Policy Gradient-based Model Free Optimal LQG Control with a Probabilistic Risk Constraint

Arunava Naha and Subhrakanti Dey

***Abstract*— In this paper, we investigate a model-free optimal control design that minimizes an infinite horizon average expected quadratic cost of states and control actions subject to a probabilistic risk or chance constraint using input-output data. In particular, we consider linear time-invariant systems and design an optimal controller within the class of linear state feedback control. Three different policy gradient (PG) based algorithms, natural policy gradient (NPG), Gauss-Newton policy gradient (GNPG), and deep deterministic policy gradient (DDPG), are developed, and compared with the optimal risk-neutral linear-quadratic regulator (LQR) and a scenario-based model predictive control (MPC) technique via numerical simulations. The convergence properties and the accuracy of all the algorithms are compared numerically. We also establish analytical convergence properties of the NPG and GNPG algorithms under the known model scenario, while the proof of convergence for the unknown model scenario is part of our ongoing work.**

I. Introduction

The linear quadratic regulator (LQR) problem has been extensively studied in the literature, and the optimal controller is known to be a linear function of states [1]. However, the LQR formulation is risk neutral, *i.e.*, it does not consider the risk or chance of occurrence of undesirable events. Such risky or undesirable events may occur due to the long tail of the process noise or disturbance. For a lot of practical control problems, it is desirable to avoid such risky events. For example, sometimes it is important for an unmanned aerial vehicle (UAV) to avoid flying over a certain area to hide from adversaries. Therefore, it is crucial to design a controller that minimizes the risk of such events along with minimizing the average expected control cost [2]. Consider the wind turbine control problem, where wind speed introduces uncertainty, and the control aim is to optimize power output while mitigating structural damage risk [3]. Similarly, in climate-controlled buildings, the objective is to minimize energy usage while ensuring occupant comfort and mitigating the risk of temperature exceeding set thresholds [4]. As studied in [4], controllers designed with hard constraints are pessimistic compared to the ones designed with softer probabilistic constraints. In other words, the designed controller will lower the control cost if we constrain the probability of risky or undesirable events instead of imposing hard constraints. For example, in the case of the wind turbine control problem, the system will produce more power if we constrain the probability of the stress level on the blades increasing a prespecified limit instead of imposing a hard constraint on the stress level.

A. Related Work

In the model predictive control (MPC) literature, a probabilistic risk is generally modeled as a chance constraint. A popular approach is to draw samples or scenarios from the distribution of the disturbance and convert the probabilistic chance constraint into an algebraic one [4], [3]. The probabilistic chance constraint is also handled by replacing it with an expected value using the Hamiltonian Monte Carlo (HMC) method or employing Chebyshev's inequality. In a different approach, the chance constraints are formulated as the probability that the state and input values remain within certain sets, also called tubes [5]. Such tube-based chance constraint control is also studied under the model-free scenario in [6]. Other than the MPC-based approaches, the occurrence of risky or undesirable events is also limited by constraining the average variance over an infinite time horizon of a quadratic function of states in [7], [8]. The optimal controller under such risk formulation is proved to be an affine function of the states [9], [2].

Reinforcement learning (RL) based techniques have performed remarkably for optimal decision-making problems, where the underlying system is partially or entirely unknown [10], [11], [12]. Policy gradient (PG) based actor-critic (AC) methods, a class of RL algorithms, are suitable for stochastic optimal control problems with continuous state and action spaces [13]. PG-based algorithms are also applied for the standard LQR problem, and their performance in terms of closed-loop stability and convergence are studied in the literature [14], [15], [16]. It has been shown that although the optimization landscape is not convex with respect to the linear control gain in these problems, global convergence can be guaranteed when the PG algorithm is initialized with a stabilizing controller. The closed-loop stability and convergence analysis of PG-based algorithms for the LQR problem with additional constraints is challenging and has only been studied for a few specific cases. For example, in [17], the constraint is H_∞ robustness constraint. The global convergence of a PG algorithm is studied for the risk-constrained LQR in [18], where the risk is modeled as an average variance of a quadratic function of states over an infinite time horizon. On the other hand, in [19], the safety constraint is modeled as the expected value of a continuous non-negative function of the states being within a specified threshold, and the optimal controller is derived using an AC algorithm for a Markov decision process (MDP). In [20], a deep deterministic policy gradient (DDPG) based AC method is studied for the probabilistic risk-constrained LQR problem, where the risk is modeled as the probability that a quadratic function of the states crosses a user-defined limit. However, analyzing the performance of PG-based AC algorithms for probabilistic risk-constrained LQR problems in general remains an open research area.

B. Our Approach and Contributions

We have studied the performance of three PG-based AC algorithms, natural policy gradient (NPG) [21], [22], Gauss-Newton policy gradient (GNPG) [14], and deep deterministic

*This work is supported by The Swedish Research Council under grants 2017-04053.

Arunava Naha and Subhrakanti Dey are with the Department of Electrical Engineering, Uppsala University, 75103 Uppsala, Sweden. e-mail: arunava.naha@angstrom.uu.se and subhrakanti.dey@angstrom.uu.se

policy gradient (DDPG) [23], for the probabilistic risk- or chance-constraint LQR problem under the unknown model scenario. We have investigated the optimal policy within the class of linear state feedback controls. Our approach to handling the probabilistic risk constraint differs from the existing methods. We have modeled the risk as the probability that a function of the one step ahead future state crosses a user-defined limit, and the risk is constrained by keeping the average expected violation probability over an infinite time horizon within a user-defined limit. We have used the indicator function in the reward structure to replace the probability when the system model is unknown. Our probabilistic constraint model is more intuitive and direct compared to some of the existing studies.

We have used a Lagrangian based primal dual formulation to handle the constraint and proved that there is no duality gap. Furthermore, we have proved that the optimization problem under study enjoys coercivity and gradient dominance properties, and the NPG and GNPG algorithms converge to the global optimum under the known model assumption. To the best of our knowledge, such a convergence property is studied for the first time for the probabilistic risk or chance constrained LQR problem, even for the known model scenario. The coercivity and L-smoothness properties also ensure that intermediate policies will maintain closed-loop stability while training, provided we start from an arbitrary stabilizing controller. The theoretical study on the convergence of the DDPG algorithm and the convergence properties of the NPG and GNPG algorithms under the unknown model scenario (where sample based estimates of relevant quantities are used in the PG update) is left for our future publication.

We evaluate PG-based AC policies against risk-neutral LQR and scenario-based MPC through simulations. As anticipated, PG-based AC policies effectively reduce risky events, albeit with a slight increase in quadratic cost compared to standard LQR. While MPC performs comparably to PG-based methods, its effectiveness depends heavily on the time horizon length chosen, increasing computational complexity. Unlike MPC, model-free PG-based methods do not require real-time optimization at each step, relying solely on a feedforward actor-network post-training, significantly reducing computational overhead compared to MPC-based methods.

C. Organization

The rest of the paper is organized as follows. In Section II, we present the problem formulation and the reward structure for the probabilistic risk or chance constrained LQR problem. In Section III, we present the NPG, GNPG, and DDPG algorithms, while the convergence properties of NPG and GNPG are studied in Section IV. In Section V, we present the numerical results and compare the performance of the PG-based AC algorithms with the risk-neutral LQR and the scenario-based MPC. Finally, we conclude the paper in Section VI.

D. Notations

Some special notations are given in Table I.

TABLE I: Notations

Symbol	Description
$\mathbb{R}^n$	The set of $n \times 1$ real vectors
$\mathbb{R}^{m\times n}$	The set of $m \times n$ real matrices
X^T	Transpose of matrix or vector X
$\mathcal{N}(\mu, \mathbf{\Sigma})$	Gaussian distribution with mean μ and variance $\mathbf{\Sigma}$
$\mathbf{\Sigma} \geq \mathbf{0}$ or > 0	$\mathbf{\Sigma}$ is positive semi-definite or definite matrix, respectively
$[\cdot]_{ij}$	i-th row and j-th column element of a matrix
$\|\| \cdot \|\|$	Frobenius norm of a matrix of Euclidean norm of a vector
$\text{tr}(\cdot)$	Trace of a matrix
$\text{E}[\cdot]$ and $\text{P}\{\cdot\}$	Expectation operator and Probability measure respectively
$\{\hat{\cdot}\}$	Estimated or approximated value
$\mathbb{1}_{\{condition\}}$	Indicator function, 1 if condition is true, 0 otherwise
$\sigma(\cdot)$ and $\rho(\cdot)$	Singular value and eigenvalue of a matrix, respectively

II. Problem Formulation

We consider the following linear time-invariant (LTI) system:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k. \tag{1}$$

Here $\mathbf{x}_k \in \mathbb{R}^n$ and $\mathbf{u}_k \in \mathbb{R}^p$ are the state and input vectors at the k-th time instant respectively, whereas $\mathbf{w}_k \in \mathbb{R}^n$ is an independent and identically distributed (iid) process noise with distribution $f_w(\mathbf{w})$. $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} \in \mathbb{R}^{n\times p}$.

We assume that all the states are measured and the system $(\mathbf{A}, \mathbf{B})$ is stabilizable. In the standard LQR problem, the following cost function is minimized.

$$J = \lim_{T\to\infty} \frac{1}{T} \sum_{k=1}^{T} \text{E}\left[\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \mathbf{u}_k^T \mathbf{R} \mathbf{u}_k\right], \tag{2}$$

where $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and $\mathbf{R} \in \mathbb{R}^{p\times p}$ are positive definite weight matrices. We also assume that $(\mathbf{A}, \mathbf{Q}^{1/2})$ is detectable. If the noise is zero mean and the second-order moment of the noise is bounded, then the optimum input appears as a fixed gain linear control signal [1], see (3).

$$\mathbf{u}_k^* = \mathbf{K}\mathbf{x}_k, \text{ and } \mathbf{K} = -\left(\mathbf{B}^T\mathbf{S}\mathbf{B} + \mathbf{R}\right)^{-1}\mathbf{B}^T\mathbf{S}\mathbf{A}, \tag{3}$$

where $\mathbf{S}$ is the solution to the following algebraic Riccati equation, $\mathbf{S} = \mathbf{A}^T\mathbf{S}\mathbf{A} + \mathbf{Q} - \mathbf{A}^T\mathbf{S}\mathbf{B}\left(\mathbf{B}^T\mathbf{S}\mathbf{B} + \mathbf{R}\right)^{-1}\mathbf{B}^T\mathbf{S}\mathbf{A}$. However, as discussed before, the cost formulation (2) does not take into account the less frequent but risky events. Therefore, we use an additional constraint on the probability of risky or undesirable events, and the optimization problem takes the following form.

$$\begin{aligned} \underset{\mathbf{u}\in\mathcal{U}}{\text{minimize}} \quad & J \\ \text{subject to} \quad & J_c \leq \delta, \end{aligned} \tag{4}$$

where J is same as given in (2), and J_c is given as follows.

$$J_c = \lim_{T\to\infty} \frac{1}{T} \sum_{k=1}^{T} E\left[P\left\{f_c\left(\mathbf{x}_{k+1}\right) \geq \epsilon \mid \Psi_k\right\}\right]. \tag{5}$$

Here, $\epsilon > 0$ and $\delta > 0$ are user selected parameters. $\Psi_k \triangleq \{\mathbf{x}_l, \mathbf{u}_l \mid k \geq l \geq 0\}$ denotes the set of all information up to the instant k.

Remark 1: The risky or undesirable event is modeled as the function $f_c(\cdot)$ of the state at the next time step crossing a threshold ϵ, and we are interested in limiting the probability of these events. Since such probability itself is a function of the random information set Ψ_k, we have taken the expectation with respect to this set in the above formulation. Furthermore, we are interested in keeping the long-term average probability bounded over an infinite time horizon.

A. Reward Structure

The constrained optimization of (4) can be converted into an unconstrained stochastic control problem using the Lagrangian multiplier λ as follows,

$$\min_{\mathbf{u}\in\mathcal{U}} \mathcal{L} = J - \lambda\left(J_c - \delta\right) = \lim_{T\to\infty} \frac{1}{T}\sum_{k=1}^{T} \mathrm{E}\left[g\left(\mathbf{x}_k, \mathbf{u}_k\right)\right], \quad (6)$$

where the per stage cost $g(\cdot)$ takes the following form,

$$g\left(\mathbf{x}_k, \mathbf{u}_k\right) = f\left(\mathbf{x}_k, \mathbf{u}_k\right) + \lambda\left(h_p\left(\mathbf{x}_k, \mathbf{u}_k\right) - \delta\right), \text{ where} \quad (7)$$

$$f\left(\mathbf{x}_k, \mathbf{u}_k\right) = \mathbf{x}_k^T\mathbf{Q}\mathbf{x}_k + \mathbf{u}_k^T\mathbf{R}\mathbf{u}_k, \quad (8)$$

$$h_p\left(\mathbf{x}_k, \mathbf{u}_k\right) = P\left\{f_c\left(\mathbf{x}_{k+1}\right) \geq \epsilon \mid \Psi_k\right\}. \quad (9)$$

Note that the per-stage cost function may generally contain an intractable probabilistic constraint. Therefore, for the RL-based algorithms, where we have access to the future states ($\mathbf{x_{k+1}}$) in the form of stored data, the reward is defined as

$$r_k = -f\left(\mathbf{x}_k, \mathbf{u}_k\right) - \lambda\left(h_r\left(\mathbf{x}_{k+1}\right) - \delta\right), \text{ where} \quad (10)$$

$$h_r\left(\mathbf{x}_{k+1}\right) = \mathbb{1}_{\{f_c(\mathbf{x}_{k+1})\geq\epsilon\}}. \quad (11)$$

Here, $\mathbb{1}_{\{\cdot\}}$ is the indicator function, which takes the value 1 if the condition inside the bracket is true, and 0, otherwise. In the following section, we will briefly introduce the PG-based AC algorithms used in our study.

III. PG-based AC Algorithms under study

In this section, we will present NPG, GNPG, and DDPG algorithms, the three PG-based AC algorithms used in our study. For the NPG and GNPG algorithms, we assume the policy to be stochastic but stationary, denoted by $\mathbf{u}_k \sim \pi_\theta(\cdot|\mathbf{x_k})$, where θ is the policy parameter. The policy is deterministic for the DDPG algorithm, denoted by $\mathbf{u}_k = \mu_\theta(\mathbf{x_k})$, where θ is the policy parameter. We will use the general notation $p(\mathbf{x_k})$ to denote the stochastic or deterministic policy. In general, the policy parameter is updated using the gradient of the expected return, *i.e.*, $\mathcal{R}$, see (12), with respect to the policy parameter.

$$\mathcal{R} = \lim_{T\to\infty} \frac{1}{T}\mathrm{E}\left[\sum_{k=1}^{T} r_k\right]. \quad (12)$$

Furthermore, the value function (13), the Q function (14) and the advantage function (15) under a policy $p(\cdot)$ are defined as follows. We have used $(\dot{\cdot})$ notation to indicate an estimated or approximated quantity. Note that even though the reward in (10) is a function of the future state $\mathbf{x}_{k+1}$, for the known model case, we can write the reward as a function of the current state $\mathbf{x}_k$ and the control input $\mathbf{u}_k$ using (1).

$$V^p(\mathbf{x}_k) = \lim_{T\to\infty} \sum_{i=k}^{T} \left\{\mathrm{E}\left[r_i \mid \mathbf{x}_k\right] - \mathcal{R}\right\}, \quad (13)$$

$$Q^p(\mathbf{x}_k, \mathbf{u}_k) = \mathrm{E}\left[r_k + V^p(\mathbf{x}_{k+1}) \mid \mathbf{x}_k, \mathbf{u}_k\right]. \quad (14)$$

$$A^p(\mathbf{x}_k, \mathbf{u}_k) = Q^p(\mathbf{x}_k, \mathbf{u}_k) - V^p(\mathbf{x}_k). \quad (15)$$

A. Natural Policy Gradient (NPG) based AC algorithm

We have adopted the NPG-based AC algorithm from [21]; see Algorithm 1. NPG methods utilize the Fisher information matrix, F, to obtain the steepest ascent direction as $F^{-1}G$, where G is the gradient of the expected return, $\mathcal{R}$ with respect to the policy parameter θ. In practice, G is estimated using the policy gradient theorem [21] from the data as follows,

$$\hat{G} = \frac{1}{N}\sum_{k=1}^{N} \hat{A}(\mathbf{x}_k, \mathbf{u}_k)\nabla_\theta \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k). \quad (16)$$

Here, $\hat{A}(\cdot)$ is the estimated advantage function. Similarly, F is estimated as follows [21],

$$\hat{F} = \frac{1}{N}\sum_{k=1}^{N} \nabla_\theta \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)\nabla_\theta \log \pi_\theta(\mathbf{u}_k|\mathbf{x}_k)^T. \quad (17)$$

The step size for the policy parameter update is evaluated in the same way as [21], ensuring the policy parameter update is not too large, see Algorithm 1. Furthermore, the advantage value is estimated using the method provided in [24] as follows,

$$\begin{aligned} \hat{A}(\mathbf{x}_k, \mathbf{u}_k) &= \sum_{l=0}^{T-1} (\gamma\eta)^l d_{k+l}, \text{ where} \\ d_{k+l} &= -\hat{V}_\phi(\mathbf{x}_{k+l}) + r_{k+l} + \eta\hat{V}_\phi(\mathbf{x}_{k+l+1}). \end{aligned} \quad (18)$$

Here, $0 < \eta < 1$ and $0 < \gamma < 1$. $\hat{V}_\phi(\cdot)$ denotes the value obtained from the critic network parameterized by ϕ. The value function parameter ϕ is updated using the steepest descent direction as $\hat{H}^{-1}s$, where s and $\hat{H}$ are evaluated as

$$s = \nabla_\phi\left(\frac{1}{N}\sum_{k=1}^{N} \| \hat{V}_\phi(\mathbf{x}_k) - \hat{V}_k \|^2\right), \text{ and} \quad (19)$$

$$\hat{H} = \frac{1}{N}\sum_{k=1}^{N} \mathcal{J}_k\mathcal{J}_k^T, \text{ where } \mathcal{J}_k = \nabla_\phi\hat{V}_\phi(\mathbf{x}_k). \quad (20)$$

$$\text{The target value } \hat{V}_k \text{ is evaluated as } \hat{V}_k = \sum_{l=0}^{T-1} (\gamma)^l r_{k+l}. \quad (21)$$

Similar to the policy parameter update, the step size for the critic parameter update is also obtained in such a way that the update is not too large, see Algorithm 1.

Algorithm 1 NPG-based AC Algorithm

1: Initialize policy parameter θ_0 and the value function parameter ϕ_0.
2: Set γ, η, α_a, and α_c.
3: **for** $i = 1, 2, \ldots$ **do**
4: **for** $j = 1$ to M **do**
5: Generate a trajectory $\{\mathbf{x}_k, \mathbf{u}_k, r_k\}_{k=1}^{T}$ using the policy $\pi_{\theta_i}(\cdot|\mathbf{x_k})$.
6: Compute d_k using (18)
7: **for** $k = 1$ to N **do**
8: Compute $\hat{A}(\mathbf{x}_k, \mathbf{u}_k)$ (18) and $\nabla_\theta \log \pi_{\theta_i}(\mathbf{u}_k|\mathbf{x}_k)$.
9: Compute $\hat{V}_k$ (21) and $\mathcal{J}_k$ (20).
10: **end for**
11: Compute $\hat{G}_j$ and $\hat{F}$ using (16) and (17), respectively.
12: Compute $\hat{H}_j$ and s_j using (20) and (19), respectively.
13: **end for**
14: Compute $\hat{G} = \frac{1}{M}\sum_{j=1}^{M} \hat{G}_j$ and $\hat{F} = \frac{1}{M}\sum_{j=1}^{M} \hat{F}_j$.
15: Update the policy parameter $\theta_{i+1} \leftarrow \theta_i + \sqrt{\frac{\alpha_a}{\hat{G}^T\hat{F}^{-1}\hat{G}}}\hat{F}^{-1}\hat{G}$.
16: Compute $\hat{H} = \frac{1}{M}\sum_{j=1}^{M} \hat{H}_j$ and $\hat{s} = \frac{1}{M}\sum_{j=1}^{M} s_j$.
17: Update the value function parameter $\phi_{i+1} \leftarrow \phi_i + \sqrt{\frac{\alpha_c}{\hat{s}^T\hat{H}^{-1}\hat{s}}}\hat{H}^{-1}\hat{s}$.
18: **end for**

B. Gauss-Newton Policy Gradient (GNPG)

GNPG algorithm is a variant of the NPG algorithm, where the estimated Fisher information matrix $\hat{F}$ is replaced by the estimated Gauss-Newton matrix $\hat{H}_a$, see (22), otherwise all the other steps are the same as Algorithm 1, [21], [24].

$$\begin{aligned} \hat{H}_{a,j} &= \frac{1}{N}\sum_{k=1}^{N} g_{a,j} g_{a,j}^T, \text{ where} \\ g_{a,j} &= \hat{A}(\mathbf{x}_k, \mathbf{u}_k)\nabla \log \pi_{\theta_i}(\mathbf{u}_k|\mathbf{x}_k). \end{aligned} \tag{22}$$

C. Deep Deterministic Policy Gradient (DDPG)

The DDPG-based AC method is based on the Algorithm 2 [23], [20]. The actor and critic networks are parameterized by θ and ϕ, respectively. The actor takes states as input and outputs control inputs, while the critic takes states and control inputs, providing a Q value for that state-action pair. In DDPG, there are two separate target networks, Q^t and μ^t, for the critic and actor, respectively. These target networks facilitate stable learning by offering consistent targets and are updated gradually to track the main networks, as described in Algorithm 2.

Algorithm 2 DDPG-based AC Algorithm

1: Set τ, learning rates α_d, initial and final variances of zero mean Gaussian noise for exploration ($\mathcal{N}_t$), *i.e.*, $\Sigma_{D,0}$ and $\Sigma_{D,F}$.
2: Initialize θ_0 and ϕ_0 and $\theta_0^t \leftarrow \theta_0$ and $\phi_0^t \leftarrow \phi_0$.
3: Initialise the replay buffer $\mathcal{D}$.
4: **for** episode = 1, M **do**
5: Receive initial observation state $\mathbf{x}_1$.
6: **for** k = 1, T **do**
7: Select action $\mathbf{u}_k = \mu_\theta(\mathbf{x}_k) + \mathcal{N}_k$ [$\mathcal{N}_k$ is zero mean Gaussian noise].
8: Execute action $\mathbf{u}_k$ and observe reward r_k and observe new state $\mathbf{x}_{k+1}$.
9: Store transition $(\mathbf{x}_k, \mathbf{u}_k, r_k, \mathbf{x}_{k+1})$ in $\mathcal{D}$.
10: Sample a random minibatch of N transitions $(\mathbf{x}_i, \mathbf{u}_i, r_i, \mathbf{x}_{i+1})$ from $\mathcal{D}$.
11: Set $y_i = r_i + \gamma \hat{Q}^t_{\phi^t}\left(\mathbf{x}_{i+1}, \mu^t_{\theta^t}(\mathbf{x}_{i+1})\right)$.
12: Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i \left(y_i - \hat{Q}_\phi(\mathbf{x}_i, \mathbf{u}_i)\right)^2$.
13: Update the actor policy using the sampled policy gradient as
$\nabla_\theta \mathcal{R} \approx \frac{1}{N}\sum_i \nabla_u \hat{Q}_\theta(\mathbf{x}, \mathbf{u})\,|_{\mathbf{x}=\mathbf{x}_i, \mathbf{u}=\mu_\theta(\mathbf{x}_i)}\, \nabla_\theta \mu_\theta(\mathbf{x})\,|_{\mathbf{x}=\mathbf{x}_i}$
14: Update the target networks as
$\theta^t \leftarrow \tau\theta + (1-\tau)\theta^t$ and $\phi^t \leftarrow \tau\phi + (1-\tau)\phi^t$
15: **end for**
16: Reduce the variance of Gaussian noise for exploration until it reaches its final value
17: **end for**

In the following section, we present some analytical results on the convergence properties of the NPG and GNPG algorithms.

IV. Analytical Results

We investigate the two fundamental properties for the theoretical analysis of PG-based AC methods for the optimal controller design problem given by (4). The first property is the convergence of the AC algorithm to a local or global optimum policy and the corresponding convergence rate, while the second property concerns the closed-loop stability of the system during the training process. We investigate the convergence properties and stability aspects of the NPG and GNPG algorithms under the known model scenario and for the linear state feedback control. The study for more general cases, such as unknown model scenarios (where only sampled based estimates of the relevant quantities are available) and the convergence analysis of the DDPG algorithm for the probabilistic risk constrained control problem, is part of ongoing work.

We assume the policy to have the following form,

$$\pi_K(\cdot|\mathbf{x}) = \mathcal{N}\left(-\mathbf{K}\mathbf{x}, \Sigma_\sigma\right), \tag{23}$$

where $\mathcal{N}(\cdot,\cdot)$ denotes the Gaussian distribution. Furthermore, $\mathbf{K}$ is a trainable parameter, and Σ_σ is a fixed covariance matrix. Additionally, for theoretical analysis, we assume zero-mean Gaussian process noise, i.e., $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \Sigma_w)$. We anticipate that analogous theoretical outcomes can be derived for Gaussian mixture process noise, although that is a part of our ongoing research. Finally, the control input at k-th time instant can be written as,

$$\mathbf{u}_k = -\mathbf{K}\mathbf{x}_k + \sigma_k,\ \sigma_k \sim \mathcal{N}(0, \Sigma_\sigma). \tag{24}$$

We also define the set of all stabilizing linear state feedback controllers as $\mathcal{K} \triangleq \{\mathbf{K} \mid \rho(\mathbf{A} - \mathbf{B}\mathbf{K}) < 1\}$, where $\rho(\cdot)$ denotes the spectral radius. Under the policy (24), the closed-loop system dynamics can be written as,

$$\mathbf{x}_{k+1} = (\mathbf{A} - \mathbf{B}\mathbf{K})\,\mathbf{x}_k + \bar{\mathbf{w}}_k, \text{where} \tag{25}$$

$$\bar{\mathbf{w}}_k = \mathbf{w}_k + \mathbf{B}\sigma_k. \tag{26}$$

Here, $\bar{\mathbf{w}}_k \sim \mathcal{N}(0, \Sigma_{\bar{w}})$, where $\Sigma_{\bar{w}} = \Sigma_w + \mathbf{B}\Sigma_\sigma\mathbf{B}^T$, which can be derived directly using (1) and (24). Additionally, we assume the following function for the constraint,

$$f_c(\mathbf{x}_{k+1}) = \mathbf{q}^T\mathbf{x}_{k+1}, \tag{27}$$

where $\mathbf{q} \in \mathbb{R}^n$ is a user defined vector.

Before discussing our theoretical results, we rewrite the Lagrangian function from (6) using the control input given by (24) as follows.

$$\mathcal{L}(\mathbf{K}, \lambda) = J(\mathbf{K}) - \lambda\left(\mathbf{J_c}(\mathbf{K}) - \delta\right), \text{ where} \tag{28}$$

$$J(\mathbf{K}) = \text{tr}\left(\left(\mathbf{Q} + \mathbf{K^T}\mathbf{R}\mathbf{K}\right)\Sigma_K + \mathbf{R}\Sigma_\sigma\right) \tag{29}$$

$$= \text{tr}\left(\mathbf{P}_K\Sigma_{\bar{w}} + \mathbf{R}\Sigma_\sigma\right), \text{ and} \tag{30}$$

$$J_c(\mathbf{K}) = \text{E}\left[Q\left(a(\mathbf{x}_k, \mathbf{K})\right)\right], \text{ where} \tag{31}$$

$$Q(a) = \frac{1}{\sqrt{2\pi}}\int_a^\infty e^{-\frac{z^2}{2}}dz, \text{ and} \tag{32}$$

$$a(\mathbf{x}_k, \mathbf{K}) = \frac{\epsilon - \mathbf{q}^T(\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}_k}{\sqrt{\mathbf{q}^T\Sigma_{\bar{w}}\mathbf{q}}} \tag{33}$$

If $\mathbf{K} \in \mathcal{K}$, then Σ_K and $\mathbf{P}_K$ are the unique solutions to the Lyapunov equations given in (34) and (35), respectively.

$$\Sigma_K = \Sigma_{\bar{w}} + (\mathbf{A} - \mathbf{B}\mathbf{K})\Sigma_K(\mathbf{A} - \mathbf{B}\mathbf{K})^T, \text{ and} \tag{34}$$

$$\mathbf{P}_K = \mathbf{Q} + \mathbf{K}^T\mathbf{R}\mathbf{K} + (\mathbf{A} - \mathbf{B}\mathbf{K})^T\mathbf{P}_K(\mathbf{A} - \mathbf{B}\mathbf{K}) \tag{35}$$

Remark 2: The derivations of (29), (30), (34) and (35) are available in [16]. It is straightforward to derive (31), (32) and (33) using (25)-(27) in (5) and considering the states, $\{\mathbf{x_k}\}$ to be ergodic.

Our first result is the following lemma, which states the coercivity property of the Lagrangian function $\mathcal{L}(\mathbf{K}, \lambda)$ given by (28).

Lemma 1: For a fixed $\lambda > 0$, the Lagrangian function $\mathcal{L}(\mathbf{K}, \lambda)$ given by (28) is coercive on $\mathcal{K}$ in the sense that $\mathcal{L}(\mathbf{K}, \lambda) \to \infty$ as $\mathbf{K} \to \delta\mathcal{K}$, where $\delta\mathcal{K}$ denotes the boundary of $\mathcal{K}$.

Proof: The proof follows from the fact that the cost function $J(\cdot)$ is coercive on $\mathcal{K}$, see [15], and the constraint function $0 \leq J_c(\cdot) \leq 1$ is bounded. ■

Remark 3: The coercivity property of $\mathcal{L}(\mathbf{K}, \lambda)$ is crucial to ensure the stability of the closed-loop system during the training process. In other words, the coercive function $\mathcal{L}(\mathbf{K}, \lambda)$ serves as a barrier function over the stable policy set $\mathcal{K}$, and no additional measure is required to ensure the stability of the closed-loop system during the training process.

To demonstrate the convergence of the NPG and GNPG algorithms to a local or global optimum, it is imperative to establish the gradient dominance property of the Lagrangian function $\mathcal{L}(\mathbf{K}, \lambda)$ (28). Furthermore, the following two lemmas are required to support this property.

Lemma 2: For a given $\lambda > 0$, the Lagrangian function $\mathcal{L}(\mathbf{K}, \lambda)$ given by (28) is twice continuously differentiable over $\mathcal{K}$.

Proof: The cost function $J(\cdot)$ is twice continuously differentiable over $\mathcal{K}$, see [15]. Since the exponential function is analytic, $Q(a)$ is also analytic in a. Finally, $a(\mathbf{x}_k, \mathbf{K})$ is an affine function of $\mathbf{K}$, so we can say $J_c(\mathbf{K})$ is an analytic function of $\mathbf{K} \in \mathcal{K}$, and hence $\mathcal{L}(\mathbf{K}, \lambda)$ is at least twice continuously differentiable with respect to $\mathbf{K}$ over $\mathcal{K}$. ■

Lemma 3: For a given $\lambda > 0$, the Lagrangian function $\mathcal{L}(\mathbf{K}, \lambda)$ given by (28) is L-smooth on $\mathcal{K}_\zeta$, where $L > 0$ is a constant and depends on the problem parameters and ζ. Here, $\mathcal{K}_\zeta \triangleq \{\mathbf{K} \in \mathcal{K} \mid \mathcal{L}(\mathbf{K}, \lambda) \leq \zeta\}$ is a compact subset.

Proof: Using Lemma 1 and Lemma 2 in Theorem 1 from [15], we can directly state Lemma 3. ■

Remark 4: The L-smoothness of $\mathcal{L}(\mathbf{K}, \lambda)$ as stated in Lemma 3 also means

$$|| \nabla_K \mathcal{L}(\mathbf{K}, \lambda) ||^2 \leq L \ \forall \mathbf{K} \in \mathcal{K}_\zeta. \tag{36}$$

$|| \cdot ||$ denotes the Frobenius norm for a matrix or the Euclidean norm for a vector.

To establish a linear convergence rate for the NPG and GNPG algorithms, we need the L-smoothness and gradient dominance properties of the Lagrangian function $\mathcal{L}(\mathbf{K}, \lambda)$. The following lemma states the gradient dominance property of $\mathcal{L}(\mathbf{K}, \lambda)$.

Lemma 4: (Gradient dominance) For a given $\lambda > 0$, the Lagrangian function $\mathcal{L}(\mathbf{K}, \lambda)$ given by (28) satisfies the following inequality,

$$\mathcal{L}(\mathbf{K}, \lambda) - \mathcal{L}(\mathbf{K}^*, \lambda) \leq \mu \, || \nabla_K \mathcal{L}(\mathbf{K}, \lambda) ||^2 \, , \forall \mathbf{K} \in \mathcal{K}, \tag{37}$$

where $\mu > 0$ is a constant, and $\mathbf{K}^* \in \mathcal{K}$ is the optimal policy parameter for a fixed λ, *i.e.*, $\mathbf{K}^* = \arg\min_{\mathbf{K} \in \mathcal{K}} \mathcal{L}(\mathbf{K}, \lambda)$.

Proof: The proof of Lemma 4 is provided in Appendix II. ■

Finally, we state the convergence rate result for the NPG algorithm as follows, which is a direct consequence of Lemma 3 and Lemma 4. The convergence of the GNPG algorithm can be shown following similar steps and a detailed proof will be provided in our ongoing work.

Lemma 5: (Convergence rate) For a given $\lambda > 0$, the NPG algorithm converge to a global optimal policy parameter $\mathbf{K}^*$ with a linear convergence rate, *i.e.*,

$$\mathcal{L}(\mathbf{K}^{'} \lambda) - \mathcal{L}(\mathbf{K}^*, \lambda) \leq \beta(\mathcal{L}(\mathbf{K}, \lambda) - \mathcal{L}(\mathbf{K}^*, \lambda)) \tag{38}$$

Here $\mathbf{K}^{'}$ is the NPG update from $\mathbf{K}$ in a single iteration. The constant $0 < \beta < 1$ depends on the problem parameters and learning rate α.

Proof: The proof of Lemma 5 is provided in Appendix III. ■

Remark 5: The convergence rate lemma (38) means if we start from an arbitrary stabilizing controller $\mathbf{K}_0 \in \mathcal{K}$, and update the policy parameter $\mathbf{K}_i$ using the NPG algorithm, then $\mathbf{K}_i \to \mathbf{K}^*$ as $i \to \infty$ at an exponential rate.

In summary, so far, we have proved that the NPG algorithm converges linearly to a global optimal policy parameter $\mathbf{K}^*$ that minimizes $\mathcal{L}(\mathbf{K}, \lambda)$ for a given $\lambda > 0$ when the relevant quantities in the algorithms are computed assuming true model knowledge.

A. Finding the optimal value of the Lagrange multiplier λ

We follow a primal-dual approach to find an optimal value of the Lagrange multiplier λ. We first define the dual problem as follows,

$$\max_{\lambda \geq 0} D(\lambda) = \max_{\lambda \geq 0} \min_{\mathbf{K} \in \mathcal{K}} \mathcal{L}(\mathbf{K}, \lambda) \tag{39}$$

The primal-dual step is given as follows,

$$\text{Step 1: } \mathbf{K_{i+1}} = \text{One step update from } \mathbf{K_i} \text{ and } \lambda_\mathbf{i} \text{ using NPG or GNPG, see Algorithm 1, and} \tag{40}$$

$$\text{Step 2: } \lambda_{i+1} = \max\left(0, \lambda_i + \alpha_{\lambda,i} \left(J_c(\mathbf{K}_i) - \delta\right)\right). \tag{41}$$

Here, $\alpha_{\lambda,i} > 0$ is the learning rate for the Lagrange multiplier λ. To prove that the pair $(\mathbf{K}^*, \lambda^*)$ is also the optimal solution to the primal constrained problem (4), we need Assumption 1 and Lemma 6.

Assumption 1: (Slater's condition) There exists a $\bar{\mathbf{K}} \in \mathcal{K}$ such that $J_c(\bar{\mathbf{K}}) < \delta$.

Lemma 6: (Strong duality) Under Assumption 1, the optimal value of the primal problem (4) is equal to the optimal value of the dual problem (39), *i.e.*, $J^* = D^*$.

Proof: The proof of Lemma 6 is provided in Appendix IV. ■

Remark 6: The update rule of λ in (41) is guaranteed to converge to an optimal value as long as the step size $\alpha_{\lambda,i}$ is square summable but not absolutely summable; see [25]. A common choice for $\alpha_{\lambda,i}$ is $\alpha_{\lambda,i} = \frac{\kappa}{i}$, where $\kappa > 0$ is some finite constant.

Remark 7: The proof for the unknown model case is challenging since we approximate the value function and estimate the advantage and the required gradients from the input-output data. For the model-free scenario, in general, it is first proved that the value function approximator will converge and provide a close approximation of the true value of the state after a sufficiently large number of iterations, say $i \geq I_T$ [16]. Then it can be shown that $|\mathcal{L}(\mathbf{K}_{i+1}, \lambda) - \mathcal{L}(\tilde{\mathbf{K}}_{i+1}, \lambda)|$ is small for $i \geq I_T$, where $\mathcal{L}(\mathbf{K}_{i+1}, \lambda)$ and $\mathcal{L}(\tilde{\mathbf{K}}_{i+1}, \lambda)$ are the Lagrangian functions evaluated using the true and approximated value functions, respectively. Combining this with the convergence rate Lemma 5, we can show that the NPG and GNPG algorithms converge to a global optimal policy parameter $\mathbf{K}^*$ with a linear convergence rate for the unknown model case. However, detailed proofs for the unknown model case are part of our ongoing work.

V. Numerical Results

In this section, we compare the performance of the NPG, GNPG, and DDPG algorithms with the risk-neutral LQR and the scenario-based MPC by numerical simulations. As our case study, we have considered an unmanned aerial vehicle (UAV) model [18], a fourth-order LTI system. The UAV model parameters and the parameter values used for the simulation study are provided in Appendix I. All three PG-based algorithms use the same network structures for actor and critic networks. The actor is just a linear function of the state, and the critic is a fully connected neural network with two hidden layers of size (10,50). We have used the $\tanh$ activation function for the hidden nodes, and the outer layer has no activation function.

The scenario-based MPC algorithm used for the comparison is given in Algorithm 3 [3]. First, we compare the training

Algorithm 3 Scenario-based chance constraint MPC

Perform the following steps at each time step t:
Measure the current state $\mathbf{x}_t$.
Generate S noise samples $\mathbf{w}_t^{(1)}, \cdots, \mathbf{w}_t^{(S)} \sim f_w(\mathbf{w})$.
Solve the following optimization problem:

$$\min_{\mathbf{u}_{1|t},\cdots,\mathbf{u}_{T|t}} \sum_{s=1}^{S}\sum_{i=1}^{T}\left(f\left(\mathbf{x}_{i|t}^{(s)},\mathbf{u}_{i|t}\right)+\lambda \mathbb{1}_{\left\{f_c\left(\mathbf{x}_{i+1|t}^{(s)}\right)\geq\epsilon\right\}}\right)$$

s.t. (1) is satisfied.

Apply the first control input $\mathbf{u}_{1|t}$ to the system.

performance of the PG-based algorithms by plotting average return $\mathcal{R}$ with respect to the training timestep data from a trial run in Fig. 1. The average return is evaluated from the separate test data using the trained models after each 5E5 timestep training data sample. When starting from the same stable controller, we observe that NPG and GNPG algorithms maintain closed-loop stability while training. On the other hand, DDPG does not necessarily maintain closed-loop stability while training under similar conditions. However, in some instances DDPG is seen to reach the optimal policy faster. Figure 2 compares the constraint

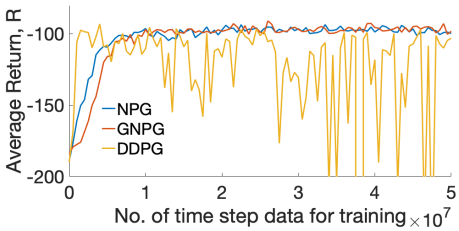

Fig. 1: Average return $\mathcal{R}$. $\lambda = 100$, $J_c = 8.7\%$ (NPG).

violation probability Jc and the control cost J for different λ values for all five algorithms. All the PG-based algorithms are trained over 5E7 timestep samples. Because of their stochastic nature, we have taken the average of the actor-network parameter of the best ten training iterations to generate the test results in Fig. 2.

As expected, the constraint violation percentage gets reduced for the PG-based methods at the expense of a small increase in the quadratic cost when compared with the standard LQR. Moreover, the performance of the MPC method has a similar

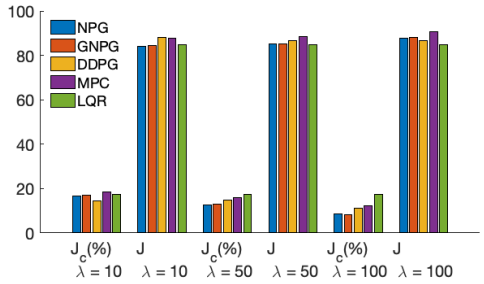

Fig. 2: Constraint violation probability J_c and control cost J for different λ values.

trend to that of PG-based algorithms. However, it is crucial to note that MPC's performance is heavily dependent on the chosen parameters $S = 20$ and $T = 5$ in Algorithm 3. While increasing these parameters can enhance MPC's performance, it comes with the trade-off of increased computational complexity, which is of the order of ST^2 [26]. In addition, MPC necessitates solving an optimization problem at every time step, in contrast to the PG-based methods, which only require evaluating the feed-forward actor-network. This distinction renders PG-based methods significantly less computationally complex than MPC. It is also important to acknowledge that MPC is a model-based method, which further differentiates it from the PG-based techniques. We also observed that the results from the DDPG algorithm do not show a clear trend as the other algorithms with the increase in λ. Because of this reason, we have excluded DDPG from the comparative plot in Fig. 3, where the constraint violation probability J_c is plotted with respect to the control cost J for different λ. We observe that NPG and GNPG algorithms perform almost similarly, outperforming the MPC for the given $S = 20$ and $T = 5$.

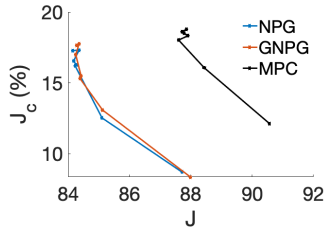

Fig. 3: Constraint violation probability J_c vs. control cost J. $\lambda = [1, 5, 10, 15, 20, 50, 100]$.

Figure 4 and Fig. 5 compare the norm of the policy gradient, *i.e.*, $||\hat{G}||$, and critic loss with respect to the number of training iterations for the NPG and GNPG algorithms. The plot shows the mean and 95% confidence interval of the quantity. We observe that NPG has a marginally better convergence rate compared to GNPG.

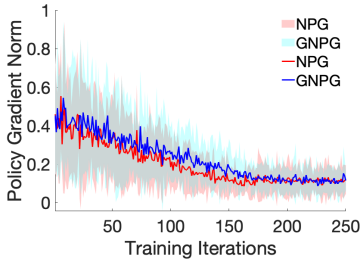

Fig. 4: Norm of the policy gradient. $\lambda = 10$, $J_c = 16.5\%$ (NPG).

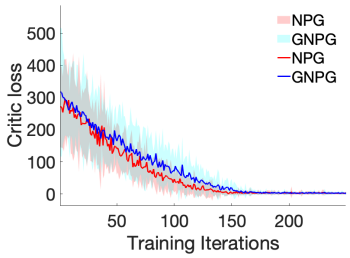

Fig. 5: Critic loss. $\lambda = 10$, $J_c = 16.5\%$ (NPG).

VI. Conclusion

We have considered PG-based AC algorithms for a probabilistic risk- or chance-constrained LQR problem under the unknown model scenario. The numerical simulations show that the NPG and GNPG-based AC methods exhibit good convergence properties and maintain closed-loop stability while training. On the other hand, DDPG has a larger variance, and the closed-loop system may not remain stable during training. Finally, we observe that all the PG-based algorithms perform similarly to the scenario-based MPC technique, which is a model-based approach and has certain computational disadvantages compared to the PG-based model-free approaches. Furthermore, we have proved analytical convergence properties of the NPG and GNPG algorithms under the known model scenario. The proof of convergence for the unknown model scenario is part of our ongoing work.

Appendix I
Parameters

$$\mathbf{A} = \begin{bmatrix} 1 & 0.5 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0.125 & 0 \\ 0.5 & 0 \\ 0 & 0.125 \\ 0 & 0.5 \end{bmatrix},$$

$\mathbf{W} = diag\left(1, 0.1, 2, 0.2\right), \mathbf{U} = \mathbf{I}, \epsilon = 5, \Sigma_w = diag(80, 0.01)$

$\mathbf{q} = [1, 0.1, 2, 0.2]^T, \alpha_c = 0.005, \alpha_a = 0.005, \alpha_d = 0.001,$

$\Sigma_{D,0} = 5\mathbf{I}, \Sigma_{D,F} = 0.01\mathbf{I}, \Sigma_u = \mathbf{I}.$

Appendix II
Proof of Lemma 4

From Lemma C.6 in [16], we can write

$$J(\mathbf{K}) - J(\bar{\mathbf{K}}^*) \leq \frac{||\Sigma_{\bar{K}^*}||}{\sigma_{\min}(\mathbf{R})} \text{tr}(\mathbf{E}_K^T \mathbf{E}_K), \tag{42}$$

where $\bar{\mathbf{K}}^*$ is the optimal policy parameter that miminizes only the cost function $J(\mathbf{K})$ and

$$\mathbf{E}_K = (\mathbf{R} + \mathbf{B}^T \mathbf{P}_K \mathbf{B})\mathbf{K} - \mathbf{B}^T \mathbf{P}_K \mathbf{A}. \tag{43}$$

$\mathbf{K}^*$ is the optimal policy parameter that minimizes the Lagrangian function $\mathcal{L}(\mathbf{K}, \lambda)$ for a given $\lambda > 0$. From (42), we can write

$$J(\mathbf{K}) - J(\mathbf{K}^*) \leq J(\mathbf{K}) - J(\bar{\mathbf{K}}^*) \leq \frac{||\Sigma_{\bar{K}^*}||}{\sigma_{\min}(\mathbf{R})} \text{tr}(\mathbf{E}_K^T \mathbf{E}_K).$$

$$\leq \frac{||\Sigma_{\bar{K}^*}||}{4\sigma_{\min}(\mathbf{R})} \text{tr}(4\boldsymbol{\Sigma}_{\mathbf{K}}{}^{-1} \boldsymbol{\Sigma}_{\mathbf{K}} \mathbf{E}_K^T \mathbf{E}_K \boldsymbol{\Sigma}_{\mathbf{K}}{}^{-1} \boldsymbol{\Sigma}_{\mathbf{K}}).$$

$$\leq \frac{n||\Sigma_{\bar{K}^*}||}{4\sigma_{\min}^2(\Sigma_K)\sigma_{\min}(\mathbf{R})} \,||\, \nabla_K J(\mathbf{K}) \,||^2 \leq \mu_1 \,||\, \nabla_K J(\mathbf{K}) \,||^2,$$

where $\mu_1 = \dfrac{n||\Sigma_{\bar{K}^*}||}{4\sigma_{\min}^2(\Sigma_K)\sigma_{\min}(\mathbf{R})}$. (44)

In (44), we have used the following results from [16],

$$\nabla_K J(\mathbf{K}) = 2\mathbf{E}_K \Sigma_K. \tag{45}$$

Taking derivative of (31) with respect to $\mathbf{K}$ we can write,

$$\nabla_K J_c(\mathbf{K}) = -\text{E}\left[\exp\left(-a(\mathbf{x}_k, \mathbf{K})^2/2\right) \frac{\mathbf{B}^T \mathbf{q} \mathbf{x}_k^T}{\sqrt{2\pi \mathbf{q}^T \Sigma_{\bar{w}} \mathbf{q}}}\right]. \tag{46}$$

Norm of the gradient of the Lagrangian function $\mathcal{L}(\mathbf{K}, \lambda)$ can be written as,

$$\begin{aligned}
&\text{tr}\left(\nabla_K \mathcal{L}(\mathbf{K}, \lambda)^T \nabla_K \mathcal{L}(\mathbf{K}, \lambda)\right) = \text{tr}\left(\nabla_K J(\mathbf{K})^T \nabla_K J(\mathbf{K})\right) \\
&+ \text{tr}\left(\lambda^2 \nabla_K J_c(\mathbf{K})^T \nabla_K J_c(\mathbf{K}) + 2\lambda \nabla_K J(\mathbf{K})^T \nabla_K J_c(\mathbf{K})\right). \\
&\geq \text{tr}\left(\nabla_K J(\mathbf{K})^T \nabla_K J(\mathbf{K})\right) + \text{tr}\left(2\lambda \nabla_K J(\mathbf{K})^T \nabla_K J_c(\mathbf{K})\right) \\
&\geq \text{tr}\left(\nabla_K J(\mathbf{K})^T \nabla_K J(\mathbf{K}) - 4\lambda \mathbf{E}_K \Sigma_K \text{E}\left[\frac{\mathbf{B}^T \mathbf{q} \mathbf{x}_k^T}{\sqrt{2\pi \mathbf{q}^T \Sigma_{\bar{w}} \mathbf{q}}}\right]\right) \\
&\text{[using (45) and (46), and } 0 \leq \exp\left(-a^2(\mathbf{x}_k, \mathbf{K})/2\right) \leq 1.] \\
&\geq \text{tr}\left(\nabla_K J(\mathbf{K})^T \nabla_K J(\mathbf{K})\right) \text{ [since E}[\mathbf{x}_k] = 0]. \qquad (47)
\end{aligned}$$

Combining (44) and (47), we can write

$$J(\mathbf{K}) - J(\mathbf{K}^*) \leq \mu_1 \text{tr}\left(\nabla_K \mathcal{L}(\mathbf{K}, \lambda)^T \nabla_K \mathcal{L}(\mathbf{K}, \lambda)\right). \tag{48}$$

Additionally, from Lemma C.6 [16], we can say $\nabla_K \mathcal{L}(\mathbf{K}, \lambda)^T \nabla_K \mathcal{L}(\mathbf{K}, \lambda)$ is lower bounded away from 0 by $\sigma(\Sigma_w)||\mathbf{R} + \mathbf{B}^T \mathbf{P}_K \mathbf{B}||^{-1} \text{tr}(\mathbf{E}_K^T \mathbf{E}_K)$. Since $|\, J_c(\mathbf{K}) - J_c(\mathbf{K}^*) \,| \leq 1$, there will exist a sufficiently large $\mu \geq \mu_1$, such that (we note that μ is dependent on λ)

$$\begin{aligned}
&\mathcal{L}(\mathbf{K}, \lambda) - \mathcal{L}(\mathbf{K}^*, \lambda) = J(\mathbf{K}) - J(\mathbf{K}^*) + \lambda(J_c(\mathbf{K}) - J_c(\mathbf{K}^*)) \\
&\leq \mu \text{tr}\left(\nabla_K \mathcal{L}(\mathbf{K}, \lambda)^T \nabla_K \mathcal{L}(\mathbf{K}, \lambda)\right). \qquad (49)
\end{aligned}$$

This completes the proof of Lemma 4.

Appendix III
Proof of Lemma 5

Here, we will prove Lemma 5 for the NPG algorithm. The update rule for the policy parameter $\mathbf{K}$ under the NPG algorithm is given by [16],

$$\mathbf{K}' = \mathbf{K} - \alpha[\mathbf{F}]^{-1} \nabla_K \mathcal{L}(\mathbf{K}, \lambda) = \mathbf{K} - \alpha \nabla_K \mathcal{L}(\mathbf{K}, \lambda) \Sigma_K^{-1}, \tag{50}$$

where $\alpha > 0$ is the learning rate, and $\mathbf{F}$ is the Fisher information matrix. $[\mathbf{F}]_{(i,j)(i',j')} = \text{E}[\nabla_{K_{ij}} \log(\pi_K(\mathbf{u}|\mathbf{x})) \nabla_{K_{i'j'}} \log(\pi_K(\mathbf{u}|\mathbf{x}))^T]$. Note that the variant of NPG algorithm as given in Algorithm 1 is $\alpha = \sqrt{\frac{\alpha_a}{\nabla_K \mathcal{L}(\mathbf{K},\lambda)^T [F]^{-1} \nabla_K \mathcal{L}(\mathbf{K},\lambda)}}$, where $\alpha_a > 0$ is a user selected parameter [21]. The minor difference in the stepsize expression from Algorithm 1 is due to the fact that the unknown parameter θ in Algorithm 1 is a vector, but in the proof, $\mathbf{K}$ is a matrix.

From the L-smoothness property of $\mathcal{L}(\mathbf{K}, \lambda)$ as given in Lemma 3, we can write the following inequality [15],

$$\begin{aligned}
&\mathcal{L}(\mathbf{K}', \lambda) - \mathcal{L}(\mathbf{K}, \lambda) \leq \\
&\text{tr}(\nabla_K \mathcal{L}(\mathbf{K}, \lambda)^T (\mathbf{K}' - \mathbf{K})) + \frac{L}{2} \,||\, \mathbf{K}' - \mathbf{K} \,||^2, \qquad (51)
\end{aligned}$$

Using (50) in (51), we can write

$$\begin{aligned}
&\mathcal{L}(\mathbf{K}', \lambda) - \mathcal{L}(\mathbf{K}, \lambda) \leq -\text{tr}\left(\alpha \Sigma_K{}^{-1} \right. \\
&\left. - \frac{L\alpha^2}{2} (\Sigma_K{}^{-1})(\Sigma_K{}^{-1})^T\right) \,||\, \nabla_K \mathcal{L}(\mathbf{K}, \lambda) \,||^2 . \qquad (52)
\end{aligned}$$

We have used the matrix trace inequality as given in Theorem 1 from [27] to get (52). To ensure convergence, we need the trace in (52)) to be strictly positive. In other words, the step size α should be $\alpha < \frac{2}{L\text{tr}(\Sigma_K^{-1})}$. Since Σ_K is the solution to the Lyapunov equation (34), we can say $\text{tr}(\Sigma_K)$ is upper bounded by a finite constant, so there exists a constant $0 < C_\Sigma < \text{tr}(\Sigma_K^{-1})$. Therefore, we can write an upper limit for $\alpha < \frac{2}{LC_\Sigma}$, which is independent of $\mathbf{K}$.

Applying the gradient dominance property of $\mathcal{L}(\mathbf{K}, \lambda)$ as given in Lemma 4, we can write

$$\mathcal{L}(\mathbf{K}^{'}, \lambda) - \mathcal{L}(\mathbf{K}^*, \lambda) \leq \beta(\mathcal{L}(\mathbf{K}, \lambda) - \mathcal{L}(\mathbf{K}^*, \lambda)), \text{ where} \quad (53)$$

$$\beta = 1 - \frac{1}{\mu}\text{tr}\left(\alpha {\Sigma_K}^{-1} - \frac{L\alpha^2}{2}({\Sigma_K}^{-1})({\Sigma_K}^{-1})^T\right) \quad (54)$$

Since we need $0 < \beta < 1$ for convergence, the step size α should satisfy the following condition

$$\begin{aligned} &0 < \frac{1}{\mu}\text{tr}\left(\alpha {\Sigma_K}^{-1} - \frac{L\alpha^2}{2}({\Sigma_K}^{-1})({\Sigma_K}^{-1})^T\right) < 1 \\ &=> \frac{1}{\mu}\text{tr}\left(\alpha {\Sigma_K}^{-1} - \frac{L\alpha^2}{2}({\Sigma_K}^{-1})({\Sigma_K}^{-1})^T\right) < 1 \text{ [if } \alpha < \frac{2}{LC_\Sigma}\text{]} \\ &=> \frac{1}{\mu}\text{tr}\left(\frac{\alpha}{\sigma_{\min}(\Sigma_{\bar{w}})} - \frac{L\alpha^2}{2}C_\Sigma^2\right) < 1, \text{ [(34) used]} \end{aligned} \quad (55)$$

Here, $\sigma_{\min}(\cdot)$ denotes the lowest singular value. Note that if we choose α sufficiently small, condition (55) can be satisfied. This completes the proof of Lemma 5.

APPENDIX IV
PROOF OF LEMMA 6

We follow the proof of Theorem 2 from [18]. The proof contains two steps.

First, it is proved that there exists a $\lambda^* \triangleq \inf\{\lambda \geq 0 | J_c(\mathbf{K}^*(\lambda)) \leq \delta\}$ such that $\lambda^* < \infty$. Although the constraint function differs in our case, we can utilize the same proof methodology as presented in [18], which relies on a contradiction argument employing Slater's condition. This proof does not rely on any specific formulation of the constraint function.

For the second step of the proof, we need to show that $\mathbf{K}^*(\lambda)$ and $J_c(\mathbf{K}^*(\lambda))$ are continuous functions of λ. We will prove this step in the following. We can directly say the gradient of the Lagrangian function $\mathcal{L}(\mathbf{K}, \lambda)$ with respect to $\mathbf{K}$ is a linear function of λ for a fixed $\mathbf{K}$. Additionally, $\nabla_K \mathcal{L}(\mathbf{K}, \lambda)$ is continuous in $K \in \mathcal{K}$, see Lemma 2. Therefore, the policy gradient steps, see Algorithm 1, will produce $\mathbf{K_i}$ that are continuous functions of λ. Finally, we have already proved that $\mathbf{K_i} \to \mathbf{K}^*$ as $i \to \infty$ in Lemma 4. Therefore, we can say the optimal policy parameter $\mathbf{K}^*(\lambda)$ and the constraint function $J_c(\mathbf{K}^*(\lambda))$ are continuous functions of λ. This completes the proof of Lemma 6.

REFERENCES

[1] D. P. Bertsekas, *Dynamic Programming and Optimal Control 3rd Edition, Volume II*. Athena Scientific, 2011.

[2] A. Tsiamis, D. S. Kalogerias, L. F. O. Chamon, A. Ribeiro, and G. J. Pappas, "Risk-Constrained Linear-Quadratic Regulators," in *2020 59th IEEE Conference on Decision and Control (CDC)*, Dec. 2020, pp. 3040–3047.

[3] G. Schildbach, L. Fagiano, C. Frei, and M. Morari, "The scenario approach for Stochastic Model Predictive Control with bounds on closed-loop constraint violations," *Automatica*, vol. 50, no. 12, pp. 3009–3018, Dec. 2014.

[4] J. Fleming and M. Cannon, "Stochastic MPC for Additive and Multiplicative Uncertainty Using Sample Approximations," *IEEE Trans. Automat. Contr.*, vol. 64, no. 9, pp. 3883–3888, Sep. 2019.

[5] E. Arcari, A. Iannelli, A. Carron, and M. N. Zeilinger, "Stochastic MPC with robustness to bounded parametric uncertainty," *IEEE Transactions on Automatic Control*, pp. 1–14, 2023.

[6] S. Kerz, J. Teutsch, T. Brüdigam, M. Leibold, and D. Wollherr, "Data-Driven Tube-Based Stochastic Predictive Control," *IEEE Open Journal of Control Systems*, vol. 2, pp. 185–199, 2023.

[7] F. Zhao, X. Fu, and K. You, "Global Convergence of Policy Gradient Methods for Output Feedback Linear Quadratic Control," *arXiv preprint arXiv:2211.04051*, 2022.

[8] F. Zhao and K. You, "Primal-dual learning for the model-free risk-constrained linear quadratic regulator," in *Learning for Dynamics and Control*. PMLR, 2021, pp. 702–714.

[9] F. Zhao, K. You, and T. Basar, "Infinite-horizon Risk-constrained Linear Quadratic Regulator with Average Cost," in *2021 60th IEEE Conference on Decision and Control (CDC)*. Austin, TX, USA: IEEE, Dec. 2021, pp. 390–395.

[10] D. Bertsekas, *Reinforcement Learning and Optimal Control*. Athena Scientific, Jul. 2019.

[11] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annual Reviews in Control*, vol. 46, pp. 8–28, Jan. 2018.

[12] V. G. Lopez, M. Alsalti, and M. A. Müller, "Efficient Off-Policy Q-Learning for Data-Based Discrete-Time LQR Problems," *IEEE Transactions on Automatic Control*, pp. 1–12, 2023.

[13] R. S. Sutton and A. G. Barto, *Reinforcement Learning, Second Edition: An Introduction*. MIT Press, Nov. 2018.

[14] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi, "Global convergence of policy gradient methods for the linear quadratic regulator," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1467–1476.

[15] B. Hu, K. Zhang, N. Li, M. Mesbahi, M. Fazel, and T. Başar, "Toward a Theoretical Foundation of Policy Optimization for Learning Control Policies," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 6, no. 1, pp. 123–158, 2023.

[16] Z. Yang, Y. Chen, M. Hong, and Z. Wang, "Provably Global Convergence of Actor-Critic: A Case for Linear Quadratic Regulator with Ergodic Cost," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[17] K. Zhang, B. Hu, and T. Başar, "Policy optimization for H_2 linear control with H_∞ robustness guarantee: Implicit regularization and global convergence," *SIAM J. Control Optim.*, vol. 59, no. 6, pp. 4081–4109, Jan. 2021.

[18] F. Zhao, K. You, and T. Başar, "Global Convergence of Policy Gradient Primal–Dual Methods for Risk-Constrained LQRs," *IEEE Trans. Automat. Contr.*, vol. 68, no. 5, pp. 2934–2949, May 2023.

[19] M. Han, Y. Tian, L. Zhang, J. Wang, and W. Pan, "Reinforcement learning control of constrained dynamic systems with uniformly ultimate boundedness stability guarantee," *Automatica*, vol. 129, p. 109689, Jul. 2021.

[20] A. Naha and S. Dey, "Reinforcement learning based optimal control with a probabilistic risk constraint," *arXiv preprint arXiv:2305.15755*, 2023.

[21] A. Rajeswaran, K. Lowrey, E. V. Todorov, and S. M. Kakade, "Towards Generalization and Simplicity in Continuous Control," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[22] S. M. Kakade, "A natural policy gradient," *Advances in neural information processing systems*, vol. 14, 2001.

[23] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning, ICLR (2016)," *arXiv preprint arXiv:1509.0297*, 2016.

[24] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-Dimensional Continuous Control Using Generalized Advantage Estimation," Oct. 2018.

[25] W. Yu and R. Lui, "Dual methods for nonconvex spectrum optimization of multicarrier systems," *IEEE Transactions on communications*, vol. 54, no. 7, pp. 1310–1322, 2006.

[26] J. Skaf and S. Boyd, "Nonlinear q-design for convex stochastic control," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2426–2430, 2009.

[27] I. Coope, "On matrix trace inequalities and related topics for products of hermitian matrices," *Journal of mathematical analysis and applications*, vol. 188, no. 3, pp. 999–1001, 1994.