEEE International Conference on Robotics and Automation*, pages 16–22, May 2012.

[5] Michael Allen and Victor Lin. Guidance and Control of an Autonomous Soaring Vehicle with Flight Test Results. In *45th AIAA Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, June 2012.

[6] Nathan T. Depenbusch, John J. Bird, and Jack W. Langelaan. The AutoSOAR autonomous soaring aircraft, part 1: Autonomy algorithms. *Journal of Field Robotics*, 35(6):868–889, 2018.

[7] Rayleigh. The Soaring of Birds. *Nature*, 27(701):534–535, April 1883.

[8] Yiyuan J. Zhao. Optimal patterns of glider dynamic soaring. *Optimal Control Applications and Methods*, 25(2):67–89, 2004.

[9] John J. Bird, Jack W. Langelaan, Corey Montella, John Spletzer, and Joachim L. Grenestedt. Closing the Loop in Dynamic Soaring. In *AIAA Guidance, Navigation, and Control Conference*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, January 2014.

[10] Zhenda Li and Jack W. Langelaan. Parameterized Trajectory Planning for Dynamic Soaring. In *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, Orlando, FL, USA, January 2020.

[11] Helmut Reichmann. *Cross Country Soaring*. Soaring Society of America, 1993.

[12] Klas Andersson, Isaac Kaminer, Vladimir Dobrokhodov, and Venanzio Cichella. Thermal Centering Control for Autonomous Soaring; Stability Analysis and Flight Test Results. *Journal of Guidance, Control, and Dynamics*, 35(3):963–975, May 2012.

[13] J. M. Wharington. Heuristic control of dynamic soaring. In *2004 5th Asian Control Conference (IEEE Cat. No.04EX904)*, volume 2, pages 714–722 Vol.2, July 2004.

[14] N Akhtar, J F Whidborne, and A K Cooke. Real-time optimal techniques for unmanned air vehicles fuel saving. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 226(10):1315–1328, October 2012.

[15] M. Deittert, A. G. Richards, C. Toomer, and A. Pipe. Dynamic Soaring Flight in Turbulence. *AIAA Guidance Navigation and Control Conference, Chicago*, August 2009.

[16] Yuyi Liu, Stefano Longo, and Eric C. Kerrigan. Nonlinear predictive control of autonomous soaring UAVs using 3DOF models. In *2013 European Control Conference (ECC)*, pages 1365–1370, July 2013.

[17] Renaud Barate, Stéphane Doncieux, and Jean-Arcady Meyer. Design of a bio-inspired controller for dynamic soaring in a simulated unmanned aerial vehicle. *Bioinspiration & Biomimetics*, 1(3):76–88, September 2006.

[18] C. Montella and J. R. Spletzer. Reinforcement learning for autonomous dynamic soaring in shear winds. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3423–3428, Chicago, IL, USA, September 2014.

[19] Seong-hun Kim, Jihoon Lee, Seungyun Jung, Hanna Lee, and Youdan Kim. Deep Neural Network-Based Feedback Control for Dynamic Soaring of Unpowered Aircraft. *IFAC-PapersOnLine*, 52(12):117–121, January 2019.

[20] Ruben E. Perez, Jose Arnal, and Peter W. Jansen. Neuro-Evolutionary Control for Optimal Dynamic Soaring. In *AIAA Scitech 2020 Forum*. American Institute of Aeronautics and Astronautics, January 2020.

[21] Nicholas Lawrance and Salah Sukkarieh. Wind Energy Based Path Planning for a Small Gliding Unmanned Aerial Vehicle. In *AIAA Guidance, Navigation, and Control Conference*, Guidance, Navigation, and Control and Co-located Conferences. American Institute of Aeronautics and Astronautics, Chicago, IL, USA, August 2009.

[22] Roland B. Stull. *An Introduction to Boundary Layer Meteorology*. Atmospheric Sciences Library. Springer, Dordrecht, first edition, 1988.

[23] Mark Cutler, Tim McLain, Randal Beard, and Brian Capozzi. Energy Harvesting and Mission Effectiveness for Small Unmanned Aircraft. In *AIAA Guidance, Navigation, and Control Conference*, Guidance, Navigation, and Control and Co-located Conferences. American Institute of Aeronautics and Astronautics, August 2010.

[24] Anil Rao. A Survey of Numerical Methods for Optimal Control. *Advances in the Astronautical Sciences*, 135, January 2010.

[25] John T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Second Edition*. Advances in Design and Control. Society for Industrial and Applied Mathematics, January 2010.

[26] Timothy D. Woodbury, Caroline Dunn, and John Valasek. Autonomous Soaring Using Reinforcement Learning for Trajectory Generation. In *52nd Aerospace Sciences Meeting*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, National Harbor, MD, USA, January 2014.

[27] Jen Chung, Nicholas Lawrance, and Salah Sukkarieh. Learning to soar: Resource-constrained exploration in reinforcement learning. *The International Journal of Robotics Research*, 34:158–172, January 2014.

[28] Kenneth O. Stanley and Risto Miikkulainen. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, 10(2):99–127, June 2002.

[29] Răzvan Florian. Correct equations for the dynamics of the cart-pole system. August 2005.

[30] Alan McIntyre, Matt Kallada, Cesar G. Miguel, and Carolina Feher da Silva. Neat-python. CodeReclaimers, 2015.

[31] Art B. Owen. *Monte Carlo Theory, Methods and Examples.* 2013.

[32] Eric J. Kim and Ruben E. Perez. Neuroevolutionary Control for Autonomous Soaring. *Aerospace*, 8(9):267, September 2021.

[33] ArduPilot. ArduPilot Autopilot Software Suite, 2010.

[34] Laminar Research. X-Plane 10, 2013.

[35] Michael Osborne. ArduPilot Mission Planner, 2015.

[36] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* Adaptive Computation and Machine Learning Series. A Bradford Book, Cambridge, MA, USA, second edition, November 2018.

Appendices

A Reinforcement Learning Case Details

This section provides additional details on the actor-critic and NEAT algorithms implemented for the cart-pole case presented in Chapter 4.

A.1 Simulation Environment

The simulation model of the cart-pole system used the parameters listed in Table A.1. The state space was continuous for both algorithms.

Table A.1: Cart-pole system parameters.

Parameter	Value
Maximum simulation time	30.0s
Discrete time step size	0.02s
Control actuation frequency	25 Hz
Cart mass	0.711 kg
Pole mass	0.209 kg
Pole length	0.652 m
Maximum cart position	±2.4 m
Maximum pole angle	±12 deg
Control force	10 N

A.2 Actor-critic Reinforcement Learning Implementation

The actor-critic algorithm use separate models to select actions through the policy structure known as the actor, and estimate the value function through the critic [36]. Both the policy and values are typically encoded in artificial neural networks. After obtaining an action a_t from the actor network, the action is taken to calculate the reward r_{t+1} and the agent's next state s_{t+1}. The critic network subsequently calculates the values of the previous $V(s_t)$ and new states $V(s_{t+1})$, which are used to compute the advantage $A(s_t, a_t)$ and update the networks through backpropagation. The advantage, a measure of how much better it is to take a specific action from a certain state than the average value of that state, is estimated using the formula for the temporal difference (TD) error δ_t, where γ is the discount factor that dictates the degree of importance attributed to future rewards:

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{A.1}$$
$$\approx A(s_t, a_t) \tag{A.2}$$

The advantage and the value of the new state are used to update the neural networks. The algorithm is summarized in Algorithm 3.

Algorithm 3 Actor-critic algorithm.

```
Create and initialize actor and critic networks with random weights
for each episode: do
    Fetch initial state s
    for each step in episode: do
        Sample action A from actor network
        Obtain next state s', reward r
        Compute neural network update targets (advantage, value of new state)
        Update network weights using the targets and the inputs
        s ← s'
```

To compare the topologies and performances of the RL and NEAT policies, it was first necessary to determine the actor network's topology. Initially, a hyperparameter search was conducted to find a set of parameters that consistently resulted in successful policies, which is listed in Table A.2.

Table A.2: Actor-critic reinforcement learning hyperparameters.

Hyperparameter	Value
Actor learning rate	1×10^{-3}
Critic learning rate	5×10^{-3}
Discount factor	0.99
Number of episodes	1000

A second parameter search was conducted to find the best-performing topology, and the results are shown in Table A.3. The network with 1 hidden layer with 32 units was shown to keep the system within state bounds for the greatest duration, and therefore, it was selected for comparison with the NEAT-evolved network.

Table A.3: Actor-critic reinforcement learning topology search results.

Number of hidden layers	Number of hidden units per layer	Maximum balancing time, s
0	0	1.60
1	4	1.34
1	16	3.20
1	32	5.50
1	64	4.38
2	4	2.04
2	16	1.74
2	32	1.64
2	64	2.90