Minimax Iterative Dynamic Game: Application to Nonlinear Robot Control Tasks.

Olalekan Ogunmolu, Nicholas Gans, and Tyler Summers.

***Abstract*— Multi-stage decision policies provide good control strategies in high-dimensional state spaces, particularly in complex autonomous tasks. However, they exhibit weak performance guarantees in the presence of disturbance, model mismatch, or model uncertainties. This brittleness limits their use in high-risk scenarios. We present a minimax iterative dynamic game for designing robust policies in the presence of an adversarial input. The algorithm is simple and is adaptable for designing meta-learning/deep policies that are robust against disturbances, model mismatch or model uncertainities, up to a disturbance bound. Videos of our results can be seen here: https://goo.gl/JhshTB.**

I. Introduction

Aided by the established principles of dynamic programming and optimal control theory, deep reinforcement learning (DRL) can learn high-dimensional real-world policies.

Our goal in this paper is to provide an underpinning for designing robust policies, leveraging on methods from H_∞ control theory [1], dynamic programming (DP) [2], differential dynamic programming (DDP) [3], and iterative LQG [4]. Essentially, we consider the performance of policies in the presence of various adversarial agents ([5]–[7]) using a minimax method. Methods of designing scalable high-dimensional policies often rely on heuristics e.g. [8], which do not always produce repeatable results. Quite often, policies are learned under partial observability, but sampling with partial observations can be unstable [9]. In the presence of model uncertainties or model mismatch between the source and target environments [10], we must therefore device policies that are robust to perturbations. While recent deep reinforcement learning (DRL) techniques produce performance efficiency for agent tasks in the real world ([11]–[17]), there are sensitivity concerns that need to be addressed, e.g. the trade-off between a system's nominal performance and its performance in the face of uncertainty or model mismatch.

Contributions

- In an *iterative, dynamic two player zero-sum Markov game*, we let each agent execute an opposite reaction to its pair: a concave-convex problem ensues, and the goal is to drive the policy toward a saddle point equilibrium, where the state is everywhere defined but possibly infinite-valued. Our iterative Dynamic Game (iDG) generates local control laws, which provide an alternating best response update of global control and adversarial policies – leading to a saddle-point equilibrium convergence. This is essentially a meta-algorithm that can in principle be extended to quantify and design the robustness of model-free, model-based RL, as well as DP/iterative linear quadratic Gaussian family of policies.

We evaluate our proposal for complex robot motor tasks using policy search [13], [18] and the ILQG algorithm [4]. The rest of this paper is thus organized: we provide a formal treatment policy sensitivity quantification and the iDG algorithm within a linearly solvable MDP [19] in Sec. II. The dynamics and model of the robot we do consider to demonstrate the feasibility of our iDG hypothesis is presented in Sec. III. Results for the two proposals of this paper are discussed in Sec. IV and we conclude the paper in Sec. V. We bring to the reader's attention that a more complete version of this work appeared in [20].

II. Two-Player Trajectory Optimization

Consider two agents interacting within an environment over a finite time horizon, T, whose states evolve according to the discrete-time stochastic dynamics,

$$\mathbf{x}_{t+1} = f_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{v}_t),\ t = 0, \dots, T-1,\ \mathbf{x}_0 = \bar{\mathbf{x}}_0,$$

where $\mathbf{x}_t \subseteq \mathcal{X}_t$ is the $n-$dimensional state vector, $\mathbf{u}_t \subseteq \mathcal{U}_t$ is the m-dimensional nominal agent's action (or control law), and $\mathbf{v}_t \in \mathcal{V}_t$ denote the adversarial agent's p-dimensional action. The nominal agent chooses its action under a (stochastic) policy $\{\pi = \pi_0, \pi_1, \dots, \pi_T\} \subseteq \Pi$, while the adversary's actions are governed by a policy $\{\psi = \psi_0, \psi_1, \dots, \psi_T\} \subseteq \Psi$. For the policy pair$(\pi, \psi)$, we define the *cost-to-go*, $\mathcal{J}(\mathbf{x}, \pi, \psi)$, of a trajectory $\{\mathbf{x}_t\}\ t = 0, \dots, T$ with initial condition $\mathbf{x}_0$, as a partial sum of costs from t to T,

$$\mathcal{J}_0(\mathbf{x}_0, \pi, \psi) = \mathbf{E}_{\mathbf{u}_t, \mathbf{v}_t} \sum_{t=0}^{T-1} \ell_t(\mathbf{x}_t, \mathbf{u}_t, \mathbf{v}_t) + L_T(\mathbf{x}_T),$$

where ℓ_t is a nonnegative function of $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{v}_t)$, denoting the stage cost, and L_T is a nonnegative function of $\mathbf{x}_T$, denoting the final cost. We seek a pair of *saddle point equilibrium* policies (π^*, ψ^*) that satisfy,

$$\mathcal{J}_0(\mathbf{x}_0, \pi^*, \psi) \leq \mathcal{J}_0(\mathbf{x}_0, \pi^*, \psi^*) \leq \mathcal{J}_0(\mathbf{x}_0, \pi, \psi^*),$$

$\forall\, \pi \in \Pi, \psi \in \Psi$ and $\mathbf{x}_0$. For the general case where we start from an initial condition $\mathbf{x}_t$, one may write the dynamic

Olalekan Ogunmolu and Nicholas Gans are with the Department of Electrical Engineering, Tyler Summers is with the Department of Mechanical Engineering, University of Texas at Dallas, Richardson, TX 75080, USA. {olalekan.ogunmolu, ngans, tyler.summers}@utdallas.edu

programming (DP) equation above as

$$V_t^*(\mathbf{x}_t) = \min_{\pi\in\Pi}\max_{\psi\in\Psi} V_t(\mathbf{x}_t,\pi,\psi),$$

where π and ψ contain the control sequences $\{\mathbf{u}_t\}$ and $\{\mathbf{v}_t\}$. The saddle point equilibrium for an optimal control sequence pair $\{\mathbf{u}_t^*, \mathbf{v}_t^*\}$ can be obtained with

$$\begin{aligned}\mathcal{J}_t^*(\mathbf{x}_t) &= \min_{\pi\in\Pi}\max_{\psi\in\Psi}\mathcal{J}_t(\mathbf{x}_t,\pi,\psi)\\ &= \min_{\pi\in\Pi}\max_{\psi\in\Psi}[\ell_t(\mathbf{x}_t,\mathbf{u}_t,\mathbf{v}_t)+\mathcal{J}_{t+1}^*\left(f_t\left(\mathbf{x}_t,\mathbf{u}_t,\mathbf{v}_t\right)\right)]\end{aligned} \tag{1}$$

A. Achieving Robustness via IDG

Suppose that the nominal policy of an agent is π, and consider an adversarial agent interacting with the nominal agent so that the closed-loop interaction of both agents is described by the linear difference equation,

$$\begin{aligned}\mathbf{x}_{t+1} &= f_t(\mathbf{x}_t,\mathbf{u}_t,\mathbf{v}_t), \quad \mathbf{u}_t\sim\pi_t\\ &= \bar{f}_t(\mathbf{x}_t,\mathbf{v}_t), \quad t=0,...,T-1.\end{aligned} \tag{2}$$

For stage costs of the form,

$$\ell_t(\mathbf{x}_t,\mathbf{u}_t,\mathbf{v}_t) = c_t(\mathbf{x}_t,\mathbf{u}_t) - \gamma g_t(\mathbf{v}_t),$$

where $c_t(\mathbf{x}_t,\mathbf{u}_t)$ represents the nominal stage cost, $g_t(\cdot)$ is a norm on the adversarial input[1] that penalizes the actions of the adversary, and $\gamma>0$ is a disturbance term that controls the strength of the adversary, the adversary faces a maximization problem of the form

$$\max_{\psi\in\Psi}\mathbf{E}_{\mathbf{u}_t\sim\pi_t}\sum_{t=0}^{T} c(\mathbf{x}_t,\mathbf{u}_t)-\gamma g(\mathbf{v}_t) = \max_{\psi\in\Psi}\mathbf{E}\sum_{t=0}^{T}\bar{\ell}_t^{\gamma}(\mathbf{x}_t,\mathbf{v}_t).$$

Varying γ increases/decreases the penalty incurred by the adversarial agent's actions. As $\gamma\to\infty$, the adversary's optimal policy is to do nothing, since any action will incur an infinite penalty; as γ decreases, the adversary incurs lower penalties, causing a larger system disturbance. The (inverse of the) smallest γ-value for which the adversary causes unacceptable performance degradation (e.g., instability) provides a measure of robustness of the nominal agent's policy π. The parameter γ is a distinguishing feature of this work; γ quantifies the $\mathcal{H}_\infty$ norm of the closed-loop system, a measure of its robustness to an adversarial input. The adversary need not represent a malicious input, but can be interpreted as a worst possible disturbance of a given magnitude.

In linear systems with quadratic costs and known nominal system dynamics, the $\mathcal{H}_\infty$ norm can be explicitly computed via generalized Riccati equations or semidefinite programs [5], [21]. In systems with nonlinear dynamics, one can use differential dynamic programming (DDP) [3] or iterative LQG [22] to optimize the *adversary* against the closed-loop system under the nominal agent's policy π. Indeed, Morimoto (in [23]) applied minimax DDP to a walking biped robot, but we noticed errors in the value function recursions[2].

[1]This formulation takes $\mathcal{V}_t$ as a vector space but one can as well define a nonnegative adversarial penalty term when $\mathcal{V}_t$ is a finite set.

[2]These corrections are given in (7).

We are motivated by the improved convergence guarantees of iterative LQG (iLQG) over DDP and its suitability for solving constrained nonlinear control problems by iteratively linearizing a nonlinear system about a local neighboring trajectory and computing the optimal local control laws. Our minimax iLQG framework facilitates learning control decision strategies that are robust in the presence of disturbances and modeling errors – improving upon nominal iLQG policies.

The adversary's policy may not be globally optimal due to the approximation of policies in continuous spaces and because the algorithm may only converge to a locally optimal solution, we obtain an upper bound on the robustness of the policy π.

B. IDG Formulation

We propose to arrive at a saddle point equilibrium with (1) by continually solving an online finite-horizon trajectory optimization problem. This is essentially a minimax framework and is applicable to any off/on-policy RL algorithms such as DQN, DDPG, or GPS. In this work, we generalize the online trajectory optimization algorithm of [22], to a two-player, zero-sum dynamic game as follows:

- we approximate the nonlinear system dynamics (c.f. (2)), starting with a schedule of the nominal agent's local controls, $\{\bar{\mathbf{u}}_t\}$, and nominal adversarial agent's local controls $\{\bar{\mathbf{v}}_t\}$ which are assumed to be available (when these are not available, we can initialize them to 0)
- we then run the system's passive dynamics with $\{\bar{\mathbf{u}}\},\{\bar{\mathbf{v}}\}$ to generate a nominal state trajectory $\{\bar{\mathbf{x}}_t\}$, with neighboring trajectories $\{\mathbf{x}_t\}$
- we choose a small neighborhood, $\{\delta\mathbf{x}_t\}$ of $\{\mathbf{x}_t\}$, which provides an optimal reduction in cost as the dynamics no longer represent those of $\{\mathbf{x}_t\}$
- discretizing time, the new state and control sequence pairs become $\delta\mathbf{x}_t=\mathbf{x}_t-\bar{\mathbf{x}}_t$, $\delta\mathbf{u}_t=\mathbf{u}_t-\bar{\mathbf{u}}_t$, $\delta\mathbf{v}_t=\mathbf{v}_t-\bar{\mathbf{v}}_t$.

Setting $V(\mathbf{x},T)=\ell_T(\mathbf{x}_T)$, the min-max over the entire control sequence reduces to a stepwise optimization over a single control, which proceeds backward in time with the *cost-to-go*

$$V(\mathbf{x}_t) = \min_{\mathbf{u}_t\sim\pi}\max_{\mathbf{v}_t\sim\psi}[\ell(\mathbf{x}_t,\mathbf{u}_t,\mathbf{v}_t)+V(f(\mathbf{x}_{t+1},\mathbf{u}_{t+1},\mathbf{v}_{t+1}))].$$

If we consider the Hamiltonian, $\ell(\cdot)+V(\cdot)$, as a perturbation around the tuple $\{\mathbf{x}_t,\mathbf{u}_t,\mathbf{v}_t\}$, the cost over the local neighborhood via an approximate Taylor series expansion becomes

$$Q_t = \ell(\mathbf{x}_t,\mathbf{u}_t,\mathbf{v}_t,t)+V(\mathbf{x}_{t+1},t+1).$$

A second-order approximation of the perturbed Q-coefficients of the LQR problem around the neighborhood $\{\delta\mathbf{x}_t\}$ of the trajectory $\{\mathbf{x}_t\}$ is defined as,

$$Q(\cdot)\approx\frac{1}{2}\begin{bmatrix}1\\ \delta\mathbf{x}_t^T\\ \delta\mathbf{u}_t^T\\ \delta\mathbf{v}_t^T\end{bmatrix}^T\begin{bmatrix}1 & Q_{\mathbf{x}_t}^T & Q_{\mathbf{u}_t}^T & Q_{\mathbf{v}_t}^T\\ Q_{\mathbf{x}_t} & Q_{\mathbf{x}\mathbf{x}_t} & Q_{\mathbf{u}\mathbf{x}_t} & Q_{\mathbf{v}\mathbf{x}_t}\\ Q_{\mathbf{u}_t} & Q_{\mathbf{u}\mathbf{x}_t} & Q_{\mathbf{u}\mathbf{u}_t} & Q_{\mathbf{v}\mathbf{u}_t}\\ Q_{\mathbf{v}_t} & Q_{\mathbf{v}\mathbf{x}_t} & Q_{\mathbf{v}\mathbf{u}_t} & Q_{\mathbf{v}\mathbf{v}_t}\end{bmatrix}\begin{bmatrix}1\\ \delta\mathbf{x}_t\\ \delta\mathbf{u}_t\\ \delta\mathbf{v}_t\end{bmatrix}, \tag{3}$$

where,

$$Q_{\mathbf{x}t} = \ell_{\mathbf{x}t} + f_{\mathbf{x}t}^T V_{\mathbf{x}t+1}, \quad Q_{\mathbf{u}t} = \ell_{\mathbf{u}t} + f_{\mathbf{u}t}^T V_{\mathbf{x}t+1}$$
$$Q_{\mathbf{v}t} = \ell_{\mathbf{v}t} + f_{\mathbf{v}t}^T V_{\mathbf{x}t+1}, \quad Q_{\mathbf{x}\mathbf{x}t} = \ell_{\mathbf{x}\mathbf{x}t} + f_{\mathbf{x}t}^T V_{\mathbf{x}\mathbf{x}t+1} f_{\mathbf{x}t}$$
$$Q_{\mathbf{u}\mathbf{x}t} = \ell_{\mathbf{u}\mathbf{x}t} + f_{\mathbf{u}t}^T V_{\mathbf{x}\mathbf{x}t+1} f_{\mathbf{x}t}, \; Q_{\mathbf{v}\mathbf{x}t} = \ell_{\mathbf{v}\mathbf{x}t} + f_{\mathbf{v}t}^T V_{\mathbf{x}\mathbf{x}t+1} f_{\mathbf{x}t}$$
$$Q_{\mathbf{u}\mathbf{u}t} = \ell_{\mathbf{u}\mathbf{u}t} + f_{\mathbf{u}t}^T V_{\mathbf{x}\mathbf{x}t+1} f_{\mathbf{u}t}, \; Q_{\mathbf{v}\mathbf{v}t} = \ell_{\mathbf{v}\mathbf{v}t} + f_{\mathbf{v}t}^T V_{\mathbf{x}\mathbf{x}t+1} f_{\mathbf{v}t}$$
$$Q_{\mathbf{u}\mathbf{v}t} = \ell_{\mathbf{u}\mathbf{v}t} + f_{\mathbf{u}t}^T V_{\mathbf{x}\mathbf{x}t+1} f_{\mathbf{v}t}.$$

Note that this is consistent with linearized methods, where linearized second moment terms will dominate the higher order terms [24]. The LQR approximation to the state and the optimal control performance index become,

$$\delta\mathbf{x}_{t+1} \approx f_{\mathbf{x}t}\delta\mathbf{x}_t + f_{\mathbf{u}t}\delta\mathbf{u}_t + f_{\mathbf{v}t}\delta\mathbf{v}_t$$
$$\ell(\mathbf{x}_t, \mathbf{u}_t, \mathbf{v}_t) \approx \frac{1}{2}\begin{bmatrix}1\\ \delta\mathbf{x}_t^T\\ \delta\mathbf{u}_t^T\\ \delta\mathbf{v}_t^T\end{bmatrix}^T \begin{bmatrix}\ell_{0t} & \ell_{\mathbf{x}t}^T & \ell_{\mathbf{u}t}^T & \ell_{\mathbf{v}t}^T\\ \ell_{\mathbf{x}t} & \ell_{\mathbf{x}\mathbf{x}t} & \ell_{\mathbf{u}\mathbf{x}t}^T & \ell_{\mathbf{v}\mathbf{x}t}^T\\ \ell_{\mathbf{u}t} & \ell_{\mathbf{u}\mathbf{x}t} & \ell_{\mathbf{u}\mathbf{u}t} & \ell_{\mathbf{u}\mathbf{v}t}\\ \ell_{\mathbf{v}t} & \ell_{\mathbf{v}\mathbf{x}t} & \ell_{\mathbf{v}\mathbf{u}t} & \ell_{\mathbf{v}\mathbf{v}t}\end{bmatrix}\begin{bmatrix}1\\ \delta\mathbf{x}_t\\ \delta\mathbf{u}_t\\ \delta\mathbf{v}_t\end{bmatrix}, \tag{4}$$

where single and double subscripts denote first and second-order derivatives[3]. The best possible (nominal agent) action and the worst possible (adversarial) action can be found by performing the respective arg min and arg max operations on the $Q-$ function in (3) so that

$$\delta\mathbf{u}_t^\star = -Q_{\mathbf{u}\mathbf{u}t}^{-1}\left[Q_{\mathbf{u}t}^T + Q_{\mathbf{u}\mathbf{x}t}\delta\mathbf{x}_t + Q_{\mathbf{u}\mathbf{v}t}\delta\mathbf{v}_t\right],$$
$$\delta\mathbf{v}_t^\star = -Q_{\mathbf{v}\mathbf{v}t}^{-1}\left[Q_{\mathbf{v}t}^T + Q_{\mathbf{v}\mathbf{x}t}\delta\mathbf{x}_t + Q_{\mathbf{v}\mathbf{u}t}\delta\mathbf{u}_t\right]. \tag{5}$$

The control strategies in (5) depend on the action of the other player. If the nominal agent first implements its strategy, then transmits its information to the adversary, which subsequently chooses its strategy; it follows that the adversary can choose a more favorable outcome since it knows what the nominal agent's strategy is. It becomes obvious that the *best* action for the nominal agent is to choose a control strategy that is an optimal response to the choice of the adversary. Similarly, if the roles of the players are changed, the nominal agent's response to the adversary's *worst* choice will be more favorable since it knows what the adversarial agent's strategy is. Therefore, it does not matter that the order of play is predetermined. We end up with an *iterative dynamic game*, where each agent's strategy depends on its previous actions. This ensures that we have a *cooperative game* in which the nominal and adversarial agent alternate between taking best possible and worst possible actions during the trajectory optimization phase. This helps maintain equilibrium around the system's desired trajectory, while ensuring robustness in local policies.

Suppose we set,

$$\mathbf{K}_{\mathbf{u}t} = \left[\left(I - Q_{\mathbf{u}\mathbf{u}t}^{-1}Q_{\mathbf{u}\mathbf{v}t}Q_{\mathbf{v}\mathbf{v}t}^{-1}Q_{\mathbf{u}\mathbf{v}t}^T\right)Q_{\mathbf{u}\mathbf{u}t}^{-1}\right]^{-1},$$
$$\mathbf{K}_{\mathbf{v}t} = \left[\left(I - Q_{\mathbf{v}\mathbf{v}t}^{-1}Q_{\mathbf{u}\mathbf{v}t}^TQ_{\mathbf{u}\mathbf{u}t}^{-1}Q_{\mathbf{u}\mathbf{v}t}\right)Q_{\mathbf{v}\mathbf{v}t}^{-1}\right]^{-1},$$

[3] Note that $\delta\mathbf{x}_k, \delta\mathbf{u}_k,$ and $\delta\mathbf{v}_k$ are measured w.r.t the nominal vectors $\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k, \bar{\mathbf{v}}_k$ and are not necessarily small.

and set

$$\boldsymbol{g}_{\mathbf{u}t} = \mathbf{K}_{\mathbf{u}t}(Q_{\mathbf{u}\mathbf{v}t}Q_{\mathbf{v}\mathbf{v}t}^{-1}Q_{\mathbf{v}t} - Q_{\mathbf{u}t}),$$
$$\boldsymbol{g}_{\mathbf{v}t} = \mathbf{K}_{\mathbf{v}t}(Q_{\mathbf{u}\mathbf{v}t}^TQ_{\mathbf{u}\mathbf{u}t}^{-1}Q_{\mathbf{u}t} - Q_{\mathbf{v}t}),$$
$$\boldsymbol{G}_{\mathbf{u}t} = \mathbf{K}_{\mathbf{u}t}\left(Q_{\mathbf{u}\mathbf{v}t}Q_{\mathbf{v}\mathbf{v}t}^{-1}Q_{\mathbf{v}\mathbf{x}t} - Q_{\mathbf{u}\mathbf{x}t}\right),$$
$$\boldsymbol{G}_{\mathbf{v}t} = \mathbf{K}_{\mathbf{v}t}\left(Q_{\mathbf{u}\mathbf{v}t}^TQ_{\mathbf{u}\mathbf{u}t}^{-1}Q_{\mathbf{u}\mathbf{x}t} - Q_{\mathbf{v}\mathbf{x}t}\right),$$

it follows that we can rewrite (5) as

$$\delta\mathbf{u}_t^\star = \boldsymbol{g}_{\mathbf{u}_t} + \boldsymbol{G}_{\mathbf{u}_t}\delta\mathbf{x}_t, \quad \delta\mathbf{v}_t^\star = \boldsymbol{g}_{\mathbf{v}_t} + \boldsymbol{G}_{\mathbf{v}_t}\delta\mathbf{x}_t. \tag{6}$$

eq. 6 gives the open-loop and closed-loop components of the control equations for both agents. Comparing coefficients in (4), we find that the value function coeeficients can be thus written

$$\Delta V_t = \boldsymbol{g}_{\mathbf{u}t}Q_{\mathbf{u}t} + \boldsymbol{g}_{\mathbf{v}t}Q_{\mathbf{v}t} + \boldsymbol{g}_{\mathbf{u}t}Q_{\mathbf{u}\mathbf{v}t}\boldsymbol{g}_{\mathbf{v}t} + \frac{1}{2}\left(\boldsymbol{g}_{\mathbf{u}t}Q_{\mathbf{u}\mathbf{u}t}\boldsymbol{g}_{\mathbf{u}t} + \boldsymbol{g}_{\mathbf{v}t}Q_{\mathbf{v}\mathbf{v}t}\boldsymbol{g}_{\mathbf{v}t}\right)$$
$$V_{\mathbf{x}t} = Q_{\mathbf{x}t} + \boldsymbol{G}_{\mathbf{u}_t}^TQ_{\mathbf{u}t} + \boldsymbol{G}_{\mathbf{v}_t}^TQ_{\mathbf{v}t} + \boldsymbol{G}_{\mathbf{u}t}^TQ_{\mathbf{u}\mathbf{u}t}\boldsymbol{g}_{\mathbf{u}t} + \boldsymbol{g}_{\mathbf{u}t}Q_{\mathbf{u}\mathbf{x}t} + \boldsymbol{g}_{\mathbf{v}t}Q_{\mathbf{v}\mathbf{x}t} + \boldsymbol{G}_{\mathbf{v}t}^TQ_{\mathbf{v}\mathbf{v}t}\boldsymbol{g}_{\mathbf{v}t} + \boldsymbol{G}_{\mathbf{v}t}^TQ_{\mathbf{u}\mathbf{v}t}^T\boldsymbol{g}_{\mathbf{u}t} + \boldsymbol{G}_{\mathbf{u}t}^TQ_{\mathbf{u}\mathbf{v}t}\boldsymbol{g}_{\mathbf{v}t}$$
$$V_{\mathbf{x}\mathbf{x}t} = \frac{1}{2}(Q_{\mathbf{x}\mathbf{x}t} + \boldsymbol{G}_{\mathbf{u}_t}^TQ_{\mathbf{u}\mathbf{u}t}\boldsymbol{G}_{\mathbf{u}_t} + \boldsymbol{G}_{\mathbf{v}_t}^TQ_{\mathbf{v}\mathbf{v}t}\boldsymbol{G}_{\mathbf{v}_t}) + \boldsymbol{G}_{\mathbf{u}_t}^TQ_{\mathbf{u}\mathbf{x}t} + \boldsymbol{G}_{\mathbf{v}_t}^TQ_{\mathbf{v}\mathbf{x}t} + \boldsymbol{G}_{\mathbf{u}_t}^TQ_{\mathbf{u}\mathbf{v}t}\boldsymbol{G}_{\mathbf{v}_t}. \tag{7}$$

These recursive value functions, essentially differentiate our value coefficient recursion equations from Morimoto's DDP recursions.

III. Dynamics Modeling and Simulation

We consider the KUKA youbot[4] platform which has four mecanum wheels, capable of spatial $\{x, y\}$ motion, *i.e.* sideways, and forward, and an in-place θ-rotation about the z-axis (see Fig. 1), with low ground friction and driving torque. It is equipped with a five DoF arm, mounted on its base. We use the complete kinematic and dynamic model of the youbot platform, accounting for the wheels' friction and mass, while neglecting the links' masses and its associated inertia forces. Since the robot moves in the $x-y$ plane, we limit ourselves to the 3-DoF dynamics in Cartesian space. The model is derived in spatial (world) coordinates as, $\boldsymbol{x}_I = \left[\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\right]^T \equiv \left[x_I, y_I, \theta_I\right]^T$, and the local frame of the robot is defined as $\boldsymbol{x}_R = \left[x_R, y_R, \theta_R\right]^T$ (see Fig. 1).

The torques that govern the robot's motion are obtained from [25]. We run our experiments in the Gazebo physics engine [26], which has its reference frame defined as x pointing forward, y pointing sideways and z pointing up. Therefore, our reference frame and robot geometry are as illustrated in Figs 1 and 2. Formally, we define the generalized Lagrangian equation of the robot as,

$$\boldsymbol{M}(\boldsymbol{x})\ddot{\boldsymbol{x}} + \boldsymbol{C}(\boldsymbol{x}, \ddot{\boldsymbol{x}})\dot{\boldsymbol{x}} + \boldsymbol{B}^T\boldsymbol{S}\boldsymbol{f} = \frac{1}{\boldsymbol{r}}\boldsymbol{B}^T\boldsymbol{\tau} \tag{8}$$

where $\boldsymbol{\tau} = [\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \boldsymbol{\tau}_3, \boldsymbol{\tau}_4]$ is the wheel torque vector, $\boldsymbol{r}$ is the wheel radius, $\boldsymbol{f} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3, \boldsymbol{f}_4]^T$ is the friction vector, and $\boldsymbol{S}$ and $\boldsymbol{B}$ map the inverse kinematics, gravity, external forces

[4] https://goo.gl/CYTjvD

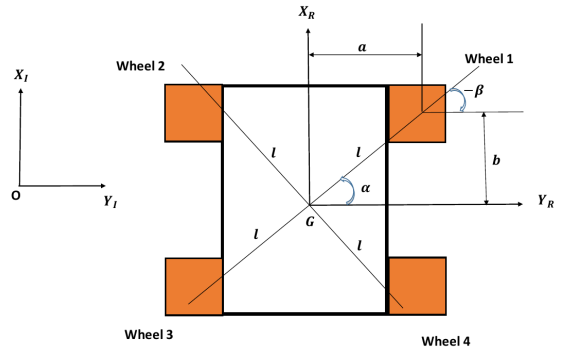


Fig. 1: Mecanum Wheels Model.

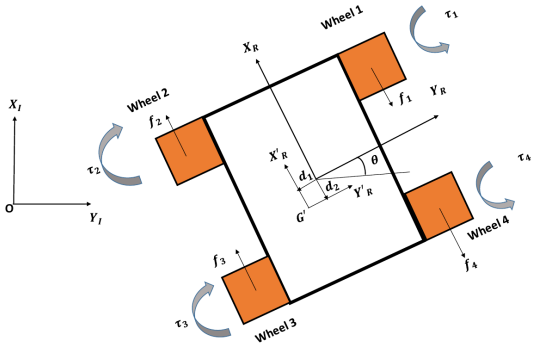


Fig. 2: Robot frames convention

and robot's angle, θ, to each wheel torque; matrices $\boldsymbol{M}$ and $\boldsymbol{C}$ denote the inertia and coriolis properties of the robot. $\boldsymbol{B}$ and $\boldsymbol{S}$ are given by,

$$\boldsymbol{B} = \begin{bmatrix} -(\cos\theta - \sin\theta) & -(\cos\theta + \sin\theta) & -\sqrt{2}l\sin(\zeta) \\ -(\cos\theta + \sin\theta) & (\cos\theta - \sin\theta) & -\sqrt{2}l\sin(\zeta) \\ (\cos\theta - \sin\theta) & (\cos\theta + \sin\theta) & -\sqrt{2}l\sin(\zeta) \\ (\cos\theta + \sin\theta) & -(\cos\theta - \sin\theta) & -\sqrt{2}l\sin(\zeta) \end{bmatrix}$$

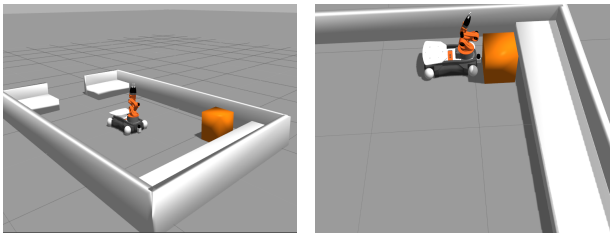
$$\boldsymbol{S} = \text{diag}\left[sgn(\dot{\phi}_1),\ sgn(\dot{\phi}_2),\ sgn(\dot{\phi}_3),\ sgn(\dot{\phi}_4);\right]$$

$\zeta = \pi/4 - \alpha$, l is the mounting distance of the wheels as shown in Fig. 1, and $\dot{\phi}_i$, is the rotation speed of each wheel about its axis of rotation. We apply the generalized force/torque vector, F_i, to the base frame of the robot, defined as, [27]

$$F_i = \sum_{j=1}^{4} \left(\boldsymbol{\tau}_j - r\, sgn(\dot{\phi}_j)\boldsymbol{f}_j\right) \frac{\partial \dot{\phi}_j}{\partial \dot{\boldsymbol{x}}_i},\ i = \{1,2,3\} \tag{9}$$

A. Trajectory Optimization: ILQR

This section describes the navigation of the robot using the nominal ILQR algorithm, the objective function design and specific initializations for the robot. The goal is for the robot to move optimally from the center of the environment in Fig. 3a to within a $0.1m$ radius of the orange box attached to the right-hand corner of Fig. 3b. We separate the state and control cost terms for easier manipulation of cost components. For the state's instanteneous cost, we define a pseudo "smooth-abs" function,



(a) Home Position. (b) Goal State.

Fig. 3: Goal Navigation Illustration

$$\ell(\mathbf{x}_t) = \sqrt{\alpha + (\mathbf{x}_t - \mathbf{x}^\star)^T\, \mathbf{diag}(\boldsymbol{w}_x)\, (\mathbf{x}_t - \mathbf{x}^\star)}, \tag{10}$$

where $\mathbf{x}^\star$ denotes the desired state, $\boldsymbol{w}_x$ is a state penalty vector and α is a constant that controls the curvature of the cost function: the lower the α value, the smoother is the robot's trajectory near the goal state. The $(\mathbf{x}-\mathbf{x}^\star)$ term encourages the robot to follow the nominal trajectory while driving toward the goal state. This l_{12} function enforces reaching a desired target exactly in 3D space. Equation 10 is particularly useful because it encourages the relative weighting of state-cost terms along different axes of motions. Inspired by [22], we choose a hyperbolic cosine control cost function instead of a quadratic cost defined as:

$$\ell(\mathbf{u}_t) = \alpha^2 \left(\cosh(\boldsymbol{w}_u^T \mathbf{u}_t) - 1\right), \tag{11}$$

where $\boldsymbol{w}_u$ is a control penalty vector. The final cost is non-dependent on the control signal. This cost function limits control outputs to an α-sized neighborhood in $\mathbf{u}-$space. α is set to 10^{-4}, $\boldsymbol{w}_u = [1,1,1,1]$ and $\boldsymbol{w}_x = [1,1,0.8]$. We found a time horizon, $T = 150$ to generally work well for this task. We proceed as follows:

- starting with an open-loop control schedule $\{\bar{\mathbf{u}}_t\}$, (initialized to $[1.3, 0.8, 0.1]$), we generate the nominal states $\{\bar{\mathbf{x}}_t\}$
- replacing $\boldsymbol{\tau}$ with $\mathbf{u}$ in (8), we compute the forward dynamics

$$\ddot{\boldsymbol{x}} = -\boldsymbol{M}^{-1}\boldsymbol{C}\dot{\boldsymbol{x}} - \boldsymbol{M}^{-1}\boldsymbol{B}^T\left(\boldsymbol{S}\boldsymbol{f} - \frac{1}{\boldsymbol{r}}\mathbf{u}\right) \tag{12}$$

- we then obtain derivatives of $\boldsymbol{f}$ and those of ℓ from (4)
- starting at time $t = T-1$, we compute the associated nominal Q coefficients in (4), obtain the open and closed-loop gains, $\boldsymbol{g}_\mathbf{u}, \boldsymbol{G}_\mathbf{u}$, and obtain the value function derivatives
- in the forward pass, we proceed from $t = \{0, \ldots, T-1\}$, and update the trajectories with a backtracking linesearch parameter, $0 < \varsigma \leq 1$, as follows

$$\begin{aligned} \hat{\mathbf{x}}(1) &= \mathbf{x}(1) \\ \hat{\mathbf{u}}(t) &= \mathbf{u}(t) + \varsigma \boldsymbol{g}_\mathbf{u}(t) + \boldsymbol{G}_\mathbf{u}(t)(\hat{\mathbf{x}}(t) - \mathbf{x}(t)) \\ \hat{\mathbf{x}}(t+1) &= \boldsymbol{f}\left(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t)\right) \end{aligned} \tag{13}$$

- the backward/forward pass informs of the change in cost $\Delta J^\star$, which is compared to the estimated reduction

$\eta = \left(J(\mathbf{u}_{1,\ldots,T-1}) - J(\hat{\mathbf{u}}_{1,\ldots,T-1})\right) / \Delta(J(\rho))$, where

$$\Delta(J(\rho)) = \rho \sum_{t=1}^{T-1} \boldsymbol{g}_{\mathbf{u}t}^T \mathbf{Q}_{\mathbf{u}t} + \frac{\rho^2}{2} \sum_{i=1}^{T-1} \boldsymbol{g}_{\mathbf{u}t}^T \mathbf{Q}_{\mathbf{u}\mathbf{u}t} \boldsymbol{g}_{\mathbf{u}t}.$$

- we accept a trajectory only if $0 < c < \eta$ (where we have chosen $c = 0.5$) following [3]'s recommendation.
- set $\bar{\mathbf{x}}_t = \mathbf{x}_t (t = 1, \ldots, T)$, $\mathbf{u}_t = \bar{\mathbf{u}}_i$, reset ρ and repeat the forward pass
- stop when $\Delta(J(\rho)) < \chi$ (where χ is a problem dependent term)

B. Trajectory Optimization: IDG

Here, our goal is to evaluate the robustness of the ILQG algorithm on the robot navigation task with the arm as the unmodeled dynamics.We provide an overview of our assumptions and specific initializations for the robot. We initialized the adversarial inputs $\mathbf{v}$ from a Gaussian distribution $\sim \mathcal{N}(\mathbf{0}, 2\boldsymbol{I})$ and augment the stage control cost of (11) as follows

$$\ell(\mathbf{u}_t, \mathbf{v}_t) = \alpha^2 \left(\cosh(\boldsymbol{w}_u^T \mathbf{u}_t) - \gamma \cosh(\boldsymbol{w}_v^T \mathbf{v}_t)\right). \tag{14}$$

$\gamma \cosh(\boldsymbol{w}_v^T \mathbf{v}_t)$ introduces a weighting term in the disturbing input, with γ as the robustness parameter. This framework facilitates learning control decision strategies that are robust in the presence of disturbances and modeling errors – improving upon generic optimal control, DDP and ILQG policies. We set $\boldsymbol{w}_v$ to $[1, 1, 1, 1]$ and run the two player, zero-sum game erstwhile described. Again, we compute the derivatives $f_{\mathbf{u}t}$ and $f_{\mathbf{v}t}$, calculate the associated derivatives in (4) and give the optimized iDG torque to the robot. We initialized the nominal disturbance vector $\bar{\mathbf{v}}$ as a multivariate Gaussian-filtered, random noise vector $\sim \mathcal{N}(\mathbf{0}, 2\mathbf{I})$. $\boldsymbol{v}_t$ is similarly initialized before we run the iDG algorithm. Our iDG process goes thus:

- starting with an open-loop control and distrurbance schedules $\{\bar{\mathbf{u}}_t\}$, $\{\bar{\mathbf{v}}_t\}$, we generate the nominal states $\{\bar{\mathbf{x}}_t\}$
- we then obtain the derivatives $f_{\mathbf{u}t}$, $f_{\mathbf{v}t}$, $f_{\mathbf{u}\mathbf{v}t}$, $f_{\mathbf{u}\mathbf{u}t}$, $f_{\mathbf{v}\mathbf{v}t}$ and those of ℓ from (4)
- in a backward pass, we obtain the associated nominal Q coefficients, the open and closed-loop gains, and obtain the improved value function coefficients with (7)
- in the forward pass, we proceed from $t = \{0, \ldots, T-1\}$, and update the trajectories with a backtracking linesearch parameter, $0 < \varsigma \leq 1$, as follows

$$\begin{aligned}
\hat{\mathbf{x}}(1) &= \mathbf{x}(1) \\
\hat{\mathbf{u}}(t) &= \mathbf{u}(t) + \varsigma \boldsymbol{g}_{\mathbf{u}}(t) + \boldsymbol{G}_{\mathbf{u}}(t)(\hat{\mathbf{x}}(t) - \mathbf{x}(t)) \\
\hat{\mathbf{v}}(t) &= \mathbf{v}(t) + \varsigma \boldsymbol{g}_{\mathbf{v}}(t) + \boldsymbol{G}_{\mathbf{v}}(t)(\hat{\mathbf{x}}(t) - \mathbf{x}(t)) \\
\hat{\mathbf{x}}(t+1) &= \boldsymbol{f}\left(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t), \hat{\mathbf{v}}(t)\right)
\end{aligned} \tag{15}$$

- the backward/forward pass informs of the change in cost $\Delta J^\star$, which is compared to the estimated reduction

$\eta = \frac{J(\mathbf{u}_{1,\ldots,T-1}, \mathbf{v}_{1,\ldots,T-1}) - J(\hat{\mathbf{u}}_{1,\ldots,T-1}, \hat{\mathbf{v}}_{1,\ldots,T-1})}{\Delta(J(\rho))}$, where

$$\Delta(J(\rho)) = \rho \sum_{t=1}^{T-1} \left[\boldsymbol{g}_{\mathbf{u}}(t)^T \mathbf{Q}_{\mathbf{u}}(t) + \boldsymbol{g}_{\mathbf{v}}(t)^T \mathbf{Q}_{\mathbf{v}}(t)\right] + \ldots$$

$$\ldots \frac{\rho^2}{2} \sum_{i=1}^{T-1} \left[\boldsymbol{g}_{\mathbf{u}}(t)^T \mathbf{Q}_{\mathbf{u}\mathbf{u}}(t) \boldsymbol{g}_{\mathbf{u}}(t) + \boldsymbol{g}_{\mathbf{v}}(t)^T \mathbf{Q}_{\mathbf{v}\mathbf{v}}(t) \boldsymbol{g}_{\mathbf{v}}(t)\right]$$

- we again accept a trajectory only if $0 < c < \eta$ (choosing $c = 0.5$)
- and set $\bar{\mathbf{x}}_t = \mathbf{x}_t (t = 1, \ldots, T)$, $\mathbf{u}_t = \bar{\mathbf{u}}_t$, $\mathbf{v}_t = \bar{\mathbf{v}}_t$, reset ρ and repeat the forward pass
- stop when $\Delta(J(\rho)) < \chi$ (where χ is a problem dependent term).

IV. Results

We run the iDG optimization algorithm on the youbot and analyze the effectiveness of using our method against standard ILQG.

A. ILQG-based Trajectory Optimization

We evaluate the trajectory optimization algorithm on the youbot KUKA robotic platform. The iLQG result is best understood by watching the ILQG video result available here: https://goo.gl/JhshTB. In the video, observe that the arm on the robot vibrates with abrupt motions while navigating toward the desired goal. This is expected, as the dynamics of the arm links were not included in the model of the platform. Regardless, the robot reaches the goal state after 76 seconds.

B. Trajectory Optimization: iDG

Here, we set T to 150, initiate the disturbance from a multivariate Gaussian filtered noise $\mathcal{N}(0, 2)$, set the gains for the torques to $\{k_{F_x} = 10, k_{F_y} = 15, k_{F_\theta} = 1\}$ respectively and run the algorithm described in III-B for various γ-disturbance values. We run two experiments: one with the goal at far left corner of the environment and the second with the goal state at the far right corner of the environment. We pose various adversarial inputs against the controller with values of γ in the range $\{10, 5, 3, 1.5, 0.65, 0.1, 0.1, 10^{-5}, 2\times10^{-3}, 2\times 10^{-5}\}$ for both experiments. The result showing the trajectory evolution and the position of the youbot after each iDG run for various γ values is better appreciated by watching the videos available on our website: https://goo.gl/JhshTB.

In both experiments I and II, observe that for values of γ in the range $1.5 \leq \gamma \leq 10$, the robot drives smoothly and reaches the goal without the arm vibrating on the platform as in the ILQG approach. This is despite the fact that we did not take the mass and inertia matrix components of the arm links into consideration. When $\gamma \leq 1.5$, we notice that the adversarial disturbance's effect on the overall system start becoming pronounced. While the robot still reaches its desired goal state for $\gamma = 0.5$ and 1.5, the motion of the arm is no longer stable. Indeed, for critical values of $\gamma \leq 10^{-3}$, the trajectory of the robot becomes perturbed as well as the balance of the robot on the arm. As γ becomes very small (below 10^{-5}), the robot's trajectory is undefined, and it converges to a spurious minimum as both experiments

show. This γ indices would correspond to the $\mathcal{J}^{\star}_{\gamma^{\star}}$ that depicts unacceptable performance.

The downside to the iDG approach is the longer time requirement to finish the trajectory optimization process (in the videos, the frames are sped up by up to $8X$). There is a tradeoff that we incur in attempting to solve a policy optimization scheme optimally with our iDG algorithm. While iLQG does achieve the goal in generally 3-4 iterations, iDG solves the same task in about 5-7 iterations. When speed is not crucial, and the safety of a real-world agent/its environment is important, iDG provides a viable alternative for designing safe policies. Without loss of generality, we envisage that our minimax iDG scenario is extensible to similar systems that compute gradients of a cost function or reward in achieveing an optimal control task. Compared to Morimoto's work [23], our minimax iDG does not have to learn the unmodeled disturbance with reinforcement learning as [23] had to do for their walking biped. We thus identify the critical value of γ to be between $0.1 - 10^{-3}$ as being the value that causes maximal disturbance in the robot's torque. To design a robust policy, one can consider γ values in this range in order to design an smooth trajectory for this particular using the minimax iDG algorithm.

V. Conclusions

Designing high-dimensional policies is a laborious task for complex autonomous agents. DRL is one of the promising ways of coping with such high-dimensional state space policy design. Despite their performance effectiveness, they are often brittle in the presence of adversarial agents (as we have shown), model mismatch or policy transfer [8], [28]. If we can provide a measure of the disturbance that a policy can withstand, we would be aware of the limits of such policy aforetime. In this work, we have presented a way of identifying the greatest upper bound on a policy's sensitivity via an alternating minimax framework. We have also presented an iDG framework that informs of how to make a policy robust to unmodeled disturbances, and uncertainities, up to a disturbance bound by converging to a saddle equilibrium. We ran four experiments in total to validate our intuition and hypothesis and the videos of our results are available on our website. This is a promising path going forward in applying this method to high-dimensional deep RL policies for continuous action spaces. We are currently improving the design and structure of our end-to-end deep reinforcement learning algorithm with policy search methods and the experimental implementation of our iDG algorithm on the youbot robot.

References

[1] K. Zhou and J. C. Doyle, *Essentials of Robust Control*. Prentice hall Upper Saddle River, NJ, 1998, vol. 104. 1

[2] R. Bellman, *Dynamic programming*. Princeton University Press, 1957. 1

[3] D. H. Jacobson and D. Q. Mayne, *Differential Dynamic Programming*. American Elsevier Publishing Company, Inc., New York, NY, 1970. 1, 2, 5

[4] E. Todorov and W. Li, "A Generalized Iterative LQG Method For Locally-optimal Feedback Control Of Constrained Nonlinear Stochastic Systems," *IEEE Conference on Decision and Control*, 2004. 1

[5] Basar, Tamer and Olsder, Geert Jan, *Dynamic Noncooperative Game Theory*. Academic Press, New York, 1999. 1, 2

[6] M. L. Littman, "Markov Games as a Framework for Multi-agent Reinforcement Learning," in *Proceedings of the Eleventh International Conference on Machine Learning*, vol. 157, 1994, pp. 157–163. 1

[7] J. Morimoto and K. Doya, "Robust Reinforcement Learning," *Neural computation*, vol. 17, no. 2, pp. 335–359, 2005. 1

[8] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust Adversarial Reinforcement Learning," *arXiv preprint arXiv:1703.02702*, 2017. 1, 6

[9] J. Garcıa and F. Fernández, "A Comprehensive Survey on Safe Reinforcement Learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015. 1

[10] A. Mandlekar, Y. Zhu, A. Garg, L. Fei-Fei, and S. Savarese, "Adversarially Robust Policy Learning: Active Construction of Physically-Plausible Perturbations," 2017. 1

[11] I. Mordatch, K. Lowrey, G. Andrew, Z. Popovic, and E. V. Todorov, "Interactive control of diverse complex characters with neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 3132–3140. 1

[12] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning Deep Control Policies for Autonomous Aerial Vehicles with MPC-Guided Policy Search," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 528–535. 1

[13] S. Levine and V. Koltun, "Learning complex neural network policies with trajectory optimization," in *International Conference on Machine Learning*, 2014, pp. 829–837. 1

[14] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," *Journal of Machine Learning Research*, vol. 17, pp. 1–40, 2016. 1

[15] W. Montgomery and S. Levine, "Guided Policy Search as Approximate Mirror Descent," *arXiv preprint arXiv:1607.04614*, 2016. 1

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. 1

[17] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," in *International Conference on Machine Learning*, 2016, pp. 1928–1937. 1

[18] M. P. Deisenroth, G. Neumann, and J. Peters, "A Survey on Policy Search for Robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1, pp. 1–142, 2011. 1

[19] K. Dvijotham and E. Todorov, "Inverse Optimal Control with Linearly-Solvable MDPs," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 335–342. 1

[20] O. Ogunmolu, N. Gans, and T. Summers, "Minimax Iterative Dynamic Game: Application to Nonlinear Robot Control Tasks," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018. 1

[21] J. Doyle, "Guaranteed Margins for LQG Regulators," *IEEE Transactions on Automatic Control*, vol. 23, no. 4, pp. 756–757, 1978. 2

[22] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2012. 2, 4

[23] J. Morimoto, G. Zeglin, and C. Atkeson, "Minimax Differential Dynamic Programming: Application to A Biped Walking Robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems.*, vol. 2, no. October, pp. 1927–1932, 2003. 2, 6

[24] M. M. Polycarpou and P. A. Ioannou, "Modelling, Identification And Stable Adaptive Control Of Continuous-time Nonlinear Dynamical Systems Using Neural Networks," in *American Control Conference, 1992*. IEEE, 1992, pp. 36–40. 3

[25] L.-C. Lin and H.-Y. Shih, "Modeling and adaptive control of an omni-mecanum-wheeled robot," *Intelligent Control and Automation*, vol. 4, no. 02, p. 166, 2013. 3

[26] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004.*, vol. 3. IEEE, 2004, pp. 2149–2154. 3

[27] M. W. Spong, S. Hutchinson, M. Vidyasagar, *et al.*, *Robot modeling and control*. Wiley New York, 2006, vol. 3. 4

[28] J. Kos and D. Song, "Delving into Adversarial Attacks on Deep Policies," *arXiv preprint arXiv:1705.06452*, 2017. 6