Model-Free μ Synthesis via Adversarial Reinforcement Learning

Darioush Keivan*[1], Aaron Havens*[2], Peter Seiler[3], Geir Dullerud[1], and Bin Hu[2]

***Abstract*— Motivated by the recent empirical success of policy-based reinforcement learning (RL), there has been a research trend studying the performance of policy-based RL methods on standard control benchmark problems. In this paper, we examine the effectiveness of policy-based RL methods on an important robust control problem, namely μ synthesis. We build a connection between robust adversarial RL and μ synthesis, and develop a model-free version of the well-known DK-iteration for solving state-feedback μ synthesis with static D-scaling. In the proposed algorithm, the K step mimics the classical central path algorithm via incorporating a recently-developed double-loop adversarial RL method as a subroutine, and the D step is based on model-free finite difference approximation. Extensive numerical study is also presented to demonstrate the utility of our proposed model-free algorithm. Our study sheds new light on the connections between adversarial RL and robust control.**

I. INTRODUCTION

Recently, policy-based reinforcement learning (RL) [1], [2] has achieved impressive performance on various control tasks [3], [4]. Despite the empirical successes, how to choose and tune policy-based RL methods for a specific control problem at hand is not fully understood [5]. This inspires an increasing interest in understanding the performance of policy-based RL algorithms on simplified linear control benchmarks. For standard linear quadratic control problems, policy-based RL methods have been proved to yield strong convergence guarantees in various settings [6]–[15]. For robust/risk-sensitive control problems, the robust adversarial reinforcement learning (RARL) framework appears to be quite relevant. An important issue for deploying RL into real-world applications is the simulation-to-real gap. Originally RARL was developed to account for this gap by jointly training a protagonist and an adversary, where the protagonist learns to robustly perform the control tasks under the possible disturbances generated by its adversary [16], [17]. Recently, the connections between policy-based RARL and robust/risk-sensitive control have been formally studied, and policy-based RARL methods relying on double-loop update rules have been developed to solve the $\mathcal{H}_2/\mathcal{H}_\infty$ mixed design problem and the Linear Exponential Quadratic Gaussian problem in a provable manner [18]–[20].

An important robust control problem whose connection with policy-based RARL has been overlooked in the past is μ-synthesis whose objective is to design a controller optimizing the so-called structured singular value (or equivalently the robust performance) [21]. Over the past decade, μ synthesis has found numerous applications in industry, e.g. for robust control of hard disk drives for cloud storage [22]. Reexamining the performance of RARL on μ synthesis is an important task which can lead to valuable insights regarding the connections between RL and robust control.

In this paper, we bridge the gap between policy-based RARL and state-feedback μ-synthesis with static D-scaling. We build upon the double-loop RARL algorithm in [18] to develop a model-free policy optimization method for solving the state-feedback μ synthesis problem. Our proposed algorithm can be viewed as a model-free version of the well-known DK-iteration. In our algorithm, the K-step is a policy-based model-free variant of the well-established central path algorithm, and relies on the use of the double-loop RARL algorithm as the main subroutine. The D-step is based on model-free finite difference approximation. Similar to DK-iteration, our proposed method alternates between the K and D steps. When the D scaling is fixed, state-feedback μ synthesis reduces to $\mathcal{H}_\infty$ state-feedback design, and our algorithm can also be directly applied. The effectiveness of the proposed RARL approach on model-free μ synthesis are demonstrated via an extensive numerical study. Our paper complements existing work on data-driven robust control [23]–[25] by establishing a a new connection between model-free adversarial RL and μ-synthesis. Our paper also bring new insights for understanding robust RL in general.

II. PROBLEM FORMULATION AND PRELIMINARIES

A. Problem Statement

In this section, we formulate the model-free, state-feedback μ synthesis problem and clarify the "black-box" simulator needed in such a data-driven setting. To motivate our formulation, consider a discrete-time robust synthesis problem as shown in Figure 1[1]. The linear time-invariant (LTI) system G is governed by the following discrete-time state-space model:

$$\begin{aligned} x_{k+1} &= A\,x_k + B_w\,w_k + B_d\,d_k + B_u\,u_k \\ v_k &= C_v\,x_k + D_{uv}\,u_k \\ e_k &= C_e\,x_k + D_{ue}\,u_k \end{aligned} \tag{1}$$

We assume that the state of G can be directly measured and a static state-feedback controller is used, i.e. $u_k = -Kx_k$.

*Equal Contribution

[1]Darioush Keivan and Geir Dullerud are with the Coordinated Science Laboratory (CSL) and the Department of Mechanical Science & Engineering, University of Illinois at Urbana-Champaign, {dk12, dullerud}@illinois.edu

[2]Aaron Havens and Bin Hu are with the Coordinated Science Laboratory (CSL) and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, {ahavens2, binhu7}@illinois.edu

[3]Peter Seiler is with the Department of Electrical Engineering and Computer Science, University of Michigan, pseiler@umich.edu

[1]To be consistent with the current RL literature, we set $u_k = -Kx_k$.

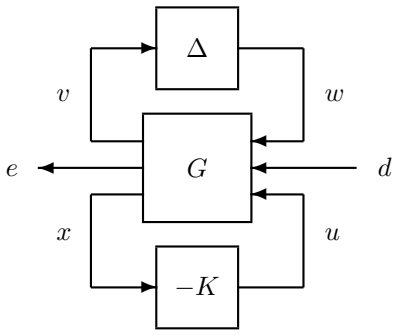

Fig. 1. Interconnection for Robust Synthesis

In this paper, we assume that all disturbance feedthrough terms are zero, however this assumptions may be possible to relax via a computationally heavy transformation. For convenient reference, we will use $\mathcal{F}_l(G,K)$ to denote the feedback interconnection of G and K. Thus, $\mathcal{F}_l(G,K)$ is a mapping from the input (w,d) to the output (v,e).

The pair (v,w) satisfies $w=\Delta v$ where Δ is a mapping in a *cone* $\mathbf{\Delta}$ of structured bounded linear operators. We call $\mathbf{\Delta}$ the uncertainty set. Details on this general interconnection for an uncertain feedback system can be found in [26], [27]. The closed-loop for a given state-feedback depends on Δ and is denoted $T_{d\mapsto e}(\Delta)$. We use $\|\cdot\|_\infty$ to denote the $\mathcal{H}_\infty$ norm. The state-feedback will be designed to optimize the robust performance of the closed-loop as defined next.

Definition 1: The controller K achieves *Robust Performance of level* γ if for all $\Delta\in\mathbf{\Delta}$ satisfying $\|\Delta\|_\infty\le\frac{1}{\gamma}$, the closed-loop is: (i) well-posed, (ii) stable, and (iii) has the mapping from d to e satisfying $\|T_{d\mapsto e}(\Delta)\|_\infty\le\gamma$. We define μ_K to be the infimum of all such γ.

Verifying robust performance is, in general, a fundamentally difficult non-convex problem, and accordingly so is computing μ_K. Hence one typically focuses on computing an upper bound. Specifically, define a set of scaling matrices $\mathbf{D}$ with the property that for each $D\in\mathbf{D}$ we have $D\Delta=\Delta D$ for all $\Delta\in\mathbf{\Delta}$. For a fixed controller K, an upper bound on μ_K is given by the following optimization:

$$\bar{\mu}_K=\inf_{D\in\mathbf{D}}\|\operatorname{diag}(D,I)\,\mathcal{F}_l(G,K)\,\operatorname{diag}(D^{-1},I)\|_\infty \quad (2)$$

This is the so-called D-scale upper bound on the robust performance metric [26]–[28]; when time-varying uncertainties are considered the set $\mathbf{D}$ contains only static matrices and $\bar{\mu}_K=\mu_K$. Although more general frequency-dependent scalings can be used for LTI uncertainties, our paper will focus on the static diagonal D-scaling case for simplicity.

Figure 2 shows a block diagram representation of the scaled system that appears in the robust performance upper bound (2). The goal for μ synthesis is to minimize the function μ_K over all stabilizing controllers K. We denote this optimal value of μ_K as $\mu^\star$. An approach to this problem is to work with the above upper bound, and related set of D-scales, to minimize the induced ℓ_2 gain from $(\tilde{w},d)$ to $(\tilde{v},e)$. Formally, the resulting synthesis is stated as the following

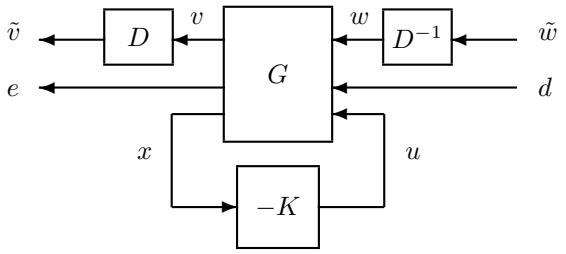

Fig. 2. Interconnection for DK Formulation

optimization problem:

$$\bar{\mu}^\star=\inf_{D\in\mathbf{D},K}\|\operatorname{diag}(D,I)\,\mathcal{F}_l(G,K)\,\operatorname{diag}(D^{-1},I)\|_\infty \quad (3)$$

Note that $\mu^\star=\bar{\mu}^\star$ for our specific problem when $\mathbf{\Delta}$ is the set of LTV uncertainty, but generally $\mu^\star\le\bar{\mu}^\star$. Since we consider the state-feedback with static D scaling, the problem can be reformulated as a convex program [29]. One issue is that this convex approach cannot be directly applied in the model-free setting. An alternative approach is the so-called DK-iteration which alternates between optimizing over D (with K fixed) and optimizing over K (with D fixed). While each step is convex, the alternation does not necessarily yield the global optimum for the joint optimization over D and K. However, such a heuristic approach can find good solutions in many practical scenario, and we will generalize this method to the model-free setting.

The focus of this paper is the model-free setting where all the state/input/output matrices in (1) are unknown. We only assume the availability of a "black-box" simulator for G. Notice that the nominal control design corresponds to $\Delta=0$. In this case, one only needs a simulator which is capable of generating the trajectories of $\{e_k\}$ given $w=0$ and any sequence $\{d_k\}$. However, solving the robust synthesis in a model-free manner requires a more powerful simulator. We assume that the simulator for (1) is able to generate the trajectories of $\{e_k,v_k,x_k\}$ for any given $\{d_k,w_k,u_k\}$. Notice that a black box simulator for the nominal model (with no uncertainty) can be modified to incorporate (w,v) channels corresponding to input multiplicative uncertainty. This is a standard uncertainty class that accounts for non-parametric error (unmodeled dynamics) at the plant input [26]–[28]. We will also demonstrate such simulator via the setting in Section IV. Then the goal of our paper is to use the above "black-box" simulator of the uncertain plant to solve the state-feedback μ synthesis problem (3) with static D-scaling in a model-free manner.

B. Model-Free Minimum-Entropy $\mathcal{H}_\infty$ Control via RARL

Our proposed model-free solution for state-feedback μ synthesis will rely on existing results on RARL for linear quadratic (LQ) games. Via the RARL framework, one can design robust policies against possible adversarial attacks by jointly training a protagonist and an adversary via a game formulation where the protagonist learns to robustly perform the control tasks under the possible disturbances generated by

its adversary. Here, we briefly review some relevant results on LQ RARL. Consider a two-player, zero-sum, LQ game:

$$J(u,h) := \mathbb{E}\sum_{k=0}^{\infty}\left\{x_k\top Q x_k + u_k^\top R^u u_k - h_k^\top R^h h_k\right\} \quad (4)$$
$$\text{subject to: } x_{k+1} = Ax_k + B_u u_k + B_h h_k,\ x_0 \sim \mathcal{D}$$

where Q, R^u, and R^h are positive definite matrices with compatible dimensions. The initial state x_0 is drawn from the distribution $\mathcal{D}$. In the RARL framework, the protagonist uses the "control action" u to minimize J while the adversary uses the "attack" h to maximize J. The expectation is taken over the trajectory $\{x_t\}$, and the only randomness stems from the random initial state satisfying $\mathbb{E}[x_0 x_0^\top] = \Sigma_0$. The goal for RARL is to solve the Nash equilibrium of the above game and obtain a pair of control-disturbance sequences $\{u_t^*\}$ and $\{h_t^*\}$ satisfying $J(u^*,h) \le J(u^*,h^*) \le J(u,h^*)$ for any u and h. It is known [30] that the Nash Equilibrium of the above LQ game can be attained by state-feedback controllers, i.e., there exists a pair of matrices (K^*, L^*), such that $u_t^* = -K^* x_t$ and $h_t^* = -L^* x_t$. Hence, it suffices to search over the *stabilizing* control gain matrices (K, L) (policy parameters). This leads to the following minimax problem where J becomes a function of (K, L):

$$\min_K \max_L J(K,L)$$
$$\text{subject to: } x_{t+1} = Ax_t + B_u u_t + B_h h_t,\ x_0 \sim \mathcal{D}$$
$$u_t = -Kx_t, h_t = -Lx_t$$

Therefore, one can apply various iterative gradient-based methods and their model-free counterparts to solve the above minimax problem. Based on [18], a double-loop algorithm can be used to guarantee convergence and stability. For our problem, the most important application of this double-loop RARL method is to provide a model-free solver for the so-called minimum-entropy $\mathcal{H}_\infty$ control problem [31].[2] For a fixed D, let $\mathcal{K}_\gamma$ denote the set of all stabilizing controllers satisfying the close-loop $\mathcal{H}_\infty$ bound γ, i.e. $\|\operatorname{diag}(D,I)\,\mathcal{F}_l(G,K)\,\operatorname{diag}((D^{-1},I)\|_\infty \le \gamma$. The minimum-entropy control aims at solving the "minimum entropy" center of $\mathcal{K}_\gamma$ for any given γ. It is well known that the minimum entropy controller can be solved via an equivalent game formulation. Therefore, we can modify the above double-loop RARL algorithm to obtain a model-free oracle **RARLSolver**(γ, K, D) which uses an internal iterative process initialized from K to generate the minimum-entropy center of $\mathcal{K}_\gamma$ for any given γ and D. The implementation details for **RARLSolver** are presented in our `arXiv` report [34].

C. Model-free $\mathcal{H}_\infty$ Evaluation via Power Iteration

Before proceeding to our proposed model-free method for solving the design problem (3), it is natural to ask whether there exists a model-free oracle for evaluating the value of the objective function $||\operatorname{diag}(D,I)\,\mathcal{F}_l(G,K)\,\operatorname{diag}(D^{-1},I)||_\infty$ given any K and D. The answer is yes. There are various methods available for the $\mathcal{H}_\infty$-norm estimation tasks [35]–[40]. One approach which is particularly suitable for our setting is the multi-input, multi-output (MIMO) power iteration method [41]. This relies on a specialized time-reversal method to estimate the $\mathcal{H}_\infty$ norm of an LTI MIMO system from the spectral radius of its finite-time approximated representation. Given a black-box simulator for a stable system $\tilde{G}$, the power iteration method provides an efficient oracle for estimating $||\tilde{G}||_\infty$ denoted as

$$\|\tilde{G}\|_\infty \approx \textbf{HinfOracle}(\tilde{G}, N) \quad (5)$$

where N is specified by the users. The **HinfOracle** uses the simulated input/output data of $\tilde{G}$ to query $\underline{\tilde{G}}_N$, which is an N-step finite-time approximation of $\tilde{G}$, and then outputs a number to estimate the following spectral radius

$$\bar{\sigma}(\underline{\tilde{G}}_N) = \sqrt{\lambda_{\max}\left(\underline{\tilde{G}}_N^\top \underline{\tilde{G}}_N\right)}.$$

The $\mathcal{H}_\infty$-norm of $\tilde{G}$ can be recovered as:

$$\|\tilde{G}\|_\infty = \lim_{N\to\infty} \bar{\sigma}\left(\underline{\tilde{G}}_N\right). \quad (6)$$

The key step in the **HinfOracle** is that time-reversal is used to access the adjoint system $\underline{\tilde{G}}_N^\top$ from the input/output data generated by the simulator of $\tilde{G}$. We refer the readers to [41] for implementation details of the power iteration method.

It should be noted that the **HinfOracle** will typically generate a *lower-bound* for the $\mathcal{H}_\infty$ norm of the original system. Some relevant theory can be found in [38]. For our purpose, a tight upper bound is desired, and we will discuss a potential fix in the next section.

III. MAIN ALGORITHM

As mentioned previously, in the case where the model G is known, the robust synthesis problem (3) is typically solved via a coordinate-descent-type method called DK-iteration. For any fixed (K, D), denote the objective function $\Gamma(K,D) := \|\operatorname{diag}(D,I)\,\mathcal{F}_l(G,K)\,\operatorname{diag}((D^{-1},I)\|_\infty$. Then DK-iteration follows the update rule:

$$K^{(n+1)} = \arg\min_K \Gamma\left(K, D^{(n)}\right) \quad (7)$$
$$D^{(n+1)} = \arg\min_D \Gamma\left(K^{(n+1)}, D\right) \quad (8)$$

where the initial $D^{(0)}$ is usually chosen as I. This approach alternates between the K-step (7) and D-step (8). When the model is known, both steps can be efficiently solved as convex programs. In the model-free case, our proposed algorithm can be viewed as a model-free counterpart of the DK-iteration method. Specifically, we will develop iterative model-free algorithms to solve both the K-step (7) and the D-step (8) in an approximate way.

[2]Depending on whether to include the "$-$" sign into the definition of the entropy, some papers adopt the terminology "maximum-entropy $\mathcal{H}_\infty$ control" to refer to the same problem [32], [33].

A. Overview

An overview summary of our proposed approach is given in Algorithm 1. Our model-free algorithm still delineates two main steps: 1) a K-step which performs $\mathcal{H}_\infty$ synthesis for a fixed scaling D, and 2) a D-step which optimizes μ over static scaling matrix D for a fixed K. The main difference is that the exact minimization (7) (8) are replaced with model-free approximation updates (9) (10).

- K-step: In contrast to solving (7) exactly, we call the oracle **Approx-Kmin** to obtain a model-free solution for the $\mathcal{H}_\infty$ synthesis with a fixed scaling $D^{(n)}$. The oracle **Approx-Kmin** runs an iterative method by itself and requires an initial policy which is not explicitly needed in the original exact minimization (7). At step n, We use the iterate $K^{(n)}$ to initialize the iterations in **Approx-Kmin** and the output of **Approx-Kmin** is used as $K^{(n+1)}$. On the conceptual level, the iterative algorithm within **Approx-Kmin** can be viewed as a model-free counterpart of the central path algorithm. The details for the model-free oracle **Approx-Kmin** are presented in Section III-B.
- D-step: Similarly, the exact optimization (8) is replaced with a model-free oracle **Approx-Dmin** that runs an iterative finite-difference optimization method to optimize D for a fixed K. At step n, the finite-difference optimization in **Approx-Dmin** is initialized with $D^{(n)}$ and will generate an output $D^{(n+1)}$. Section III-C gives details for the model-free oracle **Approx-Dmin**.

It is emphasized that both **Approx-Kmin** and **Approx-Dmin** only require the use of a "black-box" system G of the structure (1) which is able to generate the trajectories of $\{e_k, v_k\}$ given inputs $\{d_k, w_k\}$. Both oracles heavily rely on the model-free $\mathcal{H}_\infty$ estimator **HinfOracle** introduced in Section II-C as well as some model-free iterative optimization methods and hence we need to provide effective initialization when calling them. It is also worth mentioning that Algorithm 1 requires an initial nominally stabilizing controller $K^{(init)}$, which can also be obtained using standard policy-based RL methods [42]. We will now describe **Approx-Kmin** and **Approx-Dmin** in detail as well as some practical considerations important for implementation.

B. Model-free Approximation for K-step

Now we give details for how to solve the K-step in a model-free way. The pseudo code for **Approx-Kmin** is given in Algorithm 2. The goal is to perform model-free $\mathcal{H}_\infty$ synthesis for a fixed scaling D. An iterative algorithm is used. For clarity, we use $\tilde{K}_\tau$ to denote the internal controller iterations within **Approx-Kmin**. When used in the n-th iteration of the main algorithm 1, **Approx-Kmin** will initialize as $\tilde{K}_0 = K^{(n)}$ and generate the final output as $K^{(n+1)} = \tilde{K}_{\mathrm{T}}$, where T is the number of the iterations run within **Approx-Kmin**.

Next, we discuss the internal process for **Approx-Kmin**. At each iteration τ, **HinfOracle** is first called to compute the closed-loop $\mathcal{H}_\infty$ norm γ_τ for the associated controller $\tilde{K}_\tau$. Specifically, we want γ_τ to be a good estimate for $\| \operatorname{diag}(D, I)\, \mathcal{F}_l(G, \tilde{K}_\tau)\, \operatorname{diag}((D^{-1}, I)\|_\infty$. Let $\mathcal{K}_\gamma$ denote the set of all stabilizing controllers satisfying the close-loop $\mathcal{H}_\infty$ bound γ, i.e. $\Gamma(K, D) \leq \gamma$. In the robust control literature, $\mathcal{K}_\gamma$ is also termed as the "γ-admissible set." Obviously, $\tilde{K}_\tau$ is on the boundary of the set $\mathcal{K}_{\gamma_\tau}$. Intuitively, the center of $\mathcal{K}_{\gamma_\tau}$ should have a closed-loop $\mathcal{H}_\infty$ norm being smaller than γ_τ. If we move the iterations towards the center of $\mathcal{K}_{\gamma_\tau}$, we should be able to get a controller with a smaller closed-loop $\mathcal{H}_\infty$ norm and then improve the robust performance. This motivates our next step which is to call the RARL algorithm **RARLSolver** to approximately solve the "minimum-entropy" center for the γ_τ-admissible set. One technical subtlety is that running **RARLSolver**(γ, K) requires the initial point K to have a closed-loop $\mathcal{H}_\infty$ norm which is strictly smaller than γ. Therefore, we cannot run **RARLSolver**$(\gamma_\tau, \tilde{K}_\tau)$ directly. We need to slightly perturb γ_τ to enlarge the admissible set such that $\tilde{K}_\tau$ becomes an interior point that is good for the initialization purpose. Then we can call **RARLSolver**$(\gamma_\tau + \delta_\tau, \tilde{K}_\tau)$ to generate the central solution of the set $\mathcal{K}_{\gamma_\tau + \delta_\tau}$. The perturbation parameter δ_τ needs to be tuned in a case-by-case manner, and can

Algorithm 1 Model-free DK-iteration

Require: Simulator G

1: Input: Stabilizing $K^{(init)}$, and number of iterations N
2: **Initialize** $D^{(0)} := I$, $K^{(0)} := K^{(init)}$
3: **for** $n = 0, \cdots, N-1$ **do**
4: **Model-Free K-Step Section (III-B):** Call the oracle **Approx-Kmin**$(K^{(n)}, D^{(n)})$ to update K as:

$$K^{(n+1)} = \textbf{Approx-Kmin}(K^{(n)}, D^{(n)}) \tag{9}$$

5: **Model-Free D-Step Section (III-C):** Call the oracle **Approx-Dmin**$(K^{(n+1)}, D^{(n)})$ to update D as:

$$D^{(n+1)} = \textbf{Approx-Dmin}(K^{(n+1)}, D^{(n)}) \tag{10}$$

6: **end for**
7: **return** the final control design $K^{(N)}$

Algorithm 2 Model-Free Oracle **Approx-Kmin**(K, D)

1: Input: Initial K, number of iterations T, fixed scaling D, finite window length L, scalars $\{\delta_\tau\}_{\tau=0}^{\mathrm{T}-1}$
2: **Initialize** $\tilde{K}_0 := K$
3: **for** $\tau = 0, \cdots, \mathrm{T}-1$ **do**
4: Call **HinfOracle** to obtain an estimate for the $\mathcal{H}_\infty$ norm of the closed-loop system with controller $\tilde{K}_\tau$:

$$\gamma_\tau = \textbf{HinfOracle}(\operatorname{diag}(D, I)\, \mathcal{F}_l(G, \tilde{K}_\tau)\, \operatorname{diag}((D^{-1}, I), L)$$

5: Slightly perturb γ_τ and then call **RARLSolver** to solve the resultant mixed design problem:

$$\tilde{K}_{\tau+1} = \textbf{RARLSolver}(\gamma_\tau + \delta_\tau, \tilde{K}_\tau, D)$$

6: **end for**
7: Return the controller $\tilde{K}_{\mathrm{T}}$

be typically chosen as a small positive number as long as the estimate from **HinfNorm** is reasonable. Then the center of $\mathcal{K}_{\gamma_\tau+\delta_\tau}$ should be close to the center of $\mathcal{K}_{\gamma_\tau}$. Then we will just set the updated controller $\tilde{K}_{\tau+1}$ to be the "central" solution generated by **RARLSolver**. As τ increases, the set $\mathcal{K}_{\gamma_\tau}$ is expected to shrink. For sufficiently large T, the set $\mathcal{K}_{\gamma_{\mathrm{T}}}$ becomes sufficiently small, and the algorithm will return a controller $\tilde{K}_{\mathrm{T}}$ which approximates the solution for the original K-step minimization problem (7).

Connections with central path: There is a deep connection between the proposed method **Approx-Kmin** and the well-known central path algorithm for solving semi-definite programs (SDPs) [43], [44]. The central path algorithm computes the *analytic center* $p^*(\lambda)$ of a λ-cost sublevel set of the feasible LMI through a barrier function formulation. At each iteration, λ is updated as the cost achieved by the previous analytic center. Our proposed method is similar in the sense that at each iteration, $\tilde{K}_{\tau+1}$ is generated to approximate the center of the previous admissible set. Formally speaking, **RARLSolver** generates an approximate solution for the so-called minimum entropy $\mathcal{H}_\infty$ control problem, and this naturally leads to the "central solution" for the given $\mathcal{H}_\infty$-constrained set [32]. Therefore, on the conceptual level, **Approx-Kmin** can be viewed as an approximate version of the central path algorithm. It is well known that the central path algorithm yields strong convergence guarantees and is typically much faster than non-smooth methods based on the subgradient of the $\mathcal{H}_\infty$ norm. We suspect that **Approx-Kmin** will also yield strong convergence guarantees and leave such theoretical study as future work. Notice the entropy itself is not a barrier function for the $\mathcal{H}_\infty$-constrained set, and hence new theoretical arguments may be needed. Inspired by a variant of the central path algorithm, we can also apply a factor $\theta \in [0, 1)$ to interpolate $\lambda_{\tau+1} = (1-\theta)\gamma_{\tau+1} + \theta\lambda_\tau$. This interpolation further promotes feasibility of **RARLSolver**.

C. Model-Free Approximation for D-step

We now develop the model-free D-step procedure **Approx-Dmin** with pseudo code given in Algorithm 3. In the D-step, we fix the controller K and optimize (3) over static diagonal D scaling matrices via a subgradient method. Similarly to **Approx-Kmin**, we denote the internal D scale iterations by $\tilde{D}_\tau$ and their parameters by $\tilde{d}_\tau$. At the n-th iteration of our main Algorithm 1, we apply **Approx-Dmin**, with $\tilde{D}_0 := D^{(n)}$, to generate the update $D^{(n+1)} := \tilde{D}_{\mathrm{T}}$.

In the model-free setting, we can only query evaluations of the objective function (3) through our data-driven power method **HinfOracle**, so we employ the finite difference method to obtain subgradient estimates. The static diagonal D scaling is explicitly parameterized by $D := \exp(\mathrm{diag}(d))$, where $d = (d_1, \ldots, d_m)$ and m is the dimension of the diagonal uncertainty Δ. We define the function H as

$$H(d) := ||\,\mathrm{diag}(e^{\mathrm{diag}(d)}, I)\mathcal{F}_l(G, K)\,\mathrm{diag}(e^{-\mathrm{diag}(d)}, I)||_\infty \tag{11}$$

The *central difference* estimate of the jth entry of the gradient g is

$$g_j(d, \varepsilon) = \frac{H(d + \varepsilon e_j) - H(d - \varepsilon e_j)}{2\varepsilon}, \quad j \in [m] \tag{12}$$

where e_j is a vector who's jth entry is 1 and zero elsewhere and ε is some positive number sufficiently small. With the gradient g in hand, we can proceed to perform gradient descent updates on d. Using the black-box simulator and **HinfOracle** with a window of N to take evaluations of H, we have an approximate oracle for the finite difference gradient update on our internal parameter $\tilde{d}_\tau$:

$$\tilde{d}_{\tau+1} = \tilde{d}_\tau - \alpha g(\tilde{d}_\tau, \varepsilon) \approx \textbf{CentralDiffOracle}(\tilde{d}_\tau, \varepsilon, \alpha, N)$$

Where **CentralDiffOracle** is applied for a fixed number of iterations and the final scaling is given by $\tilde{D}_{\mathrm{T}} := \mathrm{diag}(\exp(\tilde{d}_\tau))$. Note that when using unreliable evaluations, ε must be chosen large enough to observe a valid descent direction, but also small enough to capture the local gradient information and avoid oscillations around a minimum.

Algorithm 3 Model-Free Oracle **Approx-Dmin**(K, D)

1: Input: Fixed K, initial parameterized scaling $D = \mathrm{diag}(\exp(d))$, number of iterations T, finite window length L, gradient step-size α, perturbation size ε.
2: **Initialize** $\tilde{d}_0 := d$
3: **for** $\tau = 0, \cdots, \mathrm{T}-1$ **do**
4: Use the **CentralDiffOracle** to estimate the central difference gradient update for the $\mathcal{H}_\infty$ norm of the closed-loop system with controller K and scaling elements $\tilde{d}_\tau$:
$$\tilde{d}_{\tau+1} = \textbf{CentralDiffOracle}(\tilde{d}_\tau, \varepsilon, \alpha, L)$$
5: **end for**
6: $\tilde{D}_{\mathrm{T}} := \mathrm{diag}(\exp(\tilde{d}_{\mathrm{T}}))$
7: Return the scaling matrix $\tilde{D}_{\mathrm{T}}$

IV. NUMERICAL CASE STUDY

In order to demonstrate the effectiveness of our proposed model-free DK-iteration algorithm 1, we present a numerical example of a MIMO system with input-multiplicative uncertainty. We will compare our model-free procedure to a standard model-based DK synthesis method to make empirical observations of accuracy and convergence characteristics.

Uncertain Coupled Spring-Mass System: The nominal system is a fully-observable, coupled mass-spring system with two control inputs and four outputs. The model is extended to include two input uncertainties and two disturbances which enter through the same channel as the control input (i.e. $B_u = B_w = B_d$). The continuous-time state-space matrices and outputs for this system are given by:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -k/m_1 & k/m_1 & 0 & 0 \\ k/m_2 & -k/m_2 & 0 & 0 \end{bmatrix}, B_u = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1/m_1 & 0 \\ 0 & 1/m_2 \end{bmatrix}$$

$$C_v = 0, \quad C_e = \begin{bmatrix} 0 \\ Q^{1/2} \end{bmatrix}, \quad D_{uv} = I, \quad D_{ue} = \begin{bmatrix} R^{1/2} \\ 0 \end{bmatrix}$$

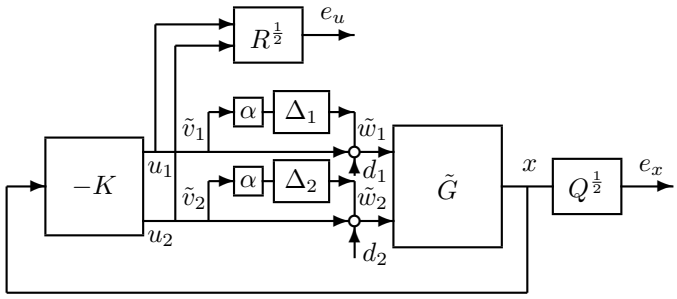

Fig. 3. Closed loop interconnection for a fully-observable coupled mass-spring system with input multiplicative uncertainty. The scaled plant $\tilde{G}$ is the interconnection of plant G and two scaled matrices D and D^{-1}. Where D is a 2×2 diagonal matrix. $\tilde{v}_1 = u_1$ and $\tilde{v}_2 = u_2$ are inputs of one dimensional uncertainties Δ_1 and Δ_2, respectively. Uncertainties outputs $\tilde{w}_1 = \Delta_1\tilde{v}_1$ and $\tilde{w}_2 = \Delta_2\tilde{v}_2$ along with control inputs u_1, u_2 and disturbance inputs d_1 and d_2 enter scaled plant $\tilde{G}$ as the plant inputs . $\tilde{w} := (\tilde{w}_1^\top, \tilde{w}_2^\top)^\top$, $d := (d_1^\top, d_2^\top)^\top$, $u := (u_1^\top, u_2^\top)^\top$ and $\tilde{v} := (\tilde{v}_1^\top, \tilde{v}_2^\top)^\top$ and $e := (e_u^\top, e_x^\top)^\top$ are system inputs and outputs in Figure 2, respectively.

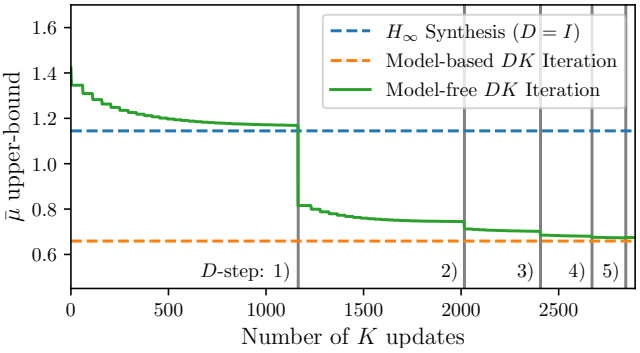

Fig. 4. The model-free DK-iteration Algorithm 1 is initialized with a stabilizing controller and five DK-iterations are run. For each update on K within **RARLSolver**, we plot the true achieved upper-bound $\bar{\mu}$ of problem (3). The value is piece-wise constant since a new $\bar{\mu}$ is only determined after completing the **RARLSolver** subroutine. The gray lines denote when a D step occurs, hence the instantaneous jumps. Values of $\bar{\mu}$ are also shown for nominal model-based $\mathcal{H}_\infty$ synthesis and DK-iteration.

with $m_1 = 1.0$, $m_2 = 0.5$, $k = 1.0$, $Q = I$ and $R = 0.1I$.

The discrete-time matrices are given by a zero-order-hold discretization of A and B_u with a sample time $t_s = 0.1$. As in the μ synthesis problem (3), we aim to minimize the gain from $h := (\tilde{w}, d)$ to $(\tilde{v}, e)$, where $e := (e_u^\top, e_x^\top)^\top$ is a performance output describing the state error e_x and control effort e_u. The multiplicative uncertainty is at the plant input as seen in Figure 3, which can account for a standard class of non-parametric errors. It is standard to design for normalized uncertainty $||\Delta||_\infty \leq 1$ (i.e. to aim for $\bar{\mu} \leq 1$). We include a constant factor of $\alpha = 0.25$ at the uncertainty input. This corresponds to an effective uncertainty of 25% in each input channel. We emphasize that our approach is model-free and does not require knowing the above model parameters.

To apply our model-free approach, we can formulate the following dynamic game cost for a fixed γ and diagonal scaling D:

$$J(u,h) = \mathbb{E}\sum_{k=1}^{\infty}\{x_k^\top Q x_k + u_k^\top (R + D^2)u_k - \gamma^2 h_k^\top \operatorname{diag}(D^2, I)h_k\}. \quad (13)$$

More details for the above formulation can be found in the appendix of our `arXiv` report [34]. Then the K-step is solved with this cost using the model-free **RARLSolver**. In this case study, the procedure **Approx-Dmin** is applied over 2×2 diagonal scaling matrices parameterized by $D = \exp(\operatorname{diag}(d_1, d_2))$. The **Approx-Kmin** (algorithm 2) is implemented using LSPI to perform the **RARLSolver** subroutine. An error constant of $\delta_0 = 1\text{e}{-1}$ and $\delta_\tau = 5\text{e}{-3}$ for all $\tau > 0$ is added to each new γ_τ estimate given by **HinfOracle**. The first constant is made to be larger because the first closed-loop under $\tilde{K}_0$ is often less robust and **HinfOracle** typically yields a higher error. We continue to apply the **RARLSolver** subroutine until γ can no longer be decreased by a small threshold value, where we then proceed to apply **Approx-Dmin** once again.

Figure 4 shows results from five total DK-iterations. We compare the values of $\bar{\mu}$ found by our Model-free DK-iteration algorithm 1 and fully model-based DK-iteration.[3] The horizontal axis of Figure 4 is the number updates on K that occur within the **RARLSolver** subroutine. **RARLSolver** will continue to update K until convergence, where it returns $\tilde{K}_{\tau+1}$ and a new γ_τ is computed, hence the piece-wise form of the graph. The iteration at which **Approx-Dmin** is applied is denoted by grey lines. As we get closer to the optimal $\bar{\mu}$ upper-bound, convergence in **Approx-Kmin** occurs more quickly and **Approx-Dmin** occurs more frequently.

In this first K-step when $D = I$, the application of **Approx-Kmin** is equivalent to $\mathcal{H}_\infty$ synthesis of the uncertain loop in Figure 3. The model-free **Approx-Kmin** approaches the optimal nominal $\mathcal{H}_\infty$ value when $D = I$. Moreover, the minimum upper bound $\bar{\mu}$ after Model-free DK-iterations (algorithm 1) approaches within 2% of the value found through model-based DK-iterations. Ultimately, the level of accuracy is determined by the accuracy of **HinfOracle** and the size of the error constants $\{\delta_\tau\}_\tau$ which must be large enough to yield an upper-bound. Clearly there is a trade-off between the number of samples expended on **HinfOracle** and the accuracy of $\bar{\mu}$. Fortunately, using a window length of $L = 100$ was sufficient to achieve the performance shown in Figure 4. Although there is no theoretic convergence guarantee for our proposed algorithm, this case study provides evidence that our algorithm is capable of achieving a near optimal γ in a model-free manner for the case of $\mathcal{H}_\infty$ synthesis and μ-synthesis. As shown in Figure 4, the convergence requires running the K-step for roughly 2500 times. Each time requires a full trajectory and consumes about 0.6 seconds of CPU time on a laptop. Hence the total CPU time is roughly 25 minutes. Of course this is much slower than model-based approaches. However, we emphasize that our goal is not to propose a computationally efficient method outperforming

[3]Notice that the model-based DK-iteration and the convex approach in [29] generate almost the same results on this example.

existing approaches in the model-based setting. It is our hope that the connection between adversarial RL and μ-synthesis can shed new light on data-driven robust control.

ACKNOWLEDGMENT

D. Keivan and G. Dullerud are partially funded by NSF under the grant ECCS 19-32735. A. Havens and B. Hu are generously supported by the NSF award CAREER-2048168 and the 2020 Amazon research award. P. Seiler is supported by the US ONR grant N00014-18-1-2209.

REFERENCES

[1] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[3] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[4] 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," *arXiv preprint arXiv:1509.02971*, 2015.

[5] A. Rajeswaran, K. Lowrey, E. Todorov, and S. Kakade, "Towards generalization and simplicity in continuous control," in *Advances in Neural Information Processing Systems*, 2017, pp. 6550–6561.

[6] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi, "Global convergence of policy gradient methods for the linear quadratic regulator," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 1467–1476.

[7] K. Krauth, S. Tu, and B. Recht, "Finite-time analysis of approximate policy iteration for the linear quadratic regulator," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8514–8524, 2019.

[8] Z. Yang, Y. Chen, M. Hong, and Z. Wang, "On the global convergence of actor-critic: A case for linear quadratic regulator with ergodic cost," *arXiv preprint arXiv:1907.06246*, 2019.

[9] L. Furieri, Y. Zheng, and M. Kamgarpour, "Learning the globally optimal distributed LQ regulator," in *Learning for Dynamics and Control*, 2020, pp. 287–297.

[10] J. P. Jansch-Porto, B. Hu, and G. E. Dullerud, "Convergence guarantees of policy optimization methods for Markovian jump linear systems," in *American Control Conference*, 2020, pp. 2882–2887.

[11] J. P. Jansch-Porto, B. Hu, and G. Dullerud, "Policy learning of MDPs with mixed continuous/discrete variables: A case study on model-free control of Markovian jump systems," in *Learning for Dynamics and Control*, 2020, pp. 947–957.

[12] B. Gravell, P. M. Esfahani, and T. H. Summers, "Learning optimal controllers for linear systems with multiplicative noise via policy gradient," *IEEE Transactions on Automatic Control*, 2020.

[13] H. Mohammadi, M. Soltanolkotabi, and M. R. Jovanović, "On the linear convergence of random search for discrete-time LQR," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 989–994, 2020.

[14] H. Mohammadi, A. Zare, M. Soltanolkotabi, and M. R. Jovanovic, "Convergence and sample complexity of gradient methods for the model-free linear quadratic regulator problem," *IEEE Transactions on Automatic Control*, 2021.

[15] N. Matni, A. Proutiere, A. Rantzer, and S. Tu, "From self-tuning regulators to reinforcement learning and back again," *arXiv preprint arXiv:1906.11392*, 2019.

[16] J. Morimoto and K. Doya, "Robust reinforcement learning," *Neural computation*, vol. 17, no. 2, pp. 335–359, 2005.

[17] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *International Conference on Machine Learning*, 2017, pp. 2817–2826.

[18] K. Zhang, B. Hu, and T. Başar, "On the stability and convergence of robust adversarial reinforcement learning: A case study on linear quadratic systems," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[19] K. Zhang, X. Zhang, B. Hu, and T. Başar, "Derivative-free policy optimization for linear risk-sensitive and robust control design: Implicit regularization and sample complexity," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[20] K. Zhang, B. Hu, and T. Başar, "Policy optimization for $\mathcal{H}_2$ linear control with $\mathcal{H}_\infty$ robustness guarantee: Implicit regularization and global convergence," *SIAM Journal on Control and Optimization*, vol. 59, no. 6, pp. 4081–4109, 2021.

[21] A. Packard, J. Doyle, and G. Balas, "Linear, multivariable robust control with a μ perspective," *ASME Journal Dynamic Systems, Measurement, and Control*, vol. 115, no. 2B, pp. 426–438, 1993.

[22] M. Honda and P. Seiler, "Uncertainty modeling for hard disk drives," in *American Control Conference*, 2014, pp. 3341–3347.

[23] T. Holicki, C. W. Scherer, and S. Trimpe, "Controller design via experimental exploration with robustness guarantees," *IEEE Control Systems Letters*, vol. 5, no. 2, pp. 641–646, 2020.

[24] H. J. van Waarde, M. K. Camlibel, and M. Mesbahi, "From noisy data to feedback controllers: Nonconservative design via a matrix S-lemma," *IEEE Transactions on Automatic Control*, vol. 67, no. 1, pp. 162–175, 2022.

[25] J. Berberich, C. W. Scherer, and F. Allgöwer, "Combining prior knowledge and data for robust controller design," *arXiv preprint arXiv:2009.05253*, 2020.

[26] G. E. Dullerud and F. Paganini, *A Course in Robust Control Theory: A Convex Approach*. Springer Science & Business Media, 2013, vol. 36.

[27] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*. Prentice Hall New Jersey, 1996, vol. 40.

[28] A. Packard and J. Doyle, "The complex structured singular value," *Automatica*, vol. 29, no. 1, pp. 71–109, 1993.

[29] A. Packard, K. Zhou, P. Pandey, and G. Becker, "A collection of robust control problems leading to lmis," in *Proceedings of the 30th IEEE Conference on Decision and Control*, 1991, pp. 1245–1250.

[30] T. Başar and P. Bernhard, *H_∞ Optimal Control and Related Minimax Design Problems: A Dynamic Game Approach*. Birkhäuser, Boston., 1995.

[31] D. Mustafa and K. Glover, "Minimum entropy $\mathcal{H}_\infty$ control," *Lecture Notes in Control and Information Sciences*, 1990.

[32] K. Glover and D. Mustafa, "Derivation of the maximum entropy $\mathcal{H}_\infty$-controller and a state-space formula for its entropy," *International Journal of Control*, vol. 50, no. 3, pp. 899–916, 1989.

[33] K. Glover and J. C. Doyle, "State-space formulae for all stabilizing controllers that satisfy an $\mathcal{H}_\infty$-norm bound and relations to risk sensitivity," *Systems & Control Letters*, vol. 11, no. 3, pp. 167–172, 1988.

[34] D. Keivan, A. Havens, P. Seiler, G. Dullerud, and B. Hu, "Model-free μ synthesis via adversarial reinforcement learning," *arXiv preprint arXiv:2111.15537*, 2021.

[35] C. R. Rojas, T. Oomen, H. Hjalmarsson, and B. Wahlberg, "Analyzing iterations in identification with application to nonparametric $\mathcal{H}_\infty$-norm estimation," *Automatica*, vol. 48, no. 11, pp. 2776–2790, 2012.

[36] B. Wahlberg, M. B. Syberg, and H. Hjalmarsson, "Non-parametric methods for l_2-gain estimation using iterative experiments," *Automatica*, vol. 46, no. 8, pp. 1376–1381, 2010.

[37] T. Oomen, R. van der Maas, C. R. Rojas, and H. Hjalmarsson, "Iterative data-driven $\mathcal{H}_\infty$ norm estimation of multivariable systems with application to robust active vibration isolation," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 6, pp. 2247–2260, 2014.

[38] S. Tu, R. Boczar, and B. Recht, "On the approximation of Toeplitz operators for nonparametric $\mathcal{H}_\infty$-norm estimation," in *American Control Conference*, 2018, pp. 1867–1872.

[39] M. Müller and C. R. Rojas, "Gain estimation of linear dynamical systems using Thompson sampling," in *The 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 89, 2019, pp. 1535–1543.

[40] S. Tu, R. Boczar, and B. Recht, "Minimax lower bounds for $\mathcal{H}_\infty$-norm estimation," in *American Control Conference*, 2019, pp. 3538–3543.

[41] T. Oomen, R. van der Maas, C. R. Rojas, and H. Hjalmarsson, "Iteratively learning the $\mathcal{H}_\infty$-norm of multivariable systems applied to model-error-modeling of a vibration isolation system," in *2013 American Control Conference*, 2013, pp. 6703–6708.

[42] A. Lamperski, "Computing stabilizing linear controllers via policy iteration," in *59th IEEE Conference on Decision and Control*, 2020, pp. 1902–1907.

[43] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. SIAM, 1994, vol. 15.

[44] S. Boyd and L. El Ghaoui, "Method of centers for minimizing generalized eigenvalues," *Linear algebra and its applications*, vol. 188, pp. 63–111, 1993.