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

Olalekan Ogunmolu, Nicholas Gans, and Tyler Summers.

Abstract—Deep reinforcement learning policies provide good control decision 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 how to quantify the sensitivity of deep reinforcement learning policies – this would inform of its robustness capacity. We then formulate 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 uncertainties, 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. However, such deep policies are often brittle to deploy on real-world systems owing to their lack of robustness [1], [2] and the data inefficiency of the learning process.

Methods of designing scalable deep policies often rely on heuristics e.g. [3], which do not always produce repeatable results. Quite often, policies are learned under partial observability, but sampling with partial observations can be unstable [4]. In the presence of model uncertainties or model mismatch between the source and target environments [5], we must therefore device policies that are robust to perturbations. Our goal in this paper is to provide an underpinning for designing robust policies, leveraging on methods from $H_{\infty}$ control theory [6], dynamic programming (DP) [7], differential dynamic programming (DDP) [8], and iterative LQG [9].

Essentially, we consider the performance of policies in the presence of various adversarial agents ([10]–[12]) using a minimax method. While recent DRL techniques produce performance efficiency for agent tasks in the real world ([13]–[19]), 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

- We provide a framework for testing the brittleness of a policy: given a nominal policy, we pit adversarial agents against it, whose magnitude is controlled by a disturbance parameter $\gamma$. If an adversarial agent causes significant performance degradation, then we have identified the upper bound on the efficiency of such policy, and such policy will probably not generalize beyond its training environment. This provides an informative performance-robustness trade-off curve that can guide in the design of the policies that we consider. We demonstrate that the most sophisticated deep policies fail in the presence of adversarial perturbations.

- In an iterative, dynamic two player zero-sum Markov game, each agent executes an opposite reaction to its pair: a concave-convex problem ensues, and our 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 continuous control tasks in complex robot motor tasks using policy search [15], [20] and the ILQG algorithm [9]. The content of this paper is thus organized: we provide a formal treatment policy sensitivity quantification and the iDG algorithm within a linearly solvable MDP [21] 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.

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,

$$x_{t+1} = f_t(x_t, u_t, v_t), \quad t = 0, \ldots, T-1, \quad x_0 = x_0,$$

where $x_t \subseteq X_t$ is the $n$–dimensional state vector, $u_t \subseteq U_t$ is the $m$-dimensional nominal agent’s action (or control law), and $v_t \subseteq V_t$ denote the adversarial agent’s $p$-dimensional action. The nominal agent chooses its action under a (stochastic) policy $\{\pi = \pi_0, \pi_1, \ldots, \pi_T\} \subseteq \Pi$, while the adversary’s actions are governed by a policy $\{\psi = \psi_0, \psi_1, \ldots, \psi_T\} \subseteq \Psi$. For the policy pair($\pi, \psi$), we define the cost-to-go, $J(x, \pi, \psi)$, of a trajectory $\{x_t\} \ t = 0, \ldots, T$ with initial condition $x_0$,
Varying \( \pi \) where \( \ell \) is a nonnegative function of \((x_t, u_t, v_t)\), denoting the stage cost, and \( L_T \) is a nonnegative function of \( x_T \), denoting the final cost. We seek a pair of saddle point equilibrium policies \((\pi^*, \psi^*)\) that satisfy,

\[
J_0(x_0, \pi^*, \psi) \leq J_0(x_0, \pi^*, \psi^*) \leq J_0(x_0, \pi, \psi^*),
\]

\(\forall \pi \in \Pi, \psi \in \Psi\) and \(x_0\). For the general case where we start from an initial condition \(x_0\), one may write the dynamic programming (DP) equation above as

\[
V^*_t(x_t) = \min_{\pi \in \Pi} \max_{\psi \in \Psi} \left[ \ell_t(x_t, u_t, v_t) + J_{t+1}^*(f_t(x_t, u_t, v_t)) \right],
\]

where \(\pi\) and \(\psi\) contain the control sequences \(\{u_t\}\) and \(\{v_t\}\). The saddle point equilibrium for an optimal control sequence pair \(\{u^*_t, v^*_t\}\) can be obtained with

\[
J^*_t(x_t) = \min_{\pi \in \Pi} \max_{\psi \in \Psi} J_t(x_t, \pi, \psi)
= \min_{\pi \in \Pi} \max_{\psi \in \Psi} \left[ \ell_t(x_t, u_t, v_t) + J^*_{t+1} \left( f_t(x_t, u_t, v_t) \right) \right]
\]

(1)

A. Quantifying a policy’s robustness

Suppose that the nominal policy, \(\pi\), of an agent has been found, 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,

\[
x_{t+1} = f_t(x_t, u_t, v_t), \quad u_t \sim \pi_t = \tilde{f}(x_t, v_t), \quad t = 0, ..., T - 1.
\]

(2)

For stage costs of the form,

\[
\ell_t(x_t, u_t, v_t) = c_t(x_t, u_t) - \gamma g_t(v_t),
\]

where \(c_t(x_t, u_t)\) 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

\[
\max_{\psi \in \Psi} E_{u_t \sim \pi_t} \sum_{t=0}^{T} c(x_t, u_t) - \gamma g(v_t) = \max_{\psi \in \Psi} E \sum_{t=0}^{T} \ell_t(x_t, v_t).
\]

Varying \(\gamma\) 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 \(\gamma\) decreases, the adversary incurs lower penalties, causing a larger system disturbance. The (inverse of the) smallest \(\gamma\)-value for which the adversary causes unacceptable performance degradation (e.g., instability) provides a measure of robustness of the nominal agent’s policy \(\pi\). The parameter \(\gamma\) is a distinguishing feature of our work; \(\gamma\) quantifies the \(H_\infty\) 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 \(H_\infty\) norm can be explicitly computed via generalized Riccati equations or semidefinite programs [10], [22]. In systems with nonlinear dynamics, one can use differential dynamic programming (DDP) [8] or iterative LQG [23] to optimize the adversary against the closed-loop system under the nominal agent’s policy \(\pi\). Indeed, Morimoto (in [24]) applied minimax DDP to a walking biped robot, but we noticed errors in the value function recursions\(^2\).

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.

A performance-robustness tradeoff curve can be computed by optimizing the adversary for various values of \(\gamma\). A prototypical curve is illustrated in Fig. 1. This curve quantifies the robustness of a fixed policy \(\pi\), defining an upper bound \(V_\gamma^*\) on the cost function that does not yield acceptable performance: this is the critical value of \(\gamma\) to be identified. This \(\gamma^*\)-value determines how much adversarial effort we need to suffer performance degradation; large \(\gamma^*\) values will give poor robustness of \(\pi\) – even with little adversarial effort. The algorithm for optimizing adversaries and generating the trade-off curve is summarized in Algorithm 1. The adversary’s policy may not be globally optimal due to the approximation of policies in continuous spaces and because the the algorithm may only converge to a locally optimal solution, we obtain...
Algorithm 1 Robustness Curve via iLQG

1: for $\gamma \in \text{range}(\gamma_{\min}, \gamma_{\max}, N)$ do
2:     for $i \in \{1, \ldots, I\}$ do
3:         Initialize $x_0 = x(0)$
4:         $p_i \leftarrow \arg \min_{p_i} E_{p_i} \left[ \sum_{t=0}^T \ell_t(x_t, v_t) \right]$
5:     end for
6: end for

an upper bound on the robustness of the policy $\pi$.

B. Achieving Robustness via IDG

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 [23], 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, $\{u_i\}$, and nominal adversarial agent’s local controls $\{v_i\}$ 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 $\{\bar{u}_i\}, \{\bar{v}_i\}$ to generate a nominal state trajectory $\{\bar{x}_i\}$, with neighboring trajectories $\{x_i\}$
- we choose a small neighborhood, $\{\delta x_i\}$ of $\{x_i\}$, which provides an optimal reduction in cost as the dynamics no longer represent those of $\{x_i\}$
- discretizing time, the new state and control sequence pairs become $\delta x_i = x_i - \bar{x}_i$, $\delta u_i = u_i - \bar{u}_i$, $\delta v_i = v_i - \bar{v}_i$.

Setting $V(x, T) = \ell_T(x_T)$, 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

$V(x_t) = \max_{u_t \sim \pi, v_t \sim \psi}[\ell(x_t, u_t, v_t) + V(f(x_{t+1}, u_{t+1}, v_{t+1}))].$

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

$Q_t = \ell(x_t, u_t, v_t, t) + V(x_{t+1}, t + 1).$

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

$Q(\cdot) \approx \frac{1}{2} \begin{bmatrix} 1 & \delta x_t^T & \delta u_t^T & \delta v_t^T \end{bmatrix} \begin{bmatrix} 1 & Q_{xx} & Q_{xu} & Q_{vx} \\ Q_{xu} & Q_{uu} & Q_{uv} & Q_{vz} \\ Q_{vx} & Q_{uv} & Q_{vv} & 0 \\ 0 & 0 & 0 & Q_{vvv} \end{bmatrix} \begin{bmatrix} 1 & \delta x_t^T & \delta u_t^T & \delta v_t^T \end{bmatrix},$  

where,

$Q_{xt} = \ell_{xt} + f_{xT}^T V_{xT+1}, \quad Q_{ut} = \ell_{ut} + f_{uT}^T V_{uT+1},$

$Q_{vt} = \ell_{vt} + f_{vT}^T V_{vT+1}, \quad Q_{xx} = \ell_{xx} + f_{xt}^T V_{xT+1} f_{xT}^T,$

$Q_{ux} = \ell_{ux} + f_{ux}^T V_{xT+1} f_{uT}^T, \quad Q_{vx} = \ell_{vx} + f_{vx}^T V_{xT+1} f_{vT}^T,$

$Q_{uv} = \ell_{uv} + f_{uv}^T V_{xT+1} f_{vT}^T,$

$Q_{vu} = \ell_{vu} + f_{vu}^T V_{vT+1} f_{uT}^T.$

Note that this is consistent with linearized methods, where linearized second moment terms will dominate the higher order terms [25]. The LQR approximation to the state and the optimal control performance index become,

$\delta x_{t+1} \approx f_{xT}^T \delta x_t + f_{uT}^T \delta u_t + f_{vT}^T \delta v_t,$

$\ell(x_t, u_t, v_t) \approx \frac{1}{2} \begin{bmatrix} 1 & \delta x_t^T & \delta u_t^T & \delta v_t^T \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Q_{uu} & Q_{uv} & 0 \\ 0 & Q_{vu} & Q_{vv} & 0 \\ 0 & 0 & 0 & Q_{vvv} \end{bmatrix} \begin{bmatrix} 1 & \delta x_t^T & \delta u_t^T & \delta v_t^T \end{bmatrix},$

where single and double subscripts denote first and second-order derivatives. 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

$\delta u^*_t = -Q_{uu}^{-1} \begin{bmatrix} Q_{uT}^T + Q_{ux} \delta x_k + Q_{uv} \delta v_t \end{bmatrix},$

$\delta v^*_t = -Q_{vv}^{-1} \begin{bmatrix} Q_{vT}^T + Q_{vx} \delta x_k + Q_{vu} \delta u_t \end{bmatrix}.$

Note that the control strategies in (5) depend on the action of the other player. Say, 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,

$K_{at} = \left[ (I - Q_{uu}^{-1} Q_{uv} Q_{vT}^{-1} Q_{uv}) Q_{uu}^{-1} \right]^{-1},$

$K_{vt} = \left[ (I - Q_{vv}^{-1} Q_{vu} Q_{vT}^{-1} Q_{vu}) Q_{vv}^{-1} \right]^{-1},$

Note that $\delta x_k$, $\delta u_k$, and $\delta v_k$ are measured w.r.t the nominal vectors $x_k$, $u_k$, $v_k$ and are not necessarily small.
These recursive value functions, essentially differentiate our ˜\(Q_{uu}\) and \(f_{arising from the control/disturbance transition matrices \(f\) the regularization can have adverse effects on the system \([\text{non-positive-definite}, we can add a suitably large positive \(C. Improved Regularization\) recursions \(value coefficient recursion equations from Morimoto’s DDP\) are \(\Delta V_{t} = g_{u}(\hat{Q}_{uu}g_{u} + g_{u}Q_{uv}g_{vt}) + \frac{1}{2} (g_{u}Q_{uu}g_{u} + g_{u}Q_{uv}g_{vt})\)
\(V_{xt} = Q_{xt} + G_{u}g_{vt} + G_{u}g_{vt} + G_{u}Q_{vu}g_{u} + g_{u}Q_{xt} + g_{v}Q_{vx}g_{vt}\)
\(V_{xx} = \frac{1}{2}(Q_{xx} + G_{u}Q_{uu}g_{u} + G_{v}Q_{vv}g_{v} + G_{u}Q_{vu}g_{u} + G_{u}Q_{vv}g_{vt})\)
Equation (6) give the open-loop and closed-loop components of the control equations for both agents. Comparing coefficients in (4), we find that the value function coefficients are
\[
\Delta V_{t} = g_{u}(\hat{Q}_{uu}g_{u} + g_{u}Q_{uv}g_{vt}) + \frac{1}{2} (g_{u}Q_{uu}g_{u} + g_{u}Q_{uv}g_{vt})
\]

\[
V_{xt} = Q_{xt} + G_{u}g_{vt} + G_{u}g_{vt} + G_{u}Q_{vu}g_{u} + g_{u}Q_{xt} + g_{v}Q_{vx}g_{vt}
\]

\[
V_{xx} = \frac{1}{2}(Q_{xx} + G_{u}Q_{uu}g_{u} + G_{v}Q_{vv}g_{v} + G_{u}Q_{vu}g_{u} + G_{u}Q_{vv}g_{vt})
\]

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

C. Improved Regularization

For nonlinear systems, the inverse of the Hessian must be strictly positive definite. When the inverse of the Hessian is non-positive-definite, we can add a suitably large positive quantity to it [26], [27], or replace the elements of the diagonal matrix in its eigen decomposition, \([V, D] = \text{eig}(Q)\), that are smaller than an adequately small \(\rho\), and then set \(Q = VDV^{T}\) [9]. In this work, \(\rho\) is added to \(Q\) when the Hessian is not well-posed. Our update rule is \(Q_{uu} = Q_{uu} + \rho I_{m}\). However, the regularization can have adverse effects on the system arising from the control/disturbance transition matrices \(f_{u}\) and \(f_{v}\); therefore, we introduce a similar penalty term used in [23] to deviations from states so that the regularization yields a quadratic state cost about the previous policy:

\[
Q_{uu} = \ell_{uu}f_{u}(V_{xt} + \rho I_{n})f_{x} + V_{xt}f_{uu}
\]

\[
Q_{uv} = \ell_{uv}f_{v}(V_{xt} + \rho I_{n})f_{x} + V_{xt}f_{vu}
\]

\[
Q_{ux} = \ell_{ux}f_{u}(V_{xt} + \rho I_{n})f_{x} + V_{xt}f_{ux}
\]

\[
Q_{vx} = \ell_{vx}f_{v}(V_{xt} + \rho I_{n})f_{x} + V_{xt}f_{vx}
\]

The adjusted gains therefore become

\[
K_{ut} = \left[\left(I - \hat{Q}_{uu}g_{u}Q_{uv}g_{vt}\right)\hat{Q}_{uu}\right]^{-1}
\]

\[
K_{vt} = \left[\left(I - \hat{Q}_{uv}Q_{uu}g_{u}Q_{uv}g_{vt}\right)\hat{Q}_{uv}\right]^{-1}
\]

\[
g_{u}(\hat{Q}_{uu}g_{u} + g_{u}Q_{uv}g_{vt}) + \frac{1}{2} (g_{u}Q_{uu}g_{u} + g_{u}Q_{uv}g_{vt})
\]

\[
V_{xt} = Q_{xt} + G_{u}g_{vt} + G_{u}g_{vt} + G_{u}Q_{vu}g_{u} + g_{u}Q_{xt} + g_{v}Q_{vx}g_{vt}
\]

\[
V_{xx} = \frac{1}{2}(Q_{xx} + G_{u}Q_{uu}g_{u} + G_{v}Q_{vv}g_{v} + G_{u}Q_{vu}g_{u} + G_{u}Q_{vv}g_{vt})
\]

\[
V_{xx} = \frac{1}{2}(Q_{xx} + G_{u}Q_{uu}g_{u} + G_{v}Q_{vv}g_{v} + G_{u}Q_{vu}g_{u} + G_{u}Q_{vv}g_{vt})
\]

D. Regularization Schedule

To accurately tweak the regularization term in \(\rho\), we adopt a regularization schedule that penalizes \(Q_{uu}\) or \(Q_{vv}\), when the backward pass fails. When the backward pass is successful, we would want \(\rho\) to decrease very rapidly so as to ensure fast convergence; otherwise, we would want to increase \(\rho\) as fast as possible, but not to a very large value as the minimum value of \(\rho\) that prevents divergence is linear order. We let \(\rho_{0}\) denote some minimal monification value for \(\rho\) (set to 1.0), and we adjust \(\rho\) as follows:

\[
\text{increase } \rho:\quad \rho \leftarrow 1.1 \rho_{0}\quad \rho_{0} = \rho
\]

\[
\text{reduce } \rho:\quad \rho \leftarrow 0.9 \rho_{0}\quad \rho_{0} = \rho
\]

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 \(\theta\)-rotation about the \(z\)-axis (see Fig. 2), 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, \(x_{t} = [x_{1}, x_{2}, x_{3}]^{T} \equiv [x_{1}, y_{1}, \theta_{1}]^{T}\), and the local frame of the robot is defined as \(x_{R} = [x_{R}, y_{R}, \theta_{R}]^{T}\) (see Fig. 2).

The torques that govern the robot’s motion are obtained from [28]. We run our experiments in the Gazebo physics engine [29], 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 2 and 3. Formally, we define the generalized Lagrangian equation of the robot as,

\[
M(x)\ddot{x} + C(x, \dot{x})\dot{x} + B^{T}Sf = \frac{1}{r}B^{T}\tau
\]

\[^{4}\text{https://goo.gl/CYTjvD}\]
where \( \tau = [\tau_1, \tau_2, \tau_3, \tau_4] \) is the wheel torque vector, \( r \) is the wheel radius, \( f = [f_1, f_2, f_3, f_4]^T \) is the friction vector, and \( S \) and \( B \) map the inverse kinematics, gravity, external forces and robot’s angle, \( \theta \), to each wheel torque; matrices \( M \) and \( C \) denote the inertia and coriolis properties of the robot. \( B \) and \( S \) are given by,

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

\[
S = \text{diag} [\text{sgn}(\dot{\phi}_1), \text{sgn}(\dot{\phi}_2), \text{sgn}(\dot{\phi}_3), \text{sgn}(\dot{\phi}_4)]
\]

\( \zeta = \pi/4 - \alpha \), \( l \) is the mounting distance of the wheels as shown in Fig. 2, 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,[30]

\[
F_i = \sum_{j=1}^{4} (\tau_j - r \text{sgn}(\dot{\phi}_j) f_j) \frac{\partial \dot{\phi}_j}{\partial x_i}, \; i = \{1, 2, 3\} \quad (11)
\]

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. 4a to within a 0.1m radius of the orange box attached to the right-hand corner of Fig. 4b. We separate the state and control cost terms for easier manipulation of cost components. For the state’s instantaneous cost, we define a pseudo “smooth-abs” function,

\[
\ell(x_t) = \sqrt{\alpha + (x_t - x^*)^T \text{diag}(w_x) (x_t - x^*)}, \quad (12)
\]

where \( x^* \) denotes the desired state, \( w_x \) is a state penalty vector and \( \alpha \) is a constant that controls the curvature of the cost function: the lower the \( \alpha \) value, the smoother is the robot’s trajectory near the goal state. The \( (x - x^*) \) 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 12 is particularly useful because it encourages the relative weighting of state-cost terms along different axes of motions. Inspired by [23], we choose a hyperbolic cosine control cost function instead of a quadratic cost defined as:

\[
\ell(u_t) = \alpha^2 (\cosh(w_u^T u_t) - 1), \quad (13)
\]

where \( w_u \) is a control penalty vector. The final cost is non-dependent on the control signal. This cost function limits control outputs to an \( \alpha \)-sized neighborhood in \( u \)-space. \( \alpha \) is set to \( 10^{-4} \), \( w_u = [1, 1, 1, 1] \) and \( w_x = [1, 1, 0.8] \). 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 \( \{\hat{u}_t\} \), (initialized to \([1.3, 0.8, 0.1]\), we generate the nominal states \( \{\hat{x}_t\} \)
- replacing \( \tau \) with \( u \) in (10), we compute the forward dynamics

\[
\dot{x} = -M^{-1}C\dot{x} - M^{-1}B^T (Sf - \frac{1}{r}u) \quad (14)
\]

- we then obtain derivatives of \( f \) and those of \( \ell \) from (4)
- starting at time \( t = T - 1 \), we compute the associated nominal Q coefficients in (4), obtain the open and closed-loop gains, \( g_u, G_u \), 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

\[
\hat{x}(1) = x(1) \\
\hat{u}(t) = u(t) + \varsigma g_u(t) + G_u(t)(\hat{x}(t) - x(t)) \\
\hat{x}(t + 1) = f(\hat{x}(t), \hat{u}(t)) \quad (15)
\]
• the backward/forward pass informs of the change in cost \(\Delta J^v\), which is compared to the estimated reduction

\[
\eta = \frac{J(u_1, \ldots, T) - J(\bar{u}_1, \ldots, T)}{\Delta(J(\rho))}
\]

where

\[
\Delta(J(\rho)) = \rho \sum_{t=1}^{T-1} g_{at}^T Q_{at} + \frac{\rho^2}{2} \sum_{i=1}^{T-1} g_{at}^T Q_{uat} g_{at},
\]

• we accept a trajectory only if \(0 < c < \eta\) (where we have chosen \(c = 0.5\)) following [8]'s recommendation.

• set \(\bar{x}_t = x_t (t = 1, \ldots, T)\), \(u_t = u_t\), reset \(\rho\) and repeat the forward pass

• stop when \(\Delta(J(\rho)) < \chi\) (where \(\chi\) 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 \(v\) from a Gaussian distribution \(\sim N(0, 2I)\) and augment the stage control cost of \((13)\) as follows

\[
\ell(u_t, v_t) = \alpha^2 (cosh(w^T u_t) - \gamma cosh(w^T v_t)).
\]

\(\gamma cosh(w^T v_t)\) introduces a weighting term in the disturbing input, with \(\gamma\) 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 \(w\) to \([1, 1, 1, 1]\) and run the two player, zero-sum gameChar strike. Again, we compute the derivatives \(f_{u_t}\) and \(f_{v_t}\), calculate the associated derivatives in \((4)\) and give the optimized dG torque to the robot. We initialized the nominal disturbance vector \(v\) as a multivariate Gaussian-filtered, random noise vector \(\sim N(0, 2I)\). \(v_t\) is similarly initialized before we run the IDG algorithm. Our IDG process goes thus:

• starting with an open-loop control and disturbance schedules \(\{\tilde{u}_t\}, \{\tilde{v}_t\}\), we generate the nominal states \(\{\tilde{x}_t\}\)

• we then obtain the derivatives \(f_{\tilde{u}_t}, f_{\tilde{v}_t}, f_{u_t}, f_{v_t}\) and those of \(\ell\) 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

\[
\begin{align*}
\tilde{x}(1) &= x(1) \\
\tilde{u}(t) &= u(t) + \varsigma g_{at}(t) + g_a(t)(\tilde{x}(t) - x(t)) \\
\tilde{v}(t) &= v(t) + \varsigma g_{vt}(t) + g_v(t)(\tilde{x}(t) - x(t)) \\
\tilde{x}(t+1) &= f(\tilde{x}(t), \tilde{u}(t), \tilde{v}(t))
\end{align*}
\]

• the backward/forward pass informs of the change in cost \(\Delta J^v\), which is compared to the estimated reduction

\[
\eta = \frac{J(u_1, \ldots, T) - J(\bar{u}_1, \ldots, T)}{\Delta(J(\rho))},
\]

\[
\Delta(J(\rho)) = \rho \sum_{t=1}^{T-1} g_{at}^T Q_{at} + \frac{\rho^2}{2} \sum_{i=1}^{T-1} g_{at}^T Q_{uat} g_{at},
\]

• we again accept a trajectory only if \(0 < c < \eta\) (choosing \(c = 0.5\))

• and set \(\bar{x}_t = x_t (t = 1, \ldots, T)\), \(u_t = u_t, v_t = v_t\), reset \(\rho\) and repeat the forward pass

• stop when \(\Delta(J(\rho)) < \chi\) (where \(\chi\) is a problem dependent term).

**IV. RESULTS**

We implement the policy sensitivity analysis using the guided policy search algorithm of [16]. We then run the IDG optimization algorithm on the yobot and analyze the effectiveness of using our method against standard ILQG.

**A. Robustness Margins**

We first generate data used to train a global neural network policy by running our ILQG optimization scheme for the local controllers. We then run an adversarial local controller in closed-loop with the nominal agent’s controller for various disturbance values, \(\gamma\), in a supervised learning of global neural network policies. We task the PR2 robot (shown in Fig. 5 and simulated in the MuJoCo physics simulator [31]), to insert a peg into the hole at the bottom of the slot. The model was learned with a mixture of \(N\) Gaussians and we parameterized the local control laws with the deep neural network described in [16]. The difficulty of this task stems from the discontinuity in dynamics from the contact between the peg and the slot’s surface. The local policies for the controller and adversary were initialized in the neighborhood of the initial state \(x_0\) using a proportional-derivative (PD) control law for the inverted pendulum task. We choose quadratic stage costs for both the state and the two controllers

\[
\ell(x_t, u_t, v_t) = w_u u_t^T u_t + w_v \|d(x_t) - d^*\|^2 - \gamma v_t^T v_t
\]

where \(d(x_t)\) denotes the end effector position at state \(x_t\) and \(d^*\) denotes the desired end effector position at the base of
the slot. The cost function quantifies the squared distance of the end effector to its desired position and energy of the controller and adversary torque inputs. We set $w_u$ and $w_x$ to $10^{-6}$ and 1, respectively. For various $\gamma$-values, we evaluate the robustness of the trained policy by training the adversary with the GPS algorithm, and observe its effect on the task performance. We run each experiment for 11 GPS iterations. Fig. 6 shows that the adversary causes a sharp degradation in the controller performance for values of $\gamma < 1.5$. This corresponds to when the optimized policy is destabilized and the arm fails to reach the desired target. As values of $\gamma$ increase above 1.5, however, we observe that the adversary has a reduced effect on the given task. Video of this result is on the robustness margins link on our website.

B. 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.

C. Trajectory Optimization: iDG

Here, we set $T$ to 150, initiate the disturbance from a multivariate Gaussian filtered noise $N(0, 2)$, set the gains for the torques to $\{k_F = 10, k_F = 15, k_F = 1\}$ respectively and run the algorithm described in III-B for various $\gamma$-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 $\gamma$ in the range $\{10, 5, 3, 1.5, 0.65, 0.1, 0.01, 10^{-5}, 2 \times 10^{-3}, 2 \times 10^{-5}\}$ for both experiments. The result showing the trajectory evolution and the position of the youbot after each iDG run for various $\gamma$ 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 $\gamma$ 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 video. This is despite that 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 getting 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 $\gamma$ becomes very small (below $10^{-5}$), the robot’s trajectory is undefined, and it converges to a spurious minimum as both experiments show. This $\gamma$ indices would correspond to the $J^*_\gamma$, that depicts unacceptable performance in Fig. 1.

The downside to the iDG approach is the longer time requirement to finish the trajectory optimization process. 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 achieving an optimal control task. Compared to Morimoto’s work [24], our minimax iDG does not have to learn the unmodeled disturbance with reinforcement learning as [24] had to do for their walking biped. We thus identify the critical value of $\gamma$ 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 $\gamma$ values in this range in order to design a 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 [2], [3]. If we can provide a measure of the disturbance that a policy can withstand, we would be aware of the limits of such policy aforesaid. 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 uncertainties, 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.
State convergence of the KUKA robot with our iDG formulation given different goal states and varying $\gamma$-values

REFERENCES

[34] H. Kelley, AIAA Astrodynamics Specialists Conference, Yale University, August 1963.