Bayesian quadrature policy optimization for spacecraft proximity maneuvers and docking

Du, Desong; Liu, Yanfang; Zhang, Ouyang; Qi, Naiming; Yao, Weiran; Pan, Wei

DOI
10.1016/j.ast.2024.109474

Publication date
2024

Document Version
Final published version

Published in
Aerospace Science and Technology

Aerospace Science and Technology 154 (2024) 109474

Contents lists available at ScienceDirect

Aerospace Science and Technology

journal homepage: www.elsevier.com/locate/aescte

Bayesian quadrature policy optimization for spacecraft proximity maneuvers and docking

Desong Du [a,b], Yanfang Liu [a], Ouyang Zhang [a], Naiming Qi [a], Weiran Yao [a], Wei Pan [c,b,*]

[a] Harbin Institute of Technology, Harbin, 150001, China
[b] Delft University of Technology, Delft, 2629 HS, Netherlands
[c] The University of Manchester, Manchester, M13 9PL, United Kingdom

ARTICLE INFO

Communicated by Chaoyong Li

Keywords:
Proximity maneuvers and docking
Reinforcement learning
Bayesian quadrature policy optimization

ABSTRACT

Advancing autonomous spacecraft proximity maneuvers and docking (PMD) is crucial for enhancing the efficiency and safety of inter-satellite services. One primary challenge in PMD is the accurate *a priori* definition of the system model, often complicated by inherent uncertainties in the system modeling and observational data. To address this challenge, we propose a novel Lyapunov Bayesian actor-critic reinforcement learning algorithm that guarantees the stability of the control policy under uncertainty. The PMD task is formulated as a Markov decision process that involves the relative dynamic model, the docking cone, and the cost function. By applying Lyapunov theory, we reformulate temporal difference learning as a constrained Gaussian process regression, enabling the state-value function to act as a Lyapunov function. Additionally, the proposed Bayesian quadrature policy optimization method analytically computes policy gradients, effectively addressing stability constraints while accommodating informational uncertainties in the PMD task. Experimental validation on a spacecraft air-bearing testbed demonstrates the significant and promising performance of the proposed algorithm.

1. Introduction

Autonomous spacecraft proximity maneuver and docking (PMD) is one of the critical technologies for many future space missions, such as on-orbit servicing, assembly, and debris capture [1,2]. These complex maneuvers require the Chaser spacecraft to navigate autonomously and safely close to the Target spacecraft, which is fundamentally a constrained optimal control problem. However, the computational demand for solving this problem often exceeds the capability of the spacecraft system, particularly when rapid re-planning is necessitated by the dynamic space environment. This dilemma creates a significant trade-off between achieving algorithmic optimality and managing computational complexity [3]. Various theoretical approaches have been explored to address this challenge [4], including artificial potential field (APF) [3,5], H_∞ or finite-time controller [6,7], model predictive control (MPC) [8,9] and so on.

APF-based methods stand out because of their computational efficiency, enabling real-time execution onboard spacecraft. Lopez et al. [10] developed an APF-based guidance method that ensures analytical convergence of the Chaser spacecraft to the Target without breaching path constraints. Zappulla et al. [3] improved this methodology by introducing an adaptive APF approach, optimizing fuel efficiency by modifying the attractive potential shaping matrix to maintain a predefined trajectory. The effectiveness of the APF-based method was further validated through its application in the Synchronized Position Hold Engage and Reorient Experimental Satellites project on the International Space Station [11]. Despite these advancements, APF-based methods do not guarantee overall performance since no cost function is explicitly minimized, and suffer the existence of local minima, which may trap the spacecraft and prevent it from completing the maneuver. Motivated by this fact, this paper aims to design a controller that not only ensures the overall performance in PMD tasks but also aligns with the computational limitations of spacecraft.

Compared with traditional technology, the machine learning (ML) technology represented by reinforcement learning (RL) possesses huge advantages in accuracy, efficiency, and real-time performance [12,13]. RL offers a promising solution to the constrained optimal control problems inherent in PMD tasks, with the added advantage of deploying RL-trained controllers directly onto spacecraft systems. There have been notable advances in the space application of various RL algorithms [14–16], such as proximal policy optimization (PPO) [17], deep deterministic policy gradient (DDPG) [18], and so on. Zhang et al. [19]

* Corresponding author.
E-mail address: wei.pan@manchester.ac.uk (W. Pan).

https://doi.org/10.1016/j.ast.2024.109474
Received 12 May 2024; Received in revised form 22 July 2024; Accepted 7 August 2024
Available online 19 August 2024

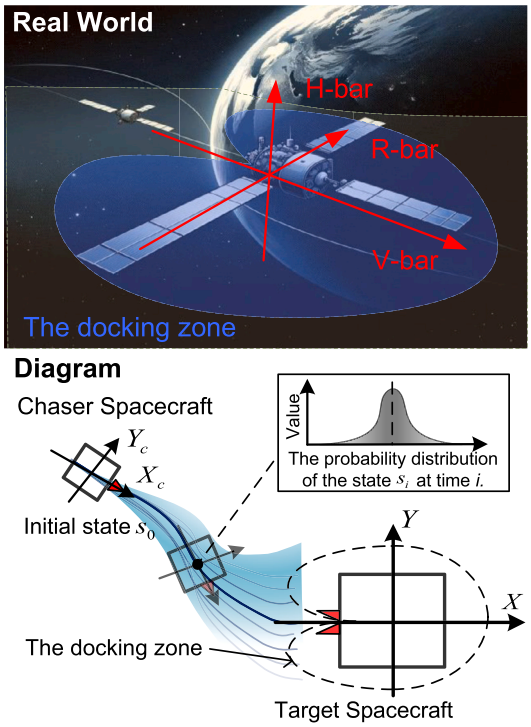

Fig. 1. Proximity maneuver and docking diagram.

employed a DDPG-based tracking controller for the space robot to capture tumbling objects. Broida and Linarees [20] explored the use of PPO in developing a closed-loop controller for satellite rendezvous missions. Qu et al. [21] further innovated with an exploration-adaptive DDPG approach for PMD tasks, incorporating meta-learning techniques to quickly adapt to analogous scenarios. Yang et al. [22] employed dynamic programming to address spacecraft rendezvous guidance and generate the decision table. However, these algorithms are predominantly based on extensive simulation data for performance evaluation, which lacks the theoretical stability guarantee. This reliance on simulated environments leads to the "sim-to-real" problem, highlighting a potential performance discrepancy when these controllers are applied in real-world scenarios. This problem highlights the urgent need for RL with stability guarantees to ensure the reliability and effectiveness of critical space operations.

The PMD task often involves stability requirements where the controller must be guaranteed to the Chaser spacecraft towards the Target spacecraft [12]. Lyapunov's theory [23] provides a principled method for ensuring system stability by constructing a scalar "energy" function, known as a Lyapunov function, that decreases over time. Recently, there is a long list of literature on learning/data-driven control that guarantees stability by utilizing data to derive a Lyapunov function, regardless of whether RL is used or not [24–27]. Thereafter, the policy/controller $\boldsymbol{\pi}$ can be sought after using the conceptual optimization procedure as follows (or its relaxations [24]):

$$\text{optimize} \quad \text{policy/controller } \boldsymbol{\pi}, \tag{1}$$

$$\text{subject to} \quad \Delta V_{\pi}(\boldsymbol{s}) \leq 0, \quad \forall \boldsymbol{s} \in \text{State space}, \tag{2}$$

where $\Delta V_{\pi}(\boldsymbol{s})$ denotes the time derivative of Lyapunov function $V(\boldsymbol{s})$ under the policy $\boldsymbol{\pi}$.

To address this issue, in our previous work [28], a data-based (model-free) approach is proposed by substituting $V(\boldsymbol{s})$ in (2) with $\mathbb{E}_{\text{data}} V_{\boldsymbol{\pi}}(\boldsymbol{s})$ in the policy optimization procedure (1), in which the state-action value function is chosen as $V_{\boldsymbol{\pi}}(\boldsymbol{s})$. A primary challenge lies in the need to specify an accurate system model *a priori*, encompassing both the type of model and the distribution of parameters. However, in PMD tasks, the presence of uncertainties in the system model and observational data is unavoidable. Although Bayesian RL methods [29,30] have been developed to mitigate these uncertainties, they have not taken into account the stability of the policy/controller. Moreover, the application of popular RL methods, such as PPO [17] and SAC [31], to PMD tasks did not yield satisfactory results in our attempts.

Motivated by these facts, this work attempts to design a novel Bayesian RL algorithm that integrates Bayesian methods with Lyapunov stability theory to enhance the reliability and safety of control policies under uncertainties in the PMD task. The contributions of this paper are listed in the following aspects.

1) A Lyapunov Bayesian actor-critic (LBAC) algorithm is proposed to learn a control policy for PMD tasks. Using the principles of Lyapunov theory, temporal difference learning is reformulated as a constrained Gaussian process regression problem, enabling the state-value function to act as a Lyapunov function.

2) A novel Bayesian quadrature policy optimization procedure is proposed to analytically compute the policy gradient with the truncated Gaussian process. This method effectively addresses stability constraints while considering informational uncertainties in PMD tasks.

3) The effectiveness of the LBAC algorithm for PMD tasks has been thoroughly evaluated through simulations and on a spacecraft air-bearing testbed. These evaluations demonstrate the capabilities of the algorithm in real-world applications, highlighting its potential to improve the safety and efficiency of autonomous spacecraft operations.

The paper is organized as follows. Section 2 defines the PMD task and introduces the spacecraft dynamic model. Section 3 details the traditional Bayesian RL algorithm. The novel constrained Gaussian processes temporal difference learning method is introduced in Section 4. The efficacy of the proposed LBAC algorithm for PMD tasks is evaluated through simulations and testbed experiments in Section 5. Finally, Section 6 presents the conclusions.

2. Modeling and problem formulation

The rendezvous mission is divided into several major phases [32], the last part of which is the PMD task, as shown in Fig. 1. The Chaser spacecraft and the Target spacecraft are close to each other and in the plane of V-bar and R-bar, where V-bar is the orbit direction of the Target spacecraft and R-bar is the direction of the center of the Earth. In PMD task, the Chaser spacecraft is required to approach and dock with the Target spacecraft and remain without the docking cone corridor during the process.

2.1. System dynamics

The related motion of the Chaser spacecraft with respect to the Target spacecraft, both in near-circular orbits, is described by the Hill-Clohessy-Wiltshire equations [32]. When both spacecraft are close together, the dynamics can be further simplified to a double integrator model as set in [8,9,33]. As shown in Fig. 2, we consider the inertial frame XYZ fixed with the Target, and the Chaser-fixed frame $X_cY_cZ_c$. Considering the uncertainty of parameters, the related dynamics of the two translational degrees and one rotational degree of freedom are described by the following equations:

$$\begin{aligned}
\ddot{x} &= \frac{f_x}{m} + \frac{m\Delta f_x - f_x \Delta m}{m(m+\Delta m)} = \frac{f_x}{m} + \Delta \ddot{x} \\
\ddot{y} &= \frac{f_y}{m} + \frac{m\Delta f_y - f_y \Delta m}{m(m+\Delta m)} = \frac{f_y}{m} + \Delta \ddot{y} \\
\ddot{\theta} &= \frac{\tau}{I_z} + \frac{I_z \Delta\tau - \tau \Delta I_z}{I_z(I_z+\Delta I_z)} = \frac{\tau}{I_z} + \Delta \ddot{\theta}
\end{aligned} \tag{3}$$

where m, I_z, Δm and ΔI_z are the nominal and bias value of the Chaser spacecraft's mass and moment of inertia, f_x, f_y, τ_z, Δf_x, Δf_y and $\Delta\tau_z$ are the nominal value of the control forces in the respective inertial X_c+ and Y_c+ direction, and the control torque axis along the Z_c+ direction.

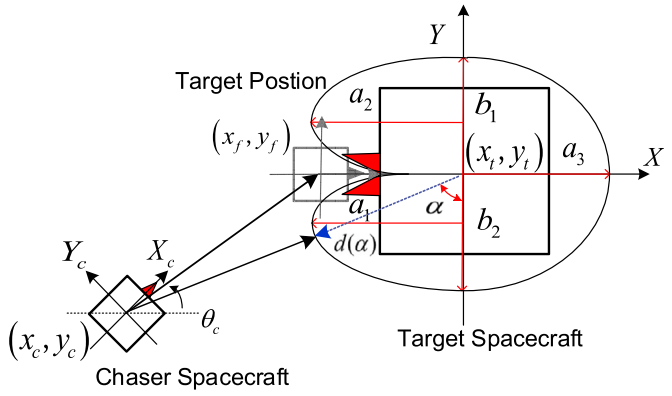

Fig. 2. Proximity maneuver and docking task setup.

2.2. Docking cone corridor

The purpose of the docking cone corridor is to establish a safe navigational area during the process. To enhance the applicability of the docking cone corridor, we employ a cardioid-like constraint, composed of four ellipses, as detailed in [33,34]. The boundaries of these ellipses are defined by the distance function $d(\alpha)$, which is oriented with respect to the coordinate system XYZ. As shown in Fig. 2, the distance function $d(\alpha)$ is defined as:

$$d(\alpha)=\begin{cases}\frac{2a_1^2 b_2\cos\alpha}{a_1^2\cos^2\alpha+b_2^2\sin^2\alpha} & \text{if } \alpha\in\left[0,\frac{\pi}{2}\right)\\ \frac{-2a_2^2 b_1\cos\alpha}{a_2^2\cos^2\alpha+b_1^2\sin^2\alpha} & \text{if } \alpha\in\left[\frac{\pi}{2},\pi\right)\\ \frac{2a_3 b_1}{\sqrt{a_3^2\cos^2\alpha+4b_1^2\sin^2\alpha}} & \text{if } \alpha\in\left[\pi,\frac{3\pi}{2}\right)\\ \frac{2a_3 b_2}{\sqrt{a_3^2\cos^2\alpha+4b_2^2\sin^2\alpha}} & \text{if } \alpha\in\left[\frac{3\pi}{2},2\pi\right)\end{cases} \tag{4}$$

where α is the angle between the $Y-$ direction and the point along the boundary, a_1, a_2, a_3, b_1 and b_2 are the lengths of the semi-major and semi-minor axes of these semi-ellipses, respectively. Note that the Target configuration is $\boldsymbol{x}=[0,0,0]^\top\in\mathbb{R}^3$; the Chaser configuration is $\boldsymbol{x}_c=\left[x_c,y_c,\theta_c\right]^\top\in\mathbb{R}^3$; the desired goal configuration within the workspace is $\boldsymbol{x}_f=\left[x_f,y_f,\theta_f\right]^\top\in\mathbb{R}^3$. Moreover, the Chaser relative to the boundary constraint of the Target, $\boldsymbol{r}_{cb}=\boldsymbol{x}_c-\boldsymbol{x}_b=\boldsymbol{r}_c-d(\alpha)\left(\boldsymbol{r}_c/\left\|\boldsymbol{r}_c\right\|_2\right)\in\mathbb{R}^3$, where $\boldsymbol{r}_c=\boldsymbol{x}_c-\boldsymbol{x}$ is the relative difference between the Chaser and the Target.

2.3. Problem statement

In RL, the system should be modeled by a Markov decision process (MDP) $\langle\mathcal{S},\mathcal{A},\boldsymbol{P},c,\rho_0,\gamma\rangle$, where $\mathcal{S}\subseteq\mathbb{R}^n$ is the state-space, $\mathcal{A}\subseteq\mathbb{R}^m$ is the action-space, $\boldsymbol{P}$ is the transition probability density function, $c:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ is the cost function,[1] ρ_0 is the distribution of starting states and $\gamma\in[0,1)$ is the discount factor.

In PMD tasks, the state at time t is defined as $\boldsymbol{s}_t=\left[\boldsymbol{x}_{fc}^\top,\dot{\boldsymbol{x}}_c^\top\right]^\top\in\mathcal{S}\subseteq\mathbb{R}^6$, where $\boldsymbol{x}_{fc}=\boldsymbol{x}_f-\boldsymbol{x}_c$ is the relative position between the Chaser and the goal and $\dot{\boldsymbol{x}}_c=\left[\dot{x}_c,\dot{y}_c,\dot{\theta}_c\right]^\top\in\mathbb{R}^3$ is the velocity of the Chaser. The Chaser then takes an action $\boldsymbol{a}_t=\left[f_x,f_y,\tau\right]^\top\in\mathcal{A}\subseteq\mathbb{R}^3$, which is the control force and torque of the Chaser, according to the stochastic policy $\boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{s}_t)$, resulting in the next state $\boldsymbol{s}_{t+1}$. The cost function is set as $c=c_a+c_b$, in which the attractive function is defined as $c_a=\boldsymbol{x}_{fc}^\top\boldsymbol{Q}_a\boldsymbol{x}_{fc}$, with $\boldsymbol{Q}_a\in\mathbb{R}^{3\times3}$ being a positive definite matrix, and the repulsive function is defined as $c_b=k_r\mathbf{x}_{fc}^\top\boldsymbol{Q}_b\mathbf{x}_{fc}/\left[2\exp\left(\mathbf{x}_{cb}^\top\boldsymbol{P}_b\mathbf{x}_{cb}-1\right)\right]$, with $k_r\in\mathbb{R}_+$, and $\boldsymbol{Q}_b$, $\boldsymbol{P}_b\in\mathbb{R}^{3\times3}$ being positive definite matrices. We consider the following control problem:

Definition 1 *(Control Problem)*. Given the MDP and the goal configuration $\boldsymbol{s}_{\text{goal}}$, find a control policy $\boldsymbol{\pi}(\boldsymbol{s})$ such that all trajectories $\boldsymbol{s}_t$ with $\boldsymbol{s}_0\sim\rho_0$ have the following properties $\lim_{t\to\infty}\mathbb{E}_{\boldsymbol{s}_t}\|\boldsymbol{s}_t-\boldsymbol{s}_{\text{goal}}\|=0$.

Note that the closed-loop transition probability is denoted as $\boldsymbol{P}_{\boldsymbol{\pi}}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t)\triangleq\int_{\mathcal{A}}\boldsymbol{\pi}(\boldsymbol{a}|\boldsymbol{s}_t)\boldsymbol{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a})\mathrm{d}\boldsymbol{a}$, where $\boldsymbol{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t)$ is the transition probability. The state distribution at time t is denoted by $\boldsymbol{P}(\boldsymbol{s}|\rho,\boldsymbol{\pi},t)$, which could be defined iteratively: $\boldsymbol{P}(\boldsymbol{s}_{t+1}|\rho,\boldsymbol{\pi},t+1)=\int_S\boldsymbol{P}_{\boldsymbol{\pi}}(\boldsymbol{s}_{t+1}|\boldsymbol{s})\boldsymbol{P}(\boldsymbol{s}|\rho,\boldsymbol{\pi},t)\mathrm{d}\boldsymbol{s},\forall t\in\mathbb{Z}_+$ and $\boldsymbol{P}(\boldsymbol{s}|\rho_0,\boldsymbol{\pi},0)=\rho_0(\boldsymbol{s})$.

3. Deep Bayesian actor-critic algorithm

We introduce the deep Bayesian actor-critic algorithm (DBAC) [29], which combines the Gaussian process temporal difference (GPTD) for the critic [35,36], with the deep Bayesian quadrature (BQ) to estimate the policy gradient for the actor [37].

3.1. Gaussian process temporal difference learning

The posterior moments of the state action value $Q(\boldsymbol{z})$, where $\boldsymbol{z}=(\boldsymbol{s},\boldsymbol{a})$, are calculated using GPTD. This approach relies on a statistical generative model that links the observed cost c to the unknown state-action value function $Q(\boldsymbol{z})$ as:

$$c\left(\boldsymbol{z}_t\right)=Q\left(\boldsymbol{z}_t\right)-\gamma Q\left(\boldsymbol{z}_{t+1}\right)+N\left(\boldsymbol{z}_t,\boldsymbol{z}_{t+1}\right) \tag{5}$$

where $N\left(\boldsymbol{z}_t,\boldsymbol{z}_{t+1}\right)$ is a zero-mean noise signal that accounts for the discrepancy between $c(\boldsymbol{z}_t)$ and $Q\left(\boldsymbol{z}_t\right)-\gamma Q\left(\boldsymbol{z}_{t+1}\right)$. Define the finite-dimensional processes $\boldsymbol{c}$, $\boldsymbol{Q}$, $\boldsymbol{N}$, and $\boldsymbol{H}$ as follows:

$$\begin{aligned}\boldsymbol{c}&=\left[c(\boldsymbol{z}_0),\ldots,c(\boldsymbol{z}_{T-1})\right]\in\mathbb{R}^T\\ \boldsymbol{Q}&=\left[Q(\boldsymbol{z}_0),\ldots,Q(\boldsymbol{z}_T)\right]\in\mathbb{R}^{T+1}\\ \boldsymbol{N}&=\left[N\left(\boldsymbol{z}_0,\boldsymbol{z}_1\right),\ldots,N\left(\boldsymbol{z}_{T-1},\boldsymbol{z}_T\right)\right]\in\mathbb{R}^T\\ \boldsymbol{H}&=\begin{bmatrix}1&-\gamma&\ldots&0&0\\0&1&\ldots&0&0\\\vdots&\vdots&\vdots&\vdots&\vdots\\0&0&\ldots&1&-\gamma\end{bmatrix}\in\mathbb{R}^{T\times(T+1)}\end{aligned} \tag{6}$$

Thus, we have a compact form for (5) as:

$$\boldsymbol{c}=\boldsymbol{H}\boldsymbol{Q}+\boldsymbol{N} \tag{7}$$

We assume the covariance σ for the state-action value function $Q(\boldsymbol{z})$, as described in [35,36]. Consequently, the noise vector $\boldsymbol{N}$ follows a normal distribution, denoted as $\boldsymbol{N}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{\Sigma}_T)$, where,

$$\boldsymbol{\Sigma}_T=\sigma^2\boldsymbol{H}\boldsymbol{H}^\top=\sigma^2\begin{bmatrix}1+\gamma^2&-\gamma&0&\ldots&0\\-\gamma&1+\gamma^2&0&\ldots&0\\\vdots&\vdots&\vdots&\vdots&\vdots\\0&0&\ldots&-\gamma&1+\gamma^2\end{bmatrix}. \tag{8}$$

When $\boldsymbol{z}_T$ is the terminal pair in the episode with $\boldsymbol{s}_T$ being the goal state, the matrix $\boldsymbol{H}$ is transformed into a square matrix $T\times T$, as defined in (6) with its last column removed. The effect on the noise covariance matrix $\boldsymbol{\Sigma}_T$ is that the bottom-right element becomes 1 instead of $1+\gamma^2$.

By imposing a Gaussian process prior to $\boldsymbol{Q}$ and assuming that $\boldsymbol{N}$ follows a normal distribution, we employ the Bayesian rule to derive the posterior moments of $\boldsymbol{Q}$:

$$\begin{aligned}\mathbb{E}(\boldsymbol{Q}(\boldsymbol{z}_i)|\boldsymbol{D})&=\boldsymbol{k}(\boldsymbol{z}_i)^\top\boldsymbol{\alpha}\\ \mathrm{Cov}\left[\boldsymbol{Q}(\boldsymbol{z}_i),\boldsymbol{Q}(\boldsymbol{z}_j)|\boldsymbol{D}\right]&=k(\boldsymbol{z}_i,\boldsymbol{z}_j)-\boldsymbol{k}(\boldsymbol{z}_i)^\top\boldsymbol{S}\boldsymbol{k}(\boldsymbol{z}_j)\end{aligned} \tag{9}$$

[1] We use $c(\boldsymbol{z}_t)$ to denote the cost instead of the reward, where c evaluates the performance of the state-action pair $\boldsymbol{z}_t=(\boldsymbol{s}_t,\boldsymbol{a}_t)$.

where $\boldsymbol{D}$ denotes the dataset including the time step T. It has the following definitions: $\boldsymbol{k}(z_i) = \left[k(z_0, z_i), \ldots, k(z_T, z_i)\right]^\top$, $\boldsymbol{K} = \left[\boldsymbol{k}(z_0), \ldots, \boldsymbol{k}(z_T)\right]$, $\boldsymbol{\alpha} = \boldsymbol{H}^\top \left(\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma}_T\right)^{-1} \boldsymbol{c}$, $\boldsymbol{S} = \boldsymbol{H}^\top \left(\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma}_T\right)^{-1} \boldsymbol{H}$.

3.2. Bayesian quadrature policy optimization

Bayesian quadrature policy optimization defined in [38] represents a methodological advancement by incorporating BQ from the probabilistic numeric. This approach reformulates the task of numerical integration, as established in Theorem 1 in [39], into a Bayesian inference problem. Specifically, it focuses on computing the gradient of the expected return as:

$$\nabla_\theta J(\theta) = \int_{\mathcal{Z}} dz \rho_{\pi_\theta}(z) \boldsymbol{u}(z) Q_{\pi_\theta}(z) = \mathbb{E}_{z\sim\rho_{\pi_\theta}}\left[\boldsymbol{u}(z) Q_{\pi_\theta}(z)\right] \tag{10}$$

where $\boldsymbol{u}(z) = \nabla_\theta \log \boldsymbol{\pi}_\theta(\boldsymbol{a}|\boldsymbol{s})$ denotes the score function, $\boldsymbol{\theta}$ are the parameters of the policy $\boldsymbol{\pi}_\theta$ and ρ_{π_θ} is the discounted state-action visitation frequency defined as: $\rho_{\pi_\theta}(z) = \sum_{t=0}^{\infty} \gamma^t \boldsymbol{\pi}_\theta(\boldsymbol{a}|\boldsymbol{s}) \boldsymbol{P}(\boldsymbol{s}|\rho, \boldsymbol{\pi}_\theta, t)$.

The first step in Bayesian quadrature policy optimization involves establishing a prior stochastic model for the integration process. This is achieved by placing a Gaussian process prior for the state-action value function Q_{π_θ}, that is, a zero mean Gaussian process $\mathbb{E}\left[Q_{\pi_\theta}(z_i)\right] = 0$, characterized by a covariance function $\mathrm{Cov}\left[Q_{\pi_\theta}(z_i), Q_{\pi_\theta}(z_j)\right] = \sigma^2$. Next, the Gaussian process prior is updated by Bayesian rule with the sampled data $\boldsymbol{D} = \{\boldsymbol{z}_t\}_{t=0}^T$, to obtain the posterior moments $\mathbb{E}\left[Q_{\pi_\theta}(z_i)|\boldsymbol{D}\right]$ and $\mathrm{Cov}\left[Q_{\pi_\theta}(z_i), Q_{\pi_\theta}(z_j)|\boldsymbol{D}\right]$.

Since the transformation from $Q_{\pi_\theta}(z)$ to $\nabla_\theta J(\theta)$ happens through a linear integral operator in (10), $\nabla_\theta J(\theta)$ also follows a Gaussian process:

$$\begin{aligned}
\boldsymbol{L}_\theta =& \mathbb{E}\left[\nabla_\theta J(\theta)|\boldsymbol{D}\right] = \int_{\mathcal{Z}} dz \rho_{\pi_\theta}(z) \boldsymbol{u}(z) \mathbb{E}\left[Q_{\pi_\theta}(\boldsymbol{u})|\boldsymbol{D}\right] \\
\boldsymbol{C}_\theta =& \mathrm{Cov}[\nabla_\theta J(\theta)|\boldsymbol{D}] = \int_{\mathcal{Z}^2} dz_i dz_j \rho_{\pi_\theta}(z_i) \rho_{\pi_\theta}(z_j) \\
& \boldsymbol{u}(z_i) \mathrm{Cov}\left[Q_{\pi_\theta}(z_i), Q_{\pi_\theta}(z_j)|\boldsymbol{D}\right] \boldsymbol{u}(z_j)^\top
\end{aligned} \tag{11}$$

where the mean vector $\boldsymbol{L}_\theta$ is the policy gradient estimate and the covariance $\boldsymbol{C}_\theta$ is its uncertainty estimation. While the integrals in (11) are still intractable for an arbitrary Gaussian process kernel k, they have a closed-form solution when k is the additive composition of a state kernel k_s (arbitrary) and the Fisher kernel k_f (indispensable) [29]:

$$\begin{aligned}
& k(z_i, z_j) = c_1 k_s(\boldsymbol{s}_i, \boldsymbol{s}_j) + c_2 k_f(z_i, z_j), \\
& \textit{with } k_f(z_i, z_j) = \boldsymbol{u}(z_i)^\top \mathbf{G}^{-1} \boldsymbol{u}(z_j),
\end{aligned} \tag{12}$$

where c_1, c_2 are hyperparameters and $\boldsymbol{G}$ is the fisher information matrix of $\boldsymbol{\pi}_\theta$. Using the matrices,

$$\begin{aligned}
& \boldsymbol{K}_f = \boldsymbol{U}^\top \boldsymbol{G}^{-1} \boldsymbol{U}, \quad \boldsymbol{K} = c_1 \boldsymbol{K}_s + c_2 \boldsymbol{K}_f, \\
& \boldsymbol{G} = \mathbb{E}_{z\sim\rho_{\pi_\theta}}[\boldsymbol{u}(z)\boldsymbol{u}(z)^\top] \approx \frac{1}{T+1} \boldsymbol{U}\boldsymbol{U}^\top
\end{aligned} \tag{13}$$

the policy gradient mean $\boldsymbol{L}_\theta$ and covariance $\boldsymbol{C}_\theta$ can be computed analytically [29,30],

$$\boldsymbol{L}_\theta = c_2 \boldsymbol{U} \boldsymbol{\alpha}, \tag{14}$$

$$\boldsymbol{C}_\theta = c_2 \boldsymbol{G} - c_2^2 \boldsymbol{U} \left(\boldsymbol{K} + \sigma^2 \mathbf{I}\right)^{-1} \boldsymbol{U}^\top, \tag{15}$$

where $\boldsymbol{U} = \left[\boldsymbol{u}(z_0), \ldots, \boldsymbol{u}(z_T)\right]$.

Before being transferred to the state kernel k_s, a deep neural network (DNN) is used as a feature extractor to enhance the adaptability of the kernel [40]. The kernel parameters $\boldsymbol{\phi}$ (DNN parameters and state kernel parameters) are tuned using the gradient of Gaussian process's log marginal likelihood $J_{\mathbf{GP}}$ as:

$$\begin{aligned}
& J_{\mathbf{GP}}(\boldsymbol{\phi}|\boldsymbol{D}) \propto \log|\boldsymbol{K}_{\boldsymbol{\phi}}| - \boldsymbol{Q}^\top \boldsymbol{K}_{\boldsymbol{\phi}}^{-1} \boldsymbol{Q}, \\
& \nabla_{\boldsymbol{\phi}} J_{\mathbf{GP}} = \boldsymbol{Q}^\top \boldsymbol{K}_{\boldsymbol{\phi}}^{-1} (\nabla_{\boldsymbol{\phi}} \boldsymbol{K}_{\boldsymbol{\phi}}) \boldsymbol{K}_{\boldsymbol{\phi}}^{-1} \boldsymbol{Q} + Tr(\boldsymbol{K}_{\boldsymbol{\phi}}^{-1} \nabla_{\boldsymbol{\phi}} \boldsymbol{K}_{\boldsymbol{\phi}})
\end{aligned} \tag{16}$$

where $\boldsymbol{Q}$ is obtained by (9).

To this end, we combine (16) and (11) with (9) to obtain the natural policy gradient as:

$$\Delta\boldsymbol{\theta} = c_2 \boldsymbol{G}_\theta^{-1} \boldsymbol{U}_\theta \boldsymbol{\alpha}_{\boldsymbol{\phi}} \tag{17}$$

where $\boldsymbol{\alpha}_{\boldsymbol{\phi}} = \boldsymbol{H}^\top \left(\boldsymbol{H}\boldsymbol{K}_{\boldsymbol{\phi}}\boldsymbol{H}^\top + \boldsymbol{\Sigma}_T\right)^{-1} \boldsymbol{c}$, $\boldsymbol{G}_\theta = \boldsymbol{U}_\theta \boldsymbol{U}_\theta^\top / (T+1)$ and $\boldsymbol{U}_\theta = \left[\nabla_\theta \log \boldsymbol{\pi}_\theta(\boldsymbol{a}_0|\boldsymbol{s}_0), \ldots, \nabla_\theta \log \boldsymbol{\pi}_\theta(\boldsymbol{a}_T|\boldsymbol{s}_T)\right]$. Then, at iteration $\tau+1$, the parameters $\boldsymbol{\theta}^{(\tau)}$ are updated as

$$\boldsymbol{\theta}^{(\tau+1)} = \boldsymbol{\theta}^{(\tau)} - \beta \Delta\boldsymbol{\theta}|_{\theta=\theta^{(\tau)}} \tag{18}$$

where β is the learning rate.

Moreover, the $\boldsymbol{Q}_{\boldsymbol{\pi}_\theta}$ can split into a state-value function $\boldsymbol{V}_{\boldsymbol{\pi}_\theta}$ and an advantage function $\boldsymbol{A}_{\boldsymbol{\pi}_\theta}$ as [30]:

$$\begin{aligned}
\boldsymbol{Q}_{\boldsymbol{\pi}_\theta}(z_t) &= (c_1 \mathbf{k}_s(\boldsymbol{s}_t) + c_2 \mathbf{k}_f(z_t))^\top \boldsymbol{\alpha} \\
&= c_1 \mathbf{k}_s(\boldsymbol{s}_t)^\top \boldsymbol{\alpha} + c_2 \mathbf{k}_f(z_t)^\top \boldsymbol{\alpha} \\
&= \boldsymbol{V}_{\boldsymbol{\pi}_\theta}(\boldsymbol{s}_t) + \boldsymbol{A}_{\boldsymbol{\pi}_\theta}(z_t)
\end{aligned} \tag{19}$$

where $\boldsymbol{V}_{\boldsymbol{\pi}_\theta}(\boldsymbol{s}_t) = \mathbb{E}_{\mathbf{a}_t\sim\boldsymbol{\pi}_\theta(\cdot|\boldsymbol{s}_t)}\left[\boldsymbol{Q}_{\boldsymbol{\pi}_\theta}(\boldsymbol{z}_t)\right]$ is the expected return under $\boldsymbol{\pi}_\theta$ from the initial $\boldsymbol{s}_t$, and $\boldsymbol{A}_{\boldsymbol{\pi}_\theta}(\boldsymbol{z}_t)$ denotes the advantage of the specified action $\boldsymbol{a}_t$ over the distribution of the policy $\boldsymbol{\pi}_\theta(\boldsymbol{s}_t)$. $\boldsymbol{V}_{\boldsymbol{\pi}_\theta}$ also can be modeled as a Gaussian process as follows:

$$\begin{aligned}
\mathbb{E}(\boldsymbol{V}(z_i)|\boldsymbol{D}) =& c_1 \boldsymbol{k}_s(\boldsymbol{s}_i)^\top \boldsymbol{\alpha} \\
\mathrm{Cov}\left[\boldsymbol{V}(z_i), \boldsymbol{V}(z_j)|\boldsymbol{D}\right] =& c_1 k_s(\boldsymbol{s}_i, \boldsymbol{s}_j) - c_1^2 \boldsymbol{k}_s(\boldsymbol{s}_i)^\top \boldsymbol{S} \boldsymbol{k}_s(\boldsymbol{s}_j) \\
\mathrm{Cov}\left[\boldsymbol{V}(z_i), \boldsymbol{Q}(z_j)|\boldsymbol{D}\right] =& c_1 k_s(\boldsymbol{s}_i, \boldsymbol{s}_j) - c_1^2 \boldsymbol{k}_s(\boldsymbol{s}_i)^\top \boldsymbol{S} \boldsymbol{k}(z_j)
\end{aligned} \tag{20}$$

where $\boldsymbol{k}_s(\boldsymbol{s}_i) = \left[k_s(\boldsymbol{s}_0, \boldsymbol{s}_i), \ldots, k_s(\boldsymbol{s}_T, \boldsymbol{s}_i)\right]^\top$.

4. Lyapunov Bayesian actor-critic algorithm

We introduce the proposed LBAC algorithm that integrates Lyapunov stability functions into the DBAC framework. Instead of embedding Lyapunov constraints within the policy gradient process as [28], we incorporate these constraints directly into the formulation of the unobserved action-state value function Q.

4.1. Lyapunov theory

The stochastic Lyapunov function plays a crucial role in analyzing and ensuring the stability of MDPs in stochastic stability theory [41].

Definition 2 *(Stochastic Lyapunov function).* A function $V : \mathcal{X} \to \mathbb{R}$ is a stochastic Lyapunov function on a closed set if it satisfies the following conditions:

$$\alpha_1 v(\boldsymbol{s}_t) \le V(\boldsymbol{s}_t) \le \alpha_2 v(\boldsymbol{s}_t) \tag{21}$$

$$\mathbb{E}_{\boldsymbol{s}_{t+1}\sim\boldsymbol{P}} V(\boldsymbol{s}_{t+1}) - V(\boldsymbol{s}_t) \le -\lambda v(\boldsymbol{s}_t) \tag{22}$$

This implies that as the system evolves from $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+1}$, the expected value $\mathbb{E}_{\boldsymbol{s}_{t+1}\sim\boldsymbol{P}} V(\boldsymbol{s}_{t+1})$ decreases, indicating that the system evolves towards the stable point.

We are essentially interested in solving the constrained Gaussian process regression problem for (7), i.e., GPTD with Lyapunov function constraints as:

$$\begin{aligned}
& \boldsymbol{c} = \boldsymbol{H}\boldsymbol{Q}_{\boldsymbol{\pi}_\theta} + \boldsymbol{N} \\
& \text{s.t. } \boldsymbol{B}\boldsymbol{V}_{\boldsymbol{\pi}_\theta} \le -\lambda \boldsymbol{v}
\end{aligned} \tag{23}$$

in which $\boldsymbol{V}_{\boldsymbol{\pi}_\theta}$ is separated from $\boldsymbol{Q}_{\boldsymbol{\pi}_\theta}$ as (19), and

$$\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{s}) = \left[v(\boldsymbol{s}_0), \ldots, v(\boldsymbol{s}_T)\right]^\top \in \mathbb{R}^T$$
$$\boldsymbol{B} = \begin{bmatrix} -1 & 1 & 0 & \ldots & 0 & 0 \\ 0 & -1 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & -1 & 1 \end{bmatrix} \in \mathbb{R}^{T\times(T+1)} \tag{24}$$

The form of $\boldsymbol{B}$ is similar to that used to approximate first-order derivatives of continuous time series data for trending filtering.

Here, $\boldsymbol{V}_{\pi_\theta}$ is selected to be the candidate for the Lyapunov function. $\boldsymbol{B}\boldsymbol{V}_{\pi_\theta} \le -\lambda \boldsymbol{V}$ translates the energy-decreasing condition in (22) into discrete-time space. To ensure that the goal state $\boldsymbol{s}_{\text{goal}}$ is reachable, we let $v(\boldsymbol{s}) = c(\boldsymbol{s})$, therefore $V(\boldsymbol{s}_{\text{goal}}) = 0$ with $c(\boldsymbol{s}_{\text{goal}}) = 0$.

Define $\boldsymbol{B}$ as a linear operator acting on instances of $\boldsymbol{V}_{\pi_\theta}$ that conform to a Gaussian process $\mathcal{GP}(\boldsymbol{\mu}_s(s), c_1\boldsymbol{k}_s(\boldsymbol{s}_p, \boldsymbol{s}_q))$. Given the property of Gaussian processes being closed under linear transformations, applying $\boldsymbol{B}$ to $\boldsymbol{Q}_{\pi_\theta}$ consequently yields another Gaussian process. It is postulated that $\boldsymbol{B}$ transforms a function from a $\mathbb{R}^{T+1}$ domain to another $\mathbb{R}^T$ domain. This paradigm of applying a linear operator to the Gaussian process has been considered in [42,43]. The mean and covariance of $\boldsymbol{B}\boldsymbol{V}_{\pi_\theta}$ are given by applying $\boldsymbol{B}$ to the mean and covariance of the original process:

$$\begin{aligned} \mathbb{E}[\boldsymbol{B}\boldsymbol{V}_{\pi_\theta}] &= \boldsymbol{B}\boldsymbol{\mu}_s, \\ \text{Cov}(\boldsymbol{B}\boldsymbol{V}_{\pi_\theta}, \boldsymbol{B}\boldsymbol{V}_{\pi_\theta}) &= c_1\boldsymbol{B}\boldsymbol{K}_s\boldsymbol{B}^\top \in \mathbb{R}^{T\times T}, \end{aligned} \tag{25}$$

and the cross-covariance is given as

$$\begin{aligned} \text{Cov}(\boldsymbol{B}\boldsymbol{V}_{\pi_\theta}, \boldsymbol{V}_{\pi_\theta}) &= c_1\boldsymbol{B}\boldsymbol{K}_s \in \mathbb{R}^{T\times(T+1)}, \\ \text{Cov}(\boldsymbol{V}_{\pi_\theta}, \boldsymbol{B}\boldsymbol{V}_{\pi_\theta}) &= c_1\boldsymbol{K}_s\boldsymbol{B}^\top \in \mathbb{R}^{(T+1)\times T}. \end{aligned} \tag{26}$$

We will make use of the properties of the Gaussian process under linear transformations to get the new policy gradient update $\Delta\hat{\theta}$.

4.2. Constrained Gaussian process temporal difference learning

Let $\hat{\boldsymbol{Q}}_{\pi_\theta}$ be a Gaussian process $\mathcal{GP}(\hat{\boldsymbol{U}}(\boldsymbol{z}), \hat{K}(\boldsymbol{z}_p, \boldsymbol{z}_q))$ with the linear inequality constraints $\hat{\boldsymbol{c}} = \boldsymbol{B}\hat{\boldsymbol{V}}_{\pi_\theta} \le -\lambda\boldsymbol{V}$ whose t-th element $\hat{c}(\boldsymbol{s}_{t-1}, \boldsymbol{s}_t) = \hat{V}_{\pi_\theta}(\boldsymbol{s}_t) - \hat{V}_{\pi_\theta}(\boldsymbol{s}_{t-1}) + \varepsilon_t$ for the i.i.d. $\varepsilon_t \sim \mathcal{N}(0, \sigma_v^2)$. We focus on modeling the posterior Gaussian process conditioned on the dataset $\boldsymbol{D}$ and on the event that $\boldsymbol{B}\hat{\boldsymbol{V}}_{\pi_\theta}(\boldsymbol{s}) \le -\lambda\boldsymbol{V}(\boldsymbol{s})$. Let $\hat{C}(\boldsymbol{D})$ denote the event $\hat{C}(\boldsymbol{D}) := \cap_{t=1}^T \{\hat{c}(\boldsymbol{s}_{t-1}, \boldsymbol{s}_t) \le -\lambda v(\boldsymbol{s}_{t-1})\}$, where the constraint $\hat{c}(\mathbf{s}_{t-1}, \mathbf{s}_t) + \varepsilon \le -\lambda v(\boldsymbol{s}_{t-1})$ is satisfied for all the points in $\boldsymbol{D}$. Therefore, the process can be considered as follows.

$$\begin{aligned} &\left\{\hat{\boldsymbol{Q}}_{\pi_\theta} | \boldsymbol{D}, \boldsymbol{c}, \hat{C}(\boldsymbol{D})\right\} \\ &= \left\{\hat{\boldsymbol{Q}}_{\pi_\theta} | \boldsymbol{c} = \boldsymbol{H}\hat{\boldsymbol{Q}}_{\pi_\theta} + \boldsymbol{N}, \boldsymbol{B}\hat{\boldsymbol{V}}_{\pi_\theta} + \boldsymbol{\varepsilon} \le -\lambda\boldsymbol{V}\right\} \end{aligned} \tag{27}$$

where $\hat{\boldsymbol{Q}}_{\pi_\theta}(\boldsymbol{z})$ is a Gaussian process, the data set $\boldsymbol{D}$ consists of the state-action pairs $\boldsymbol{z}$ and cost $\boldsymbol{c}$, $\boldsymbol{N}$ and $\boldsymbol{\varepsilon}$ are multivariate Gaussian with diagonal covariance matrices of elements σ^2, respectively.

4.3. Updated policy gradient evaluation

With the update of $\hat{\boldsymbol{Q}}_{\pi_\theta}$, the expression of $\boldsymbol{\alpha}$ in (17) requires a revision to $\hat{\boldsymbol{\alpha}}$, to obtain the estimate of the policy gradient $\hat{\boldsymbol{\theta}}$. The crux of this process lies in the derivation of the posterior predictive distribution for $\{\hat{\boldsymbol{Q}}^*_{\pi_\theta} | \boldsymbol{z}^*, \boldsymbol{D}, \boldsymbol{c}, \hat{C}(\boldsymbol{D})\}$. This involves determining the distribution of $\boldsymbol{Q}^*_{\pi_\theta}$ for new inputs $\boldsymbol{z}^*$, conditioned on the observed data $\mathbf{c} = \boldsymbol{B}\hat{\boldsymbol{Q}}_{\pi_\theta} + \boldsymbol{N}$ and the constraint $\boldsymbol{B}\hat{\boldsymbol{Q}}_{\pi_\theta} + \boldsymbol{\varepsilon} \le \mathbf{0}$.

The posterior predictive distribution of $\{\hat{\boldsymbol{Q}}^*_{\pi_\theta} | \boldsymbol{z}, \boldsymbol{c}, \hat{C}\}$ manifests itself as a compound Gaussian characterized by a truncated Gaussian mean as follows:

$$\begin{aligned} &\left\{\hat{\boldsymbol{Q}}^*_{\pi_\theta} | \hat{C}(\boldsymbol{z}), \boldsymbol{z}, \boldsymbol{c}\right\} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}^* + \hat{\boldsymbol{A}}(\boldsymbol{C} - \boldsymbol{B}\hat{\boldsymbol{\mu}}) + \hat{\boldsymbol{B}}(\boldsymbol{c} - \boldsymbol{H}\hat{\boldsymbol{\mu}}), \boldsymbol{\Sigma}), \\ &\left\{\boldsymbol{C} = \hat{\boldsymbol{c}} | \boldsymbol{z}, \boldsymbol{c}, \hat{C}\right\} \sim \mathcal{TN}(\boldsymbol{B}\hat{\boldsymbol{\mu}} + \hat{\boldsymbol{A}}_1(\boldsymbol{c} - \boldsymbol{H}\hat{\boldsymbol{\mu}}_s), \hat{\boldsymbol{B}}_1, -\infty, -\lambda\boldsymbol{v}), \end{aligned} \tag{28}$$

where $\hat{\boldsymbol{c}} = \left[\hat{c}(\boldsymbol{s}_0, \boldsymbol{s}_1), \hat{c}(\boldsymbol{s}_1, \boldsymbol{s}_2), \ldots, \hat{c}(\boldsymbol{s}_{T-1}, \boldsymbol{s}_T)\right]^\top \in \mathbb{R}^T$, $\mathcal{TN}(\cdot, \cdot, -\infty, -\lambda\boldsymbol{v})$ is the truncated Gaussian distribution $\mathcal{N}(\cdot, \cdot)$ conditioned on the hyper-rectangle $(-\infty, -\lambda v(\boldsymbol{s}_0)] \times \cdots \times (-\infty, -\lambda v(\boldsymbol{s}_{T-1})]$, and

$$\begin{aligned} &\hat{\boldsymbol{A}}_1 = c_1\boldsymbol{B}\boldsymbol{K}_s\boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma}_T)^{-1}, \\ &\hat{\boldsymbol{A}}_2 = \mathbf{k}(\boldsymbol{z}^*)^\top\boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma}_T)^{-1}, \\ &\hat{\boldsymbol{B}}_1 = c_1\boldsymbol{B}\boldsymbol{K}_s\boldsymbol{B}^\top + \sigma_v^2\boldsymbol{I}_N - c_1\hat{\boldsymbol{A}}_1\boldsymbol{H}\boldsymbol{K}_s\boldsymbol{B}^\top, \\ &\hat{\boldsymbol{B}}_2 = \boldsymbol{K}_{\boldsymbol{z}^*,\boldsymbol{z}^*} - \hat{\boldsymbol{A}}_2\boldsymbol{H}\boldsymbol{k}(\boldsymbol{z}^*), \\ &\hat{\boldsymbol{B}}_3 = c_1\boldsymbol{k}_s(\boldsymbol{s}^*)^\top\boldsymbol{B}^\top - c_1\hat{\boldsymbol{A}}_2\boldsymbol{H}\boldsymbol{K}_s\boldsymbol{B}^\top, \\ &\hat{\boldsymbol{A}} = \hat{\boldsymbol{B}}_3\hat{\boldsymbol{B}}_1^{-1}, \quad \hat{\boldsymbol{B}} = \hat{\boldsymbol{A}}_2 - \hat{\boldsymbol{A}}\hat{\boldsymbol{A}}_1, \quad \boldsymbol{\Sigma} = \hat{\boldsymbol{B}}_2 - \hat{\boldsymbol{A}}\hat{\boldsymbol{B}}_3^\top. \end{aligned}$$

Proof 1. We start by observing that $(\hat{\boldsymbol{Q}}^*_{\pi_\theta}, \boldsymbol{C}, \boldsymbol{c})$ is a jointly Gaussian with mean and covariance.

$$\begin{aligned} &\mathbb{E}([\hat{\boldsymbol{Q}}^*_{\pi_\theta}, \boldsymbol{C}, \boldsymbol{c}]^\top) = [\hat{\boldsymbol{\mu}}^*, \boldsymbol{B}\hat{\boldsymbol{\mu}}_s, \boldsymbol{H}\hat{\boldsymbol{\mu}}]^\top, \\ &\text{Cov}([\hat{\boldsymbol{Q}}^*_{\pi_\theta}, \boldsymbol{C}, \boldsymbol{c}]^\top) = \\ &\quad \begin{bmatrix} \boldsymbol{K}_{\boldsymbol{z}^*,\boldsymbol{z}^*} & c_1\boldsymbol{k}_s(\boldsymbol{s}^*)^\top\boldsymbol{B}^\top & \boldsymbol{k}(\boldsymbol{z}^*)^\top\boldsymbol{H}^\top \\ c_1\boldsymbol{B}\boldsymbol{k}_s(\boldsymbol{s}^*) & c_1\boldsymbol{B}\boldsymbol{K}_s\boldsymbol{B}^\top + \sigma_v^2\boldsymbol{I}_T & c_1\boldsymbol{B}\boldsymbol{K}_s\boldsymbol{H}^\top \\ \boldsymbol{H}\boldsymbol{k}(\boldsymbol{z}^*) & c_1\boldsymbol{H}\boldsymbol{K}_s\boldsymbol{B}^\top & \boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma}_T \end{bmatrix}. \end{aligned}$$

Conditioned on $\boldsymbol{c}$, we obtain

$$\begin{bmatrix} \hat{\boldsymbol{Q}}^*_{\pi_\theta} \\ \boldsymbol{C} \end{bmatrix} | \boldsymbol{c} \sim \mathcal{N}\left(\begin{bmatrix} \hat{\boldsymbol{\mu}}^* + \hat{\boldsymbol{A}}_2(\boldsymbol{c} - \boldsymbol{H}\hat{\boldsymbol{\mu}}) \\ \boldsymbol{B}\hat{\boldsymbol{\mu}}_s + \hat{\boldsymbol{A}}_1(\boldsymbol{c} - \boldsymbol{H}\hat{\boldsymbol{\mu}}_s) \end{bmatrix}, \begin{bmatrix} \hat{\boldsymbol{B}}_2 & \hat{\boldsymbol{B}}_3 \\ \hat{\boldsymbol{B}}_3^\top & \hat{\boldsymbol{B}}_1 \end{bmatrix}\right). \tag{29}$$

Conditioning on $\boldsymbol{C}$, then we have

$$\hat{\boldsymbol{Q}}^*_{\pi_\theta} | \boldsymbol{c}, \boldsymbol{C} \sim \mathcal{N}\left(\boldsymbol{\mu}^* + \hat{\boldsymbol{A}}(\boldsymbol{C} - \boldsymbol{B}\hat{\boldsymbol{\mu}}_s) + \hat{\boldsymbol{B}}(\boldsymbol{c} - \boldsymbol{H}\hat{\boldsymbol{\mu}}), \boldsymbol{\Sigma}_T\right), \tag{30}$$

where $\hat{\boldsymbol{A}} = \hat{\boldsymbol{B}}_3\hat{\boldsymbol{B}}_1^{-1}$, $\hat{\boldsymbol{B}} = \hat{\boldsymbol{A}}_2 - \hat{\boldsymbol{A}}\hat{\boldsymbol{A}}_1$ and $\boldsymbol{\Sigma}_T = \hat{\boldsymbol{B}}_2 - \hat{\boldsymbol{A}}\hat{\boldsymbol{B}}_3^\top$.

Assuming the prior mean function $\hat{\boldsymbol{\mu}}$ is established as zero, the posterior predictive mean function can be derived as:

$$\begin{aligned} &\mathbb{E}(\hat{\boldsymbol{Q}}_{\pi_\theta} | \boldsymbol{D}, \mathbf{c}, \hat{C}(\boldsymbol{D})) \\ &= (\hat{\boldsymbol{A}}_2 - \hat{\boldsymbol{A}}\hat{\boldsymbol{A}}_1)\boldsymbol{c} + \hat{\boldsymbol{B}}_3\hat{\boldsymbol{B}}_1^{-1}\boldsymbol{C} \\ &= \left(c_1\boldsymbol{k}_s(\boldsymbol{s}^*)^\top\boldsymbol{B}^\top - c_1\boldsymbol{k}(\boldsymbol{z}^*)^\top\boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma}_T)^{-1}\boldsymbol{H}\boldsymbol{K}_s\boldsymbol{B}^\top\right) \\ &\quad \hat{\boldsymbol{B}}_1^{-1}(\boldsymbol{C} - \hat{\boldsymbol{B}}_1\boldsymbol{c}) + \boldsymbol{k}(\boldsymbol{z}^*)^\top\boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma}_T)^{-1}\boldsymbol{c} \\ &= \boldsymbol{k}(\boldsymbol{z}^*)^\top(\boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma}_T)^{-1}\boldsymbol{c} \\ &\quad - c_1\boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma}_T)^{-1}\boldsymbol{H}\boldsymbol{K}_s\boldsymbol{B}^\top\hat{\boldsymbol{B}}_1^{-1}(\boldsymbol{C} - \hat{\boldsymbol{B}}_1\boldsymbol{c})) \\ &\quad + c_1\boldsymbol{k}_s(\boldsymbol{s}^*)^\top\boldsymbol{B}^\top\hat{\boldsymbol{B}}_1^{-1}(\boldsymbol{C} - \hat{\boldsymbol{A}}_1\boldsymbol{c}) \\ &= \boldsymbol{k}(\boldsymbol{z}^*)^\top\hat{\boldsymbol{\alpha}} + \boldsymbol{k}_s(\boldsymbol{s}^*)^\top\boldsymbol{\beta} \end{aligned}$$

where $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha} - c_1\boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma}_T)^{-1}\boldsymbol{H}\boldsymbol{K}_s\boldsymbol{B}^\top\hat{\boldsymbol{B}}_1^{-1}(\boldsymbol{C} - \hat{\boldsymbol{A}}_1\boldsymbol{c})$, $\boldsymbol{\beta} = c_1\boldsymbol{B}^\top\hat{\boldsymbol{B}}_1^{-1}(\boldsymbol{C} - \hat{\boldsymbol{A}}_1\boldsymbol{c})$ and $\boldsymbol{\alpha} = (\boldsymbol{H}^\top(\boldsymbol{H}\boldsymbol{K}\boldsymbol{H}^\top + \boldsymbol{\Sigma}_T)^{-1}\boldsymbol{c}$. Therefore, the evaluation of the policy gradient is updated as $\Delta\hat{\boldsymbol{\theta}} = c_2\boldsymbol{G}_\theta^{-1}\boldsymbol{U}_\theta\hat{\boldsymbol{\alpha}}$.

The LBAC framework integrates updates to the policy gradient with constraints of the Lyapunov function, the training process of which is off-line as illustrated in Fig. 3. During the training process, we repeatedly run the PMD simulator with the updated policy $\boldsymbol{\pi}_\theta$ and the initial state $\boldsymbol{s}_0 \sim \boldsymbol{\rho}_0$. At each step, we collect data samples and these historical data are stored as $\{\boldsymbol{s}_t, \boldsymbol{a}_t, c_t, \boldsymbol{s}_{t+1}\}$ in the memory. In each policy evaluation, we randomly sample a batch of historical data from the memory to update the policy $\boldsymbol{\pi}_\theta$ and the Lyapunov function $\boldsymbol{Q}_\phi$. The whole training process is described in Algorithm 1.

5. Numerical simulation and experiment

In this section, a thorough evaluation of the proposed LBAC and DBAC algorithms for the PMD task is provided. Initially, we elaborate on the design and operational framework of LBAC, veering into a detailed discourse on its convergence properties and empirical performance in

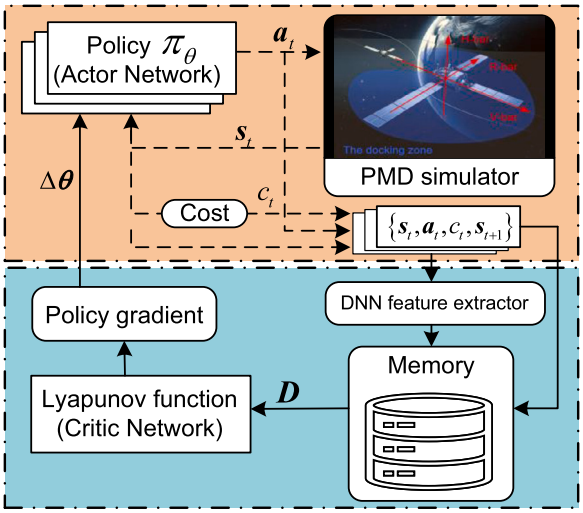

Fig. 3. Overview of the LBAC framework.

Algorithm 1 Lyapunov Bayesian Actor-Critic (LBAC).

Require: Initial policy parameters θ, episodes for gradient evaluation M, termination threshold ϵ, learning rate β

for $i = 0$ to M **do**
  Sample s_0 according to ρ_0,
  for $t = 0$ to T **do**
    Sample a_t from $\pi_\theta(a_t|s_t)$;
    Step forward and observe s_{t+1}, c_t;
    Store $\{s_t, a_t, c_t, s_{t+1}\}$ in D;
  end for
  done = false
  while not done **do**
    Update $\hat{\alpha}$, U and G^{-1} according to (28);
    $\theta = \theta - \beta\Delta\theta$ with $\Delta\theta = c_2 G^{-1} U \hat{\alpha}$;
    if $\Delta\theta < \epsilon$, **then** done = true;
  end while
end for
Return $\theta^* = \theta$.

the simulation. Subsequently, the LBAC's "sim-to-real" applicability is validated via an air-bearing test platform, specifically tailored for the PMD task. For additional insights and a more granular understanding, Supplementary Materials, including a series of experimental videos, are made available.

5.1. Experiment setup

We employ an air-bearing test bed, a nearly frictionless planar environment, to replicate the dynamics of space, as described by [26]. This innovative setup is crucial for accurately simulating the conditions of outer space, providing a realistic platform for rigorous testing and validation. As illustrated in Figs. 4 and 5, the system features a Chaser spacecraft simulator. It achieves levitation through air bearings and is powered by eight ducted fans that generate the necessary forces and torques for maneuverability. The simulator is equipped with an onboard computer that features an Intel Core N2920 processor and 4 GB of RAM, running on a real-time operating system. This configuration ensures the precise and timely execution of the controller/policy and the effective processing of sensor data. In addition, the simulator employs fluorescent markers for accurate attitude and position measurement and is supported by a compressed air tank to support the air bearings.

The physical specifications of the Chaser simulator are designed conscientiously to mirror those of a real spacecraft, significantly enhancing the authenticity of the simulation experience. It has a mass of $m = 9.6\,\mathrm{kg}$ and a moment of inertia around the z-axis of $I_z = 0.23\,\mathrm{kgm}^2$. The simula-

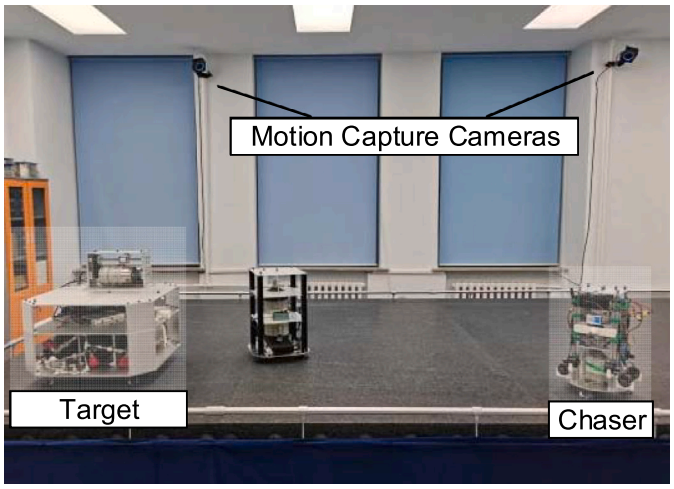

Fig. 4. Experiment setup for the PMD task.

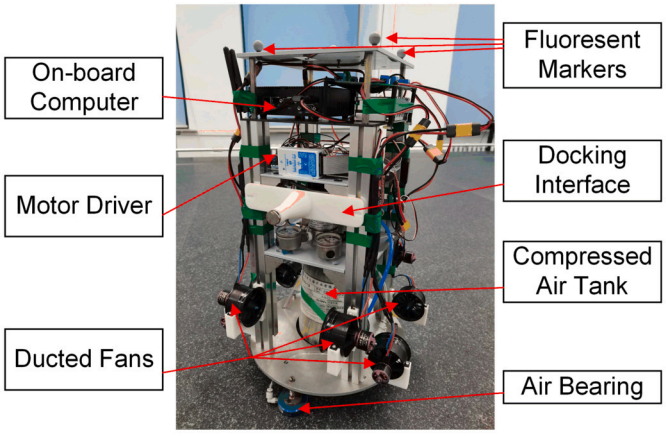

Fig. 5. The Chaser spacecraft simulator.

Table 1
Specific parameters of the system.

Parameter	Value
Sample interval	0.05s
(a_1, a_2, a_3)	(0.6, 0.6, 0.8)m
(b_1, b_2)	(0.6, 0.6)m
k_a, k_r	1.0
Q_a	diag([0.025, 0.025, 0.075])
Q_b	diag([0.0125, 0.0125, 0.175])
P_b	diag([20, 20, 0])s^{-2}
N	$I_{3\times3}$
σ	0.125
ψ	0.2

Table 2
Hyperparameter settings in LBAC and DBAC.

Hyperparameter	Value
Batch size	20000
Total episode	1300
Discount factor γ	0.99
Trust region constrain/step size	0.01
c_1 (for kernel k_s)	1
c_2 (for kernel k_f)	5×10^{-5}
Gaussian process noise σ^2	10^{-4}

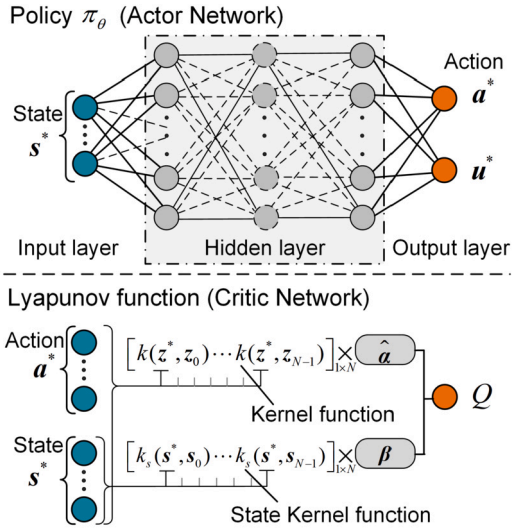

Fig. 6. Approximation of π_θ and Q_{π_θ}.

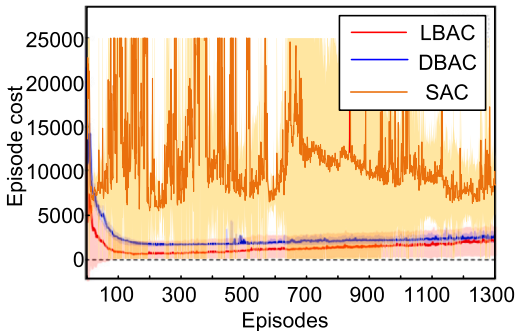

Fig. 7. The cumulative cost during training.

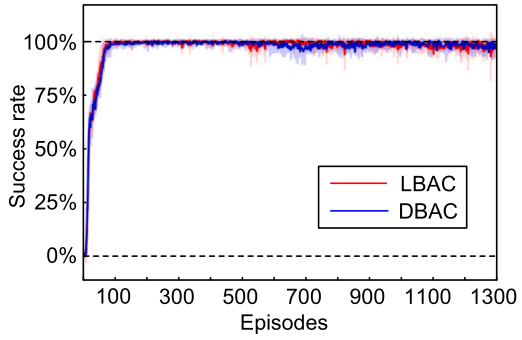

Fig. 8. The success rate during training.

Table 3
Comparison of the maneuvering time.

Algorithms	Total time(s)
LBAC	16.8
DBAC	17.5
APF	24.6

tor control system is capable of exerting a maximum force of $f_{\max} = 1\,\mathrm{N}$ and a torque of $\tau_{\max} = 0.5\,\mathrm{Nm}$.

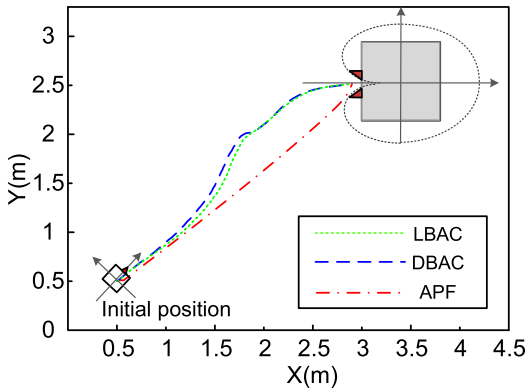

Fig. 9. The trajectories of the Chaser.

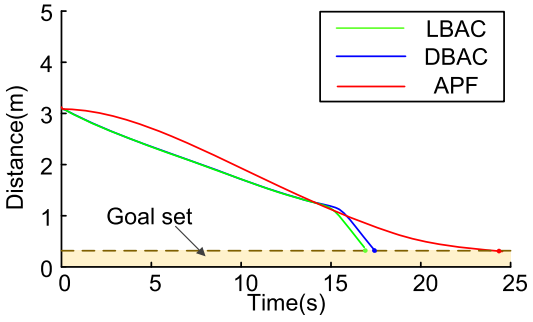

Fig. 10. The distance between the Chaser and the goal state.

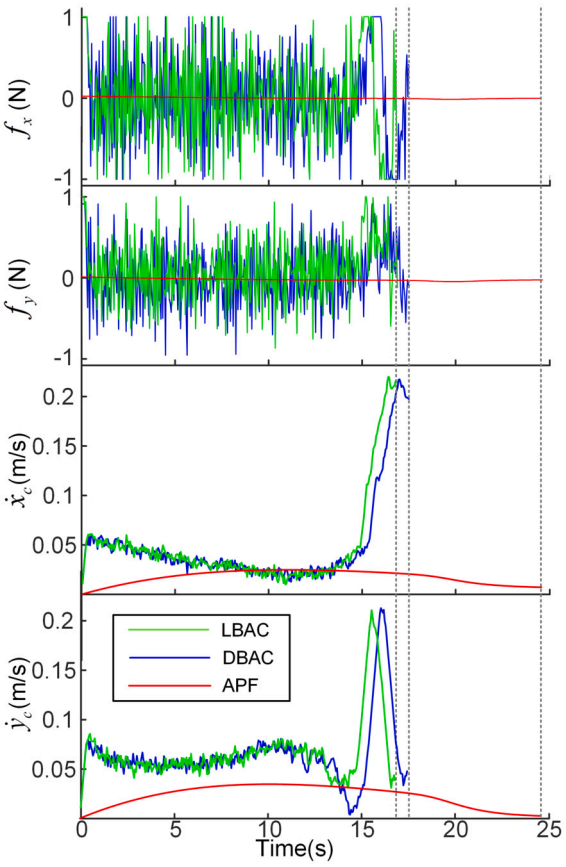

Fig. 11. The Chaser's velocity and force.

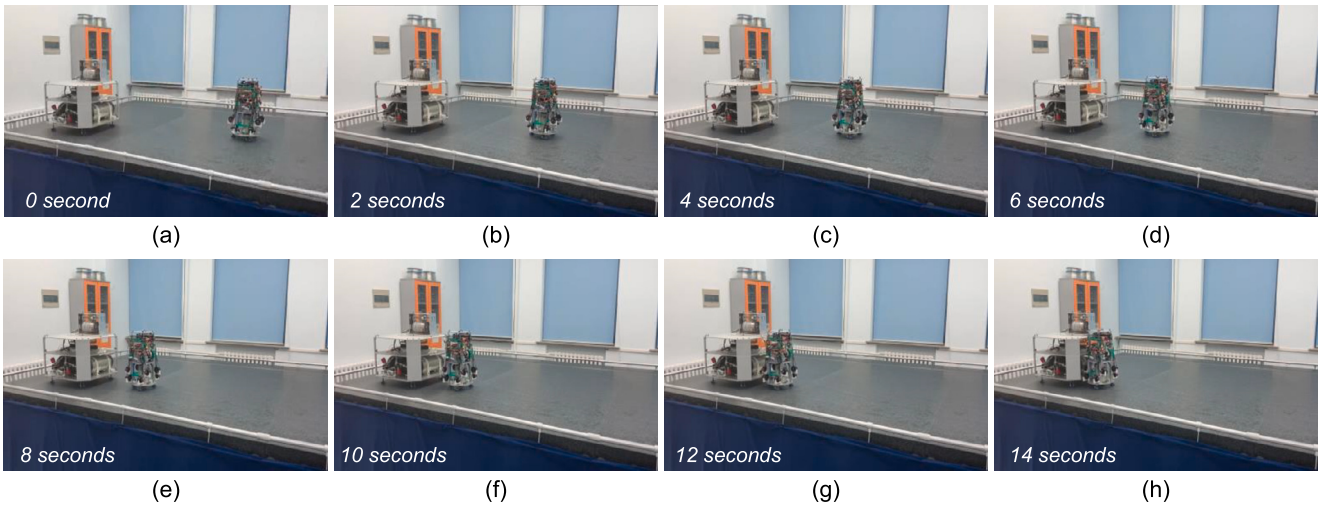

Fig. 12. The trajectory of the Chaser in the air bearing testbed.

5.2. Convergence in simulation

According to the setup of the experiment, the state space is set as $\mathcal{S} = \left\{\boldsymbol{s} : \boldsymbol{s}_{\text{lb}} \leq \boldsymbol{s} \leq \boldsymbol{s}_{\text{ub}}\right\}$ with $\boldsymbol{s}_{\text{lb}} = [0\,\text{m}, 0\,\text{m}, -\pi\text{rad}, -1\,\text{m/s}, -1\,\text{m/s}, -\pi\text{rad/s}]^{\top}$ and $\boldsymbol{s}_{\text{ub}} = [4\,\text{m}, 3\,\text{m}, \pi\text{rad}, 1\,\text{m/s}, 1\,\text{m/s}, \pi\text{rad/s}]^{\top}$, and the goal set is $\mathcal{S}_{\text{goal}} = \left\{\boldsymbol{s} : \sqrt{x_{fc}^2 + y_{fc}^2} \leq 0.3\,\text{m}\right\}$. The action space is set as $\mathcal{A} = \left\{\boldsymbol{a} : -\boldsymbol{a}_{\text{b}} \leq \boldsymbol{a} \leq \boldsymbol{a}_{\text{b}}\right\}$ with $\boldsymbol{a}_{\text{b}} = [1\,\text{N}, 1\,\text{N}, 0.5\,\text{Nm}]$. Other detailed simulation parameters are cataloged in Table 1. Within the simulation, an episode ends prematurely if the Chaser exceeds the bounds of $\mathcal{S}$ or successfully enters the set of goals $\mathcal{S}_{\text{goal}}$. Terminal costs are assigned as $c_{\text{end}} = 25000$ for exceeding state-space limits and $c_{\text{end}} = 0$ for reaching the goal state, establishing a clear metric for success and boundary conditions in the simulation environment.

In LBAC and DBAC algorithms, the policy $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ is instantiated as a DNN, specifically a three-layer multilayer perceptron, as shown in Fig. 6. Each hidden layer comprises 128 hidden units and utilizes the hyperbolic tangent activation function. The Lyapunov function $Q_{\boldsymbol{\pi}_{\boldsymbol{\theta}}}$ is modeled as the Gaussian process, employing a composite kernel defined as $k = c_1 k_s + c_2 k_f$. Here, k_s represents the deep radial basis function kernel [40], while k_f denotes the Fisher kernel. Furthermore, the hyperparameter settings for the LBAC and DBAC algorithms are detailed in Table 2. Note that inspired by the methodology outlined in [30], the implementation is instantiated on the *GPyTorch* [44], harnessing the computational efficiency of GPU-accelerated black-box matrix operations.

We employ the SAC, LBAC and DBAC algorithms to train the policy, maintaining uniform parameter configurations across these methodologies. Each algorithm is subjected to a rigorous training regimen spanning 1300 episodes, with five different random seeds to evaluate the consistency and robustness of the algorithms. As depicted in Fig. 7, both the LBAC and DBAC algorithms exhibit a notable propensity for rapid convergence, typically achieving this within the first 130 episodes, while the SAC algorithm still has no convergence trend before 1300 episodes. This rapid convergence underscores the effectiveness of the algorithms in mastering the nuances of the PMD task. Furthermore, Fig. 8 presents a progressive increase in the average success rate for the DBAC and LBAC algorithms, culminating in an impressive success rate of approximately 99% after just about 110 episodes. In contrast, the SAC algorithm has had little success. A salient finding from our experimental analysis is the marginally lower total cost incurred by LBAC compared to DBAC, albeit with a somewhat higher variance observed in LBAC. Therefore, employing a broader range of random seeds increases the likelihood of obtaining policies of superior efficacy using the LBAC algorithm.

It should be noted that our attempts to implement popular RL algorithms, such as PPO [17] and SAC [31], in the PMD task context were met with comparatively less success. Despite serious efforts, these algorithms demonstrated challenges in attaining convergence throughout the training process. This observation accentuates the specialized adaptability and efficacy of both LBAC and DBAC for addressing the intricate requirements of the PMD task, in stark contrast to the performance of other RL algorithms.

5.3. Performance in simulation

To validate the effectiveness and superiority of our proposed algorithms, we perform a comparative simulation with the APF method [26]. The parameters of the APF method are aligned with those specified in Table 1. This evaluation pits the performance of policies trained by the LBAC and DBAC algorithms against the APF method, with the initial relative position and velocity settings between the Chaser and the goal state established as $\boldsymbol{x}_{fc} = [2.4\,\text{m}, 2\,\text{m}, 0.9\text{rad}]^{\top}$ and $\dot{\boldsymbol{x}}_{fc} = \left[0\,\text{m/s}, 0\,\text{m/s}, 0\text{rad/s}\right]^{\top}$, respectively. The trajectories of the different methods are shown in Fig. 9, and the time histories of the distance to the goal position and the velocity and the control force of the Chaser are shown in Figs. 10 and 11. Note that the dashed line in Fig. 11 represents the end of the PMD task. Table 3 enumerates the maneuvering time for each algorithm.

Our analysis reveals that the maneuvering time of the LBAC is marginally shorter than that of the DBAC and significantly shorter than that of the APF. Furthermore, the trajectories generated by LBAC and DBAC exhibit a pronounced preference for safety, maintaining a greater separation from the Target boundary than those produced by the APF method. This comparison underscores the efficiency and safety advantages of the LBAC and DBAC algorithms that execute the PMD task.

5.4. Evaluation in the air-bearing testbed

To validate the practical applicability of policies trained using the LBAC algorithm, we transition from simulation to the air-bearing testbed. The most effective policy, identified based on its performance in five random seeds, is selected for real-world evaluation. Each test is initiated from various positions and velocities, manually set to introduce a diverse range of starting conditions. Our experiment includes 10 distinct trials. The results are encouraging, with the policy achieving a high success rate of approximately 90%, as evidenced by 9 successful maneuvers out of 10 attempts. This high success rate underscores the efficacy and promise of the LBAC approach in the real world, although it also highlights the challenges in achieving universal success.

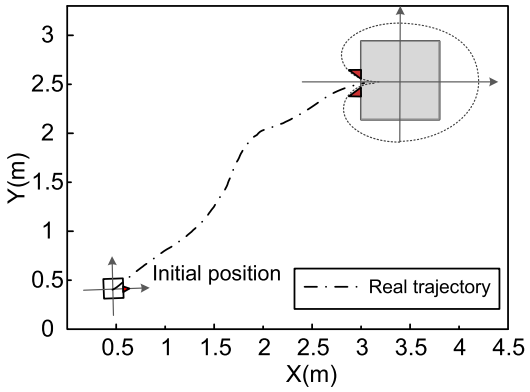

Fig. 13. The trajectory of the Chaser.

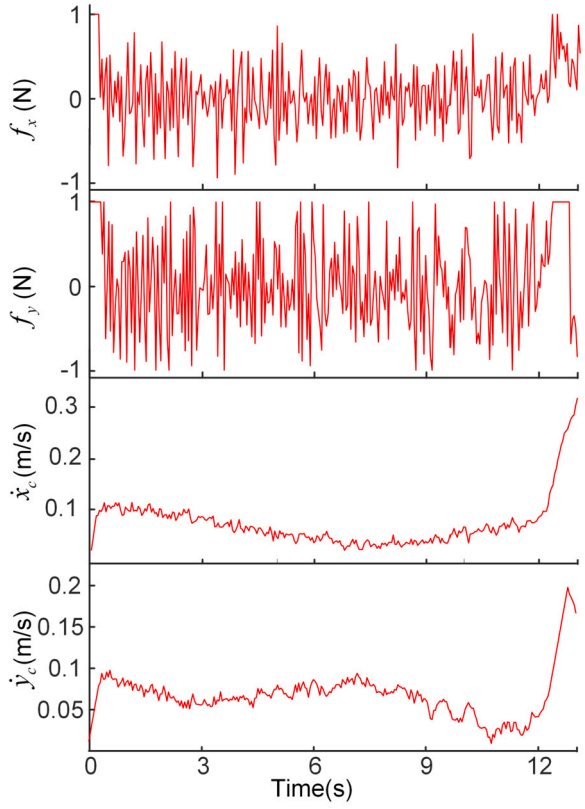

Fig. 14. The time histories of the state and action.

To illustrate the performance of the LBAC algorithm, one of the successful trajectories is depicted in Fig. 12, and the trajectory and time histories of the state and action are shown in Figs. 13 and 14. The results indicate that the Chaser successfully completes the PMD task, reaching the goal position without entering the docking cone corridor. However, there was one instance of failure where the Chaser deviated from its intended path, unexpectedly moving out of the test field instead of approaching the Target. We hypothesize that this failure may stem from discrepancies between the simulated and real-world dynamics, as well as the possibility of encountering states not covered during training. This highlights a significant challenge in the application of RL algorithms, emphasizing the need to adapt to diverse and unexpected scenarios. Documentation of these trials, including video evidence of successful and unsuccessful attempts, is provided in Supplementary Materials (Movie S1).

6. Conclusion

In this paper, we develop a novel LBAC algorithm to accomplish the PMD task within the docking cone constraint. The concerned problem is divided into three parts: First, the PMD task is required to be modeled as a MDP, which is suitable to be solved with RL algorithms; Second, we propose a constrained Gaussian process temporal difference learning, integrating Lyapunov-based stability constraints; Third, a novel Bayesian quadrature policy optimization is introduced to analytically compute the policy gradient while ensuring stability through Lyapunov constraints. Our results demonstrate that both LBAC and DBAC offer significant efficiency and safety advantages over traditional APF methods in simulations, with LBAC marginally outperforming DBAC in terms of maneuvering time (16.8 s vs. 17.5 s). Furthermore, the sim-to-real capability of the LBAC algorithm was validated on an air-bearing testbed, achieving an impressive 90% success rate.

For future work, our aim is to extend our methodology to include the detection and avoidance of static and dynamic obstacles within the PMD task. In addition, we plan to develop novel model-based RL algorithms that integrate control theory and dynamic model information, ensuring improved stability and safety in complex environments.

CRediT authorship contribution statement

Desong Du: Writing – review & editing, Writing – original draft, Software, Data curation. **Yanfang Liu:** Writing – review & editing, Funding acquisition. **Ouyang Zhang:** Software, Formal analysis, Data curation. **Naiming Qi:** Supervision, Project administration. **Weiran Yao:** Writing – review & editing, Visualization. **Wei Pan:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Acknowledgement

This work was supported in part by China Scholarship Council (202006120130), the National Key Research and Development Program (2022YFB3902701), National Science Foundation of China under Grant 52272390, in part by Natural Science Foundation of Heilongjiang Province of China under Grant YQ2022A009 and National High-Level Young Scholars Program (Q2022335).

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ast.2024.109474.

References

[1] A. Flores-Abad, O. Ma, K. Pham, S. Ulrich, A review of space robotics technologies for on-orbit servicing, Prog. Aerosp. Sci. 68 (2014) 1–26.
[2] M.B. Quadrelli, L.J. Wood, J.E. Riedel, M.C. McHenry, M. Aung, L.A. Cangahuala, R.A. Volpe, P.M. Beauchamp, J.A. Cutts, Guidance, navigation, and control technology assessment for future planetary science missions, J. Guid. Control Dyn. 38 (7) (2015) 1165–1186.
[3] R. Zappulla, H. Park, J. Virgili-Llop, M. Romano, Real-time autonomous spacecraft proximity maneuvers and docking using an adaptive artificial potential field approach, IEEE Trans. Control Syst. Technol. 27 (6) (2018) 2598–2605.
[4] C. Zagaris, M. Baldwin, C. Jewison, C. Petersen, Survey of spacecraft rendezvous and proximity guidance algorithms for on-board implementation, Adv. Astronaut. Sci. 155 (2015) 131–150.

[5] M. Mancini, N. Bloise, E. Capello, E. Punta, Sliding mode control techniques and artificial potential field for dynamic collision avoidance in rendezvous maneuvers, IEEE Control Syst. Lett. 4 (2) (2019) 313–318.
[6] M. Romano, D.A. Friedman, T.J. Shay, Laboratory experimentation of autonomous spacecraft approach and docking to a collaborative target, J. Spacecr. Rockets 44 (1) (2007) 164–173.
[7] Y. Guo, D. Zhang, A. Li, S. Song, C. Wang, Z. Liu, Finite-time control for autonomous rendezvous and docking under safe constraint, Aerosp. Sci. Technol. 109 (2021) 106380.
[8] M. Mammarella, M. Lorenzen, E. Capello, H. Park, F. Dabbene, G. Guglieri, M. Romano, F. Allgöwer, An offline-sampling smpc framework with application to autonomous space maneuvers, IEEE Trans. Control Syst. Technol. 28 (2) (2018) 388–402.
[9] M. Mammarella, E. Capello, H. Park, G. Guglieri, M. Romano, Tube-based robust model predictive control for spacecraft proximity operations in the presence of persistent disturbance, Aerosp. Sci. Technol. 77 (2018) 585–594.
[10] I. Lopez, C.R. Mclnnes, Autonomous rendezvous using artificial potential function guidance, J. Guid. Control Dyn. 18 (2) (1995) 237–241.
[11] S.B. McCamish, M. Romano, S. Nolet, C.M. Edwards, D.W. Miller, Flight testing of multiple-spacecraft control on spheres during close-proximity operations, J. Spacecr. Rockets 46 (6) (2009) 1202–1213.
[12] X. Huang, S. Li, B. Yang, P. Sun, X. Liu, X. Liu, Spacecraft guidance and control based on artificial intelligence, Rev., Acta Aeronaut. Astronaut. Sin. 42 (2021) 524201.
[13] K. Thangavel, R. Sabatini, A. Gardi, K. Ranasinghe, S. Hilton, P. Servidia, D. Spiller, Artificial intelligence for trusted autonomous satellite operations, Prog. Aerosp. Sci. 144 (2024) 100960.
[14] B. Gaudet, R. Furfaro, R. Linares, Reinforcement learning for angle-only intercept guidance of maneuvering targets, Aerosp. Sci. Technol. 99 (2020) 105746.
[15] H. Yuan, D. Li, Deep reinforcement learning for rendezvous guidance with enhanced angles-only observability, Aerosp. Sci. Technol. 129 (2022) 107812.
[16] G. Peng, B. Wang, L. Liu, H. Fan, Z. Cheng, Real-time adaptive entry trajectory generation with modular policy and deep reinforcement learning, Aerosp. Sci. Technol. 142 (2023) 108594.
[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint, arXiv:1707.06347, 2017.
[18] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint, arXiv:1509.02971, 2015.
[19] O. Zhang, W. Yao, D. Du, C. Wu, J. Liu, L. Wu, Y. Sun, Trajectory optimization and tracking control of free-flying space robots for capturing non-cooperative tumbling objects, Aerosp. Sci. Technol. 143 (2023) 108718.
[20] J. Broida, R. Linares, Spacecraft rendezvous guidance in cluttered environments via reinforcement learning, in: Proceedings of the 29th AAS/AIAA Space Flight Mechanics Meeting, American Astronautical Society, Hawaii, USA, 2019.
[21] Q. Qu, K. Liu, W. Wang, J. Lü, Spacecraft proximity maneuvering and rendezvous with collision avoidance based on reinforcement learning, IEEE Trans. Aerosp. Electron. Syst. 58 (6) (2022) 5823–5834.
[22] Z. Yang, L. Xing, Z. Gu, Y. Xiao, Y. Zhou, Z. Huang, L. Xue, Model-based reinforcement learning and neural-network-based policy compression for spacecraft rendezvous on resource-constrained embedded systems, IEEE Trans. Ind. Inform. 19 (1) (2022) 1107–1116.
[23] A.M. Lyapunov, The general problem of the stability of motion, Ann. Math. Stud. (1892).
[24] F. Berkenkamp, M. Turchetta, A. Schoellig, A. Krause, Safe model-based reinforcement learning with stability guarantees, in: Proceedings of the 31st Advances in Neural Information Processing Systems, The MIT Press, Long Beach, USA, 2017.
[25] Y.-C. Chang, N. Roohi, S. Gao, Neural Lyapunov control, in: Proceedings of the 33rd Advances in Neural Information Processing Systems, The MIT Press, Vancouver, Canada, 2019.
[26] S.M. Richards, F. Berkenkamp, A. Krause, The Lyapunov neural network: adaptive stability certification for safe learning of dynamical systems, in: Conference on Robot Learning, PMLR, Zürich, Switzerland, 2018.
[27] C. Dawson, Z. Qin, S. Gao, C. Fan, Safe nonlinear control using robust neural Lyapunov-barrier functions, in: Conference on Robot Learning, PMLR, Auckland, New Zealand, 2022.
[28] M. Han, L. Zhang, J. Wang, W. Pan, Actor-critic reinforcement learning for control with stability guarantee, IEEE Robot. Autom. Lett. 5 (4) (2020) 6217–6224.
[29] M. Ghavamzadeh, S. Mannor, J. Pineau, A. Tamar, Bayesian Reinforcement Learning: A Survey, Foundations and Trends in Machine Learning, 2015.
[30] R.T. Akella, K. Azizzadenesheli, M. Ghavamzadeh, A. Anandkumar, Y. Yue, Deep Bayesian quadrature policy optimization, in: Proceedings of the 35th AAAI Conference on Artificial Intelligence, PMLR, Vancouver, Canada, 2021.
[31] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, Stockholm, Sweden, 2018.
[32] W. Fehse, Automated Rendezvous and Docking of Spacecraft, vol. 16, Cambridge University Press, 2003.
[33] Richard Zappulla, Hyeongjun Park, Josep Virgili-Llop, Marcello Romano, Real time autonomous spacecraft proximity maneuvers and docking using an adaptive artificial potential field approach, IEEE Trans. Control Syst. Technol. 27 (6) (2018) 2598–2605.
[34] R.I. Zappulla, H. Park, J. Virgili-Llop, M. Romano, Experiments on autonomous spacecraft rendezvous and docking using an adaptive artificial potential field approach, in: Proceedings of the 26th AAS/AIAA Space Flight Mechanics Meeting, American Astronautical Society, California, USA, 2016.
[35] Y. Engel, S. Mannor, R. Meir, Bayes meets Bellman: the Gaussian process approach to temporal difference learning, in: Proceedings of the 20th International Conference on Machine Learning, PMLR, Washington, USA, 2003.
[36] Y. Engel, S. Mannor, R. Meir, Reinforcement learning with Gaussian processes, in: Proceedings of the 22nd International Conference on Machine Learning, PMLR, Bonn, Germany, 2005.
[37] A. O'Hagan, Bayes–Hermite quadrature, J. Stat. Plan. Inference 29 (3) (1991) 245–260.
[38] M. Ghavamzadeh, Y. Engel, Bayesian actor-critic algorithms, in: Proceedings of the 24th International Conference on Machine Learning, PMLR, Corvallis, USA, 2007.
[39] R.S. Sutton, D. McAllester, S. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Proceedings of the 13rd Advances in Neural Information Processing Systems, The MIT Press, Colorado, USA, 1999.
[40] A.G. Wilson, Z. Hu, R. Salakhutdinov, E.P. Xing, Deep kernel learning, in: Artificial Intelligence and Statistics, PMLR, 2016, pp. 370–378.
[41] H.J. Kushner, Stochastic Stability and Control, vol. 33, Academic Press, New York, 1967.
[42] C. Agrell, Gaussian processes with linear operator inequality constraints, J. Mach. Learn. Res. (2019).
[43] L.P. Swiler, M. Gulian, A.L. Frankel, C. Safta, J.D. Jakeman, A survey of constrained Gaussian process regression: approaches and implementation challenges, J. Mach. Learn. Model. Comput. 1 (2) (2020).
[44] J. Gardner, G. Pleiss, K.Q. Weinberger, D. Bindel, A.G. Wilson, Gpytorch: blackbox matrix-matrix Gaussian process inference with gpu acceleration, in: Proceedings of the 32nd Advances in Neural Information Processing Systems, The MIT Press, Montréal, Canada, 2018.