*aerospace*

MDPI

*Article*

# Neuroevolutionary Control for Autonomous Soaring

**Eric J. Kim and Ruben E. Perez ***

Department of Mechanical and Aerospace Engineering, Royal Military College of Canada, P.O. Box 17000, Kingston, ON K7K 7B4, Canada; Eric.Kim@rmc.ca
* Correspondence: Ruben.Perez@rmc.ca

**Abstract:** The energy efficiency and flight endurance of small unmanned aerial vehicles (SUAVs) can be improved through the implementation of autonomous soaring strategies. Biologically inspired flight techniques such as dynamic and thermal soaring offer significant energy savings through the exploitation of naturally occurring wind phenomena for thrustless flight. Recent interest in the application of artificial intelligence algorithms for autonomous soaring has been motivated by the pursuit of instilling generalized behavior in control systems, centered around the use of neural networks. However, the topology of such networks is usually predetermined, restricting the search space of potential solutions, while often resulting in complex neural networks that can pose implementation challenges for the limited hardware onboard small-scale autonomous vehicles. In exploring a novel method of generating neurocontrollers, this paper presents a neural network-based soaring strategy to extend flight times and advance the potential operational capability of SUAVs. In this study, the Neuroevolution of Augmenting Topologies (NEAT) algorithm is used to train efficient and effective neurocontrollers that can control a simulated aircraft along sustained dynamic and thermal soaring trajectories. The proposed approach evolves interpretable neural networks in a way that preserves simplicity while maximizing performance without requiring extensive training datasets. As a result, the combined trajectory planning and aircraft control strategy is suitable for real-time implementation on SUAV platforms.

**Keywords:** dynamic soaring; thermal soaring; neurocontrol; artificial neural network; neuroevolution; unmanned aerial vehicle; flight trajectory; aircraft control; trajectory optimization; optimal control



**Citation:** Kim, E.J.; Perez, R.E. Neuroevolutionary Control for Autonomous Soaring. *Aerospace* **2021**, *8*, 267. https://doi.org/10.3390/aerospace8090267

Academic Editor: Hailong Huang

Received: 29 July 2021
Accepted: 14 September 2021
Published: 17 September 2021



## 1. Introduction

Interest in small unmanned aerial vehicles (SUAVs) has continually been increasing due to their utility in numerous applications ranging from scientific data acquisition to military surveillance. Nevertheless, the power storage limits onboard small-scale aircraft greatly restrict their maximum flight times, negatively impacting their ability to perform long-range operations. As a result, there exists practical value in improving the energy management of SUAVs. Recent work in addressing this issue from an aeronautical perspective has been motivated by a significant interest in solar-powered vehicles. However, biologically inspired mechanisms can offer other solutions to the problem. In nature, certain species of birds such as the albatross and the frigatebird have evolved methods of extracting energy from wind by soaring in particular patterns that exploit the differences in wind velocities at varying altitudes and regions. One such technique known as dynamic soaring can be observed in albatross birds, who perform cyclic maneuvers using horizontal winds to travel across large, transoceanic distances without flapping their wings [1]. Similarly, thermal soaring enables frigatebirds to continuously loiter over oceans using columns of rising air, or thermals [2]. Since these discoveries, the study of autonomous soaring for SUAVs has been a growing field pursuing a multidimensional challenge that involves wind field mapping, trajectory planning, and aircraft control. A particular difficulty lies in creating an autonomous system that can compute energy-efficient dynamic soaring trajectories and pass control commands to a computationally limited autopilot platform.

To understand the energy efficiency of dynamic soaring, a typical albatross bird weighing 8.5 kg would need to generate 81.0 W of power to maintain an airspeed of 38 knots for a long-endurance flight. An equivalent gasoline engine would consume almost a liter of fuel per day, and for the over 15,000 km-long transoceanic travel commonly observed of wandering albatrosses, such an energy expenditure would be calorically unsustainable and physiologically destructive without the evolved technique of dynamic soaring [3]. Natural observation of the method's capacity to significantly contribute to flight endurance with minimal cost illustrates its potential value in aerial vehicles. Similarly, pioneering research in thermal soaring found that in one instance, the application of the technique in SUAVs could result in a remarkable twelve-hour increase in flight endurance on an electrical vehicle with a base endurance of two hours [4]. Yet, despite the clear benefits, the implementation of these techniques remains a challenge due to the problems of planning a sustainable soaring trajectory, as well as controlling an aircraft along the trajectory.

In addressing the issue of trajectory planning, much of the existing research has involved numerical trajectory optimization (TO) techniques to calculate the optimal energy-neutral flight path. The extensive computational resources required to perform the large number of calculations have primarily seen the use of trajectory optimization for exploring the theoretical feasibility of soaring maneuvers [5,6]. Even so, through continued research efforts, real-time optimization techniques have now largely bypassed the problem of timely trajectory generation [7–9]. As for the issue of aircraft control, a significant body of research has explored the popular approach of explicitly tracking a calculated trajectory [10,11] in a category of methods known as tracking control. However, the individual addressing of the planning and control problems reveals another challenge with this distinct categorization of the components of autonomous soaring: such approaches depend heavily on the reference trajectory, causing any deviations from the planned path to compromise the entire soaring cycle as additional energy must be expended to correct tracking errors. When considering the stringent energy management required for sustained soaring, this susceptibility of the approach becomes a major obstacle towards successful implementation.

Due to these considerations, in addition to modern improvements in the accessibility of neural network training algorithms, the research in dynamic soaring implementation has seen a recent interest in the field of neurocontrol. A subfield of intelligent control, neurocontrol is characterized by the use of neural networks in control systems. Specifically, the use of reinforcement learning (RL) algorithms and deep neural networks has already been initially explored in the context of dynamic soaring [12–15] to train neural networks capable of exhibiting generalized and adaptable soaring behavior. That being said, the majority of the existing literature around the use of neural networks in aerial control systems has been focused on fixed-topology networks, where the structure of the nodes and connections are kept constant [16–18]. In such approaches, the size of the neural network is a design choice that must be hand-tuned to obtain the desired performance. Unfortunately, this often leads to large, complex, deep neural networks that pose difficulties in training, interpretation, and implementation, which are exacerbated by the extensive training datasets required for some learning methods [12]. Alternatively, the neuroevolution of augmenting topologies strategy developed by Stanley et al. [19] progressively generates not only the weights and biases, but also the unique topology of the structure to produce simple and effective neural networks.

This paper, building upon a previous work by Perez et al. [20] introducing the applicability of the NEAT algorithm to optimal dynamic soaring, presents a combined trajectory planning and control strategy that can generate sustainable, energy-efficient soaring trajectories with simple and computationally inexpensive neural networks. The strategy's ability to evolve efficient and effective controllers makes it a promising candidate for creating a practically implementable system on existing SUAV autopilot architectures. The current work details the neuroevolutionary approach's unique penalty and fitness functions, uses a flight dynamics model that takes into account wind from all three Cartesian directions, demonstrates the method's applicability to various soaring techniques from higher-altitude

dynamic soaring to thermal soaring, illustrates the similarities in energy-efficiency between the resulting neurocontrol trajectories and those of biological albatross birds, and provides a comparison of the simulated results against numerical trajectory optimization, as well as other neural network training methods. The paper is organized as follows: Section 2 defines the dynamic and thermal soaring problems. Section 3 describes the NEAT-based neurocontroller training strategy. Section 4 presents the results of various neurocontroller test cases, and Section 5 provides concluding remarks.

## 2. Soaring Problem

The dynamic and thermal soaring problems can be described using the framework of trajectory optimization, since soaring maneuvers aim to optimize a particular parameter such as energy, flight time, or travel distance. This section presents the general structure of trajectory optimization before detailing the aircraft and wind models used for this work.

### *2.1. Trajectory Optimization*

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 state $\mathbf{x}(t)$, the control $\mathbf{u}(t)$, the initial time $t_0$, and the final time $t_f$ that minimizes the cost function $J$, with boundary conditions $\Phi$ and Lagrange term performance index $\mathcal{L}$, all summarized as [21]:

$$J = \Phi\Big[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f\Big] + \int_{t_0}^{t_f} \mathcal{L}[\mathbf{x}(t), \mathbf{u}(t), t]dt \tag{1}$$

Although the cost function helps define soaring behavior, it is also necessary to impose the rules of the problem. Therefore, there exists 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 $\phi$, which limit the initial and final system states:

$$\dot{\mathbf{x}}(t) = \mathrm{f}[\mathbf{x}(t), \mathbf{u}(t), t] \tag{2}$$

$$C[\mathbf{x}(t), \mathbf{u}(t)] \leq 0 \tag{3}$$

$$\phi_{min} \leq \phi\Big[\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f\Big] \leq \phi_{max} \tag{4}$$

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 $\mu$:

$$h(t) \geq 0 \tag{5}$$

$$n(t) = \frac{L}{mg} \leq n_{max} \tag{6}$$

$$C_{L_{min}} \leq C_L(t) \leq C_{L_{max}} \tag{7}$$

$$-\mu_{max} \leq \mu(t) \leq \mu_{max} \tag{8}$$

Boundary conditions, or equality constraints, are also specified to define the initial and final states of the trajectory and are necessary to distinguish between different soaring techniques. In all, these elements define the trajectory optimization problem, which can be solved through various techniques. For instance, a nonlinear-programming (NLP)-based direct collocation approach discretizes and transcribes the continuous-time soaring

task into a nonlinear programming problem by dividing the time interval into $N$ discrete subintervals and $M = N + 1$ nodes or collocation points [22]:

$$t_I = t_1 < t_2 < \ldots < t_M = t_F \tag{9}$$

In the trapezoidal collocation method, multiple polynomial splines are used to represent the optimal state and control trajectories, while ensuring that the defects $\zeta$, or the discrepancies between collocation intervals, are zero to ensure continuity in the system dynamics. The dynamics are discretely approximated using the trapezoid rule, where $y_k$ is an NLP decision variable and $f_k = f(\mathbf{x}_k, \mathbf{u}_k, t_k)$ is the evaluation of the dynamics at each node. The entire process, summarized as follows, is also subject to path and boundary constraints:

$$\text{minimize} \quad J = \int_{t_0}^{t_k} \mathcal{L}[\mathbf{x}(t_k), \mathbf{u}(t_k), t_k) dt] \tag{10}$$

$$\text{with respect to} \quad y_k, k = 1, 2, \ldots M \tag{11}$$

$$\text{subject to} \quad \zeta_k = y_{k+1} - y_k = \frac{h_k}{2}[f_{k+1} + f_k] = 0 \tag{12}$$

$$C[\mathbf{x}(t_k), \mathbf{u}(t_k), t_k] \leq 0 \tag{13}$$

$$\Phi[\mathbf{x}(t_0), t_0, \mathbf{x}(t_M), t_M] = 0 \tag{14}$$

The solution to a trajectory optimization problem is represented by a set of state and control histories. In the case of soaring, the position and attitude of an aircraft would be specified for every time interval, along with the values of each control variable, which can be tracked by a control scheme to obtain the desired behavior. Although this work does not explicitly use trajectory optimization, the control scheme to be presented incorporates many of the concepts and elements outlined above. Nonetheless, in addition to the problem formulation, trajectory optimization requires models of the flight agent and the environment.

### *2.2. Flight Dynamics Model*

The system dynamics used in this work was based on a three-degree-of-freedom (3DOF) point mass model of an SUAV. The motion of the aircraft is relative to an inertial frame located on the Earth's surface. Figure 1 depicts a free body diagram of the model, and the equations of motion [5,23], adapted for wind in the $x$, $y$, and $z$ directions, are defined as follows, where $V$ is airspeed, $g$ is standard gravity, $L$ is lift, $D$ is drag, $m$ is the aircraft's mass, $\psi$ is the heading angle, $\gamma$ is the pitch angle, $\mu$ is the roll angle, $x$, $y$, and $h$ are the aircraft's positions in the $x$, $y$, and $z$ directions relative to the inertial frame, and $W_x$, $W_y$, and $W_z$ are the environmental wind strengths along their respective axes:

$$\dot{V} = -\frac{D}{m} - g\sin(\gamma) - \dot{W}_x\cos(\gamma)\sin(\psi) - \dot{W}_y\cos(\gamma)\cos(\psi) + \dot{W}_z\sin(\gamma) \tag{15}$$

$$\dot{\psi} = \frac{L\sin(\mu)}{Vm\cos(\gamma)} - \frac{\dot{W}_x\cos(\psi)}{V\cos(\gamma)} + \frac{\dot{W}_y\sin(\psi)}{V\cos(\gamma)} \tag{16}$$

$$\dot{\gamma} = \frac{1}{V}\left(\frac{L\cos(\mu)}{m} - g\cos(\gamma) + \dot{W}_x\sin(\gamma)\sin(\psi) + \dot{W}_y\sin(\gamma)\cos(\psi) + \dot{W}_z\cos(\gamma)\right) \tag{17}$$

To track the aircraft's states, the kinematic equations of motion are defined, relating the motion of the aircraft with respect to the inertial frame:

$$\dot{x} = V\cos(\gamma)\sin(\psi) + W_x \tag{18}$$

$$\dot{y} = V\cos(\gamma)\cos(\psi) + W_y \tag{19}$$

$$\dot{h} = V\sin(\gamma) + W_z \tag{20}$$

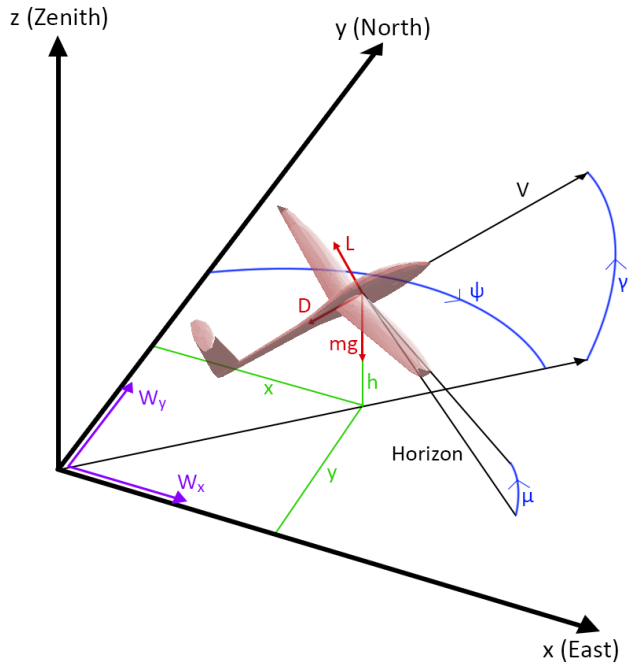



**Figure 1.** Free body diagram of the flight agent model.

Lift and drag are computed as shown below, with local air density $\rho$, wing reference area $S$, lift coefficient $C_L$, and drag coefficient $C_D$:

$$L = \frac{1}{2}\rho V^2 S C_L \tag{21}$$

$$D = \frac{1}{2}\rho V^2 S C_D \tag{22}$$

The drag coefficient is a function of both parasitic and lift-induced drag, where $C_{D0}$ is the zero-lift drag coefficient, $K$ is the induced drag factor, and $E_{max}$ is the maximum lift-to-drag ratio:

$$C_D = C_{D0} + K C_L^2 \tag{23}$$

$$K = \frac{1}{4 C_{D0} E_{max}^2} \tag{24}$$

Furthermore, the total energy of the aircraft $e_T$ comprises both the kinetic $e_K$ and potential $e_P$ energies as described:

$$e_T = e_K + e_P \tag{25}$$

$$e_K = \frac{1}{2} m V^2 \tag{26}$$

$$e_P = mgh \tag{27}$$

The elements of the state vector shown below were purposefully selected to only include variables that would typically be measurable onboard an SUAV. In addition, the control variables were chosen to be navigation-level commands that can be conveyed to an autopilot system:

$$\mathbf{x} = \begin{bmatrix} V \\ \psi \\ \gamma \\ h \\ \dot{h} \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} C_L \\ \mu \end{bmatrix} \tag{28}$$

This mathematical model of the aircraft is required for trajectory optimization and, more generally, can be used to simulate a physical system. However, soaring techniques rely on naturally occurring winds, which must also be expressed through models.

### *2.3. Dynamic Soaring*

A traveling dynamic soaring cycle, illustrated in Figure 2, can be characterized by five primary segments: upwind climb (1), high altitude turn (2), downward sink (3), low altitude turn (4), and an intermediary traveling component for expending any additional energy gained over the cycle [1]. An agent undergoing dynamic soaring trades kinetic energy for potential energy as it gains altitude by turning upwards into the wind field and subsequently trades the potential energy for kinetic energy after it turns downwards away from the wind. This cycle repeats as the agent travels in a specific direction over soaring cycles.

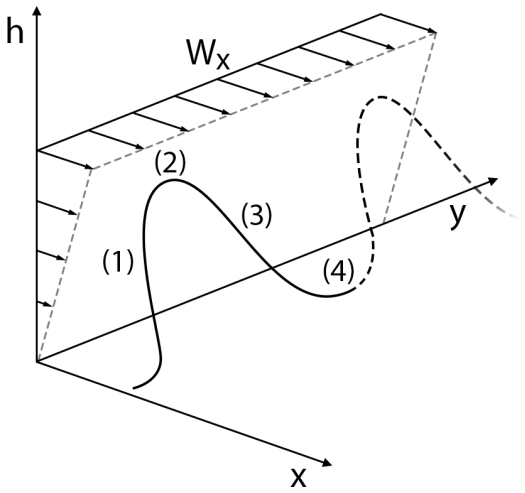



**Figure 2.** Typical dynamic soaring trajectory with a linear wind gradient (adapted from [5]).

From the initial formulation by Zhao [5], the horizontal wind profile from which energy is extracted can be modeled through a horizontal wind speed $W_x$, represented by a shape parameter $A_x$, an average gradient slope $\beta_x$, a maximum wind speed $W_{max_y}$, and a transitional altitude $h_{tr_x}$:

$$W_x = \begin{cases} \beta\left[Ah + \frac{1-A_x}{h_{tr_x}}h^2\right] & h < h_{tr_x} \\ W_{max_x} & h \geq h_{tr_x} \end{cases} \tag{29}$$

$$\beta = \frac{W_{max_x}}{h_{tr_x}} \tag{30}$$

The rate of change of the wind profile is defined as:

$$\dot{W}_x = \beta_x\left[A_x + 2\frac{1-A_x}{h_{tr_x}}h\right]V\sin(\gamma) \tag{31}$$

Figure 3 shows that this model represents a logarithmic wind profile when $0 < A_x < 1$ and an exponential profile when $1 < A_x < 2$. Furthermore, for $W_x$ to not exceed $W_{max_x}$, $A_x$ must be limited to $[0, 2]$, with a shape value $A_x = 1$ corresponding to a linear profile. An identical model can be used to simulate wind $W_y$ in the $y$ direction.

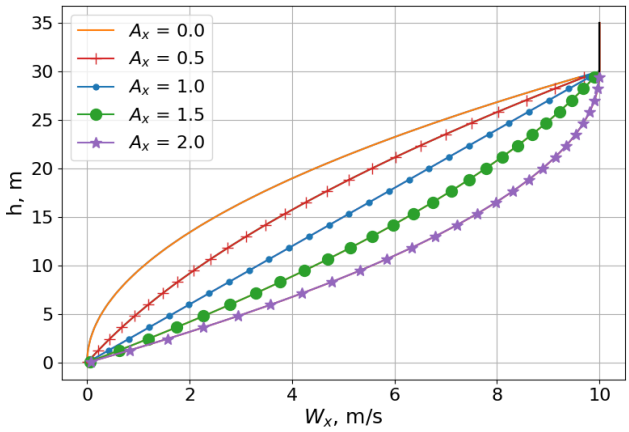


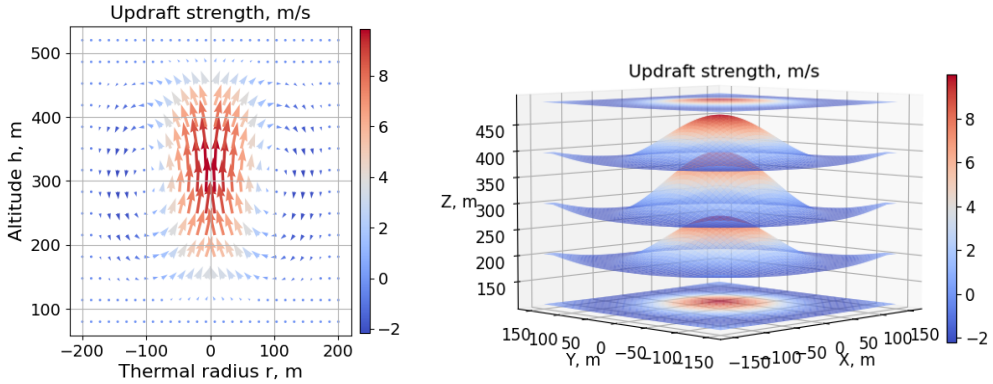

**Figure 3.** Horizontal wind profile shapes, with $W_{max_x} = 10$ m/s and $h_{tr_x} = 30$ m.

In the trajectory optimization framework, the boundary constraints for dynamic soaring would include initial and final conditions on the airspeed, heading angle, pitch angle, and coordinate states. For the traveling case, the latter would not be subjected to final conditions, as the distance covered would be a variable result of the optimized trajectory output.

$$\begin{aligned} V(0) &= V_0 & V(t_f) &= V_f \\ \psi(0) &= \psi_0 & \psi(t_f) &= \psi_f \\ \gamma(0) &= \gamma_0 & \gamma(t_f) &= \gamma_f \\ x(0) &= x_0 \\ y(0) &= y_0 \\ h(0) &= h_0 \end{aligned} \tag{32}$$

### *2.4. Thermal Soaring*

In thermal soaring, a flight agent circles around a region containing an updraft 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 [24], 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 4.




(**a**) Vector field (**b**) Vertical wind speed cross-sections

**Figure 4.** Toroidal thermal bubble model.

The model, minimally altered from the original formulation to fit the coordinate system presented in Section 2.2, is defined by the agent's distance from the thermal center $r$, 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 velocity $\dot{h_t}$, where $x$, $y$, and $h$ are the agent's location in the inertial frame:

$$r = \sqrt{x^2 + y^2}, \qquad k = \frac{r_z}{r_{xy}} \tag{33}$$

$$W_z = \begin{cases} w_{core} & r = 0, \quad h \in [h_t - r_z, h_t + r_z] \\ \cos(\frac{2\pi}{4r_z}(h - h_t))\frac{r_{xy} w_{core}}{\pi r} \sin(\frac{\pi r}{r_{xy}}) & r \in (0, 2r_{xy}], \quad h \in [h_t - r_z, h_t + r_z] \end{cases} \tag{34}$$

$$W_x = -Wz\frac{h - h_t}{(r - r_{xy})k^2}\frac{x}{r} \tag{35}$$

$$W_y = -Wz\frac{h - h_t}{(r - r_{xy})k^2}\frac{y}{r} \tag{36}$$

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 the thermal remains sufficiently strong. In all, these models of the flight agent and wind conditions were used to simulate an environment in which controllers were trained and soaring tests conducted.

## 3. Neurocontrol

This section describes the neural-network-based evolutionary control scheme by explaining the motivation for its use, providing a description of the main algorithm and detailing the specific parameters and functions used for this work.

### *3.1. Neural Network Topology*

Although classic trajectory optimization techniques provide numerically optimal solutions, recent advancements in neural network research have encouraged the exploration of artificial-intelligence-based approaches in generating soaring trajectories in a more general, behavior-like manner that can be executed on the limited computing hardware onboard SUAVs. For instance, the field of dynamic soaring has already seen the application [12–15] of one of the most common disciplines of artificial intelligence, known as reinforcement learning. In such approaches, neurocontrollers are trained to behave in a particular manner or to follow a policy $\pi$ by constructing a mapping of states $S$ to actions $A$:

$$\pi : S \rightarrow A \tag{37}$$

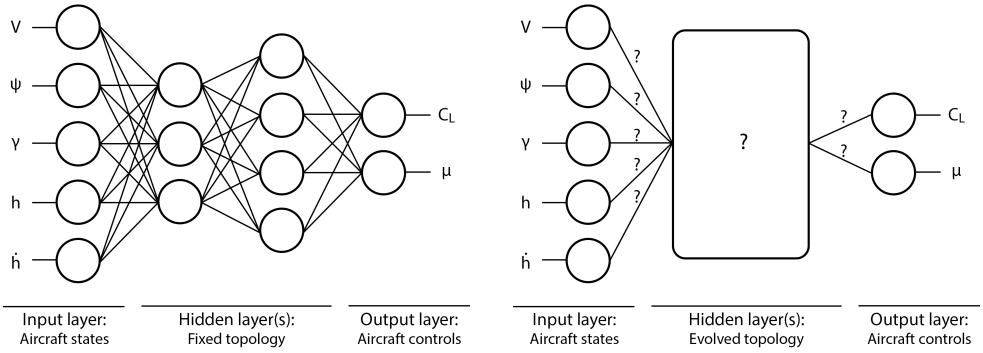
For complex, nonlinear systems, this process often involves the use of neural networks as function approximators that estimate the value $V$ of taking a specific action from a particular state. As a learning agent interacts with the environment either directly or through a simulation, it receives rewards $R$ that encourage or discourage certain actions, based on a defined evaluation criterion $J$. For autonomous soaring, the costly nature of learning through trial and error necessitates a simulation-based training approach. Furthermore, control actions that result in an energy-neutral trajectory can only be evaluated after a full soaring cycle, and as such, the optimal policy $\pi^*$ can be defined as the mapping between states $\mathbf{x}(t)$ and actions $\mathbf{u}(t)$ that maximizes this energy-based performance criteria, or achieves the greatest reward:

$$\pi^* : \mathbf{x}(t) \rightarrow \mathbf{u}(t) \tag{38}$$

Often, this policy is either indirectly encoded through stochastic exploratory and exploitative mechanisms or more consistently and directly defined in the form of a neural network as is the case for more complex RL algorithms. In this way, different learning algorithms can be applied to the soaring problem to obtain neural-network-based control

strategies, where inputs such as aircraft states can be fed-forward to obtain the control action that will maximize the reward.

However, to avoid the structural limitations of fixed-topology networks and the consequent omission of a subset of potential solutions, NEAT can be used to evolve unique networks that are optimized in node and connection weights, as well as in topology. This difference is illustrated in Figure 5.




(**a**) Fixed-topology network (**b**) NEAT network

**Figure 5.** Comparison of artificial neural network structures.

Canonical tests in solving artificial intelligence and control problems have demonstrated the NEAT algorithm's ability to generate networks that often perform better while being much simpler than fixed-topology networks [19].

### *3.2. Neuroevolutionary Strategy*

The NEAT algorithm operates on the principle of Darwinian fitness, which is a measure of how well an individual is able to propagate itself through successive generations. In the context of this study, the fitness of a neurocontroller is the value that defines how well a neural network is able to perform autonomous soaring in a virtual environment, which in turn directly influences its survivability in the evolutionary process.

Initially, the algorithm populates a generation with random members, or neurocontrollers, with randomly generated features. These features, consisting of the unique topology and weights of a member, are encoded within its genotype, a data structure that contains information about each node and connection, the value of each node and connection, and the point in the evolutionary process at which each node and connection was created. In addition, members that share similar network shapes are grouped together into subcategories classified as species, which is a protective structure that allows for the 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, innovative mutations have a chance to mature. Nonetheless, each genotype completely characterizes a unique neurocontroller, and this particular neural network encoding method allows for the tracking and evolution of network topologies.

After every member of a new population is evaluated in a virtual test environment and assigned a fitness value, species undergo reproduction by eliminating the members with the lowest fitnesses before creating offspring and undergoing mutation. 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 process allows for two neurocontrollers to merge and create a better-performing network. 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 gene history encoded in each genotype. The crossover and mutation operations allow for the creation of new, unique neural networks while persevering the genes or characteristics that contribute to high fitnesses. Lastly, competition between species is addressed through stagnation, which is when the highest fitness achieved by a member of a species has not improved after a certain number of generations. Such species are marked as stagnant and are prohibited from reproducing further, eventually becoming extinct through the generational culling process. However, 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 once, and outputs the best-performing neural network after a fitness threshold is reached or a certain number of generations has elapsed. The evolutionary algorithm is depicted in the outer loop of Figure 6, the entirety of which presents the main effort of this work. The inner loop tests each member of the generation one at a time in a simulation environment that includes a model of the flight agent and wind profile. At every time interval of the assessment, the neural network being evaluated receives aircraft states to produce control outputs used to propagate the simulation and obtain the flight agent's subsequent states.

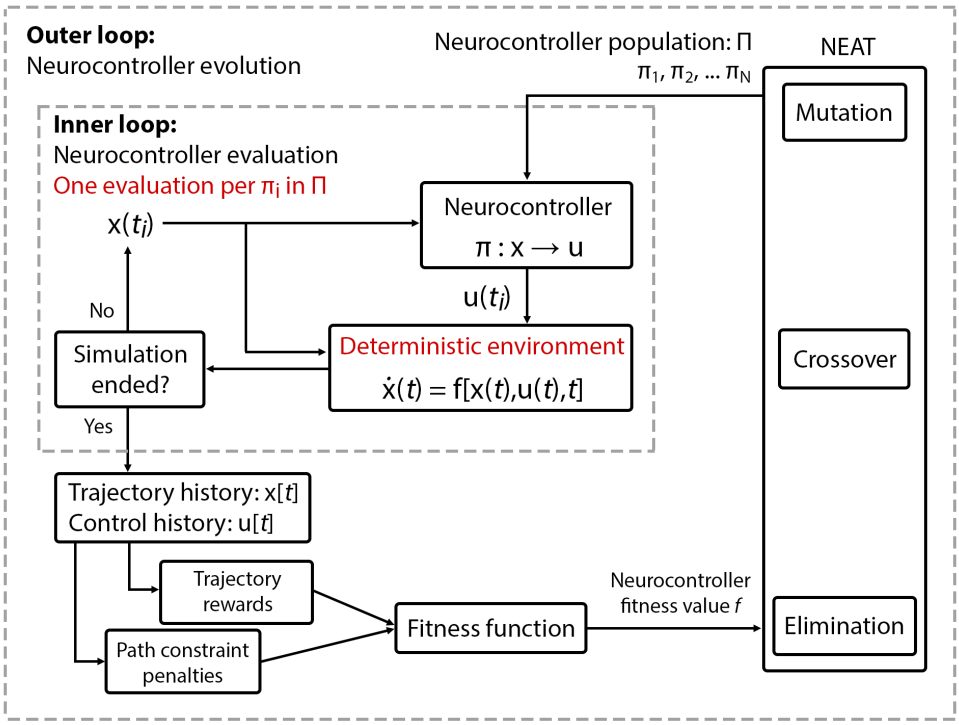



**Figure 6.** Neurocontroller evolution scheme.

At the end of each test, an evaluation process examines the trajectory along which the neural network controlled the flight agent to determine the member's fitness score. For the test cases presented in the following sections, this value was the numerical output of a specifically designed function that included penalties proportional to the degree to which the neurocontroller exceeded flight constraints, as well as rewards dependent on the total distance that the flight agent traveled. The NEAT algorithm then receives and stores the fitness of each member until every member of the population is evaluated, before creating new members and extinguishing unsuccessful networks. In this way, the presented procedure progressively creates better-performing neurocontrollers.

### *3.3. Neuroevolutionary Implementation*

The fitness function that was used to evaluate the performance of each network during the evolutionary cycles is a critical design element that has a significant influence on the network behavior. The functions used for the dynamic and thermal soaring test cases presented in the next section were similar in that they both included penalties that discouraged the flight agent from exceeding trajectory constraints. Since the evaluation process occurs after a member completes a flight episode in the simulation environment, the histories of the states and controls experienced during the simulation were examined for any values that surpassed limits. These included the following variables:

$$\begin{aligned} \text{Airspeed:} &\quad V_{min} \leq V \leq V_{max} \\ \text{Height:} &\quad h_{min} \leq h \leq h_{max} \\ \text{Pitch angle:} &\quad \gamma_{min} \leq \gamma \leq \gamma_{max} \\ \text{Load factor:} &\quad n \leq n_{max} \\ \\ \text{Pitch rate of change:} &\quad \dot{\gamma}_{min} \leq \dot{\gamma} \leq \dot{\gamma}_{max} \\ \text{Heading rate of change :} &\quad \dot{\psi}_{min} \leq \dot{\psi} \leq \dot{\psi}_{max} \\ \text{Lift coefficient rate of change:} &\quad \dot{C}_{Lmin} \leq \dot{C}_L \leq \dot{C}_{Lmax} \\ \text{Roll rate of change:} &\quad \dot{\mu}_{min} \leq \dot{\mu} \leq \dot{\mu}_{max} \end{aligned}$$

The penalties for each of these trajectory variables were calculated as the sum of the value by which the limits were exceeded over every time step $t$ of the simulation. This formulation accumulated penalties proportional to how severely a limit was surpassed. The scheme is expressed below, where $x_{history}$ is the array of values collected over the duration of the simulation, which lasted from $t_0 = 0$ to $t_f$:

$$x_{penalty} = \sum_{t=0}^{t_f} \Big[ max(0, x_{history}(t) - x_{max}) + max(0, x_{min} - x_{history}(t)) \Big] \tag{39}$$

The penalties of all variables $x_{penalty} = x_{pen}$ were then squared, summed, and negated to result in a single value that became the fitness $f$ of a member $\pi$:

$$f_\pi = -(V_{pen}^2 + h_{pen}^2 + \gamma_{pen}^2 + n_{pen}^2 + \dot{\gamma}_{pen}^2 + \dot{\psi}_{pen}^2 + \dot{C}_{L\,pen}^{\;2} + \dot{\mu}_{pen}^2) \tag{40}$$

For the traveling dynamic soaring case, however, a reward mechanism was necessary to incentivize the neurocontroller to steer the flight agent along a direction. The reward $r_\pi$ was equal to the square of the agent's displacement along the $xy$ plane:

$$r_\pi = (x_f - x_i)^2 + (y_f - y_i)^2 \tag{41}$$

In the case of a reward, the overall fitness function was a linear combination of the reward and penalty values, where $k_1$ and $k_2$ were experimentally tuned constants that prevented either value from dwarfing the other:

$$f_\pi = (k_1 \times reward) - (k_2 \times penalty) \tag{42}$$

In addition, to strongly dissuade neural networks from exceeding system-critical constraints, an extremely large penalty was attributed to members who steered the model into the ground, induced excessive load factors, or stalled the model.

### *3.4. Simulation Environment*

An explicit time simulation environment was used to train the neurocontrollers presented in the following section. Using the equations of motion for the flight agent along

with the wind models described in Section 2, the simulation time was incremented by a discrete time step $\delta > 0$, and future state values were computed through a first-order approximation. The forward Euler method used to advance the simulation is detailed below, where a state $x_k$ at time step $k$ is updated to $x_{k+1}$:

$$x_{k+1} = x_k + \delta \dot{x}_k \tag{43}$$

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 so that once the simulation terminates either due to the member crashing the flight agent or exhibiting sustained flight for the maximum simulation duration, the member's fitness score can be evaluated from the complete trajectory. Therefore, the dynamics function advancing the simulation was programmed separately from the fitness function called at the end of a simulation, which by extension is where path constraint checks and penalties were also calculated. The simulation is summarized by Algorithm 1.

**Algorithm 1** Flight simulation.

**for** neural network $\pi_N$ in population $\Pi$ **do**
  **while** $t < t_f$ **do**
    $states \leftarrow get\_states()$ ▷ fetch aircraft states
    **if** $states > constraints$ **then**
      break
    $actions \leftarrow \pi_N(states)$ ▷ obtain control commands
    $W_x, W_y, W_z \leftarrow get\_wind()$ ▷ calculate local wind profile
    $x_{k+1} \leftarrow x_k + \delta \dot{x}_k$ ▷ apply dynamics model
    $t \mathrel{+}= \delta$
  $f_\pi \leftarrow get\_fitness()$ ▷ compute rewards and penalties

Each run of the evolutionary neurocontrol optimization presented in this paper consisted of a population size of 250 members, repeated over 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 simulation time of 600 s. Overall, the simulation environment was integrated into a NEAT implementation matching the original NEAT formulation that was ported into Python. As the fitness of each neural network was evaluated, a new instance of the simulation environment was initialized, which stepped through discrete time intervals until the neurocontroller either failed or succeeded at exhibiting feasible soaring patterns.

## 4. Results and Discussion

The following autonomous soaring test cases show the evolutionary approach's capacity to generate simple and effective neurocontrollers. In the first case, a controller was evolved in a bidirectional wind environment using a biological albatross model to compare the trajectory of the neurocontroller to that of an albatross bird. The second case shows a neurocontroller evolved using a typical commercial SUAV model to demonstrate the NEAT-based training approach's applicability to aerial vehicles, and the third test case presents an instance of SUAV thermal soaring. As was shown in Section 2.1, the state vector used as inputs to the evolving neural networks does not contain any explicit information on the local wind field, and as such, the following neurocontrollers evolved by interacting with the wind model without prior knowledge of the environmental conditions. Similarly, the networks also did not receive the simulation time, indicating that the following results did not arise from a generated schedule of control commands.

Although the length of the simulations can vary greatly between population members due to the inherently unique nature of neurocontroller species, the neural networks shown in the next section required an average CPU time of 0.224 min across all three test cases to

evolve a successfully soaring neurocontroller. For reference, the entire evolutionary process was run on an Intel four-core CPU with 16 GB of memory.

### *4.1. Albatross Dynamic Soaring*

The characteristics of the albatross model [1] are shown in Table 1. The parameters of both the flight agent and the wind were selected to compare the results against the data collected and presented by Sachs et al., who recorded the energy extraction cycles of wandering albatrosses through GPS signal tracking [3]. Parameters with the subscript 0 represent initial conditions at time $t_0 = 0$.

**Table 1.** Albatross neurocontroller simulation parameters.

| Albatross Parameters | Value | Wind Parameters | Value | Trajectory Parameters | Value |
|---|---|---|---|---|---|
| $g$ (m/s$^2$) | 9.8 | $A_x$ | 1.0 | $V_0$ (m/s$^2$) | 9.1 |
| $\rho$ (kg/m$^3$) | 1.225 | $h_{tr_x}$ (m) | 9.1 | $\psi_0$ (deg) | $-25$ |
| $m$ (kg) | 8.5 | $W_{max_x}$ (m/s$^2$) | 10.2 | $\gamma_0$ (deg) | 0 |
| $S$ (m$^2$) | 0.65 | | | $h_0$ (m) | 6.1 |
| $C_{D_0}$ | 0.033 | $A_y$ | 1.0 | $x_0$ (m) | 0 |
| $E_{max}$ | 20.0 | $h_{tr_y}$ (m) | 9.1 | $y_0$ (m) | 0 |
| $n_{max}$ | 5 | $W_{max_y}$ (m/s$^2$) | 4.8 | $h_{min}$ (m) | 0 |
| $C_{L_{max}}$ | 1.6 | | | $\dot{\gamma}_{max}$ (deg/s) | 100 |
| $C_{L_{min}}$ | $-0.25$ | | | $\dot{\psi}_{max}$ (deg/s) | 100 |
| $\mu_{max}$ (deg) | 60 | | | $\dot{C}_{Lmax}$ | 0.25 |
| | | | | $\dot{\mu}_{max}$ (deg/s) | 90 |
| | | | | $t_f$ (s) | 600 |

The neurocontroller that was trained using the albatross flight model with a two-dimensional wind profile described in Section 2.3 is shown in Figure 7. Evolved with both the distance-based reward and penalty functions detailed in Section 3.3, the neural network uses a reduced input space of three nodes and no hidden layers, defined only by direct connections to the output nodes.

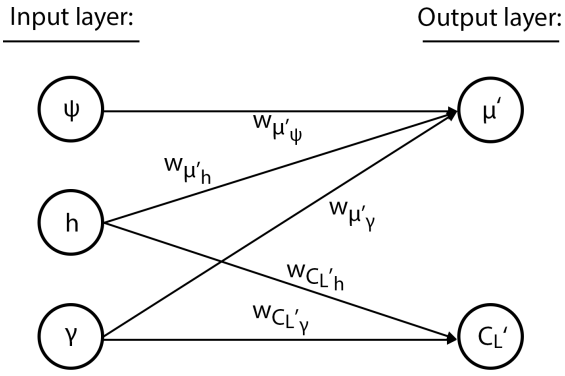



**Figure 7.** Topology of the dynamic soaring albatross neurocontroller.

The bias nodes $b_{C_L'}$ and $b_{\mu'}$, as well as the connection weights $w$ take the values:

$$b_{C_L'} = 2.86 \qquad w_{\mu'_\psi} = -1.96$$
$$b_{\mu'} = -1.37 \qquad w_{\mu'_h} = 0.0759$$
$$w_{\mu'_\gamma} = -2.16$$
$$w_{C_L'_h} = 1.73$$
$$w_{C_L'_\gamma} = 1.62$$

Mathematically, the feedforward network can be described as shown below, where Equation (47) converts the normalized outputs $C_L'$ and $\mu'$ of the sigmoid function $\sigma(x)$ to values within the aircraft control limits:

$$C_L{}' = \sigma(w_{C_L{}'_{\dot{h}}}\dot{h} + w_{C_L{}'_{\gamma}}\gamma + w_{C_L{}'_{V}}V + b_{C_L{}'}) \tag{44}$$

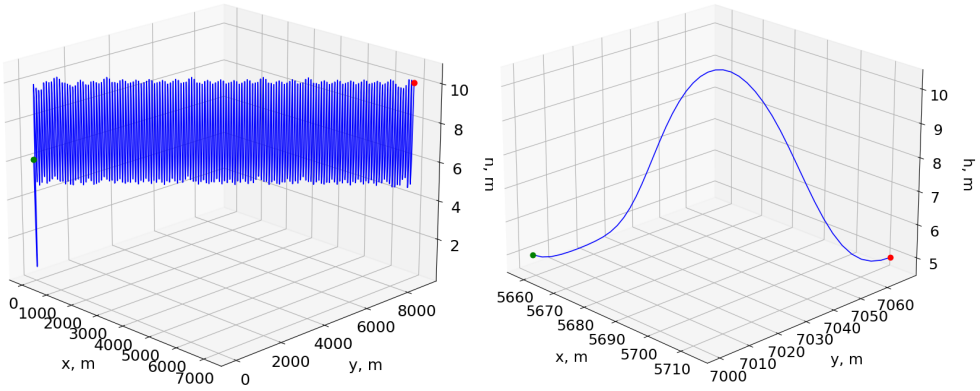
$$\mu' = \sigma(w_{\mu_h}h + b_{\mu'}) \tag{45}$$

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{46}$$

$$\begin{bmatrix} C_L \\ \mu \end{bmatrix} = \begin{bmatrix} C_L{}' & 0 \\ 0 & \mu' \end{bmatrix} \begin{bmatrix} C_{L_{max}} - C_{L_{min}} \\ 2\mu_{max} \end{bmatrix} + \begin{bmatrix} C_{Lmin} \\ -\mu_{max} \end{bmatrix} \tag{47}$$

The simplicity of the neural network revealed that the roll angle was determined by the agent's height, with the lift coefficient determined by the height rate of change, the pitch angle, and the airspeed. This interpretability is important for the implementation and adoption of neurocontrol systems, where understanding the internal mechanisms of networks is a significant component in building trust in such systems. This characteristic of the neuroevolutionary method is further examined in the next test case.

Figure 8a shows the resulting trajectory of simulating this neurocontroller for 600 s in the bidirectional wind environment, and Figure 8b shows a single period of the same trajectory after the aircraft reaches a stable pattern. The flight agent undergoes an initial transitional period before it reaches an equilibrium point conducive to sustained cyclical soaring.




(**a**) 121 cycles (600.0 s) (**b**) 1 cycle (4.9 s)

**Figure 8.** Simulated trajectories of the dynamic soaring albatross neurocontroller, with start marker (green), end marker (red), and path (blue).

The smooth states of a consecutive five-cycle segment, shown in Figure 9, mimic the soaring trajectories of albatross birds [3]. Furthermore, examination of the maximum total energies of the five cycles reveals that the aircraft experiences positive net changes in energy between some periods and negative net changes for others, with this pattern of gaining and losing excess energy continuing throughout the test. Constantly accumulating additional energy between cycles is both naturally limited by the finite wind gradient and undesirable when maintaining stable soaring cycles. Therefore, sustained dynamic soaring is a problem of ensuring that the total energy of the agent does not fall below a certain threshold under which the ability to extract sufficient energy for future cycles is compromised. The test trajectories showed that the neurocontroller can manage the agent's energy for continuous dynamic soaring, balancing any losses in energy with gains in subsequent cycles. This ability to generate continuous and stable soaring cycles after simple feedforward passes through a sparse neural network contrasts the much more limited trajectory horizons of numerical optimization, which typically only plots trajectories of a single period after

extensive computation. By ensuring that the fitness criteria in the genetic algorithm are in part a function of the time for which a particular species survives, the proposed method evolves sustainable soaring as a trait within the controller itself.

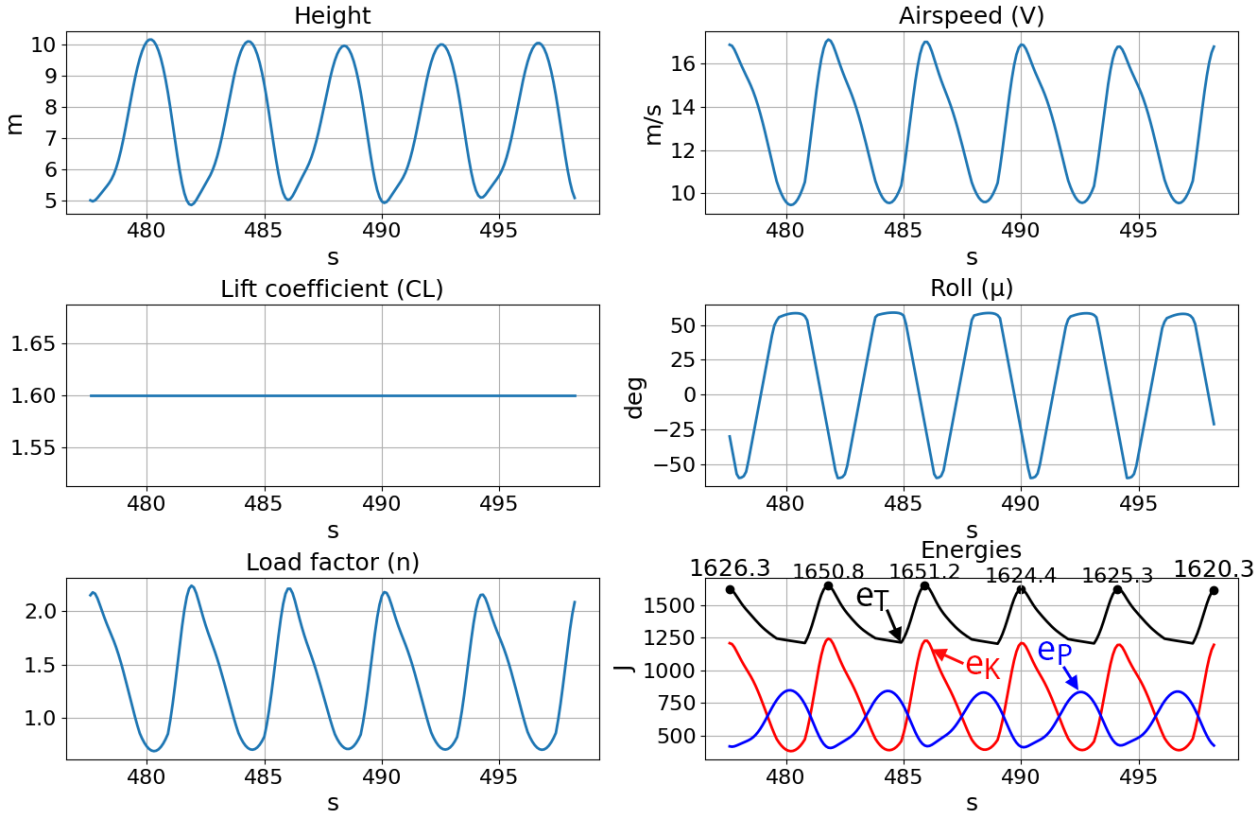
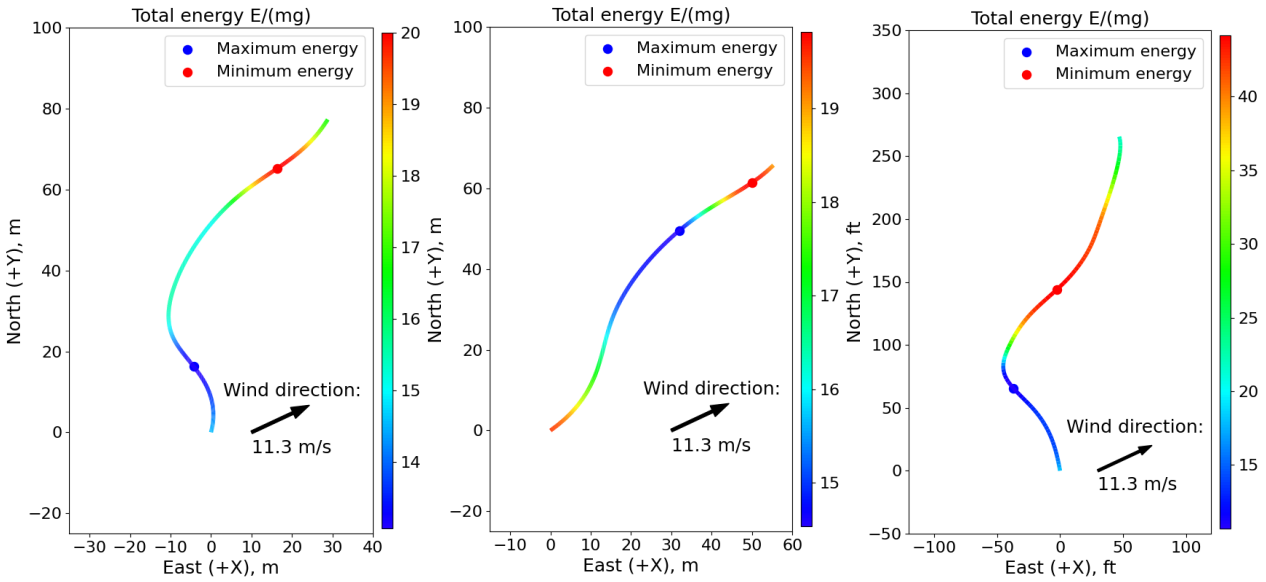



**Figure 9.** Trajectory trace of the dynamic soaring albatross neurocontroller.

The differences in the total energy histories in the initial transitional phase of the soaring test, its stable phase, and a recorded cycle of a biological albatross flight captured by Sachs et al. [3] are illustrated in Figure 10, presented as specific energies. For the transitional cycle, the point of minimum energy occurs during the upwind climb phase as the flight agent uses the wind gradient to accumulate sufficient energy for sustained soaring. After increasing its total energy over multiple transitional cycles, the agent eventually reaches the altitude limit of the gradient and achieves a maximum airspeed that does not exceed control and structural limits. In the stable cycles, the minimum total energy point exists immediately before the downward sink, showing that the aircraft expends most of its energy during the upwind climb and high altitude turn phases before rapidly accelerating into the dive maneuver. This sharp fluctuation between extremes leaves little time to accumulate additional energy, which is necessary to prevent the agent from soaring out of the wind gradient layer or exceeding aircraft limits.

The recorded albatross flight resembles the transitory cycle of the neurocontroller better than the stable period, showing a stronger turn into the wind during the upwind climb phase to accrue excess energy. This aggressive maneuver from the albatross may have been necessary in the presence of a varying and uneven wind profile, unlike the deterministic wind experienced by the neurocontroller. In addition, a flying albatross is likely motivated by much more complex objectives such as minimizing control effort, maximizing flight time, and pursuing prey that extend beyond the relatively simple distance maximization criterion of the neurocontroller. Regardless, the trajectories of the neurocontroller show the smooth, continuous flight that is characteristic of albatross birds undergoing dynamic soaring. Discrepancies between flight paths can be attributed to the simulation's unique wind modeling parameters and the inherently more complex behaviors of biological organisms.

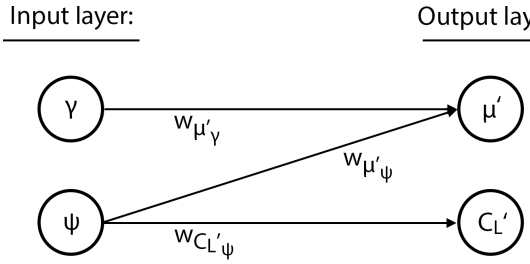



(**a**) Transitory cycle (**b**) Stable cycle (**c**) Biological albatross cycle

**Figure 10.** Albatross neurocontroller and biological albatross single-cycle energies.

### *4.2. SUAV Dynamic Soaring*

Another neurocontroller was evolved using an SUAV model for the flight agent and a different set of wind parameters, all of which are described in Table 2. The resulting neural network of Figure 11 was evolved in a unidirectional wind environment and also only defined by input and output layers much the same as the albatross network, relying solely on the heading and pitch angles.




$b_{C_L'} = 1.48$ $w_{\mu'_\gamma} = -0.929$
$b_{\mu'} = 0.876$ $w_{\mu'_\psi} = -2.39$
$w_{C_{L'\psi}} = -1.39$

**Figure 11.** Topology of the dynamic soaring SUAV neurocontroller.

**Table 2.** Dynamic soaring SUAV neurocontroller simulation parameters.

| SUAV Parameters | Value | Wind Parameters | Value | Trajectory Parameters | Value |
|---|---|---|---|---|---|
| $g$ (m/s$^2$) | 9.8 | $A_x$ | 1.0 | $V_0$ (m/s$^2$) | 10.7 |
| $\rho$ (kg/m$^3$) | 1.225 | $h_{tr_x}$ (m) | 304.8 | $\psi_0$ (deg) | 0 |
| $m$ (kg) | 4.3 | $W_{max_x}$ (m/s$^2$) | 6.1 | $\gamma_0$ (deg) | 0 |
| $S$ (m$^2$) | 1.0 | | | $h_0$ (m) | 301.8 |
| $C_{D_0}$ | 0.025 | $A_y$ | N/A | $x_0$ (m) | 0 |
| $E_{max}$ | 20.0 | $h_{tr_y}$ (m) | N/A | $y_0$ (m) | 0 |
| $n_{max}$ | 15 | $W_{max_x}$ (m/s$^2$) | N/A | $h_{min}$ (m) | 0 |
| $C_{L_{max}}$ | 1.5 | | | $\dot{\gamma}_{max}$ (deg/s) | 100 |
| $C_{L_{min}}$ | $-0.2$ | | | $\dot{\psi}_{max}$ (deg/s) | 100 |
| $\mu_{max}$ (deg) | 60 | | | $\dot{C}_{Lmax}$ | 0.25 |
| | | | | $\dot{\mu}_{max}$ (deg/s) | 90 |
| | | | | $t_f$ (s) | 600 |

However, the topology of this neurocontroller is even more sparse than that of the albatross network. This reduction in network complexity can be attributed to the unidirectional wind profile of the environment in which the SUAV controller was evolved. There is simply one less dimension that the network must account for, and since the network must infer the wind model through its effects on the aircraft without receiving measurements of the local wind, this reduction of the wind profile has a nontrivial impact on the resulting topology. Nevertheless, the simplicity of the network further enables its interpretation. The network is rolling based on the aircraft's pitch and heading angles while determining the angle of attack based on where the aircraft is headed. The NEAT 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 was used to determine when to execute pitch and roll maneuvers that led to dynamic soaring trajectories. To further illustrate the close relation between the states and controls, Figure 12 shows the input signals plotted against the output control commands. Even without the effect of the sigmoid function and the subsequent scaling of the normalized controls, the outputs can be seen to track their respective inputs, albeit with offsets that are simply the result of the specific connection weights and biases. 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 a precise, 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.

Figure 13 shows the trajectories of different scales as a result of simulating this neurocontroller in a unidirectional wind environment. The plots show that the SUAV model achieved a stable pattern near the transition height of the wind profile, which was uniquely shaped in this test case. The altitudes at which the aircraft would soar were too high to use the wind profile detailed in Section 2.3, since the vertical gradient would be insufficiently weak when stretched over hundreds of meters. Therefore, the model used for this test case compressed the profile into a much smaller altitude range, as shown in Figure 14, allowing the lightweight SUAV to extract enough energy from the environment. To perform dynamic soaring with greater altitude fluctuations and longer soaring cycles, a stronger wind profile would be required, with a proportionately sufficient vertical wind gradient, as well as a more robust aircraft that can withstand the load factors associated with more strenuous flight maneuvers. Regardless, the trajectories exemplify the ability to evolve neurocontrollers for dynamic soaring trajectories at altitudes much higher than those experienced by soaring seabirds, with narrower wind gradients.

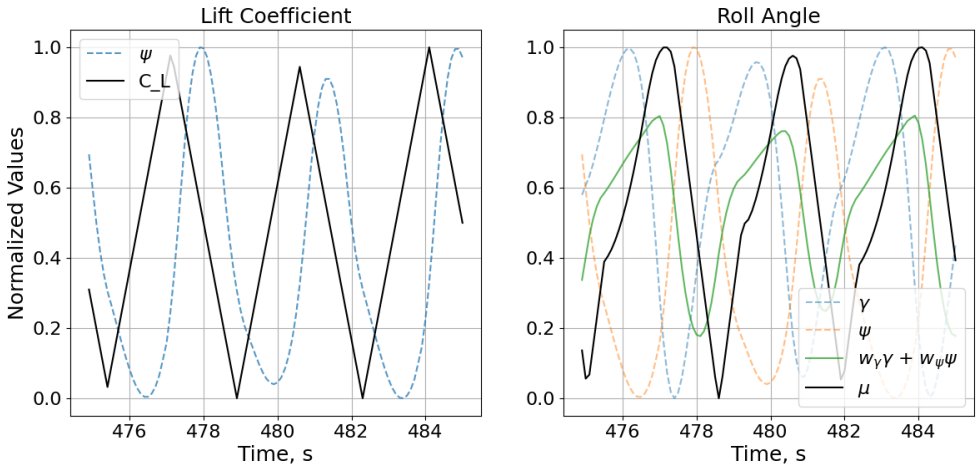
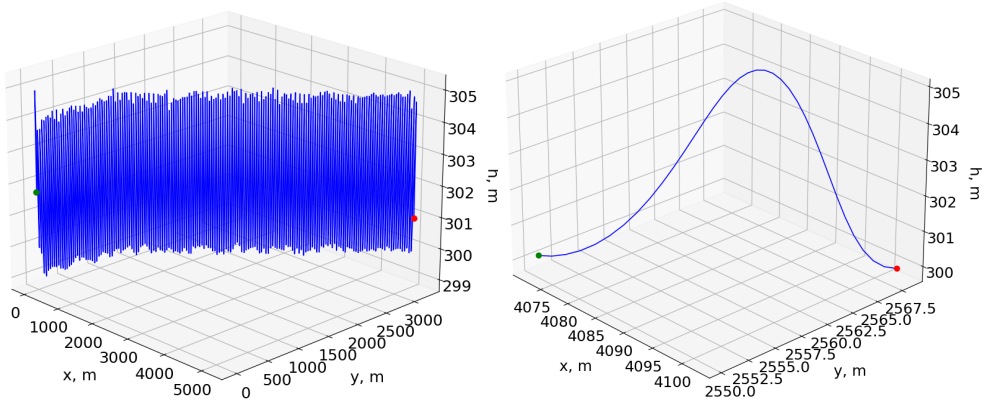



**Figure 12.** Network signal tracing of the dynamic soaring SUAV neurocontroller.




(**a**) 171 cycles (600.0 s) (**b**) 1 cycle (3.3 s)

**Figure 13.** Simulated trajectories of the dynamic soaring SUAV neurocontroller.

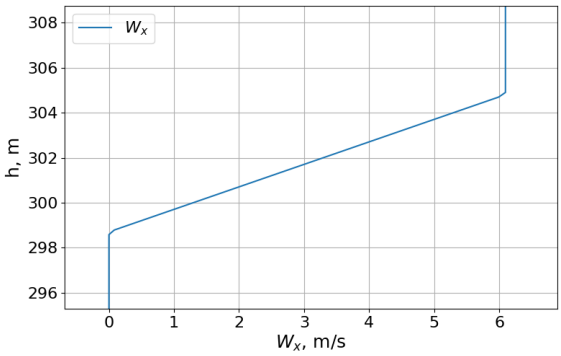



**Figure 14.** Wind profile for the dynamic soaring SUAV neurocontroller.

The smooth states of the multiperiod trajectory, shown in Figure 15, are desired when considering the response rates and limitations of real control systems and physical vehicles. The relatively abrupt changes in the lift coefficient and roll angle controls, however, are required for the rapid maneuvers of dynamic soaring, a characteristic that is also reflected by the cyclic history of the load factor. Additionally, the energy histories do not contain any significant reduction in the total energy between cycles, demonstrating that the neurocontroller can continuously soar given a constant wind profile. In reality, the finite nature of environmental winds would pose a greater obstacle to indefinite soaring than any periodic reduction in energy.

Figure 16 provides a dynamic visualization of the aircraft as it soars through a cycle of the trajectory. The observation of dynamic soaring behavior from such a simple neural network highlights the advantages of the neuroevolutionary method, which are that such controllers are more easily implementable on hardware-limited SUAV platforms than more complex neural networks while being interpretable.

To compare different trajectory planning and control approaches, a numerical trajectory optimization was performed for a single soaring cycle through a direct trapezoidal collocation nonlinear programming approach as described in Section 2.1, where the optimization was conducted with pyOPT using the SNOPT optimizer [25]. The SUAV and wind models were identical to those described in Table 2, and the boundary and path constraints are detailed in Table 3, with the initial and final conditions based on the neurocontroller trajectory of Figure 13b. The trajectory optimization also used the cost function shown in Equation (41) to maximize the total distance traveled over the soaring cycle.

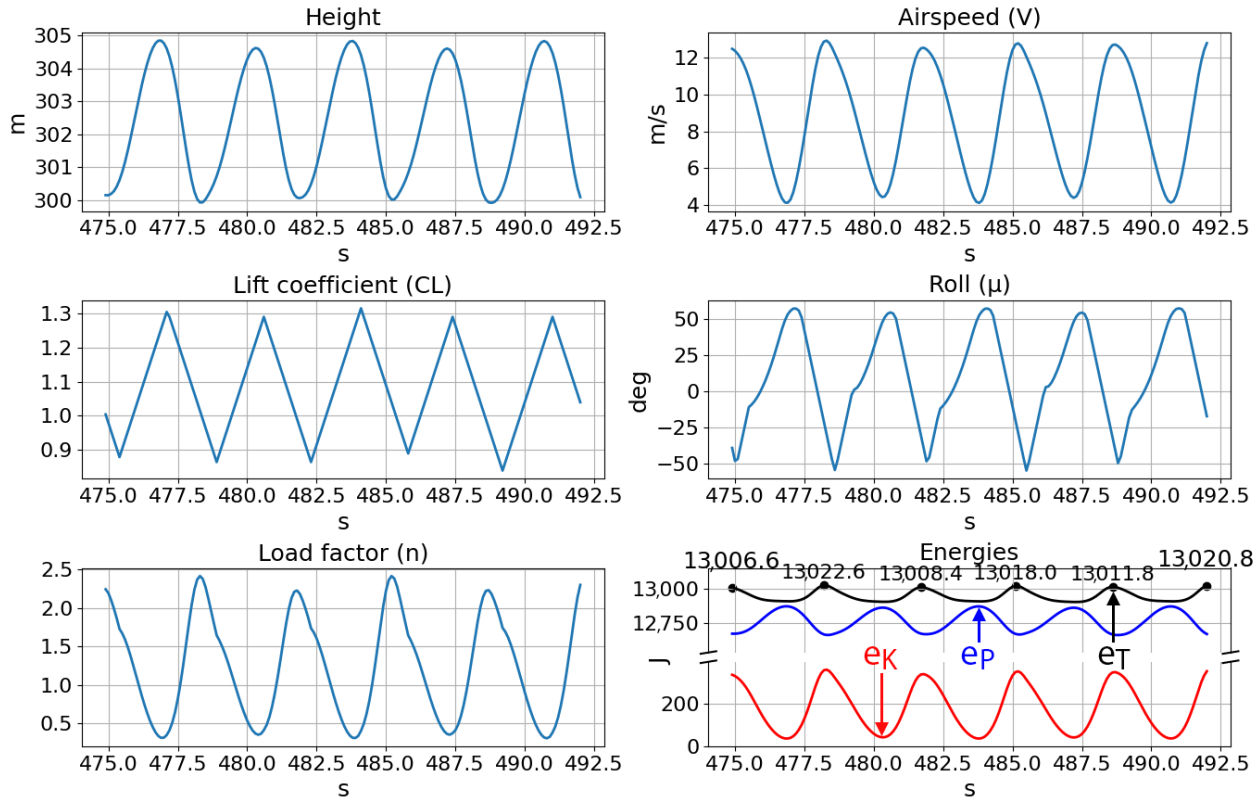



**Figure 15.** Trajectory trace of the dynamic soaring SUAV neurocontroller.

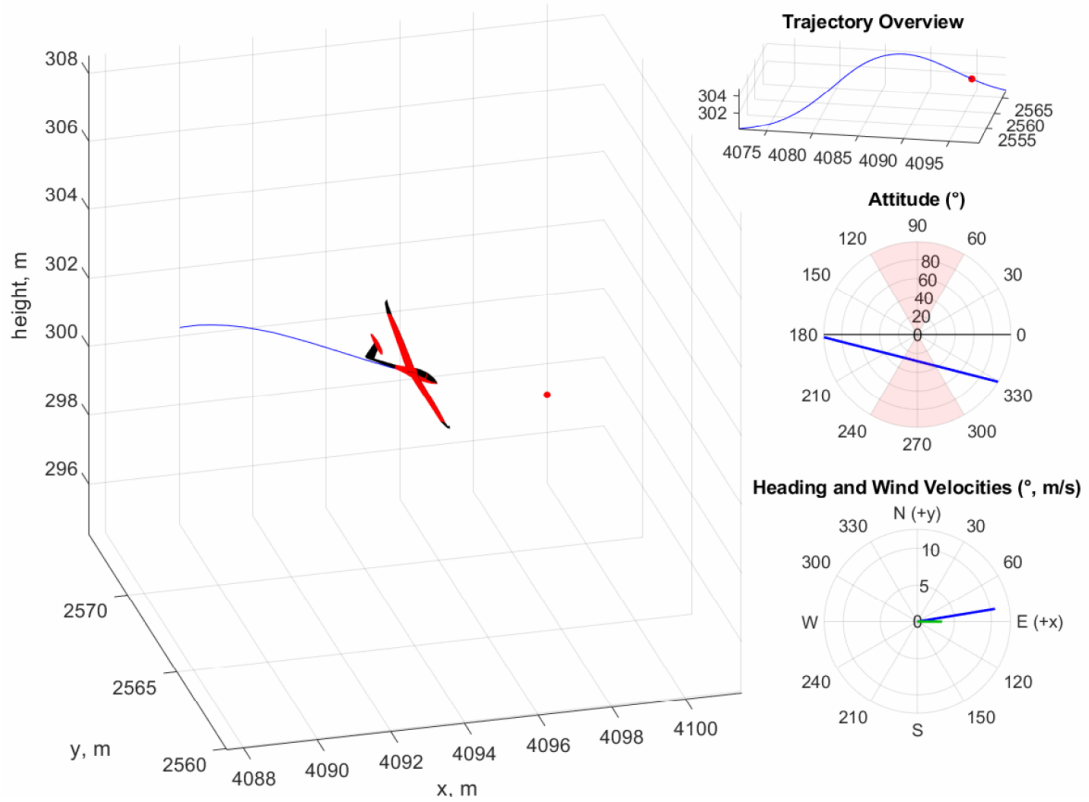



**Figure 16.** Aircraft visualization of the dynamic soaring SUAV neurocontroller (Supplementary Materials). The attitude indicator displays the roll angle along its circumference, as well as the pitch angle through the concentric circles within the plot. The shaded red zones represent the roll angle limits. The heading and wind velocities are shown by the blue and green vectors, respectively, where their orientations indicate direction and their lengths indicate the speed.

**Table 3.** Dynamic soaring SUAV trajectory optimization constraints.

| Boundary $t_0$ | Value | Boundary $t_f$ | Value | Path | Min | Max |
|---|---|---|---|---|---|---|
| $V_0$ (m/s) | 12.5 | $V_f$ (m/s) | 12.5 | $V$ (m/s) | 0.0 | 1000.0 |
| $\psi_0$ (deg) | 60.0 | $\psi_f$ (deg) | 60.0 | $\dot{\psi}$ (deg/s) | −100 | 100 |
| $\gamma_0$ (deg) | 0.0 | $\gamma_f$ (deg) | 0.0 | $\dot{\gamma}$ (deg/s) | −100 | 100 |
| $z_0$ (m) | 300.0 | $z_f$ (m) | 300.0 | $n$ | 0.0 | 15.0 |
| $x_0$ (m) | 0.0 | | | $C_L$ | −0.2 | 1.5 |
| $y_0$ (m) | 0.0 | | | | | |
| $C_{L_0}$ | 1.0 | $C_{L_f}$ | 1.0 | | | |
| $\mu_0$ (deg) | −40.0 | $\mu_f$ (deg) | −10.0 | | | |

The optimization process, consisting of 50 collocation points, took a CPU time of 1.55 min, nearly seven-times longer than the entire evolutionary process that produced the SUAV neurocontroller of Figure 11, which once trained, procedurally generates trajectories in real time. Furthermore, although the flight paths and traces of the two methods superimposed for comparison in Figures 17 and 18 show similar trajectories, the energy histories indicate that the optimized flight path ultimately cannot be used for dynamic soaring. While the boundary conditions were satisfied, the optimization process does not take into account the practical requirement for repeatable trajectories when computing individual cycles at a time. The singular focus on the cost function precludes any consideration of sustainable soaring, resulting in the decrease in the total energy after a single cycle. Attempting to calculate a longer trajectory comprised of multiple soaring cycles is also significantly more costly in terms of computation time and resources, and optimizations performed with twice the number of collocation points yielded identical results.

Many nonlinear programming solvers require an initial guess for the trajectory solution. Consequently, trajectory optimization results are directly dependent on this initial solution provided to the optimizer, and therefore, the computed optimal trajectories are local solutions by nature. In contrast, NEAT allows for the emergence of more global results, since the random member initialization and speciation mechanisms enable the exploration of numerous different network structures and weights, expanding the search space of potential trajectory and control solutions.

Another consideration of using pre-optimized trajectories is that they are decoupled from the control scheme. Either a control framework must be designed around the limits of the trajectory optimization method, or the optimization method must take into account the tracking control scheme prior to the lengthy computation. On the contrary, the presented neurocontroller scheme combines both trajectory planning and aircraft control by imposing and enforcing physical constraints during the evolutionary process. The control outputs and resulting flight path are direct reactions to the state of the aircraft and its environment, reducing the complexity of an autonomous system from a multilayered planning and control approach to one that combines both aspects of soaring. These considerations make it difficult to directly apply numerical trajectory optimization to soaring problems.

The relative simplicity of the NEAT-based neurocontrollers presented in this section becomes further apparent when comparing their topologies to those of related neural networks. For instance, a recent work involving deep neural networks for dynamic soaring control trained three separate neurocontrollers for the angle of attack, bank angle, and wing morphing parameter, each of which consisted of 5 network layers of 16 nodes with 1201 network weight values [12]. Along with the extensive dataset of 1000 optimal trajectories that was required for training, such neurocontrollers make interpretation and implementation on physical systems a significant challenge.

Another work by Li and Langelaan introduced a parameterized trajectory planning method that aimed to solve the lengthy computation times of numerical trajectory optimization [26]. The deep neural network resulting from the actor–critic reinforcement learning method used to generalize the parameterization approach consisted of an actor

and critic network, both of which were comprised of two fully interconnected layers, each with sixteen neurons. Considering that the decision-making actor network was solving only for parameters that represent a dynamic soaring trajectory and not the control commands themselves, the relative complexity of the neural network contrasts the simple yet effective NEAT-based neurocontrollers. These related works in the field of dynamic soaring showcase the typical complexities of deep neural networks and highlight the training and implementation advantages of the evolutionary neurocontrol approach.

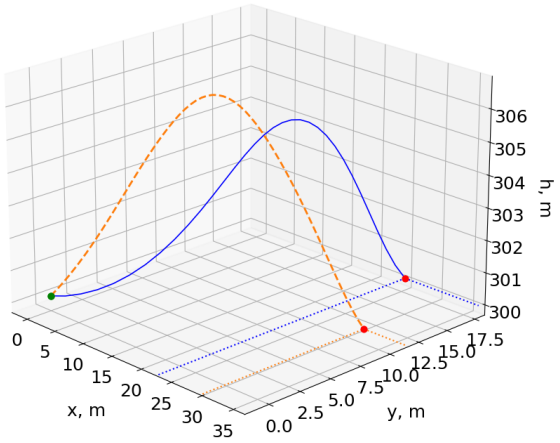



**Figure 17.** NEAT and TO simulated trajectories of the dynamic soaring SUAV. Neurocontroller in solid, optimized trajectory in dotted.

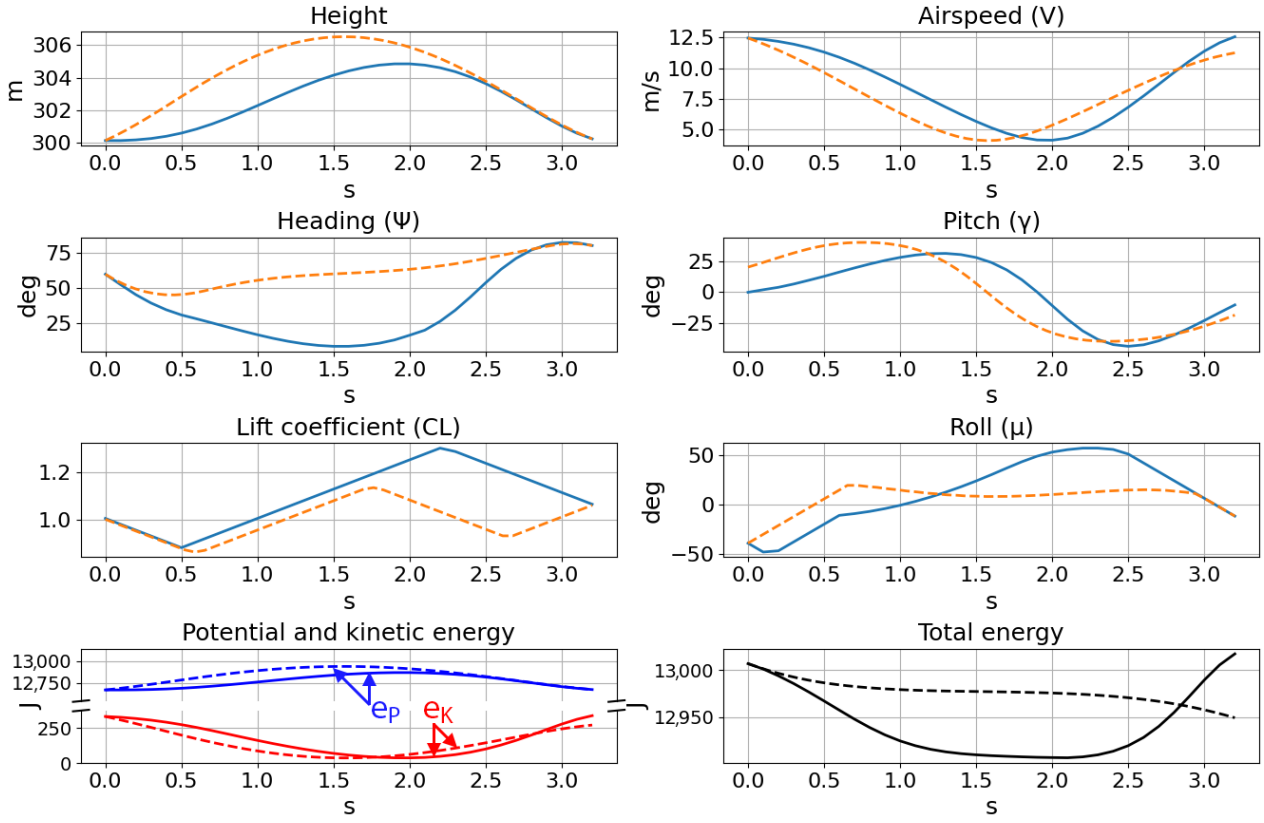



**Figure 18.** NEAT and TO traces of the dynamic soaring SUAV. The neurocontroller in solid lines; the optimized trajectory in dotted lines.

### *4.3. SUAV Thermal Soaring*

Table 4 details the parameters used to evolve and test the thermal soaring neurocontroller of Figure 19. The horizontal wind model used in the dynamic soaring test cases was disabled and replaced by the toroidal thermal bubble model described in Section 2.4.

**Table 4.** Thermal soaring SUAV neurocontroller simulation parameters.

| SUAV Parameters | Value | Wind Parameters | Value | Trajectory Parameters | Value |
|---|---|---|---|---|---|
| $g$ (m/s$^2$) | 9.8 | $W_{core}$ (m/s) | 3.05 | $V_0$ (m/s$^2$) | 9.1 |
| $\rho$ (kg/m$^3$) | 1.225 | $h_{t_0}$ (m) | 91.4 | $\psi_0$ (deg) | 0 |
| $m$ (kg) | 4.3 | $r_{xy}$ (m) | 30.5 | $\gamma_0$ (deg) | 0 |
| $S$ (m$^2$) | 1.0 | $r_z$ (m) | 61.0 | $h_0$ (m) | 106.7 |
| $C_{D_0}$ | 0.025 | $\dot{h_t}$ (m/s) | 0.213 | $x_0$ (m) | 0 |
| $E_{max}$ | 20.0 | | | $y_0$ (m) | $-38.1$ |
| $n_{max}$ | 15 | $W_{max_x}$ (m/s$^2$) | N/A | $h_{min}$ (m) | 0 |
| $C_{L_{max}}$ | 1.5 | $W_{max_x}$ (m/s$^2$) | N/A | $\dot{\gamma}_{max}$ (deg/s) | 100 |
| $C_{L_{min}}$ | $-0.2$ | | | $\dot{\psi}_{max}$ (deg/s) | 10 |
| $\mu_{max}$ (deg) | 60 | | | $\dot{C}_{Lmax}$ | 0.25 |
| | | | | $\dot{\mu}_{max}$ (deg/s) | 90 |
| | | | | $t_f$ (s) | 600 |

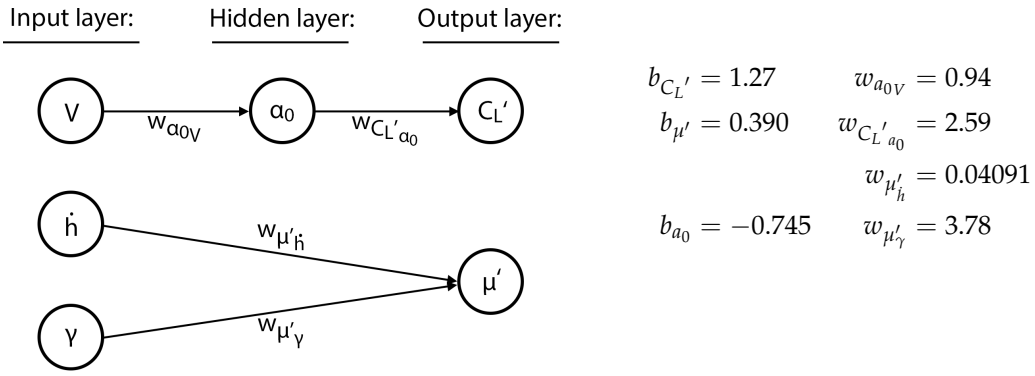


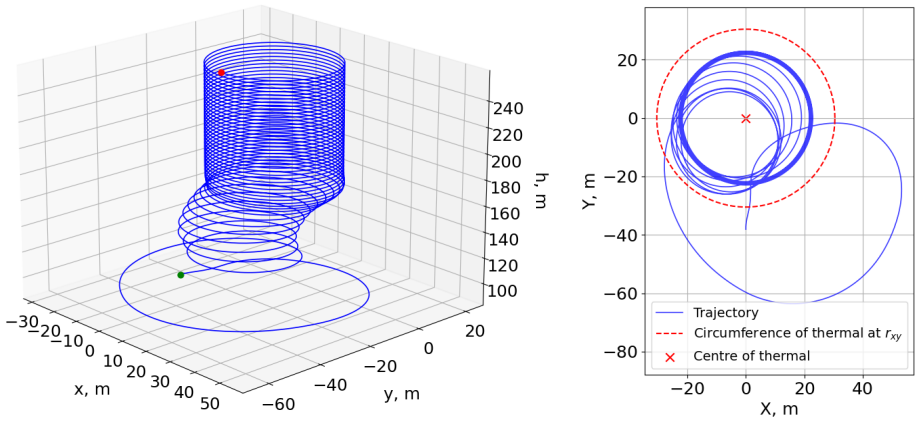

**Figure 19.** Topology of the thermal soaring SUAV neurocontroller.

The evolved neurocontroller has a single hidden neuron between the airspeed network input and the roll angle control output. The lift coefficient is a function of the height rate of change and the pitch angle, similar to the albatross network of Section 4.1, suggesting that despite the stochastic nature of the evolutionary process, there exists a set of typical connections that are more closely correlated with aerodynamics and control rather than any specific model of the wind or flight agent.

In addition, the fitness function used to generate this neurocontroller only consisted of penalties, unlike the reward-mechanism-containing fitness of the dynamic soaring test cases. The trajectories of Figure 20 demonstrate that an aversion to the extreme penalty of crashing the model was a sufficient motivator for the evolutionary process to develop soaring behavior. The controller initially finds one edge of the thermal bubble before circling around and centering the rising toroid at a radius from the thermal center where the updraft is sufficiently strong for a sustained climb.

Lastly, the trajectory histories of five soaring loops presented in Figure 21 show the smooth and simple states and controls of the neurocontroller. The roll is maintained at a constant 26.5 degrees, and the lift coefficient also remains at the maximum value so that the SUAV can remain in an optimal soaring region. For instance, at too great a radius from the thermal center, the updraft strength will be insufficient for continued soaring. On the contrary, at too small a radius, the roll angle required to circle the thermal will be greater, resulting in less lift acting on the aircraft, compromising future soaring cycles.

Due to these simple controls and the consequent behavior of the flight agent, the SUAV continually gains potential energy while its kinetic energy remains constant. In all, this test case demonstrated the developed neuroevolutionary method's applicability to other flight maneuvers.




(**a**) 47 cycles (600.0 s) (**b**) Bird's eye view

**Figure 20.** Simulated trajectories of the thermal soaring SUAV neurocontroller.

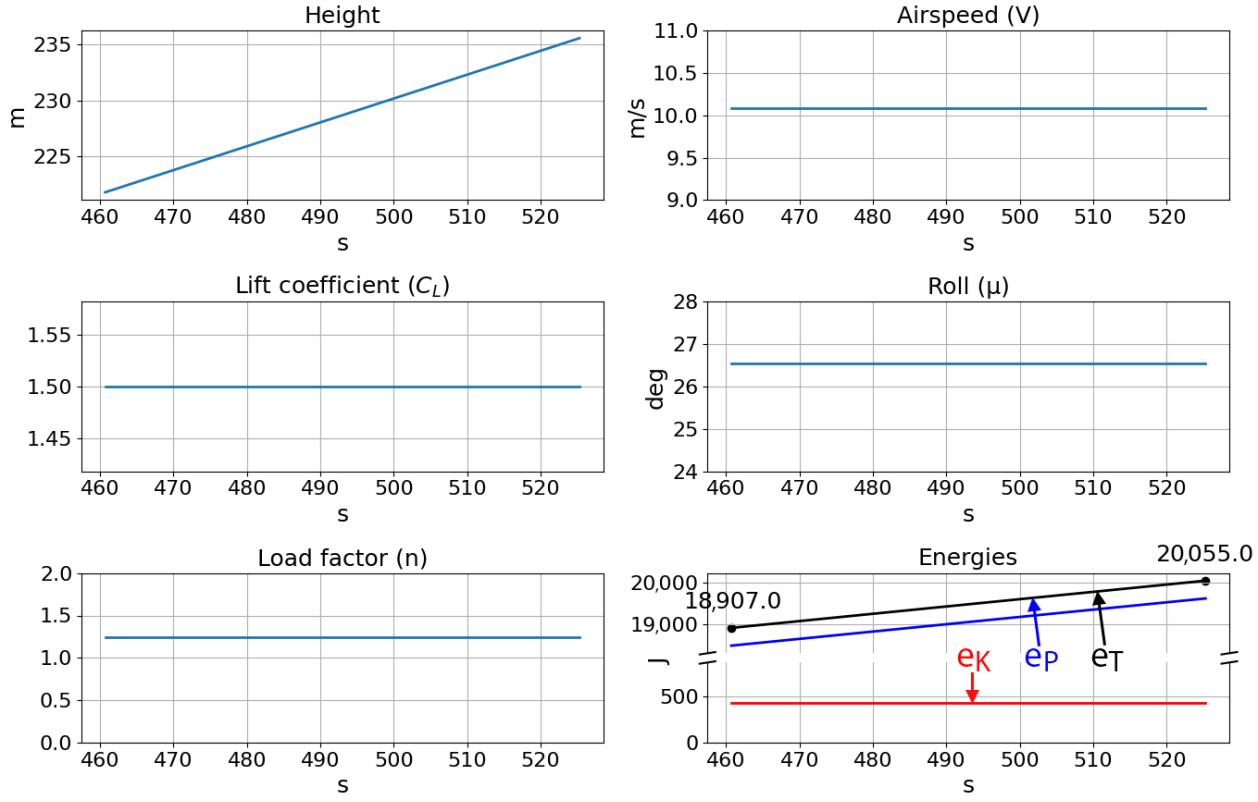



**Figure 21.** Trajectory trace of the thermal soaring SUAV neurocontroller.

## 5. Conclusions

This paper presented a method of evolving neurocontrollers for autonomous soaring based on the neuroevolution of augmenting topologies (NEAT) algorithm. The approach was shown to generate simple and efficient neural networks through three distinct test cases. The first used an albatross model to illustrate the similarities between a trained neurocontroller and the flights of biological albatrosses, demonstrating comparable energy extraction maneuvers; the second showed that the proposed approach can be applied to

an SUAV model for dynamic soaring at higher altitudes while comparing the resulting trajectories to those of other notable approaches; the third test case presented the method's applicability to thermal soaring, emphasizing the method's applicability to other flight techniques. Examining the resulting dynamic soaring trajectories against those of numerical trajectory optimization showed that the neuroevolutionary method generated more practical, cyclic flight paths that were generated in real time without extensive computation. Compared to other neural-network-based learning architectures for trajectory planning and aircraft control, the presented approach only requires a model of the aircraft and wind environment instead of extensive training datasets and yields extremely simple networks exhibiting smooth control signals that are traceable, interpretable, and ultimately implementable. Finally, although this work used deterministic models of the environment, the evolutionary scheme may be adapted for the creation of robust neurocontrollers that can perform soaring in stochastic conditions, moving further towards significantly reducing the energy requirements and flight limits of SUAVs.

**Supplementary Materials:** The following are available at https://www.mdpi.com/article/10.3390/aerospace8090267/s1.

**Author Contributions:** Conceptualization, R.E.P.; funding acquisition, R.E.P.; investigation, E.J.K. and R.E.P.; software, E.J.K.; supervision, R.E.P.; visualization, E.J.K.; writing—original draft, E.J.K.; writing—review and editing, E.J.K. and R.E.P. Both authors have read and agreed to the published version of the manuscript.

**Funding:** This research is sponsored by a Canadian Defence Academy Research Program Grant, No. 757734, Enhancing UAV Persistent Surveillance through AI-Driven Optimal Atmospheric Energy Extraction. The authors acknowledge as well a scholarship from Defence Research and Development Canada that enabled this research as part of a Master's program.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Sachs, G. Minimum shear wind strength required for dynamic soaring of albatrosses. *Ibis* **2005**, *147*, 1–10. [CrossRef]
2. Pennycuick, C.J. Thermal Soaring Compared in Three Dissimilar Tropical Bird Species, Fregata Magnificens, Pelecanus Occidentals and Coragyps Atratus. *J. Exp. Biol.* **1983**, *102*, 307–325. [CrossRef]
3. Sachs, G.; Traugott, J.; Nesterova, A.P.; Dell'Omo, G.; Kümmeth, F.; Heidrich, W.; Vyssotski, A.L.; Bonadonna, F. Flying at No Mechanical Energy Cost: Disclosing the Secret of Wandering Albatrosses. *PLoS ONE* **2012**, *7*, e41449. [CrossRef] [PubMed]
4. Allen, M. Autonomous Soaring for Improved Endurance of a Small Uninhabitated Air Vehicle. In Proceedings of the 43rd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, USA, 10–13 January 2005; American Institute of Aeronautics and Astronautics: Reno, NV, USA, 2005. [CrossRef]
5. Zhao, Y.J. Optimal patterns of glider dynamic soaring. *Optim. Control Appl. Methods* **2004**, *25*, 67–89. [CrossRef]
6. Sachs, G.; Grüter, B. Trajectory Optimization and Analytic Solutions for High-Speed Dynamic Soaring. *Aerospace* **2020**, *7*, 47. [CrossRef]
7. Shaw-Cortez, W.E.; Frew, E. Efficient Trajectory Development for Small Unmanned Aircraft Dynamic Soaring Applications. *J. Guid. Control Dyn.* **2015**, *38*, 519–523. [CrossRef]
8. Akhtar, N.; Whidborne, J.F.; Cooke, A.K. Real-time optimal techniques for unmanned air vehicles fuel saving. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2012**, *226*, 1315–1328. [CrossRef]
9. Gao, C.; Liu, H.H.T. Dubins path-based dynamic soaring trajectory planning and tracking control in a gradient wind field. *Optim. Control Appl. Methods* **2017**, *38*, 147–166. [CrossRef]
10. Lawrance, N.R.J.; Sukkarieh, S. A guidance and control strategy for dynamic soaring with a gliding UAV. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 3632–3637. [CrossRef]
11. Pogorzelski, G.; Silvestre, F.J. Autonomous soaring using a simplified MPC approach. *Aeronaut. J.* **2019**, *123*, 1666–1700. [CrossRef]
12. Kim, S.h.; Lee, J.; Jung, S.; Lee, H.; Kim, Y. Deep Neural Network-Based Feedback Control for Dynamic Soaring of Unpowered Aircraft. *IFAC-PapersOnLine* **2019**, *52*, 117–121. [CrossRef]

13. Montella, C.; Spletzer, J.R. Reinforcement learning for autonomous dynamic soaring in shear winds. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 3423–3428. [CrossRef]
14. Woodbury, T.D.; Dunn, C.; Valasek, J. 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, 2014. [CrossRef]
15. Chung, J.; Lawrance, N.; Sukkarieh, S. Learning to soar: Resource-constrained exploration in reinforcement learning. *Int. J. Robot. Res.* **2014**, *34*, 158–172. [CrossRef]
16. Muliadi, J.; Kusumoputro, B. Neural Network Control System of UAV Altitude Dynamics and Its Comparison with the PID Control System. *J. Adv. Transp.* **2018**, *2018*, 3823201. [CrossRef]
17. Efe, M.Ö. Neural Network Assisted Computationally Simple PID Control of a Quadrotor UAV. *IEEE Trans. Ind. Inform.* **2011**, *7*, 354–361. [CrossRef]
18. Bansal, S.; Akametalu, A.K.; Jiang, F.J.; Laine, F.; Tomlin, C.J. Learning Quadrotor Dynamics Using Neural Network for Flight Control. *arXiv* **2016**, arXiv:1610.05863.
19. Stanley, K.O.; Miikkulainen, R. Evolving Neural Networks through Augmenting Topologies. *Evol. Comput.* **2002**, *10*, 99–127. [CrossRef] [PubMed]
20. Perez, R.E.; Arnal, J.; Jansen, P.W. Neuro-Evolutionary Control for Optimal Dynamic Soaring. In Proceedings of the AIAA Scitech 2020 Forum, Orlando, FL, USA, 6–10 January 2020; American Institute of Aeronautics and Astronautics: Orlando, FL, USA, 2020. [CrossRef]
21. Rao, A. A Survey of Numerical Methods for Optimal Control. *Adv. Astronaut. Sci.* **2010**, *135*, 497–528.
22. Betts, J.T. Survey of Numerical Methods for Trajectory Optimization. *J. Guid. Control Dyn.* **1998**, *21*, 193–207. [CrossRef]
23. Miele, A. *Flight Mechanics, Theory of Flight Paths*; Addison-Wesley: Reading, MA, USA, 1962; Volume 1.
24. Lawrance, N.; Sukkarieh, S. Wind Energy Based Path Planning for a Small Gliding Unmanned Aerial Vehicle. In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Chicago, IL, USA, 10–13 August 2009; American Institute of Aeronautics and Astronautics: Chicago, IL, USA, 2009. [CrossRef]
25. Perez, R.E.; Jansen, P.W.; Martins, J.R.R.A. pyOpt: A Python-based object-oriented framework for nonlinear constrained optimization. *Struct. Multidiscip. Optim.* **2012**, *45*, 101–118. [CrossRef]
26. Li, Z.; Langelaan, J.W. Parameterized Trajectory Planning for Dynamic Soaring. In Proceedings of the AIAA Scitech 2020 Forum, Orlando, FL, USA, 6–10 January 2020; American Institute of Aeronautics and Astronautics: Orlando, FL, USA, 2020. [CrossRef]