A Data-Driven Guidance Strategy for Target Interception with Terminal Time Constraint

Zichao Liu[a,*], Ziyi Wu[a,*], Toshiharu Tabuchi[b], Nguyen Hung[b], Cristino DSouza[b], Chang-Hun Lee[c], Shaoming He[a,**]

[a]*School of Aerospace Engineering, Beijing Institute of Technology, Beijing 100081, China*
[b]*Autonomous Robotics Research Centre, Technology Innovation Institute, Abu Dhabi, United Arab Emirates*
[c]*Department of Aerospace Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, Korea*

Abstract

This paper proposes a data-driven guidance strategy to tackle the target interception problem with terminal time constraints. The developed guidance algorithm employs the prediction-correction concept to derive commands effectively. We first develop a two-branch neural network to predict time-to-go, assuming there is no constraints on target's maneuvers. Using the predicted time-to-go, we then propose a biased proportional navigation guidance (PNG), where the the biased command is computed by a deep reinforcement learning (DRL) network, to correct the impact time. Results from intensive Monte-Carlo simulations validates the efficacy of the method.

Keywords: Impact time guidance, Computational guidance, Maneuvering target, Deep reinforcement learning

1. Introduction

Irecent years, the increasing defense capabilities of aircraft platforms brings challenges to the interception of a target using a single interceptor. Consequently, there has been a growing research focus on cooperative guidance for interceptors, particularly in the context of multi-missile salvo attack that introduces terminal time constraints into guidance law design. The ability to control the exact time of impact allows forces to synchronize attacks, even when deploying munitions from multiple platforms over varied distances. This synchronization can maximize operational effectiveness, enabling simultaneous strikes that overwhelm enemy defenses. Moreover, controlling the impact time aids in minimizing collateral damage by allowing munitions to detonate at precise moments to achieve the intended tactical outcome while preserving surrounding infrastructure and reducing non-combatant casualties. In general, there are two ways to implementing impact time control in cooperative guidance strategies: controlling time-to-go individually through impact

[*]These authors contributed equally.
[**]Corresponding author
Email address: `shaoming.he@bit.edu.cn` (Shaoming He)

time control guidance (ITCG) laws [1, 2, 3, 4, 5, 6, 7, 8], or receiving information from neighbor missiles to regulate time-to-go cooperatively [9, 10, 11].

The terminal time-constraint was first introduced into guidance laws in [12], and later many different approaches were reported on developing the ITCG laws. To achieve the desired impact time, the authors in [13] designed a generalized ITCG based on PNG, where the impact time error feedback term in [13] was introduced to control the impact time. While earlier studies focused on intercepting stationary targets, the work in [14] extended the ITCG law to the scenery of intercepting non-maneuvering targets. In the interception of maneuvering targets, uncertainties in target behavior present significant challenges for the derivation of ITCG. With constant speed assumption, a sliding-model control form ITCG law was proposed based on the explicit time-to-go estimation for a deviated pursuit-guided interceptor [15, 2]. To mitigate the effect of varying speed, the authors in [16, 6] proposed to improve the impact time prediction by numerical integration time at the price of increasing computational burden.

To diminish the impact of target uncertainty, several cooperative guidance methods utilize inter-missile communication networks to incorporate the information of other interceptors. The work in [9] developed a cooperative proportional guidance law to cooperatively intercept targets, when each missile could obtain the time-to-go of all other missiles. Another cooperative guidance law discussed in [10] required communication networks between neighbor interceptors, which rely on the consistency theory. However, the utilization of inter-missile communication networks may also lead to the emergence of new problems, such as network delay and vulnerability, hence indicating the value of independent estimation of time-to-go by the interceptor.

Notice that the aforementioned analytical ITCG methods that rely on closed-form or numerical time-to-go prediction may suffer from accuracy issues in two aspects. On one hand, both closed-form or numerical time-to-go prediction approaches cannot consider unknown target maneuver and hence the accuracy might degrade. On the other hand, analytical guidance law requires constant speed assumption in command derivation, which cannot cater for fast-moving vehicles. With deep learning technology performing excellently in numerous fields, researchers begin to investigate its application in guidance strategies to address the challenges associated with analytic methods [17, 18, 19, 20, 21]. Unlike analytical guidance laws, deep learning techniques rely on data-driven approaches rather than principles of dynamics, allowing for the handling of interception problems without significant simplification of dynamics. The work in [21] used deep neural networks to improve prediction-correction guidance, resulting in a significant reduction of computing power consumption. The problem of maneuvering penetration was modeled as a Markov decision process (MDP) in [22] and employed the DRL to achieve model-free penetration. In a similar vein, the DRL was utilized in [23] to optimize energy consumption and shorten interception time. Another approach proposed in [24] used DRL to develop a guidance strategy that could intercept maneuvering targets using only angle measurement information. The work in the previous work in [25] employed the proximal policy optimization (PPO) algorithm to achieve maneuvering near-space target interception. Additionally, the authors in

[26] designed a DRL strategy for defending an attacking missile, which makes decisions on the timing of launching a defense missile and the maneuvering strategy of an evading target. In the context of ITCG, deep learning can can be applied as a time-to-go predictor based on the prediction-correction framework [27], and as a part of the ITCG law in guidance gain tuning [28]. However, current predictors lack the ability to predict the time-to-go accurately for maneuvering targets due to the absence of target information [27]. Moreover, although the strategy in [28] determined the parameters in the ITCG law using DRL, they still used an analytical guidance law based on the constant speed assumption, and hence the performance significantly degrades when considering aerodynamic forces.

Taking inspiration from previous research, this study introduces a computational impact-time-control guidance (CITCG) algorithm for intercepting maneuvering targets in consideration varying-speed aerodynamic models. Since impact time control guidance is a general control problem with terminal constraint, one simple strategy of utilizing DRL to solve this kind of problem is to formulate a reward function that gives positive value when the missile intercepts the target with desired terminal time and negative values if it fails. However, the issue with this naive formulation is that the agent explores the action space randomly during the training phase, resulting sparse reward of this problem since the probability of successfully intercepting the target with specified impact time is very low under random guidance commands. This can be also attributed to the fact that the reward function is formulated based on current state and cannot evaluate the terminal state. Since reinforcement learning is known to be inefficient to cope with this type of problem due to sparse reward [23], we develop a computational prediction-correction approach for guidance algorithm design: this guidance algorithm employs DRL to generate guidance commands, and deep supervised learning is utilized to predict the time-to-go necessary for generating the commands. Extensive numerical simulations reveal that the proposed guidance algorithm is capable to satisfy the impact constraint with different types of target maneuvers and significantly improves the interception performance compared to conventional guidance laws.

The contributions of this paper are twofold. On one hand, a two-branch neural network is developed to precisely predict the time-to-go of intercepting maneuvering targets by processing the state of interceptor and the target trajectory sequence simultaneously. On the other hand, the impact time constraint is satisfied through the model-free DRL with aerodynamic effect and target maneuver. Since the prediction considers time-varying aerodynamics and infers the influence of target maneuver, the prediction can be significantly improved compared with using constant-speed model and thereby helps to improve the impact time error.

The later sections of this paper are organized as follows: Section 2 proposes an overview of the background and preliminary information related to the proposed research. In section 3, the computational ITCG algorithm proposed in this study is introduced, which includes the predictor that utilizes a two-branch neural network and the corrector that employs PPO algorithm. Section 4 presents the numerical simulation results and discussion. Finally, Section 5 provides the conclusion of this article.

2. Backgrounds and preliminaries

We now present the interceptor-target engagement dynamics models and then state the main problem addressed in the paper.

2.1. Nonlinear Mathematical Models

For simplicity, we consider a 2D interception scenario and the interceptor-target engagement geometry in the vertical plane is depicted in Fig. 1. The line of sight (LOS) angle and relative range are defined as λ and R, respectively. The subscript I denotes the interceptor, while the subscript T denotes the target. The interceptor's flight path angle and speed are expressed by θ_I and V_I, and the gravity, lift, and drag forces are expressed by F_G, F_L, and F_D. The symbol V_T, θ_T, and a_T stand for the target's moving speed, flight path angle, and maneuvering acceleration. Since we have no information on the target maneuver, we

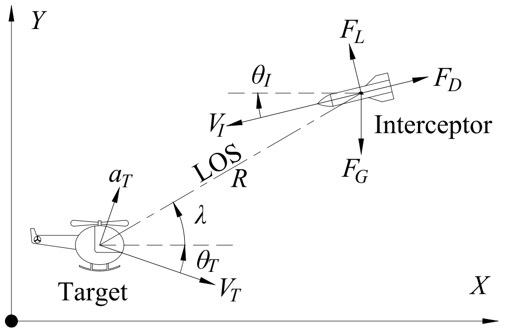


Figure 1: Interceptor-target engagement geometry.

suppose the target escapes at a constant speed, then the dynamics and kinematics of the engagement can be described as in [29] as follows

$$\begin{cases} \dot{V}_I = \dfrac{F_D - F_G \sin\theta_I}{m} \\ \dot{\theta}_I = \dfrac{F_L - F_G \cos\theta_I}{mV_I} \\ \dot{X}_I = V_I \cos\theta_I \\ \dot{Y}_I = V_I \sin\theta_I \\ \dot{V}_T = 0 \\ \dot{\theta}_T = \dfrac{a_T}{V_T} \\ \dot{X}_T = V_T \cos\theta_T \\ \dot{Y}_T = V_T \sin\theta_T \end{cases} \tag{1}$$

where the forces on the interceptor are given by

$$\begin{cases} F_G = mg \\ F_L = C_L Q S \\ F_D = C_D Q S \end{cases} \tag{2}$$

where m and S are the mass and the reference area of the interceptor. Above, the symbol C_L and C_D are the lift and drag coefficients, respectively. The gravitational acceleration is denoted by notation g. The symbol Q stands for the dynamic pressure that can be defined as

$$Q = \frac{\rho V_I^2}{2} \tag{3}$$

where ρ represents the standard air density, which varies with respect to the height.

According to the relationship between aerodynamic coefficient and angle-of-attack (AOA), an approximation can be made as follows

$$\begin{cases} C_L = C_L^{\alpha} \alpha \\ C_D = C_{D0} + C_{D0}^{\alpha^2} \alpha^2 \end{cases} \tag{4}$$

where α denotes the AoA, and it is related to the interceptor acceleration a_I by

$$\alpha = \frac{m a_I}{C_L^{\alpha} Q S} \tag{5}$$

To achieve salvo interception, all interceptors are required to impact the target at the same time. We use $t_\mathrm{d}, t_\mathrm{go}, t_f$ to denote the desired interception time, time-to-go, and terminal time, respectively, and the goal of impact time control is to adjust t_f to be equal to t_d. Since $t_f = t_\mathrm{go} + t$, the t_f control problem can be converted into t_go control problem. Also, we assume the moving speed of the missile is faster than that of the target, i.e., $V_I > V_T$, to make the ITCG solution feasible. To address this issue, we define the ITCG problem as follows:

Problem 1: Given the target-intercepter engagement model described by Eqs. (1)-(5), design the lateral acceleration command a_I that ensures intercepting a maneuvering target at the desired interception time t_d.

2.2. Algorithm Overview

Notice that designing guidance law for impact time is generally a control problem with hard terminal constraint. To address this problem, we propose to leverage the prediction-correction concept to address the impact time control problem. Specifically, a neural predictor is used to accurately estimate the remaining interception time under PNG considering varying aerodynamics and unknown target maneuver. A DRL corrector is developed to regulate the impact time error. Based on this concept, the command of the proposed method is designed as a biased PNG, i.e.,

$$a_I = a_0 + a_b \tag{6}$$

where a_b is biased command for diminishing the impact time error, and a_0 is the PNG command with gravity compensation term, which is given by [29]

$$a_0 = 3V_I\dot{\lambda} + g\cos\theta_I \tag{7}$$

The working principle of the proposed guidance algorithm is illustrated in Fig. 2, where the variable $\hat{t}_{\mathrm{go}}$ denotes the estimated time-to-go and $\varepsilon_t = t_{\mathrm{d}} - t - \hat{t}_{\mathrm{go}}$ is the impact time error. In this work, we first propose a predictor by a novel two-branch neural network, capable of inferring the unknown target maneuver by processing the time-history data, to cater for maneuvering targets interception. The biased guidance term a_b is then generated by a computational DRL corrector to guide the missile reach the target at predetermined intercept time. This biased command is learned by the widely-accepted PPO with properly designed reward functions.

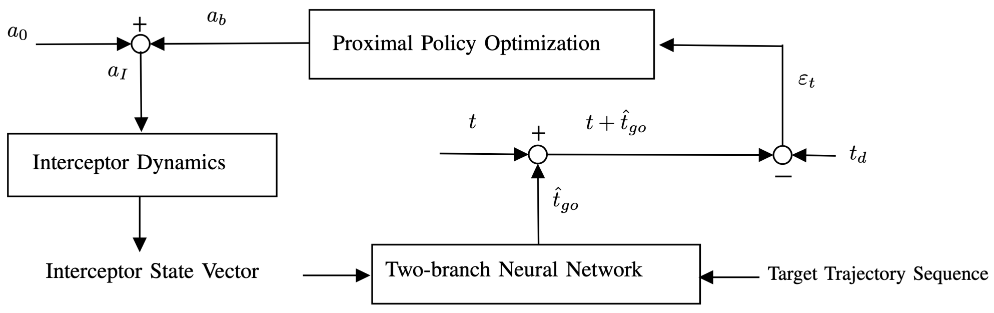


Figure 2: Closed-loop guidance system under the proposed guidance algorithm.

Remark 1. *Notice that applying DRL in guidance law design with terminal constraint usually encounters the issue of sparse reward since the possibility of generating the command, with randomly explored actions, to intercept the target at a predetermined time constant is very low. To overcome this technical challenge, we transform the terminal constraint into a state regulation problem by the prediction-correction concept.*

3. Computational Terminal Time-Constrained Guidance Design

This section introduces a computational ITCG method against maneuvering targets. We first present a two-branch neural network to predict t_{go}, and then introduce a DRL method to compute the biased command a_b.

3.1. Predictor of Time-to-Go

Notice that accurate prediction of the time-to-go is important for impact time control and finding analytic solution is generally intractable when considering aerodynamic effect and unknown target maneuver. To simplify the problem, many existing works usually assume the heading angle is small, the interceptor is constant-moving and the target is non-maneuvering to enable a closed-form t_{go} estimation [16]. However, these assumptions might be violated for maneuvering target interception and hence result in decreased performance. Generally, if the target maneuver is available, accurate time-to-go can be found by numerically solving Eq. (1) as

$$t_{\text{go}} = f\left(V_I, \theta_I, R, \lambda, a_T\right) \tag{8}$$

where the relative range R and LOS angle λ are functions of the geometrical states as

$$\begin{cases} R = \sqrt{\left(X_I - X_T\right)^2 + \left(Y_I - Y_T\right)^2} \\ \tan\lambda = \dfrac{Y_I - Y_T}{X_I - X_T} \end{cases} \tag{9}$$

Because numerically solving Eq. (1) is computationally-expensive, we can leverage deep neural network to learn the nonlinear mapping, as given in Eq. (8). However, the main challenge is that the target maneuver is naturally unknown to the interceptor. To address this issue, we propose to indirectly infer the target maneuver by using historical target trajectory data since the target trajectory is influenced by its maneuvers. Then, the target maneuver at the kth time instant can be inferred using n consequential time-history data as

$$a_T = f\left(X_{T,k-n+1}, Y_{T,k-n+1}, \cdots, X_{T,k}, Y_{T,k}\right) \tag{10}$$

where $(X_{T,k}, Y_{T,k})$ denotes the inertial position of the target at the kth time instant.

Also, we notice that the same target trajectory and relative engagement provide similar target maneuver information, then all position information is shifted to a dynamic coordinate system with the target position as the origin to infer the target maneuver as

$$a_T = f\left(X_{T,k-n+1} - X_{T,k}, Y_{T,k-n+1} - Y_{T,k}, \cdots, 0, 0\right) \tag{11}$$

Since the relative range and bearing angle can be directly measured from the onboard seeker and the position of the interceptor can also be obtained from the onboard sensor, the historical target trajectory data $(X_{T,k}, Y_{T,k})$ is available to the interceptor. Consequently, nonlinear mapping, as shown in Eq. (8), can be transformed into

$$t_{\text{go}} = f\left(V_{I,k}, \theta_{I,k}, R_k, \lambda_k, T_k\right) \tag{12}$$

where

$$T_k = \{(X_{T,k-n+1} - X_{T,k}, Y_{T,k-n+1} - Y_{T,k}), \ldots, (0,0)\} \tag{13}$$

Notice that Eq. (12) contains both sequential and non-sequential inputs, directly stack these inputs into a single vector and feed to a conventional deep neural network might result in the explosion of inputs. To mitigate this issue and improve the training efficiency, we propose a novel two-branch neural network architecture that effectively combines two neural networks into a unified network to learn the unknown nonlinear mapping function represented by Eq. (12). The proposed network simultaneously processes both sequential and non-sequential inputs and directly predicts t_{go} without relying on prior knowledge of the target flight state. Furthermore, this network significantly reduces the training time by combining the training process of two neural networks. The architecture of the two-branch neural network is presented in Fig. 3.

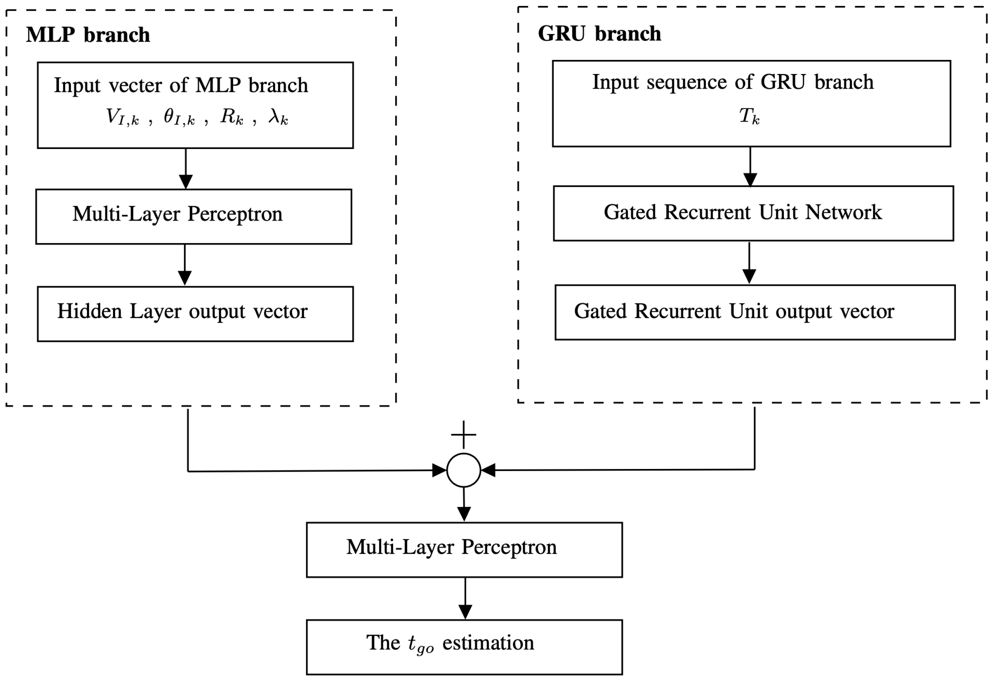

MLP branch
Input vecter of MLP branch
$V_{I,k}$, $\theta_{I,k}$, R_k , λ_k
Multi-Layer Perceptron
Hidden Layer output vector
GRU branch
Input sequence of GRU branch
T_k
Gated Recurrent Unit Network
Gated Recurrent Unit output vector
+
Multi-Layer Perceptron
The t_{go} estimation

Figure 3: The architecture of two-branch neural network.

The proposed neural network architecture comprises two input branches. The first branch, a multi-layer perceptron (MLP), receives a vector containing information about the interceptor's flight state and the relative motion. The second branch, a gated recurrent unit (GRU), receives a sequence of historical target positions sampled at intervals of 0.1 seconds. The intermediate vectors generated from the two branches share the same shape. The addition of these two vectors produces a combined vector that is input into an MLP, which then outputs the t_{go} estimation. The MLPs in the network consist of two fully-connected hidden layers, each layer containing 100 neurons, while the GRU network comprises 10 cells, each with 100 neurons. The two-branch neural network is trained with data collected from Monte Carlo flight simulation experiments using aerodynamic models. During the initialization of the simulation, the original position and velocity of the interceptor, as well as the maneuver acceleration and velocity of the target, are randomly assigned. Once the sample acquisition is completed, the two-branch neural network is trained using the collected data.

Assuming the network parameter is represented as β, the conventional loss function is utilized in the training process as

$$J(\beta) = \frac{1}{N_D} \sum_{i=1}^{N_D} \left(\hat{t}_{\text{go},i} - t_{\text{go},i}\right)^2 \tag{14}$$

where the network's estimate of t_{go} is represented by $\hat{t}_{\text{go},i}$, while $t_{\text{go},i}$ expresses the real t_{go}, and N_D denotes the number of samples. The stochastic gradient descent algorithm is utilized to optimize the loss function. The update formula for the network parameters can be expressed as follows:

$$\beta_{\text{new}} = \beta_{\text{old}} - \alpha_\beta \nabla_\beta J(\beta) \tag{15}$$

where α_β represents the learning rate.

Remark 2. *The authors in [30] derived the closed-form solution of time-to-go for PNG and the expression is given by*

$$\hat{t}_{\text{go}} = \frac{R}{V_I} \sum_{i=0}^{M} \frac{(2i)!}{2^{2i}\left[1 + 2i(N-1)\right](i!)^2} \tag{16}$$

where the ideal analytic solution is achieved by setting $M = \infty$, indicating that the accuracy of time-to-go computation is improved as M increases in Eq. (16)*, especially when the lead angle $\sigma = \theta_I - \lambda$ is big.*

Although Eq. (16) *provides a closed-from time-to-go estimation under PNG, the main issue lies that the derivation ignores the varying speed (i.e., no aerodynamic forces) and target maneuver. Hence, the prediction accuracy will significantly degrade when considering realistic aerodynamic dynamics and maneuvering targets, as illustrated in the simulation sections.*

Remark 3. *Compared with numerical impact time prediction proposed in [16, 6], the proposed neural prediction is expected to improve the accuracy in two aspects. On the one hand, we leverage the time history of the target measurable states to indirectly consider the target maneuver in the prediction process. On the other hand, the proposed neural predictor can be easily implemented online once the network is properly trained, whereas the numerical prediction requires more powerful computational resources.*

3.2. Corrector of Impact Time Error

Once the estimation of the t_{go} is accomplished, the next step is to design the guidance command to regulating the t_{go} error. Since the dynamics of the learning-based t_{go} estimation is unknown, deriving the analytical form of ITCG laws is intractable. To this end, this paper proposes a computational guidance command that is based on a state-of-the-art DRL algorithm, called PPO. In the reinforcement learning paradigm, the controller is regarded as an agent, and the controlled object as the environment, where the agent receives observations from the environment, interacts with it by taking actions, and receives rewards. By maximizing the cumulative reward under the present observation, the agent learns to optimize its behavior through trial and error. The PPO algorithm employs two neural networks: an actor network, which outputs guidance commands based on the input observations, and a critic network, which evaluates the quality of these commands.

3.2.1. Reinforcement Learning Formulation of the ITCG Problem

The problem of DRL is commonly formulated as a MDP, which can be described as a tuple consisting of several sets: states $\mathcal{S}$, observations $\mathcal{O}$, actions $\mathcal{A}$, state transition probabilities $\mathcal{P}$, and rewards $\mathcal{R}$. At time t, the environment is in state s_t, the agent obtains observation o_t from the internal state s_t, takes action a_t, and the environment transits to state s_{t+1} according to the state transition probabilities $\mathcal{P}$ and provides a reward r_t. Prior to applying reinforcement learning to solve the ITCG problem, the observation $\mathcal{O}$, action $\mathcal{A}$, and reward $\mathcal{R}$ need to be designed. Obviously, the action can be determined as the bias command a_b. During the trial-and-error training stage, the bias command may become excessively large, and hence it needs to be clipped as

$$a_t = a_b, \quad \|a_b\| \leq a_{\max} \tag{17}$$

where $a_{\max}$ represents the limit value of the bias command.

The observation can be designed as a vector consisting of flight speed and impact time error, as larger errors in impact time and higher speed require larger a_b. Therefore, we have

$$o_t = (k_V V_I, k_e \varepsilon_t) \tag{18}$$

where k_V and k_e are the scaling factors to transform the observation elements into similar scales.

The reward function plays a major role in determining the efficacy and convergence of the reinforcement learning process. In this context, we have considered several factors, such as impact time error, energy consumption, speed, and altitude in designing the reward function for reinforcement learning. Recent research suggested the optimal error dynamics for general guidance law design [31], represented as

$$\dot{\varepsilon}_t + \frac{k}{t_{\text{go}}}\varepsilon_t = 0 \tag{19}$$

where $k > 0$ is a design parameter to adjust the convergence speed of ε_t.

It has been shown in [31] that if the error satisfies the optimal error dynamics given by Eq. (19), the error will converge to zero once the target position is reached. Inspired by this fact, we propose the following reward function for ε_t as

$$r_i = e^{-\left(\frac{\varepsilon_t}{t_{\text{go}}}\right)^2} \tag{20}$$

which indicates that when the DRL tries to maximize the reward, more penalty is posed on the impact error as $t_{go} \to 0$. This helps to gradually reduce the impact time error when the missile is close to the target.

In missile guidance, the energy consumption, i.e., integral of the square of the acceleration $\int_{t_0}^{t_f} a_I^2 dt$, is an important factor since it is directly related to the induced drag. Hence, minimizing the square of the lateral acceleration is helpful to minimize the velocity loss. Notice that the evaluation of the potential value of observations involves the integration of the rewards, the effect of energy consumption can be considered in reward design as

$$r_e = e^{-a_I^2} \tag{21}$$

Speed is also an important factor to consider when designing the reward function. In particular, additional maneuvers could cause a loss of speed, which might result in an extended t_{go}, and ultimately, missing t_{d}. Moreover, a decrease in speed could decrease the likelihood of interception. To overcome this issue, we propose the reward term for speed in the following form

$$r_V = e^{-k_V V_I} \tag{22}$$

where k_V is the scaling factor, which has the same value as in Eq. (18).

Finally, excessive loss of altitude during maneuvers could result in an increased atmospheric density at low altitude, further accelerating the loss of flight speed. To mitigate this issue, we design the reward term to minimize the loss of altitude during maneuvers as

$$r_Y = e^{-k_Y (Y_I - R)} \tag{23}$$

where k_Y is the altitude scaling factor.

After summing the weighted reward functions, the reward function that considers all factors is shown as

$$r_t = \sigma_1 r_i + \sigma_2 r_e + \sigma_3 r_V + \sigma_4 r_Y - \left[|R_f| + (t_{\mathrm{d}} - t_f)^2 \right] \tag{24}$$

where $\sigma_1, \sigma_2, \sigma_3$ and σ_4 are weight coefficients of the rewards. The symbol R_f represents the terminal miss distance between the missile and the target. Note that the term $\left[|R_f| + (t_d - t_f)^2 \right]$ in the reward function is utilized to improve the guidance accuracy.

3.2.2. Proximal Policy Optimization Algorithm

Reinforcement learning aims to enable artificial agents to make a sequence of decisions within a given environment, with the ultimate goal of maximizing a cumulative reward, which is the accumulation of rewards obtained by the agent over time. To achieve this, the PPO algorithm was introduced by OpenAI [32, 33], which is an RL technique that uses an actor-critic architecture for policy learning. The structure of typical PPO is illustrated in Fig. 4.

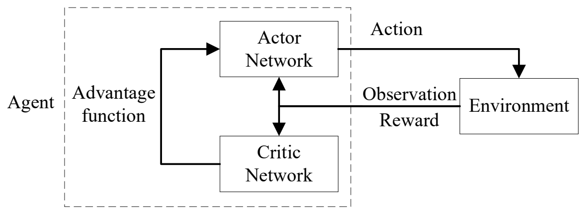


Figure 4: The information flow of PPO.

In the PPO, the actor network plays an essential role in the selection of actions. The actor network takes o_t as input, outputs a probability distribution of actions, commonly referred to as the policy $\pi(a_t|o_t)$. Subsequently, $\pi(a_t|o_t)$ is sampled to determine the actual action that the agent will take. The actor network in the PPO is often combined with a critic network, which estimates the value function $v_\pi(o_t)$ of each observation. The critic network serves as a vital component of the actor-critic architecture by providing a baseline for the advantage function and reducing the variance of the policy gradients. The advantage function $\hat{A}_\pi$ is a metric that quantifies the gap between the estimated value of a state and the expected value if a specific action is taken, enabling the PPO algorithm to update the actor network accordingly. To train the agent, the PPO relies on the samples generated through the interaction between agent and environment. As an on-policy algorithm, the PPO requires that the policy employed to generate samples be consistent with the recently updated policy of the agent. When training the PPO, the agent utilizes the old policy $\pi_{\text{old}}(a_t|o_t)$ to interact with the environment N times and subsequently calculates $\hat{A}_\pi^{(N)}$ using the generalized advantage estimator

$$\hat{A}_{\pi t}^{(N)} = \sum_{l=0}^{N-1} \gamma^l \delta_{t+l}^v, \quad \delta_t^v = -v_\pi(o_t) + r_t + \gamma v_\pi(o_{t+1}) \tag{25}$$

where δ_t^v is the temporal difference residual, which represents the gap between the actual value and the predicted value of a state. The discount factor $\gamma \in (0, 1]$ serves as a parameter that determines the significance of immediate versus future rewards in the decision-making process of the agent.

At this point, the critic network can be trained using the loss function determined by

$$J(\xi) = \frac{1}{N} \sum_{t=1}^{N} \|A_\pi(o_t, a_t)\|^2 \tag{26}$$

where the parameters of the critic network are represented by ξ.

Due to the lack of knowledge about the optimal action sequence, it is difficult to calculate the error between optimal action and executed action to construct a loss function for the action network. As a solution, the advantage function is employed to calculate the advantage of a given action over the average action that can be taken from a given state. The surrogate loss, which quantifies the amount of policy change in each iteration of the algorithm, is then calculated using the advantage function. To improve the policy while not diverging too much from the previous policy, a `clip` function is introduced to impose a constraint on how much the policy can change in each iteration, which is controlled by a hyperparameter called the clipping parameter. In summary, the loss function in training the actor network can be determined as

$$J(\beta) = \frac{1}{N}\sum_{t=1}^{N}[\min(c_t(\beta)A_\pi, \texttt{clip}(c_t(\beta), c_\vartheta)A_\pi)] \tag{27}$$

where c_ϑ is the clipping parameter, β represents the parameters of the actor network, $c_t(\beta)$ is the ratio function to measure the magnitude of policy updates during the training session. The `clip` function and the ratio function are determined by

$$\texttt{clip}(r_t(\beta), c_\vartheta) = \begin{cases} 1 - c_\vartheta, & c_t(\beta) < 1 - c_\vartheta \\ 1 + c_\vartheta, & c_t(\beta) > 1 + c_\vartheta \\ c_t(\beta), & \text{others} \end{cases} \tag{28}$$

$$c_t(\beta) = \frac{\pi_{\text{new}}(a_t \mid o_t)}{\pi_{\text{old}}(a_t \mid o_t)} \tag{29}$$

With the objective functions, shown in Eqs. (26) and (27), the network parameters ξ and β are then updated by the stochastic gradient descent (SGD) optimizer. The update formula can be expressed as follows:

$$\xi_{\text{new}} = \xi_{\text{old}} - \alpha_\xi \nabla_\xi J(\xi) \tag{30}$$

$$\beta_{\text{new}} = \beta_{\text{old}} - \alpha_\beta \nabla_\beta J(\beta) \tag{31}$$

where α_ξ and α_β are positive learning rates.

Remark 4. *Notice that the leveraging DRL to generate the biased guidance command is a kind of 'blackbox' algorithm that provides no closed-form solution and hence it is difficult to directly analyze the characteristics and performance as conventional guidance laws. However, it has been theoretically proven in [34] that PPO provides convergence to the local minima under some mild assumptions. This indicates that the convergence of the developed learning-type guidance algorithm to the optimal solution can be theoretically ensured.*

Remark 5. *Unlike the original PPO algorithm, we employ a deep neural network to evaluate the terminal state in the actor network and this strategy helps to transform the original problem with terminal constraint into a state regulating problem. Therefore, we can directly design a proper reward function to minimize the state regulation error and resolve the typical sparse reward issue.*

4. Simulation Results

We conduct simulation experiments in the longitudinal plane to evaluate the performance of the computational terminal time-constrained guidance algorithm. First, the accuracy of the t_{go} predictor is investigated. Second, the efficacy of the PPO-based corrector is evaluated via a series of simulation experiments. Finally, Monte Carlo simulations are executed to perform statistical analysis of the impact time errors.

4.1. Performance of Predictor

Proper training of the predictor is essential prior to its application. To this end, numerous simulations with an interceptor attacking a maneuvering target under PNG are conducted to generate the training samples for the predictor. The initial values of the target and the interceptor are given in Table 1, which encompass the initial position, speed, flight path angle, and maneuver amplitude. The aerodynamic characteristics of the interceptor are summarized in Table 2 for specific certain Mach numbers, and the first-order interpolation is utilized to obtain the aerodynamic coefficients at other Mach numbers. It should be noted that we suppose the target escapes at a constant speed and thus do not consider the aerodynamic force on the target. The gravitational acceleration is 9.81m/s^2.

Table 1: The initial values of the target and the interceptor.

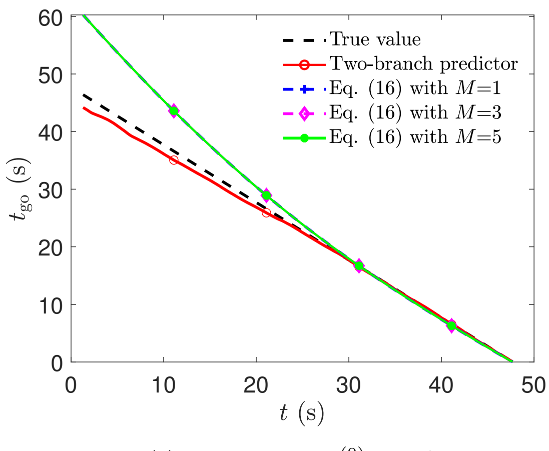
Variable	Definition	Interval or Value	Units
$X_T^{(0)}$	Initial target position in $x-$axis	0	km
$Y_T^{(0)}$	Initial target position in $y-$axis	5	km
$V_T^{(0)}$	Initial target speed	(100, 200)	m/s
$\theta_T^{(0)}$	Initial flight path angle of the target	(0, 90)	°
$a_T^{(0)}$	Initial maneuver of the target	(-2, 2)	g
$X_I^{(0)}$	Initial interceptor position in $x-$axis	(0, 20)	km
$Y_I^{(0)}$	Initial interceptor position in $y-$axis	(10, 30)	km
$V_I^{(0)}$	Initial velocity of the interceptor	(400, 500)	m/s
$\theta_I^{(0)}$	Initial flight path angle of the interceptor	180	°
m	The mass of the interceptor	200	kg
S	The reference area of the interceptor	0.0572556	m^2

We collect 30,000 trajectories, including 7,500,000 samples for training the predictor. Each sample is composed of the input $(V_I, \theta_I, X_I - X_T, Y_I - Y_T, R, \lambda, T_t)$ and output t_{go} pair of the network. These samples are categorized into two sets: the training set, which comprises 80% of the samples, and the validation set, which includes the remaining 20%, using the 80/20 rule. The learning rate α_β is set to be 0.001.

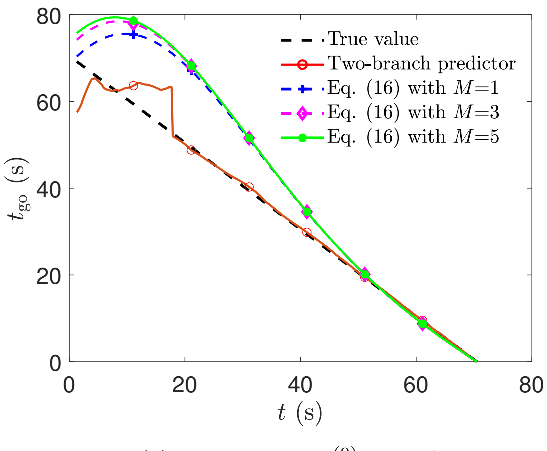
We first evaluate the performance of the predictor and compare with analytic solution (16) through a single experiment. The initial values are set as $X_I^{(0)} = 20$km, $Y_I^{(0)} = 20$km, $V_I^{(0)} = 400$m/s, $\theta_I^{(0)} = -150°$ or $-225°$, $X_T^{(0)} = 0$km, $Y_T^{(0)} = 5$km, $V_T^{(0)} = 100$m/s, $\theta_I^{(0)} = 0°$, and $a_T^{(0)} = 0.5g$. The scenario with $\theta_I^{(0)} = 210°$ corresponds to small initial lead angle $\sigma^{(0)} \approx -7°$ while $\theta_I^{(0)} = 135°$ corresponds to large initial lead angle $\sigma^{(0)} \approx -82°$. Notice that the proposed predictor is trained using samples drawn from Table 1 and the scenarios $\theta_I^{(0)} = -150°$ or $-225°$ are not included in the training set. Hence, we can evaluate the generalization performance by using these two test scenarios. The predictor is then employed to generate

Table 2: Aerodynamic model in the considered scenario.

Mach Number	C_L^{α}/rad^{-1}	C_{D0}	$C_D^{\alpha^2}/rad^{-2}$
0.4	39.056	0.4604	39.072
0.6	39.468	0.4635	39.242
0.8	40.801	0.4682	40.351
0.9	41.372	0.4776	41.735
1.0	41.878	0.4804	43.014
1.1	42.468	0.4797	42.801
1.2	41.531	0.4784	42.656
1.3	41.224	0.4771	42.593
1.4	40.732	0.4768	42.442
1.5	40.321	0.4761	42.218
1.6	40.033	0.4756	42.034
1.7	39.912	0.4751	41.977
1.8	39.756	0.4748	41.893
1.9	39.501	0.4743	41.808
2.0	39.344	0.4739	41.754



(a) Scenario with $\sigma^{(0)} \approx -7°$



(b) Scenario with $\sigma^{(0)} \approx -82°$

Figure 5: Time-to-go estimate comparisons.

real-time estimations of $\hat{t}_{\rm go}$, and the actual $t_{\rm go}$. The results are presented in Fig. 5, which reveals that the estimated values and actual values almost completely overlap in the graph, suggesting excellent performance by the predictor. It is observed that the initial phase of the trajectory exhibits substantial errors, but as

Table 3: Statistical comparison of time-to-go prediction errors.

Algorithm	RMSE	Mean	Maximum
CITCG	1.037	0.653	4.441
Eq. (16) with $M = 1$	6.018	4.320	11.736
Eq. (16) with $M = 3$	6.023	4.324	11.742
Eq. (16) with $M = 5$	6.023	4.324	11.742

the interceptor approaches the target, the prediction error gradually diminishes. The results also indicate that the analytic solution (16) cannot provide accurate time-to-go estimation in realistic scenarios. Even by increasing the order n we cannot improve the estimation performance. The reason behind is that the analytic solution is derived based on non-maneuvering target and hence the estimation performance will degrade if target performs unknown maneuvers. We then assess the statistic performance of the predictor using the test set as shown in Table 3, which clearly indicates that the proposed predictor provides significantly improved performance as the analytical solution.

4.2. Performance of Corrector

The scenarios for training PPO utilized the same initial values as described in Table 1. In each episode, t_d is set as the initial predicted terminal time plus a time randomly chosen between 5s and 10s. The hyperparameters used in the PPO training are presented in Table 4. It is noteworthy that the effect of PPO is impacted by the tuning of hyperparameters. Thus we conducted several trial and error tests to fine-tune these hyperparameters for our guidance problem. After running the training process of the PPO algorithm for 150 episodes, it is observed that the reward value reached a state of stabilization. In order to calculate the average reward of multiple flight experiments, a sliding window average is employed. The learning curve during the PPO training process is illustrated in Fig. 6. The figure depicts that the proposed corrector achieves a stabilized average reward within just 100 episodes.

To evaluate robustness of the corrector, we conduct a series of simulations across various scenarios. The initial flight states are determined as following:

$$\begin{gathered} X_I^{(0)} = 20\text{km}, Y_I^{(0)} = 20\text{km}, V_I^{(0)} = 400\text{m/s}, \theta_I^{(0)} = 210^\circ, \\ X_T^{(0)} = 0\text{km}, Y_T^{(0)} = 5\text{km}, V_T^{(0)} = 100\text{m/s}, \theta_T^{(0)} = 0^\circ \end{gathered} \tag{32}$$

In Fig. 7, the results of simulations involving interception trajectory, t_go history, moving speed, and guidance command are presented for the proposed corrector using a target maneuver amplitude of 0.5g and varying desired impact times of t_d = 50s, 55s, 60s. In Fig. 7 (a), the dotted line represents the target trajectory and the dashed line stands for the interceptor's trajectory (we will also use the same trajectory line types in the following simulations). The results demonstrate that the computational ITCG algorithm

Table 4: Hyperparameters in the PPO training.

Hyperparameter	Definition	Value
c_ϑ	Ratio clipping	0.2
α_β	Actor learning rate	0.0001
α_ξ	Critic learning rate	0.0002
γ	Discounting factor	0.99
N	Size of the experience buffer	32
k_V	The scaling factor of V_I	0.01
k_e	The scaling factor of ε_t	1
k_Y	The scaling factor of Y_I	0.001
σ_1	The weight of r_i	0.8
σ_2	The weight of r_e	0.1
σ_3	The weight of r_V	0.05
σ_4	The weight of r_Y	0.05

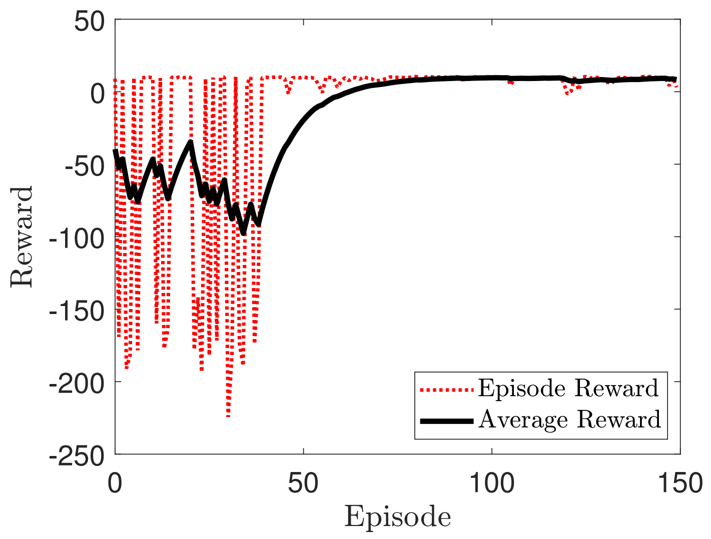
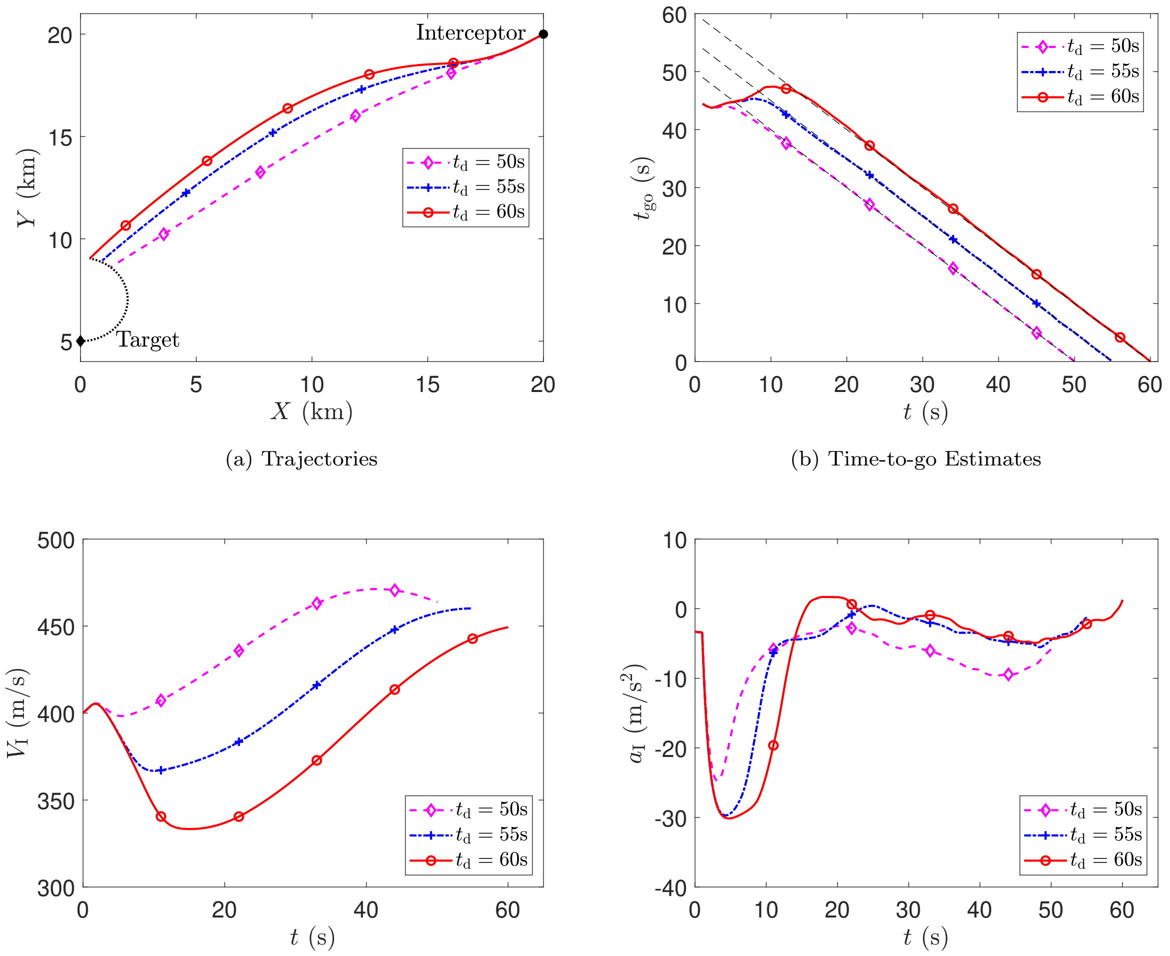


Figure 6: The Learning curve of the corrector.

proposed in this paper could accomplish intercepting a maneuvering target at the desired time. We can also note from the figure that the velocity loss increases with bigger discrepancy between the desired impact time and the predicted terminal time. The reason is that by increasing the desired impact time, the missile needs to maneuver more to adjust the trajectory and hence consumes more energy.

To investigate the effects of target maneuvers on the performance of the corrector, we conduct two sets of



(a) Trajectories

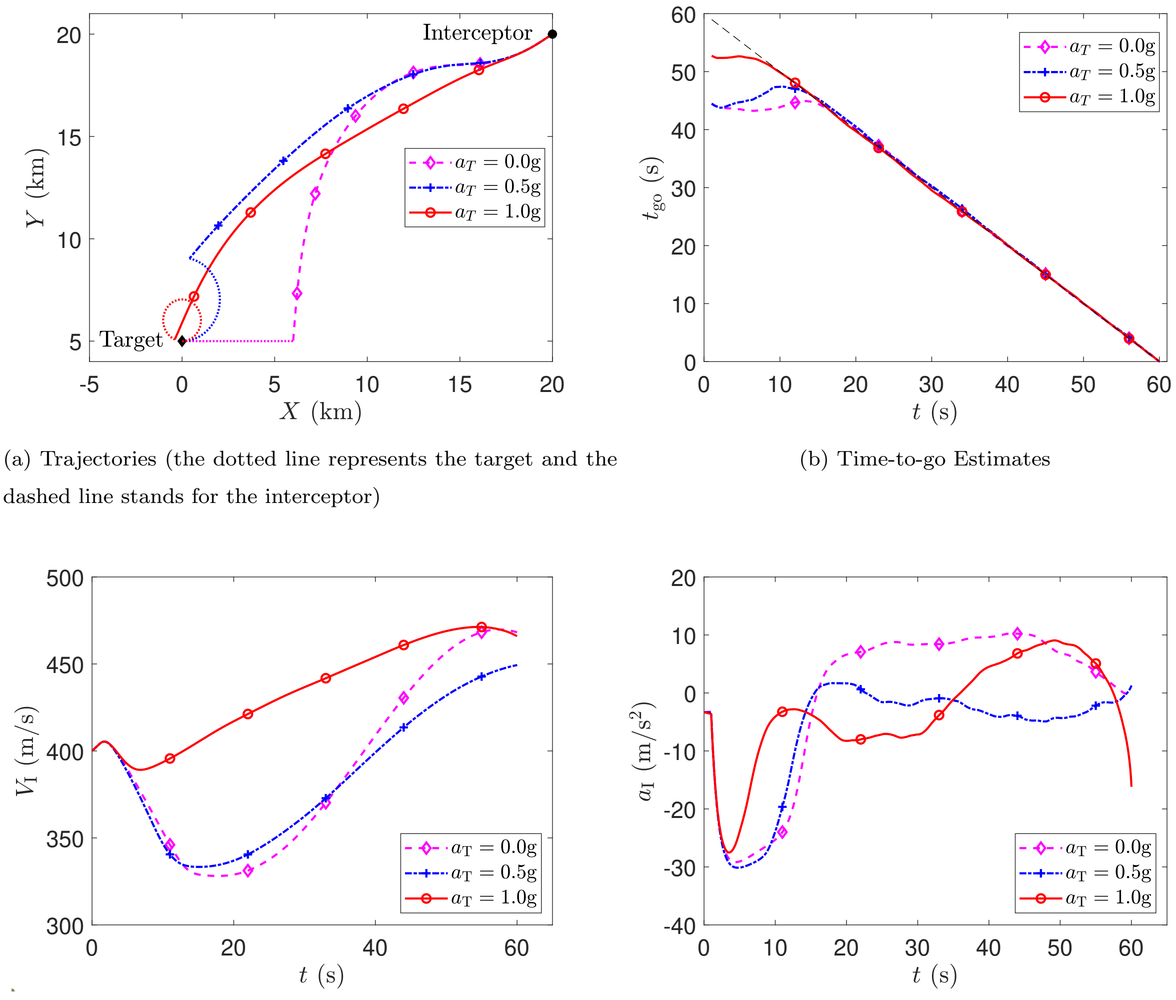
(b) Time-to-go Estimates

(c) Velocities

(d) Guidance Commands

Figure 7: Simulation result of intercepting maneuvering target at different t_{d}.

experiments with $t_{\mathrm{d}} = 60\mathrm{s}$. The initial values are the same as Eq. (32). The first set compares the guidance performance of targets with different maneuvering accelerations, namely 0g, 0.5g, 1g. As illustrated in Fig. 8, the interceptor successfully achieves the desired impact time for different target maneuvering accelerations. In the second set of experiments, we compare the interception with various maneuver forms, namely constant, square-wave, and sinusoidal maneuvers, where the square-wave and sinusoidal maneuvers have a time period of 40s. As shown in Fig. 9, the interceptor successfully intercepts targets with different maneuver forms. It is worth noting that the predictor's accuracy for the weaving maneuver decreases due to the use of a constant maneuver during the collection of the training set samples. However, as the target's feasible region and the variability gradually decrease when the interceptor closes in the target, the prediction error of t_{go} gradually converges, allowing the interceptor to successfully attack the target at t_{d}.



(a) Trajectories (the dotted line represents the target and the dashed line stands for the interceptor)

(b) Time-to-go Estimates

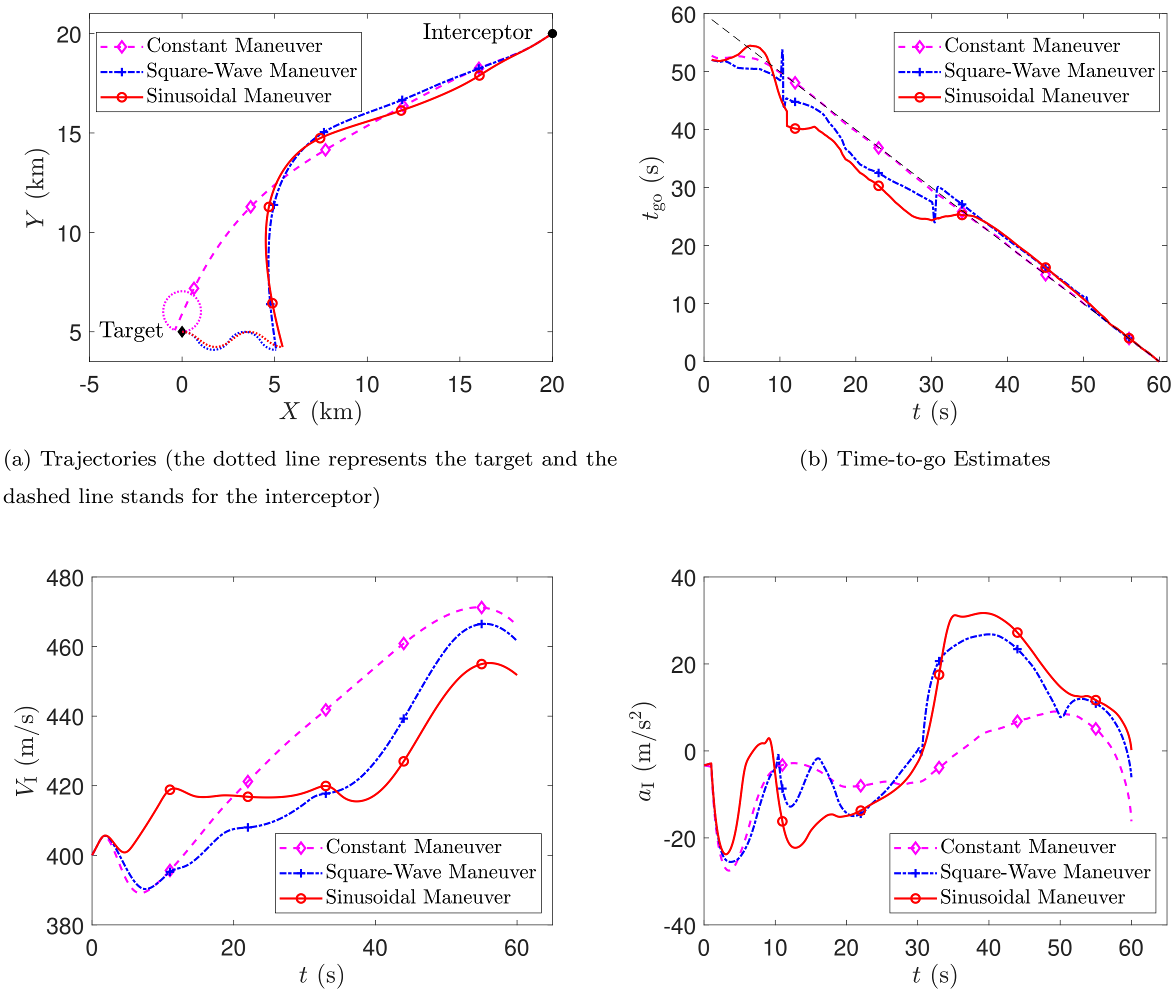
(c) Velocities

(d) Guidance Commands

Figure 8: Simulation result of intercepting maneuvering target with different a_T.

4.3. Monte-Carlo Analysis

This section presents results obtained from a Monte-Carlo simulation campaign to assess the performance of the computational ITCG method across a range of scenarios. We use uniformly distributions to sample the initial conditions and ranges of the initial conditions in the sampling is the same as shown in Table 1. The results of these simulations, including the interception trajectory, the error of interception time, moving speed, and guidance command, are presented in Fig. 10, where we utilize black dotted line to plot the trajectory of the target and color solid line to plot the trajectory of the interceptor. It is evident from the figure that the interceptor could successfully hit the target in all scenarios, and errors of impact time are evenly distributed around zero. Although a few errors are observed to be close to 3s, the majority of errors



(a) Trajectories (the dotted line represents the target and the dashed line stands for the interceptor)

(b) Time-to-go Estimates

(c) Velocities

(d) Guidance Commands

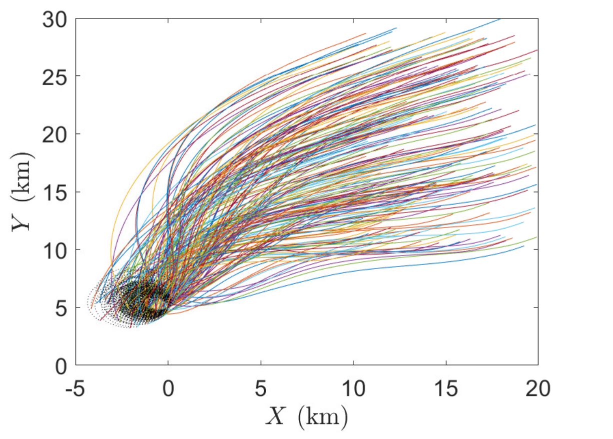
Figure 9: Simulation result of intercepting maneuvering target with different form of target maneuver.

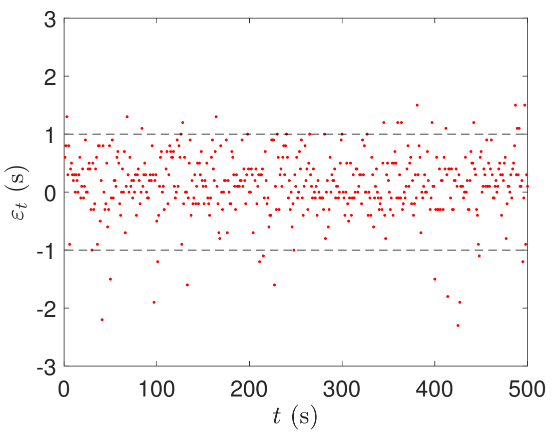
were within 1s. Consequently, the proposed ITCG algorithm provides good performance across different scenarios is verified by the Monte Carlo simulations.

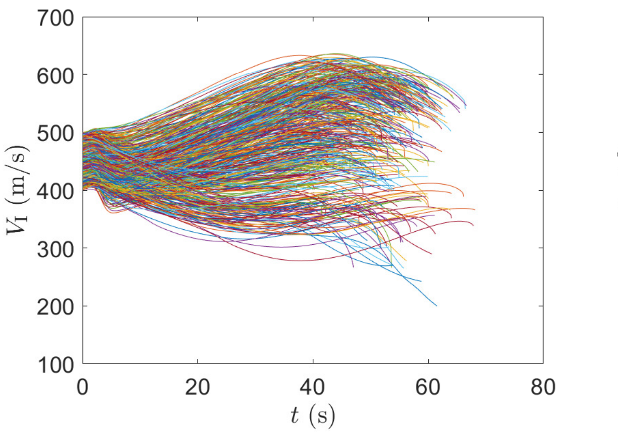
4.4. Comparison with Analytical Guidance Laws

To further show the advantages of the proposed algorithm, this subsection reports simulation results, in comparison with existing ITCG laws [7, 6]. Notice that both guidance laws are developed based on the biased PNG using the prediction-correction concept. Hence, they share similar characteristics as the proposed algorithm. The command of ITCG1 proposed in [7] is given by

$$a_I = NV_I\dot{\lambda} + \frac{K(2N-1)V_I^2}{R\sigma t_{go}}\varepsilon_t \tag{33}$$


(a) Trajectories (the dotted line represents the target and the dashed line stands for the interceptor)


(b) Impact Time Error


(c) Velocities

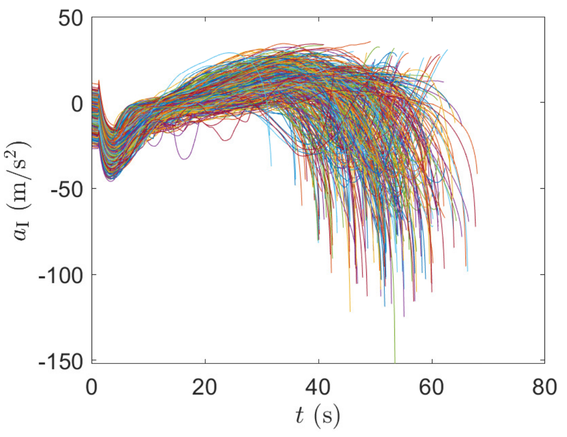

(d) Guidance Commands

Figure 10: Monte-Carlo simulation of intercepting maneuvering target.

where $N > 0$ and $K > 0$ are guidance gains which can be tuned.

The analytic ITCG2 law suggested in [6] improves the performance by using a varying gain as

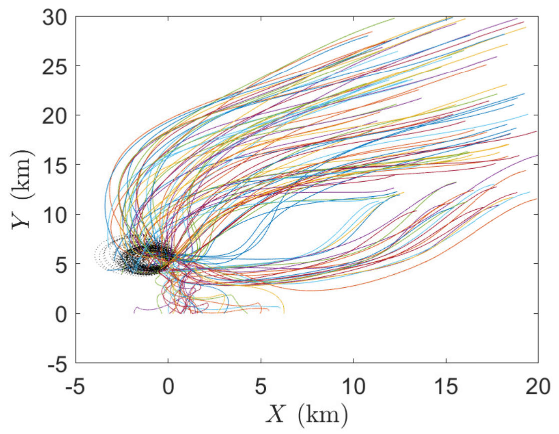
$$a_I = \mathcal{N}(\sigma) V_I \dot{\lambda} + \frac{K(2N-1)V_I^2}{R\sigma t_{go}} \varepsilon_t \tag{34}$$

where

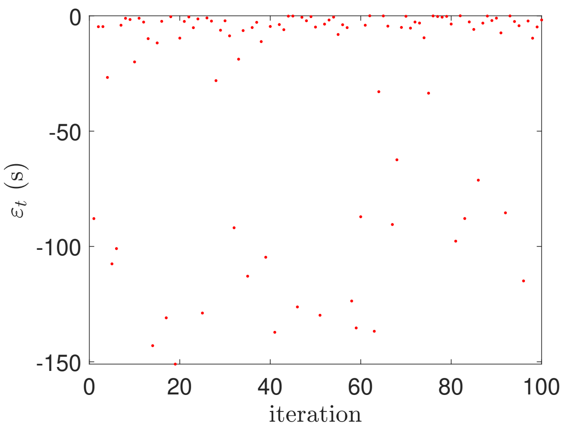
$$\mathcal{N}(\sigma) = \frac{(2N-1)(1-\cos\sigma)}{\sigma \sin\sigma} - \frac{\sigma\cos\sigma}{2\sin\sigma} + 1 \tag{35}$$

The impact time error ε_t in commands (33) and (34) are obtained using Eq. (16) with $M = 1$. The guidance gains in guidance commands (33) and (34) are chosen as $N = 4$ and $K = 6$. To better compare the performance, we run 100 Monte Carlo simulations of two analytical guidance methods. As shown in

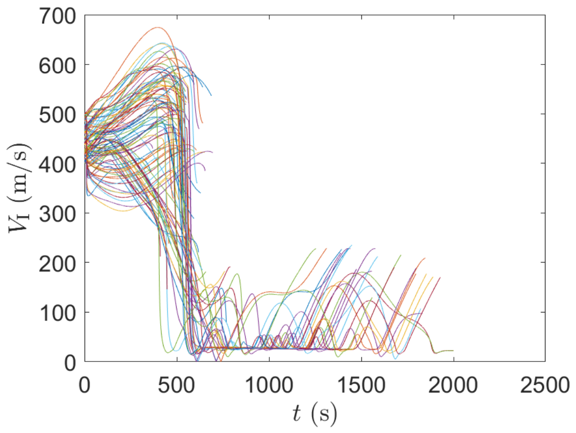
Figs. 11 and 12, under some initial conditions, the analytical guidance laws fail to hit the target, resulting in large terminal errors. The statistics show that ITCG1 generates 32 times of off-target and ITCG2 miss the target 27 times. Figure 13 shows the comparison when all analytical guidances can intercept the target with acceptable miss distance under certain initial condition. We find that the guidance commands of the analytical guidance laws diverge at the time of impact, and the final velocity of the proposed CITCG is significantly larger than the other two. Further statistics of the terminal errors of CITCG, ITCG1 and ITCG2 are shown in Table 5. Notice that only the terminal errors when all can hit the target are counted since excessive errors when missing the target will cause excessive deviations of the statistics. It can be seen that the errors of CITCG are smaller than the analytical guidance laws since the time-varying aerodynamic forces and unknown target maneuvers are taken into account.



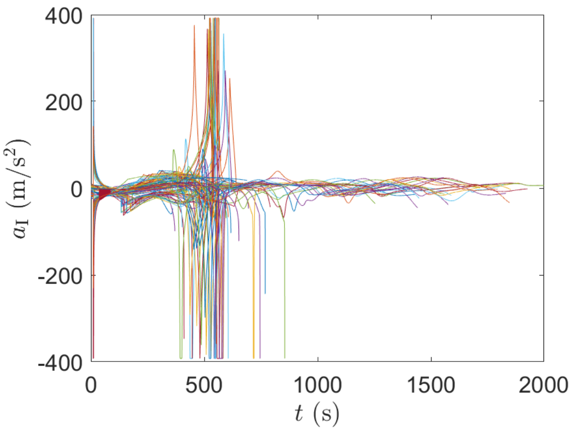
(a) Trajectories



(b) Time-to-go Estimation Error

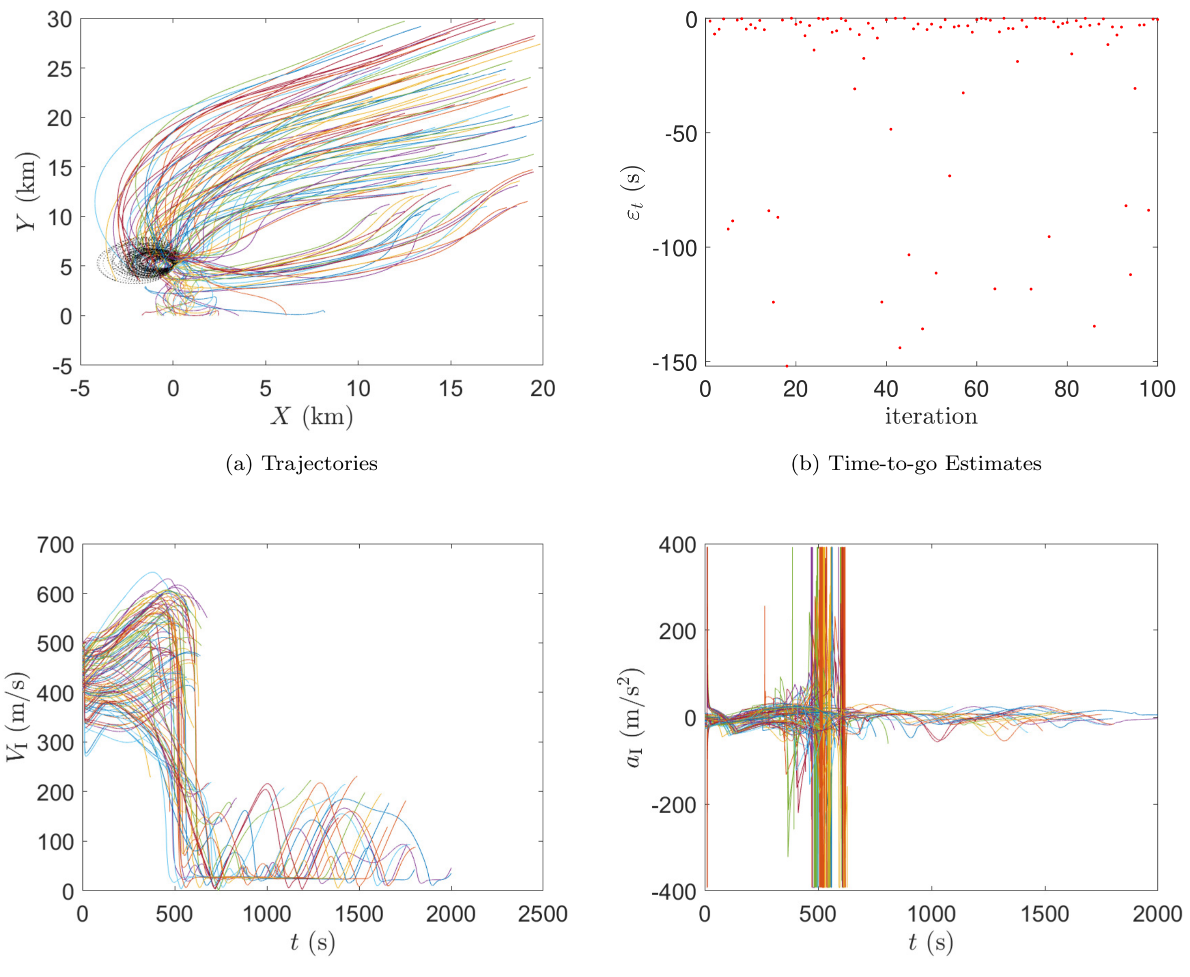


(c) Velocities



(d) Guidance Commands

Figure 11: Monte-Carlo simulation of ITCG1.



(a) Trajectories

(b) Time-to-go Estimates

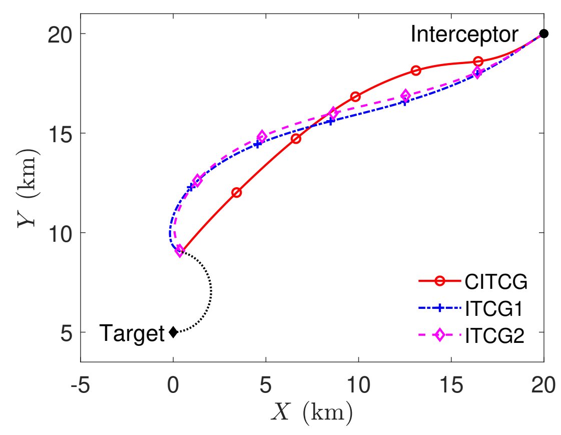
(c) Velocities

(d) Guidance Commands

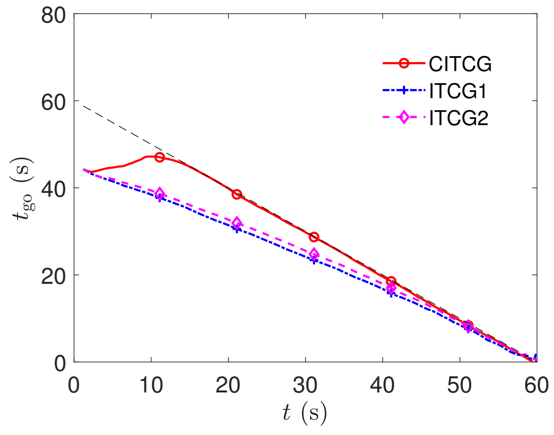
Figure 12: Monte-Carlo simulation of ITCG2.

Table 5: Comparison of terminal errors with analytical ITCG Laws

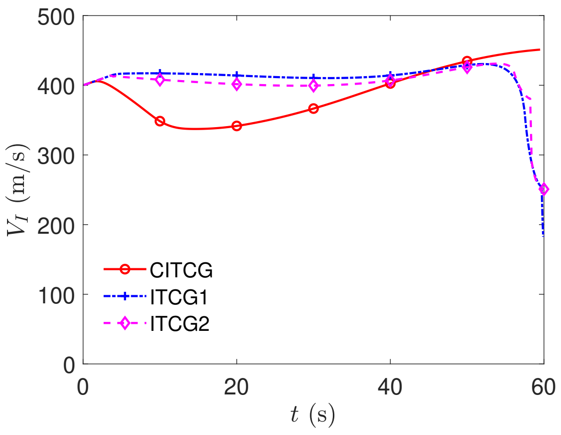
	CITCG			ITCG1			ITCG2		
	Mean	Maximum	Median	Mean	Maximum	Median	Mean	Maximum	Median
$\Delta x(m)$	0.9411	3.2073	0.7909	1.3724	4.6074	0.9236	1.3153	4.2636	0.9149
$\Delta y(m)$	1.0317	1.4043	1.2527	1.7029	4.8252	1.2400	1.6036	4.0965	1.1334
$\Delta t(s)$	0.1685	2.1001	0.2300	2.0262	4.8700	2.0900	1.9541	4.9600	1.6500



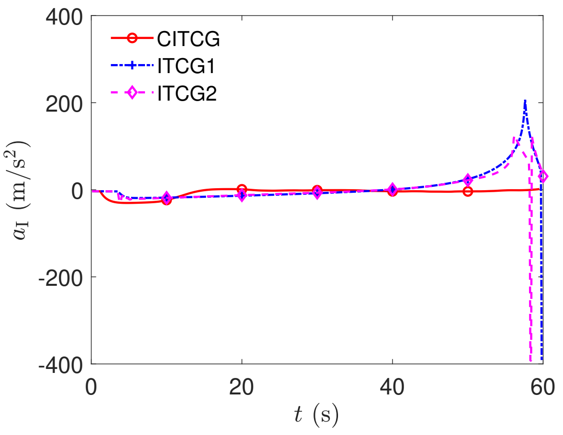
(a) Trajectories



(b) Time-to-go Estimation Error



(c) Velocities



(d) Guidance Commands

Figure 13: Comparison with analytic ITCG laws.

5. Conclusion

In this study, a novel computational guidance algorithm for controlling impact time against a maneuvering target is introduced. A new architecture called two-branch neural networks is proposed to process the interceptor's state vector and the target's trajectory sequence, leading to improved time-to-go prediction accuracy with time-varying speed models. Additionally, a biased command using reinforcement learning techniques is proposed to nullify the the impact time error. Numerical simulations of various desired impact times and target maneuvers indicate that the proposed computation ITCG algorithm demonstrates promising performance in implementation. Future works include extending the proposed algorithm to 3D scenarios and considering measurement uncertainties in performance evaluation.

Acknowledgement

This work was supported by the Technology Innovation Institute under Contract No. TII/ARRC/2154/2023 and National Natural Science Foundation of China under Grant No. 52302449.

References

[1] J. Yu, Z. Shi, X. Dong, Q. Li, J. Lv, Z. Ren, Impact time consensus cooperative guidance against the maneuvering target: theory and experiment, IEEE Transactions on Aerospace and Electronic Systems (2023).

[2] S. R. Kumar, D. Mukherjee, Terminal time-constrained nonlinear interception strategies against maneuvering targets, Journal of Guidance, Control, and Dynamics 44 (1) (2021) 200–209. `doi:10.2514/1.G005455`.

[3] S. He, D. Lin, Three-dimensional optimal impact time guidance for antiship missiles, Journal of Guidance, Control, and Dynamics 42 (4) (2019) 941–948.

[4] G. Merkulov, M. Weiss, T. Shima, Minimum-effort impact-time control guidance using quadratic kinematics approximation, Journal of Guidance, Control, and Dynamics (2021) 1–14`doi:10.2514/1.G006190`.

[5] S. Lee, N. Cho, Y. Kim, Impact-time-control guidance strategy with a composite structure considering the seeker's field-of-view constraint, Journal of Guidance, Control, and Dynamics 43 (8) (2020) 1566–1574. `doi:10.2514/1.G005063`.

[6] W. Dong, C. Wang, J. Wang, M. Xin, Varying-gain proportional navigation guidance for precise impact time control, Journal of Guidance, Control, and Dynamics 46 (3) (2023) 535–552. `doi:10.2514/1.G007174`.

[7] M.-J. Tahk, S.-W. Shim, S.-M. Hong, H.-L. Choi, C.-H. Lee, Impact time control based on time-to-go prediction for sea-skimming antiship missiles, IEEE Transactions on Aerospace and Electronic Systems 54 (4) (2018) 2043–2052. `doi:10.1109/taes.2018.2803538`.

[8] M. Kim, B. Jung, B. Han, S. Lee, Y. Kim, Lyapunov-based impact time control guidance laws against stationary targets, IEEE Transactions on Aerospace and Electronic Systems 51 (2) (2015) 1111–1122. `doi:10.1109/taes.2014.130717`.

[9] I.-S. Jeon, J.-I. Lee, M.-J. Tahk, Homing guidance law for cooperative attack of multiple missiles, Journal of Guidance, Control, and Dynamics 33 (1) (2010) 275–280. `doi:10.2514/1.40136`.

[10] K. Li, J. Wang, C.-H. Lee, R. Zhou, S. Zhao, Distributed cooperative guidance for multivehicle simultaneous arrival without numerical singularities, Journal of Guidance, Control, and Dynamics 43 (7) (2020) 1365–1373. `doi:10.2514/1.G005010`.

[11] S. He, W. Wang, D. Lin, H. Lei, Consensus-based two-stage salvo attack guidance, IEEE Transactions on Aerospace and Electronic Systems 54 (3) (2017) 1555–1566.

[12] J. In-Soo, L. Jin-Ik, T. Min-Jea, Impact-time-control guidance law for anti-ship missiles, IEEE Transactions on Control Systems Technology 14 (2) (2006) 260–266. `doi:10.1109/tcst.2005.863655`.

[13] I.-S. Jeon, J.-I. Lee, M.-J. Tahk, Impact-time-control guidance with generalized proportional navigation based on nonlinear formulation, Journal of Guidance, Control, and Dynamics 39 (8) (2016) 1885–1890. `doi:10.2514/1.G001681`.

[14] R. V. Nanavati, S. R. Kumar, A. Maity, Three-dimensional suboptimal nonlinear impact time guidance against non-maneuvering target, Aerospace Science and Technology 115 (2021). `doi:10.1016/j.ast.2021.106831`.

[15] S. R. Kumar, D. Mukherjee, Deviated pursuit based interception at a priori fixed time, Journal of Guidance, Control, and Dynamics 42 (9) (2019) 2124–2131. `doi:10.2514/1.G004284`.

[16] S. Guoxin, W. Qiuqiu, X. Zhiqiang, X. Qunli, Impact time control using biased proportional navigation for missiles with varying velocity, Chinese Journal of Aeronautics 33 (3) (2020) 956–964.

[17] C. Peng, H. Zhang, Y. He, J. Ma, State-following-kernel-based online reinforcement learning guidance law against maneuvering target, IEEE Transactions on Aerospace and Electronic Systems 58 (6) (2022) 5784–5797. `doi:10.1109/taes.2022.3178770`.

[18] T. Jin, S. He, Ensemble transfer learning midcourse guidance algorithm for velocity maximization, Journal of Aerospace Information Systems 20 (4) (2023) 204–215. doi:10.2514/1.I011070.

[19] D. Izzo, E. Öztürk, Real-time guidance for low-thrust transfers using deep neural networks, Journal of Guidance, Control, and Dynamics 44 (2) (2021) 315–327. doi:10.2514/1.G005254.

[20] S. You, C. Wan, R. Dai, J. R. Rea, Learning-based onboard guidance for fuel-optimal powered descent, Journal of Guidance, Control, and Dynamics 44 (3) (2021) 601–613. doi:10.2514/1.G004928.

[21] L. Cheng, F. Jiang, Z. Wang, J. Li, Multiconstrained real-time entry guidance using deep neural networks, IEEE Transactions on Aerospace and Electronic Systems 57 (1) (2021) 325–340. doi:10.1109/TAES.2020.3015321.

[22] X. Qiu, C. Gao, W. Jing, Maneuvering penetration strategies of ballistic missiles based on deep reinforcement learning, Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering 236 (16) (2022) 3494–3504. doi:10.1177/09544100221088361.

[23] S. He, H.-S. Shin, A. Tsourdos, Computational missile guidance: A deep reinforcement learning approach, Journal of Aerospace Information Systems 18 (8) (2021) 571–582. doi:10.2514/1.I010970.

[24] B. Gaudet, R. Furfaro, R. Linares, Reinforcement learning for angle-only intercept guidance of maneuvering targets, Aerospace Science and Technology 99 (2020). doi:10.1016/j.ast.2020.105746.

[25] W. Chen, C. Gao, W. Jing, Proximal policy optimization guidance algorithm for intercepting near-space maneuvering targets, Aerospace Science and Technology 132 (2023). doi:10.1016/j.ast.2022.108031.

[26] V. Shalumov, Cooperative online guide-launch-guide policy in a target-missile-defender engagement using deep reinforcement learning, Aerospace Science and Technology 104 (2020). doi:10.1016/j.ast.2020.105996.

[27] Z. Liu, J. Wang, S. He, H.-S. Shin, A. Tsourdos, Learning prediction-correction guidance for impact time control, Aerospace Science and Technology 119 (2021). doi:10.1016/j.ast.2021.107187.

[28] N. Wang, X. Wang, N. Cui, Y. Li, B. Liu, Deep reinforcement learning-based impact time control guidance law with constraints on the field-of-view, Aerospace Science and Technology 128 (2022). doi:10.1016/j.ast.2022.107765.

[29] P. Zarchan, Tactical and strategic missile guidance, American Institute of Aeronautics and Astronautics, Inc., 2012.

[30] N. Cho, Y. Kim, Modified pure proportional navigation guidance law for impact time control, Journal of Guidance, Control, and Dynamics 39 (4) (2016) 852–872.

[31] S. He, C.-H. Lee, Optimality of error dynamics in missile guidance problems, Journal of Guidance, Control, and Dynamics 41 (7) (2018) 1624–1633. doi:10.2514/1.G003343.

[32] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International Conference on Machine Learning, 2015, pp. 1889–1897. doi:10.48550/arXiv.1502.05477.

[33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017). doi:10.48550/arXiv.1707.06347.

[34] M. Holzleitner, L. Gruber, J. Arjona-Medina, J. Brandstetter, S. Hochreiter, Convergence proof for actor-critic methods applied to ppo and rudder, in: Transactions on Large-Scale Data-and Knowledge-Centered Systems XLVIII: Special Issue In Memory of Univ. Prof. Dr. Roland Wagner, Springer, 2021, pp. 105–130.