Aerospace Science and Technology 104 (2020) 105996

Contents lists available at ScienceDirect

Aerospace Science and Technology

www.elsevier.com/locate/aescte

Cooperative online Guide-Launch-Guide policy in a target-missile-defender engagement using deep reinforcement learning

Vitaly Shalumov

Technion - Israel Institute of Technology, Haifa, 32000, Israel

ARTICLE INFO

Article history:
Received 19 February 2019
Received in revised form 18 May 2020
Accepted 2 June 2020
Available online 15 June 2020
Communicated by Roberto Sabatini

ABSTRACT

A target-missile-defender engagement is considered, in which the missile attempts to intercept the target and the defender tries to prevent this interception via missile's interception. In this engagement, finding an optimal launch time of the defender and an optimal target guidance law before and after launch, which can be formulated as a switched system optimization problem, is crucial for improving performance of the target-defender team. The objective of this paper is to examine the potential of using deep reinforcement learning in switched system optimization. To that end, we propose estimating the optimal launch time of the defender and the optimal guidance law of the target online, using a reinforcement learning based method. A policy suggesting at each decision time the bang-bang target maneuver and whether or not to launch the defender was obtained and analyzed via simulations. Simulations showed the ability of the reinforcement learning based method to obtain a close to optimal level of performance in terms of the suggested cost function.

1. Introduction

In the scenario of an attacking missile and an evading target, numerous guidance laws were devised for both scenario participants [2,11]. In an effort to avoid interception, the target can deploy countermeasures. For example, a defending missile can be deployed by the target, in order to intercept the attacking missile, a scenario which is commonly referred in the literature as a target-missile-defender (TMD) engagement. The guidance laws of the TMD scenario participants were investigated using tools such as optimal control theory [30,22,23], differential games [25,21,28], nonlinear control theory [12], etc.

A common assumption in such an investigation is the given initial conditions of all the participants, an approach that avoids addressing the launching stage of the defender. By introducing the launch of the defender, the problem of optimal control has to be solved for two stages: the first stage, which consist out of a maneuvering target and an attacking missile, and the second stage, which in addition to the participants of the first stage, includes the defender. In addition, the optimal transition between the stages, i.e., the optimal defender launch time has to be obtained. The authors of [29] addressed this issue by assuming a constant acceleration maneuver of the target prior to launch, followed by a weapon-target-allocation algorithm, to obtain sub-optimal defender launch time. The problem of obtaining an optimal guidance law for at least one of the scenario participants, together with obtaining an optimal launch time with respect to a given cost function, can be viewed as a switched system optimization problem.

A switched system is composed out of sub-systems and a switching law specifying the active sub-system at each time instance [42]. In a case in which the switching sequence between sub-systems are an external input to the system, the system is called an externally forced switching system (EFS). For an internal input case [17], the system is called an internally forced switching system (IFS). Another distinction between several classes of switched systems is the presence of the controller in the model: for a model with an external controller, the system is named non-autonomous, while the system without an external controller is named autonomous.

Two main approaches are used in the literature to solve optimal control problems of continuous-time non-autonomous switched systems with EFS. The first approach [38,39] uses two-stage optimization techniques to find the optimal control switching times and the optimal control input. The second approach [4,36] uses an embedding transformation technique, which is utilized to transform the problem into a nonlinear optimization, solution of which can be obtained using nonlinear programming techniques. In the context of missile guidance, [34] developed the target evasion strategy from a missile performing multiple switches in the guidance law. The authors suggested the use of a matrix game for

E-mail address: vitaly.shalumov@gmail.com.

https://doi.org/10.1016/j.ast.2020.105996
1270-9638/

a case of unknown switching times, yielding a maximin evasion strategy, which guarantees a miss distance not smaller than the lower game value.

The optimal control of discrete-time non-autonomous switched systems with EFS, which will be investigated in this paper, was addressed for both linear systems with quadratic cost functions [41,10,40] and for nonlinear switched systems [8]. In [41], the authors obtained the explicit optimal piecewise linear state feedback controller via a solution of a set of difference Riccati equations (DRE). The switching times and the switching sequence were obtained through dynamic programming. In addition, the authors pointed out the problem of exponential growth of the positive semi-definite matrix set with time. A similar problem was encountered in [10], which the authors tackled by considering a sub-optimal cost function. The authors of [8], addressed the optimal control problem of discrete-time nonlinear switched systems, by decomposing the problem into two sub-problems and iteratively solving said sub-problems. In the first sub-problem, the optimal control input was obtained for a given switching sequence, while in the second sub-problem, the optimal switching sequence was obtained, utilizing the discrete filled function method. The authors of [7,6] suggested a solution to the autonomous switched system with fixed switching sequence problem by suggesting a formula for the cost gradient, and utilized said formula in the associated gradient-descent algorithms. For online implementations, the authors of [5] used a Newton-like optimization algorithm and showed that the convergence rate of the algorithm is quadratic for small estimation errors.

The aforementioned solutions to the switched system optimization problem suffer from the distinct drawback, if online performance is required, due to the fact that each set of initial conditions posses a different numerical optimization problem, which is computationally demanding. A different methodology is the utilization of reinforcement learning (RL) based methods, the advantage being a general, state dependent policy, that can be viewed as a closed loop guidance and switching law. In particular, deep reinforcement learning (DRL) uses a deep neural network (an artificial neural network [18], with a deep architecture, i.e., several hidden layers) as a function approximator in the RL algorithms. For an autonomous switched system, the problem can be treated as a classification or a regression problem of an unknown parameter out of a priori known set (discrete or continuous respectively), by treating the dynamical system as a time series with a given interval which can be solved using the deep learning (DL) method, as presented in [27].

RL is a mathematical framework in artificial intelligence that concerns autonomous learning using experience [33]. The RL agent interacts with an environment, thus collecting useful data regarding the dynamics of the environment, and uses said information to improve its policy, i.e., controller, with respect to a reward collected till the end of the scenario. In general, RL is divided into two types of methods (other divisions exist, such as gradient-based and gradient-free methods): the value function based methods and the policy gradient based methods. The value function based methods are based on the evaluation of functions such as the state-value function v or the action-value function q, and the synthesis of policy π from said value functions. Two of the popular algorithms utilizing those value functions are Q-learning [35] and SARSA [24]. On the other hand, the policy gradient methods are based on evaluating the policy directly. The approach of directly evaluating the policy has spawned numerous algorithms, such as REINFORCE [37], guided policy search [14], trust region policy optimization [26], and actor-critic methods such as the deterministic policy gradients [32].

In the past, the RL framework lacked scalability and thus were limited to low-dimensional problems. This problem was mitigated in the recent years, thanks to the advances in deep learning, which showed the great potential of deep neural networks in representing a high dimensional data such as images in a compact low-dimensional form. The use of deep learning algorithms within the RL framework spawned the field of deep reinforcement learning. In general, DRL is based on training deep neural networks to approximate the optimal policy π_*, and/or the optimal value functions v_* and q_*. The DRL framework was used to tackle previously intractable decision making problems [20,31,15,13,9]. In [20], Atari 2600 video games were played using a DRL algorithm, with super-human performance, thus demonstrating the fact that the RL agent can be trained solely based on the reward signal. In [31], the authors used supervised and reinforcement learning together with a heuristic search to defeat the human world champion at the game Go. In robotics, DRL algorithms were used to synthesize a control policy directly from camera inputs [13], for tasks such as screwing on a bottle cap and placing a shaped block in the correct hole. The authors of [19] utilized the RL formulation to obtain the navigation capability. Un/self-supervised tasks were used to improve data efficiency and task performance. In [16], RL was utilized for the design of an optimal controller for a nonlinear system with non-linear faults and unmeasured states. A brief survey of DRL methods is given in [1].

In this paper, we propose utilizing the RL framework in order to solve a switched system optimization problem. More specifically, we intend to use the DRL method in order to obtain a sub-optimal launch time of the defender and a sub-optimal bang-bang controller for the target in the TMD scenario. The advantage of using this approach as apposed to the classical optimization techniques is that it is capable of producing an accurate prediction of the optimal defender launch time and optimal target maneuver online. This capability stems from the fact that at the online stage, a neural network which represents the maneuver and the launch time, uses pre-defined function transformations that are determined at the offline stage, without the need of performing additional numerical simulations for the given initial conditions. The additional advantages of this approach are its validity for both linear and non-linear systems, and the minimal additional online computational cost for the extension to a multi-agent engagement.

The remainder of the paper is organized as follows. Section 2 formulates the engagement. Section 3 presents the suggested solution to the optimal launch time and guidance problem. Section 4 presents the implementation of the DRL estimator and its performance in terms of the total reward. Section 5 presents the main conclusions of the paper.

2. Engagement

In this section, let us present the formulation of the following engagement, addressed throughout the course of this paper: an attacking missile tries to intercept the target, the target tries to evade the missile, and the defender tries to intercept the missile before it intercepts the target. In addition, let us make the following engagement assumptions:

1. Perfect state knowledge of all the scenario participants, i.e., the guidance laws of the defender and the missile are assumed to be known.
2. The scenario takes place in the end game phase. This assumption allows us to use linearization along the initial line of sights (LOSs). The speed of the vehicles is assumed constant during this phase. Note that the methodology of the problem solution does not depend on this assumption, and its made strictly for engagement simplification purposes.
3. The target uses a bang-bang controller, applied at discrete times, defined by the decision time interval Δt_c. The launch of the defender occurs at one of those discrete times, mean-

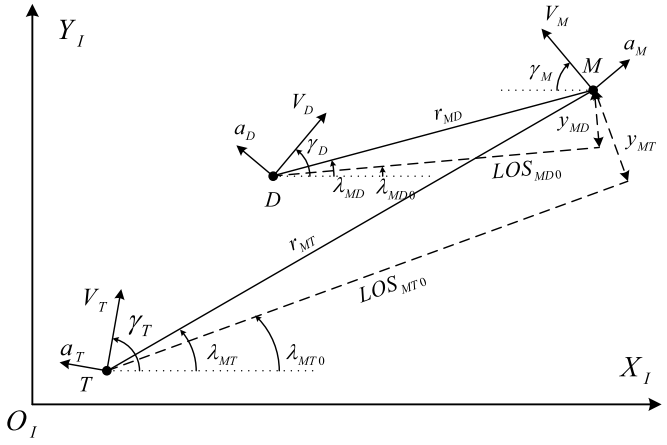


Fig. 1. Planar engagement geometry.

ing, at that time instant, the target launches the defender and then executes its control command.

2.1. General kinematics and dynamics

Fig. 1 presents a schematic view of the planar end-game geometry of a target, a missile chasing said target, and a defender chasing the missile, where $X_I - O_I - Y_I$ is a Cartesian inertial reference frame. The notations MT and MD denote the attacking missile with an evading target and the attacking missile with defending defender duo, respectively. The speed, normal acceleration, and flight-path angles are denoted by V, a, and γ, respectively. The range between the adversaries is r, and λ is the angle between a LOS and the X_I axis. Note that LOS_{MT0} is defended at the beginning of the end-game phase, while LOS_{MD0} is defined at the defender launch time.

The nonlinear kinematic equations for planer end-game geometry, with neglected gravitation for the missile-target duo are:

$$\dot{r}_{MT} = -V_M \cos(\gamma_M + \lambda_{MT}) - V_T \cos(\gamma_T - \lambda_{MT}) \tag{1a}$$

$$\dot{\lambda}_{MT} = [V_M \sin(\gamma_M + \lambda_{MT}) - V_T \sin(\gamma_T - \lambda_{MT})]/r_{MT} \tag{1b}$$

Similarly, the kinematic equations for the missile-defender duo are:

$$\dot{r}_{MD} = -V_M \cos(\gamma_M + \lambda_{MD}) - V_D \cos(\gamma_D - \lambda_{MD}) \tag{2a}$$

$$\dot{\lambda}_{MD} = [V_M \sin(\gamma_M + \lambda_{MD}) - V_D \sin(\gamma_D - \lambda_{MD})]/r_{MD} \tag{2b}$$

The rate of the path angle is defined as:

$$\dot{\gamma}_k = a_k/V_k, \qquad k = \{M, T, D\} \tag{3}$$

2.2. Linear scenario

Although the methodology of the problem solution is general, let us exemplify the solution on a sub-class of the problem, by assuming that the dynamics in the end game of all the participants can be represented by linear equations, and that the initial LOS's of the target and the defender are identical.

The system can be formulated as a switched system with EFS. The dynamics of the defender before and after launch can be formulated as follows:

$$\begin{cases} \dot{\mathbf{x}}_k = \mathbf{A}_k\mathbf{x}_\mathbf{k} + \mathbf{B}_k u_k \\ a_k = \mathbf{C}_k\mathbf{x}_k + d_k u_k \quad k = \{M, T\},\ before\ launch \\ \mathbf{x}_D = \mathbf{x}_T \\ \dot{\mathbf{x}}_k = \mathbf{A}_k\mathbf{x}_\mathbf{k} + \mathbf{B}_k u_k \\ a_k = \mathbf{C}_k\mathbf{x}_k + d_k u_k \quad k = \{M, T, D\},\ after\ launch \end{cases} \tag{4}$$

where $\mathbf{x}_k$ is the vector of an agent's internal state variables and u_k is its controller.

2.2.1. Linearized kinematics

In the endgame phase, the vehicles are near the collision triangles, meaning we can linearize the kinematics around the initial LOSs. We denote y_{MT} as the relative displacement between the missile and the target, normal to LOS_{MT0}. Similarly, y_{MD} is the relative displacement between the missile and the defender, normal to LOS_{MD0} (see Fig. 1). The state vector of the linearized problem is:

$$\mathbf{x} = \left[\mathbf{x}_{MT}^T\ \mathbf{x}_{MD}^T\right]^T \tag{5}$$

where

$$\mathbf{x}_{MT} = \left[y_{MT}\ \dot{y}_{MT}\ \mathbf{x}_M^T\ \mathbf{x}_T^T\right]^T \tag{6}$$

$$\mathbf{x}_{MD} = \left[y_{MD}\ \dot{y}_{MD}\ \mathbf{x}_D^T\right]^T \tag{7}$$

The equations of motion are:

$$\dot{\mathbf{x}} = \begin{cases} \dot{x}_{MT,1} = x_{MT,2} \\ \dot{x}_{MT,2} = a_T - a_M \\ \dot{\mathbf{x}}_M = \mathbf{A}_M\mathbf{x}_M + \mathbf{B}_M u_M \\ \dot{\mathbf{x}}_T = \mathbf{A}_T\mathbf{x}_T + \mathbf{B}_T u_T \\ \dot{x}_{MD,1} = x_{MD,2} \\ \dot{x}_{MD,2} = a_M - a_D \\ \dot{\mathbf{x}}_D = \mathbf{A}_D\mathbf{x}_D + \mathbf{B}_D u_D \end{cases} \tag{8}$$

and can be rewritten in a vector form as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}[u_T\ u_D]^T + \mathbf{C}u_M \tag{9}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{MT} & [0] \\ \mathbf{A}_{21} & \mathbf{A}_{MD} \end{bmatrix},\ \mathbf{B} = \begin{bmatrix} \mathbf{B}_{MT} & [0] \\ [0] & \mathbf{B}_{MD} \end{bmatrix},\ \mathbf{C} = \begin{bmatrix} \mathbf{C}_{MT} \\ \mathbf{C}_{MD} \end{bmatrix} \tag{10}$$

$$\mathbf{A}_{MT} = \begin{bmatrix} 0 & 1 & [0] & [0] \\ 0 & 0 & -\mathbf{C}_M & \mathbf{C}_T \\ [0] & [0] & \mathbf{A}_M & [0] \\ [0] & [0] & [0] & \mathbf{A}_T \end{bmatrix},\ \mathbf{B}_{MT} = \begin{bmatrix} 0 \\ d_T \\ [0] \\ \mathbf{B}_T \end{bmatrix},\ \mathbf{C}_{MT} = \begin{bmatrix} 0 \\ -d_M \\ \mathbf{B}_M \\ [0] \end{bmatrix} \tag{11}$$

$$\mathbf{A}_{MD} = \begin{bmatrix} 0 & 1 & [0] \\ 0 & 0 & -\mathbf{C}_D \\ [0] & [0] & \mathbf{A}_D \end{bmatrix},\ \mathbf{B}_{MD} = \begin{bmatrix} 0 \\ -d_D \\ \mathbf{B}_D \end{bmatrix},\ \mathbf{C}_{MD} = \begin{bmatrix} 0 \\ d_M \\ [0] \end{bmatrix} \tag{12}$$

$$\mathbf{A}_{21} = \begin{bmatrix} [0] & [0] & [0] \\ [0] & \mathbf{C}_M & [0] \\ [0] & [0] & [0] \end{bmatrix} \tag{13}$$

with [0] denoting a matrix of zeros with appropriate dimensions.

2.2.2. Timeline

After linearization, under the assumption of small deviations from the collision triangle, the expected interception time is calculated as follows:

$$t_{f,MT} = -\frac{r_{MT0}}{\dot{r}_{MT0}} = \frac{r_{MT0}}{V_M \cos(\gamma_{M0} + \lambda_{MT0}) + V_T \cos(\gamma_{T0} - \lambda_{MT0})} \tag{14}$$

$$t_{f,MD} = -\frac{r_{MD0}}{\dot{r}_{MD0}} = \frac{r_{MD0}}{V_M \cos(\gamma_{M0} + \lambda_{MD0}) + V_D \cos(\gamma_{D0} - \lambda_{MD0})} \tag{15}$$

Δt is defined as the difference between the time at which missile is expected to intercept the target and the time the defender is expected to intercept the missile, thus requiring Δt to be greater than zero.

$$\Delta t = t_{f,MT} - t_{f,MD} \tag{16}$$

2.2.3. Missile and defender guidance laws

Let us now present the general family of linear guidance laws, which are assumed to be used by both the missile and the defender.

Under the assumptions of linear kinematics, perfect information (including perfect knowledge about the future controls of the evader), and unbounded controls, the following general cost function can be minimized by using optimal control theory:

$$J = y^2(t_f) + \frac{\beta_P}{2}\int_0^{t_f} u_P^2 dt \tag{17}$$

where $y^2(t_f)$ is the squared miss distance perpendicular to the initial LOS, subscript P stands for pursuer, and β_P is weight on the control effort $\int_0^{t_f} u_P^2 dt$.

The optimal solution results in a family of linear guidance laws, being a function of pursuer-evader engagement state variables and the evader's control. In the case where the missile is the pursuer and target is the evader we get:

$$u_M = \mathbf{K}_{MT}(t_{go,MT}, \beta_M)\mathbf{x}_{MT} + S_{MT}(t_{go,MT}, \beta_M)u_T \tag{18}$$

$$\mathbf{K}_{MT}(t_{go,MT}, \beta_M) = \begin{bmatrix} K_{1,MT} & K_{2,MT} & \mathbf{K}_{M,MT} & \mathbf{K}_{T,MT} \end{bmatrix} \tag{19}$$

whereas for the case where the defender is the pursuer and missile is the evader we get:

$$u_D = \mathbf{K}_{MD}(t_{go,MD}, \beta_D)\begin{bmatrix} \mathbf{x}_{MD}^T & \mathbf{x}_M^T \end{bmatrix}^T + S_{MD}(t_{go,MD}, \beta_D)u_M \tag{20}$$

$$\mathbf{K}_{MD}(t_{go,MD}, \beta_D) = \begin{bmatrix} K_{1,MD} & K_{2,MD} & \mathbf{K}_{D,MD} & \mathbf{K}_{M,MD} \end{bmatrix} \tag{21}$$

Among them, the familiar proportional navigation (PN), augmented proportional navigation (APN), and optimal guidance law (OGL), that can be written as:

$$\begin{aligned} u_M &= N'_{MT,j}\frac{Z_{MT,j}}{t_{go,MT}^2} \\ u_D &= N'_{MD,j}\frac{Z_{MD,j}}{t_{go,MD}^2} \end{aligned}, \qquad j = \{PN, APN, OGL\} \tag{22}$$

where for the PN guidance law, Z_{PN} is equal to:

$$Z_{MT,PN} = y_{MT} + \dot{y}_{MT} t_{go,MT} \tag{23a}$$

$$Z_{MD,PN} = y_{MD} + \dot{y}_{MD} t_{go,MD} \tag{23b}$$

and for a finite weight β, N'_{PN} is equal to:

$$N'_{MT,PN} = \frac{3t_{go,MT}^3}{3\beta_M + t_{go,MT}^3} \tag{24a}$$

$$N'_{MD,PN} = \frac{3t_{go,MD}^3}{3\beta_D + t_{go,MD}^3} \tag{24b}$$

Remark: we chose the weights β_D and β_M to be finite, to account for the fact that, in a real scenario, the controls are bounded (the lower the bound on the controls, the larger the weights should be chosen). This choice leads, however, to a miss distance perpendicular to the initial LOS, even for a known evader control strategy.

3. Guide-Launch-Guide policy

Under the assumption of given guidance laws for the missile and the defender, the problem addressed in this paper is defined as finding an optimal defender launch time and an optimal guidance law of the target before and after launch, in a single target, single missile, and single defender scenario. The minimal and maximal launch times are constrained by the time instance at which the attacking missile is discovered and by the required safe interception (defender - missile) distance from the target. The problem of finding the optimal launch time of the defender and the optimal guidance law of the target before and after launch (named the Guide-Launch-Guide policy) can be solved using various optimization techniques (see section 1). The drawback of most of those techniques is the fact that they require a significant amount of computational resources, which in turn translates to run times, which are not suitable for online performance, thus making the use of those techniques undesirable for the solution of our problem. For example, a greedy search of 20 optimal bang-bang maneuvers and an optimal launch time, results in $20 \cdot 2^{20} \approx 2.1e7$ permutations, for which the simulation has to be run.

To achieve the required online performance, let us suggest the use of the deep reinforcement learning method in order to identify the defender's sub-optimal launch time, and to obtain the sub-optimal bang-bang guidance law before and after launch. Note that this approach scales up to a multi-agent engagement, without increasing the online computational load, as apposed to most online optimizers, that require significant amount of additional computations for each agent added to the scenario.

3.1. Simulated environment

In order to synthesize the optimal bang-bang controller, together with the optimal defender launch time, a simulated environment is defined based on section 2.1, with ideal dynamics of all the scenario participants. The guidance laws for the defender and the missile are taken from section 2.2.

Remark: The weights β_D and β_M are tuning parameters which are responsible for limiting pursuer's controller. Increasing the weight value decreases maneuvering capabilities, as can be seen from Eq. (22) and Eq. (24). The values of the weights were chosen as to reflect the class of the missile and the defender. The attacking missile is an anti-aircraft missile designed to intercept targets with relatively low maneuvering capabilities, thus it is assumed that the missile will have relatively low maneuvering capabilities as well, a fact that is reflected by the value of $\beta_M = 1$. The defender is designed to intercept missiles, which are generally much more agile than aircrafts, thus a smaller weight on the control effort is typically used in guidance laws of such missiles. Following that logic, the value of $\beta_D = 0.01$ was chosen for the guidance law of the defender. For the initial conditions addressed in section 4,

the miss distance of the defender was less than 1 [m] and the absolute value of the acceleration for all agents was under 15 [g].

The simulated environment consists out of two main stages:

1. **Initialization**:
   In the initialization stage, random initial conditions are selected. The random variables are the X position of the missile, and the path angle of the target and the missile. This selection fully defines the 2D geometry of the engagement, for given guidance law parameters and vehicles' velocities (the guidance law parameters and vehicles' velocities were not randomized in order to keep the offline computation time short). In addition, at the initialization stage, the initial values of the following 11 states of the problem are calculated:
   - Target's position (two states)
   - Defender's position (at the initialization stage it is identical to the target's position - two states)
   - Missile's position (two states)
   - The vehicles' path angle (three states)
   - The expected interception times of the target by the defender and the missile by the target (two states - according to Eq. (14) and Eq. (15)).
2. **Run**:
   (a) Input: the simulation receives the current state of the engagement and an action command to the target. The action has four possible values:
   - **0** - Do not launch, use maximum controller.
   - **1** - Do not launch, use minimum controller.
   - **2** - Launch the defender and then use maximum controller.
   - **3** - Launch the defender and then use minimum controller.

   After launch, actions **0** and **2** are identical (the same goes for actions **1** and **3**).

   (b) Output: the output of the simulation is determined by whether or not an interception occurred.
   - No interception: the simulation calculates the reward, which is the minus of the weighted control effort of the defender in the decision time interval Δt_c:

$$R_i = -\beta_D \int_{t_i}^{t_{i+1}=t_i+\Delta t_c} u_D^2 dt \tag{25}$$

   This reward represents the fact that the defender has a limited control capability.
   - Interception: the simulation returns the reward for the last step R_{last}, which the minus of the sum of the control effort of the defender and the squared interception distance $d_{MD}^2(t_{f,MD})$ (not to be confused with $y_{MD}^2(t_{f,MD})$ which is the squared miss distance perpendicular to the initial LOS). This is the stopping condition of the simulation.

$$R_{last} = -\left(d_{MD}^2(t_{f,MD}) + \beta_D \int_{t_i}^{t_{f,MD}} u_D^2 dt \right) \tag{26}$$

3.2. Reinforcement learning essentials

In this section, let us present the RL framework (section 3.2.1) and two important methods within RL: value function based methods (section 3.2.2) and policy gradient based methods (section 3.2.3).

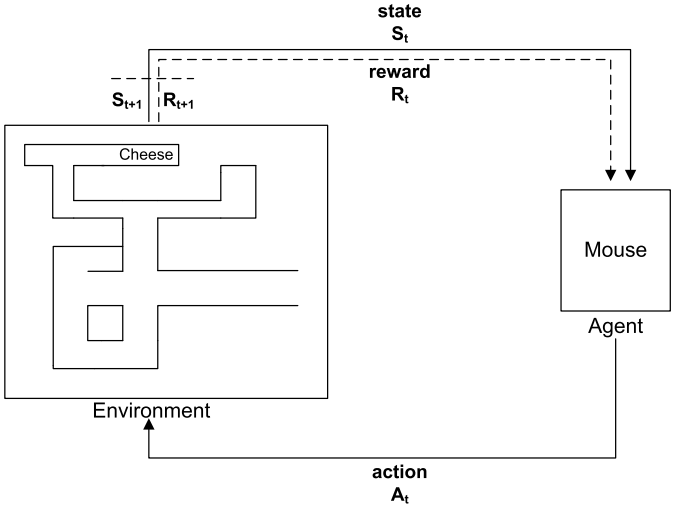


Fig. 2. The agent - environment loop in a Markov Decision Process (adapted from [1]).

3.2.1. Reinforcement learning framework

An autonomous RL agent, used to learn and interact with the environment can be formally described as a Markov Decision Process (MDP). The agent interacts with the environment at time t using an action $A_t \in \mathcal{A}(S_t)$, based on the state $S_t \in \mathcal{S}$ ($\mathcal{S}$ is the set of possible states and $\mathcal{A}(S_t)$ is the set of actions available at state S_t) and the reward $R_t \in \mathbb{R}$. The action has the consequence of moving the agent to state S_{t+1} and receiving the reward of R_{t+1}. The goal of the agent is to learn a control strategy (policy) π, that maximizes the expected return G_t, which is the cumulative and discounted reward until the termination of scenario (in the case of an episodic MDP):

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{27}$$

In an episodic MDP, the state is reset after each episode of length T. Each episode is named a rollout, in which the reward discount factor, $\gamma \in [0, 1]$, is responsible for placing more emphasis on the immediate rewards. In the episodic MDP described above, the probability of specific values of the states and rewards are given by:

$$p(s', r \,|\, s, a) \doteq \Pr\left\{ S_t = s', R_t = r \,|\, S_{t-1} = s, A_{t-1} = a \right\} \tag{28}$$

The policy π is defined as a mapping from states to probabilities of selecting each possible action. The goal of the RL agent is to find the optimal policy π^*.

The scheme of agent's interaction with the environment in the MDP setting is given in Fig. 2.

3.2.2. Value function methods

The value function methods are based on estimating the value of how good it is to be in the given state, or how good it is to be in the given state and take the given action. The notion of "how good" is evaluated using the expected return [33].

The state-value function is given by:

$$v_\pi(s) \doteq \mathbb{E}_\pi \left[G_t \,|\, S_t = s \right] \tag{29}$$

whereas the action-value function (otherwise known as the quality function) is given by:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi \left[G_t \,|\, S_t = s, A_t = a \right] \tag{30}$$

To learn the state-value function, the Markov property is used, to obtain the Bellman equation [3] for the state-value function:

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_\pi(s') \right] \tag{31}$$

similarly, the Bellman equation for the action-value function is:

$$q_\pi(s, a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a') \right] \tag{32}$$

The optimal state-value function is the maximal state-value function achievable by any policy for state s, and is given by the state-value Bellman optimality equation:

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_*(s') \right] \tag{33}$$

The optimal action-value function is the maximal action-value function achievable by any policy for state s and action a, and is given by the action-value Bellman optimality equation:

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right] \tag{34}$$

3.2.3. Policy gradient methods

The policy gradient methods, as opposed to the value function methods seek to directly find the policy $\pi(a|s, \boldsymbol{\theta})$ (without necessarily using a value function for action selection) and update the parameters $\boldsymbol{\theta}$ to maximize the expected return $\mathbb{E}(G|\boldsymbol{\theta})$ by gradient based or gradient free optimization.

Gradients provide a strong learning signal in terms of in which direction to improve the policy. The computation of the expected returns requires averaging, that in the model-free RL setting uses a Monte Carlo estimate of the expected return. This sample based approach hinders the use of gradient based methods, so an estimator of the gradient is commonly used. This estimator, known as the REINFORCE rule, computes the gradient of expectation over a function of a random variable X with respect to parameters $\boldsymbol{\theta}$:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_X [f(X, \boldsymbol{\theta})] = \mathbb{E}_X [f(X, \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(X)] \tag{35}$$

The use of rollout's empirical return results in a high variance gradient. To reduce the variance, unbiased estimates are introduced, via baseline substraction. An algorithm which uses the REINFORCE rule with a baseline is the selected RL algorithm for this paper and thus will be extensively presented in section 3.3.1.

3.3. Reinforcement learning model

In this section let us present the chosen deep reinforcement learning model. In the current setting, we use 11 states and 4 actions as described in section 3.1. At the start of each decision time interval Δt_c, the RL agent suggests an action, and receives the updated states and reward. Prior to interception, the reward at each Δt_c is R_i, given Eq. (25), while at the last step prior to interception the reward is R_{last}, given Eq. (26).

Thus the total reward of the episode is given by the sum all the intermediate reward:

$$R = \sum_{i=0}^{last} R_i = d_{MD}^2(t_{f,MD}) + \beta_D \int_0^{t_{f,MD}} u_D^2 dt \tag{36}$$

3.3.1. Reinforcement learning algorithm

As previously mentioned, we used REINFORCE with baseline as our RL algorithm. REINFORCE uses the following algorithm to update the parameters $\boldsymbol{\theta}$ (which in our case are the weights of the neural network) of the policy π:

Algorithm 1: REINFORCE with Baseline, adapted from [33].

Input: a differentiable policy parametrization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value parametrization $\hat{v}(s, \mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0, \alpha^{\mathbf{w}} > 0$

Initialize policy parameter $\boldsymbol{\theta}$ and state-value weights $\mathbf{w}$
Repeat forever:
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  For each step of the episode $t = 0, \ldots, T-1$:
    $G_t \leftarrow$ return from step t
    $\delta \leftarrow G_t - \hat{v}(S_t, \mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \gamma^t \delta \nabla_{\mathbf{w}} \hat{v}(S_t, \mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta})$

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t, \boldsymbol{\theta}_t)}{\pi(A_t|S_t, \boldsymbol{\theta}_t)} \tag{37}$$

Due to the fact that REINFORCE uses the complete return G_t it is classified as a Monte Carlo algorithm, and as such may be of high variance and thus produce slow learning. To reduce the variance and thus speed up the learning process, a baseline can be used, in which case, the update rule of the policy weights becomes:

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha (G_t - b(S_t)) \frac{\nabla_{\boldsymbol{\theta}} \pi(A_t|S_t, \boldsymbol{\theta}_t)}{\pi(A_t|S_t, \boldsymbol{\theta}_t)} \tag{38}$$

One possible choice for the baseline is an estimate of the state-value function $\hat{v}(S_t, \mathbf{w})$, where $\mathbf{w} \in \mathbb{R}^m$ are the state-value function weights. The pseudocode for the REINFORCE with baseline algorithm is given Algorithm 1, using the learned state-value function as the baseline.

The algorithm is set to run 2e4 iterations, i.e., 2e4 random initial conditions are considered.

As a function approximation, deep neural networks were used. Both the neural network of the policy and the neural network of the baseline consist out of 5 fully connected layers with N neurons at each layer. In the current paper, two networks are considered: $FC1$, in which $N = 850$ and $FC2$ in which $N = 400$. The exploration algorithm used is the ϵ-greedy, with linear annealing from 0.1 at the first iteration to 0 at the 14000 iteration. The discount factor used is equal to 0.92.

We evaluated several RL algorithms, including DQN and Actor-Critic algorithms with several deep neural network architectures. We omit a full comparison from the paper, presenting only the REINFORCE with baseline algorithm with a fully connected deep neural network, which achieved the best results.

3.4. Genetic algorithms

Throughout this section, a RL-based solution was demonstrated in detail. To evaluate the performance of the RL method, let us solve the problem of optimal launch time and guidance law using a computationally heavy optimization method, that is able to tackle mixed integer optimization problems. One possible method is the genetic algorithm (GA).

The GA is based on natural selection. The algorithm defines a population of individuals, each one with certain genes (which are the parameters to be optimized, as to achieve the minimization of the cost function, otherwise known as a fitness function). At each generation, the population evolves via three possible mechanisms:

- **Elitism**: the individual from the previous generation, called a parent, carries over to the next generation. The individual in the next generation is called a child.
- **Mutation**: several genes from the parent are modified to produce a child.
- **Cross-over**: a combination of two parents is selected to produce a child.

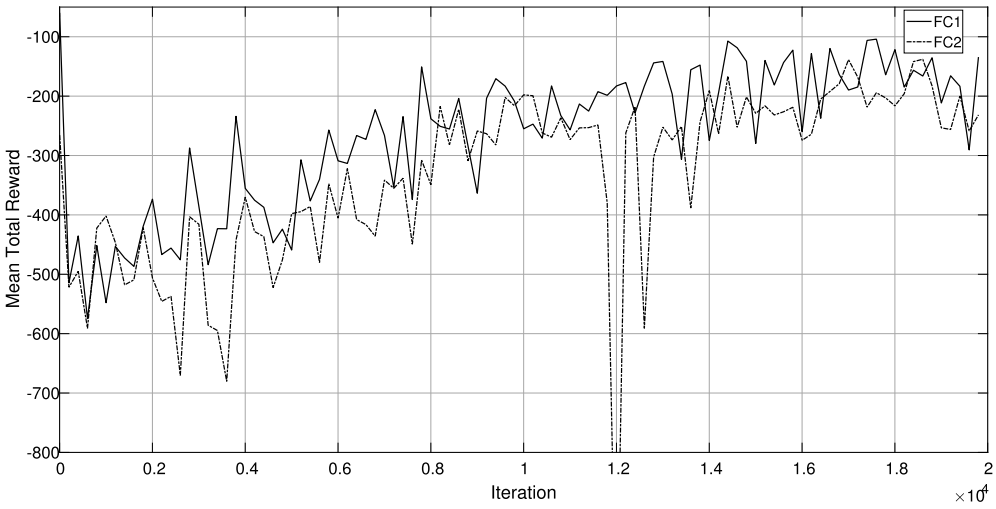


Fig. 3. Mean total reward vs. iteration, for neural network architectures $FC1$ and $FC2$.

Table 1
Initial conditions.

Name	X_M [m]	γ_M [rad]	γ_T [rad]
$IC1$	6601	0.049	−0.164
$IC2$	5331	−0.243	−0.058
$IC3$	6500	−0.062	−0.125

The algorithm stops when a certain stopping criterion is met (such as maximum generations number, time limit, etc.).

4. Analysis and discussion

In this section, the performance of the DRL method is evaluated using the simulated environment, described in section 3.1. The scenario was simulated for participants having ideal dynamics, with speeds of 500$[m/s]$, 1000$[m/s]$ and 1000$[m/s]$ for the target, the missile, and the defender respectively. The weights used by the proportional navigation guidance laws of the defender and the missile are $\beta_D = 0.01$ and $\beta_M = 1$ respectively. The decision time interval Δt_c is taken as 0.2$[s]$, and the absolute value of the target maneuver is taken as 10$[g]$. The maximal launch time of the defender is equal to $t_{f,MT} - 1$ to ensure safe interception (defender - missile) distance from the target.

The performance of the trained RL policy is exemplified for three initial conditions, presented in Table 1.

To present the learning process of the RL algorithm, and to compare the performance of several neural network architectures ($FC1$ and $FC2$, as defined in section 3.3.1), for the REINFORCE with baseline algorithm, let us compute the mean total reward (total reward, as presented in Eq. (36), averaged for the last 100 episodes) at each episode. Fig. 3 presents the mean total reward (down-sampled for presentation purposes) vs. iteration (episode number).

From Fig. 3, it is evident that the RL agent improves the policy as more episodes are introduced, for both neural network architectures (the overall slope of the mean total reward is positive), until it reaches a policy capable of producing a mean total reward of about −150 for $FC1$ and about −200 for $FC2$. Due to the fact that slightly better results were obtained using architecture $FC1$, it is selected to be used in our RL model throughout the remainder of this section.

To illustrate the obtained policy, let us evaluate it for the initial conditions $IC1$, $IC2$, and $IC3$ (Table 1). The obtained actions of the target and the trajectories of the vehicles are given in Fig. 4 and Fig. 5 respectively. In Fig. 4, the actions are marked by ◯ and the launch of the defender is marked by +. The doted line in Fig. 5 is the trajectory of the target (marked by T), from which the defender (marked by D) is launched towards the attacking missile (marked by M), until the defender - missile interception point (marked by $*$).

It is interesting to note that according to Fig. 4, the target chooses to launch the defender before it reaches the maximal allowable launch time of the defender for $IC1$ and $IC3$ (which was set to $t_{f,MT} - 1[s]$ to ensure safe defender - missile interception distance from the target), while for $IC2$ the safety limit served as the launch trigger.

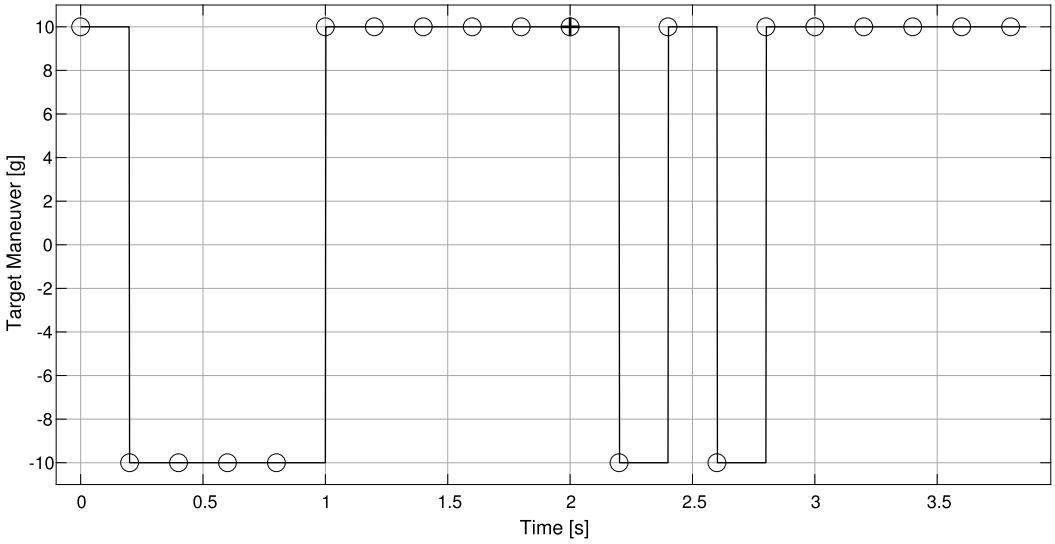
From Fig. 5 it is evident that the target tries to position itself to launch the defender in a state which will ensure a relatively linear trajectory of the defender, thus minimizing its control effort, which is the second objective in our multi-objective cost function, which was introduced in Eq. (36) (the scale of the y axes in Fig. 5(c) is tens of meters, which leads to the misleading appearance of a nonlinear defender trajectory). Some maneuvering of the defender is expected, as to ensure the minimization of the squared interception distance, which is the first part of our multi-objective cost function. The fact that the launch of the defender was chosen at the second half of the flight time interval is consistent with the intuition that the target will first attempt to maneuver to a favorable interception (by the defender) geometry without any penalty, and only then will launch the defender to ensure safe defender - missile interception distance from the target.

To evaluate the performance of the RL method in terms of policy optimality, the performance of the trained RL policy is compared to the solution obtained by the GA optimization technique. Fig. 6 presents the mean and the best individual scores of the GA at each generation for $IC1$, evaluated using a loss function which is the minus of the total reward. The GA optimization ran for 50 generations, with the population size of 40 individuals.

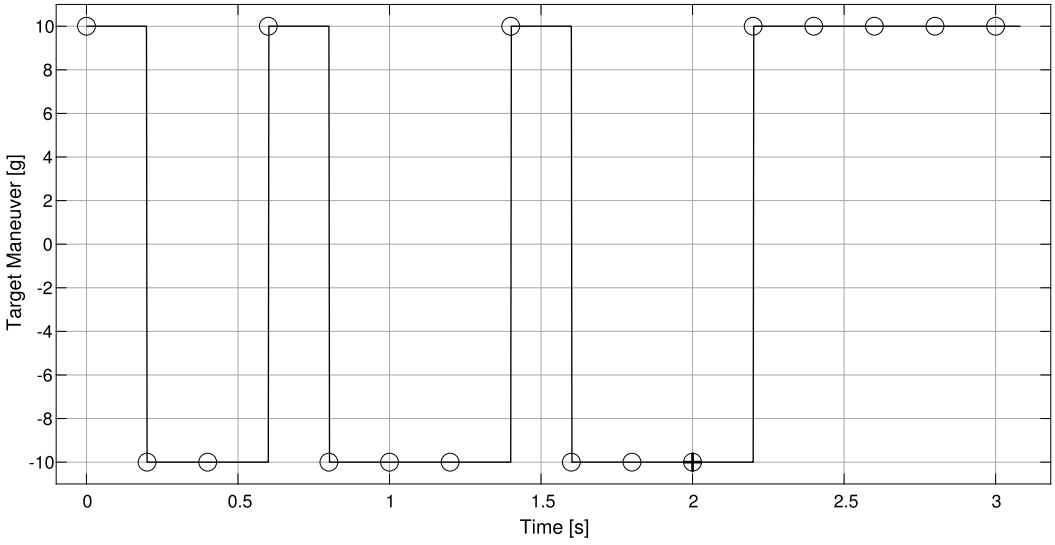
From the evolution of the loss presented in Fig. 6, it is evident that the GA has troubles with converging to a minimum, illustrating the fact that finding an optimal solution to the Guide-Launch-Guide optimization problem is difficult.

Fig. 7 presents the mean and the best individual scores of the GA, for a generation which achieved the best total reward (minimal loss) for the initial conditions of $IC1$, $IC2$, and $IC3$, in comparison to the results obtained by the RL method (recall that the loss is minus of the total reward).

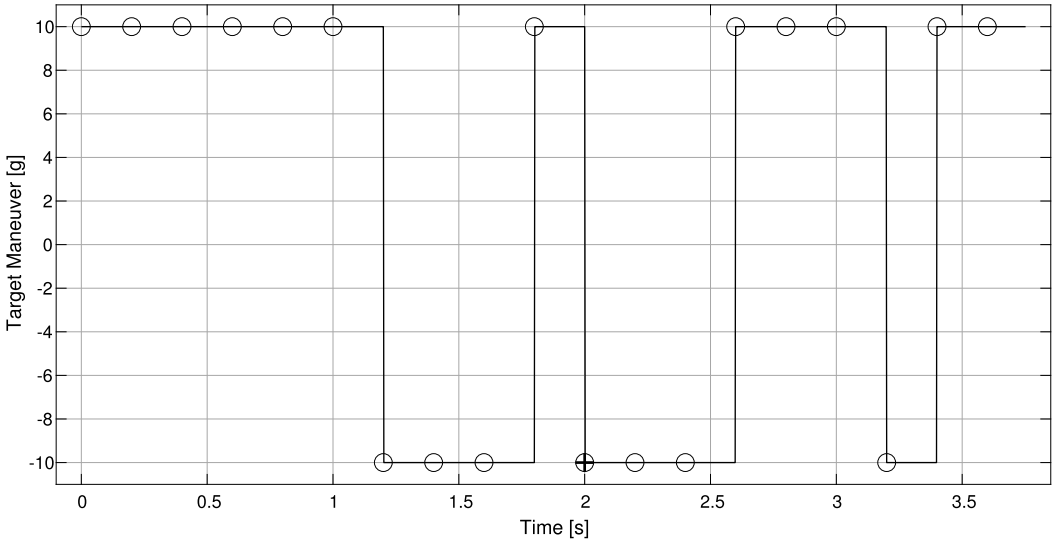
From Fig. 7, it can be seen that both the RL and the GA methods are able to produce near optimal results (recall that the reward is non-positive by the definition of our reward function), with GA producing better results. The mean reward of each GA generation



(a) IC1

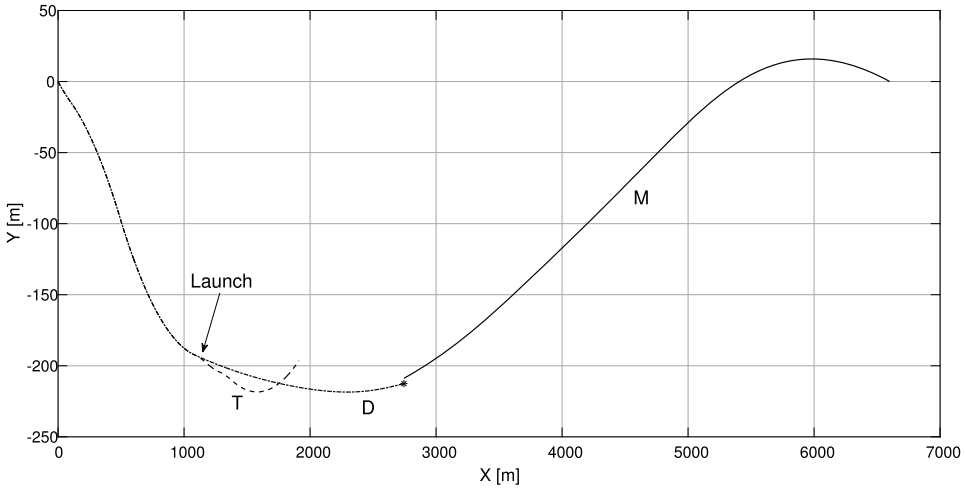


(b) IC2

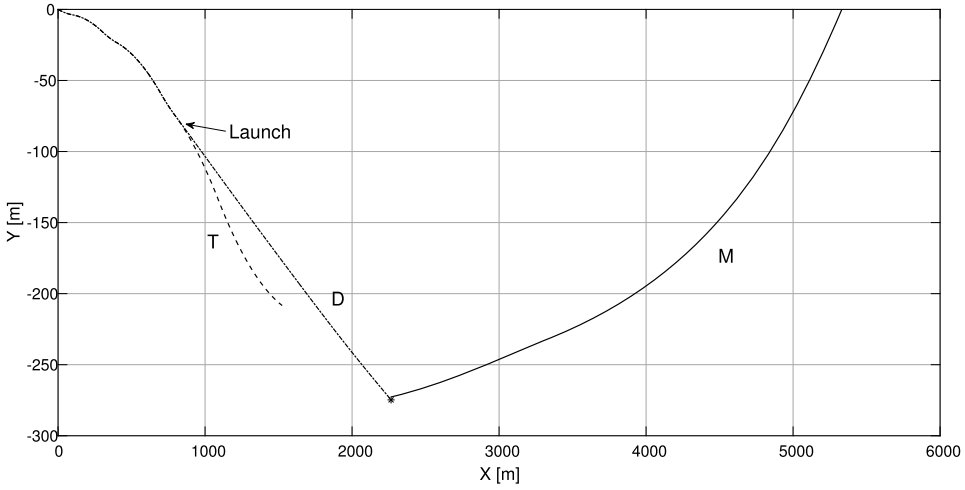


(c) IC3

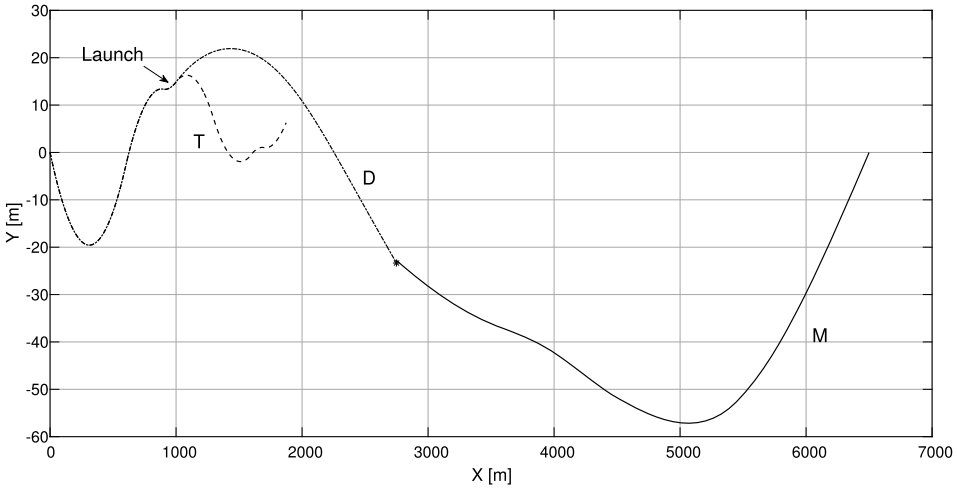
Fig. 4. Bang-bang actions using the trained policy for three initial conditions: (a) $IC1$; (b) $IC2$; (c) $IC3$.



(a) IC1



(b) IC2



(c) IC3

Fig. 5. Vehicles' trajectories using the trained policy for three initial conditions: (a) $IC1$; (b) $IC2$; (c) $IC3$.

presented in Fig. 7 is around 600, emphasizing the quality of the policy obtained by the RL method.

To further illustrate the performance level of the proposed approach, let us compare its performance to another online TMD method [27] which addressed switched system optimization. In it, the defender's launch time is optimized using DL under the assumption of a constant target maneuver. This assumption is necessary to reduce the number of classes to be classified by the DL estimator. The performance comparison is given in Fig. 8.

Fig. 8 demonstrates the fact that the RL method, which is not subject to the constraint of a constant target maneuver can obtain higher level of performance in terms of the selected cost function.

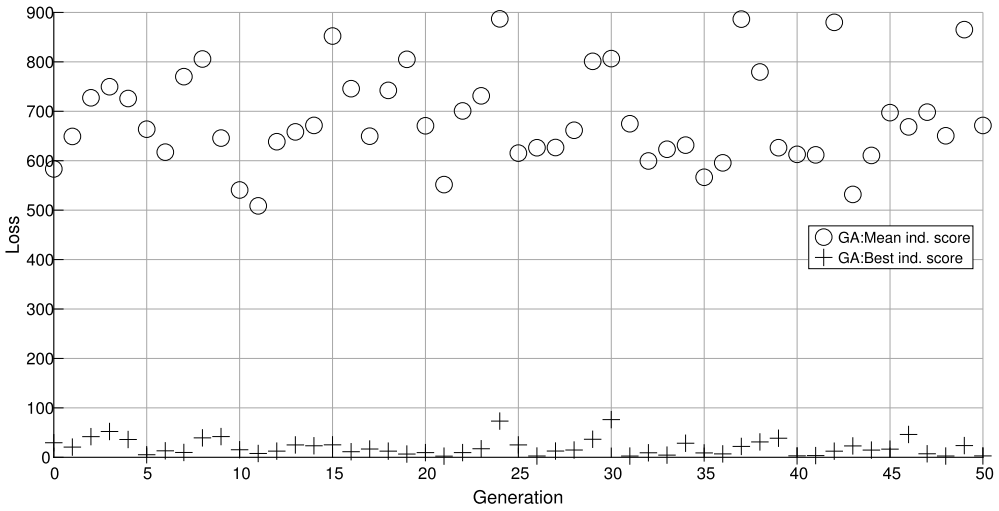


Fig. 6. Loss vs. generation, for initial conditions $IC1$ using GA.

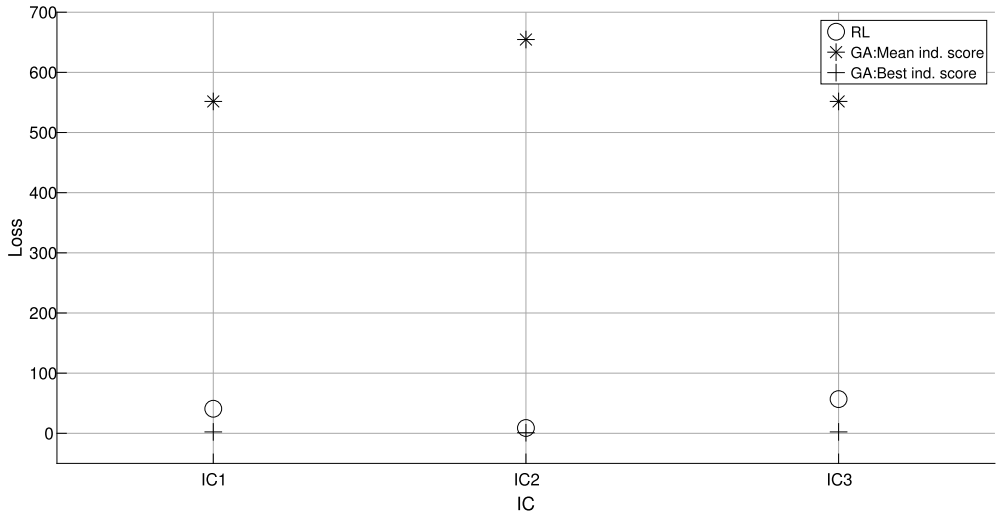


Fig. 7. Loss comparison between GA and RL, for three initial conditions: $IC1$, $IC2$, and $IC3$.

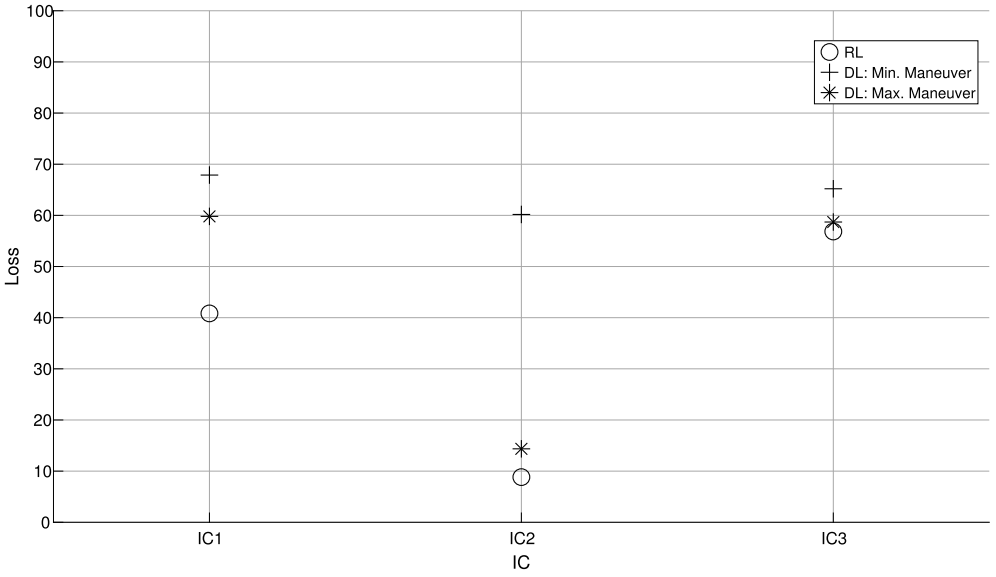


Fig. 8. Loss comparison between RL and DL, for three initial conditions: $IC1$, $IC2$, and $IC3$.

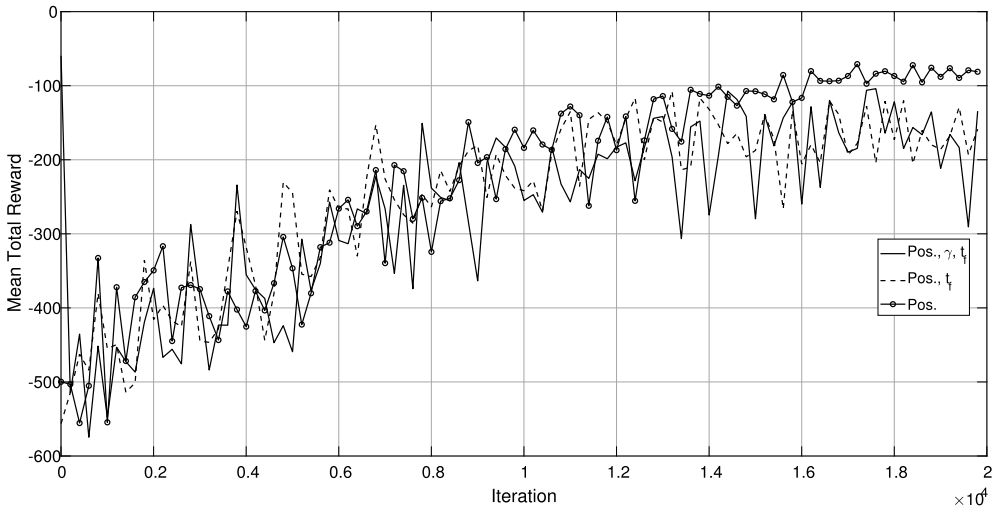


Fig. 9. Mean reward vs. iterations for different selections of states used in the RL algorithm.

Up until now, 11 states were used in the state-space representation of the RL method, as presented in section 3.1. Let us evaluate the contribution of particular states to the learning process, by training the policy using a subset of those states. Fig. 9 presents the mean total reward vs. iteration on a subset of states in comparison to the full 11 states model. The set containing all 11 states consists out of position, path angle, and the expected interception time states and is appropriately named as "$Pos., \gamma, t_f$". The second set consists out of position and the expected interception time states and is named as "$Pos., t_f$". The third set contains only the position states and is named as "$Pos.$".

From Fig. 9 it is clear that the RL method can obtain the close to optimal strategy (under the previously mentioned assumptions), based only on the position of the vehicles. Furthermore, the position only set yielded the best performance (in terms of the mean total reward), suggesting a high compatibility between the number of the states in the "$Pos.$" set, and the DRL parameters (such as the architectures of the neural networks and the RL algorithm parameters).

5. Conclusions

Throughout the course of this paper, a general method for calculating a sub-optimal launch time and a sub-optimal guidance law was presented and evaluated for a scenario in which a target aircraft is protected by a defending missile from an incoming interceptor. The method utilized deep reinforcement learning, thus performing most of the calculations at the offline stage.

The method consists out of training a function approximator in a form of a neural network, using the REINFORCE with baseline algorithm, which belongs to a class of deep reinforcement learning algorithms. As the product of the proposed method, a trained neural network is obtained, which given a particular state of vehicles, returns the required target action, i.e., whether or not to launch the defender, and which maneuver to perform.

The actions obtained by the reinforcement learning method were found to be near optimal, by comparing the results to the ones produced by genetic algorithms and to the lower limit of the non-negative loss. In addition, the investigation of the sensitivity of the results to state selection pointed out the fact that using only the position of the vehicles as the representing states yields better performance the full set of states which consisted of position, path angle, and the expected interception time.

The fact that the presented level of performance can be achieved online, makes reinforcement learning an attractive method in switched system optimization problems that demand online performance. Furthermore, the fact that the proposed approach can be scaled up without a significant addition of computational time at the online application stage, due to the nature of the training/application stages of the method, makes the proposed approach even more suitable for multiple switches case in switched system optimization in general, or in the proposed application, in a multi-agent launch time and maneuver optimization.

The method was exemplified on a perfect information case. If missile's guidance law could unexpectedly change mid-flight, an estimator, such as the multiple model adaptive estimator or the interacting multiple model estimator must be added to the simulation which produces the data set for the neural network. To add robustness against guidance law switches, the proposed strategy can utilize a neural network that is trained on a data set consisting out of scenarios in which such switches occurred. One method to obtain such data set is by simulating scenarios in which missile's guidance law switches randomly from one type to the other (with some constraints on the number of switches). The investigation of this topic is left for future research.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, A brief survey of deep reinforcement learning, arXiv preprint, arXiv:1708.05866, 2017.
[2] R.B. Asher, J.P. Matuszewski, Optimal guidance with maneuvering targets, J. Spacecr. Rockets 11 (1974) 204–206, https://doi.org/10.2514/3.62041.
[3] R. Bellman, On the theory of dynamic programming, Proc. Natl. Acad. Sci. 38 (1952) 716–719.
[4] S.C. Bengea, R.A. DeCarlo, Optimal control of switching systems, Automatica 41 (2005) 11–27.
[5] X.C. Ding, Y. Wardi, M. Egerstedt, On-line optimization of switched-mode dynamical systems, IEEE Trans. Autom. Control 54 (2009) 2266–2271.
[6] M. Egerstedt, Y. Wardi, H. Axelsson, Transition-time optimization for switched-mode dynamical systems, Trans. Autom. Control 51 (2006) 110–115.
[7] M. Egerstedt, Y. Wardi, F. Delmotte, Optimal control of switching times in switched dynamical systems, in: 42nd IEEE Conference on Decision and Control, 2003, pp. 2138–2143.
[8] Z. Feng, K.L. Teo, V. Rehbock, A discrete filled function method for the optimal control of switched systems in discrete time, Optim. Control Appl. Methods 30 (2009) 585–593.

[9] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A.A. Kalyanpur, A. Lally, J.W. Murdock, E. Nyberg, J. Prager, et al., Building Watson: an overview of the deepqa project, AI Mag. 31 (2010) 59–79.
[10] D. Gorges, M. Izák, S. Liu, Optimal control and scheduling of switched systems, IEEE Trans. Autom. Control 56 (2011) 135–140.
[11] S. Gutman, O. Goldan, S. Rubinsky, Guaranteed miss distance in guidance systems with bounded controls and bounded noise, J. Guid. Control Dyn. 35 (2012) 816–823.
[12] S.R. Kumar, T. Shima, Cooperative nonlinear guidance strategies for aircraft defense, J. Guid. Control Dyn. 40 (2017) 124–138, https://doi.org/10.2514/1.G000659.
[13] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, J. Mach. Learn. Res. 17 (2016) 1334–1373.
[14] S. Levine, V. Koltun, Guided policy search, in: International Conference on Machine Learning, 2013, pp. 1–9.
[15] S. Levine, P. Pastor, A. Krizhevsky, D. Quillen, Learning hand-eye coordination for robotic grasping with large-scale data collection, in: International Symposium on Experimental Robotics, Springer, 2016, pp. 173–184.
[16] Y. Li, K. Sun, S. Tong, Observer-based adaptive fuzzy fault-tolerant optimal control for siso nonlinear systems, IEEE Trans. Cybern. 49 (2018) 649–661.
[17] Y. Li, S. Tong, Adaptive neural networks prescribed performance control design for switched interconnected uncertain nonlinear systems, IEEE Trans. Neural Netw. Learn. Syst. 29 (2017) 3059–3068.
[18] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (1943) 115–133.
[19] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A.J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al., Learning to navigate in complex environments, arXiv preprint, arXiv:1611.03673, 2016.
[20] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (2015) 529.
[21] A. Perelman, T. Shima, I. Rusnak, Cooperative differential games strategies for active aircraft protection from a homing missile, J. Guid. Control Dyn. 34 (2011) 761–773, https://doi.org/10.2514/1.51611.
[22] O. Prokopov, T. Shima, Linear quadratic optimal cooperative strategies for active aircraft protection, J. Guid. Control Dyn. 36 (2013) 753–764, https://doi.org/10.2514/1.58531.
[23] A. Ratnoo, T. Shima, Guidance strategies against defended aerial targets, J. Guid. Control Dyn. 35 (2012) 1059–1068, https://doi.org/10.2514/1.56924.
[24] G.A. Rummery, M. Niranjan, On-Line Q-Learning Using Connectionist Systems, vol. 37, University of Cambridge, Department of Engineering, 1994.
[25] I. Rusnak, The lady, the bandits, and the bodyguards - a two team dynamic game, IFAC Proc. Vol. 38 (2005) 441–446, 16th IFAC World Congress.
[26] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International Conference on Machine Learning, 2015, pp. 1889–1897.
[27] V. Shalumov, Online launch-time selection using deep learning in a target–missile–defender engagement, J. Aerosp. Inform. Syst. (2019) 224–236.
[28] V. Shalumov, Optimal cooperative guidance laws in a multiagent target–missile–defender engagement, J. Guid. Control Dyn. (2019) 1–14.
[29] V. Shalumov, T. Shima, Weapon-target-allocation strategies in multiagent target-missile-defender engagement, J. Guid. Control Dyn. 40 (2017) 2452–2464, https://doi.org/10.2514/1.G002598.
[30] T. Shima, Optimal cooperative pursuit and evasion strategies against a homing missile, J. Guid. Control Dyn. 34 (2011) 414–425, https://doi.org/10.2514/1.51765.
[31] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (2016) 484–489.
[32] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: ICML, 2014.
[33] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, 2nd edition, MIT Press, 2018.
[34] V. Turetsky, T. Shima, Target evasion from a missile performing multiple switches in guidance law, J. Guid. Control Dyn. 39 (2016) 2364–2373.
[35] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (1992) 279–292.
[36] S. Wei, K. Uthaichana, M. Žefran, R.A. DeCarlo, S. Bengea, Applications of numerical optimal control to nonlinear hybrid systems, Nonlinear Anal. Hybrid Syst. 1 (2007) 264–279.
[37] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, in: Reinforcement Learning, Springer, 1992, pp. 5–32.
[38] X. Xu, P.J. Antsaklis, A dynamic programming approach for optimal control of switched systems, in: Proceedings of the 39th IEEE Conference on Decision and Control, 2000, pp. 1822–1827.
[39] X. Xu, P.J. Antsaklis, On time optimal control of integrator switched systems with state constraints, Nonlinear Anal. 62 (2005) 1453–1465.
[40] W. Zhang, A. Abate, J. Hu, Efficient suboptimal solutions of switched lqr problems, in: American Control Conference, IEEE, 2009, pp. 1084–1091.
[41] W. Zhang, J. Hu, Optimal quadratic regulation for discrete-time switched linear systems: a numerical approach, in: American Control Conference, IEEE, 2008, pp. 4615–4620.
[42] F. Zhu, P.J. Antsaklis, Optimal control of hybrid switched systems: a brief survey, Discrete Event Dyn. Syst. 25 (2015) 345–364, https://doi.org/10.1007/s10626-014-0187-5.