Received January 7, 2022, accepted February 7, 2022, date of publication February 10, 2022, date of current version February 24, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3150926

A Deep Deterministic Policy Gradient Learning Approach to Missile Autopilot Design

ANGELO CANDELI[1], GIANMARIA DE TOMMASI[2], (Senior Member, IEEE), DARIO GIUSEPPE LUI[2], (Member, IEEE), ADRIANO MELE[3], STEFANIA SANTINI[2], (Member, IEEE), AND GAETANO TARTAGLIONE[4], (Member, IEEE)

[1]MBDA Italia SpA, 00131 Rome, Italy
[2]Department of Electrical Engineering and Information Technology, University of Naples Federico II, 80125 Naples, Italy
[3]Dipartimento di Economia, Ingegneria, Società e Impresa (DEIM), Università Degli Studi Della Tuscia, 00110 Viterbo, Italy
[4]Department of Engineering, University of Naples Parthenope, 80143 Naples, Italy

Corresponding author: Gaetano Tartaglione (gaetano.tartaglione@uniparthenope.it)

ABSTRACT In this paper a Deep Reinforcement Learning algorithm, known as Deep Deterministic Policy Gradient (DDPG), is applied to the problem of designing a missile lateral acceleration control system. To this aim, the autopilot control problem is recast in the Reinforcement Learning framework, where the environment consists of a 2-Degrees-of-Freedom nonlinear model of the missile's longitudinal dynamics, while the agent training procedure is carried out on a linearized version of the model. In particular, we show how to account not only for the stabilization of the longitudinal dynamic, but also for the main performance indexes (settling-time, undershoot, steady-state error, etc.) in the DDPG reward function. The effectiveness of the proposed DDPG-based missile autopilot is assessed through extensive numerical simulations, carried out on both the linearized and the fully nonlinear dynamics by considering different flight conditions and uncertainty in the aerodynamic coefficients, and its performance is compared against two model-based control strategies in order to check the capability of the proposed data-driven approach to achieve prescribed closed-loop response in a completely model-free fashion.

INDEX TERMS Missiles, autopilot, reinforcement learning.

I. INTRODUCTION

An autopilot system for a modern missile must be able to stabilize the missile rotational dynamics and to effectively track the sequence of acceleration commands provided by the navigation and guidance system to follow the desired trajectory. Generally, to achieve these objectives, missile autopilots are designed exploiting classical model-based approaches, mainly relying on linearization of nonlinear dynamics and gain scheduling control (see for example [1] and the references therein). However, since the closed-loop performance might be significantly deteriorated by the presence of highly nonlinear terms in the plant dynamics [2], several nonlinear control strategies have been proposed to tackle this issue, ranging from sliding mode approaches [3] to backstepping [4], to nonlinear model predictive control [5] and H_∞ techniques [6], [7]. All these solutions are model-based and require accurate knowledge of the plant dynamics. However, this may be a restrictive assumption, as in practice unavoidable uncertainties arise due to unmodeled dynamics, time-varying parameters, or unpredictable environmental effects. When modeling becomes difficult or inaccurate, a data driven-based approach to control design might prove advantageous. To this aim, a field of machine learning known as Reinforcement Learning (RL) [8], [9] has recently attracted a wide research interest, thanks to its ability in learning an optimal control policy by exploring an unknown environment with the objective of maximizing a numerical reward signal, without any precise description of the plant.

RL algorithms proposed in literature can be classified according to two different paradigms, namely *model-based* and *model-free*, depending on the assumed knowledge of the environment model [10]. Although the approaches that belong to the first class, i.e. the model-based RL, have been extensively investigated in real applications (see for example [11] and the references therein), they are generally designed under the restrictive assumption that model information is available to the agent. Therefore, the

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos.

performance of these approaches highly rely on the accuracy of the model [12]. These concepts advise using model-free approaches when this information is not available for the training phase. Indeed, conversely to the first methods, model-free RL requires more interactions with the external environment and bases its functionality mainly on the environment changes and feedbacks, without the need to deeply understand its functioning [11]. Therefore, these methodologies do not require an estimation of the Markov Decision Process (MDP) model, and the value or policy function can be evaluated directly by sampling in order to approximate the task solution [10]. Although these features can limit the applicability of these strategies in some real applications, they can be used in all cases where there is no *a priori* information useful for the training phase and, therefore, can be exploited to address the more challenging case of a completely unknown environment. Moreover, recent developments and remarkable achievements in image processing [13], face recognition [14] and natural language processing [15] fields suggested to integrate the Deep Learning theory into the RL framework, leading to the concept of Deep Reinforcement Learning (DRL), which leverages the ability of Deep Neural Networks to serve as universal function approximators to achieve improved control performance [16], [17]. Thanks to DRL, it is possible to deploy RL-based control systems for all those applications where continuous or high-dimensional state and action spaces make traditional RL strategies, such as Q-learning, impractical or insufficient. In particular, *Deep Deterministic Policy Gradient* (DDPG, [17]) is currently one of the most common approaches in this field of research. RL and DRL have been successfully applied to various control engineering problems, ranging from autonomous vehicles [18]–[21], to energy and electrical systems [22]–[25], robotics [26], [27], IoT security [28], [29] and maritime applications [30], [31].

Surprisingly, despite their significant potential, only few recent works propose the use of RL techniques as a control strategy for tackling different air vehicles problems, the most representative being perhaps [32] and [33], where the authors exploit a DDPG approach to design the inner-loop controller providing attitude control for a quadrotor, the autopilot of an Unmanned Combat Aerial Vehicle, respectively, or more recently [34] where a RL-based missile path-planning algorithm is proposed for head-on interception. In addition, in [12] a DDPG approach is exploited to tune the control gains of a typical fixed-structure three-loop autopilot [7], with the aim of optimizing the missile autopilot performance.

In this perspective, the objective of this work is to investigate the possibility of successfully exploiting high-performance learning tools for the design of data-driven missile autopilot control in a model-free fashion. To this aim, a policy gradient model-free RL approach, specifically the DDPG strategy, is adopted to stabilize the longitudinal dynamics of a missile and to satisfy some performance requirements through the choice of a suitable reward function. DDPG is a relatively simple Policy Gradient (PG) actor-critic algorithm based on the use of deep neural networks, which has been chosen for the purposes of this work due to its sample efficiency and the small number of hyper-parameters involved, which makes the tuning procedure more straightforward when compared to more sophisticated RL techniques. Indeed, deep RL algorithms usually have a quite large number of free parameters (the structure of the neural networks, the learning rates, the soft update policy in case of twin neural networks, as in TD3, and so on) whose effect on the final result is not always obvious or immediately interpretable. In recent years, several DRL algorithms have been proposed in the literature, some of which can improve the characteristics of the agent's training with respect to the DDPG algorithm exploited here; indeed, DDPG is sometimes prone to training instability issues (mostly because it does not implement any explicit bound on the gradient ascent stepsize).

For the sake of completeness and to better motivate our work, despite our focus is on DDPG, in the following discussion we will try to give to the reader an overview of other comparable DRL methods, while further details of the DDPG algorithm are instead given in Section IV.

In general, PG RL algorithms aim at exploiting some form of gradient ascent to optimize the policy so as to maximize some given objective function, based on the reward obtained at each time step. However, the gradient method does not prescribe a way to choose a safe step-size in the optimization procedure. For this reason, the Trust Region Policy Optimization (TRPO) algorithm was proposed in [35], which proposes to limit the Kullack-Leibler divergence between the old and updated policies in order to limit the gradient steps amplitude. Proximal Policy Optimization (PPO) [36] is a revised version of TRPO, which exploits a clipping mechanism in order to obtain a Trust Region-like optimization algorithm which is compatible with the classical Stochastic Gradient Descent. It is worth to remark that both PPO and TRPO implicitly call for stochastic policies.

On the other hand, RL research moved along a parallel path in order to increase the *sample efficiency* of the training algorithms for agents which employ neural networks (especially in the actor-critic framework). The simplest algorithm belonging to this class of techniques is DDPG, which contains ideas that stem from the Deep Q-Network algorithm, but that is naturally suited for continuous actions spaces, and which exploits a replay buffer technique. In some implementations target networks are also used to improve the algorithm's stability.

Modifications to DDPG have been proposed in the technical literature to improve some aspects of the agent's training procedure; in [37] the TD3 algorithm was proposed, which adds some devices to avoid overestimation and reduce variance, providing better stability properties in some application cases, while a maximum entropy version of DDPG/TD3 named Soft Actor-Critic (SAC) has been introduced in [38].

In this view, our final choice fell on the DDPG algorithm, which tends to be more sample efficient than PPO on one hand, and to have less tuning parameters than more sophisticated techniques such as TD3 or SAC on the other. Thus, in this paper, we first recast the missile autopilot design control problem into the RL framework, with the primary aim of testing this approach in terms of control performance (settling time, undershoot, etc.) and then we compare the fully data-driven DDPG controller against classical model-based control strategies (such as H_∞ [6] and Model Reference Adaptive Control [39]).

Most notably, despite the nonlinearities that affect the process under exam, it was found that, when applying the DDPG approach to the autopilot problem, a deep knowledge of the plant model is not required and a linear model can be effectively used during the training procedure, to reduce the required computational burden, without degrading the performance of the *real* closed-loop system, at least close to the considered equilibrium point. Along this line, the agent obtained through the proposed method is then validated on the 2-Degrees of Freedom (2-DoF) fully nonlinear model.

The analysis further discloses how the careful study and definition of the reward function allows to easily shape the performance in the transient behavior, for example by decreasing the undershoot phenomena. In addition, comparison results in a realistic flight scenario confirm that the excellent capabilities of the proposed RL approach in capturing the underlying unknown nonlinear behaviors allow providing satisfactory closed-loop performances, which are comparable to those of *state-of-the-art* model-based techniques, without the need for running a detailed model of the process in real-time or for having a detailed a priori knowledge of the nonlinear dynamics. In addition, simulations at different Mach numbers and with random variations in the aerodynamic coefficients employing a Monte Carlo approach are performed in order to provide some meaningful insight on the robustness of the closed-loop.

It is finally worth noting that the need for pioneering solutions to respond to unmet challenges as well as to new opportunities derived from the application of AI techniques to this research field is confirmed by the autopilot system very recently designed in [40] leveraging a modified TRPO agent trained on a detailed nonlinear model of the plant dynamics. In particular, such system exploits a transformed acceleration signal as the controlled variable to overcome the inherent non-minimum phase characteristics of the missile dynamic. This approach does not allow the authors to take into account, during the training of the RL agent, the typical undershoot that characterizes the transient response of a missile to a step request in the acceleration. As opposed to [40], the present work instead tries to investigate the capability of a purely data-driven missile autopilot by explicitly considering the main performance indexes (settling-time, undershoot, steady-state error, etc.) in the DDPG reward function.

The rest of the paper is organized as follows. Sections II and III describe the control requirements and the missile nonlinear 2-DoF model, respectively, while in Section IV a brief introduction to the DDPG algorithm is provided. The details of the proposed RL approach, in terms of agent structure, reward function engineering and training procedure, are described in Section V while, simulation results are discussed in Section VI, where the performance of the proposed RL agent is compared to those of a self-scheduled $\mathcal{H}_\infty$ autopilot [6] and of the Augmented Adaptive Controller presented in [39]. Eventually some conclusive remarks are given in Section VII.

II. PROBLEM STATEMENT

This section defines the control requirements that will be taken into account in the design of the proposed autopilot based on a RL control approach.

During the flight, the longitudinal dynamics of a missile can be unstable, depending on the relative location between the center of pressure and the center of mass, i.e. the center of pressure is the point where the lifting force is considered to act, as shown in Fig. 1. In order to stabilize and to control the longitudinal dynamic of the missile a tail fin is introduced. It follows that the controller must generate the required tail deflection to produce the desired normal acceleration, while stabilizing the airframe rotational motion. Moreover, the transient response of the missile to a step request in the normal acceleration is characterized by an initial undershoot, which is reflected by the fact that the associated linearized model is a non-minimum phase one [39]. Ideally, this undershoot should be kept as small as possible; however, as it will be shown in what follows, this results in a slower response, hence a trade-off between the bandwidth of the closed-loop system and the maximum undershoot must be sought.

Based on the previous observations, the following qualitative requirements are considered in Section V-B to design the reward function of the proposed DDPG approach, in order to ensure performance that are similar to those of other solutions available in literature ([6], [39], [41]):

1) the control system shall ensure the stability of the closed-loop system over the largest possible operating range, defined in terms of angle of attack $\alpha(t)$ and the Mach number M; it should be noticed that a wider range in terms of $\alpha(t)$ is preferable since typical applications foresee the scheduling of different controllers as a function of M (see [41] as an example);
2) the control system shall take into account the maximum deflection that can be applied to the tail;
3) in tracking a step command in the normal acceleration, the control system shall minimize the following quantities:
   a) the rising time at the 90% of the final value;
   b) the overshoot;
   c) the undershoot;
   d) the steady-state error.

III. LONGITUDINAL MISSILE DYNAMIC MODEL

In order to simulate the missile dynamics and to prove the effectiveness of the proposed autopilot system, the following simplified 2-DoF nonlinear model proposed in

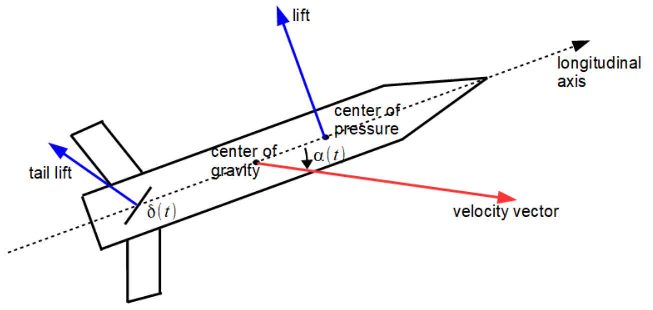


FIGURE 1. Simplified scheme of the considered missile: velocity vector is depicted in red while the lift forces in blue.

literature [39], [41] is considered, which is capable of describing the longitudinal dynamics of a tailed controlled missile (see Fig. 1) under the following assumption.

Assumption 1 (Fully Decoupled Dynamics): It is assumed that the pitch, yaw and roll channels are decoupled, so coupling phenomena are ignored. ■

Given Assumption 1, the longitudinal dynamic of a missile can be described as follows:

$$\dot{\alpha}(t) = K_\alpha M C_n\left(\alpha(t), \delta(t), M\right)\cos(\alpha(t)) + q(t), \tag{1a}$$
$$\dot{q}(t) = K_q M^2 C_m\left(\alpha(t), \delta(t), M\right), \tag{1b}$$
$$\dot{\delta}(t) = \delta_v(t), \tag{1c}$$
$$\dot{\delta}_v(t) = -\omega_a^2\delta(t) - 2\zeta\omega_a\delta_v(t) + \omega_a^2\delta_c(t), \tag{1d}$$
$$\eta(t) = K_z M^2 C_n\left(\alpha(t), \delta(t), M\right), \tag{1e}$$

being $\alpha(t)$ [rad] the angle of attack, $q(t)$ [rad/s] the pitch rotational rate, M the Mach number, $\delta_c(t)$ [rad] the tail fin deflection command, $\delta(t)$ [rad] the tail fin deflection, $\delta_v(t)$ [rad/s] the tail fin angular speed and $\eta(t)$ [lb/slugs] the normal acceleration.

Equations (1c) and (1d) define a second-order linear model of the actuator that links the tail fin deflection command $\delta_c(t)$ to the actual deflection $\delta(t)$, where ζ is the actuator damping ratio and ω_a is its natural frequency. Part of the system's nonlinearity lies in the definition of the aerodynamic coefficients for both the normal force and the pitch momentum, respectively C_n and C_m, which are given by:

$$C_n(\alpha, \delta, M) = a_n\alpha^3 + b_n\alpha|\alpha| + c_n\left(2 - \frac{1}{3}M\right) \times \alpha + d_n\delta, \tag{2a}$$
$$C_m(\alpha, \delta, M) = a_m\alpha^3 + b_m\alpha|\alpha| + c_m\left(-7 + \frac{8}{3}M\right) \times \alpha + d_m\delta, \tag{2b}$$

where time dependency is dropped to simplify the notation. Table 1 shows the values of the model parameters.

IV. DEEP DETERMINISTIC POLICY GRADIENT

In the RL approach, an agent must learn to interact with an unknown environment in a way that maximizes the expected cumulative value of a given reward function. Usually, the environment is modeled as a Partially Observable MPD

TABLE 1. Parameters of the missile nonlinear model (1)**.**

Parameter	Value
Static pressure at $20,000\,ft$, P_0	$973.3\ \frac{lbs}{slugs}$
Surface area, S	$0.44\,ft^2$
Mass, m	$13.98\,slugs$
Speed of sound at $20,000\,ft$, v_s	$1036.4\ \frac{ft}{s}$
Diameter, d	$0.75\,ft$
Pitch moment of inertia, I_y	$182.5\,slug\,ft^2$
Drag coefficient, C_a	-0.3
Actuator damping ratio, ζ	0.7
Actuator natural frequency, w_a	$150\ \frac{rad}{s}$
Coefficient of aerodynamic model, a_n	0.000103
Coefficient of aerodynamic model, b_n	-0.00945
Coefficient of aerodynamic model, c_n	-0.1696
Coefficient of aerodynamic model, d_n	-0.034
Coefficient of aerodynamic model, a_m	0.000215
Coefficient of aerodynamic model, b_m	-0.0195
Coefficient of aerodynamic model, c_m	0.051
Coefficient of aerodynamic model, d_m	-0.206
Coefficient of aerodynamic model, K_α	$0.7\frac{P_0S}{mv_s}$
Coefficient of aerodynamic model, K_q	$0.7\frac{P_0Sd}{I_y}$
Coefficient of aerodynamic model, K_z	$0.7\frac{P_0S}{m}$
Coefficient of aerodynamic model, A_x	$0.7\frac{P_0SC_a}{m}$

(PO-MDP); in particular, at each time instant, the agent receives from the environment an observation and must pick an action a_t based on such observation. In principle, the observation may differ from the system's actual state. However, for simplicity, we confuse the state and the observation since s_t is the state of the agent's internal representation of the environment, modelled as a PO-MDP. The computed action affects the next state transition of the system, from s_t to s_{t+1}, after which the agent receives a reward r_{t+1} and a new observation. The objective of the training process is to find a *policy* that maximizes the *cumulative reward*, defined as

$$R_t = \sum_{k=0}^{N} \gamma^k r_{t+k+1}, \quad \gamma \in [0, 1), \tag{3}$$

where the *discount factor* γ is generally close to (but less than) 1. Moreover, the reward is computed over several *episodes*, each consisting of (up to) N steps.

In classical RL tabular methods, discrete action and observation spaces are considered. The name tabular reflects the fact that, in such methods, the agent usually stores a table that associates to each state-action pair the value of the expected cumulative reward R_t (represented by the *action-value function* $Q(s_t, a_t)$). If the agent had access to the true value of Q for each action-state couple, the optimal policy would be the choice of a that maximizes Q for each state s (*greedy* policy). As a consequence, the objective of the training boils down to finding a table that accurately represents $Q(s, a)$.

Tabular methods, however, are limited in working only with discrete action and observation spaces, being inefficient in the presence of continuous and high dimensional spaces. To overcome this limitation, several extensions have been proposed in the technical literature, mainly exploiting deep neural networks and their capability of serving as universal

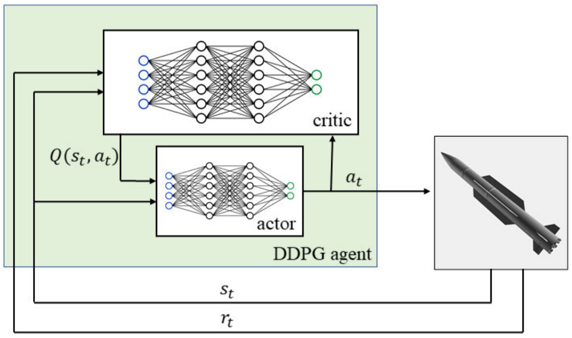


FIGURE 2. Basic scheme of a DDPG agent showing the interaction between the actor, the critic (represented as deep neural networks) and the environment (represented as a missile). According to the DDPG technique, the environment provides the state s_t and the reward r_t to the nets, the actor provides the action a_t to the critic and the environment while the critic evaluates the action-value function $Q(s_t, a_t)$, providing it to the actor.

function approximators. The combination of Deep Learning techniques with Reinforcement Learning algorithms is usually referred to as Deep Reinforcement Learning. In particular, in *actor-critic* methods, the RL problem is separated into two subproblems:

- **critic**: finds a good approximation of the action-value function $Q(s, a)$ (where s and a may assume continuous values);
- **actor**: exploits the critic to improve the policy, represented with another approximator $\mu(s)$.

In this study, an actor-critic method known as DDPG algorithm, originally proposed in [17], is considered. DDPG is a model-free, off-policy approach that extends the DPG [16] with the exploitation of deep neural networks. A simple representation of the DDPG paradigm is shown in Fig. 2. In DDPG, an actor network $\mu(s|\theta^\mu)$ is used to represent the current policy, while a critic network $Q(s, a|\theta^Q)$ is used to approximate the action-value function $Q(s, a)$ (θ^μ and θ^Q indicate the corresponding network's parameters). In particular, the critic network is trained so as to minimize the following loss function

$$L(\theta^Q) = \mathbb{E}\Big[(Q(s_t, a_t|\theta^Q) - y_t)^2\Big] \approx \frac{1}{N}\sum_i (Q(s_i, a_i|\theta^Q) - y_i)^2, \tag{4}$$

being $y_t = r_{t+1} + \gamma Q(s_{t+1}, \mu(s_{t+1})|\theta^Q)$ (or just $y_t = r_{t+1}$, if the next state is terminal). The average is usually computed across a *mini-batch*, randomly extracted from a replay buffer $\mathcal{B}$ containing a (large) collection of the past transitions $(s_t, a_t, r_{t+1}, s_{t+1})$. The actor weights are updated in the direction of the critic action-value gradient, according to the chain rule applied to the expected return J w.r.t. the actor parameters

$$\nabla_{\theta^\mu} J = \mathbb{E}\Big[\nabla_a Q(s, a|\theta^Q)_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s=s_t}\Big] \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s=s_i}\,. \tag{5}$$

Since the Q update is prone to divergence, target networks are employed in order to improve the learning stability. A copy of the critic and actor networks, indicated as $Q'(s, a|\theta^{Q'})$ and $\mu'(s|\theta^{\mu'})$ respectively, are used in order to evaluate the target values [17]. The critic and actor parameters, i.e. θ^Q and θ^μ are updated according to (4), while the target networks are updated either periodically or in a *soft* fashion, i.e. $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$, with $\tau \ll 1$. Finally, to find a balance between the exploration of the state-action space and the exploitation of the current policy, a noise sampled process $\mathcal{N}$ can be added to the actor policy $\mu'(s_t) = \mu(s_t|\theta^\mu) + \mathcal{N}$, where $\mathcal{N}$ is an Ornstein-Uhlenbeck process [17]. The DDPG algorithm steps are listed in Algorithm 1 (see also [17]).

V. RL CONTROL SYSTEM FOR MISSILE

In this section, the proposed DDPG control algorithm is introduced, focusing on the control system architecture, neural networks and details concerning the reward function proposed for the training phase.

A. CONTROLLER ARCHITECTURE

Starting from the state variables of the 2-DoF nonlinear missile model in (1), the observations vector for the agent training has been chosen as:

$$Obs(t) = \begin{bmatrix}\alpha(t) & q(t) & \delta(t) & \eta_{ref}(t)\end{bmatrix}^T$$

where $\eta_{ref}(t)$ is the acceleration reference value, generated by the guidance and navigation system of the missile. It is worth to remark that, despite the actuator angular speed $\delta_v(t)$ is a state variable, it was not included among the observations due to the technological difficulties in obtaining a reliable measurement of this quantity. The only control action considered is the missile's tail fin deflection request $\delta_c(t)$, and the control sampling time has been set equal to 0.01 s.

The structures of the neural networks (see Fig. 2) have been defined through a trial-and-error procedure in terms of the number of hidden layers and neurons, activation functions, etc., considering a trade-off between performance and limited computational capacity available on board. The main results of the analysis that was carried out are summarized in what follows.

The architecture of the critic neural network is shown in Table 2. This neural network is characterized by 5 input variables, i.e. observation and the action variables, and a single output variable, representing the critic's estimate of the action-value function. Note that, all input variables were normalized so as to take values in the range [0, 1]. Five fully connected layers connect inputs and outputs, each of them characterized by 100 neurons and REctified Linear Units (RELU) activation function. In particular, the fully connected layers 1 and 2 process in sequence the observation variables, while the fully connected layer 3 processes the action variable. Then, the outputs of layers 2 and 3 are summed before passing through the fully connected layers 4 and 5.

Algorithm 1 Deep Deterministic Policy Gradient (DDPG)

```
1. Randomly initialize critic and actor networks $Q(s, a|\theta^Q)$ and $\mu(s, a|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$;
2. Initialize the target networks $Q'$ and $\mu'$ with weights
       $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$;
3. Initialize the replay buffer $\mathcal{B}$;
for $episode = 1, M$ do
    4. Initialize the random process noise $\mathcal{N}$ for the action exploration;
    5. Receive initial observation state $s_1$;
    for $t = 1, T$ do
        6. Select an action $a_t = \mu(s_t|\theta^\mu)+\mathcal{N}_t$ according to the current policy and the exploration noise;
        7. Execute the action $a_t$ and take reward $r_t$ and the new state $s_{t+1}$;
        8. Store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{B}$;
        9. Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $\mathcal{B}$;
        10. Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})\,\theta^{Q'})$;
        11. Update the critic minimizing the loss function by using equation (4)
        12. Update the actor policy using the sampled policy gradient in equation (5)
        13. Update the target networks:
               $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$;
               $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$.
    end for
end for
```

TABLE 2. Architecture and parameters of the critic NN.

Layer	Parameter	Value
input layer	input size	5
fully connected layer 1	number of neurons state activation function	100 RELU
fully connected layer 2	number of neurons state activation function	100 RELU
fully connected layer 3	number of neurons state activation function	100 RELU
fully connected layer 4	number of neurons state activation function	100 RELU
fully connected layer 5	number of neurons state activation function	100 RELU
output layer	output size	1

TABLE 3. Architecture and parameters of the actor NN.

Layer	Parameter	Value
input layer	input size	4
fully connected layer 1	number of neurons state activation function	100 RELU
fully connected layer 2	number of neurons state activation function	100 RELU
fully connected layer 3	number of neurons state activation function	100 RELU
fully connected layer 4	number of neurons state activation function	100 tanh
output layer	output size	1

The architecture of the actor's neural network is shown in Table 3. This network has 4 input variables, i.e. the observation variables. Also in this case, the input variables have been normalized to take values in the range [0, 1]. The only output of the network is the control action. Between the input and output layers there are four fully connected layers, each containing 100 neurons. The first three layers have RELU activation function, while the last has a hyperbolic tangent (tanh) activation function, which produces an output in the range $[-1, 1]$. The output of this layer is then scaled taking into account the maximum allowed actuator deflection $\bar{\delta}_c$.

B. REWARD ENGINEERING

Once the structure of the agent has been set, a reward function must be defined, taking into account the requirements discussed in Section II. To attain the desired goals, the following reward function has been used:

$$r(t) = -\omega_1(t)P_{fail}+(1-\omega_1(t))\Big[P_{step}-K_1e^2(t) - K_2q^2(t) - K_3\dot{\delta}(t)^2 - \omega_2(t)K_4e^2(t) - (1-\omega_3(t))\,K_5\delta^2(t) + \omega_3(t)P_{win}\Big], \tag{6}$$

where P_{fail}, P_{step}, P_{win} and K_i for $i = 1, \ldots, 5$ are positive constants, $\omega_i(t)$ for $i = 1, 2, 3$ are Boolean variables and $e(t) = \eta_{ref} - \eta(t)$ is the tracking error. In particular, once a range for the lateral acceleration has been fixed, P_{fail} defines a penalty which is applied when $\eta(t)$ exceeds the prescribed bounds, causing the premature termination of the current episode; otherwise, the bonus P_{step} is applied. Finally, an additional bonus P_{win} is given to the agent when the norm of the tracking error is less than a given threshold. The Boolean variables $\omega_i(t)$, $i = 1, 2, 3$ allow to apply the penalty and bonus defined previously. In particular, $\omega_1(t)$ is true if η is inside the desired range, $\omega_2(t)$ takes the value `true` if the step response shows an undershoot, and $\omega_3(t)$ is true if the tracking error is less than a given threshold.

It can be seen how the control policy is rewarded by function (6) when the missile acceleration is steered and kept close to the reference, i.e. requirements 3a and 3d are verified, while it is penalized when the missile motion exceeds a prescribed range of lateral acceleration values. The quadratic terms in the missile angular velocity and actuator deflection and deflection speed are used to take into account the requirements about the overshoot and the undershoot, and to limit the control power. Indeed, due to the non-minimum phase behavior of the linearized plant, a further error penalty is

TABLE 4. Parameters of the training problem.

Parameter	Value
Agent sampling time	$0.01\,s$
Maximum permissible steps	150
Maximum permissible episodes	15000
Size of experience buffer	10^6
Size of mini-batch samples	256
Actor learning rate	0.001
Critic learning rate	0.001
Initial variance of exploration noise	0.015
Variance decay rate	10^{-6}
Gradient upper bound	1
Discounting factor	0.99

considered to limit the undershoot. Since some requirements conflict with each other, e.g. rising time and overshoot, the positive constants K_i, $i = 1, \dots, 5$ weight the terms in the reward function so that the resulting control policy will be a trade-off solution.

C. TRAINING PROCEDURE

According to the reward function (6), a training procedure has been performed on the missile linearized model around the equilibrium point defined by $M = 3$ and $\alpha(t) = 0$ [deg], where each episode consists in a simulation of random maneuver. In this way, the policy has been optimized with respect to a single flight condition; then, the controller has been validated in all the considered operating range, in order to verify requirement 1.

More specifically, each training episode is characterized by a different step command, whose amplitude is chosen randomly within the range $[-1, 1]$ [g], and terminates when the simulation time reaches its maximum value, chosen as $T_{\max} = 1.5$ [s], or when the lateral acceleration exceeds the desired range of values. Table 4 contains the values of the training parameters.

VI. SIMULATION RESULTS

In this section, the effectiveness of the proposed controller is characterized through numerical simulations.

The DDPG agent has been trained by implementing the procedure defined in Section V-C. In particular, the constants in the reward function (6) have been chosen equal to

$$K_1 = 1, \quad K_2 = 0.2,\ K_3 = 0.002,\ K_4 = 5,\ K_5 = 25,$$
$$P_{fail} = 150, \quad P_{step} = 3,\ P_{win} = 25,$$

in order to obtain a maximum undershoot less than the 50% of the reference value, a maximum rising time less than 1 $[s]$ and a maximum steady-state error less than the 5% of the reference value.

Section VI-A shows how the proposed data-driven approach is capable *to learn* the nonlinear behaviour of the missile described by (1) from the limited *experience* that it gets from the response of the linearized model for specific flight conditions. Moreover, we evaluate the robustness of the control system for different flight conditions and in presence of uncertainty in the aerodynamic coefficients. A further assessment is carried out in Section V-B, by comparing the DDPG trained agent with two robust model-based strategies, i.e. the self-scheduled H_∞ control and the Adaptive Augmenting Controller (ACC), previously presented in literature in [6] and [39], respectively. This comparison shows the efficiency of the proposed model-free autopilot in guaranteeing proper closed-loop tracking performances and exhibiting, at the same time, a lower undershoot, according to the proposed engineering reward function (6). The comparison among these three controllers is carried out by emulating the effects of the guidance law in the outer-loop, whose aim is to provide the proper missile acceleration [42], as a time-varying reference signal, which proves the capability of the proposed approach to work in more complex scenarios as realistic missile guide systems.

A. CONTROLLER VALIDATION

In this section the closed-loop responses of the linearized and nonlinear models are compared to validate the trained DDPG agent. A maneuver starting from the flight condition considered in the training phase and with three different acceleration requests is considered (see the black trace in Fig. 3). The simulation results reported in Fig. 3 show that the closed-loop responses to the first request of one additional g are similar, hence we can consider the control requirements satisfied in both cases. Furthermore, this simulation also shows that, when the requested acceleration brings the system far from the reference equilibrium for the linearized model, the DDPG agent still exhibits acceptable performance, hence proving its capabilities in *learning* a control law that generalizes the nonlinear behavior, although the training procedure was based on a linear approximation of the missile response.

Moreover, the robustness of the proposed approach has been evaluated considering 820 different nonlinear simulations performed for a step command of magnitude $\eta_{ref} = 1$ [g], with different initial angles of attack $\alpha(0) \in [-10\,, 10]$ [deg], and with different Mach number $M \in [2\,, 4]$. The control performance have been evaluated changing the cumulative reward at the end of each simulation. Results in Fig. 4 show that the reward is *almost* independent on the initial value of $\alpha(t)$, while the impact of M is more evident. However, this degradation can be avoided by considering the Mach number as a scheduling parameter, as mentioned in Section II. It is worth to observe that the narrow red band for $M = 3 \div 3.2$ depends on the fact that the regime value differs from the desired value of less than the 5%, i.e., $\omega_3(t)$ is true in (6). Fig. 5 shows a comparison among the closed-loop nonlinear responses according to its variation, maintaining the initial angle of attack $\alpha(0) = 0$ [deg]. Specifically, it can observe how all control requirements are verified for $M = 3.1$, for which the maximum cumulative reward is attained, while a decrease of M causes an increase of the rise time and a less percentage value of the undershoot phenomena, with a converse behavior when M increases.

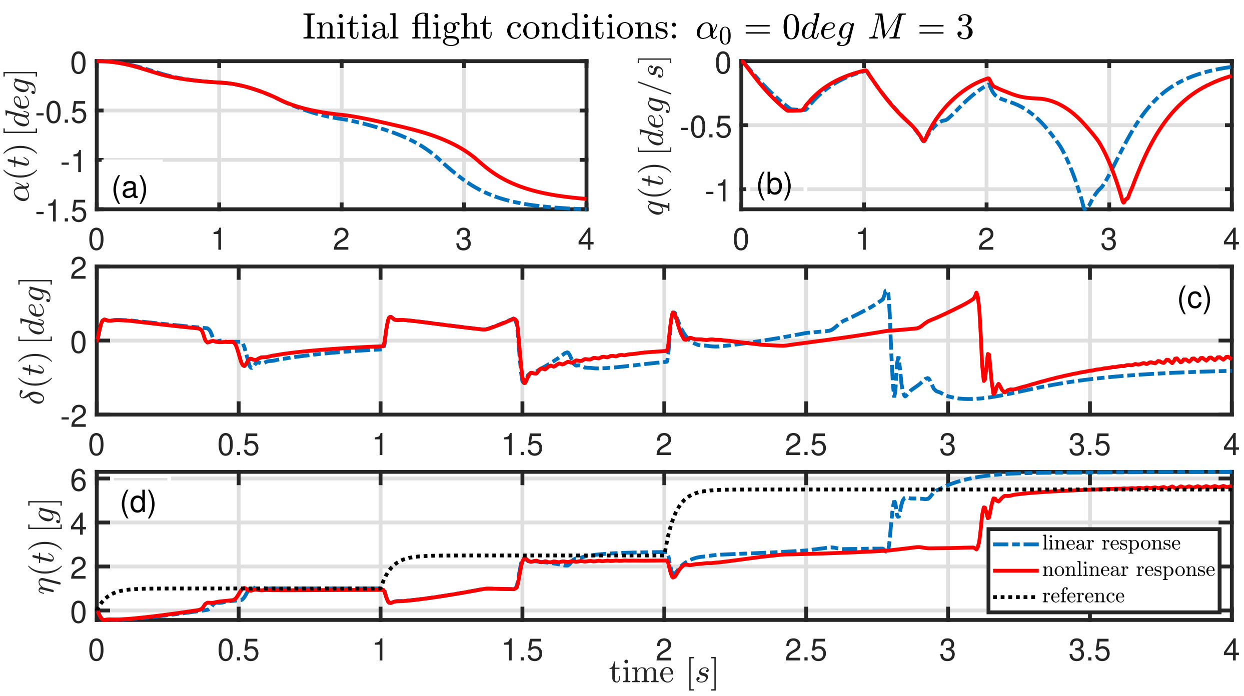


FIGURE 3. **Comparison between the closed-loop responses of the linearized (blue) and nonlinear (red) models. Time traces of: (a) angle of attack; (b) pitch rotational rate; (c) tail fin deflection; (d) normal acceleration.**

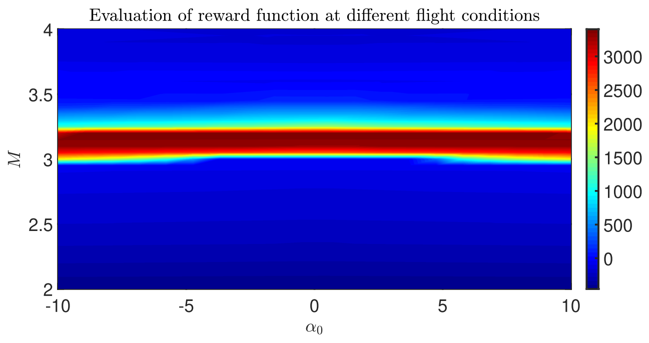


FIGURE 4. **The figure show the variation of the cumulative reward R_t, as defined in (6), over the considered flight envelope. In particular, cumulative reward has been evaluated for the same maneuver starting from different initial attack angles α_0 and at different Mach numbers M.**

Furthermore, the robustness of the proposed approach has been evaluated through Monte Carlo simulations performed with the nonlinear model, that start at the initial flight condition $\alpha_0 = 0$ [deg], $M = 3$, and where an additive uncertainty on aerodynamic coefficients ΔC_n and ΔC_m has been introduced in the range $[-20, 20]$ % of the corresponding nominal value.

Fig. 6 shows the results for 100 runs when a step command of magnitude $\eta_{ref} = 1$ [g] is applied. Despite this significant variation in the model parameters, the approach still guarantees the closed-loop stability in all the perturbed scenarios. As for the case of variations in the Mach number, robustness against model uncertainty was not considered during the training phase. Therefore, as expected, some slight performance degradation can be observed. However, obviously this degradation can be counteracted by including also robustness as a further objective of the training phase.

B. COMPARISON WITH MODEL-BASED METHODOLOGIES

To better discuss the advantages of the proposed DDPG strategy in tracking the missile lateral acceleration, we now compare its closed-loop behaviour with two different robust model-based strategies proposed in the literature to solve the same control problem. Specifically, the former has been presented in [6], where authors developed a robust self-scheduled H_∞ control to regulate the lateral acceleration of a missile, while the latter consists of an adaptive control mechanism named Adaptive Augmenting Controller (AAC) [39].

The design procedure for the self-scheduled $\mathcal{H}_\infty$ controller relies on a Linear Parameter-Varying (LPV) model of the missile, whose state space representation depends on both α and M. In this case, robustness is achieved by guaranteeing $\mathcal{H}_\infty$ performance for the LPV plant, when α ranges in the interval $[-20, 20]$ [deg], while $M \in [2, 4]$.

Similarly, the AAC achieves robustness by designing a baseline state-feedback that guarantees robust stability for all the models belonging to the convex hull defined by the linearized models with $M = 3$ and α equal to $\{0, 5, 10, 15, 20\}$ $[deg]$. Moreover, for both the model-based controller, the gains have been tuned so as to obtain a similar undershoot when performing a $1\,[g]$ maneuver.

The simulation results are shown in Figs. 7 and 8, where the closed-loop responses have been compared by performing

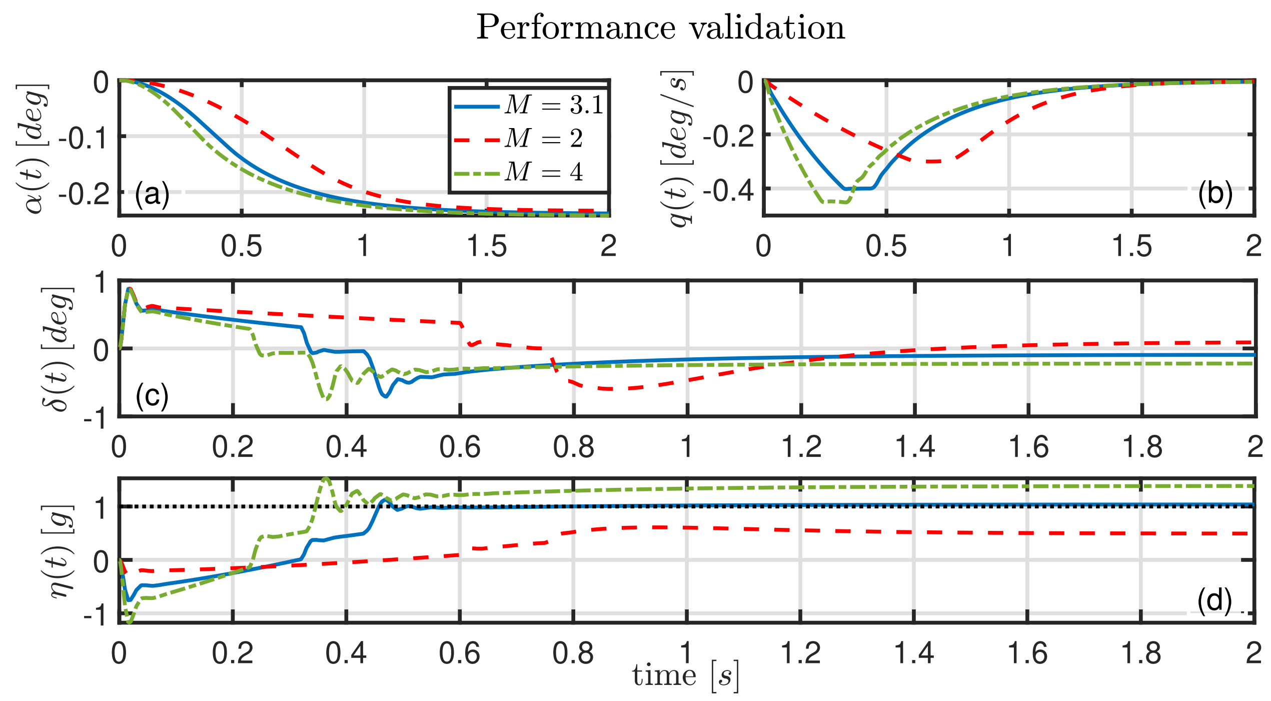


FIGURE 5. Controller performance at different Mach numbers for the same 1 [*g*] step maneuver. Time traces of: (a) angle of attack; (b) pitch rotational rate; (c) tail fin deflection; (d) normal acceleration.

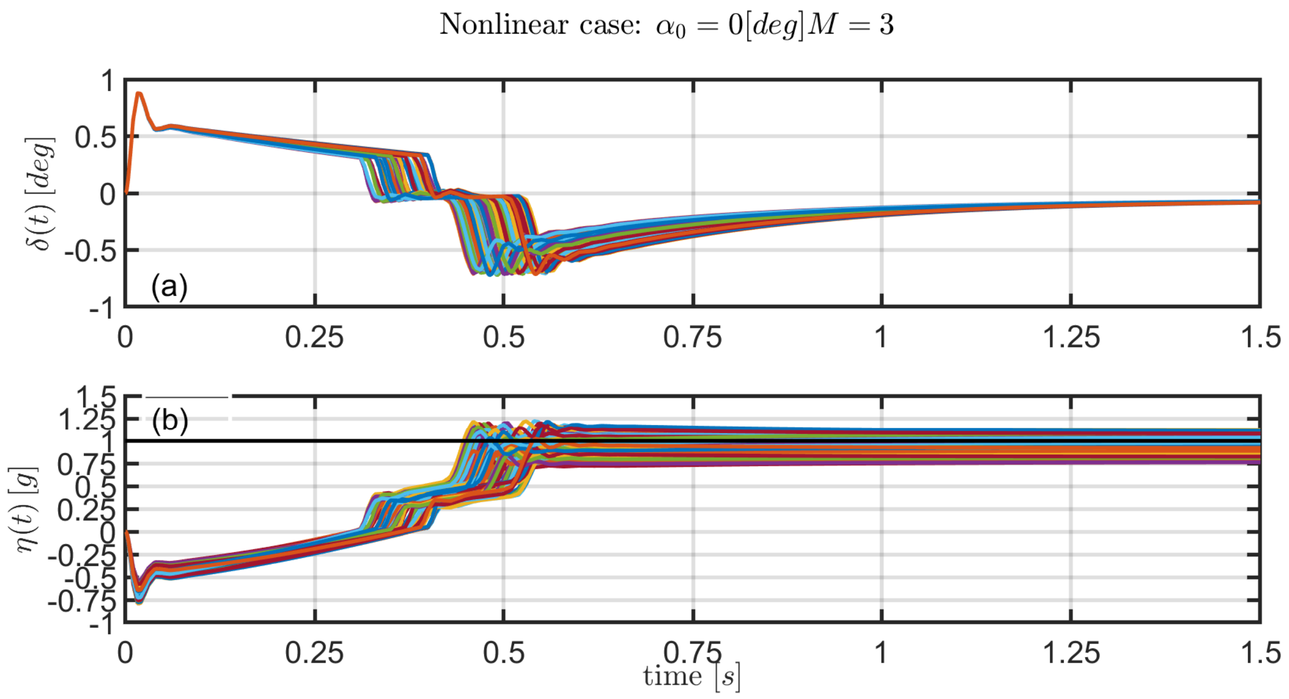


FIGURE 6. Monte Carlo robustness analysis for 100 random uncertainty realizations of the aerodynamic coefficients $C_n + \Delta C_n$ and $C_m + \Delta C_m$, when a 1 [*g*] step maneuver is applied. Both ΔC_n and ΔC_n are allowed to vary in the range [−20%, 20%] of the nominal value of the respective coefficients. Time traces of: (a) tail fin deflection; (b) normal acceleration.

two different maneuvers. For the comparison in Fig. 7 we have considered a sequence of three step commands. When tracking the first 1 [g] step reference, all the controllers show the same undershoot, while the response of the RL agent is characterized by a slightly shorter settling time. When a reference changes larger than 1 [g] is requested, the response of the data-driven controller is always characterized by the smallest undershoot and the smallest control effort when

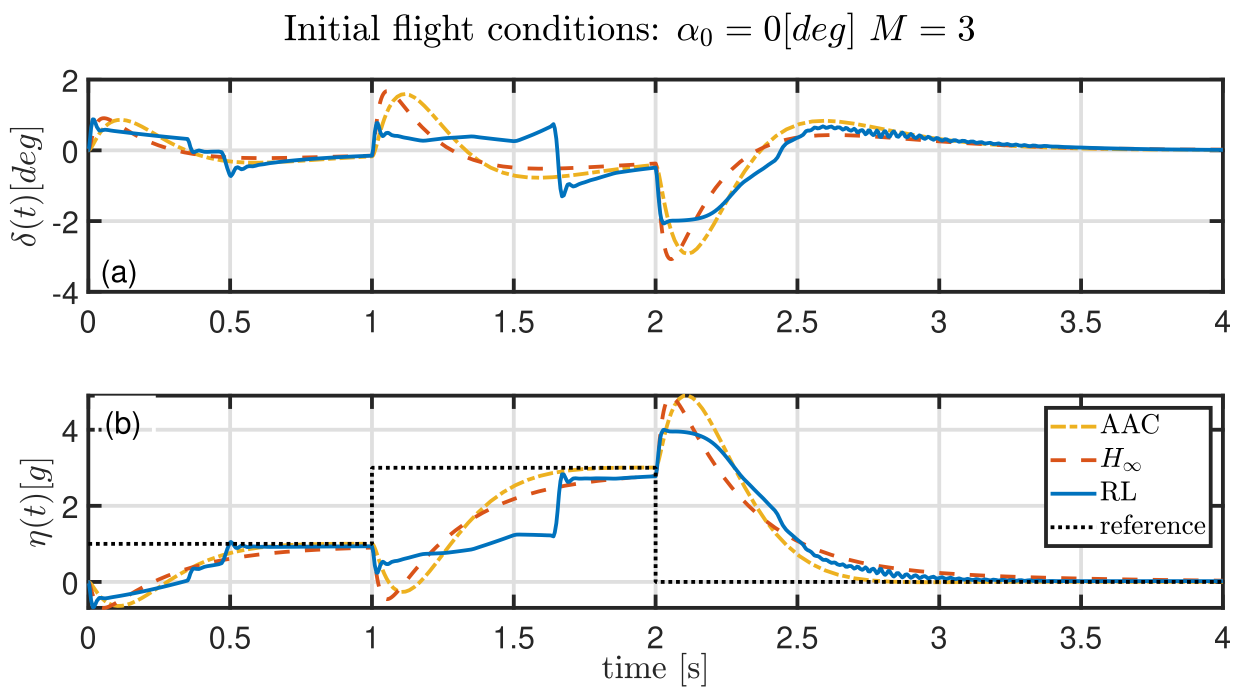


FIGURE 7. Performance comparison among the proposed data driven controller and two model based controllers in tracking a reference signal which model a sequence of step maneuvers of different magnitudes. Time traces of: (a) tail fin deflection; (b) normal acceleration.

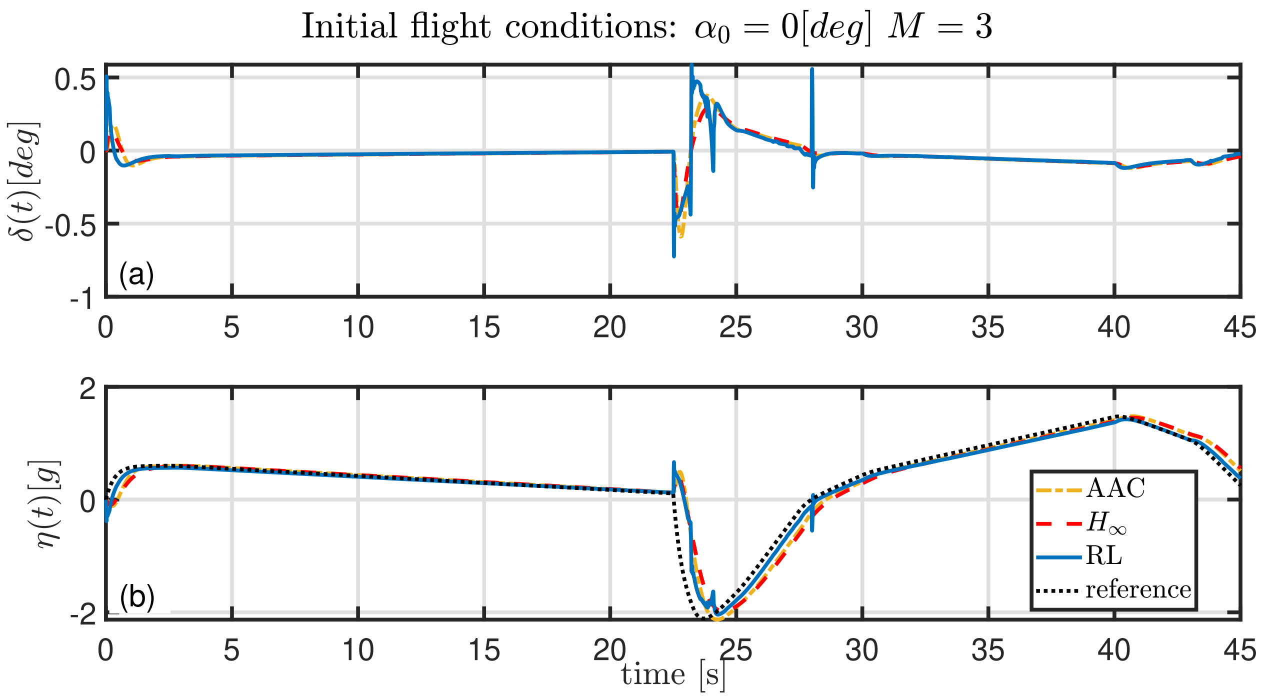


FIGURE 8. Performance comparison among the proposed data driven controller and two model based controller in tracking a reference signal which emulates the effects of a guidance system for hitting a moving target. Time traces of: (a) tail fin deflection; (b) normal acceleration.

compared with the two model-based controllers. Moreover, in the worst case, the settling time for the RL controller is similar to those of the other two considered approaches.

The further simulations shown in Fig. 8 refer to the response to a reference signal similar to the one computed by a guidance system, as proposed in [42]. Here we want to remark that, despite the RL controller was not trained using such a class of reference signals, it shows similar performance when compared to the two model-based controllers. From these results, it is possible to conclude that the proposed DDPG autopilot shows the same robustness against model uncertainties as to the two model-based approaches.

This result is achieved without the need for a detailed system model, as required by both the self-scheduled $\mathcal{H}_\infty$ controller and the AAC. Indeed the former required an LPV description of the missile response, and the latter more than one linearized model, while the proposed RL agent practically achieves the same performance exploiting just a single linear model. This further proves the effectiveness of model-free data-driven approaches for the design of robust autopilot systems.

VII. CONCLUSION

The feasibility of a model-free controller for the lateral acceleration of a missile has been investigated in this article. Specifically, exploiting the DDPG approach, a RL agent has been trained on the linearized dynamics of a 2-DoF nonlinear missile model, taking into account the main performance indexes. To assess the effectiveness of the proposed approach, different scenarios have been simulated on a 2-DoF nonlinear model, proving the efficiency of the data-driven approach in stabilizing the rotational dynamics, satisfying the control requirements in design flight conditions. Furthermore, a robustness analysis is provided to show the capability of the proposed approach in guaranteeing closed-loop stability in a wide range of flight conditions and in presence of model uncertainty. Along this line, future works will involve the improvement of the robustness w.r.t. variations of the Mach number, model uncertainties and measurement noise, by the explicit inclusion of robustness as a further objective during the training phase.

REFERENCES

[1] B. Panchal, K. Subramanian, and S. E. Talole, "Robust missile autopilot design using two time-scale separation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 3, pp. 1499–1510, Jun. 2018.

[2] K. Wise, E. Lavretsky, N. Hovakimyan, C. Cao, and J. Wang, "Verifiable adaptive control: UCAV and aerial refueling," in *Proc. AIAA Guid., Navigat. Control Conf. Exhib.*, Aug. 2008, p. 6658.

[3] D. Hwang and M.-J. Tahk, "Robustness improvement for a three-loop missile autopilot using discontinuous state feedback," *Int. J. Aeronaut. Space Sci.*, vol. 19, no. 3, pp. 661–674, Sep. 2018.

[4] G. Mattei and S. Monaco, "Nonlinear autopilot design for an asymmetric missile using robust backstepping control," *J. Guid. Control Dyn.*, vol. 37, no. 5, pp. 1462–1476, 2014.

[5] V. Bachtiar, C. Manzie, and E. C. Kerrigan, "Nonlinear model-predictive integrated missile control and its multiobjective tuning," *J. Guid., Control, Dyn.*, vol. 40, no. 11, pp. 2961–2970, Nov. 2017.

[6] P. Apkarian, P. Gahinet, and G. Becker, "Self-scheduled H_∞ control of linear parameter-varying systems: A design example," *Automatica*, vol. 31, no. 9, pp. 1251–1261, 1995.

[7] J.-H. Kim and I. H. Whang, "Augmented three-loop autopilot structure based on mixed-sensitivity H_∞ optimization," *J. Guid., Control, Dyn.*, vol. 41, no. 3, pp. 751–756, Mar. 2018.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[9] A. Alharin, T.-N. Doan, and M. Sartipi, "Reinforcement learning interpretation methods: A survey," *IEEE Access*, vol. 8, pp. 171058–171077, 2020.

[10] H. Sun, W. Zhang, R. Yu, and Y. Zhang, "Motion planning for mobile robots—Focusing on deep reinforcement learning: A systematic review," *IEEE Access*, vol. 9, pp. 69061–69081, 2021.

[11] T. Zhang, M. Xiao, Y. Zou, and J. Xiao, "Robotic constant-force grinding control with a press-and-release model and model-based reinforcement learning," *Int. J. Adv. Manuf. Technol.*, vol. 106, nos. 1–2, pp. 589–602, Jan. 2020.

[12] H.-S. Shin, S. He, and A. Tsourdos, "A domain-knowledge-aided deep reinforcement learning approach for flight control design," 2019, *arXiv:1908.06884*.

[13] T. Tao, K. Liu, L. Wang, and H. Wu, "Image recognition and analysis of intrauterine residues based on deep learning and semi-supervised learning," *IEEE Access*, vol. 8, pp. 162785–162799, 2020.

[14] M. Wang and W. Deng, "Deep face recognition: A survey," *Neurocomputing*, vol. 429, pp. 215–244, Mar. 2021.

[15] I. Lauriola, A. Lavelli, and F. Aiolli, "An introduction to deep learning in natural language processing: Models, techniques, and tools," *Neurocomputing*, vol. 470, pp. 443–456, Jan. 2022.

[16] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[18] Z. Ma, Q. Huo, T. Zhang, J. Hao, and W. Wang, "Deep deterministic policy gradient based energy management strategy for hybrid electric tracked vehicle with online updating mechanism," *IEEE Access*, vol. 9, pp. 7280–7292, 2021.

[19] Y. H. Xu, C. C. Yang, M. Hua, and W. Zhou, "Deep deterministic policy gradient (DDPG)-based resource allocation scheme for NOMA vehicular communications," *IEEE Access*, vol. 8, pp. 18797–18807, 2020.

[20] H. Lee, N. Kim, and S. W. Cha, "Model-based reinforcement learning for eco-driving control of electric vehicles," *IEEE Access*, vol. 8, pp. 202886–202896, 2020.

[21] Z. Wu, L. Sun, W. Zhan, C. Yang, and M. Tomizuka, "Efficient sampling-based maximum entropy inverse reinforcement learning with application to autonomous driving," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5355–5362, Oct. 2020.

[22] T. Yang, L. Zhao, W. Li, and A. Y. Zomaya, "Reinforcement learning in sustainable energy and electric systems: A survey," *Annu. Rev. Control*, vol. 49, pp. 145–163, Jan. 2020.

[23] K. K. Nguyen, T. Q. Duong, N. A. Vien, N.-A. Le-Khac, and L. D. Nguyen, "Distributed deep deterministic policy gradient for power allocation control in D2D-based V2 V communications," *IEEE Access*, vol. 7, pp. 164533–164543, 2019.

[24] X. Liu, J. Ospina, and C. Konstantinou, "Deep reinforcement learning for cybersecurity assessment of wind integrated power systems," *IEEE Access*, vol. 8, pp. 208378–208394, 2020.

[25] A.-S. rMohammed and M. Petr, "Reinforcement learning-based distributed BESS management for mitigating overvoltage issues in systems with high PV penetration," *IEEE Trans. Smart Grid*, vol. 11, no. 4, pp. 2980–2994, Jul. 2020.

[26] B. Sangiovanni, G. P. Incremona, M. Piastra, and A. Ferrara, "Self-configuring robot path planning with obstacle avoidance via deep reinforcement learning," *IEEE Control Syst. Lett.*, vol. 5, no. 2, pp. 397–402, Apr. 2021.

[27] C. He, Y. Wan, Y. Gu, and F. L. Lewis, "Integral reinforcement learning-based multi-robot minimum time-energy path planning subject to collision avoidance and unknown environmental disturbances," *IEEE Control Syst. Lett.*, vol. 5, no. 3, pp. 983–988, Jul. 2021.

[28] A. Uprety and D. B. Rawat, "Reinforcement learning for IoT security: A comprehensive survey," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8693–8706, Jun. 2021.

[29] X. Liu, W. Yu, F. Liang, D. Griffith, and N. Golmie, "On deep reinforcement learning security for industrial Internet of Things," *Comput. Commun.*, vol. 168, pp. 20–32, Feb. 2021.

[30] T. Yang, J. Li, H. Feng, N. Cheng, and W. Guan, "A novel transmission scheduling based on deep reinforcement learning in software-defined maritime communication networks," *IEEE Trans. Cognit. Commun. Netw.*, vol. 5, no. 4, pp. 1155–1166, Dec. 2019.

[31] S.-P. Choi, J.-U. Lee, and J.-B. Park, "Application of deep reinforcement learning to predict shaft deformation considering hull deformation of medium-sized oil/chemical tanker," *J. Mar. Sci. Eng.*, vol. 9, no. 7, p. 767, Jul. 2021.

[32] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for UAV attitude control," *ACM Trans. Cyber-Phys. Syst.*, vol. 3, no. 2, pp. 1–21, Feb. 2019.

[33] C.-H. Lee and M.-J. Tahk, "Autopilot design for unmanned combat aerial vehicles (UCAVs) via learning-based approach," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2019, pp. 476–481.

[34] W. Li, Y. Zhu, and D. Zhao, "Missile guidance with assisted deep reinforcement learning for head-on interception of maneuvering target," *Complex Intell. Syst.*, pp. 1–12, Nov. 2021, doi: 10.1007/s40747-021-00577-6.

[35] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[37] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

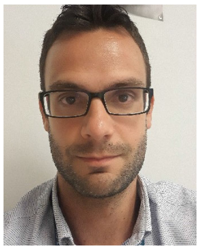
[38] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[39] H. Mahdianfar and E. Prempain, "Adaptive augmenting control design for a generic longitudinal missile autopilot," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2016, pp. 3138–3143.

[40] B. Cortez, "Reinforcement learning for robust missile autopilot design," 2020, *arXiv:2011.12956*.

[41] R. T. Reichert, "Dynamic scheduling of modern-robust-control autopilot designs for missiles," *IEEE Control Syst.*, vol. 12, no. 5, pp. 35–42, Oct. 1992.

[42] Y. B. Shtessel, I. A. Shkolnikov, and A. Levant, "Guidance and control of missile interceptor using second-order sliding modes," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 45, no. 1, pp. 110–124, Jan. 2009.



ANGELO CANDELI was born in Naples, Italy, in 1983. He received the bachelor's and master's degrees in automatic engineering from the University of Naples Federico II, in 2005 and 2008, respectively. He is the Head of the Guidance, Control and Navigation Department in the missile dynamics function inside the Missile Design Directorate at MBDA Italia SpA.

GIANMARIA DE TOMMASI (Senior Member, IEEE) was born in Milan, Italy, in 1975. He received the Laurea degree *(summa cum laude)* in electronic engineering and the Research Doctorate degree in computer and automatic engineering from the University of Naples Federico II, in 2001 and 2005, respectively. He is currently a Full Professor of automatic control with the Department of Electrical Engineering and Information Technology, University of Naples Federico II. Since 2002, he has been a Visiting Researcher at the Joint European Torus, U.K.; the ITER Organization, France; the EAST Tokamak, China; and the International Fusion Research Centre, Japan, where he has participated to various projects connected to the plasma magnetic control systems. His current research interests include control of nuclear fusion devices, fault detection and identification of discrete event systems modeled with Petri nets, and stability of hybrid systems. He has published more than 200 journals and conference papers on these topics, and has coauthored two monographs titled "Finite-Time Stability and Control" and "Finite-Time Stability: An Input-Output Approach."

DARIO GIUSEPPE LUI (Member, IEEE) was born in Avellino, in 1990. He received the master's degree (Hons.) in electronics engineering for automation and telecommunications (specialization in automation) and the Ph.D. degree in information technology for engineering from the University of Sannio, Benevento, Italy, in 2015 and 2020, respectively. He is currently a Research Fellow at the University of Naples Federico II. His work focuses on the distributed control of multi-agent systems in the presence of communication impairments, with application to automotive field and reinforcement learning.

ADRIANO MELE received the M.Sc. degree (Hons.) in automation engineering from the University of Naples Federico II, in 2015, and the joint Ph.D. degree in nuclear fusion science and engineering from the University of Naples Federico II and the University of Padua, in 2019. He has been a Visiting Researcher at the EAST Tokamak, Hefei, China; the Swiss Plasma Center of EPFL, Lausanne, Switzerland; the ITER Remote Experimentation Center, Rokkasho, Japan, and the French Commissariat à l'énergie atomique et aux énergies alternatives, Cadarache, France. He is currently a Researcher at the University of Tuscia, Viterbo. He held graduate and post-graduate courses on information technologies for industrial automation, industrial control systems, and plasma magnetic control in Tokamak reactors. His current research interests include control engineering, in particular with applications to fusion plasmas, including the investigation of data-driven and machine learning methods. He was a recipient of an EUROFusion Engineering Grant dedicated to the development of diagnostic systems for the forthcoming DTT Tokamak.

STEFANIA SANTINI (Member, IEEE) received the M.Sc. degree in electronic engineering and the Ph.D. degree in automatic control from the University of Naples Federico II, Naples, Italy, in 1996 and 1999, respectively. She is currently an Associate Professor of automatic control. She is involved in many projects with industry, including small- and medium-sized enterprises operating in the automotive field. Her research interests include the area of the analysis and control of nonlinear systems with applications to automotive engineering, transportation technologies, computational biology, and energy systems.

GAETANO TARTAGLIONE (Member, IEEE) was born in Caserta, Italy, in 1989. He received the master's degree (Hons.) in aerospace engineering from the University of Naples Federico II, Italy, in 2014, and the Ph.D. degree in information engineering from the University of Naples Parthenope, Italy, in 2018. Currently, he is a Researcher at the Engineering Department, University of Naples Parthenope. His research interests include control of nuclear fusion devices, multi-agent cooperative control, and finite-time stability and stabilization for the class of linear time-varying stochastic systems.