NON-LINEAR APPROACH FOR UNMANNED, COMPOUND HELICOPTER CONTROL

Sara Waśniewska, Warsaw University of Technology, Warsaw, Poland
Sebastian Topczewski, Warsaw University of Technology, Warsaw, Poland

Abstract

The paper presents part of the work conducted within the project “Automatic Control of a Compound Helicopter” led by Warsaw University of Technology and sponsored by the BOEING Company. The aim of the paper is to present a proposal for the realisation of a control system for a small-scale compound helicopter utilizing one category of machine learning – reinforcement learning. For a given purpose, the theoretical foundations of reinforcement learning are presented and the DDPG methods, as a proposed approach for solving the control issue, is described. The comparison between the traditional control system and the way reinforcement learning works demonstrates the applicability of the artificial intelligence subcategory to control issues. The presented control object, helicopter ARCHER, is developed and evaluated in FLIGHTLAB software. An important part of the paper is the suggested implementation of the reinforcement learning method in MATLAB / SIMULINK software. Further plans for the development and testing of the control system are also revealed.

1. INTRODUCTION

Unmanned aerial vehicles, due to their wide range of capabilities, have found applications not only in the military field, but in recent years they have also started to appear in more and more branches of science, industry (Ref. 1). Especially aircraft with vertical take-off and landing (VTOL) capabilities such as multi-rotorcraft, helicopters attract interest. However, the aerodynamic phenomena (mainly stall and compressibility effects) occurring on the lifting rotors place limitations on any rotorcraft in terms of maximum achievable forward airspeeds. A solution could be compound helicopters, which are developed to obtain vehicles with simultaneous hovering maneuver capabilities and achieve high flight speeds. The applied additional components, such as pusher and puller propellers, lifting surfaces, enable the main rotor to be offloaded in term of the required values of the generated thrust and lift forces (Ref. 2). Thus, the main rotor’s angular velocity could be decreased, which not only reduces the required profile power, but also shifts the occurrence of undesirable aerodynamic phenomena to appear at higher flight speeds and at the same time increases the maximal airspeed value. Therefore, in a qualitative manner, it is possible to distinguish three operational ranges of compound helicopters: in hover and low airspeeds, in high airspeeds and in transition mode with medium airspeeds.

Crucial features of rotorcraft are their highly non-linear, complex dynamics and couplings between control channels, which considerably complicates the control of these inherently unstable objects (Ref. 3). Compared to classical configurations in compound helicopters, there is an extra need to consider the impact of added subsystems on aircraft performance and control approaches. Moreover, the necessity to ensure effective control of the vehicle in the three qualitatively described flight speed ranges impose additional requirements on the developed control system. In the case of aircraft with small dimensions and low masses, their sensitivity to any disturbance, wind gusts, and control signals becomes enhanced, which further complicates the control issue. Therefore, it is important to properly develop control systems according to the type of control objects and the scope of their operation. These systems can not only assist pilots in

stabilizing and performing simple maneuvers, but also automatically execute planned missions.

A review of the literature indicates that to date, a lot of work has been done to develop control systems for multi-rotorcraft and classic helicopters especially in hover and for low-speed maneuvers. Meanwhile, hardly any work has been encountered that deals with the control of small, compound helicopters.

The classical control approach associated with the use of proportional-integral-derivative controller, PID, is commonly encountered in industry. For example, the PID controllers are used in Ref. 4. However, this type of controller is appropriate for linear, SISO models, provides adequate operation only near the point of work for which it is designed. In order to ensure that the aircraft, as more complex systems, meet performance requirements while fulfilling certain physical limitations, the linear, optimal controllers have begun to be considered as more appropriate for multi-input, multi-output systems. The linear quadratic regulator, LQR, provides minimization of the cost function and gives better results than with the use of PID controller. In Ref. 5 is shown the application of LQR to control a small-scale helicopter. However, in the presence of parameter uncertainties and disturbances, the aforementioned controller is still inadequate, since regardless of the uncertainties' character, in LQR they are modelled as Gaussian, white noise. Because of linear controllers' limitations, there are eager attempts to control unmanned aircraft using non-linear techniques. In Ref. 6 a PID controller is supported with a feedback linearisation controller. In Ref. 7 a predictive control method is proposed. In Ref. 8 a controller using gradient descent optimization method is presented. In Ref. 9 a controller using robust and perfect tracking is developed. In Ref. 10 for the compound helicopter, the Pi–Sigma neural network adaptive controller is shown. An intriguing approach classified as non-linear methods is intelligent control using deep reinforcement learning, which allows the development of an optimal or suboptimal controller without any knowledge of the control object's dynamics model. Considering the subjects of unmanned aircraft, reinforcement learning (RL) might be applied in such areas as path planning (Ref. 11), navigation (Ref. 12) and control (Ref. 13). It is worth mentioning the strengths of hybrid systems, where by using more than one type of controllers, it is possible to merge the advantages of the individual controllers used. In Ref. 14 a combination of fuzzy-logic based and PID controllers is used. In Ref. 15 a fuzzy controller coupled with a model-based, Takagi-Sugeno controller are used.

The aim of this work is to present a theoretical consideration, a proposal to develop a control method for an unmanned helicopter using reinforcement learning and the implementation of the control system. Ultimately, a compound helicopter model made in FLIGHTLAB software will be used and the control system will be developed in MATLAB / SIMULINK software. During validation of the finished system, the data exchange between the two programs will be ensured. Due to the complex and non-linear dynamics of the chosen control object, it has been decided to consider using a neural network-based controller trained by a reinforcement learning algorithm to develop the control system. The choice of an approach involving one category of artificial intelligence is dictated partly by the possibility of obtaining a controller with similar performance to optimal controllers, as well as by the lack of requirement for knowledge of the control object's dynamics.

The following sections of the paper include the theoretical background of reinforcement learning, present the possibility of applying the discussed category of machine learning to design a control system for flying object, provide a detailed treatment of the agent training method, which is selected with regard to the solved issue, introduce the control model and elaborate on the proposed implementation of previously discussed theory.

2. FUNDAMENTALS OF REINFORCEMENT LEARNING

Reinforcement learning (RL), together with supervised and unsupervised learning, constitute the main categories of machine learning. However, RL does not rely on static data sets prepared before starting the learning process (as the other aforementioned categories), but requires an environment model that is the source of information – the data used in the training process – and that is closely related to the objective defined in the solved task. RL represents a computational approach to learning and decision-making related to a specific target (Ref 16).

The basis of RL is the interaction of some part of the software called an agent with dynamic environment. The agent, which is a learning component, at time step collects information about the current state of the environment, s_t, and the immediate reward, R_t, received from environment for previous actions, a_{t-1},

taken in the previous state of the environment, s_{t-1}, and based on this, takes actions, a_t, according to the actual policy π and thus the agent can influence a change in the environmental state. The environment is called the agent's operating space. The interaction of the two mentioned main elements is a dynamic and iterative process.

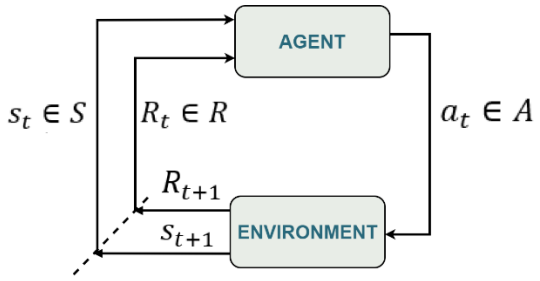

Figure 1: Interaction between the agent and the environment

In general, the acquired information about the real–number reward values is feedback on the correctness of the performer actions and is the only means enabling the agent to learn, since according to RL theory, any goal set for the agent can be transformed into the task of maximising the expected value of the return, G_t, defined as the sum of consecutive, discounted, immediate rewards received from the environment, $G_t \coloneqq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$,. The discount factor, $0 \leq \gamma \leq 1$, enables consideration of the tradeoff between exploration and exploitation of the environment by the agent during the training phase. The goal of agent training is to learn the optimal policy π^* to ensure that actions are taken that maximise the expected, cumulative, long – term reward. In the way of determining the value of the reward signal, the destination of the agent is taken into account, which includes the desired state of the environment. The agent is not given information about which actions in which states are appropriate, but is required to figure out on its own (usually by trial and error) which actions in specific states result in capturing the greatest reward value.

Formally, RL may be represented as a Markov Decision Process (MDP), which is a probabilistic model of a sequential decision-making process. In MDP, future states do not depend on the past, but only on current states and the actions taken in them, which is a so-called Markov property. MDP is described by tuples: $\{S, A, P, R\}$, where S is the state space, A is the action space, $P(s'|s,a) \coloneqq \Pr\{s_{t+1} = s', |s_t = s, a_t = a\}$ is the state transition function that determines the probability of passing to the next state of the environment, s', as a result of taking an action, a, in the current state, s, and $r(s,a) \coloneqq E[R_{t+1}|s_t = s, a_t = a]$ is the reward function defining the expected value of the immediate reward, R_{t+1}, received when transitioning between successive states s and s' as a result of taking action a. In RL, the environment model is assumed to be uncontrollable and beyond the knowledge of the agent, so that there might be stochastic changes in the environmental states with probability distributions unknown to the agent. In addition, the agent has no influence on the parameters of the environment, including the disturbances acting on it. The agent can modify only its behaviour, the parameters of the established strategy structure. The part of the agent responsible for mapping states into actions is the policy, which in non-deterministic form is referred to as conditional probability: $\pi(a|s) \coloneqq \Pr\{a_t = a|s_t = s\}$. In addition to policy, the agent also includes a reinforcement learning algorithm, which varies depending on the applied specific RL method and, therefore, the issue being solved, and which is used during the training phase. Also important in RL are state value functions, $v^{\pi}(s) \coloneqq \mathrm{E}[G_t|s_t = s]$, or state-action value functions, $q^{\pi}(s,a) \coloneqq \mathrm{E}[G_t|s_t = s,\ a_t = a]$, which are defined as the expected values of the returns received when initiating actions in the current state and indicate the benefits of being in a given state and taking actions that are in line with the current strategy. The quantity searched for is the optimal policy which, for each state, results in taking the optimal action, thus maximising the long-term reward, so also realising the goal of RL. Optimal policies correspond to optimal value functions.

Theoretically issues being defined as MDPs could be solved using Bellman's recursive optimality equations (1) and (2), from which optimal value functions and optimal policies are determined.

$$v^{\pi}(s) = \max_{a} \mathrm{E}[R_{t+1} + \gamma v^{\pi*}(s_{t+1})|s_t = s, a_t = a\ ] \quad (1)$$

$$q^{\pi}(s,a) = \\ = \mathrm{E}\left[R_{t+1} + \gamma \max_{a'} q^{\pi*}(s_{t+1}, a')\middle| s_t = s, a_t = a\right] \quad (2)$$

However, most often in real applications it is not possible to directly use the given equations, due to impossibility of simultaneously satisfying three assumptions: accurate knowledge of the dynamics of the environment in the form of a transition function, having sufficient computational resources, satisfying the Markov property. For given reasons, methods that give approximate solutions to Bellman optimality equations are used. In the RL algorithms, collected

experience in the form of information about transitions between successive environment states is used in place of expected values to solve Bellman optimality equations. The agent's ability to learn the optimal strategy represents the ideal case that is aimed when using RL algorithms. For issues with small and finite state and action spaces, tabular methods are used, in which the approximated quantities are stored in the form of tables or matrices. However, it is much more common to encounter issues with continuous or large spaces, for which it would be impracticable to present the relevant quantities in tabular form, therefore it is appropriate to represent policies and/or value functions using function approximators, which can be neural networks. By virtue of neural networks versatility, almost any complex input-output (state-action) relationship could be mapped, and the same, unchanging network structure could be used with different environments. One of the advantages of using RL for approximate solving of MDP issues is that, despite using only approximated quantities, it is possible to train the model in such a way that it generates actions that more closely correspond to optimal actions for more frequently occurring states while reducing the time spent learning the correct actions for less frequently occurring ones.

3. THE USE OF REINFORCEMENT LEARNING AS A CONTROL METHOD

One possible approach to the control issue, different from the methods known from classical control theory, is the approach using reinforcement learning. When considering the anatomy of RL, it could be easy to perceive the similarity to the issue of control, as in both cases the aim is to determine the appropriate input signals that will result in the desired system behaviour. It could be noted that RL and classical control theory use slightly different approaches and terminology to achieve the same effect. Classical control methods mainly rely on models, while control using RL is based on collected experience, which can be used to control objects that are difficult to model or have unknown model dynamics. The development of the agent together with the police can be compared with the development of a controller, which converts the current input (observed state and reward value) into a control signal (action), which in turn affects the control object (environment). The reference signal can be included in both the reward function and the observed state of the environment. The given approach is correct because, from the RL point of view, the environment is everything except the agent, so the environment includes not only the control object, by also feedback and reference signals, disturbances and noise. The reward value generated using a reward function designed for the individual solved task should reflect the goals of the agent’s performance, since in RL this is the only feedback signal indicating the correctness or incorrectness of the undertaken actions, and therefore also showing whether the agent is learning in the desired way, corresponding to what the system designer envisioned. In addition, the reward function could be compared to the cost function known from optimal control theory. This is due to the desire in both approaches to achieve optimal values of certain quantities. In optimal control, the most common goal is to achieve the minimum value of certain quality indicator, while in RL the aim is to maximise received rewards, which is associated with the search for approximations of optimal strategies that will ensure taking action that result in the highest possible reward values in an unknown and dynamic environment. For this reason, it is encountered to call RL the optimal control method. Figure 2 schematically illustrates the analogies between a traditional control system with feedback and a system using RL.

4. DDPG ALGORITHM FOR HELICOPTER CONTROL

In order to solve the unmanned helicopter control issue, it is proposed to use the model-free, online and off-policy DDPG (Deep Deterministic Policy Gradient) method belonging to the group of actor-critic methods, which are currently among the more commonly used reinforcement learning techniques. In actor-critic methods, there is approximation of both policies and value functions. This makes it possible to use the desired properties of both types of methods. Techniques using value functions are distinguished by the use of estimates to update the estimated quantities, which introduced bias and causes dependence of the results on the quality of approximated functions. On the other hand, methods based on policy approximations, despite their tendency to reach local minima and slow learning rates, allow implementation of algorithms in real time and are applied to tasks with continuous action spaces.

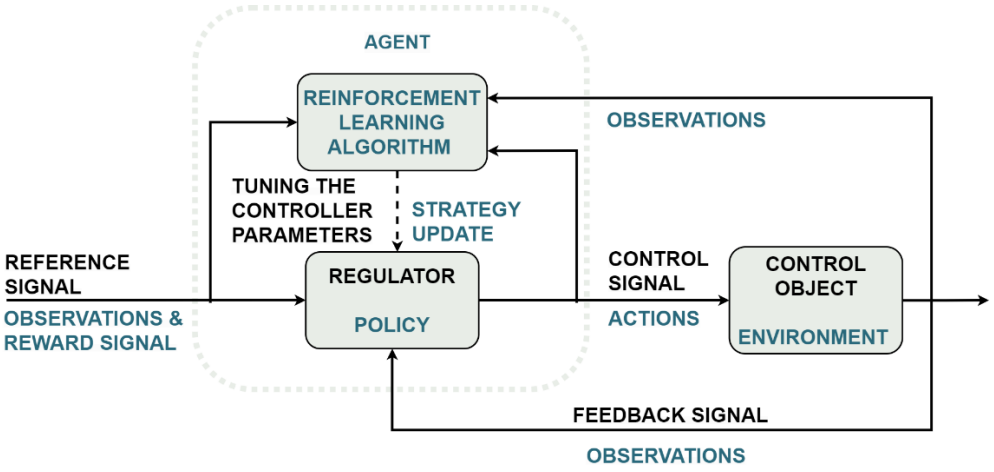

Figure *2*: Comparison of RL with traditional control approach

The purpose of applying the critic, associated with the approximation of the value function, is to assess the correctness of the action taken by the agent in accordance with the applicable policy, which in turn is associated with the other part of the agent – the actor. The process of policy learning, whose approximation takes place using two neural networks (the actor and critic), begins with the delivery of the observed state to both networks. The actor, following the current policy, through taken actions changes the state of the environment, and additionally provides the critic with information about the types of actions taken. The critic network predicts the value of the value function on the basis of the observed state-action pair. At a given time step, the estimated value function is compared with an also estimated value function, but obtained from the immediate reward received from the environment and the discounted value function for the current state. The difference calculated between the two estimated values of the value function is used to update the critic's network parameters. Subsequently, the actor's network parameters are adjusted with the critic's recommended direction of greatest increase in reward value being used instead of using the reward signal directly (Ref. 17).

The proposed DDPG method is used with continuous state and action spaces and is chosen primarily because it is a crucial and willingly used technique in control issues (Ref. 17). DDPG uses a value function approximation and a deterministic method to determine the strategy gradient as it combines the two methods: DQN (Deep Q-Networks) and DPG (Deterministic Policy Gradient). Actor-critic agent to estimate the deterministic strategy, μ, and the corresponding state-action value function, Q^{μ}, uses the Belman equations and an approach from the off-policy method characterized by the use of two strategies: one which is being improved to obtain its optimal form and the other used to explore the state-action space and generate data. The DDPG method uses approximators of four relationships: the actor's network (based on observations, it returns actions that maximise the long-term reward), the actor's target network (introduced to improve the stability of optimization; the parameters of the target network are updated periodically using the latest parameters of the actor's network), the critic's network (based on observations and undertaken actions, returns the expected long – term reward) and the critic's target network (introduced to improve the stability of optimization; as in the actor's target network, the parameters are adjusted with the use of the critic's network parameters). During training, neural network parameter values are tuned, and after completion of the training process, they are maintained at the tuned values (Ref. 18). An important issue is the use of the typical for DQN method's elements: replay buffer, which is a memory in which the agent's experience is stored in the form of a certain amount of data samples, and an additional target network. Both the actor and critic networks are updated using random samples drawn from the replay buffer to ensure that there is no correlation between the data used for training.

In the DDPG it is convenient to use the following form of the Bellman equation for the state-action value function:

$$Q^{\mu}(s_t, a_t) = E\left[R_{t+1} + \gamma E\left[Q^{\mu}\left(s_{t+1}, \mu(s_{t+1})\right)\right]\right] \quad (3)$$

The policy is defined using a greedy approach as $\mu(s) = \arg\max_{a} Q^{\mu}(s, a)$. The parameter vector of the

critic's network, referred to as Θ^Q and corresponding to the approximated function Q^μ, is determined by minimizing the loss function $L(\theta^Q)$:

$$L(\theta^Q) = E[(Q^\mu(s_t, a_t|\theta^Q) - R_{t+1} - \gamma Q^\mu(s_{t+1}, \mu(s_{t+1})|\theta^Q))^2] \quad (4)$$

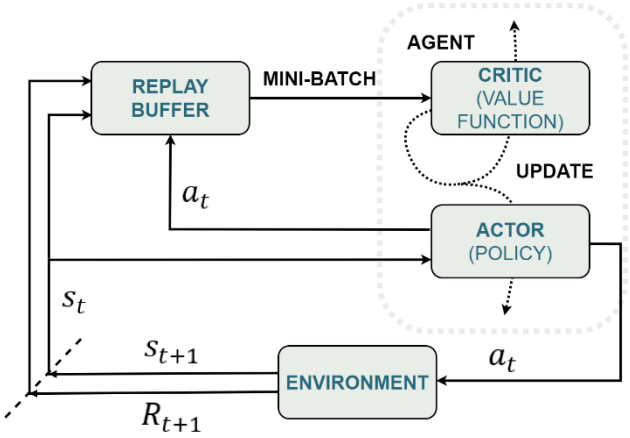

Figure 3: Scheme of the DDPG method

In order to ensure the convergence of the learning process of the critic's network by delaying the determination of the value function, the critic's target network $Q^{T\mu}$ is used with the corresponding parameter vector $\Theta^{TQ}_{\mathrm{ANN}}$. Moreover, also to avoid the appearance of divergence when calculating Q^μ, an actor target network μ^T with a parameter vector $\Theta^{T\mu}_{\mathrm{ANN}}$ is introduced. The parameters of both the actor's and critic's target networks are varied in a smooth manner using the hyperparameter – smoothing factor τ:

$$\Theta^T_{\mathrm{ANN}} \leftarrow \tau\Theta_{\mathrm{ANN}} + (1-\tau)\Theta^T_{\mathrm{ANN}} \quad (5)$$

Utilizing techniques form the DPG method makes it possible to include an actor neural network with parameters Θ^μ_{ANN} representing an approximation of the deterministic strategy $\mu(s|\Theta^\mu_{ANN})$ which maps current states directly to actions taken by agent instead of using action probabilities. The actor's network is trained using a gradient method, where the quality function is defined as $J(\Theta^\mu_{ANN}) = E[G_t]$, and its gradient is approximated by the following relation:

$$\nabla_{\Theta^\mu} J(\Theta^\mu_{ANN}) \approx E[\nabla_a Q^\mu(s = s_t, a = \mu(s_t)|\Theta^Q_{ANN}) \nabla_{\Theta^\mu} \mu(s = s_t|\Theta^\mu_{ANN})] \quad (6)$$

One of the advantages of using the off-policy DDPG method is the ability to take into account and decouple the issue of exploration from the RL algorithm. To better ensure exploration, noise is added to the deterministic actions taken by agent during the training phase. Originally, temporally correlated Ornstein-Uhlenbeck noise was recommended, however, uncorrelated Gaussian noise also gives good results (Ref. 19). Due to the addition of a noise value sampled from a random process, $\mathcal{N}$, a new policy is constructed, $\mu'(s_t) = \mu(s_t|\Theta^\mu_{ANN}) + \mathcal{N}$. However, it should be remembered that noise introduction only takes place during the learning phase, not during system testing.

The algorithm of the DDPG method is shown in table 1 (Ref. 18, 20). M stands for the number of episodes with a maximum of T time steps in each episode.

5. CONTROL OBJECT

The control object is the unmanned helicopter ARCHER, which was created in 2018 for a project conducted at Warsaw University of Technology (Ref. 21). The aircraft provides a research platform for easily acquiring different helicopter configurations by adding components such as puller and pusher propellers, wings and stabilators. In the basic version, the helicopter has one main rotor and tail rotor, both powered by separate electric motors. The basic parameters of the ARCHER in classical configuration are shown in table 2.

Figure 4: ARCHER in classical configuration

At present, the physical model of the used helicopter exists in classical configuration. The helicopter simulation model developed in FLIGHTLAB corresponds to a physical one (Ref. 22). There are also developed numerical models of the helicopter in the compound configuration including thrust-compounding configuration with additional pusher propeller mounted at the end of the tail boom and fully-compounding configuration which includes vertical and horizontal stabilizers, wings and two puller propellers symmetrically located on the lifting surfaces.

The two-blade main rotor is modelled using the blade element method. Its aerodynamic loads determined with a quasi-steady model are validated from ARCHER baseline flight data, and its induced velocity is determined with a Peters-He Six state model.

Table 1: DDPG algorithm

DDPG algorithm

1) Randomly initialize the network parameter vectors Θ^{μ}_{ANN} and Θ^{Q}_{ANN} for the actor's network, $\mu(s|\Theta^{\mu}_{ANN})$, and critic's network, $Q^{\mu}(s,a|\Theta^{Q}_{ANN})$, respectively

2) Initialize the target network parameter vectors $\Theta^{T\mu}_{ANN} = \Theta^{\mu}_{ANN}$ and $\Theta^{TQ}_{ANN} = \Theta^{Q}_{ANN}$ for the actor's target network, $\mu^{T}(s|\Theta^{T\mu}_{ANN})$, and critic's target network, $Q^{T\mu}(s,a|\Theta^{TQ}_{ANN})$, respectively

3) Initialize replay buffer

for $\boldsymbol{episode = 1 : M}$ **do**

a) Initialize a random process $\mathcal{N}$ for action exploration

b) Receive initial observation state, s_1

for $\boldsymbol{t = 1 : T}$ **do**

i) Select action $a_t = \mu(s_t|\Theta^{\mu}_{ANN}) + \mathcal{N}_t$ according to the current policy and exploration noise

ii) Execute action a_t, observe new state s_{t+1} end reward R_{t+1}

iii) Store experience $(s_t, a_t, R_{t+1}, s_{t+1})$ in replay buffer

iv) Sample a random minibatch of N experiences $(s_i, a_i, R_{i+1}, s_{i+1})$ from replay buffer

v) If s_{i+1} is a terminal state, set $y_i = R_i$. Otherwise, set:

$$y_i = R_{i+1} + \gamma Q^{T\mu}(s_{i+1}, \mu^{T}(s_{i+1}|\Theta^{T\mu}_{ANN})|\Theta^{TQ}_{ANN})$$

vi) Update critic by minimizing the loss function:

$$L = \frac{1}{2N}\sum_{i}\left(y_i - Q^{\mu}(s_i, a_i|\Theta^{Q}_{ANN})\right)^2$$

vii) Update the actor parameters using the sampled policy gradient:

$$\nabla_{\Theta^{\mu}} J(\Theta^{\mu}_{ANN}) \approx \frac{1}{N}\sum_{i}\nabla_a Q^{\mu}(s = s_i, a = \mu(s_i)|\Theta^{Q})\nabla_{\Theta^{\mu}}\mu(s = s_i|\Theta^{\mu}_{ANN})$$

viii) Update the target networks:

$$\Theta^{T\mu}_{ANN} \leftarrow \tau\Theta^{\mu}_{ANN} + (1-\tau)\Theta^{T\mu}_{ANN}$$
$$\Theta^{TQ}_{ANN} \leftarrow \tau\Theta^{Q}_{ANN} + (1-\tau)\Theta^{TQ}_{ANN}$$

end

end

Table 2: Basic ARCHER characteristics.

Characteristic	Metric
Main rotor diameter	1,78 m
No. of MR blades	2
MR RPM	2'000
Tail rotor diameter	0,158 m
No. of TR blades	2
TR RPM	7'200
MR & TR rotorhubs	rigid, no flapping
TOW	8 kg
MR blades airfoil	NACA23012
TR blades airfoil	NACA23012

The tail rotor, pusher propeller and two puller propellers are modelled in the same way – they have two blades each and their loads are derived using more numerically efficient disk theory. In addition, because the use of rigid type for all rotors, the blade flapping is prevented. The two symmetrical wings and the horizontal and vertical stabilizers attached to the end of the tail boom are modelled as single segment lifting surfaces. The fuselage is modelled as a rigid object with 6 DOF, which aerodynamic characteristic are determined by the CFD model.

The control of the ARCHER depends on its configuration. In the case of all the aforementioned configurations, the control is carried out by selected control variables typical for the classical part of the helicopter (main rotor longitudinal and lateral cyclic pitch angles, main rotor collective pitch angle and tail rotor collective pitch angle with the control variables respectively XB, XA, XC, XP), and depending on the additional components included, there is also a possibility of using variable rotational speeds and collective pitch angles of the puller and pusher propellers. The ability to independently change both mentioned rotors' parameters makes it possible to apply different control strategies for additional components in the overall control system.

Figures 5 – 7 show exemplary performance for fully compound helicopter model developed in FLIGHTLAB software. Figures 5 and 6 present examples of the aircraft's trim variables in forward flight, namely helicopter roll (ϕ) and pitch (θ) angles, lateral stick, longitudinal stick, collective stick and pedal positions as a function of total airspeed, labelled as "speed". Furthermore, figure 7 shows collective pitch angles (Θ) for both right, R, and left, L, puller propellers. For clarity, the right propeller has positive lateral coordinates in the conventional, aircraft coordinate system. The notations used are as follows: Θ_0 and Ω are the initially preset values of collective pitch angles and angular velocities of the puller propellers respectively, "classical configuration" corresponds to the helicopter's basic version.

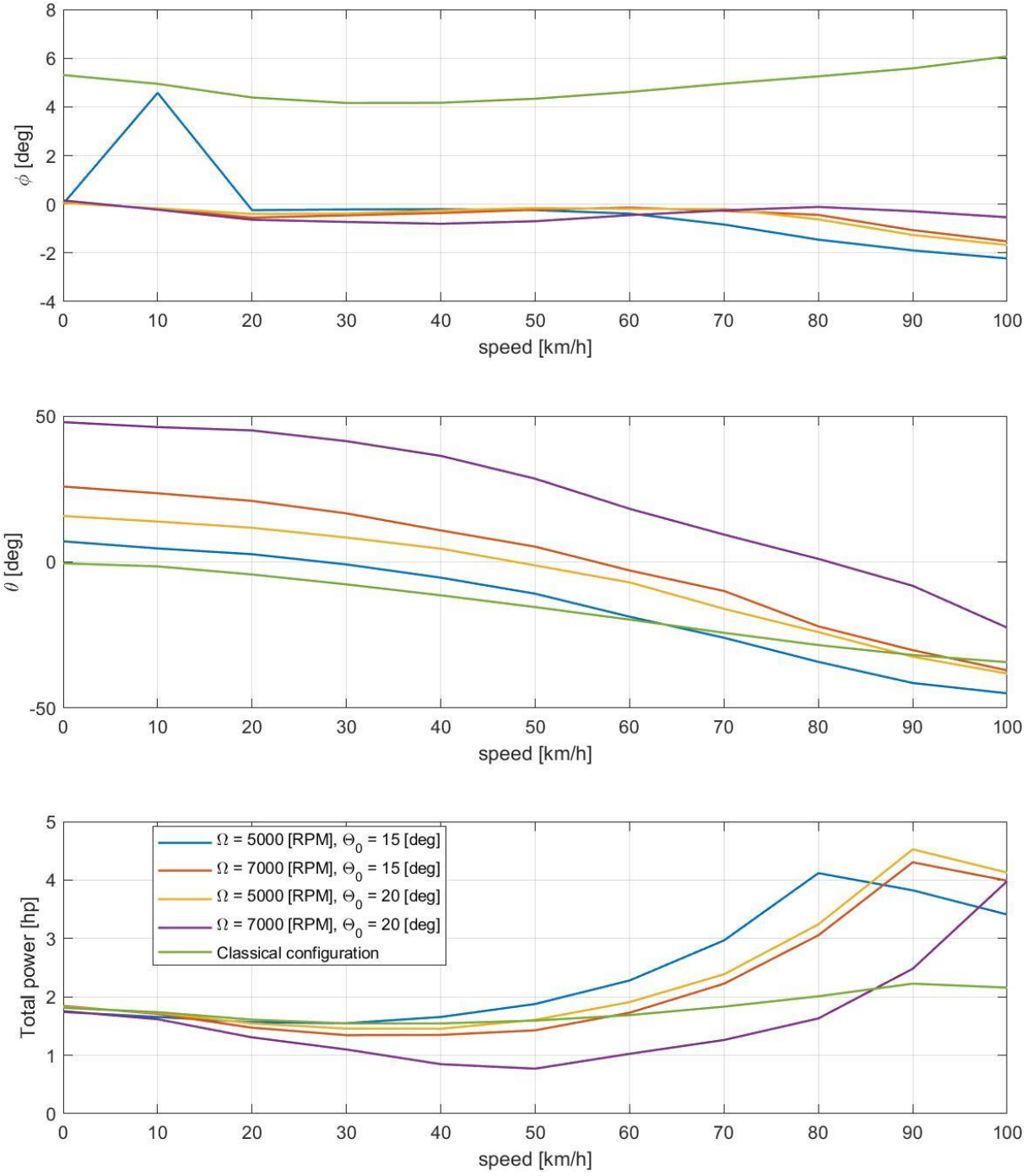

Figure 5: Attitude roll and pitch angles and total required power in trimmed forward flight

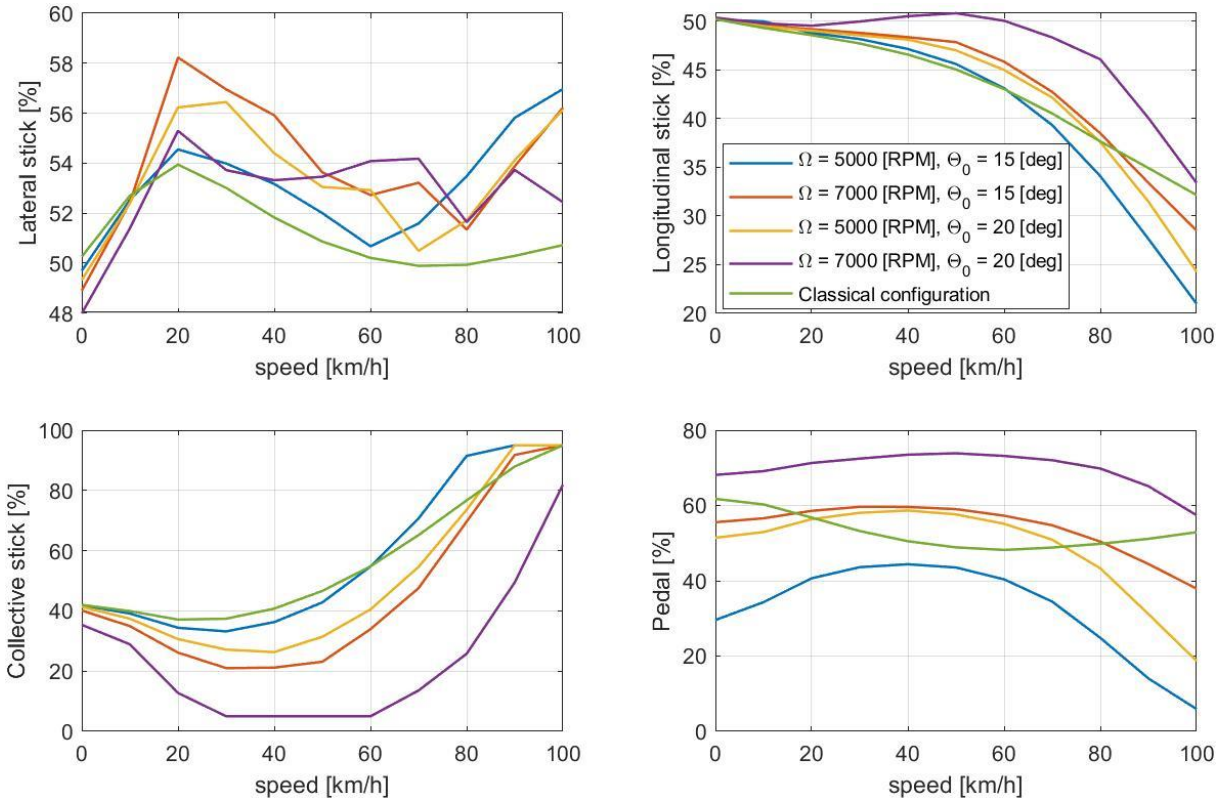

Figure 6: Control variables in trimmed forward flight

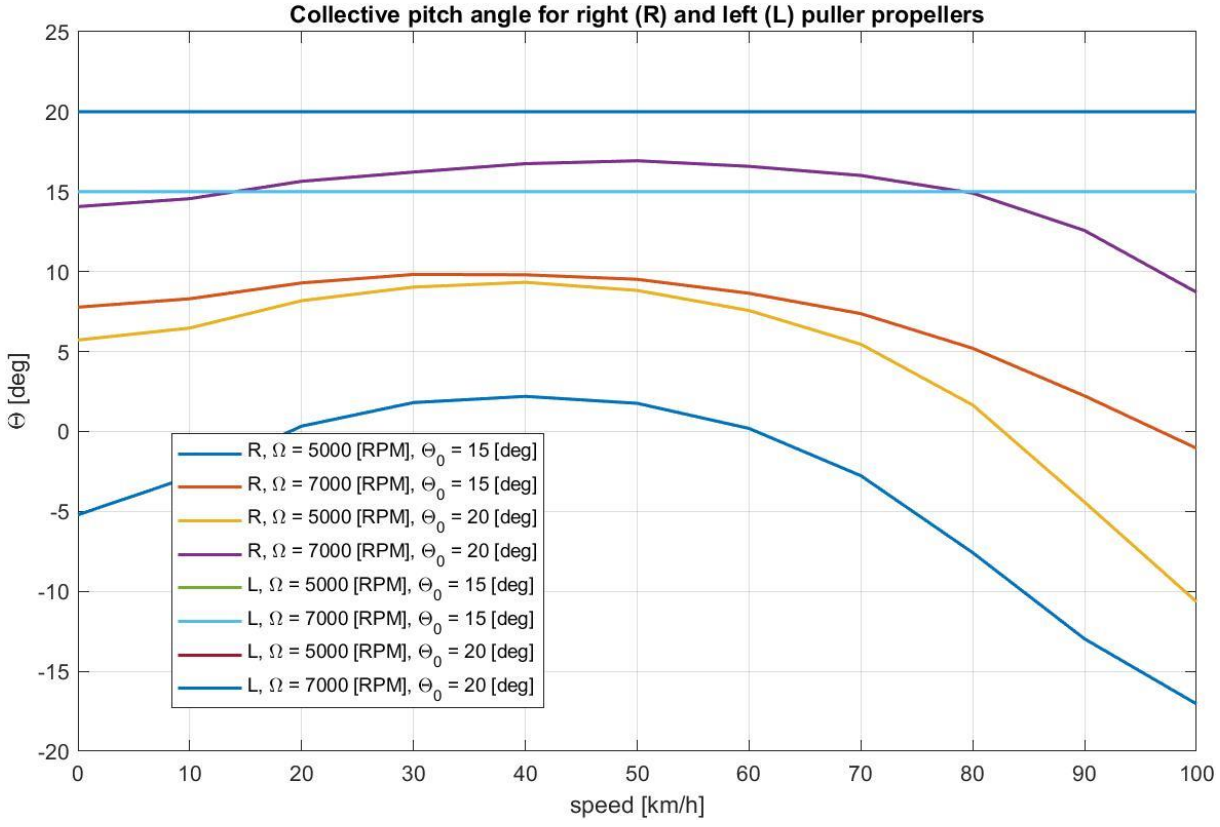

Figure 7: Collective pitch angles of the puller propellers in trimmed forward flight

The curves in figures 5 – 7 show performance for fully compound ARCHER's configuration for different combinations of constant, initial values of pulling propellers' parameters and for one wing angle of incidence with respect to the classical configuration. Despite the assumption of fixed pitch values, for the right propeller they are changeable as a function of speed. This is due to the necessity to compensate for the drag torque from the main rotor.

The aircraft model is fully ready to be used as a control object in an automated control system, the development of which is the step currently being performed and required to complete the overall concept of the work.

6. IMPLEMENTATION

In the given chapter, a proposition for implementing one of the RL methods to create a control system for an unmanned helicopter is described. In particular, the proposal for the values of the parameters used, the reward function, the observed variables and control signals are highlighted. It should be mentioned that the information shown relates to the implementation of a control system for a helicopter in a classical configuration. The choice is driven by two reasons:

1) Successful agent training for a simpler helicopter model (in the given case in the classical configuration) will allow confirmation of the correctness of the selected method, chosen settings, values of variable coefficients and parameters, reward function, as well as the proposed type and number of observed variables and control variables. In addition, subsequent modifications of the system carried out to adapt it to the helicopter's compound configuration might by simpler than the direct correct creation of the system for the compound helicopter.

2) Separate development of the control system for the classical part of the helicopter makes it possible to use a hybrid control system for the compound configuration, in which additional components specific to compound aircraft (puller and pusher propellers) could be controlled using controllers of a different type than RL.

The chosen programming environment in which the control system is to be developed is MATLAB / SIMULINK software.

In the initial stage of training, the helicopter model in classical configuration is implemented in the form of state equations, and the matrices required for it are taken from the linearised aircraft model, which in turn is derived from the non-linear numerical model developed in FLIGHTLAB. As the agent training stage progress, there will be an opportunity to directly use the non-linear helicopter model through data exchange at each time step between the two programs. The observed environmental state variables used by the agent are: helicopter position in the inertial frame of reference (x, y, z), helicopter linear velocities in the body frame of reference (V_x, V_y, V_z), helicopter roll, pitch and yaw angles in the body frame of reference (ϕ, θ, ψ), helicopter angular velocities in the body frame of reference (p, q, r), as well as the differences between the reference and current position coordinates (Δx, Δy, Δz), yaw angle ($\Delta\psi$) and angular velocities (Δp, Δq, Δr). Considering differences is supposed to help the agent learn to adapt to various states of the environment (Ref. 23). Before feeding the observed state variable to the agent, the given quantities are normalized, which is a necessary activity when using neural networks (Ref. 24). The establisher means of normalization is to divide the appropriate signals at each time step by the fixed values, which for variables related to position, linear velocities, attitude angles and angular velocities are respectively 1000, 300, 1/2 π and 100/180 π. The output signals from the agent (actions) and, simultaneously, the input to the control object are four control variables typical for a classical helicopter configuration: XB, XA, XC, XP. When converting the control system to a version corresponding to a compound helicopter, it is proposed to add variable rotational speeds and/or collective pitch angles of the puller/pusher propellers as additional control variables – additional actions taken by agent.

It is worth noting that in the fully compound ARCHER model, the variable XP is distributed equally between the two puller propellers and changes the pitch angles by increments of the same value (in one propeller adding and in the other subtracting an increment). However, this does not compensate for the torque from both puller propellers, since the torque is not linearly dependent on the collective pitch angles. Therefore, it is necessary to take into account supplementary propeller control that will ensure generation of such forces and moments that it will be possible to execute of the preset manoeuvres.

The agent, which is created using MATLAB's built-in functions rlQValueFunction and rlContinousDeterministicActor, consists of two neural networks: a critic

and an actor, whose architectures and characteristic features are shown in figures 8 and 9.

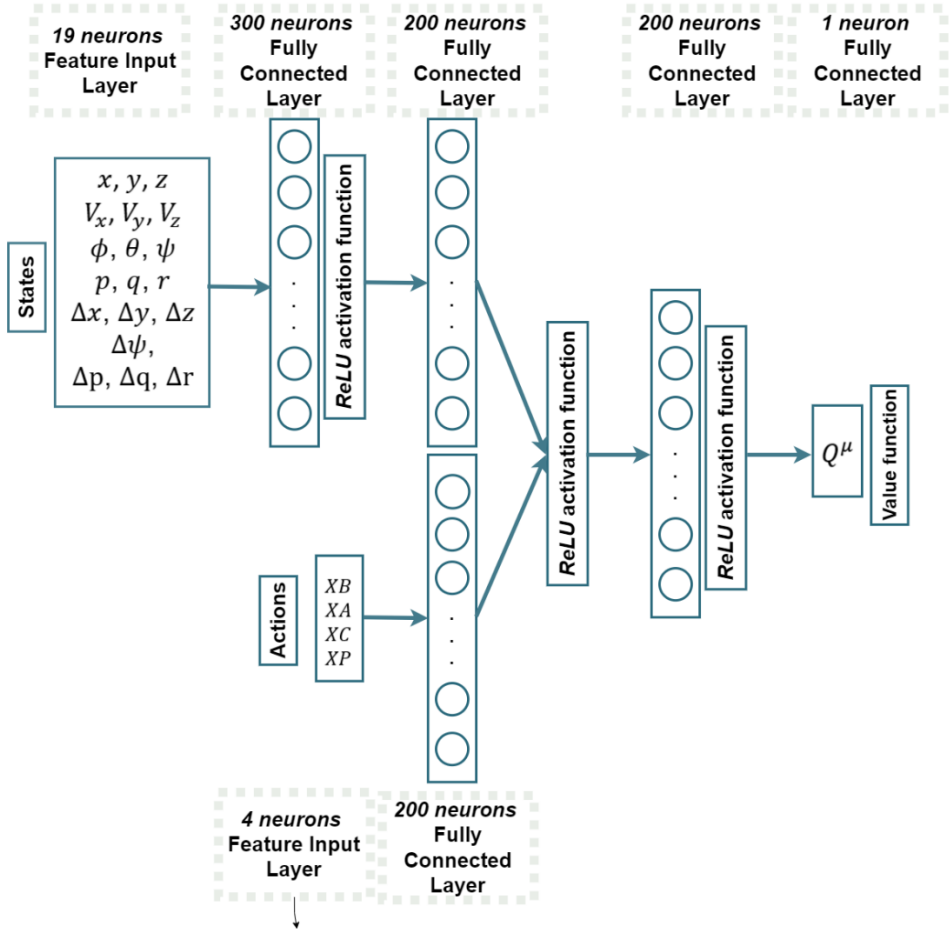

Figure 8: Critic neural network for DDPG agent

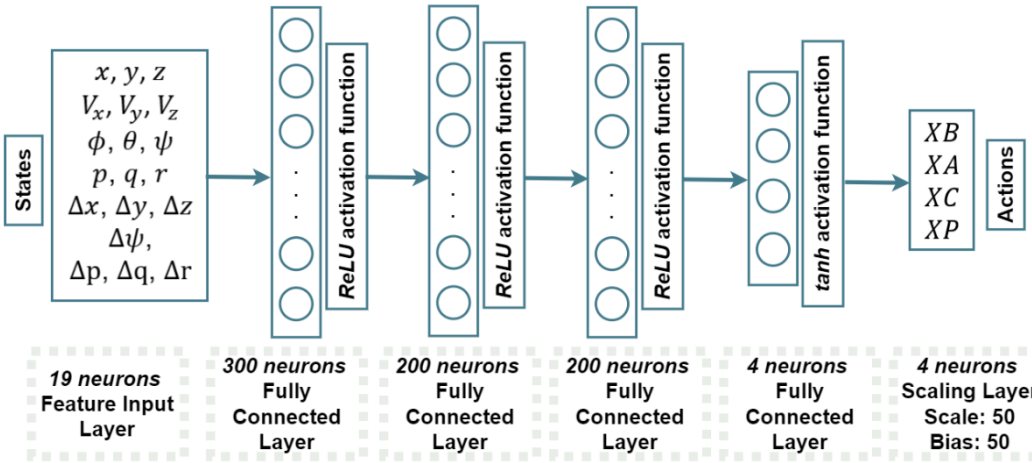

Figure 9: Actor neural network for DDPG agent

Despite the risk of adverse phenomena associated with the nonlinear tanh activation function (such as neuron saturation and the associated zero or small gradient flow through the network in the process of tuning network parameters) (Ref. 25), it is decided to use that activation function in the last layer of the

actor network, which is due to the easy scaling of the output of the tanh function and the need to have action signals in a limited range from 0 to 100.

The other proposed settings related to the DDPG agent and the training process are listed below. The sampling time is 0.01 seconds. Learn rate, typical for optimization algorithms, is assumed to be $5 \cdot 10^{-4}$ for both actor and critic networks. The threshold value for the gradient, above which it is clipped, is 1. The chosen optimizer to train the network is algorithm *adam* (Ref. 26). The size of experience buffer, in which the training-gathered experience is stored, has a length of 10^6. During each training episode, an experience mini-batch of size 256 is sampled from the experience buffer to calculate gradients for updating the critic's network parameters. The value of the smooth factor parameter is set to 0.001, and the discount factor, applied to future rewards during training and related to the tradeoff between exploitation and exploration of the environment, is set to 0.99. The important parameters for the DDPG agent are the parameters in the noise model used, which according to the MATLAB documentation is Ornstein – Uhlenbeck action noise. Considering the remark that for continuous action signals the value of standard deviation, std, should satisfy the relation that $std \cdot \sqrt{sampling\ time}$ is equal to $1 \div 10\%$ of the action range, where actions take the values is $1 \div 10$, $std = 50$ is applied. Standard deviation decay rate, related to the number of samples after which the value of the std decreases by half is 0.0015.

The general idea of the agent training process is to achieve such control of the helicopter which allows flight along a preset trajectory. To achieve this, it is necessary to start the learning with the simplest tasks. As a first step, the agent is planned to learn how to keep the helicopter in one place (the initial position is the same as the target position and the agent is only expected to compensate for non-zero angular velocities, non-zero differences between the current and preset yaw angles and obtain roll and pitch angles as small and as close as possible to those occurring during hovering maneuver). With the progress of the training process, the distance between the initial and preset positions of the helicopter is to be increased while maintaining the requirements of leading the control object to the preset location and then keeping the hover condition.

The reference state variables (with the subscript *ref*) have zero values, except for the coordinates of the three-dimensional position, which can have any values from the assumed range. However, it should be remembered that in the hovering state, the helicopter's orientation angles - pitch and roll - are not zero, their values are derived from equilibrium conditions. Only the reference yaw angle may be correctly assumed as zero. The given issue is reflected in the reward function defined in the following part of the present chapter. The initial environmental state variables (with a subscript of *0*) are updated at the beginning of each episode using a custom reset function. The initial linear velocities, angular velocities and attitude angles are assumed to have zero values each time an episode is started, and only the initial position is variable with random position coordinate values from an established range around the reference position coordinates.

Before training the agent's network, the maximum number of episodes and time steps in each episode are determined. The first value depends on the agent's rate of learning and the second one is assumed to be 1000 with time step of 0.01 seconds. Averaging the scores, rewards, and number of steps is assumed with a window length of 10. The terminal state ending each episode occurs at the moment of execution of the maximum number of time steps or the occurrence of a helicopter collision, which happens when the limit angles of orientation are exceeded (±π for roll, ± π /2 for pitch and ±2 π for yaw) or when the helicopter is outside the accepted area with horizontal coordinates from -100 to 100 ft and vertical coordinate from 0 to 200 ft.

The crucial element affecting the agent's ability to learn correctly, encouraging the agent to pursue the goal set in the task is the reward function. It could be defined in many ways depending on the creativity of the author of the system designer, the solved issue and the set goal. The proposed reward function includes the following elements, with the used notation dK indicating the absolute value of the difference between the initial or current value of the variable and the corresponding reference quantity, $dK = |K_0 - K_{ref}|$, and the notation dKD corresponds to the analogous quantity from the previous time step.

1) Reward, r_T, received for keeping and object alive:

$$r_T = 10^4 \cdot (0.5 + time) \tag{7}$$

2) Penalty, r_P, for finding the helicopter in a position away from the reference position:

$$r_{p1} = -25 \cdot (10 + time) \cdot \sqrt{dx^2 + dy^2 + dz^2} \quad (8)$$

$$r_{P2} = \begin{cases} 800, & \sqrt{dx^2 + dy^2 + dz^2} \leq 0.5 \\ 0, & \sqrt{dx^2 + dy^2 + dz^2} > 0.5 \end{cases} \quad (9)$$

$$r_{P3} = -15 \cdot (10 + time) \cdot (dx + dy + dz) \quad (10)$$

$$r_P = r_{P1} + r_{P2} + r_{P3} \quad (11)$$

3) Penalty, r_A, for deviations of attitude angles from zero values:

$$r_A = -1000 \cdot (10 + time) \cdot (d\phi + d\theta + 2 \cdot d\psi) \quad (12)$$

4) Penalty, r_R, for non-zero angular velocities:

$$r_R = -1000 \cdot (10 + time) \cdot (dp + dq + dr) \quad (13)$$

5) Penalty/reward, r_D, for moving away/approaching selected state variables from/to reference values in successive time steps:

$$r_{D1} = -10^5 \cdot (dx - dxD + dy - dyD + dz - dzD) \quad (14)$$

$$r_{D2} = -10^7 \cdot (dp - dpD + dq - dqD + dr - drD) \quad (15)$$

$$r_{D3} = -5 * 10^6 \cdot \big(d\phi - d\phi D + d\theta - d\theta D + 2 \cdot (d\psi - d\psi D)\big) \quad (16)$$

$$r_D = r_{D1} + r_{D2} + r_{D3} \quad (17)$$

After summing the listed components and dividing the result by the constant value, the total reward function is as follows:

$$r = \frac{r_T + r_P + r_A + r_R + r_D}{10000} \quad (18)$$

Analogous to the observed state variables, it is recommended to normalize the reward signal before providing it to the neural network. An approach using a logarithmic function for reward rescaling is proposed (Ref. 24). The normalized reward is given by (19):

$$r_{norm} = sign(r) \log_{10}(1 + |r|) \quad (19)$$

The variability of the coefficients as a function of time is due to the need to acquire the right proportions between the values of rewards and penalties at different states of the environment and the number of time steps passed in each episode. The included variability is also intended to encourage the agent to keep for a longer time the helicopter in a state close to the desired one and to punish the agent for keeping the object in a state far from the reference one for a long time.

When designing the reward function, it is important to examine a given function for the presence of local extremes, since their appearance can prevent convergence during the learning process.

7. STATUS OF THE WORK

At the given stage of work helicopter models made in FLIGHTLAB software are prepared for implementation in the designed control system. Regarding control system development, the agent training phase is currently in progress to achieve control of the classical helicopter configuration, for which the linearised ARCHER model is used. This is a long-running part of the control system design process, as it requires testing many combinations of tuned parameter values and improving the self-defined reward function. After successfully completing a given step, the trained agent will be examined for correct control of the linearised helicopter model. First, it is planned to test the ability of performing hovering, then to reach set position in space and the possibility of doing turns (by changing the yaw angle). After that, retraining of the agent is expected to occur with the use of a more accurate, non-linear model of the helicopter in classical configuration. Having a controller developed in a given way, it will be possible to develop a hybrid control system for a compound helicopter, in which additional components will be controlled using a different type of controller than RL. In case of choosing to execute the control system exclusively using RL, any settings and functions used for the successful developed controller for the classical configuration will be able to be transformed accordingly in order to adjust them to the compound configuration. However, the agent training process will then be required again. The test campaign of the final control system will include the above-mentioned tasks checking the correctness of the pre-developed control system of the classical configuration, and also will be performed a flight of the compound helicopter along the set trajectory. Further plan for the development of the work is to acquire a control system that ensures proper control of the helicopter both in hover and at high forward speeds, achievement of which is desirable especially for compound configurations.

8. CONCLUSION

The paper addresses a proposal for the design and implementation of an unmanned helicopter control system using reinforcement learning. For the given purpose, the foundations of RL are discussed and DDPG methods, which is addressed to issues similar

to those presented within the paper, is detailed. The analogies presented between the traditional control system and the way RL works highlight the possibility of applying the discussed subcategory of artificial intelligence to control issues. In the paper the ARCHER helicopter model, which is created in FLIGHTLAB software and which will be included as the control object in the finally developed control system, is described. Lastly, the control system implementation proposal, the current status of the work and plans for further development and testing of the system being created are presented.

Author contact:

Sara Waśniewska sara.wasniewska.stud@pw.edu.pl
Sebastian Topczewski sebastian.topczewski@pw.edu.pl

9. ACKNOWLEDGMENTS

Research conducted as part of the grant entitled "Automatic control of a compound helicopter" coordinated by the Warsaw University of Technology, sponsored by the Boeing Company.

10. REFERENCES

1. Tijani, I. B., Akmeliawati, R., Legowo, A., Budiyono, A., Abdul Muthalif, A. G., "H∞ robust controller for autonomous helicopter hovering control," *Aircraft Engineering and Aerospace Technology*, Vol. 83, (6), 2011, pp. 363-374. DOI: 10.1108/00022661111173243.

2. Ormiston, R. A., "Revitalising advanced rotorcraft research – and the compound helicopter," *The Aeronautical Journal*, Vol. 120, (1223), 2016, pp. 83-129.

3. Leishman, J., *Principles of Helicopter Aerodynamics*, Cambridge University Press, New York, NY, 2000.

4. Bayisa, A. T., "Controlling Quadcopter Altitude using PID-Control System," *International Journal of Engineering Research & Technology (IJERT),* Vol. 8, (12), 2019.

5. Budiyono, A., Wibowo, S. S., "Optimal Tracking Controller Design for a Small Scale Helicopter," *Journal of Bionic Engineering*, Vol. 4, (4), pp. 271-280, 2007.

6. Huang, H., Hoffmann, G. M., Waslander, S. L., Tomlin, C. J., "Aerodynamics and Control of Autonomous Quadrotor Helicopters in Aggressive Maneuvering," 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, May 2009, DOI: 10.1109/ROBOT.2009.5152561.

7. Guerreiro, B., Silvestre, C., Cunha, R., "Terrain Avoidance model Predictive Control for Autonomous Rotorcraft," Proceedings of the 17th World Congress The International Federation of Automatic Control, Seoul, Korea, July 6-11 2008.

8. Kadmiry, B., Palm, R., Driankov, D., "Autonomous Helicopter Control Using Gradient Descent Optimization Method," 4th Asian Conference on Robotics and its Applications, ACRA 2001, Singapore, June 6-8 2001.

9. Chen, B. M., "Design and Implementation of a Flight Control System for an Unmanned Rotorcraft using RPT Control Approach," *Asian Journal of Control*, Vol. 15, (1), pp. 1-25, 2013.

10. Zheng, F., Xiong, B., Zhang, J., Zhen, Z., Wang, F., "Improved neural network adaptive control for compound helicopter with uncertain cross-coupling in multimodal maneuver," Springer, 2022, DOI: 10.21203/rs.3.rs-714233/v1.

11. Song, Y., Steinweg, M., Kaufmann, E., Scaramuzza, D., "Autonomous Drone Racing with Deep Reinforcement Learning," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, 2021.

12. Chikhaoui, K., Ghazzai, H., Massoud, Y., "PPO-based Reinforcement Learning for UAV Navigation in Urban Environments," 2022 IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS), 2022, DOI: 10.1109/MWSCAS54063.2022.9859287.

13. Bou-Ammar, H., Voos, H., Ertel, W., "Controller Design for Quadrotor UAVs using Reinforcement Learning," IEEE Multi-conference on System and Control, Yokohama, Japan, Sept. 8-10, 2010.

14. Sanchez, E. N., Becerra, H. M., Velez, C. M., "Combining fuzzy, PID and regulation control for an autonomous mini-helicopter," *Information Sciences*, Vol. 177, (10), 2007, pp. 1999-2022, DOI: 10.1016/j.ins.2006.10.001.

15. Kadmiry, B., Driankov, D., "Fuzzy Control of an Autonomous Helicopter," IFSA World Congress and 20th NAFIPS International Conference, Vancouver, Canada, July 25-28, 2001.

16. Sutton, R. S., Barto, A. G., *Reinforcement Learning: An Introduction*, The MIT Press, London, England, 2017, Chapters 1, 3, 13.

17. Azar, A.T., Koubaa, A., Mohamed, N.A., Ibrahim, H.A., Ibrahim, Z.F., Kazim, M., Ammar, A., Benjdira, B., Khamis, A.M., Hameed, I.A., Casalino, G., "Drone deep reinforcement learning: a review," MDPI Electronics, Vol. 10, (9), 2001, DOI: 10.3390/electronics10090999.

18. The MathWorks, Inc., 2023, Deep Deterministic Policy Gradient (DDPG), accessed 21 June 2023, https://ch.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html.

19. Deep Deterministic Policy Gradient, accessed 21 June 2023, https://spinningup.openai.com/en/latest/algorithms/ddpg.html.

20. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., "Continous control with deep reinforcement learning," Proceedings of the International Conference on Learning Representations (ICLR 2016), London, UK, 2016.

21. Bedyńska, A., Bibik, P., Curyło, P., Czarnota, N., Ceniuk, M., Grabowski, Ł., Łukasiewicz, M., "ARCHER – Autonomous Reconfigurable Compound Helicopter for Education and Research," 45th European Rotorcraft Forum, Warsaw, Poland, Sept. 17-20, 2019.

22. Topczewski, S., Bibik, P., Waśniewska, S., Cioć, T., "Automatic Flight Control System for the Small-scale Compound Helicopter," 48th European Rotorcraft Forum, Winterthur, Switzerland, Sept. 6-8, 2022.

23. Stray, T. J., "Application of deep reinforcement learning for control problems," Master's thesis, Norwegian University of Science and Technology, 2019.

24. Carton, F., "Exploration of reinforcement learning algorithms for autonomous vehicle visual perception and control," Institut Polytechnique de Paris, 2021.

25. Stanford University School of Engineering, Lecture 6, Training Neural Networks, accessed 27 July 2023, https://www.youtube.com/watch?v=wEoyxE0GP2M&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=6.

26. Kingma, D.P., Ba, J. L., "Adam: a method for stochastic optimization," Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, 2015.