Received 28 June 2023, accepted 28 July 2023, date of publication 3 August 2023, date of current version 10 August 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3301726*

RESEARCH ARTICLE

# Cascaded Deep Reinforcement Learning-Based Multi-Revolution Low-Thrust Spacecraft Orbit-Transfer

**SYED MUHAMMAD TALHA ZAIDI[1], PARDHA SAI CHADALAVADA[2], (Student Member, IEEE), HAYAT ULLAH[1], ARSLAN MUNIR[1], (Senior Member, IEEE), AND ATRI DUTTA[2], (Member, IEEE)**

[1]Department of Computer Science, Kansas State University, Manhattan, KS 66506, USA
[2]Department of Aerospace Engineering, Wichita State University, Wichita, KS 67260, USA

Corresponding author: Arslan Munir (amunir@ksu.edu)

This work was supported in part by the National Aeronautics and Space Administration (NASA) under Grant NASA-20-2020EPSCoR-0017.

**ABSTRACT** Transferring an all-electric spacecraft from a launch injection orbit to the geosynchronous equatorial orbit (GEO) using a low thrust propulsion system presents a significant challenge due to the long transfer time typically spanning several months. To address the challenge of determining such long time-scale orbit-raising maneuvers to GEO, this paper presents a novel technique to compute transfers starting from geostationary transfer orbit (GTO) and super-GTO. The transfer is complex, involving multiple eclipses and revolutions. To tackle this challenge, we introduce a cascaded deep reinforcement learning (DRL) model to guide a low-thrust spacecraft towards the desired orbit by determining an appropriate thrust direction at each state. To ensure mission requirements, a gradient-aided reward function incorporating the orbital elements, guides the DRL agent to obtain the optimal flight time. The obtained results demonstrate that our proposed approach yields optimal or near-optimal time-efficient spacecraft orbit-raising. DRL implementation is important for spacecraft autonomy; in this context, we demonstrate that our DRL-based trajectory planning provides significantly better transfer time as compared to state-of-the-art approaches that allow for automated trajectory computation.

**INDEX TERMS** Deep reinforcement learning, cascaded reinforcement learning, soft actor-critic algorithm, spacecraft orbit-transfer, solar-electric propulsion, optimization, low-thrust orbit-raising.

## I. INTRODUCTION

Over the last two decades, autonomy has gained considerable attention within the space industry. Autonomous decision-making onboard a spacecraft can significantly improve the flexibility of space missions, as well as responsiveness to unplanned or uncertain events. Onboard trajectory planning is one of such autonomous operations that can have great value for space mission planning and operations; especially, it reduces the reliance on ground personnel support for mission operations that involve complex maneuvering by the spacecraft. The utilization of a computationally efficient onboard autonomous guidance system can assist the spacecraft during the low-thrust multi-revolution orbit-transfer in the context of trajectory adjustment/optimization.

It is common for Earth-orbiting satellites to be equipped with chemical propulsion systems for the purpose of conducting orbital transfers and station-keeping. With the advent of solar-electric propulsion systems, many satellite operators have been shifting to satellites that are capable of conducting all orbital maneuvers (transfers as well as station-keeping) using electric thrusters. The development of such all-electric propulsion architectures by Boeing and Airbus have allowed the manufacture of all-electric satellites. Conducting large

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

transfers such as orbit-raising is challenging, primarily due to the low thrust levels produced by electric propulsion devices. Low thrust implies large transfer times, often of the order of months to reach geosynchronous equatorial orbit (GEO) starting from a geosynchronous transfer orbit (GTO). The propulsion system derives power from the solar arrays, and the existence of eclipses along the satellite's path prohibits thrusting during passage through the Earth's shadow. Furthermore, solar array degradation during passage through Van Allen belts means that thrust capability of the propulsion system diminishes during the transfer, which in turn increases the time of an already long transfer. Computation of an optimal orbit-raising trajectory is a challenging problem because it requires the solution of a nonlinear, nonconvex, multi-phase optimal control problem.

The trajectory optimization for low-thrust multi-revolution transfer has been actively explored over the years, with numerous methods proposed to determine optimal or sub-optimal solutions to the underlying optimal control problem [1], [2], [3], [4], [5], [6]. These methods can be broadly categorized into direct [7], [8] and indirect [9], [10] optimization techniques. Direct optimization methods discretize continuous trajectories into short segments, transforming the original continuous optimization problem into a discrete nonlinear one [11]. On the other hand, indirect optimization methods utilize calculus of variations to determine the necessary conditions of optimality and formulate a two-point boundary value problem [12]. For the computation of low-thrust trajectories, two primary approaches are commonly used: the shooting technique [13] and direct collocation [14], [15]. However, these traditional optimization-based approaches heavily rely on high-quality initial guesses provided by trained mission designers, and therefore are not suitable for the development of fully automated trajectory planning solvers. In a recent work by Sreesawet and Dutta [16], an automated solver has been developed that eliminates the need for user-provided initial guess solutions making it suitable for automated computation of low-thrust orbit-raising trajectories. Nonetheless, this approach sacrifices optimality in the solutions. To this end, this paper seeks to investigate an alternate method of automated orbit-raising trajectory planning by using a deep reinforcement learning (DRL) framework.

The remarkable success of machine learning (ML) and deep learning (DL) over the years have gained the attention of aerospace community for the applications of efficient autonomous and real-time guidance systems for next generation spacecrafts. The ML/DL-based autonomous guidance systems have the potential to replace traditional control and guidance methods by providing real-time on-board guidance and control functionalities to spacecraft during mission. Having these motivations in mind, several ML/DL-based approaches are proposed for spacecraft trajectory optimization [17], [18], [19], [20], prediction [21], [22], and guidance [23], [24]. These ML/DL models are usually trained offline (i.e., on the ground) to solve complex control optimal problems on the data, which is either provided by the aerospace experts or collected by the network itself through repeated simulations for target mission scenario. For instance, Zhu and Luo [18] presented a learning-based method for quick evaluation of low-thrust transfer. First, they developed an efficient data-generation method to obtain optimal transfer data and then they trained the multilayer perceptron classifier and regressor on the obtained data to estimate the optimal fuel consumption and feasibility of the transfer. Arora and Dutta [25] developed a deep neural network (DNN) based optimization framework to solve a low-level optimization problem, where the DNN framework was designed to adjust the weights of an objective function for the low-thrust orbit rising problem. In another study, Li et al. [20] proposed a neural network-based approach to estimate the initial costates, optimal time, and the optimal control law for trajectory optimization in time-optimal low-thrust orbital transfer.

To model the underlying orbit prediction (OP) errors from the collected historical observations, Li et al. [22] proposed an ML-based approach, that first identified the underlaying OP errors and then applied it to the future physics-based OP results for correction. Their method consisted of three steps: construction of historical OP error set, training of ML algorithm on constructed historical OP error set, and correction of future physics-based OP results. Peng and Bai [26] also extensively researched the field using support vector machines (SVM) to enhance simplified general perturbations model 4 (SGP4) based prediction by learning historical errors. Their error-correcting model demonstrated improved OP for resident space object (RSOs) in simulated catalogs. They also compared artificial neural networks (ANN), Gaussian processes (GP), and SVM models [21], [26], [27] and found that ANN achieved the best results, despite overfitting risks. Furthermore, they evaluated SVM on a real dataset [28] and proposed a data fusion approach [29] to combine uncertainty information from the extended Kalman filter orbit determination process and the Gaussian process model. Mughal et al. [30] designed and evaluated an ML framework, focusing on DNNs, to predict the transfer time of spacecraft instead of solving traditional orbit-raising optimization problems for each of the mission scenarios. Their experimental results indicated that their designed DNNs could predict the transfer time for different scenarios with an accuracy of over 99.97%. Izzo and Öztürk [24] proposed a DL-based real-time guidance system called backward generation of optimal examples for low-thrust transfer. They claimed that their method generated data which is in the orders of magnitude and contained optimal trajectories, that helped the neural network to estimate the optimal thrust. Caldas and Soare [31] provided a brief overview of the application of ML in orbit determination, OP, and atmospheric density modeling to enhance accuracy in tracking and predicting orbits for RSOs. Although these ML/DL based approaches have shown some improvements over traditional trajectory optimization methods for low-thrust multi-revolution orbit transfer, there are several limitations of these methods which make them unsuitable for reliable and robust trajectory design. For instance, ML/DL methods work well when they are used for an optimization problem that falls inside the set of incorporated expert demonstration in training but fail to perform well when they are used for an optimization problem that has to perform outside the training realm.

To overcome the limitations of standard ML/DL methods, several reinforcement learning (RL) based approaches are presented to solve the spacecraft dynamic problems for low-thrust multi-revolution trajectory design [32], [33], trajectory optimization for cislunar environment [34], [35], and development of autonomous guidance system for docking and rendezvous maneuvers [36], [37]. For instance, Miller and Linares [32] proposed an RL-based optimal control system for low-thrust orbit transfer. They developed a complex policy by training a Proximal Policy Optimization (PPO) to solve an optimal control system. Scorsoglio et al. [34] presented an RL-based feedback guidance approach to deal with docking maneuvers in a cislunar environment. They developed an algorithm in the circular restricted three-body problem framework for the near-rectilinear halo orbits (NRHOs) scenario in the earth-moon system. In another study, Hovell and Ulrich [37] proposed a learning-based method called Deep Guidance. Their method was based on an RL framework which iteratively learned guidance strategies rather than using predefined strategies. They combined control theory with the deep reinforcement learning (DRL) to reduce the learning overhead and enable the transfer of trained RL model from simulation to reality. Continuing research efforts in this direction, Gaudet etl al. [38] presented an RL-based integrated navigation, guidance, and control system for the six degrees of freedom (6)-DOF) planetary landing. They trained RL algorithm to learn policy mapping for lander's estimated state and commanded thrust for each engine. They claimed that their method obtained accurate and fuel-efficient trajectories for a realistic deployment ellipse. Holt et al. [6] proposed an actor-critic RL framework to make the control parameters of the Lyapunov-based Q-law state-dependent. Their method enabled the controller to adapt itself as the dynamics vary during transfer. The continuous adaptation of the controller thus helped to provide an improved and acceptable solution for mission analysis. All these RL-based studies have shown reasonable improvements over the standard ML/DL approaches for trajectory optimization problem. However, most of the existing RL-based methods are based on traditional mathematical models, such as Q-law guidance. The dependency of current RL-based optimization methods on traditional models make them unfeasible to exploit the model-free characteristics of RL. Kwon et al. [39] implemented a single reinforcement learning (RL) agent to optimize the transfer time of an electric spacecraft. Despite utilizing the model-free RL approach, their method did not achieve tolerances that are similar to those obtained by traditional approaches. This means that the transfer would end further away from GEO, and will also underestimate the time required for the transfer. Even then, their obtained transfer time for GTO-GEO transfer was sub-optimal compared to the literature.

To overcome the limitations of contemporary deep learning and RL-based approaches for spacecraft trajectory optimization for the orbit-raising problem, we present a novel approach for improving low-thrust multi-revolution orbit transfer by proposing a cascaded reinforcement learning method that employs the state-of-the-art soft actor-critic (SAC) algorithm, which is a model-free RL method and learns from experience rather than pre-existing datasets. The model starts learning from the initial predefined values of the underlying dynamic model and iteratively adjust these values during training. This model is publically available at GitHub.[1] The key contributions of our work are as follows:

1) We propose a novel cascaded deep reinforcement learning approach that can cascade multiple DRLs to help enable a mission designer to achieve a higher degree of precision in reaching the target position as compared to the traditional approaches.
2) We have introduced a gradient-aided reward function for the DRL that utilizes orbital elements to provide necessary feedback to the DRL agent to attain minimum-time low-thrust orbit-raising trajectories to GEO.
3) Results indicate that our proposed cascaded DRL approach minimizes the transfer time of spacecraft during low-thrust electric orbit-raising scenarios, specifically in the transition from GTO to GEO. The results show that the approach achieves optimal performance in GTO to GEO transfer scenario. Furthermore, our approach yields near-optimal results in the transition from Super-GTO to GEO.

The subsequent sections of this paper are organized in the following manner. Section II provides an introduction to reinforcement learning and explains the proposed cascaded deep reinforcement learning method. It also discusses the soft actor-critic algorithm which is utilized in the method. In Section III, we comprehensively explain the dynamic model and implementation framework as part of the methodology. Section IV presents the experimental results for GTO to GEO and Super-GTO to GEO transfer scenarios. Section V provides a discussion on experimental results as well as comparative analysis of different transfer scenarios. Lastly, Section VI concludes this paper and also provides directions for future research.

## II. REINFORCEMENT LEARNING

RL is a special type of ML, which learns to make a sequence of decisions and implement these decisions by taking an action at each decision epoch to perform a given task. Unlike

[1]https://github.com/talhazaidi13/Cascaded-Deep-Reinforcement-Learning-Based-Multi-Revolution-Low-Thrust-Spacecraft-Orbit-Transfer.git

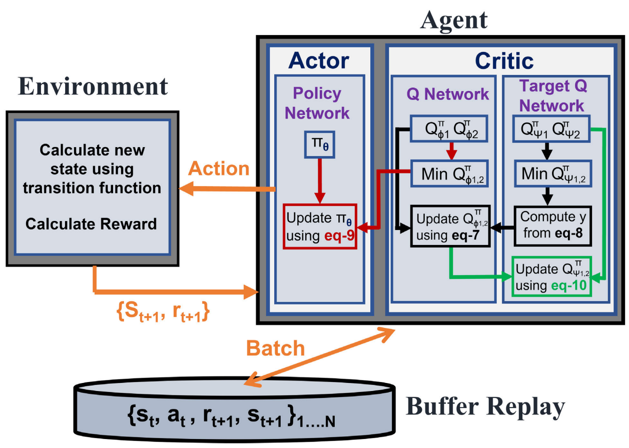

Agent
Actor
Critic
Environment
Policy Network
Q Network
Target Q Network
$\pi_\theta$
$Q^\pi_{\phi 1}$ $Q^\pi_{\phi 2}$
$Q^\pi_{\Psi 1}$ $Q^\pi_{\Psi 2}$
Min $Q^\pi_{\phi 1,2}$
Min $Q^\pi_{\Psi 1,2}$
Calculate new state using transition function
Calculate Reward
Action
Update $\pi_\theta$ using eq-9
Update $Q^\pi_{\phi 1,2}$ using eq-7
Compute y from eq-8
Update $Q^\pi_{\Psi 1,2}$ using eq-10
$\{S_{t+1}, r_{t+1}\}$
Batch
$\{s_t, a_t, r_{t+1}, s_{t+1}\}_{1....N}$
Buffer Replay

**FIGURE 1.** Soft actor-critic algorithm. $Q^\pi_{\phi_1}$ and $Q^\pi_{\phi_2}$ are two Q-networks, $Q^\pi_{\psi_1}$ and $Q^\pi_{\psi_2}$ are two target Q-networks, and $\pi_\theta$ represents policy network. The figure represents the workflow of an agent networks' update and its interaction with the environment and buffer replay.

supervised learning, RL does not have access to the solution of the problem. Instead, it learns from its own experience by interacting with the environment, which is defined by a known or unknown dynamic model. RL can be defined as a Markov Decision Process (MDP) that consists of a five-tuple $\boldsymbol{S}, \boldsymbol{A}, \Gamma, \boldsymbol{R}, \gamma$ [40], [41]. The five-tuple includes a finite state called state space ($\boldsymbol{S}$), a finite set called the action space ($\boldsymbol{A}$), a transition function ($\Gamma : s_t \times a_t \rightarrow s_{t+1}$), which returns the subsequent state $s_{t+1} \in S$, the numeric reward value $r_{t+1} \in R$, and a discount factor ($\gamma$), which is a value between 0 and 1, and determines how much importance is to be given to immediate reward and future rewards.

The RL agent interacts with the environment by taking an action $a \in A$ at state $s_t \in S$ and obtains an immediate reward $r_t \in R(s, a)$ and transitions to a subsequent state $s' \sim \Gamma(s, a)$, at each time step [40]. This results in a sequence like $s_0, a_0, r_1, s_1, a_1, r_2, \ldots$.. The ultimate goal of the agent is to maximize the accumulated reward values in minimum time steps, where one time step refers to one interaction with the environment. The accumulated reward value is also known as the return. To achieve this goal of maximizing reward, the agent learns some strategies to maximize the reward value by selecting optimal actions. This strategy is known as the policy of the network and is denoted as $\pi(a|s)$. The policy can be deterministic or stochastic, but in RL problems with continuous state and action space, a stochastic policy is preferable.

In a stochastic policy, the policy function $\pi(a|s)$ maps the state spaces to the probability distribution of the action spaces. Mathematically, the policy function can be defined as

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s] \tag{1}$$

In deep reinforcement learning, this mapping is calculated through a deep neural network called the actor-network, which is used to sample actions at each time step. Mathematically, this can be shown as

$$a_t \sim \pi_\theta(.|s) \tag{2}$$

where $\theta$ represents the weights and biases of a neural network and $\sim$ indicates the sampling operation.

A value function or critic plays a crucial role in estimating the value of a given state or state-action pair. The value function evaluates the expected sum of rewards, which determines the importance of a state or state-action pair. Specifically, the action-value function $Q^\pi(s, a)$, estimates the expected sum of rewards when taking an action $a$ at a given state $s$, while following a stochastic policy $\pi$. The action-value function can be mathematically expressed as follows in Eq. (3):

$$Q^\pi_\phi(s, a) = \mathbb{E}\left[\sum_{\kappa=t}^{T} \gamma^{\kappa-t} R_{\kappa+1} | S_t = s, A_t = a\right] \tag{3}$$

where $T$ represents the time step at the end of the episode, and $\phi$ denotes the weights and biases of the critic network.

In actor-critic-based RL algorithms, both policy (actor) and value (critic) networks work collaboratively and are updated together at each training step. Specifically, the policy network is updated based on the estimated policy gradient to maximize the expected sum of rewards, while the value network is updated to minimize the mean squared Bellman error. The standard objective function is defined in eq(4):

$$J(\pi) = \mathbb{E}a_t \sim \pi(.|s_t)\left[\sum_t \gamma^t (r(s_t, a_t))\right] \tag{4}$$

where $J(\pi)$ represents the expected sum of rewards under policy $\pi$, $r(s_t, a_t)$ is the reward function, and $\gamma$ is the discount factor that determines how much importance is given to immediate and future rewards. The objective function is optimized by updating the policy and value networks iteratively until convergence.

## *A. SOFT ACTOR-CRITIC (SAC) ALGORITHM*

SAC is a cutting-edge off-policy algorithm which is widely recognized as one of the most effective RL algorithms for real-world applications. It is off-policy as it learns the target policy using a replay buffer containing trajectories from a different behavioral policy, unlike on-policy algorithms that use the same policy. The key innovation of SAC lies in providing a novel solution to the critical exploration-exploitation dilemma that has long plagued DRL algorithms. Specifically, SAC addresses this issue by introducing an entropy regularization term ($H$) into the objective function. In this way, the agent is not only driven to maximize lifetime rewards but also to maximize the entropy of the policy. By increasing the entropy of the policy, SAC encourages the agent to assign equal probabilities to actions that have similar or identical Q values. Consequently, the agent explores promising regions more thoroughly while avoiding overfitting to any quirks in the Q value. The modified objective function for SAC is

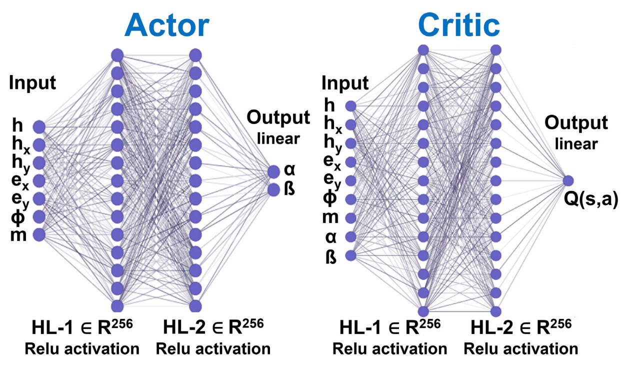


**FIGURE 2.** Actor-critic network parameters, where HL-1 and HL-2 represent hidden layers one and hidden layer two respectively.

defined in Eq. (5).

$$J(\pi) = \mathbb{E}_{a_t \sim \pi(.|s_t)} \left[ \sum_t \gamma^t (r(s_t, a_t) + \alpha \mathbb{H}(\pi(.|s_t)) \right] \tag{5}$$

The variable $\alpha$ is a control coefficient that ranges from 0 to 1, and it is used to set the level of entropy in the policy. The entropy, denoted by $\mathbb{H}$, can be calculated using the following formula:

$$\mathbb{H}(x) = -\mathbb{E}\left[log_2 P(x)\right] = -\sum_i P(x_i) log_2 P(x_i) \tag{6}$$

To achieve optimal performance of the objective function, the SAC algorithm leverages three distinct networks: a policy (actor) network that is parameterized by $\theta$, two Q-functions (critic) that are parameterized by $\phi_1$ and $\phi_2$, and two *target* Q-functions that are parameterized by $\psi_1$ and $\psi_2$. The use of two Q-functions addresses the well-known issue of overestimation of Q-values, by utilizing the minimum of these two for policy function updates. This approach ensures a reliable estimation of Q-values and reduces the risk of overestimation.

The loss function for the Q-networks is defined in Eq. (7)

$$\mathbb{L}_{\phi_i} = \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left( Q_{\phi_i}(s, a) - y(r, s', d) \right)^2 \tag{7}$$

where $B$ is the batch of dataset stored in the experience replay and $y(r, s', d)$ is calculated through the target Q networks $Q_\psi$ in eq (8)

$$y(r, s', d) = r + \gamma(1-d) \left( \min_{i=1,2} Q_{\psi_i}(s', a'_\theta) - \alpha \log \pi(a'_\theta|s) \right), \quad a'_\theta \sim \pi_\theta(\cdot|s') \tag{8}$$

The parameter $d$ in the equation denotes a *done* parameter, that takes a value of one when the episode has reached the terminal state and zero otherwise. Action $a'_\theta$ is sampled from the policy $\pi_\theta$ having next state $s'$ as input. During the sampling process, independent Gaussian noise and tanh functions are employed to approximate the gradient of Eq. (5) with respect to the policy parameter $\theta$. The policy loss function can be described in Eq. (9).

$$\mathbb{L}_\theta = \frac{1}{|B|} \sum_{s \in B} \left( \min_{j=1,2} Q_{\phi_j}(s, a_\theta(s)) - \alpha log \pi_\theta(a_\theta(s)|s) \right) \tag{9}$$

where $a_\theta(s) \sim \pi_\theta(\cdot|s)$ i.e. sampled from the policy $\pi_\theta$ having current state s as input. Target Q-networks ($\psi_k$) are utilized to improve training stability and are updated at a slower rate as compared to the Q-functions ($\phi_k$), as given in the following equation.

$$\psi_k \leftarrow p\psi_k + (1-p)\phi_k \quad for \;\; k = 1, 2 \tag{10}$$

where $p$ is a real number that can take values between 0 and 1, that is, $p \in [0, 1]$, and it controls the rate of change of the weights of target Q-networks ($\psi_k$). This whole algorithm is represented in Fig. 1.

## B. CASCADED REINFORCEMENT LEARNING

In the domain of reinforcement learning, solving complex problems can be accomplished through different approaches. We propose a novel approach for solving complex problems that utilizes multiple independent reinforcement learning agents sequentially to achieve the final solution, where each DRL agent solves a sub-problem that is a step towards the solution of the complex problem. We refer to this approach as *cascaded deep reinforcement learning*. The objective of this approach is to break down a complex task into simpler subtasks or intermediate goals that can be achieved more efficiently.

In this study, we develop a cascaded reinforcement learning approach using two SAC networks to achieve a target position with higher accuracy and resolution as compared to the single-agent approach, which is the simplest version. The first SAC network generates actions that allow the agent to achieve an intermediate goal, defined in such a way that it reaches the vicinity of the target position. This approach is particularly useful when the final target position is difficult to achieve directly, or when the state space is large and the agent needs to break down the task into simpler subtasks. The second SAC network takes the output of the first network as input and learns to execute this subtask to achieve the final target position with higher accuracy and resolution. The proposed approach can be extended to multiple SAC networks, but in this work we demonstrate it for two SACs, and compare it to a single-agent approach, which is the simplest version of RL.

One of the advantages of the proposed cascaded DRL approach is that it allows an agent to learn an efficient and robust policy by breaking down complex tasks into simpler subtasks. By focusing on achieving intermediate goals, the agent is able to explore the state space more efficiently and learn a better policy. Furthermore, the use of two SAC networks enables the agent to learn an accurate and fine-grained policy for achieving the final target position, as the second network can focus on achieving high-resolution goals. The use of multiple SAC networks in our cascaded reinforcement learning approach enables us to achieve a high

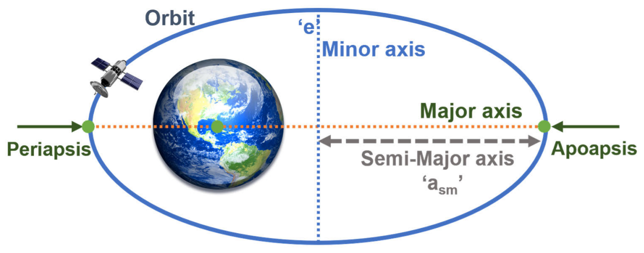


FIGURE 3. Semi-major axis '$a_{sm}$' and eccentricity '$e$' of an orbit.

resolution/precise solution that could not be achieved using a single SAC network.

However, there are also potential drawbacks to this approach. One challenge is designing a suitable decomposition of the task into subtasks or intermediate goals, which may not always be straightforward or intuitive. Additionally, the training of multiple DRL networks may increase the training time of the system. Finally, the effectiveness of this approach may depend on the specific task and the complexity of the state space. Despite these potential drawbacks, the cascaded DRL approach shows promise in tackling complex tasks in the domain of DRL.

## III. METHODOLOGY

To define the dynamics of the spacecraft and its orbital raising, our approach aligns with that of Sreesawet and Dutta. Reference [16]. Specifically, we adopt their technique of transforming the Earth-centred inertial (ECI) reference frame, denoted as $\mathcal{I}$(I,J,K), through a 2-1-3 Euler rotational sequence. In this sequence, the initial rotation is about the J-axis by an angle of $\zeta$, yielding the new reference frame denoted as $\mathcal{I}'$(I′, J′, K′). The subsequent rotation by an angle of $\eta$ about the first axis (I′) leads to the next reference frame, that is, $\mathcal{O}$(i′,j′,k′). Finally, the third rotation is about the third axis (k′) by an angle of $\phi$ to obtain the final frame of reference, $\mathcal{R}$, with a set of unit vectors $\mathcal{R}$(r′,n′,h′). In this frame, the first axis (r′) points towards the spacecraft while the third axis (h′) is aligned with the angular momentum vector $h$, which is perpendicular to the orbital plane. Figure 4 summarizes this rotation sequence. The directional cosine matrices for this rotational sequence are available in [16]. We use these matrices to convert the orbital elements or spacecraft kinematic description in Cartesian coordinates to the state description used in this paper.

### A. DYNAMIC MODEL

We model the spacecraft dynamics using the same state elements as defined by Sawet and Dutta. [16]. These state elements are presented in Eq. (11).

$$s = \begin{bmatrix} h & h_x & h_y & e_x & e_y & \phi \end{bmatrix} \tag{11}$$

here, $h$ denotes the magnitude of the specific angular momentum of the spacecraft, while $h_x$ and $h_y$ denote the x and y components of specific angular momentum along the I and J directions, respectively, in the Inertial frame of reference (I,J,K). On the other hand, $e_x$ and $e_y$ denote the components

**Algorithm 1** SAC Algorithm

```
1: Initialize networks parameters: $\phi_1$,$\phi_2$, $\theta$, $\psi_1$, $\psi_2$
2: Initialize experience replay: B
3: Training:
4: for each iteration do
5:     $a_t \sim \pi_\theta(a_t|s_t)$
6:     $seg \sim$ Segment_Size_Select($e$,$i$,$a_{sm}$)
7:     $s_{t+1}, r_{t+1}, d \sim$ Env($s_t, r_t, seg$)
8:     $B \longleftarrow BU(s_t, a_t, r_{t+1}, s_{t+1}, d)$
9:     for 0,1,2,3,...,N training updates do
10:        Sample batch from experience Replay ($b \sim B$)
11:        Compute target values using Eq. (8)
12:        Update Q-functions by gradient ascent, using
               Eq. (7)
13:        Update policy function by gradient ascent, using
               Eq. (9)
14:        Update target networks by using Eq. (10)
15:    end for
16: end for
```

**Algorithm 2** Segment Size Select Algorithm

```
1: Input: $e$, $i$, $a_{sm}$
2: if $e \leq 0.01$ & $i \leq 0.1$ & $a_{sm} \in \{a_{sm}^{tar} \pm 200\ km\}$ then
3:     $seg = 0.1$
4: else if $e \leq 0.01$& $i \leq 0.1$ & $a_{sm} \in \{a_{sm}^{tar} \pm 2100\ km\}$
   then
5:     $seg = 1$
6: else
7:     $seg = 10$
8: end if
9: Output: Segment size (seg)
```

of eccentricity along the i′ and j′ directions in the second rotated frame of reference O(i′,j′,k′). The angle $\phi$ represents the true anomaly-like angle, which determines the position of the spacecraft.

The force model for the solar-electric propulsion spacecraft is provided in Eq. (12).

$$F = \frac{2\lambda P}{g_0 I_{sp}} \tag{12}$$

where P is the power available from the solar arrays to operate the electric thrusters, $\lambda$ is the efficiency of the propulsion system, $g_0$ is the acceleration due to gravity at the surface of the Earth and $I_{sp}$ is the specific impulse of the propulsion system. For the current dynamic model, the spacecraft keeps the constant thrust value throughout the transfer, except for the eclipse portion of the orbit. During the eclipse, solar-electric propulsion systems cannot generate power, unless there is some onboard energy storage mechanism available [42]. In the context of this paper, we consider zero thrusts during the eclipse. The direction of the thrust vector due to the onboard propulsion system is denoted as angles $\alpha$ and $\beta$, which are bounded in the ranges of $[-\pi,\pi]$ and

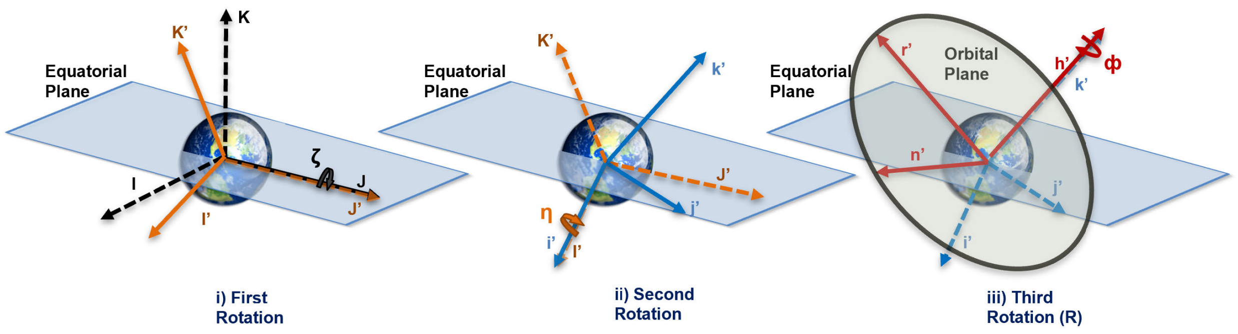


FIGURE 4. 2-1-3 Euler orbital rotation. (I′,J′,K′), (i′,j′,k′) and (r′,n′,h′) are x-axis, y-axis and z-axis representations after the first, second, and third orbital rotations, along y-axis, x-axis and z-axis, respectively.

**Algorithm 3** Next State and Reward Calculation (Env)

```
1: **Input:** $s_t$, $a_t$, seg
2: **Steps:**
3: **Compute** $s_{t+1} \sim P(s_{t+1}|s_t, a_t, r)$ using Eq. (13 - 15)
4: **Compute** reward ($r_{t+1}$) using Eq. (23 - 24)
5: **Compute** $d$ using Eq. (20 - 22)
6: **Output:** $s_{t+1}$, $r_{t+1}$, $d$
```

$[\frac{-\pi}{2}, \frac{\pi}{2}]$, respectively. Using these angles, the thrust components along the orbital plane are defined in the following equation (13):

$$u = \begin{bmatrix} F_r & F_n & F_h \end{bmatrix} = F \begin{bmatrix} -\sin\alpha\cos\beta \\ \cos\alpha\cos\beta \\ \cos\beta \end{bmatrix} \tag{13}$$

The transition function, which can provide the spacecraft transnational dynamics can be represented in the form of first-order differential equations as follow [16]:

$$\begin{bmatrix} \dot{h} \\ \dot{h}_x \\ \dot{h}_y \\ \dot{e}_x \\ \dot{e}_y \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \frac{\mu^2 \mathbf{B}^2}{\mathbf{h}^3} \end{bmatrix} \begin{bmatrix} h \\ h_x \\ h_y \\ e_x \\ e_y \\ \phi \end{bmatrix} + \mathbf{G} \begin{bmatrix} F_r \\ F_n \\ F_h \end{bmatrix} \tag{14}$$

The matrix G serves as a mathematical transformation that maps non-Keplerian acceleration terms to the rate of change of state variables. The matrix G is defined in Eq. (15), as shown at the bottom of the next page, with A and B representing mathematical expressions ($A = e_x \sin\phi - e_y \cos\phi$, $B = 1 + e_x \cos\phi + e_y \sin\phi$) derived from the eccentricity vector components, $e_x$ and $e_y$, and the true anomaly angle, $\phi$. Additionally, the earth's gravitational parameter, $\mu$, is used in the definition of G. By leveraging the state parameters specified in Eq. (11) and the thrust values outlined in Eq. (13), the change in state parameters can be calculated using equations (14) and (15).

### 1) DYNAMIC MODEL ASSUMPTIONS

As already mentioned, we consider the spacecraft thrust to be constant, unless the spacecraft is passing through the shadow of the Earth. In addition, we undertake the following modeling assumptions related to the dynamic model of the spacecraft: (1) The effect of orbital perturbations are neglected in this paper. While the force terms in Eq. (14) can represent the sum of any number of forces acting on the spacecraft (such as thrust, $J_2$ perturbation, and solar radiation pressure), we consider that the force is only due the onboard thrust. (2) The effect of radiation damage is neglected in this paper. In reality, the spacecraft solar arrays experience degradation during the transit through the Van Allen belts. Due to this degradation, the available thrust for the transfer deteriorates as well, as per Eq. (12). We ignore the radiation damage and assume that the thrust available during the Sun-lit part of the trajectory is constant. These assumptions allow us make a fair comparison with the sequential approach [16] and the previously studied DRL approach [39] in the literature. If one wishes to incorporate orbital perturbations in the problem, additive terms in Eq. (14) representing those perturbations need to be inserted. Similarly, if one wishes to incorporate the effect of radiation damage, one can incorporate artificial neural network based radiation damage prediction [42] within the framework.

### 2) ECLIPSE MODEL ASSUMPTIONS

As the spacecraft makes multi revolutions around the Earth to reach the final orbit using all-electric propulsion, it is highly probable that it passes through the Earth's shadow. When the spacecraft passes through the earth's shadow, it can use its onboard batteries to power the thrusters or turn them off and coast. This work assumes that the spacecraft coasts when it passes through the Earth's shadow. We need to have a shadow model to identify the regions where the spacecraft passes through the Earth's shadow. In this work, we adopt the cylindrical eclipse model to determine when the spacecraft is in Earth's shadow. In the cylindrical Earth shadow model, we assume that the shadow of Earth is cylindrical in shape, and we further assume that it is fixed in space and not moving. The conditions to check whether the spacecraft is in eclipse are given below:

$$X_I < 0, \tag{16}$$

$$\sqrt{Y_I^2 + Z_I^2} < R_E \tag{17}$$

where $X_I$, $Y_I$, and $Z_I$ are the components of the Cartesian position vector of the spacecraft in the Inertial frame, and $R_E$ is the radius of the Earth. The equations to convert the state vector of the spacecraft used in this work to Cartesian coordinates are discussed in [16].

### B. DRL MODELING AND IMPLEMENTATION DETAILS

The implementation framework for optimizing the space trajectory for the orbit-raising problem involves dividing the problem into two subtasks. The first subtask aims to carry the spacecraft in the vicinity of the target position, while the second subtask focuses on achieving the final target position with high precision. To accomplish this, two sequentially connected SAC reinforcement learning algorithms are implemented. The actor and critic networks follow standard architectural parameters, which are illustrated in Figure 2. The SAC algorithm is chosen due to its effectiveness in maximizing entropy, encouraging the agent to explore thoroughly during training and avoid early termination. Compared to other off-policy algorithms such as deep deterministic policy gradient (DDPG) and advantage actor-critic (A2C), SAC is less sensitive to hyperparameter tuning [43]. In addition, research has shown that SAC does not produce significant differences in output with different hyperparameters, making it a suitable choice for training agents in a dynamic model-free environment.

To implement this algorithm, we have modeled the MDP parameters, which include state, reward, action, next state, and training episode termination status ($\mathbf{M} = \{s_t, a_t, r_{t+1}, s_{t+1}, d\}$). The same elements are used to define the state, which is shown in Eq. (11), and Eq. (18).

$$s = \left[h \; h_x \; h_y \; e_x \; e_y \; \phi\right]^T \tag{18}$$

The action is defined as the two thrust angles, $\alpha$ and $\beta$, which were described in Section III-A and are used to calculate the thrust components in Eq. (13), as shown below:

$$a = \left[\alpha \; \beta\right]^T \tag{19}$$

Three terminal conditions are checked at every time step to determine if the spacecraft has reached the desired position. These conditions include the magnitudes of the semi-major axis($\mathrm{a_{sm}}$), eccentricity($e$), and inclination($i$). These parameters are calculated using the state parameters defined in Eq. (18).

The first condition checks the eccentricity of the spacecraft's orbit, as shown in Eq. (20). The magnitude of eccentricity is calculated from the $e_x$ and $e_y$ parameters, which are part of the state vector. Here $e_{tol}$ denotes the tolerance value for eccentricity.

$$0 \leq \left[e = \sqrt{e_x^2 + e_y^2}\right] \leq e_{tol} \tag{20}$$

The second condition checks the semi-major axis of the spacecraft's orbit, as shown in Eq. (21). The semi-major axis is calculated using the $h$, $e_x$, and $e_y$ parameters from the state vector, as well as the gravitational parameter $\mu$. The value of $\mathrm{a_{sm}^{tar}}$ is the desired target value for the semi-major axis, and $\mathrm{a_{sm}^{tol}}$ is the tolerance value for the semi-major axis.

$$\mathrm{a_{sm}^{tar}} - \mathrm{a_{sm}^{tol}} \leq \left[\mathrm{a_{sm}} = \frac{h^2}{\mu(1 - e_x^2 - e_y^2)}\right] \leq \mathrm{a_{sm}^{tar}} + \mathrm{a_{sm}^{tol}} \tag{21}$$

The third condition checks the inclination angle of the spacecraft's orbit, as shown in Eq. (22).

$$0 \leq \left[i = \sqrt{\frac{h_x^2 + h_y^2}{h}}\right] \leq i_{tol} \tag{22}$$

In the above equation, the inclination angle $i$ is calculated using the $h_x$ and $h_y$ parameters from the state vector, as well as the magnitude of the specific angular momentum $h$. Here, $i_{tol}$ denotes the tolerance value for the inclination angle.

By checking these three conditions at every time step, the algorithm can determine if the spacecraft has reached the desired target position or not.

A potential-based ($\phi(s') - \phi(s)$) shaping reward function is applied to train the DRL agents. This reward function includes a distance-based heuristic and a shaping function that contains the gradient information [39] of the parameter's error values. The shaping function is a negative-powered exponential factor that becomes active when the difference between the current and target parameters is sufficiently small. The reward function is defined in Eq. (23) and Eq. (24)

$$\mathbf{G} = \begin{bmatrix} 0 & \dfrac{h^2}{\mu m \mathbf{B}} & \dfrac{\mu^2 \mathbf{B}^2}{h^3} \\ 0 & \dfrac{h h_x}{\mu m \mathbf{B}} & \dfrac{h^2\sqrt{h^2-h_x^2-h_y^2}\,\mathbf{B}^2}{\mu m \mathbf{B}\sqrt{h^2-h_y^2}} \sin\phi + \dfrac{h h_x h_y}{\mu m \mathbf{B}\sqrt{h^2-h_y^2}} \cos\phi \\ 0 & \dfrac{h h_y}{\mu m \mathbf{B}} & -\dfrac{h\sqrt{h^2-h_y^2}}{\mu m \mathbf{B}} \cos\phi \\ \dfrac{h\sin\phi}{\mu m} & \dfrac{2h\cos\phi}{\mu m} + \dfrac{hA\sin\phi}{\mu m B} & \dfrac{h e_y h_y}{\mu m B\sqrt{h^2-h_y^2}} \sin\phi \\ -\dfrac{h\cos\phi}{\mu m} & \dfrac{2h\sin\phi}{\mu m} + \dfrac{hA\cos\phi}{\mu m B} & -\dfrac{h e_x h_y}{\mu m B\sqrt{h^2-h_y^2}} \sin\phi \\ 0 & 0 & -\dfrac{h h_y}{\mu m B\sqrt{h^2-h_y^2}} \sin\phi \end{bmatrix} \tag{15}$$

**TABLE 1.** State parameters defined for the GEO-orbit raising problem.

| Scenarios | Position | State Parameters | | | | | | Orbital Elements | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | h (km$^2$/s) | h$_x$ (km$^2$/s) | h$_y$ (km$^2$/s) | e$_x$ | e$_y$ | $\phi$ | a$_{sm}$ (km) | $e$ | $i$ (deg) |
| GTO-1 | Initial | 67288.41965 | 0 | -32107.258 | 0.7306 | 0 | 0 | 24364 | 0.7306 | 28.5 |
| | Target | 29640.2292 | 0 | 0 | 0 | 0 | free | 42164 | 0 | 0 |
| | Conv | – | – | – | – | – | – | $\leq \pm 0.2$ | $\leq 5e^{-5}$ | $\leq 0.08$ |
| GTO-2 | Initial | 67246.21689 | 0 | -30529.143 | 0.7310 | 0 | 0 | 24364 | 0.7310 | 27 |
| | Target | 29640.2292 | 0 | 0 | 0 | 0 | free | 42164 | 0 | 0 |
| | Conv | - | - | - | - | - | - | $\leq \pm 0.08$ | $\leq 8.7e^{-5}$ | $\leq 0.08$ |
| GTO-3 | Initial | 68196.3519 | 0 | -8311.044 | 0.725 | 0 | 0 | 24505 | 0.725 | 28.5 |
| | Target | 29640.2292 | 0 | 0 | 0 | 0 | free | 42164 | 0 | 0 |
| | Conv | - | - | - | - | - | - | $\leq \pm 12$ | $\leq 9e^{-4}$ | $\leq 0.08$ |
| Super- GTO | Initial | 70532.88179 | 0 | -26991.765 | 0.8705 | 0 | 0 | 51525 | 0.8705 | 22.5 |
| | Target | 29640.2292 | 0 | 0 | 0 | 0 | free | 42164 | 0 | 0 |
| | Conv | - | - | - | - | - | - | $\leq \pm 0.5$ | $\leq 6.9e^{-5}$ | $\leq 0.08$ |

below:

$$\phi(s_t) = -w_1(|s_t - s_{tar}|) + \sum w_2 \mathbf{e}^{-(w_3|s_t - s_{tar}|)} \tag{23}$$

$$r' = \phi(s') - \phi(s) - \tau - 5(\xi) + 100(d) \tag{24}$$

here, $w_1$, $w_2$, and $w_3$ are the weights, and $\tau$ is a constant value that accounts for time penalty, which punishes the agent as the episode duration becomes longer. Its value is set to 0.005 in this study. $\xi$ is the boundary flag, and it indicates if the agent crosses the boundary conditions, which are defined in terms of the orbital elements (a$_{sm}$,$e$,$i$) and restrict their values within a defined range to encourage the agent to explore in the right direction. The $\xi$ is set to one when the agent crosses these boundary conditions, indicating an unsuccessful episode. The done flag **d** is set to zero and it becomes one only when the agent reaches the target destination, and zero otherwise. The negative reward function in Eq. (23) reinforces the desired behavior of the agent. The parameters chosen for calculating the reward values in Eq. (23), play a crucial role in the early convergence of the algorithm. Previous studies [39], [44] utilized state parameters, as in Eq. (18), in their reward functions. In this study, the orbital elements, defined in equations (20-22), are used as the reward parameters. Eq. (24) finds the gradient of state parameters by essentially calculating the difference of state functions, which guides the agent towards the desired destination.

The transition function $\boldsymbol{P}$(s$'$|s,a), uses the same dynamic model described in section(III-A). With the current state and action parameters in equations (18-19), transition function $\boldsymbol{P}$(s$'$|s,a) uses the equations (13-19) to calculate the next state s$'$.

The SAC algorithm's time steps are determined based on the spacecraft's true anomaly-like angle, $\phi$, which ranges from 0 to $2\pi$ and starts at an integral multiple of $2\pi$ at each revolution. The agent takes an action at every **n** degrees of revolution, with **n** being the segment size. One time step is equivalent to 360/**n** degrees. By reducing the segment size, the agent takes more frequent actions during each revolution, potentially leading to more optimal results but a higher number of timesteps and longer episodes. To balance optimality with episode length, we use a variable

**TABLE 2.** Convergence of DRL networks for GTO-GEO transfer.

| Scenarios | Networks | Orbital Elements | | |
|---|---|---|---|---|
| | | a$_{sm}$ (km) | $e$ | $i$ (deg) |
| GTO-1 | Initial | 24364 | 0.7306 | 28.5 |
| | 1st DRL | $\leq \pm 55$ | $\leq 0.01$ | $\leq 0.1$ |
| | 2nd DRL | $\leq \pm 0.2$ | $\leq 5e^{-5}$ | $\leq 0.08$ |
| GTO-2 | Initial | 24364 | 0.7310 | 27 |
| | 1st DRL | $\leq \pm 55$ | $\leq 0.1$ | $\leq 0.1$ |
| | 2nd DRL | $\leq \pm 0.08$ | $\leq 8.7e^{-5}$ | $\leq 0.08$ |
| GTO-3 | Initial | 24505 | 0.725 | 28.5 |
| | 1st DRL | $\leq \pm 55$ | $\leq 0.01$ | $\leq 0.1$ |
| | 2nd DRL | $\leq \pm 12$ | $\leq 9e^{-4}$ | $\leq 0.08$ |

segment-size strategy that decreases the segment size as the agent advances. Initially, the segment size is set to 10 degrees, and as the agent approaches the target position, we increase the level of optimality by reducing the segment size to 1 degree. Finally, when the agent is very close to the target position, we set the segment size to its final value of 0.1 degrees, providing the agent with a higher resolution of action selection as it approaches the final state.

To provide a comprehensive view of our methodology and its implementation, we have detailed our approach in three different algorithms. Algorithm 1 outlines the actor-critic networks and training process used in the SAC algorithm. To enhance the optimization process, Algorithm 2 describes the variable segment size strategy we utilized. Finally, Algorithm 3, explains the steps taken in the environment to calculate the next state and reward. Together, these algorithms provide a detailed description of our methodology and its implementation.

## IV. EXPERIMENTS

Our proposed cascaded DRL methodology addresses two different orbit-raising problems: the transfer from GTO to GEO and the transfer from a Super-GTO to GEO. For the GTO-GEO transfer, we have compared our results using different spacecraft parameters, which are detailed in Table 1 along with the initial and target state and orbital parameters. To train the RL agent for transfers, we employ a reward function (Eq. 24) and define the corresponding weights for the first DRL networks in Table 4. The weights are denoted by W= [$w_1$; $w_2$; $w_3$], where W[:,1] pertains to the weightage of $a_{sm}$, W[:,2] corresponds to the weightage of $e$, and W[:,3] corresponds to the weightage of $i$ in Eq. 24. This is reflected in the first, second, and third columns of Table 4, respectively. For the second DRL networks, we increased the weights for $a_{sm}$ (W[:,1]) and $e$ (W[:,2]) by a factor of three to achieve a more precise target position.

We have conducted experiments on an Intel(R) Xeon(R) CPU E5-1620 v4 operating at a frequency of 3.5GHz with 8 cores, along with the NVIDIA GeForce GTX 1080 graphics processing unit (GPU), and 32GB of random access memory (RAM) to meet the computational requirements. We have implemented our DRL algorithms in Python 3.7 using the PyTorch framework, which is based on the OpenAI stable baselines. Our hardware/software combination has facilitated efficient and effective model development and training.

We have performed the training of DRL algorithms in multiple iterations/episodes to optimize the algorithms' performance and ensure reliable results. We have conducted our DRL training with the objective to achieve stable reward values for successful episodes in orbit-raising scenarios. In our experiments, training has been conducted until stability is attained, which typically required a maximum of 900 episodes. To identify the optimal model weights, we have employed a selection criterion based on the minimum episodic convergence time, which ensures superior performance. Subsequently, these carefully chosen weights are utilized during the inference phase to generate optimized transfer time trajectories.

To guarantee the consistency and reproducibility of our findings, we have conducted multiple trials throughout the training phase. We have verified that the reported results remain consistent across all episodes during the inference stage. This rigorous approach strengthens the validity of our research outcomes.

## A. RESULTS

In this section, we present the main findings of our study, which we have categorized into two subcategories based on the type of transfer: from GTO to GEO transfer and from Super-GTO to GEO transfer. The corresponding findings for the two types of transfer scenarios are detailed below.

### 1) TRANSFER TYPE 1 (GTO TO GEO)

To investigate the effect of spacecraft parameters and initial orbit parameters on the resulting transfer, we consider a GTO to GEO transfer and analyze three different cases of initial GTO. Each scenario is designed to test different spacecraft parameters, while we have used constant thrust values for the spacecraft with the aim of ensuring rigorous comparison with other research in the field. During the orbit transfer, the constant thrust is used for all scenarios, but we have implemented Earth eclipses for the first two cases (GTO-1 and GTO-2), which reduce thrust to zero when the spacecraft passes through the eclipse. In contrast, the third scenario (GTO-3) utilizes constant thrust values throughout, consistent with the prior work by Kwon et al. [39]. The resulting performance of each scenario is presented in Table 3, which shows the orbit-raising transfer times and compares them to reference times. The proposed method involves the training of two cascaded DRL agents to achieve maximum episodic return (score) and minimum transfer time during complete transfer training. The total time in the table reflects the accumulated time obtained from both the first and the second DRL networks, and Table 2 provides convergence tolerance values for the DRL networks in each of the three cases. The primary objective in determining these convergence values for the two DRL networks is to ensure that the first network enables the spacecraft to reach the vicinity of the target position, while the second network achieves the target value with a higher resolution and precision. This approach resulted in substantially narrower tolerance ranges, allowing the spacecraft to reach the target position with greater accuracy. Specifically, the tolerances were improved by at least an order of magnitude, two orders of magnitude, and three orders of magnitude for inclination, semi-major axis, and eccentricity, respectively, thus, better equipping the spacecraft to accomplish the mission objectives. However, for the GTO-3 transfer case, the final tolerances (listed in Table 2) were deliberately kept higher to maintain consistency with [39] and ensure a fair comparison between the results.

The results of the GTO-2 training are presented in Figure 5, demonstrating a decrease in training score variation as both the cascaded DRL networks converge to a stable score value. The transfer time for the converged episodes is also shown to decrease, with both training networks converging in under 800 episodes. Similar trends for score and time convergence are observed for other scenarios and are omitted here for brevity.

Figure 6 provides valuable insights into the behavior of the DRL agents, showcasing the variations in predicted action values (thrust angles $\alpha$ and $\beta$) throughout one complete transfer episode. The initial stages of the first DRL network exhibit high variation in thrust angle values due to the large segment size 'n', which causes the spacecraft to take fewer actions per revolution. As the spacecraft approaches the target, the segment size decreases, leading to more actions per revolution and thus reducing the variation in action values. In contrast, the segment size remains consistent in the second DRL network, as the spacecraft is already closer to the target orbit. The constant thrust values for both the DRL agents are also depicted in Figure 6, with the first agent's thrust value varying between a constant thrust and zero due to the earth eclipse model, while the second agent maintains a constant thrust value as it does not pass through the eclipse. These results underscore the importance of accounting for the dynamics and constraints of the system in training optimal policies for efficient spacecraft control.

**TABLE 3.** Comparison of results for GTO-GEO transfer cases. $I_{sp}$ denotes the specific impulse, and $\lambda$ and P represent the efficiency and power of the propulsion system, respectively.

| Scenarios | Spacecraft Parameters | Thrust N | Transfer Time (days) 1st DRL | Transfer Time (days) 2nd DRL | Transfer Time (days) Total Time | Transfer Time (days) Automated Approaches | Transfer Time (days) Traditional Approaches |
|---|---|---|---|---|---|---|---|
| GTO-1 | $I_{sp} = 1800$s <br> $\lambda = 55\%$ <br> $P = 5$kW <br> $m = 1200$kg | F= 0.31 <br> No thrust during shadow | 116.75 | 0.25 | **117** | 139.76 [16] | 121.22 [1] <br> 112.11-118.36 [2] <br> 118.29 [3], 118.36 [4] <br> 119.20 [5], 119.35-120.19 [6] |
| GTO-2 | $I_{sp} = 3300$s <br> $\lambda = 65\%$ <br> $P = 5$kW <br> $m = 450$kg | $F = 0.2007$ <br> No thrust during shadow | 70.08 | 0.31 | **70.38** | 83.43 [16] | 65.9 [1] |
| GTO-3 | $I_{sp} = 3000$s <br> $m = 2000$kg | $F = 0.35$ <br> Constant thrust all the time | 145.19 | 1.24 | **146.43** | 150.4 [39] | — |

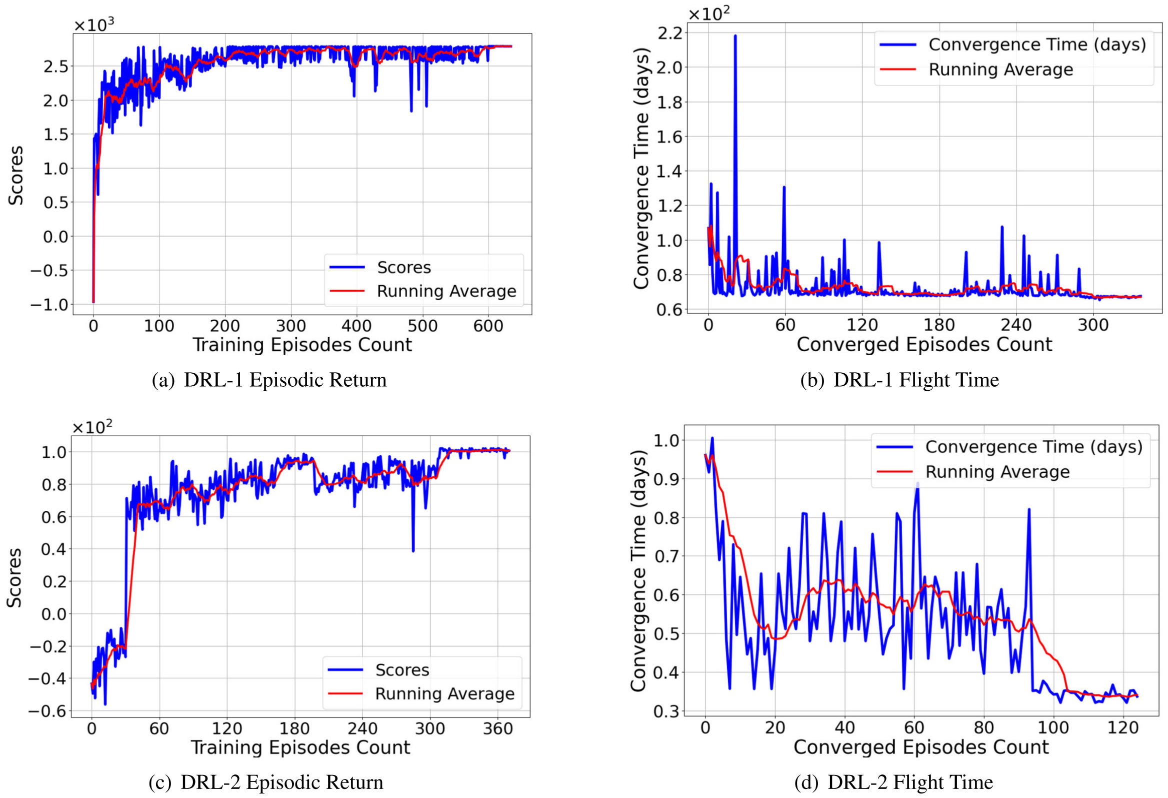


**FIGURE 5.** Episodic return and convergence times over training steps. DRL-1 and DRL-2 represent the two cascaded DRL agents. Episodic return is for all training episodes, whereas the flight time is for converged episodes during the training.

**TABLE 4.** Weights (W= $[w_1; w_2; w_3]$) used in calculating the reward function for the first DRL Network.

| | GTO to GEO | | | Super-GTO to GEO | | |
|---|---|---|---|---|---|---|
| | $W[:,1]$ | $\mathbf{W}[:,2]$ | $\mathbf{W}[:,3]$ | $W[:,1]$ | $W[:,2]$ | $\mathbf{W}[:,3]$ |
| $\mathbf{w}_1$ | $1e^3$ | $2e^3$ | $3e^2$ | $1e^3$ | $4e^3$ | $3e^2$ |
| $\mathbf{w}_2$ | $1e^{-2}$ | $1.9e^{-7}$ | $3e^{-5}$ | $1e^{-2}$ | $1.9e^{-9}$ | $3e^{-5}$ |
| $\mathbf{w}_3$ | $5e^2$ | $7e^2$ | $3e^2$ | $5e^2$ | $2e^3$ | $3e^2$ |

The graphical representation of the spacecraft's trajectory in three-dimensional space from the initial elliptical GTO to the final circular GEO is presented in Figure 7. The figure also shows the earth's shadow, enabling a detailed depiction of the transfer's dynamics. The transfer begins with high inclination and eccentricity values, gradually reducing to zero inclination and eccentricity values as the spacecraft reaches the target orbit. The above figures (Figure 5-7) illustrate the orbital transfer for the GTO-3 to GEO transfer (please refer to Table 1). The other two GTO transfers exhibit similar trajectory profiles and are thus not shown here for conciseness.

**Convergence Parameters Evaluation:** In this study, we examine the evolution of various orbital elements during a complete transfer from the first and second geosynchronous transfer orbits (GTO-1 and GTO-2 in Table 1) to the GEO, and from the third geosynchronous transfer orbit (GTO-3 in Table 1) to the GEO. The parameters of interest include

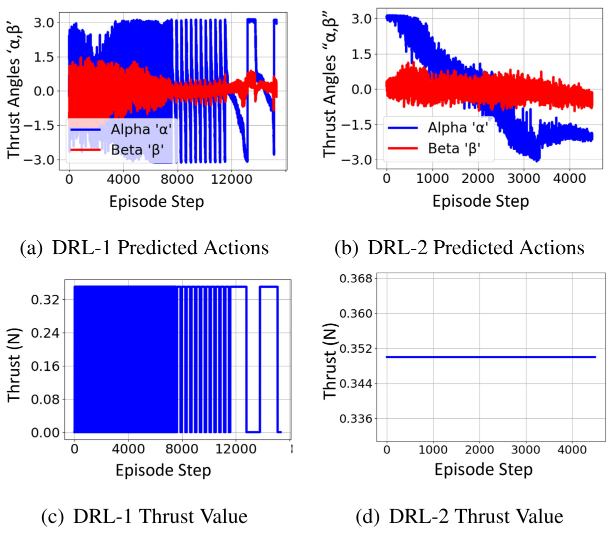


**FIGURE 6.** Predicted actions (thrust angles (rad)) and thrust variations due to earth eclipses for both deep reinforcement learning agents (DRL-1 and DRL-2).

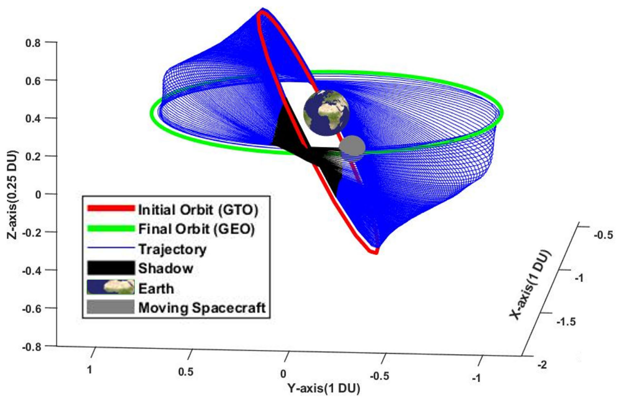


**FIGURE 7.** Three dimensional orbit transfer trajectory for GTO-GEO, where one distance unit (DU) is equivalent to the radius of GEO.

the semi-major axis ($a_{sm}$), eccentricity ($e$), and inclination ($i$). To ensure successful completion of the transfer, all three of these parameters must converge to their target values with respective tolerance values simultaneously (please refer to Table 1 for target values of state parameters and their tolerance values).

Our study found that both GTO-1 and GTO-2 transfers successfully reached their tolerance values. However, GTO-1 required 178.07 revolutions to complete the transfer, whereas GTO-2 required only 103.81 revolutions. This difference in revolution count can be attributed to GTO-1's higher initial mass-to-thrust ratio of 3870, which is 1.726 times greater than GTO-2's value of 2242. Consequently, GTO-1 took longer to complete the transfer and required more revolutions. Additionally, the spacecraft used in GTO-1 experienced a greater mass loss of 173.84 kg as compared to 36.29 kg in GTO-2.

Results further show that in contrast to the first two cases, the transfer from GTO-3 to GEO did not result in coherent changes in eccentricity, inclination, and semi-major axis.

**TABLE 5.** Convergence of DRL networks for Super-GTO to GEO transfer.

| Scenarios | Networks | Orbital Elements | | |
|---|---|---|---|---|
| | | $a_{sm}$ (km) | $e$ | $i$ (deg) |
| Super-GTO | Initial | 51525 | 0.8705 | 22.5 |
| | 1st DRL | $\leq \pm 10$ | $\leq 0.002$ | $\leq 0.1$ |
| | 2nd DRL | $\leq \pm 0.5$ | $\leq 6.9e^{-5}$ | $\leq 0.08$ |

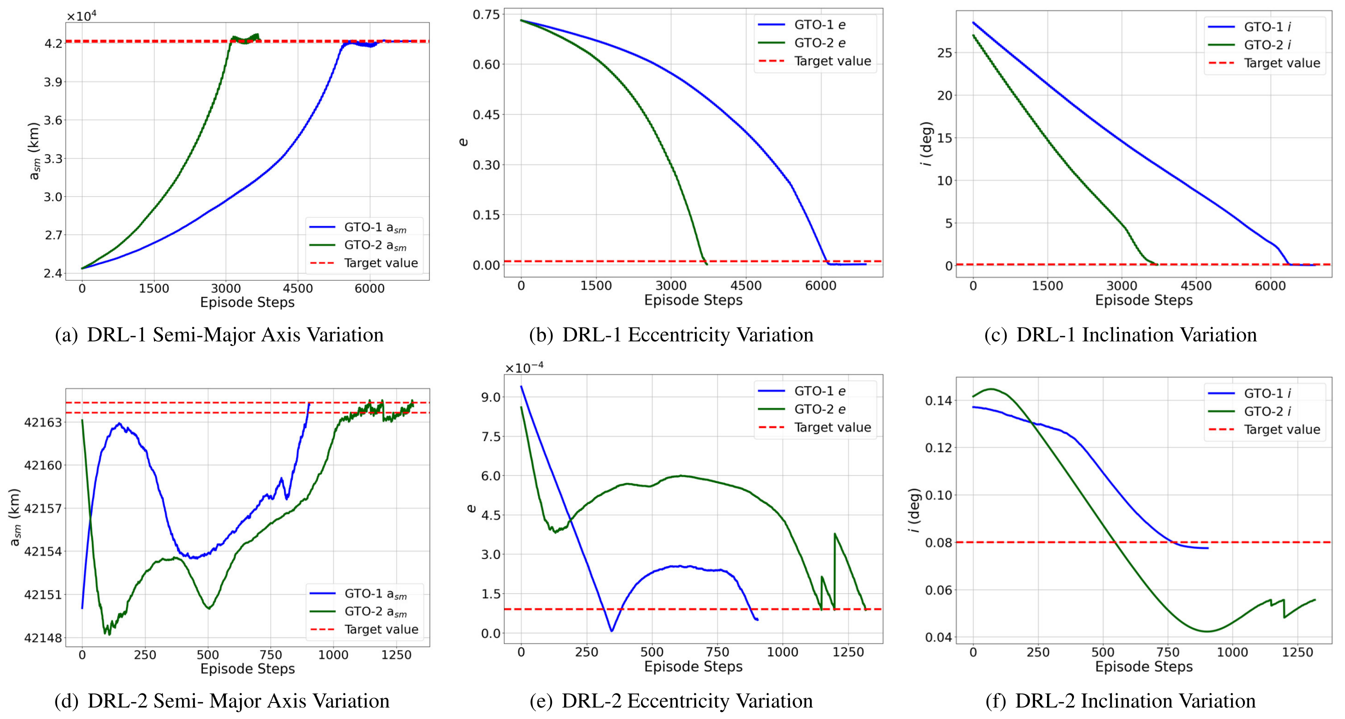
Specifically, the inclination decreased abruptly at the start and then tried to maintain its value around the target position. Furthermore, the transfer from GTO-3 to GEO took a total of 223.16 revolutions to complete, which is considerably longer than the previous two transfers. This extended duration could be attributed to GTO-3's higher initial mass-to-thrust ratio of 5714, which is 1.47 times greater than GTO-1's ratio and 2.54 times greater than GTO-2's. This observation suggests that the higher mass-to-thrust ratio in GTO-3 could be a contributing factor to the increased number of revolutions required for the transfer. The spacecraft used in GTO-3 also lost a total of 150.47 kg during the transfer with 149.2 kg and 1.27 kg lost traversing the trajectories determined by the first and the second DRL networks, respectively. Overall, the results demonstrate the effects of various parameters and initial conditions on geosynchronous orbit transfers as depicted in Figure 8 and Figure 9.

## 2) TRANSFER TYPE 2 (SUPER-GTO TO GEO)

In this particular transfer scenario, we consider the initial orbit to be a Super-GTO, which is characterized by a higher perigee altitude and greater eccentricity than the GTO orbit. To accomplish this transfer in the shortest possible time, we utilize two cascaded DRL networks using the same technique we employed in the GTO transfer cases. We employ the same spacecraft parameters and simulation conditions as those reported in literature to facilitate comparison with previous results. The convergence conditions for both DRL networks are presented in Table 5 ensuring that the spacecraft arrives in the vicinity of the target position during the first DRL network and achieves the target location precisely in the second DRL network as in the GTO transfer cases. We have endeavored to attain the same final convergence values reported in literature. The cumulative time taken by both DRL networks represents the total time required for the transfer, as shown in Table 6.

The training outcomes for both DRL agents are presented in Figure 10, which indicate that the training score increased over time and stabilized at its highest value. Furthermore, as the training progressed, the flight time of each converged episode decreased to its minimum value.

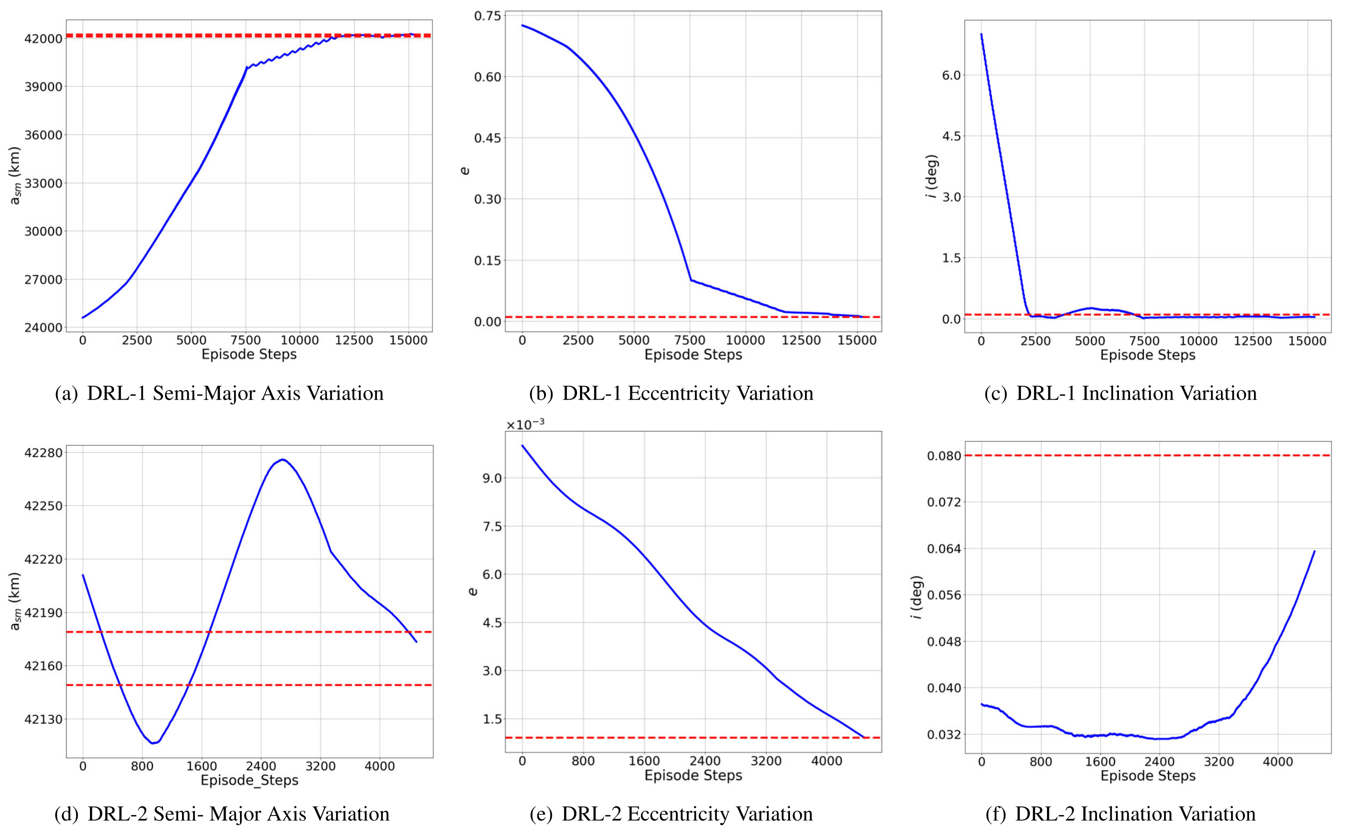
The action values predicted by the DRL agents for the Super-GTO to GEO transfer are depicted in Figure 12. The plot displays the thrust angles $\alpha$ and $\beta$ along with the variation of the thrust value that occurs due to the Earth's eclipse. At the start of the transfer for the first DRL agent, we observe high variations of the thrust angles due to greater maneuverability around the apoapsis in the relatively weaker gravitational



(a) DRL-1 Semi-Major Axis Variation (b) DRL-1 Eccentricity Variation (c) DRL-1 Inclination Variation

(d) DRL-2 Semi- Major Axis Variation (e) DRL-2 Eccentricity Variation (f) DRL-2 Inclination Variation

**FIGURE 8.** **Orbital elements variation in one complete episode for GTO-1 and GTO-2 to GEO transfer scenario.**

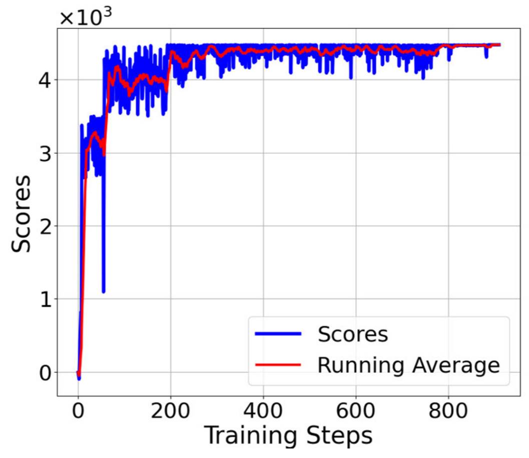


(a) DRL-1 Semi-Major Axis Variation (b) DRL-1 Eccentricity Variation (c) DRL-1 Inclination Variation

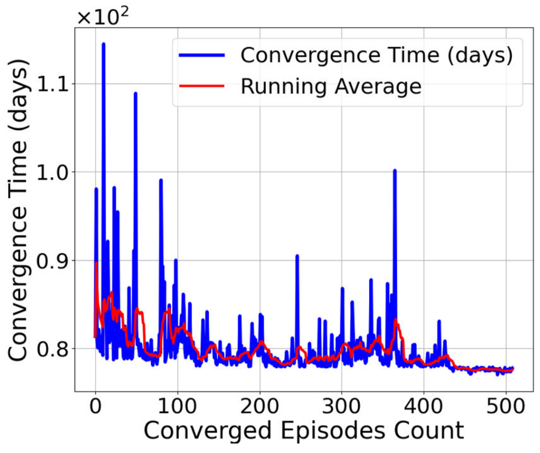
(d) DRL-2 Semi- Major Axis Variation (e) DRL-2 Eccentricity Variation (f) DRL-2 Inclination Variation

**FIGURE 9.** **Orbital elements variation in one complete episode for GTO-3 to GEO transfer scenario.**

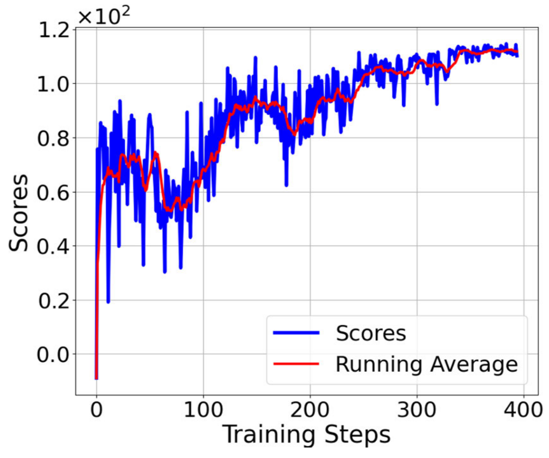
field, resulting in fewer actions taken per revolution by the spacecraft. As with the GTO transfer case, we employed a fixed segment size of 0.1 degrees for the second DRL network. Furthermore, for the second DRL agent, thrust value



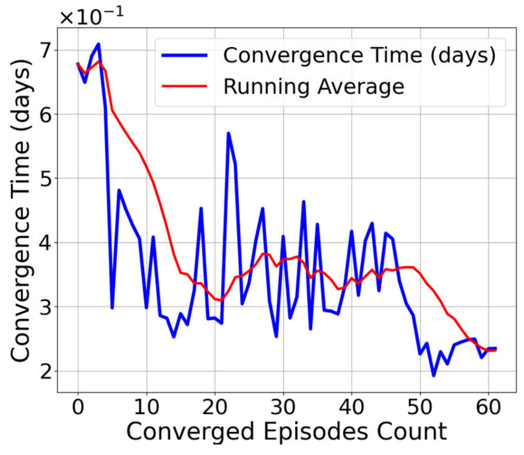
(a) DRL-1 Episodic Return



(b) DRL-1 Flight Time



(c) DRL-2 Episodic Return



(d) DRL-2 Flight Time

**FIGURE 10.** Episodic return and convergence times over training steps. DRL-1 and DRL-2 represent the two deep reinforcement learning agents. Episodic return is for all episodes, whereas flight time is for converged episodes during the training.

**TABLE 6.** Comparison of results for super-GTO to GEO transfer case. $I_{sp}$ denotes the specific impulse, and $\lambda$ and P represent the efficiency and power of the propulsion system, respectively.

| Scenarios | Spacecraft Parameters | Thrust N | Transfer Time (days) 1st DRL | Transfer Time (days) 2nd DRL | Transfer Time (days) Total Time | Transfer Time (days) Automated Approaches | Transfer Time (days) Traditional Approaches |
|---|---|---|---|---|---|---|---|
| Super-GTO | $I_{sp} = 3300$s<br>$\lambda = 65$<br>P = 10kW<br>m = 1200kg | F = 0.4015<br>No Thrust during shadow | 80.6 | 0.27 | **80.27** | 82.11 [16] | 72.60-75.42 [1] |

remained constant as the spacecraft did not pass through the eclipse during this phase of the transfer.

The three-dimensional trajectory for the Super-GTO to GEO transfer is presented in Figure 13, where the axis values are normalized to one distance unit (DU) which corresponds to the radius of GEO. The plot illustrates the trajectory from the Super-GTO orbit to the final GEO orbit. The shadow of the earth is visible where the spacecraft used zero thrust value. In addition, the figure shows an abrupt decrease in the inclination angle at the start of the transfer, which is more pronounced than in the GTO-3 transfer case and contrasts with GTO-1 and GTO-2 transfers. This result is also evident from Figure 11.

In order to demonstrate the convergence of our algorithm to the target values, we present the history of osculating orbital elements for a single episode with the desired convergence tolerances, as shown in Figure 11. It is noteworthy that all of the orbital elements, including the semi-major axis, eccentricity, and inclination, reach the target values simultaneously at the end of the transfer. The spacecraft requires a total of 74.51 revolutions to reach the target location, with 73.63 revolutions for DRL-1 and 0.88 revolutions for DRL-2, while converging at the episodic steps of 3421 and 999 for DRL-1 and DRL-2, respectively.

Although the initial mass is similar to that of the GTO-1 transfer scenario, the spacecraft takes significantly fewer

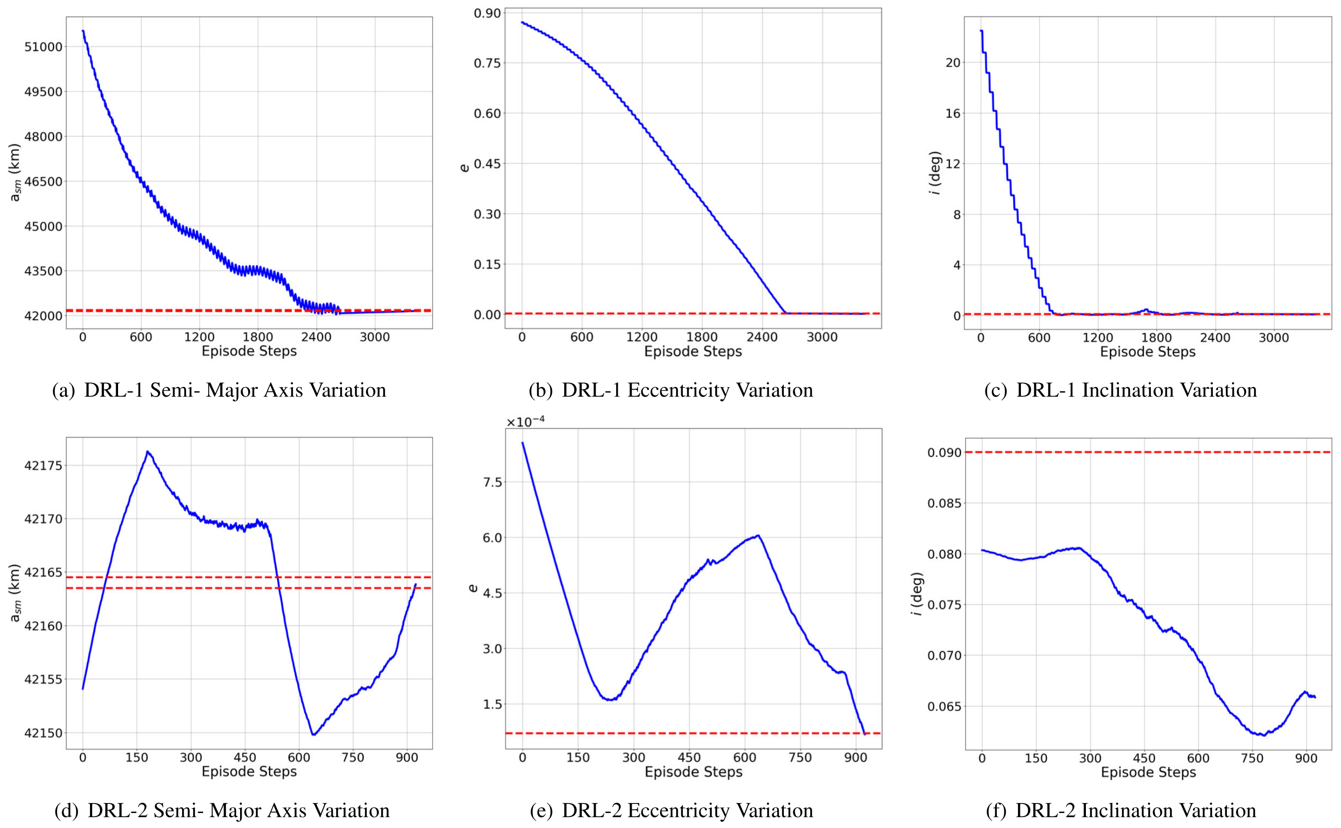


**FIGURE 11.** Orbital elements variation in one complete episode for Super-GTO to GEO transfer scenario.

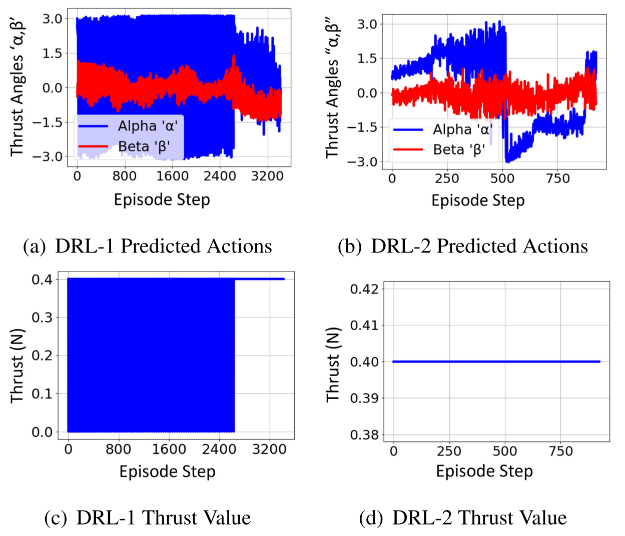


**FIGURE 12.** Predicted actions (thrust angles (rad)) and thrust variations due to earth eclipses for both deep reinforcement learning agents (DRL-1, and DRL-2).

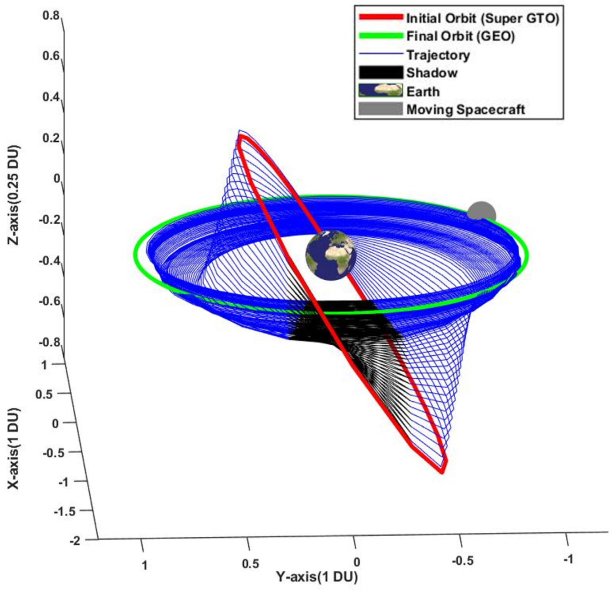


**FIGURE 13.** Three dimensional orbit- transfer trajectory for Super-GTO to GEO (1 DU = radius of GEO).

revolutions to reach the target position than in the GTO-1 transfer case. This could be attributed to the fact that the thrust value is slightly higher in this case, and the initial orbital parameters are distinct from those of the GTO, being more eccentric and having a higher semi-major axis value. As a result, this problem is more challenging to solve but takes fewer revolutions to achieve the target position. Furthermore, due to the fewer number of revolutions required to reach the target, the decrease in the spacecraft mass is also less than that of the GTO-1 transfer, totaling 84.56 kg loss (84.28 during DRL-1 and 0.28 during DRL-2).

## V. DISCUSSION AND COMPARATIVE ANALYSIS

To validate the effectiveness of our proposed method, we have conducted a comparison with other established orbit-raising techniques. We have compared the performance of our proposed method with both traditional optimization solvers [1], [2], [3], [4], [5], [6], which rely on initial guesses from mission designers for optimization, as well as automated solvers [16], [39] that operate independently of user-provided initial guess solutions, in the context of the GTO to GEO transfer case. Results indicate that our approach achieves optimal or near-optimal results in terms of transfer time for GTO to GEO and Super-GTO to GEO orbit-raising scenarios. It is important to note that the results obtained from some of the traditional optimization solvers (e.g., [2], [6]) are presented in a range due to differences in the underlying modeling for dynamic model, shadow geometry, shadow rotation, and/or perturbations. Additionally, [1] and [39] employ different parameters for the same transfer, as defined in GTO-1 and GTO-2 scenarios (please refer to Table 1 for the initial and target state parameters). The comparative assessment reveals that the proposed method outperforms the automated approaches, in terms of transfer time, in all of the three cases. Regarding the comparison with traditional numerical approaches, the proposed method delivers superior results in the GTO-1 case, where the obtained outcomes exceed the upper range value of the best-case scenario of traditional approaches [2]. For the GTO-2 scenario, our proposed method achieves comparable results to traditional approaches.

The transfer from Super-GTO to GEO proved to be a particularly arduous undertaking. Achieving convergence within the given tolerance on the terminal position of the spacecraft presented a grueling obstacle. One possible explanation is that the Super-GTO is more eccentric than the GTO. We endeavored to overcome this hurdle by doubling the eccentricity weights ($W[:, 2]$) (Table 4) of the reward function from the GTO transfer weights. This strategic adaptation enabled us to attain convergence at the target location at the end of each episode. This outcome is visually apparent in Figure 11. Our proposed approach achieves faster transfer times over automated optimization solver [16], while being sub-optimal when compared to traditional numerical solvers [1]. Note here that the modeling assumptions for the transfer are the same as in [16] to allow a direct and fair comparison; however, there are some modeling differences with [1] leading to some numerical differences in the solution. We further note that the advantage over the sequential approach [16] comes at the expense of additional training procedure, albeit offline, which the DRL based methodologies have to incorporate. We further compare our GTO results with those of [39], where they used a single reinforcement learning agent to solve the same problem but with different parameters. We match their spacecraft parameters and compare our results for the GTO-3 transfer case. From the comparative assessment, we observe that the convergence tolerance values achieved by our method are tighter meaning that the terminal state achieved is numerically closer to GEO in our case. Results show that our cascaded deep reinforcement learning networks leveraging orbital elements in gradient-aided reward function significantly surpass the approach in [39] by 2.64%.

Our experimental results demonstrate that our proposed approach achieves significantly better transfer times as compared to automated optimization solvers in all scenarios. Furthermore, our proposed approach surpasses traditional numerical optimizers in some transfer scenarios while being comparable in others. It is worth mentioning that the proposed approach offers several advantages over traditional numerical methods. First, our cascaded DRL technique is lightweight and requires minimal computational resources as compared to traditional methods. Secondly, unlike traditional approaches, our method does not require initial guess inputs from trained mission designers, making it a suitable choice for automated optimization solvers. Third, the cascaded DRL technique enables efficient exploration and exploitation of the state-action space, resulting in better convergence tolerances while achieving the target location. Moreover, our approach provides a stable and efficient learning process by utilizing orbital elements in a gradient-aided reward function that guides the reinforcement learning process towards optimal solutions. This feedback mechanism effectively leverages the strengths of DRL, which can handle complex and nonlinear dynamics more effectively than traditional numerical and machine learning techniques.

During our investigation of spacecraft orbit-raising problems, we encountered specific challenges that required careful attention. The most notable challenge was achieving convergence due to the highly nonlinear nature of the problem. Simultaneously achieving precise tolerance values for eccentricity, semi-major axis, and inclination posed a significant difficulty. To tackle this, dedicated efforts were invested in devising an effective reward function that leveraged gradient information derived from orbital state parameters. The aim was to guide the reinforcement learning agent towards achieving convergence. The previously referenced reward function, as outlined in [39], employed state parameters ($\mathrm{h}$, $\mathrm{h_x}$, $\mathrm{h_y}$, $\mathrm{e_x}$, $\mathrm{e_y}$, $\phi$). However, these parameters did not provide sufficient orbital information to facilitate convergence. Consequently, we enhanced the reward function (Eq. (24)) by incorporating orbital parameters ($\mathrm{a_{sm}}$, $e$, $i$), which proved instrumental in assisting the reinforcement learning agent to converge successfully. Furthermore, attaining narrower tolerance values comparable to those obtained by sequential and traditional trajectory optimization approaches proved to be challenging, particularly for eccentricity, which demanded precision down to five decimal points. To address this, we proposed a cascaded design for the reinforcement learning approach. This innovative design enabled us to achieve comparable tolerance values to those of sequential and traditional trajectory optimization approaches, and significantly improved convergence performance of our proposed approach while attaining optimal or near-optimal time-efficient spacecraft orbit-raising.

## VI. CONCLUSION

In this paper, we have proposed cascaded deep reinforcement learning (DRL) — a novel technique to solve complex optimization problems. We have utilized our proposed cascaded DRL to solve the orbit-raising problem from GTO to GEO and from Super-GTO to GEO. Our approach utilizes a gradient-aided reward function, developed by using orbital elements, to optimize the spacecraft's trajectory during orbit-raising. The results show that our proposed approach outperforms automated optimization solvers in achieving optimal transfer time in all transfer scenarios from GTO to GEO and super-GTO to GEO. Furthermore, our proposed approach surpasses traditional numerical optimizers in some scenarios while being comparable in others.

Our study highlights the potential for DRL to optimize trajectories in complex and nonlinear dynamical systems such as an all-electric spacecraft. The demonstrated success makes DRL a promising tool for space exploration applications. Furthermore, DRL is lightweight requiring minimal computational resources, thus making our approach particularly suitable for onboard trajectory optimization and navigation for spacecraft.

To further improve the effectiveness and versatility of DRL in space exploration, continued research in this area is necessary. While the proposed approach has demonstrated its efficacy in GTO to GEO and super-GTO to GEO transfer scenarios, the next step will be to expand this approach to other orbit-raising scenarios, which may require additional training and fine-tuning of the DRL networks. Future studies could also explore the use of more sophisticated reward functions, different optimization techniques, and different spacecraft design parameters. These approaches could potentially help to overcome the challenges posed by the highly nonlinear nature of the orbit-raising problem and improve the overall performance of DRL in space exploration.

## ACKNOWLEDGMENT

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NASA.

## REFERENCES

[1] K. F. Graham and A. V. Rao, "Minimum-time trajectory optimization of low-thrust earth-orbit transfers with eclipsing," *J. Spacecraft Rockets*, vol. 53, no. 2, pp. 289–303, Mar. 2016.

[2] C. A. Kluever and B. Conway, "Low-thrust trajectory optimization using orbital averaging and control parameterization," in *Spacecraft Trajectory Optimization*. Cambridge, U.K.: Cambridge Univ. Press, Aug. 2010, pp. 112–138. [Online]. Available: https://books.google.com/books?id=gde1Vb104coC&dq=Kluever,+Craig+A.,+and+B.+Conway.+%22Low-thrust+trajectory+optimization+using+orbital+averaging+and+control+parameterization.%22+Spacecraft+Trajectory+Optimization+(2010):+112-138.&lr=&source=gbs_navlinks_s

[3] R. Falck and J. Dankanich, "Optimization of low-thrust spiral trajectories by collocation," in *Proc. AIAA/AAS Astrodyn. Spec. Conf.*, Aug. 2012, p. 4423.

[4] B. A. Conway, *Spacecraft Trajectory Optimization*, vol. 29. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[5] J. L. Shannon, M. T. Ozimek, J. A. Atchison, and M. Christine, "Q-law aided direct trajectory optimization for the high-fidelity, many-revolution low-thrust orbit transfer problem," *Adv. Astron. Sci.*, vol. 168, pp. 781–800, Jan. 2019.

[6] H. Holt, R. Armellin, A. Scorsoglio, and R. Furfaro, "Low-thrust trajectory design using closed-loop feedback-driven control laws and state-dependent parameters," in *Proc. AIAA Scitech Forum*, Jan. 2020, p. 1694.

[7] K. F. Graham and A. V. Rao, "Minimum-time trajectory optimization of multiple revolution low-thrust earth-orbit transfers," *J. Spacecraft Rockets*, vol. 52, no. 3, pp. 711–727, 2015.

[8] J. L. Shannon, M. T. Ozimek, J. A. Atchison, and C. M. Hartzell, "Q-law aided direct trajectory optimization of many-revolution low-thrust transfers," *J. Spacecraft Rockets*, vol. 57, no. 4, pp. 672–682, Jul. 2020.

[9] X. Pan and B. Pan, "Practical homotopy methods for finding the best minimum-fuel transfer in the circular restricted three-body problem," *IEEE Access*, vol. 8, pp. 47845–47862, 2020.

[10] B. Pan, X. Pan, and P. Lu, "Finding best solution in low-thrust trajectory optimization by two-phase homotopy," *J. Spacecraft Rockets*, vol. 56, no. 1, pp. 283–291, Jan. 2019.

[11] D. Izzo, "PyGMO and PyKEP: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization)," in *Proc. 5th Int. Conf. Astrodyn. Tools Techn. (ICATT)*, 2012.

[12] L. S. Pontryagin, *Mathematical Theory of Optimal Processes*. Boca Raton, FL, USA: CRC Press, 1987.

[13] C. L. Bottasso, F. Luraghi, and G. Maisano, "Efficient rotorcraft trajectory optimization using comprehensive models by improved shooting methods," *Aerosp. Sci. Technol.*, vol. 23, no. 1, pp. 34–42, Dec. 2012.

[14] M. Ezati, P. Brown, B. Ghannadi, and J. McPhee, "Comparison of direct collocation optimal control to trajectory optimization for parameter identification of an ellipsoidal foot–ground contact model," *Multibody Syst. Dyn.*, vol. 49, no. 1, pp. 71–93, May 2020.

[15] W. Roh and Y. Kim, "Trajectory optimization for a multi-stage launch vehicle using time finite element and direct collocation methods," *Eng. Optim.*, vol. 34, no. 1, pp. 15–32, Jan. 2002.

[16] S. Sreesawet and A. Dutta, "Fast and robust computation of low-thrust orbit-raising trajectories," *J. Guid., Control, Dyn.*, vol. 41, no. 9, pp. 1888–1905, Sep. 2018.

[17] D. Izzo, M. Märtens, and B. Pan, "A survey on artificial intelligence trends in spacecraft guidance dynamics and control," *Astrodynamics*, vol. 3, no. 4, pp. 287–299, Dec. 2019.

[18] Y.-H. Zhu and Y.-Z. Luo, "Fast evaluation of low-thrust transfers via multilayer perceptions," *J. Guid., Control, Dyn.*, vol. 42, no. 12, pp. 2627–2637, Dec. 2019.

[19] H. Li, S. Chen, D. Izzo, and H. Baoyin, "Deep networks as approximators of optimal low-thrust and multi-impulse cost in multitarget missions," *Acta Astronautica*, vol. 166, pp. 469–481, Jan. 2020.

[20] H. Li, H. Baoyin, and F. Topputo, "Neural networks in time-optimal low-thrust interplanetary transfers," *IEEE Access*, vol. 7, pp. 156413–156419, 2019.

[21] H. Peng and X. Bai, "Comparative evaluation of three machine learning algorithms on improving orbit prediction accuracy," *Astrodynamics*, vol. 3, no. 4, pp. 325–343, Dec. 2019.

[22] B. Li, J. Huang, Y. Feng, F. Wang, and J. Sang, "A machine learning-based approach for improved orbit predictions of LEO space debris with sparse tracking data from a single station," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 6, pp. 4253–4268, Dec. 2020.

[23] L. Cheng, Z. Wang, and F. Jiang, "Real-time control for fuel-optimal moon landing based on an interactive deep reinforcement learning algorithm," *Astrodynamics*, vol. 3, no. 4, pp. 375–386, Dec. 2019.

[24] D. Izzo and E. Öztürk, "Real-time guidance for low-thrust transfers using deep neural networks," *J. Guid., Control, Dyn.*, vol. 44, no. 2, pp. 315–327, Feb. 2021.

[25] L. Arora and A. Dutta, "Reinforcement learning for sequential low-thrust orbit raising problem," in *Proc. AIAA Scitech Forum*, Jan. 2020, p. 2186.

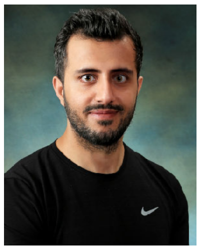
[26] H. Peng and X. Bai, "Exploring capability of support vector machine for improving satellite orbit prediction accuracy," *J. Aerosp. Inf. Syst.*, vol. 15, no. 6, pp. 366–381, Jun. 2018.

[27] H. Peng and X. Bai, "Gaussian processes for improving orbit prediction accuracy," *Acta Astronautica*, vol. 161, pp. 44–56, Aug. 2019.

[28] H. Peng and X. Bai, "Machine learning approach to improve satellite orbit prediction accuracy using publicly available data," *J. Astron. Sci.*, vol. 67, no. 2, pp. 762–793, Jun. 2020.
[29] H. Peng and X. Bai, "Fusion of a machine learning approach and classical orbit predictions," *Acta Astronautica*, vol. 184, pp. 222–240, Jul. 2021.
[30] A. H. Mughal, P. Chadalavada, A. Munir, A. Dutta, and M. A. Qureshi, "Design of deep neural networks for transfer time prediction of spacecraft electric orbit-raising," *Intell. Syst. With Appl.*, vol. 15, pp. 1–16, Jun. 2022.
[31] F. Caldas and C. Soares, "Machine learning in orbit estimation: A survey," 2022, *arXiv:2207.08993*.
[32] D. Miller and R. Linares, "Low-thrust optimal control via reinforcement learning," in *Proc. 29th AAS/AIAA Space Flight Mech. Meeting*. Ka'anapali, HI, USA: American Astronautical Society, 2019, pp. 1–18.
[33] C. J. Sullivan and N. Bosanac, "Using reinforcement learning to design a low-thrust approach into a periodic orbit in a multi-body system," in *Proc. AIAA Scitech Forum*, Jan. 2020, p. 1914.
[34] A. Scorsoglio, R. Furfaro, R. Linares, and M. Massari, "Actor-critic reinforcement learning approach to relative motion guidance in near-rectilinear orbit," *Adv. Astron. Sci.*, vol. 168, pp. 1737–1756, Feb. 2019.
[35] N. B. LaFarge, D. Miller, K. C. Howell, and R. Linares, "Guidance for closed-loop transfers using reinforcement learning with application to libration point orbits," in *Proc. AIAA Scitech Forum*, Jan. 2020, p. 0458.
[36] J. Broida and R. Linares, "Spacecraft rendezvous guidance in cluttered environments via reinforcement learning," in *Proc. 29th AAS/AIAA Space Flight Mech. Meeting*. Ka'anapali, HI, USA: American Astronautical Society, 2019, pp. 1–15.
[37] K. Hovell and S. Ulrich, "On deep reinforcement learning for spacecraft guidance," in *Proc. AIAA Scitech Forum*, Jan. 2020, p. 1600.
[38] B. Gaudet, R. Linares, and R. Furfaro, "Deep reinforcement learning for six degree-of-freedom planetary landing," *Adv. Space Res.*, vol. 65, no. 7, pp. 1723–1741, Apr. 2020.
[39] H. Kwon, S. Oghim, and H. Bang, "Autonomous guidance for multi-revolution lowthrust orbit transfer via reinforcement learning," in *Proc. AAS*, vol. 315, 2021, p. 315.
[40] A. Munir and A. Gordon-Ross, "An MDP-based dynamic optimization methodology for wireless sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 4, pp. 616–625, Apr. 2012.
[41] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[42] P. Chadalavada, T. Farabi, and A. Dutta, "Sequential low-thrust orbit-raising of all-electric satellites," *Aerospace*, vol. 7, no. 6, p. 74, Jun. 2020. [Online]. Available: https://www.mdpi.com/2226-4310/7/6/74
[43] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.
[44] D. S. Kolosa, *A Reinforcement Learning Approach to Spacecraft Trajectory Optimization*. Kalamazoo, MI, USA: Western Michigan Univ., 2019.

**SYED MUHAMMAD TALHA ZAIDI** received the Bachelor of Science degree from the University of Engineering and Technology Lahore, Lahore, in 2014, and the master's degree from Istanbul Medipol University, Turkey, in 2020. He is currently pursuing the Ph.D. degree with the Department of Computer Science, Kansas State University. He is also a Research Assistant with the ISCAAS Laboratory. He has developed a keen interest in the field of machine learning, artificial intelligence, deep reinforcement learning, and their applications. He is committed to conduct research in these areas with the aim of advancing the current knowledge and understanding of these fields.

**PARDHA SAI CHADALAVADA** (Student Member, IEEE) received the bachelor's degree in aerospace engineering from SRM University, Chennai, the master's degree in aerospace engineering from San Jose State University, San Jose, and the Ph.D. degree in aerospace engineering from Wichita State University, with a research focus on flight dynamics and controls. His research interests include astrodynamics, remote sensing, simulation, modeling, optimization, and controls.



**HAYAT ULLAH** received the bachelor's degree in computer science from Islamia College Peshawar, Peshawar, Pakistan, in 2018, and the master's degree in computer science from Sejong University, Seoul, Republic of Korea, in 2021. He is currently pursing the Ph.D. degree in computer science with Kansas State University, Mahattan, KS, USA. He is also a Research Assistant with the Intelligent Systems, Computer Architecture, Analytics, and Security (ISCAAS) Laboratory, Kansas State University, exclusively working on multi-model human actions modeling and activity recognition. He has published several articles in well-reputed journals, including IEEE Internet of Things Journal and IEEE Transactions on Image Processing. His research interests include image processing, object detection, image segmentation, video analytics, deep learning applications in surveillance, and image enhancement.

**ARSLAN MUNIR** (Senior Member, IEEE) received the M.A.Sc. degree in ECE from The University of British Columbia (UBC), Vancouver, Canada, in 2007, and the Ph.D. degree in ECE from the University of Florida (UF), Gainesville, FL, USA, in 2012. He is currently an Associate Professor with the Department of Computer Science, Kansas State University. He was a Postdoctoral Research Associate with the Electrical and Computer Engineering (ECE) Department, Rice University, Houston, TX, USA, from May 2012 to June 2014. From 2007 to 2008, he was a Software Development Engineer with Mentor Graphics Corporation, Embedded Systems Division. His current research interests include embedded and cyber-physical systems, secure and trustworthy systems, parallel computing, artificial intelligence, and computer vision. He received many academic awards, including the doctoral fellowship from Natural Sciences and Engineering Research Council (NSERC) of Canada. He earned gold medals for best performance in electrical engineering, gold medals and academic roll of honor for securing rank one in pre-engineering provincial examinations (out of approximately 300,000 candidates).

**ATRI DUTTA** (Member, IEEE) received the Bachelor of Technology degree in aerospace engineering from the Indian Institute of Technology Kharagpur, Kharagpur, and the M.S. and Ph.D. degrees in aerospace engineering from the Georgia Institute of Technology. He is currently an Associate Professor with the Department of Aerospace Engineering, Wichita State University. His research interests include space dynamics, control theory, and space mission analysis.