SAND2021-14913C

Test and Evaluation of Reinforcement Learning via Robustness Testing and Explainable AI for High-Speed Aerospace Vehicles

Ali K. Raz
Systems Engineering and Operations Research
George Mason University
Fairfax, VA 22030
araz@gmu.edu

Sean Matthew Nolan
School of Aeronautics and Astronautics
Purdue University
West Lafayette, IN 47907
nolans@purdue.edu

Winston Levin
School of Aeronautics and Astronautics
Purdue University
West Lafayette, IN 47907
wlevin@purdue.edu

Kshitij Mall
School of Aeronautics and Astronautics
Purdue University
West Lafayette, IN 47907
mall@purdue.edu

Ahmad Mia
Systems Engineering and Operations Research
George Mason University
Fairfax, VA 22030
amia2@gmu.edu

Linas Mockus
School of Aeronautics and Astronautics
Purdue University
West Lafayette, IN 47907
lmockus@purdue.edu

Kris Ezra
School of Aeronautics and Astronautics
Purdue University
West Lafayette, IN 47907
kris@purdue.edu

Kyle Williams
Autonomous Sensing and Control
Sandia National Laboratories
Albuquerque, NM 87185
kwilli2@sandia.gov

***Abstract*—Reinforcement Learning (RL) provides an ability to train an artificial intelligent agent in dynamic and uncertain environments. RL has demonstrated an impressive performance capability to learn nearly optimal policies in various application domains including aerospace. Despite the demonstrated performance outcomes of RL, characterizing performance boundaries, explaining the logic behind RL decisions, and quantifying resulting uncertainties in RL outputs are major challenges that slow down the adoption of RL in real-time systems. This is particularly true for aerospace systems where the risk of failure is high and performance envelopes of systems of interest may be small. To facilitate adoption of learning agents in real-time systems, this paper presents a three-part Test and Evaluation (T&E) framework for RL built from Systems engineering for artificial intelligence (SE4AI) perspective. This T&E framework introduces robustness testing approaches to characterize performance bounds on RL, employs Explainable AI techniques, namely Shapley Additive Explanations (SHAP) to examine RL decision-making, and incorporates validation of RL outputs with known and accepted solutions. This framework is applied to a high-speed aerospace vehicle emergency descent problem where RL is trained to provide an angle of attack command and the framework is utilized to comprehensively examine the impact of uncertainties in the vehicle's altitude, velocity, and flight path angle. The robustness testing characterizes acceptable ranges of disturbances in flight parameters, while SHAP exposes the most significant features that impact RL selection of angle of attack—in this case the vehicle altitude. Finally, RL outputs are compared to trajectory generated by indirect optimal control methods for validation.**

Keywords: Reinforcement Learning, Robustness Testing, Explainable AI, SHAP.

978-1-6654-3760-8/22/$31.00 ©2022 IEEE

TABLE OF CONTENTS

1. INTRODUCTION

Reinforcement Learning (RL) provides the ability to train an artificial intelligence (AI) agent to operate in dynamic and uncertain environments and is becoming increasingly popular in a variety of domains including aerospace [1]. In this particular domain, RL has been used to provide missile guidance [2], flight control [3], and motion planning for swarms of autonomous vehicles [4], and the list of potential application areas continues to grow. RL has demonstrated an impressive capability to approximate nearly optimal solutions in these applications—and many others—with comparatively low requirements for computational power and training requirements in well-curated domains. Despite the demonstrated performance outcomes of RL, characterizing performance boundaries, explaining the logic behind RL decisions, and quantifying resulting uncertainties in RL outputs are major challenges that slow the adoption of RL in real-time systems. This is particularly true for aerospace systems where the risk

of failure is high and performance envelopes of systems of interest may be small.

Systems engineering for artificial intelligence (SE4AI) is an emerging area of research in systems engineering community with the goal of applying systems engineering methods to the design and operation of learning-based systems [5]. The role of SE4AI becomes crucial as RL and other machine learning approaches become increasingly viable for real-time, practical systems. In this paper, we consider RL from the standpoint of SE4AI and propose approaches to facilitate test and evaluation (T&E) of RL solutions. From this perspective, we identify improved methods for providing RL performance characterization, explanation of decision-making, and quantification of uncertainty.

Considering RL from this SE4AI standpoint is particularly relevant because RL—and more generally machine learning—approaches essentially become an element of a larger, enterprise system as complexity of the system and mission-space increase. This AI-driven constituent system must interact predictably and reliably with the rest of the system elements despite the fact that machine learning is often treated as a "black box" in practice. Commensurately, the SE4AI viewpoint applies a systems engineering approach to study, design, verify, and validate the interaction of learning based elements with other elements.

In this paper, we use RL to select maneuvers for a high-speed aerospace vehicle to conduct an emergency descent. The application of RL is particularly important here since closed-form, optimal solutions for spontaneous maneuvers are likely to be unavailable in real-time for high-speed vehicles with narrow performance (and safety) envelopes.

First, we demonstrate training of an AI agent using RL to achieve the proposed mission for a high-speed aerospace vehicle. Second, we propose a T&E framework for RL that includes statistical and mathematical methods for analysis of RL outcomes by building on a SE4AI perspective. This framework, which has three parts, provides a characterization of performance bounds and sensitivities on the vehicle mission based on RL-guided decisions that not only facilitates a comprehensive understanding of RL decision-making and capabilities, but can also be used to derive performance requirements for other system elements.

There are three primary elements in our proposed T&E framework:

1. **Robustness Testing of RL:** The purpose of robustness testing is to characterize response/output sensitivity to variations in a system's inputs which includes training conditions for learning-based systems. Since the design space of these variations can be large for high-speed aerospace applications, we propose a Latin-Hypercube Sampling (LHS) [6] strategy from Design of Experiments (DoE) [7].

2. **Explainability of RL Model:** AI Explainability methods interrogate the deep neural network that underlies RL decision-making and identifies significant (and insignificant) features that influence RL actions [8]. Our framework uses an explainable AI (XAI) technique known as Shapley Additive Explanations (SHAP) to identify which input features in the high-speed aerospace mission are most influential for RL decision making.

3. **RL Comparison with Traditional Optimal Control Techniques:** This step examines RL outputs compared to optimal solutions. These solutions can validate RL output and inform improvements to the framework prior to training on more complex scenarios on which traditional optimal control struggles.

The remainder of the paper is organized as follows. Section 2 provides a brief background on RL and motivates the need for a T&E framework. Section 3 provides details on the proposed T&E framework for RL and includes formulation of the three primary elements described above. Section 4 describes the high-speed aerospace systems mission example and demonstrates the use of RL to provide guidance commands to a vehicle. Section 5 discusses the implementation methodology for the T&E framework and demonstrates the ability of RL to guide a vehicle to satisfy its mission while closely (but not exactly) following the optimal trajectory. Further, we demonstrate 1) how robustness testing helps to establish upper and lower performance bounds on the RL outputs, and 2) how the explainability insights derived from the SHAP model aid in understanding RL decision-making constructs. Finally, Section 6 provides summary and future work directions.

2. BACKGROUND

Introduction to Reinforcement Learning

This section provides a brief and simplified description of RL necessary to support the formulation of the T&E framework and high-speed aerospace vehicle application problem. In RL, an AI agent learns by interacting with its environment to develop a mapping between the available action space and the corresponding environmental changes that maximize its reward function [1]. RL agent training is defined by this reward-driven learning that sets it apart from other variants of machine learning requiring large data sets for training and validation. With no-prior or limited knowledge of the underlying dynamics and attributes of the environment, the RL agent can learn optimal or nearly optimal policies to achieve its goal and adapt to changing conditions [1, 3, 9].

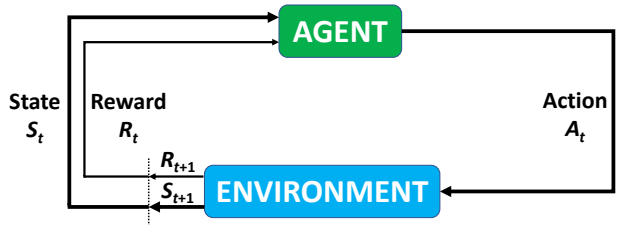


Figure 1. Reinforcement Learning framework.

Figure 1, adapted from [1], displays the fundamental constructs of an RL problem space. Here the *Environment* provides a dynamic behavior of the problem of interest which is typically unknown to the RL agent at the outset. The environment produces a state vector S_t and scalar reward value R_t based on current state and the learning objective i.e., reaching a desired state in the environment, which is the mission objective. The RL agent has access to a predefined action space, A_t, at any given time, t, whose selection introduces a change in the state and reward value at the next time step, S_{t+1} and R_{t+1}, respectively. Underlying the RL agent is a deep neural network (DNN) whose training is accomplished via learning policies such as Proximal Policy Optimization (PPO) [10], Deterministic Policy Gradient, Deep Deterministic Policy

Gradient [11], etc. DNN training proceeds over multiple learning episodes under the governing learning policy; to put it succinctly, a reward value R_t is obtained by selecting an action A_t and observing the corresponding changes in S_{t+1} and R_{t+1}.

The Need for Test and Evaluation Framework of RL

An extensive body of literature and conceptual demonstrations is available to illustrate the efficacy of RL across many domains including aerospace. Most publications related to RL focus on algorithmic formulations and domain-specific demonstration of performance. This is reasonable considering that learning policy developments and configurations are often unintuitive or esoteric; however, comprehensive testing and validation of RL—including near or outside the boundaries of the training data—is also extremely important. This is not to claim or insinuate that no RL validation testing is performed on RL outputs: most authors and developers do include a set of test cases (albeit selective and/or nominal) to showcase RL performance. For example, Sun and Kampen utilize Monte Carlo simulations and additive zero-mean white noise disturbances in state observations for characterizing RL implementation for flight control solutions [3]. Wu et al., introduce new environment regions in RL testing [12]. Fedrici et al., specify six different test cases that include variations in initial conditions and state observations and evaluate the learning performance using Monte Carlo simulations, whereas Izzo and Ozturk consider two cases of environment initial conditions for studying DNN performance [13, 14]. McGuire et al., propose a cumulative reward as a metric to compare different policies using analysis of variance, while Buzii et al. employ a similar metric of cumulative reward to compare different learning policies via Monte Carlo simulations [15, 16]. The main point here is that all of these test and evaluation approaches are valid and necessary, but none are independently rigorous or comprehensive. More directly, none are sufficient to meaningfully characterize performance implications outside of—or at the boundary of—the trained input conditions.

Once trained, an RL agent is essentially a black box DNN whose decisions are generally unexplainable except that they are driven by a user-specified reward function. While a black box is sufficient for simple problems, identifying RL performance bounds, sensitivities, uncertainties, and motivation are vitally important in real-time operational systems with high cost, high risk, or both. A comprehensive and widely applicable T&E framework is necessary to satisfy this need for RL testing, validation, and explanation. Following the systems engineering approach, identifying the features and attributes of RL by considering it as an integrated part of a complex system provides the right type of perspective to answer these questions. We propose a T&E framework that can be utilized by RL developers and testers, particularly in aerospace applications, to validate and explain RL-driven system outcomes.

3. TEST AND EVALUATION METHODOLOGY FOR REINFORCEMENT LEARNING

Testing, evaluation, and validation of RL is a multi-faceted problem that requires a suite of methods and tools to accomplish. Sensitivity analysis of RL performance to both modeled (i.e., included in the training) and unmodeled considerations is important and so is validating RL performance against any available known solutions. Furthermore, understanding RL decision-making and knowing why and how an RL agent is going to select what action is also of significant value to system designer and operators.

The T&E approach proposed in this section attempts to incorporate these multi-faceted requirements of RL performance evaluation and postulates a framework with various engineering methods and tools that can be tailored to accomplish RL T&E goals. This RL T&E framework, shown in Figure 2, is composed of three interdependent and complementary elements that provide a comprehensive evaluation of RL-based solutions when applied together. The subsequent sections describe the purpose, methodology, and value of these three elements (namely, Robustness Testing, XAI, and Comparison with known solutions).

Robustness Testing of RL

The purpose of *Robustness Testing* is to quantify upper and lower bounds of RL performance and characterize impact of uncertainties on that performance in practice.

One of the first steps in Robustness Testing is to identify sources of uncertainty and the scope of input and environment variations in a system design space. Best practices in systems engineering emphasizes identification of these variations in factors that are both within and outside the designer's influence and control [17]. From an RL perspective, factors such as the tuning parameters of RL learning policies and definition of the reward function (commonly known as reward function shaping) remain under the control of designers and are thoroughly investigated to create RL solutions. However, it is important to note that once an RL solution is integrated into a larger system, it may face additional uncertainties originating in the environment, state space, or the action space. For example, consider an RL agent that selects an angle of attack (α) command for an aircraft, which is then implemented by a flight control system. The flight control system moves control surfaces to the desired value within a tolerance limit, say $\alpha \pm \mu$, but may not exactly match α as specified by RL. It then becomes important to characterize the behaviour of RL for variations of α within $\pm\mu$ and assess how those unintended variances may impact the behavior of the system. The purpose of Robustness Testing is to holistically identify such variations and establish RL performance bounds when variations occur.

An intuitive question that may arise here is why not include these known variations in RL training in the first place? Of course, these variations should be included in the RL training to best extent possible, but it is also important to note that the efficacy and convergence of RL solution decreases as the number of factors and uncertainties increase in the RL design space. Moreover, including the uncertainties and variations in the training does not absolve the need for Robustness Testing because the robustness and performance envelopes of trained RL solutions must still be established—which is a core motivation of the Robustness Testing. Such variations may be themselves functions of the environment (i.e., very high or very low temperatures may increase variation) and may be difficult to provide as training inputs.

In addition to the training policies and reward function definition, the uncertainties and variations in the other constituent elements of RL—i.e., the environment, the state space, and the action space—must also be considered. Table 1 provides summary of the different types of variations that are commonly expected in aerospace applications and include approaches on how to model them for the Robustness Testing.

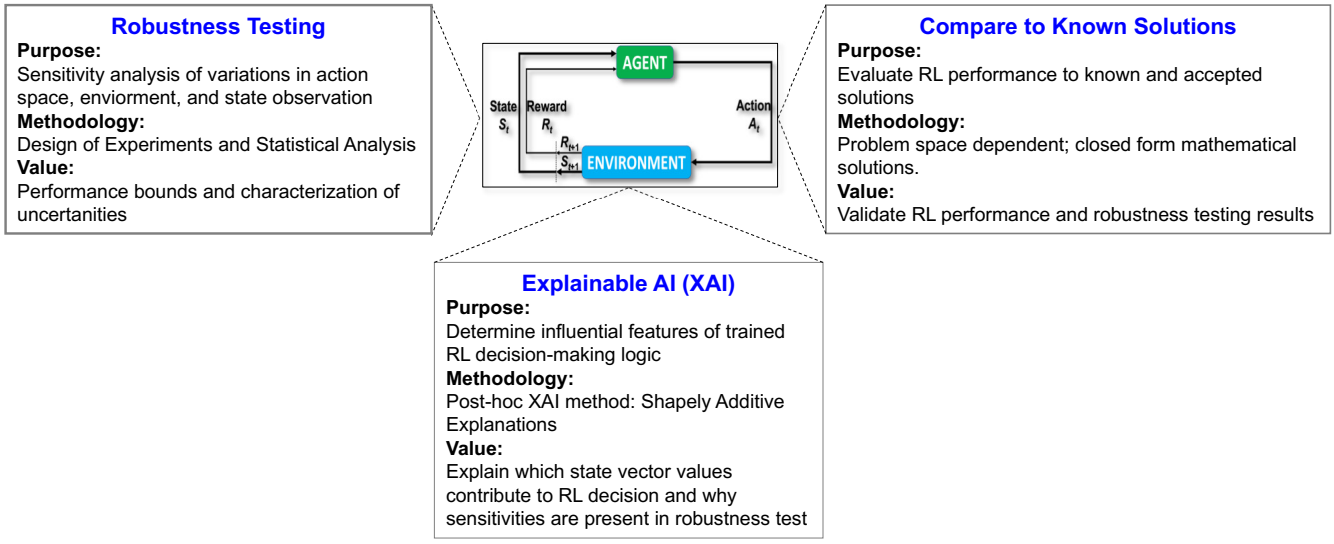


Figure 2. RL Test and Evaluation framework.

Table 1. Sources of variation in RL for robustness testing.

Source	Nature of Variation	Modeling Approach for Robustness Testing
Environment	Initial Conditions	Latin Hypercube Sampling Monte Carlo Simulations Design of Experiments
Action Space or State Space	Tolerance and Sensitivity	Expected probability distribution with parameters (e.g., $\mathcal{N}(\mu, \sigma^2)$)
	Impulses and Hard Overs	Expected magnitude and time duration

Considering the different options of variations identified in Table 1, two potential challenges that arise in the Robustness Testing of RL are: 1) how to sample the design space for building data sets with sufficient coverage of the design space, and 2) how to analyze the resulting data to derive Robustness Testing insights.

For the first challenge, Monte Carlo methods and random sampling are commonly used in Robustness Testing, whereas we propose the use of LHS for generating data sets for analysis. Monte Carlo sampling generates a random sample of some number N of points for each uncertain input variable of a model. Since Monte Carlo sampling relies on pure randomness, it can be inefficient because one might end up with some points clustered closely, while other intervals within the design space get no samples. LHS, on the other hand, aims to spread the sample points more evenly across all possible values. It partitions each input distribution into N intervals of equal probability, and selects one sample from each interval. Furthermore, it can drastically reduce the number of runs necessary to achieve a reasonably accurate result. For additional information on using LHS for engineering design and analysis, the interested reader is referred to Refs. [18] and [19]. For the second challenge, we propose utilizing DoE in conjunction with statistical analysis methods such as Analysis of Variance and hypothesis testing as demonstrated in Ref. [20].

The results section of this paper discusses how the variations identified in Table 1 are applied for Robustness Testing of RL for high-speed aerospace applications and aids in establishing the performance bounds.

It is important to note that while the Robustness Testing immensely helps in understanding performance attributes and limitations of trained RL agent, it does not expose any decision-making logic or constructs of the RL agent. In other words, Robustness Testing will highlight sensitivities of the RL agent without describing why such sensitivities are present. XAI techniques attempt to answer this 'why' question and are described in the next section.

Explainable AI for RL

The actions taken by an RL agent are sometimes non-intuitive and hard to explain. To improve the explainability of RL agent's actions, XAI analysis was carried out. Two different types of XAI analysis are used based on two different types of AI models: *Transparent Models* and *Post-Hoc Analysis*. While simple AI models like linear/logistic regression can be explained by transparent XAI models, post-hoc analysis is required for more complicated AI models like RL. Since the scope of this paper involves an RL agent, post-hoc analysis was chosen.

There are several types of approaches in post-hoc analysis, including visualization, model simplification, text explanation, local explanations, explanations by example, and feature importance [23]. Since the RL agent developed in this study involves various input features (see Section 4), the feature importance approach was selected.

This study employs a popular, recently devised python-based XAI tool named as SHAP [24]. The application of XAI to RL, dubbed as explainable RL (XRL), has been sparsely studied [25, 26] and implemented [27–29]. Based on the literature study carried out for this paper, this possibly is the first instance of applying XRL analysis in high-speed aerospace systems domain, which is one of the main contributions of this paper. Literature in this domain suggests the use of SHAP for explaining the actions of RL [25, 26]. As

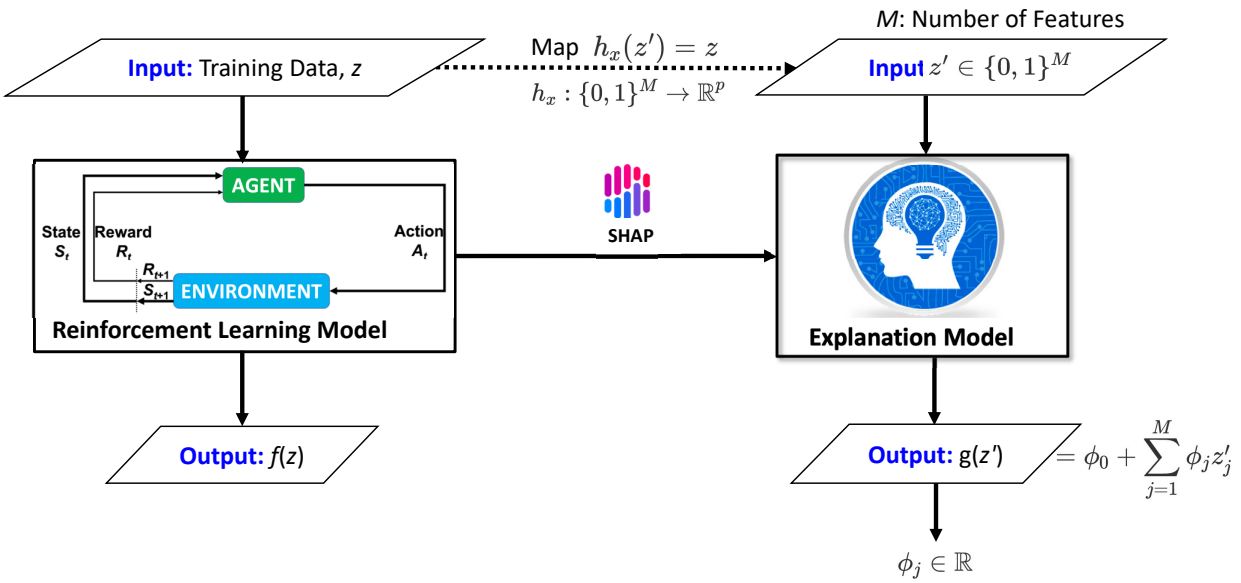


Figure 3. Converting the RL black box model to a more explainable model using SHAP [21, 22].

the name indicates, SHAP is based on Shapley values that were first introduced by Dr. Lloyd Shapley in 1953 [30]. A Shapley value captures the contribution of the corresponding feature towards the prediction of interest from among all features used as input for the model. Further, SHAP involves *additive* explanations because the Shapley values sum to give the output generated by the learning model.

Figure 3 shows the implementation of SHAP to obtain a simple linear explanation model from the complex RL model used in this study. The input data, z, is mapped to simpler inputs, z', using a mapping function, $h_z(z')$. Thus, z is converted to z', with elements taking values only of 0 and 1, corresponding to a feature being absent or present, respectively. The linear XAI model outputs $g(z')$ that has the form given by eq. (1). Let z_S be the set of features in z with non-zero entries in z'. Then, the ϕ in eq. (1) are attributions for each feature present in z', and the SHAP values are averages of these attributions over all possible ordered subsets z_S of z such that the values satisfy local accuracy, "missingness", and consistency properties [31]. Note that SHAP values can have positive or negative values. The additive nature of explanations provided by SHAP is also captured in eq. (1).

$$f(z) = g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i \qquad (1)$$

For this study, we employed the SHAP toolbox developed by Lundberg [24] for all SHAP analysis. As shown in fig. 4, four different type of explainers are available in the SHAP toolbox: Tree Explainer, Deep Explainer, Gradient Explainer, and Kernel Explainer. Tree Explainer is relevant only for decision tree based models. On the other hand, Deep Explainer and Gradient Explainer are used for deep learning models, but are not supported for RL models. Therefore, Kernel Explainer was chosen for this study as it uses a model agnostic approach. Kernel Explainer is based on Local Interpretable Model-Agnostic Explanations (LIME) [32] and Shapley values [30].

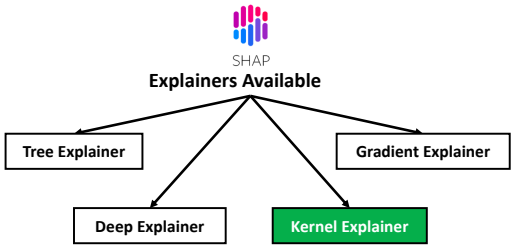


Figure 4. Available SHAP explainers.

The positive and negative SHAP values are indicated by red and blue colors respectively in the plots generated by this toolbox. The SHAP results obtained for this study are shown in Section 5.

Comparison with Known Solutions

In applications where known solutions can be found without the use of RL to identical or even similar problems, it becomes necessary to evaluate the quality of RL output as compared to those solutions. This can assist in validating RL solutions and identify the strength and weaknesses of both RL and known solutions. For example, if the purpose of deploying RL is to provide control for a vehicle, the RL solution will need to be compared to the optimal control solution. Here the objective is not to mimic the optimal control solution entirely (although in some cases it maybe possible), but to develop an understanding of the solution profile including commonalities and difference between the two.

4. HIGH SPEED AEROSPACE VEHICLE PROBLEM FORMULATION

Problem Description

The problem of interest for this paper is trajectory planning for a high-speed, unthrusted passenger vehicle that experiences some emergency situation and must descend to a safe altitude in minimum time.

The objective functional (eq. (2)), equations of motion (eq. (3)), and nominal boundary conditions (eq. (4)) describe the problem mathematically for a starting altitude of 30 km at 3 km/s and a safe altitude of 3 km. The state variables are altitude h, downrange angle θ, velocity v, and flight path angle γ. The control variable is angle of attack α. Vehicle information including mass m and aerodynamic functions for lift $L(\alpha)$ and drag $D(\alpha)$ are taken from the CAV-H model given in [33].

$$\min_{\alpha} J = t_f = \int_0^{t_f} dt \tag{2}$$

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{h} \\ \dot{\theta} \\ \dot{v} \\ \dot{\gamma} \end{bmatrix} = \begin{bmatrix} v \sin\gamma \\ \dfrac{v}{h+R_e}\cos\gamma \\ \dfrac{-D(\alpha)}{m} - \dfrac{\mu}{(h+R_e)^2}\sin\gamma \\ \dfrac{L(\alpha)}{mv} + \left(\dfrac{v}{h+R_e} - \dfrac{\mu}{v(h+R_e)^2}\right)\cos\gamma \end{bmatrix} \tag{3}$$

$$\begin{aligned} \boldsymbol{\Psi}_0 &= \begin{bmatrix} h - 30\,\text{km} \\ \theta \\ v - 3\,\text{km/s} \\ \gamma \end{bmatrix}\Bigg|_{t=0} = \mathbf{0} \\ \boldsymbol{\Psi}_f &= \begin{bmatrix} h - 3\,\text{km} \\ \gamma \end{bmatrix}\Bigg|_{t=t_f} = \mathbf{0} \end{aligned} \tag{4}$$

The problem is solvable with traditional indirect optimal control methods allowing for the validation of the RL implementation in Section 5.

Reinforcement Learning Implementation

The RL problem is implemented in Python following the framework depicted in fig. 1.

RL Agent—The vehicle's trajectory was optimized using the PPO method of training a DNN [10]. This method was implemented to train the network, log the training process, and save the trained DNN using the stable-baselines3 package in Python [34]. This trained DNN is the RL agent shown in Figure 1.

Environment—The optimal descent problem along with the vehicle equations of motion and boundary conditions was implemented with a target altitude $h_{target} = 3000$ m. Each episode ran for up to $t_{max} = 50$ seconds starting with a randomly selected normally distributed initial states given in eq. (5).

State Space—The four states t, h, v, and γ were used as the state space input to the DNN.

Action Space—The action space for DNN was to chose one of 11 values for α distributed evenly between $\alpha_{min} = -20^\circ$ and $\alpha_{max} = 20^\circ$. The α chosen by the DNN lasted for a 1-second time step, after which the state was observed and a new action was chosen.

Reward Function—After each 1-second time step, a reward was also calculated as given in eq. (7). The observed state recorded whether the episode had successfully completed the mission (done, success) or had violated constraints that caused the mission to end and fail (done, ¬ success). If after the 1-second time step the episode had not ended, the incremental reward was calculated (otherwise).

The reward function is calculated from $\bar{t}, \bar{h}$, and $\bar{\gamma}$ as defined in eq. (6). The incremental reward was kept negative to incentivize achieving the goal in the minimum number of time steps, and is calculated from $\bar{h}$ to guide the vehicle to the destination altitude. The successful terminal reward is weighted between valuing minimum time and achieving $\gamma = 0$.

$$\boldsymbol{x_0} = \begin{bmatrix} h_0 \\ \theta_0 \\ v_0 \\ \gamma_0 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 30\text{ km} \\ 0^\circ \\ 3\text{ km/s} \\ 0^\circ \end{bmatrix}, \begin{bmatrix} 5\text{ km} \\ 0^\circ \\ 0.5\text{ km/s} \\ 2.5^\circ \end{bmatrix}\right) \tag{5}$$

$$\begin{aligned} \bar{h} &= \left|\frac{h - h_{target}}{h_0 - h_{target}}\right| \\ \bar{t} &= t/t_{max} \\ \bar{\gamma} &= |\gamma/5^\circ| \end{aligned} \tag{6}$$

$$\begin{cases} 80(1-\bar{t}) + 20(1-\bar{\gamma}) & \text{if done, success} \\ -100\bar{h} + 16(1-\bar{t}) + 4(1-\bar{\gamma}) & \text{if done, } \neg \text{ success} \\ -\bar{h} & \text{otherwise} \end{cases} \tag{7}$$

The *Optuna* package for Python was used to optimize the hyperparameters governing the PPO training [35]. Using 64 trials with 200,000 steps was sufficient to produce good hyperparameters for training. After optimizing the hyperparameters, the DNN was trained for 500,000 episodes.

Results of Training for Nominal Case

The results of the agent training using the reward defined in eq. (7) are shown in Figure 5. Once trained, the agent returns a cumulative reward of about +30, which is sufficient to develop a policy to feasibly accomplish the mission of descending the vehicle from an altitude of 30 km to an altitude of 3 km. Figure 6 shows the results for the nominal case where the vehicle starts at 30 km and the RL agent selects a sequence of α (angle of attack) commands to guide the vehicle to its target altitude. For more discussion on validation of the trained agent, see Section 5.

5. T&E FRAMEWORK IMPLEMENTATION AND RESULTS

In this section, we exercise the trained agent with RL T&E framework described earlier in Section 3. First, we perform robustness testing on the trained agent for examining the impact of variations on the RL performance. Second, we investigate the DNN with the XAI SHAP method to identify

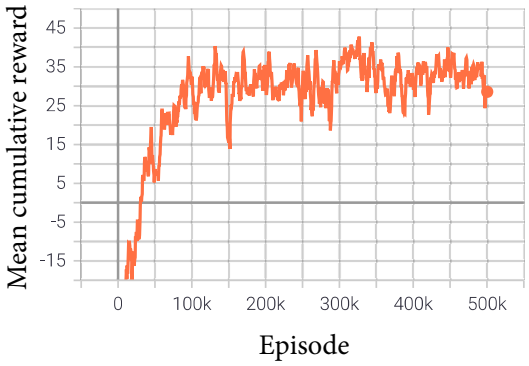


Figure 5. Reward curve for the training agent, calculated from eq. (7).

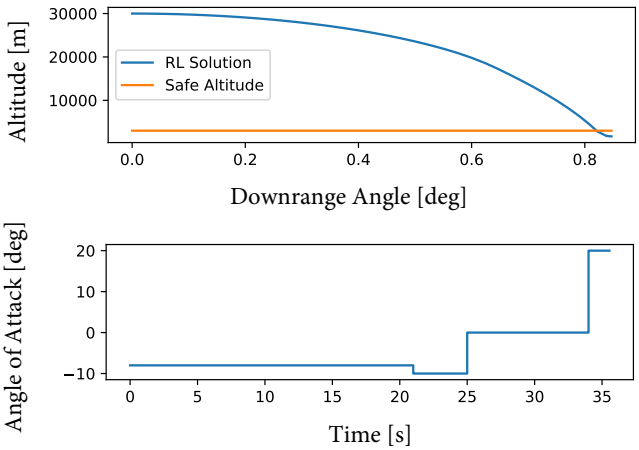


Figure 6. Trajectory from control history generated by the trained RL agent.

most influential inputs on RL agent's angle of attack selection. Finally, we validate the RL solution by comparing it to the trajectory generated by optimal control methods. The implementation details for T&E framework and results are described in the following paragraphs.

Robustness Testing Results

The Robustness Testing of the trained RL agent is performed by identifying sources of variation in the agent's environment model, action space, and observation state space. Table 5 provides details on the Robustness Testing tests cases (TCs) that were created to incorporate these variations based on the problem description and implementation details provided in Section 4.

TC-1: Individually Vary Environment ICs —In TC-1, the trained DNN is executed with different ICs for altitude, velocity, and FPA within the ranges of $\pm 10\%$, $\pm 20\%$, $\pm 30\%$, and $\pm 40\%$. Note that the objective function of the RL agent is to descend from 30 km to 3 km starting at 3 km/s velocity and 0° FPA. This TC examines the performance of RL agent outside these nominal ICs. Figure 7 provides the results for the $\pm 30\%$ variation in the ICs.

Table 2. Robustness test cases.

Test Cases	Objective
TC-1	Individually vary environment Initial Conditions (ICs) (i.e., altitude, velocity, and FPA) to examine RL performance
TC-2	Quantify performance bounds on ICs variations with LHS
TC-3	Sensitivity to impulses on the action space
TC-4	Sensitivity to random variations in the action space
TC-5	Sensitivity to impulses on the state space
TC-6	Sensitivity to random variations in the state space

From the velocity and FPA variation plots, it is evident that the vehicle is able to reach the target despite the variation in ICs. However, the altitude plot illustrates that for higher altitude values, the RL is unable to accomplish the mission. Figure 7 can be used to establish that the RL agent is able to perform across a variety of ICs but does fail under high altitude considerations. Now it becomes necessary to statistically establish the performance bounds individually on these ICs, which is the objective of TC-2.

TC-2: Varying Multiple ICs using LHS—In TC-2, we utilize LHS to generate a data set of varying ICs defined by eq. (8). The LHS method allows for testing a wider sample space with comparatively less number of samples by selectively identifying sample points and providing a comprehensive coverage. The LHS space is generated by using the *pyDOE* package in Python and using a normal distribution with the inputs from eq. (8) for altitude h, velocity v, and FPA γ and generating 50 samples. Figure 8 depicts the results of executing the trained DNN with the ICs specified by the LHS set with remarkable insights on DNN's performance (each cell shows percentage of successfully completed missions out of 50 missions). Figure 8 clearly shows that among the three ICs, the DNN is most sensitive to variation in altitude. The DNN successfully completes 100% mission for only 5% variation in altitude magnitude, whereas it is able to tolerate up to 10% and 20% variation in FPA and velocity magnitudes, respectively. A comprehensive analysis of Figure 8 will be instrumental in identifying the limitations of the trained DNN (Figure 9 illustrates selected failed mission trajectories based on altitude variations).

$$\boldsymbol{x_0} = \begin{bmatrix} h_0 \\ v_0 \\ \gamma_0 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 30\text{ km} \\ 3\text{ km/s} \\ 0^\circ \end{bmatrix}, \begin{bmatrix} 8\text{ km} \\ 0.8\text{ km/s} \\ 5.33^\circ \end{bmatrix} \right) \tag{8}$$

TC-3: Impulse Injection in the Action Space—This test aimed to identify the sensitivity of the DNN to constant action impulse changes. In this case, the prediction from the model was overwritten and instead a new command was given to the vehicle as an impulse with a 5-seconds duration. The vehicle was able to reach the target when the impulse was introduced before the first 20 seconds. Figure 10 shows the case where the vehicle was unable to reach the target, the impulses occur after 25 seconds. It is important to note the angle of attack command values frequently change after 20 seconds (see Figure 6), which may make the impact of disturbances particularly detrimental.

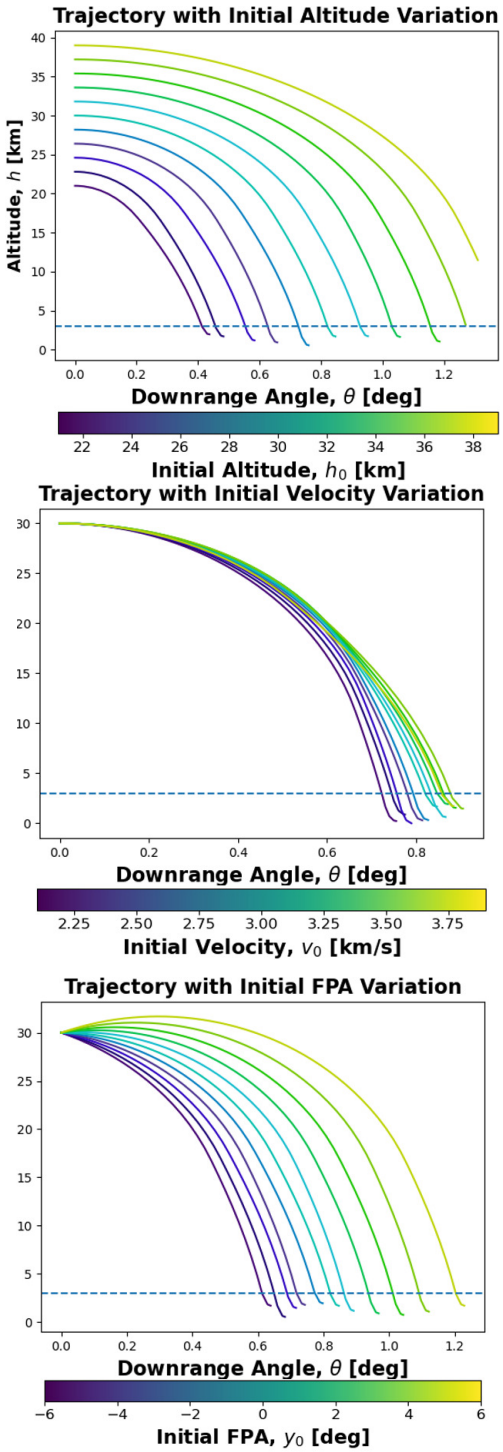


Figure 7. Initial condition variations in altitude, velocity and FPA.

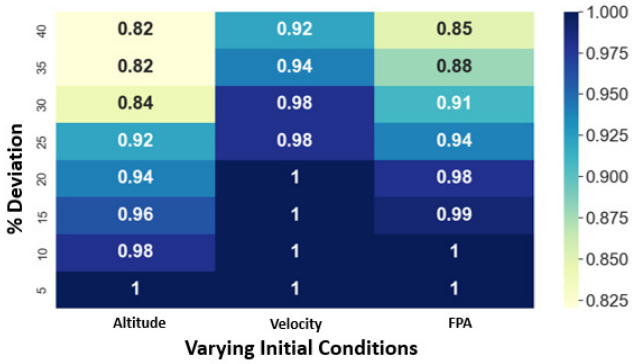


Figure 8. L-H heat map.

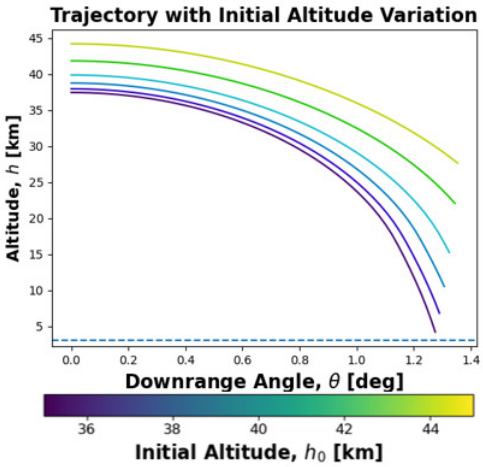


Figure 9. Failed missions with initial condition variations in high altitude values using LHS.

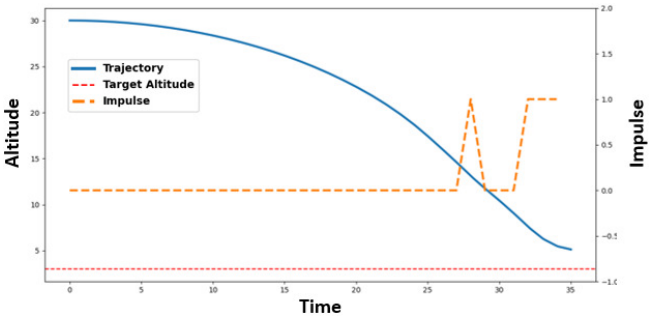


Figure 10. Action impulse injection.

TC-4 Angle of Attack Tolerance—In order to determine the angle of attack tolerance of the vehicle, the angle of attack command options needed to be expanded. This was done by allowing the model to command an angle of attack and then randomly choosing a value between $\pm 2^\circ$ and then later to $\pm 4^\circ$. When modifying the angle of attack command within the range of $\pm 2^\circ$, it did not impact the vehicle's ability to descend to the target altitude. Figure 11 shows results where the angle of attack is not modified (i.e., the nominal case) and is able to descend to the target successfully. Results in Figure 12 show the failed case generated from where the angle of attack values were randomly chosen to be within $\pm 4^\circ$ of the intended angle of attack command. This failure was observed after 1000 different runs, where the angle of attack command was modified to be within $\pm 4^\circ$ Hence, the tolerance of the angle of attack command is within $\pm 4^\circ$.

TC-5 Impulse in the State Space—In this test the observation data for the altitude was injected with a constant impulse at random intervals during the vehicle descent. These observation disturbances were introduced for 10 seconds and were

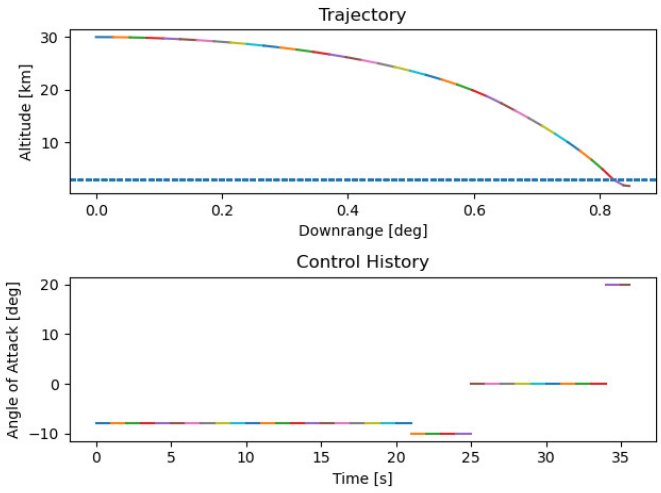


Figure 11. Vehicle trajectory for unmodified angle of attack.

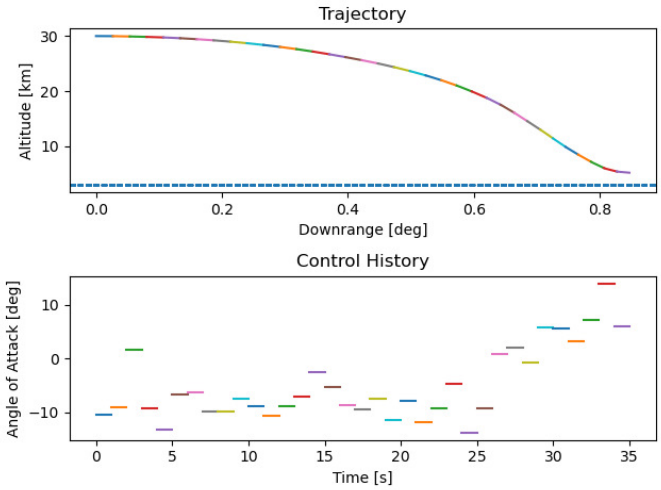


Figure 12. Vehicle trajectory for perturbed angle of attack.

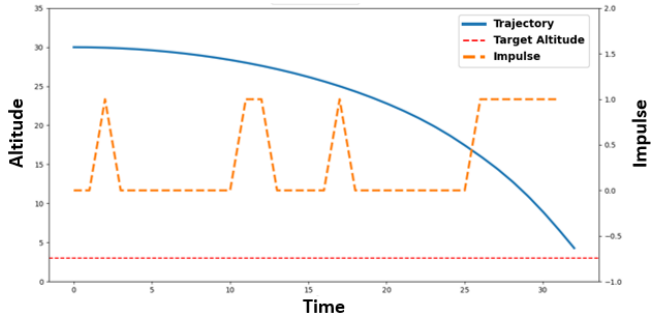


Figure 13. Altitude disturbance.

most significant impact on the RL output. The reason for this discrepancy is how these TCs are modeled. As described in the test case formulation, the state space observations are only perturbed on the RL state input (e.g., modeling an erroneous sensor), whereas the ICs are changed in the environment and directly effect the vehicle dynamics. Therefore, a potential mismatch is created between the RL reward value (calculated based on a true state) and the observed state by the agent (this is also evident by the vehicle's trajectory plot in Figure 13 and Figure 14). In future, we plan to expand these test cases to include state space disturbance modeling directly in the vehicle dynamics.

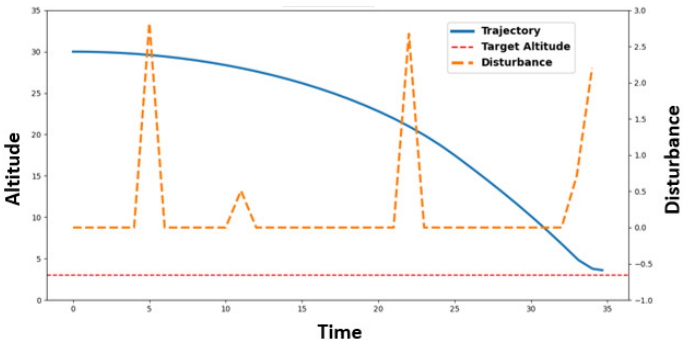


Figure 14. Random Altitude Disturbance

accomplished by overwriting the observation data for the altitude. It was observed that the altitude impulse presented no noticeable changes in the vehicle's ability to reach the target when the perturbation was solely present within the first 20 seconds, if the altitude impulse was less than twice the observed altitude. When the altitude observation disturbance occurred within the 20-40 seconds mark and lasted at least 10 seconds, a mission failure was observed. Figure 13 depicts the case where the vehicle was unable to descend to the target altitude.

TC-6 Random Variation in the State Space—This is an extension of the prior TC but rather than introducing a constant impulse, random impulses are introduced during the run. These disturbances range from 0.5 to 3 times the observed altitude and were injected at random times throughout the run and last for 5 seconds. Figure 14 show the case where the vehicle was unable to descend to the target altitude successfully.

A note on TC-5 and TC-6: These two test cases counter intuitively highlight that altitude disturbances injected in the state space in the form of impulse or random variation has limited impact on the RL output despite the altitude IC having the

Explainable AI Results

The SHAP results were generated from around 39,000 data points obtained from 1,000 trajectories. As previously mentioned, a Kernel SHAP explainer was employed. For testing explainability, 999 trajectories were chosen as the training set and 1 feasible trajectory was chosen as the testing set. The entire input data for SHAP was normalized between 0 and 1.

The SHAP summary plot is shown in Figure 15. The most important feature is altitude, which is shown at the top of Figure 15. The least significant feature is FPA and is shown at the bottom.

Note that there can be many different feature values that can result in the same SHAP value for that particular feature. This means that there is a non-linear interaction between that feature and the remaining features. The SHAP summary plot shows a spread of feature values and the corresponding SHAP values. The values of features are represented by different colors. Red color indicates high values, blue indicates low values and purple color shows medium values for the features. Negative SHAP value indicates the feature is negatively impacting the outcome or the action chosen by the RL agent and

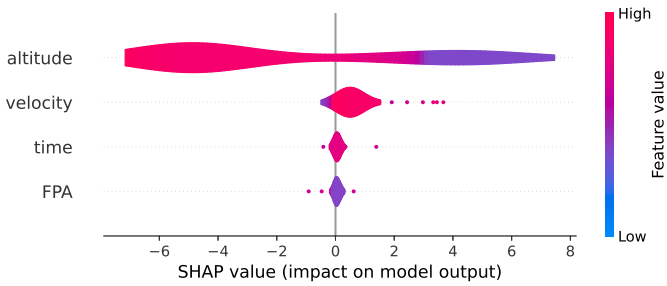


Figure 15. SHAP summary plot.

vice versa.

It is not intuitive at first which of the features is most important. SHAP helps unveil the fact that altitude impacts the decisions made by the RL agent the most, followed by velocity. Although this is an overall observation, for a particular point in the trajectory altitude might not be the most important feature. This is captured in the following discussion.

One particular test trajectory was chosen from the 1,000 trajectories in the data-set. This trajectory, as shown in Figure 16, is feasible.

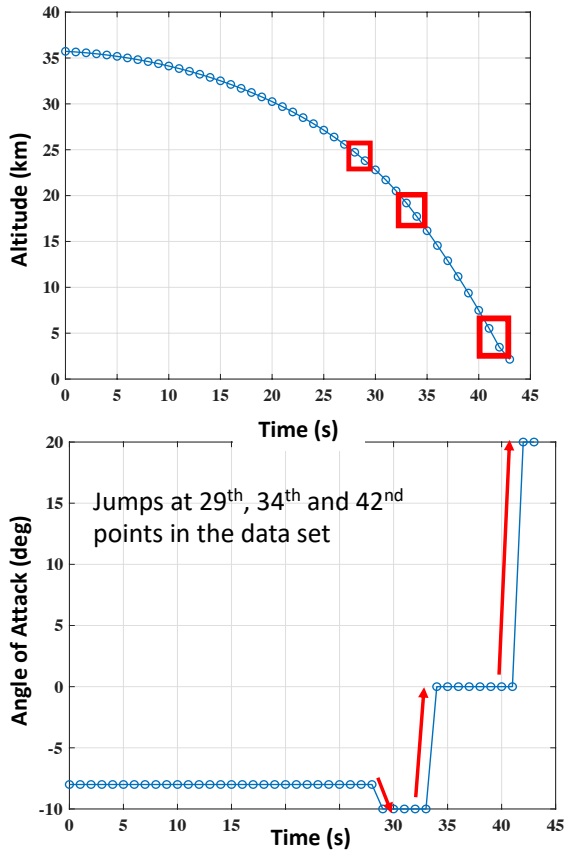


Figure 16. A chosen feasible trajectory for explaining the results obtained using SHAP along with the jump points. The corresponding angle of attack profile is also included.

The chosen trajectory has jumps occurring at node points 29, 34 and 42, which are shown in Figure 16. The corresponding SHAP values are shown in Table 3.

Table 3. SHAP values of different features corresponding to jump points in control profile of the chosen trajectory.

Data Point #	Angle of Attack (deg)	SHAP Values			
		Time	Altitude	Velocity	FPA
29	-8	0.3431	1.2848	3.6734	0.2792
30	-10	0.1581	1.8660	3.4531	0.1032
33	-10	0.1613	3.3503	1.9171	0.1517
34	0	0.1488	3.8376	1.4204	0.1736
41	0	0.1330	5.2111	0.1003	0.1362
42	20	0.1116	5.3742	-0.0201	0.1147

Although no immediate trend can be seen in the SHAP values for the different features at the jump points, certain interesting observations can be made. The dive from -8° to -10° between points 28 and 29 is supported more by the velocity as compared to the altitude. For the remaining two jump points, altitude is strongly supportive of gaining higher angle of attack values to reach the final altitude of 3 km as further diving could lead to violating this terminal altitude constraint. Thus, the altitude is most important feature for most parts of the trajectory, but not for all the points in the trajectory. At certain parts, velocity dominates the decisions made by the RL agent. The time and FPA features have feeble impact on the RL agent's decisions as is captured in Table 3.

Optimal Validation Results

The trajectory generated by the RL is validated in two ways. First, the trajectory is validated by satisfying the boundary conditions given in eq. (4). For the terminal boundary conditions, we treat $h - 3\,\mathrm{km}$ as an inequality constraint $h \leq 3\,\mathrm{km}$ and we allow a tolerance of $\pm 5^\circ$ for γ. Satisfying the boundary conditions ensures that the RL-derived trajectory is feasible. Second, the RL-derived trajectory is compared to the optimal solution solved with indirect optimization via the beluga package in Python [36]. In the indirect optimization, however, the action space for the control input α is continuous and is changed continuously instead of in 1-second increments. The trajectories are therefore not expected to be identical, but similarity between the two trajectories ensures the RL-derived trajectory is near optimal.

Figure 17 shows the optimal solution with the one generated by the RL trained network for the nominal case. The optimal final time is 27.1 seconds. The optimal control profile starts with the angle-of-attack at its minimum value of -20° for more than half of the trajectory to descend as fast as possible. Then, the angle of attack increases up to the maximum angle of attack of 20° so that the vehicle levels out to satisfy the terminal flight path angle constraints. The RL solution satisfies the constraints, but takes a noticeably higher time of 35.5 seconds as a result of the more moderate control in the beginning of the trajectory.

Discussion

The three-fold T&E framework used to evaluate the emergency descent problem demonstrates not only that the agent achieves the mission, but also characterizes its performance in a novel way. Comparison with the trajectory generated via indirect optimal control demonstrates that the agent takes more time than the optimal solution in the nominal case (Figure 17). SHAP values incorporated into the T&E (Figure 15) help characterize why the agent is more or less

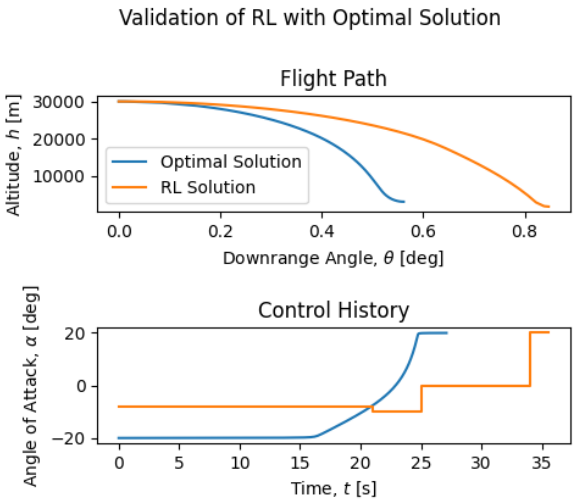


Figure 17. RL output compared to optimal solution obtained via traditional trajectory optimization method.

robust to different perturbations (Figure 8). In the case of the optimal descent model, it is shown that the altitude has by far the biggest impact on the control decisions made by the DNN. Correspondingly, the agent is far more sensitive to perturbation to altitude than it is to velocity or FPA.

The training and the T&E framework identify several key benefits of employing RL in this optimal descent and other similar aerospace problems. One benefit is that the DNN, unlike the exact indirect solution, adapts the control output given a perturbed state and the acceptable ranges/distributions of these perturbations can be quantified by the T&E framework. These ranges and distributions expose limitations of the RL agent and can help in setting performance requirements for other system elements such as lower-level flight control systems that implement actions or sensors that provide state information. Moreover, once trained, the DNN requires minimal computational resources to evaluate, thereby keeping the real-time application potential very strong. The trained DNN can be re-evaluated rapidly along the trajectory so that control inputs are generated real-time in a closed-loop manner. This is in contrast to implementing an exact indirect solution in open-loop. In training the DNN, the majority of the computational burden is off-line, where the model can be arbitrarily sophisticated. Conversely, implementing optimal control in a receding horizon framework [37], typically requires a computationally "light-weight" model for real-time execution.

6. Summary and Future Work

In this paper we have outlined the current state of RL research in practice for aerospace applications and noted a persistent deficiency in the treatment of performance characterization under uncertainty from the perspective of SE4AI. To this end, we have generated a sample problem and defined, then exercised a T&E methodology for RL applications. In particular, we have noted the types of variations that are likely to be present in this type of problem, shown the steps that can be taken to explain the decisions made by a trained DNN using RL, and compared the outcome to known and accepted solutions. This exercise has not been exhaustive since it may not necessarily transfer to all possible applications of RL in aerospace, but it has demonstrated the impact of variations that may otherwise have gone unexamined. It is our hope that the framework of evaluation proposed in this work stimulates further discussion in this area to continue effectively leveraging RL in real-time applications.

In the future, we intend to continue to refine this approach in an effort to inspire other researchers in RL to consider the benefits and limitations of their efforts. Determining relationship between SHAP values and optimal control theory is also part of the future work.

Acknowledgments

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multi-mission laboratory managed and operated by the National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This work describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

References

[1] R. Sutton and A. Barto, "The Reinforcement Learning Problem," in *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: The MIT Press, 2015.

[2] S. He, H.-S. Shin, and A. Tsourdos, "Computational Missile Guidance: A Deep Reinforcement Learning Approach," *Journal of Aerospace Information Systems*, vol. 18, no. 8, pp. 571–582, Aug. 2021, publisher: American Institute of Aeronautics and Astronautics. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.I010970

[3] B. Sun and E.-J. van Kampen, "Reinforcement-Learning-Based Adaptive Optimal Flight Control with Output Feedback and Input Constraints," *Journal of Guidance, Control, and Dynamics*, pp. 1–8, Jun. 2021. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.G005715

[4] S. Kim, J. Park, J.-K. Yun, and J. Seo, "Motion Planning by Reinforcement Learning for an Unmanned Aerial Vehicle in Virtual Open Space with Static Obstacles," in *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, Oct. 2020, pp. 784–787, iSSN: 2642-3901.

[5] T. McDermott, D. DeLaurentis, P. Beling, M. Blackburn, and M. Bone, "AI4SE and SE4AI: A Research Roadmap," *INSIGHT*, vol. 23, no. 1, pp. 8–14, 2020, https://onlinelibrary.wiley.com/doi/pdf/10.1002/inst.12278. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/inst.12278

[6] M. Stein, "Large sample properties of simulations using latin hypercube sampling," *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987.

[7] J. Antony, *Design of experiments for engineers and scientists*. Elsevier, 2014.

[8] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, Jun. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253519308103

[9] J. Junell, T. Mannucci, Y. Zhou, and E.-J. Van Kampen, "Self-tuning Gains of a Quadrotor using a Simple Model for Policy Gradient Reinforcement Learning," in *AIAA Guidance, Navigation, and Control Conference*, ser. AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, Jan. 2016. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2016-1387

[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[12] G. Wu, M. Fan, J. Shi, and Y. Feng, "Reinforcement Learning based Truck-and-Drone Coordinated Delivery," *IEEE Transactions on Artificial Intelligence*, pp. 1–1, 2021, conference Name: IEEE Transactions on Artificial Intelligence.

[13] L. Federici, B. Benedikter, and A. Zavoli, "Deep Learning Techniques for Autonomous Spacecraft Guidance During Proximity Operations," *Journal of Spacecraft and Rockets*, pp. 1–12, Aug. 2021, publisher: American Institute of Aeronautics and Astronautics. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.A35076

[14] D. Izzo and E. Öztürk, "Real-Time Guidance for Low-Thrust Transfers Using Deep Neural Networks," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 2, pp. 315–327, Feb. 2021, publisher: American Institute of Aeronautics and Astronautics. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.G005254

[15] S. McGuire, P. Michael Furlong, C. Heckman, S. Julier, and N. Ahmed, "Human-Aware Reinforcement Learning for Fault Recovery Using Contextual Gaussian Processes," *Journal of Aerospace Information Systems*, vol. 18, no. 7, pp. 429–441, Jul. 2021, publisher: American Institute of Aeronautics and Astronautics. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.I010921

[16] P. G. Buzzi, D. Selva, and M. S. Net, "Autonomous Delay Tolerant Network Management Using Reinforcement Learning," *Journal of Aerospace Information Systems*, vol. 18, no. 7, pp. 404–416, Jul. 2021, publisher: American Institute of Aeronautics and Astronautics. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.I010920

[17] G. S. Parnell, *Trade-Off Analytics: Creating and Exploring the System Tradespace*. Somerset, Uinted States: John Wiley & Sons, Incorporated, 2016.

[18] L. Xin, "Numerical Methods for Engineering Design and Optimization: Latin Hypercube Sampling (LHS)," 2014.

[19] J. C. Helton and F. J. Davis, "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems," *Reliability Engineering & System Safety*, vol. 81, no. 1, pp. 23–69, 2003, publisher: Elsevier.

[20] A. K. Raz, C. R. Kenley, and D. A. DeLaurentis, "System architecting and design space characterization," *Systems Engineering*, vol. 21, no. 3, pp. 227–242, May 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sys.21439

[21] A. Dachowicz, K. Mall, P. Balasubramani, A. Maheshwari, A. Raz, J. H. Panchal, and D. Delaurentis, "Mission Engineering and Design using Real-Time Strategy Games: An Explainable-AI Approach," *Journal of Mechanical Design*, pp. 1–15, 10 2021.

[22] D. A. DeLaurentis, J. H. Panchal, A. K. Raz, P. Balasubramani, A. Maheshwari, A. Dachowicz, and K. Mall, "Toward automated game balance: A systematic engineering design approach," in *3rd IEEE Conference on Games*. IEEE conference proceedings, 2021.

[23] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[24] S. Lundberg, "A game theoretic approach to explain the output of any machine learning model." Jan. 2021, original-date: 2016-11-22T19:17:08Z. [Online]. Available: https://github.com/slundberg/shap

[25] E. Puiutta and E. M. Veith, "Explainable reinforcement learning: A survey," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer, 2020, pp. 77–95.

[26] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," *Knowledge-Based Systems*, vol. 214, p. 106685, 2021.

[27] R. Leissner, "Do you want to train a simplified self-driving car with reinforcement learning?" 2020.

[28] J. Dohmen, R. Liessner, C. Friebel, and B. Bäker, "Longicontrol: A reinforcement learning environment for longitudinal vehicle control." in *ICAART (2)*, 2021, pp. 1030–1037.

[29] R. Leissner, "Dear reinforcement learning agent, please explain your actions." 2021.

[30] L. Shapley, "A value for n-person games,[in:] contributions to the theory of games ii, aw tucker, hw kuhn," 1953.

[31] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in neural information processing systems*, 2017, pp. 4765–4774.

[32] M. Ribeiro, S. Singh, and C. Guestrin, "Local interpretable model-agnostic explanations (lime): An introduction," 2019.

[33] G. Duan, Y. Sun, M. Zhang, Z. Zhang, and X. Gao, "Aerodynamic coefficients models of hypersonic vehicle based on aero database," in *2010 First International Conference on Pervasive Computing, Signal Processing and Applications*, 2010, pp. 1001–1004.

[34] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," https://github.com/DLR-RM/stable-baselines3, 2019.

[35] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019.

[36] T. Antony, M. Grant, M. Sparapany, S. Nolan *et al.*, "Beluga," https://github.com/Rapid-Design-of-Systems-Laboratory/beluga/graphs/contributors, 2021.

[37] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

BIOGRAPHY

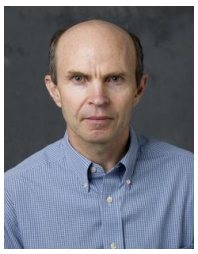
Ali K. Raz *is an Assistant Professor of Systems Engineering and Operations Research and an Assistant Director for C4I & Cyber Center at George Mason University. His research interests are in integration of intelligent aerospace systems and information fusion systems. Prior to joining Mason, he was a Visiting Assistant Professor in School of Aeronautics and Astronautics at Purdue University. He holds a BSc. and MSc. in Electrical Engineering from Iowa State University and a PhD. in Aeronautics and Astronautics from Purdue University. He is a senior member of both IEEE and AIAA.*

Sean Matthew Nolan *is a Ph.D. student and Research Assistant at Purdue University and AIAA student member. He received his B.S. in Aeronautical and Astronautical Engineering and M.S. Aeronautics and Astronautics from Purdue in 2015 and 2018 respectively. His work focuses on trajectory optimization using indirect methods and its applications.*

Winston Levin *is a M.S. student and research assistant at Purdue University in the Center for Integrated Systems in Aerospace (CISA). He earned his B.S. in Aeronautical and Astronautical Engineering from Purdue in 2020. His research assistance for CISA focuses on using reinforcement learning and indirect optimization for autonomy of high speed vehicle mission planning.*

Kshitij Mall *is a post-doctoral research associate at the Center for Integrated Systems in Aerospace, Purdue University. He obtained his Ph.D. and Masters degrees from the School of Aeronautics & Astronautics, Purdue University. He was a Post-doctoral Research Fellow in the department of Aerospace engineering at Auburn University in 2019. Previously, he completed B. Tech. in Mechanical Engineering at JSSATE Noida, India and then worked for a year at Infosys Technologies Ltd. as a Computer Systems Engineer Trainee. His research interests lie in the areas of Optimal Control Theory, Atmospheric Flight Mechanics, Explainable Artificial Intelligence, and Human-Class Mars missions. He is a member of AIAA.*

Ahmad Mia *is a Masters' student and research assistant at George Mason University at the Center of Excellence in Command, Control, Communications, Computing, Intelligence and Cyber (C4I and Cyber). He received his B.S. in Bioengineering from George Mason in 2017. His current research work focuses on assisting with performing robustness testing on reinforcement learning models.*



Linas Mockus *is Senior Research Scientist at Purdue University. His interests lie in the application of AI and Bayesian statistics to complex problems in Engineering. He received Ph.D. in Chemical Engineering from Purdue University, Ph.D. in Computer Science, M.Sc. and B.Sc. in Applied Mathematics from Moscow Institute of Physics and Technology.*

Kris Ezra *is a Research Scientist at Purdue University in the Center for Integrated Systems in Aerospace who specializes in modeling and simulation approaches for system-of-systems problems and software design and interoperability. He received his B.S. (2010), M.S. (2011), and Ph.D. (2015) degrees in Aeronautics and Astronautics from Purdue University under the direction of Dr. Daniel DeLaurentis.*

Kyle Williams *received the B.S. and M.S. degrees in Mechanical Engineering from Purdue University in 2005 and 2007. From 2010 to 2019 he was a Control Systems Engineer with Caterpillar, Inc. He received the Ph.D. degree from Purdue University in 2018 in Dynamic Systems, Measurement and Control. Since 2019 he has been a Principal Member of the Technical Staff with Sandia National Labs. His current research interests focus on enhanced control and autonomy for aerospace systems.*