Safe Interactions via Monte Carlo Linear-Quadratic Games

BENJAMIN A. CHRISTIE and DYLAN P. LOSEY, Virginia Tech, USA

Safety is critical during human-robot interaction. But — because people are inherently unpredictable — it is often difficult for robots to plan safe behaviors. Instead of relying on our ability to anticipate humans, here we identify robot policies that are robust to unexpected human decisions. We achieve this by formulating human-robot interaction as a zero-sum game, where (in the worst case) the human's actions directly conflict with the robot's objective. Solving for the upper value of this game provides robot policies that combine safety and performance across a wide range of human actions. Existing approaches attempt to find these optimal policies by leveraging Hamilton-Jacobi analysis (which is often intractable) or linear-quadratic approximations (which are often inexact). By contrast, in this work we propose a computationally efficient and theoretically supported method that approaches a robust, Stackelberg-style policy. Our approach (which we call *MCLQ*) leverages linear-quadratic games to obtain an initial guess at safe robot behavior, and then iteratively refines that guess with a Monte Carlo search. Not only does MCLQ provide real-time safety adjustments, but it also enables the designer to tune how conservative the robot is — preventing the system from focusing on unrealistic human behaviors. Our simulations and user study suggest that this approach advances safety for human-robot interaction in terms of both computation time and expected performance.

CCS Concepts: • **Computing methodologies** → *Robotic planning*; **Planning under uncertainty**.

Additional Key Words and Phrases: Human-Robot Interaction, Safety

1 INTRODUCTION

Interacting with people is challenging because humans are inherently unpredictable. Consider Figure 1 where an autonomous drone is flying near a human worker. This robot has some high-level task it wants to complete (e.g., environment monitoring), as well as a low-level controller which dictates how the robot should accomplish this task (e.g., circling the room). If the robot knew precisely what the human was going to do, it could anticipate the human's actions and choose behaviors that maintain a safe distance between agents. But because people often take unexpected actions, real human behavior will deviate from the robot's model. As a result, the robot's original plan — which it thought was safe — may actually be unsafe. Returning to our example, a drone that turns left (because it predicts the human will stop) actually puts both agents in danger (because the human unexpectedly keeps walking forwards).

To address this problem, today's robots recognize that they are often uncertain about the human's actions. Instead of assuming the human will follow an exact model, these robots search for plans that are safe across a distribution of human behaviors. There are two general approaches here. The first is a *precise* method: the robot reasons over the distribution of possible human trajectories, and then chooses the optimal control policy that maximizes safety across the distribution [5, 11, 15]. The second approach is based on *approximations*: the robot simplifies the system dynamics and objectives, and then obtains a closed-form policy that maximizes safety under the simplified setting [13, 33]. Unfortunately, both types of approaches have significant limitations. Theoretically exact solutions are often computationally intractable; e.g., our autonomous vehicle could not apply these methods in real-time to adjust its behaviors. On the other hand, approximate solutions are imprecise, and may result in unsafe behaviors that are *locally* optimal but globally unsafe as the system becomes increasingly complex and nonlinear.

This work is supported in part by NSF Grant #2129201.
Authors' address: Benjamin A. Christie, benc00@vt.edu; Dylan P. Losey, losey@vt.edu, Virginia Tech, Department of Mechanical Engineering, 635 Prices Fork Rd, Blacksburg, VA, 24060, USA.

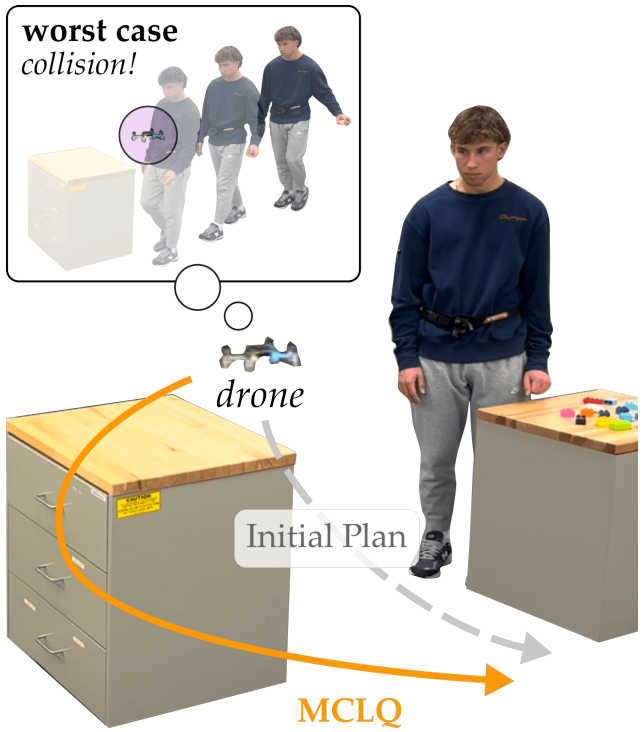

Fig. 1. Human and drone moving in a shared workspace. Under our proposed method (MCLQ), the drone reasons about worst case human actions within designer-specified bounds, and then selects safe behaviors in response to those actions. For instance, here the drone moves across the table to prevent a potential collision.

In this paper we propose an action-augmentation method for human-robot interaction that robots can leverage to tune their real-time behavior and enhance safety around unpredictable humans. Our proposed method achieves precise and tractable performance by combining both types of prior approaches. Specifically, our insight is that:

We can leverage approximate methods to seed safe robot behavior, and then apply a stochastic local search to guide that initial guess towards optimal solutions.

We show an example of our method in Figure 1. The robot approximates the interaction as a linear-quadratic (LQ) system, and finds an action sequence that maximizes robot performance under the worst case human response. In parallel, the robot refines this action sequence using sampling methods that noisily converge towards the open-loop optima of a zero-sum game while stochastically escaping local optima. Robots that execute the resulting actions are robust to unpredictable humans within designer-specified bounds: even if the person suddenly changes direction, the autonomous drone has planned a safe trajectory.

Overall, we make the following contributions:

Identifying Robust Behavior. To maintain safety despite unpredictable humans, we frame interaction as a zero-sum game. Here the robot tries to minimize its cost while realizing an adversarial human might take actions to maximize that cost. The game's upper value defines a robot policy that is robust to unexpected human behavior; we develop a method (MCLQ) to tractably approximate this robust policy.

Uniting Prior Works. Our approach to find robust robot policies blends aspects of approximate and precise methods. Specifically, we leverage LQ games to get an initial action sequence, and then refine that sequence with Metropolis-Hastings sampling. Our analysis proves that in the best case MCLQ converges to the Nash Equilibrium, and in the worst case our proposed approach is as good as current approximations.

Adjusting Safety Margins. Our MCLQ framework enables practical advantages that go beyond existing methods. For example, designers can constrain the safety margin, i.e., the range of human actions the robot reasons over. Decreasing this safety margin causes the robot to rely more on its human predictions, preventing the robot from focusing on unrealistic human behaviors and becoming overly conservative.

Conducting Experiments. We conduct simulations and a user study to compare our approach to state-of-the-art methods. In simulation, we show that MCLQ solves for safe control policies more rapidly than HJB and iLQ methods, while also achieving higher performance than existing iLQ approximations. In the real world, participants reported feeling "safer" when interacting with drones that leverage MCLQ compared to alternatives, due to MCLQ producing more "predictable" and "attentive" behavior. MCLQ is able to complete tasks more quickly and safely than iLQ alternatives.

2 RELATED WORK

Related research has introduced a variety of control algorithms to ensure safety during human-robot interaction [15]. Within this larger field, our approach is most connected to game-theoretic controllers. These methods frame the human and robot as two agents with their own objectives, and solve for robot policies that are robust (i.e., minimize worst-case costs) while accounting for the human's decisions.

Exact Methods. Game-theoretic works treat human-robot interaction as interconnected system: both agents have a task they are trying to accomplish, and the actions of each agent affect one another. We can formalize these interconnected dynamical games with Hamilton-Jacobi equations [7, 36]. Precisely solving the Hamilton-Jacobi equations provides the optimal, game-theoretic behavior for each agent; e.g., a policy for the autonomous car that avoids collisions with the human. Hamilton-Jacobi analysis has therefore become a gold standard for safe interaction [3–6, 12]. Unfortunately, most Hamilton-Jacobi equations have no analytical solution, and numerical methods are often intractable — requiring both exponential time and computational memory [13, 32]. As such, within this paper we develop a real-time approximation that converges towards the ideal solution provided by the Hamilton-Jacobi framework.

Linear-Quadratic Approximations. We are not the first to try to approximate the game-theoretic solution. For example, [6, 11, 19, 24, 25] propose neural network models, and [9, 13, 20] outline open-loop methods. One particularly promising approximation recognizes that we can analytically solve the Hamilton-Jacobi equations in a special case: if the system dynamics are linear and the costs are quadratic [7]. For these linear-quadratic (LQ) games the optimal robot policy is closely connected to linear-quadratic regulators [1, 38]. Robots can solve LQ games quickly, providing a tractable method to obtain safe behaviors that match the gold standard of Hamilton-Jacobi analysis. Indeed — even if the system is not linear-quadratic — recent works have tried to apply sequential linear-quadratic (iLQ) approximations [13, 23]. Under these iLQ approaches the robot iteratively linearizes the dynamics and quadraticizes the cost before solving the resulting LQ game to find the human and robot control gains. On the one hand, iLQ provides an increasingly prevalent method to tractably identify safe controllers. But on the other hand, this approximation is fundamentally limited by its reliance on linear-quadratic systems — as the real interaction diverges from the

simplification, iLQ falls short. Overall, this gap motivates our work towards methods that combine theoretical exactness and practical implementation.

Accounting for the Human. Using the game-theoretic solution leads to policies that are robust to worst-case disturbances. However, doing so can make robot policies overly conservative — even when the human is not interfering with the robot's objective. Recent works [13, 16, 17, 26, 29, 33] have attempted to mitigate this issue by using a model of the human that updates over the course of an interaction. The authors of [33] maintain an estimate of the current user's *role* in a Stackleberg-style formulation. If they are a follower, then the robot must be more cautious, whereas if they are a leader, then the robot can find calculate an optimal response. Unfortunately, this approach is limited to settings where a backwards-reachable tube (representing the agents' actions) can be calculated in real-time, such as in linear-quadratic games. Likewise in [16], a shielding-aware dual control method is proposed that has theoretical guarantees on robot safety while maintaining an estimate of the uncertain agent's parameters. This sampling-based approach is similar to our proposed approach, but it is limited to control-affine dynamics and Boltzmann-rational human models. Other works such as [10, 21, 26–28] leverage learned models that adapt to the human and predict their future intent. Although these approaches effectively optimize for performance in the *average* interaction, they are not robust to *worst-case* actions that the human may select. In this work we propose a flexible, real-time approach that accounts for worst-case human actions and determines robot responses which are robust to these actions within designer specified bounds.

3 PROBLEM STATEMENT

We are interested in interactions between a single human and a single robot. The robot has a task to perform, and to complete this task the robot must reason over the human (e.g., avoid colliding with the human worker). Without loss of generality, we assume the robot has an initial control policy for the task, as well as a nominal model for predicting the human's behavior. Our approach will adjust the robot's policy to enhance safety even when the human deviates from the nominal model. To achieve this safety, we consider the worst-case interaction (within designer-specified bounds): i.e., we identify which actions the human can take that will have the worst impact on the robot's performance. Below we formalize this problem setting.

Zero-Sum Game. Let the system have state $x \in \mathcal{X}$ (e.g., the position of both agents). The robot takes action $u \in \mathcal{U}$, and the human takes action $w \in \mathcal{W}$ (e.g., the human and robot velocities). At each timestep t, the system state transitions according to the deterministic, discrete-time dynamics:

$$x^{t+1} = f(x^t, u^t, w^t) \tag{1}$$

An interaction lasts for a total of T timesteps. The system begins the interaction in state x^0 and during the interaction it follows a trajectory $\xi = \{(x, u, w)^0, \cdots, (x, u, w)^{T-1}, x^T\}$. Since the dynamics are deterministic, we can abbreviate this trajectory as a sequence of robot and human actions $\xi(x^0) = \{(u, w)^0, \cdots, (u, w)^{T-1}\} = (u^{0:T}, w^{0:T})$.

The robot's goal is to select actions that will cause the system state to update in a way that completes the robot's task. More formally, the robot has a cost function it seeks to *minimize* across the interaction:

$$\begin{aligned} J\left(x^t, u^{t:T}, w^{t:T}\right) &= \sum_{\tau=t}^{T-1} j(x^\tau, u^\tau, w^\tau) + D(x^T) \\ \text{s.t. } x^{\tau+1} &= f\left(x^\tau, u^\tau, w^\tau\right) \end{aligned} \tag{2}$$

Here J is the cumulative cost, $j(x, u, w)$ is the cost at a single timestep, and $D(x^T)$ is the bequest state cost. We emphasize that the robot's cost in Equation (2) depends on the human's actions w.

For example, the autonomous drone will incur a significant penalty if the human moves into that agent. To optimize cost, the robot has an initial policy $\hat{\pi}_{\mathcal{R}}$ that determines how it will complete the task. The robot may also have some guess for how the human agent will behave: $\hat{\pi}_{\mathcal{H}}$. Both policies are mappings from states x to actions:

$$\hat{\pi}_{\mathcal{R}} : \mathcal{X} \mapsto \mathcal{U} \tag{3}$$

$$\hat{\pi}_{\mathcal{H}} : \mathcal{X} \mapsto \mathcal{W} \tag{4}$$

In practice, the real human will inevitably deviate from the robot's model. A worst-case human selects actions that *maximize* the cost in Equation (2). Formally, in this worst-case the human and robot are participating in a two-player *zero-sum* game [7]: the robot is trying to minimize its cost, whereas the antagonistic human is attempting to maximize that same cost, directly opposing the robot's objective. One benefit of focusing on the worst-case human is that — if the robot's behavior is safe in this worst case — we know it is safe for other human responses.

In this work, we assume that the cost function J and state transition function f are sufficiently smooth and have continuous first- and second-derivatives with respect to x. We further assume that there exists a norm $\|\circ\|$ such that $\|\nabla_x f(x,u,w)\| \leq 1$, i.e. the system state is controllable and does not grow unbounded for finite actions. Additionally, we assume that the action sets $\mathcal{U}$ and $\mathcal{W}$ are compact, bounded, and convex. In practice, this assumption is naturally satisfied by the limits of robotic constraints and human biomechanics. This assumption is not required by our proposed method, but it is necessary for the following robust minimax formulation.

Robust Minimax Formulation and Game Values. Similar to prior works [31, 33, 37], we have formulated human-robot interaction as a zero-sum game. The advantage of this formulation is that identifying robot behavior that is safe in the worst-case ensures safety regardless of the human's actual deviation. To determine what this worst-case behavior is, we turn to the bounds of optimal behavior in zero-sum games. In general, zero-sum games are characterized by their upper and lower values. The lower value of the game represents a scenario where the human must commit to an action sequence first, allowing the robot to optimally respond:

$$\underline{V}(x) = \sup_{w^{0:T} \in \mathcal{W}^T} \inf_{u^{0:T} \in \mathcal{U}^T} \left(J(x, u^{0:T}, w^{0:T}) \right) \tag{5}$$

In this situation, the robot is at an advantage. Note that the inner inf operator is evaluated for a specific, fixed human action $w^{0:T}$. Since the robot can see what the human's action is, it can always choose the best response that minimizes the overall cost J. Conversely, the upper value of the game represents the scenario where the robot commits to a trajectory, and the adversarial human responds with the worst-case optimal sequence:

$$\overline{V}(x) = \inf_{u^{0:T} \in \mathcal{U}^T} \sup_{w^{0:T} \in \mathcal{W}^T} \left(J(x, u^{0:T}, w^{0:T}) \right) \tag{6}$$

Here, the robot is at a disadvantage. It cannot observe the human's action: instead, it must plan an action that minimizes the overall cost, regardless of what the human chooses. Since the human — wanting to maximize the robot's cost — can observe the robot's action, they can always choose an action that leads to worst-case performance. A pure-strategy equilibrium point between these two values only exists if the Isaacs' Condition holds (for details, see Sion's Minimax Theorem in [30] and Isaacs' Condition in [18]). However, because our interaction involves complex non-linear dynamics (such as articulated manipulators) and non-quadratic objective costs (such as logarithmic barrier functions), the cost landscape J is generally neither strictly convex in u nor strictly concave in w, which violates Isaacs' condition. Thus, we recognize that there may be an information gap between the two values where $\overline{V}(x) \geq \underline{V}(x)$.

To guarantee safety, it is strictly required that the robot prepares for the worst-case: the *upper* value of the game. By planning against $\overline{V}(x)$, the robot identifies a robust Stackelberg-style policy where it acts as the leader, committing to an open-loop plan and anticipating the worst-case best-response from the human follower. Therefore, our objective is to find the robust open-loop robot policy such that:

$$\pi_{\mathcal{R}} = \underset{u^{0:T} \in \mathcal{U}^T}{\arg\inf} \sup_{w^{0:T} \in \mathcal{W}^T} \Big(J(x^t, u^{0:T}, w^{0:T}) \Big) \tag{7}$$

In other words, the robot policy should consider the worst-case action of the human before it selects its own action. The robot policy that maximizes *robustness* follows the upper value of the game.

Intractability of the Robust Minimax. Directly solving for the robust policy $\pi_{\mathcal{R}}$ is computationally intractable for continuous, high-dimensional action spaces. While traditional approaches attempt to solve the continuous-time counterpart via the Hamilton-Jacobi-Isaacs (HJI) partial differential equations to find global feedback policies, such methods suffer heavily from the curse of dimensionality and cannot be run in real-time. Instead, other approaches aim to find a *locally* robust policy by approximating the system as linear-quadratic. The information gap that results from this approximation is theoretically 0 and an optimal robot policy can be found in closed form. Unfortunately, since the resulting lower and upper values of this approximation are not the same as the original game, the policy that is maximally robust in the approximate game may not be robust in the original system. Given that the robot has some initial policy $\hat{\pi}_{\mathcal{R}}$, our goal is to develop a real-time approach that *adjusts* the robot's policy so that it converges towards the upper value of the underlying zero-sum game. Because we cannot tractably identify this ideal policy through Equation (7), we must develop an approximation that the robot can leverage efficiently in real-time.

4 MONTE CARLO LINEAR-QUADRATIC GAMES

In this section we propose our real-time method for safe interactions based on zero-sum games. To approximate $\pi_{\mathcal{R}}$ in real-time, we propose a two-stage approach: (1) an analytical "warm-start" using a Linear-Quadratic (LQ) game approximation, and (2) a dual-loop Metropolis-Hastings search that explores the non-convex landscape to minimize the upper value $\overline{V}$. Our overall approach — Monte Carlo Linear-Quadratic Games (**MCLQ**) — combines rapid LQ solutions with a parallel local search. We theoretically demonstrate that MCLQ improves upon existing LQ baselines and converges towards a robust robot policy. We conclude with the practical advantages of our MCLQ framework, including a designer-specified safety margin that prevents the robot from becoming overly conservative (Section 4.3). An implementation of MCLQ is available here.

4.1 Obtaining an Initial Action Trajectory

Our method attempts to find the sequence of actions $u^{0:T}$ that optimize Equation (7). To get an initial estimate of these actions, we simplify the actual system into an LQ approximation. Starting with f, we linearize the dynamics:

$$f(x, u, w, t) \approx Ax(t) + Bu(t) + Dw(t) \tag{8}$$

and quadraticize the state-action cost function J:

$$J(x, u, w, t) \approx x(t)'Qx(t) + u(t)'R_u u(t) - w(t)'R_w w(t) \tag{9}$$

where $\circ'$ represents transposition and $Q, R_u, R_w \succ 0$. In this localized, quadratic regime, the cost function is strictly convex in u and strictly concave in w. Consequently, the Isaacs' condition holds locally, and a unique feedback Nash Equilibrium exists. We perform this LQ approximation about the most recently calculated trajectory. If this approximation is being performed at $t = 0$, then we sample a random inital trajectory. Note that this LQ approximation is processed for the entire

trajectory across timesteps $\hat{t} \in \{t, \dots, T+t\}$. Put another way, the LQ approximation is performed for each state-action pair in the trajectory. Under this LQ approximation the Nash Equilibrium policies exhibit linear state-feedback behavior [7]. These policies are captured by the gain matrices:

$$\begin{aligned} u(t) &= -K(t)x(t) \\ w(t) &= -L(t)x(t) \end{aligned} \tag{10}$$

Crucially, we do not use these gains for closed-loop control. Instead, we use them to generate an initial open-loop seed by rolling out the linearized dynamics. This seed provides an informed starting point, ensuring that the subsequent stochastic search begins in a region of high dynamic feasability. To find the gain matrices, we employ the discrete-time algebraic Riccati Equation (DARE). For a particular robot gain matrix K, the optimal human gain matrix L can be found in closed form by solving the DARE:

$$\begin{aligned} \boldsymbol{P}_{\boldsymbol{K},\boldsymbol{L}(\boldsymbol{K})} &= \boldsymbol{Q} + \boldsymbol{K}'\boldsymbol{R}_u\boldsymbol{K} + (\boldsymbol{A}-\boldsymbol{B}\boldsymbol{K})'\widetilde{\boldsymbol{P}}_{\boldsymbol{K},\boldsymbol{L}(\boldsymbol{K})}(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{K}) \\ \text{s.t.} \quad & \Re\left(\|\boldsymbol{P}_{\boldsymbol{K},\boldsymbol{L}(\boldsymbol{K})}\|\right) \geq 0 \\ & \Re\left(\|\boldsymbol{R}_w - \boldsymbol{D}'\boldsymbol{P}_{\boldsymbol{K},\boldsymbol{L}(\boldsymbol{K})}\boldsymbol{D}\|\right) > 0 \end{aligned} \tag{11}$$

where $\widetilde{\boldsymbol{P}}_{\boldsymbol{K},\boldsymbol{L}(\boldsymbol{K})}$ is condensed for brevity:

$$\widetilde{\boldsymbol{P}}_{\boldsymbol{K},\boldsymbol{L}(\boldsymbol{K})} \equiv \boldsymbol{P}_{\boldsymbol{K},\boldsymbol{L}(\boldsymbol{K})} + \boldsymbol{P}_{\boldsymbol{K},\boldsymbol{L}(\boldsymbol{K})}\boldsymbol{D}\cdot(\boldsymbol{R}_w - \boldsymbol{D}'\boldsymbol{P}_{\boldsymbol{K},\boldsymbol{L}(\boldsymbol{K})}\boldsymbol{D})^{-1}\boldsymbol{D}'\boldsymbol{P}_{\boldsymbol{K},\boldsymbol{L}(\boldsymbol{K})} \tag{12}$$

and the terminal condition is $\boldsymbol{P}_{\boldsymbol{K},\boldsymbol{L}(\boldsymbol{K})} = \boldsymbol{Q}$. The bolded matrices shown in Equation (11) are the compacted notation presented in [38] with appropriate padding:

$$\begin{gathered} \boldsymbol{Q} \equiv \operatorname{diag}(Q^0, \dots, Q^T), \quad \boldsymbol{R_u} \equiv \operatorname{diag}(R_u^0, \dots, R_u^{T-1}), \quad \boldsymbol{R_w} \equiv \operatorname{diag}(R_w^0, \dots, R_w^{T-1}) \\ \boldsymbol{A} \equiv \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \operatorname{diag}(A^0, \dots, A^T) & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{B} \equiv \begin{bmatrix} \boldsymbol{0} \\ \operatorname{diag}(B^0, \dots, B^{T-1}) \end{bmatrix}, \quad \boldsymbol{D} \equiv \begin{bmatrix} \boldsymbol{0} \\ \operatorname{diag}(D^0, \dots, D^{T-1}) \end{bmatrix} \\ \boldsymbol{K} \equiv \begin{bmatrix} \operatorname{diag}(K^0, \dots, K^{T-1}) & \boldsymbol{0} \end{bmatrix}, \quad \boldsymbol{L} \equiv \begin{bmatrix} \operatorname{diag}(L^0, \dots, L^{T-1}) & \boldsymbol{0} \end{bmatrix} \end{gathered}$$

The unique human gain matrix $\boldsymbol{L}(\boldsymbol{K})$ that *maximizes* the quadraticized cost is:

$$\boldsymbol{L}(\boldsymbol{K}) = (-\boldsymbol{R}_w + \boldsymbol{D}'\boldsymbol{P}_{\boldsymbol{K},\boldsymbol{L}(\boldsymbol{K})}\boldsymbol{D})^{-1}\boldsymbol{D}'\boldsymbol{P}_{\boldsymbol{K},\boldsymbol{L}(\boldsymbol{K})}(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{K}) \tag{13}$$

Note that $\boldsymbol{L}(\boldsymbol{K})$ is the *worst-case* gain matrix given the robot's policy and objective; we have found the human actions that maximize the robot's cost. To develop the correct response to this worst-case, we use a similar equation to compute the robot gain matrix $\boldsymbol{K}(\boldsymbol{L})$:

$$\boldsymbol{K}(\boldsymbol{L}) = (\boldsymbol{R}_u + \boldsymbol{B}'\boldsymbol{P}_{\boldsymbol{K}(\boldsymbol{L}),\boldsymbol{L}}\boldsymbol{B})^{-1}\boldsymbol{B}'\boldsymbol{P}_{\boldsymbol{K}(\boldsymbol{L}),\boldsymbol{L}}\,(\boldsymbol{A}-\boldsymbol{D}\boldsymbol{L}) \tag{14}$$

By approximating the system as LQ and recomputing the coupled DAREs for both Equation (13) and Equation (14), we converge to feedback policies that form the local Nash Equilibrium for zero-sum LQ games. Expanding these matrices provides our initial guess of the optimal action trajectory $\xi(x^0) = (u^{0:T}, w^{0:T})$. As a reminder, we do not use this trajectory for closed-loop control. Instead, we use this trajectory as a seed for the stochastic search presented in Section 4.2.

The traditional formulation of linear-quadratic approximations for zero-sum games can violate the eigenvalue conditions of Equation (11), even when a closed-form NE exists for the underlying non-LQ system. In practice, this can be alleviated by adjusting R_u and R_w manually. The gain matrices can be solved directly through Equation (11) or with readily available open-source software, such as Scipy's `solve_discrete_are` [35].

4.2 Refining the Action Trajectory

The initial action trajectory produced by the local equilibrium policies in Section 4.1 are optimal if the system is linear-quadratic. However, as established in Section 3, the environments that we are interested in may feature non-linear dynamics and non-quadratic costs, meaning that the LQ solution is merely a local approximation. These nonlinearities and nonconvexities cause the information gap $\overline{V}(x) - \underline{V}(x) > 0$. To ensure safety, we must bridge the gap between this local optimum and the true upper value of the game $\overline{V}(x)$.

To tractably solve this nested optimization, we propose using the LQ solution as a highly-informed "warm start" that is subsequently refined through stochastic gradient descent. Specifically, we apply a nested Metropolis-Hastings (MH) sampling algorithm. While other Monte Carlo methods could be substituted within our general framework, the ergodic properties of the MH sampler are particularly well suited for escaping the local optima that frequently trap pure LQ-approximate methods [2].

As a reminder, our ultimate goal is to reach the robust robot policy defined in Equation (7), which can be treated as a nested optimization problem. In the outer loop the robot proposes a sequence of actions $u^{0:T}$ to minimize its cost, and in the inner loop the antagonistic human responds with actions $w^{0:T}$ that maximize that cost. Because the robust policy takes the form of a minmax optimization, our MH sampler is explicitly divided into two coupled phases [14]: an inner loop that models the worst-case human response, and an outer loop that optimizes the robot's robust commitment. An outline of our resulting approach is shown in Algorithm 1.

Inner Loop. The inner loop seeks to identify worst-case human actions that increase the robot's cost. Given a fixed robot commitment ξ_u, the adversarial human wants to maximize J. We define the target distribution for the human's trajectory as a Gibbs measure:

$$P_{\pi_{\mathcal{H}}}(\xi_w \mid \xi_u, x) \propto \exp\Big(\beta J(x, \xi_u, \xi_w)\Big) \tag{15}$$

where $\beta > 0$ is the inverse temperature. As $\beta \to \infty$, the exponential function heavily exaggerates differences in J and the probability measure collapses into a Dirac delta function centered strictly on the global maximum of J. Therefore, sampling from $\pi_{\mathcal{H}}$ at a high β is theoretically equivalent to solving the inner sup problem. Sampling directly from this distribution is intractable for continuous and high-dimensional systems. Instead, we use the Metropolis-Hastings sampler [8] by proposing random perturbations $\Delta\xi_w$ of the human's action. These perturbations are accepted if they increase the total cost, or if they satisfy the acceptance rate:

$$\exp\Big(\beta \cdot \Big(J(x, \xi_u, \xi_w + \Delta\xi_w) - J(x, \xi_u, \xi_w)\Big)\Big) > \eta, \quad \eta \sim U[0,1] \tag{16}$$

In practice, using a β that is too large leads to numerical instability and reduces exploration, so we use a $\beta < 20$ and opt to do the comparison logarithmically. The inner loop repeats N times before terminating with a new human action sequence $\xi'_w = \xi_w + \Delta\xi_w^N$. Sampling random perturbations $\Delta\xi_w$ according to this scheme will find human actions that are the worst-case for a given robot action ξ_u. The mechanism for proposing actions human actions can instead use the nominal human model $\hat{\pi}_{\mathcal{H}}$; this process is detailed in Section 4.3.

Outer Loop. The outer loop takes the updated choice of human actions ξ'_w, and seeks to find robot actions that are robust to the human. Similar to the inner loop, we define the target distribution for the robust robot trajectory as:

$$P_{\pi_{\mathcal{R}}}(\xi_u \mid x) \propto \exp\left(-\beta \mathop{\mathbb{E}}_{\xi_w \sim P_{\pi_{\mathcal{H}}}}\Big[J(x, \xi_u, \xi_w)\Big]\right) \tag{17}$$

Algorithm 1 Monte Carlo Linear-Quadratic Games (MCLQ)

Require: f, J, β, M, N ▷ Dynamics and Cost Functions
procedure LQ Approximation(x, f, J)
  $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{D} \leftarrow$ Linearize(f)
  $\boldsymbol{Q}, \boldsymbol{R}_u, \boldsymbol{R}_w \leftarrow$ Quadraticize(J)
  $\boldsymbol{K}, \boldsymbol{L} \leftarrow$ DARE Solution
  $\xi_u \leftarrow -\boldsymbol{K}\boldsymbol{x}$
  $\xi_w \leftarrow -\boldsymbol{L}\boldsymbol{x}$
  return ξ_u, ξ_w
end procedure ▷ Provides initial guess of ξ_u, ξ_w
procedure Monte Carlo Search(x, ξ_u, ξ_w, f, J)
  $\Delta_u \leftarrow 0, \Delta_w \leftarrow 0$
  $J_0 \leftarrow J(x, \xi_u, \xi_w)$
  for $m = 1 \dots M$ **do** ▷ Outer Loop: find the best-case robot action
    for $n = 1 \dots N$ **do** ▷ Inner Loop: find the worst-case human response
      $\Delta_n \leftarrow \text{Perturb}(\xi_w, \Delta_w)$ ▷ Optionally use the safety margin λ or model $\hat{\pi}_H$
      $J_w \leftarrow J(x, \xi_u + \Delta_u, \xi_w + \Delta_n)$
      if $J_w > J_0 \vee \exp(\beta(J_w - J_0)) > \eta,\ \eta \sim U[0,1]$ **then**
        $J_0 \leftarrow J_w$
        $\Delta_w \leftarrow \Delta_n$
      end if ▷ Updates worst-case human actions
    end for
    $\Delta_m \leftarrow \text{Perturb}(\xi_u, \Delta_u)$
    $J_u \leftarrow J(x, \xi_u + \Delta_m, \xi_w + \Delta_w)$
    if $J_u < J_0 \vee \exp(\beta(J_0 - J_u)) > \eta,\ \eta \sim U[0,1]$ **then**
      $J_0 \leftarrow J_u$
      $\Delta_u \leftarrow \Delta_m$
    end if ▷ Updates robot response to human
  end for
  return $\xi'_u = \xi_u + \Delta_u, \xi'_w = \xi_w + \Delta_w$
end procedure

Like Equation (15), directly sampling from this distribution is difficult. We follow the same tractable sampling scheme as before. The outer loop modifies ξ_u by proposing random perturbations $\Delta\xi_u$. These perturbations are accepted if they decrease total cost, or if they satisfy the acceptance rate:

$$\exp\left(\beta \cdot \left(J\left(x, \xi_u, \xi'_w\right) - J\left(x, \xi_u + \Delta\xi_u, \xi'_w\right)\right)\right) > \eta, \quad \eta \sim U[0,1] \tag{18}$$

Importantly, the inner loop (i.e., the human) is *maximizing* the cost, while here the outer loop (i.e., the robot) is *minimizing* that same cost. This outer loop terminates after M iterations, and outputs a final robot action trajectory $\xi'_u = \xi_u + \Delta\xi_u^M$.

4.3 Balancing Performance and Safety

One key advantage of our approach is that — as we will show — it can be implemented in real-time. But another core aspect is that the designer can tune the *safety margin*, i.e., how conservative the robot's behavior is. Recall that $\hat{\pi}_{\mathcal{H}}$ is a predictive model of the human's actions. We recognize that the real human will inevitably deviate from $\hat{\pi}_{\mathcal{H}}$; but how different should we expect the human's

actions to be? Within a zero-sum game formulation, if the human can take any action at any time, then the robot is forced to overly conservative behaviors (e.g., the autonomous drone always moving away from the human). Unlike LQ approximations, our approach is structured to resolve this conflict between performance and safety. Specifically, during Monte Carlo sampling we can impose hard constraints on the perturbations $\Delta\xi_w$ so that they are close to the predictions of our human model. Let $\hat{\xi}_w \sim \hat{\pi}_{\mathcal{H}}$ be a predicted human trajectory. We can limit the range of considered human actions such that:

$$\|\hat{\xi}_w - (\xi_w + \Delta\xi_w)\|_2^2 \leq \lambda \tag{19}$$

where λ is a design parameter that determines conservatism. As $\lambda \to \infty$, the search space approaches the original trajectory space $\Xi_{\mathcal{H}}$, and as $\lambda \to 0$, the robot is increasingly confident in its human model. Importantly, Equation (19) does not determine *if* the human's behavior aligns with our nominal model: it restricts the search space to trajectories that are *similar* to our nominal model.

Instead of restricting the sampled actions such that they are close to the nominal model's prediction, we can change the sampling distribution of Equation (15) entirely. If we want the adversarial inner-loop to align with our expectation of the human's behavior, we can constrain the Gibbs measure of Equation (15) to be:

$$\arg\max_{P \in \mathcal{P}_{\pi_{\mathcal{H}}}} D(P \parallel \hat{\pi}_{\mathcal{H}}), \quad \text{s.t. } D(P \parallel \hat{\pi}_{\mathcal{H}}) \leq \lambda \tag{20}$$

where D is a divergence metric (such as the Kullback-Leibler Divergence [22]). This accomplishes the same goal as Equation (19): limiting conservatism in the robot policy. However, instead of enforcing the restriction λ on the sampled trajectories, we instead enforce it on the action distribution itself. In practice, this is accomplished by changing the perturbation distribution $\Delta\xi_w \sim P$ from a uniform distribution to one that aligns more closely — according to the conservatism parameter λ — with the nominal human model.

4.4 Summary

Taken together, the LQ game in Section 4.1 provides a first guess at the robot's action trajectory ξ. This action trajectory is then refined by nested Metropolis-Hastings samplers in Section 4.2 that update ξ to solve for the robust robot policy defined in Equation (7). The resulting trajectory ξ' contains the sequence of actions a robust robot should execute over the next T timesteps; this robust action sequence is recomputed online (i.e., at each timestep t). Notably, we compute the LQ game approximation *once* at the beginning of the interaction and use the nested Metropolis-Hastings samplers thereafter. To limit conservatism, the inner loop of the nested Metropolis-Hastings samplers can be tuned with a design parameter λ and a nominal human model using Section 4.3. In what follows, we will first theoretically justify that this approach is strictly better than alternatives that just use a local LQ-approximation, then experimentally validate this claim in Sections 5 and 6.

The Metropolis-Hastings samplers presented in Section 4.2 asymptotically converge to the target distributions $P_{\pi_{\mathcal{R}}}$ and $P_{\pi_{\mathcal{H}}}$. Since we run each sampler for a finite time, the samplers are not guaranteed to produce optimal trajectories. Following from [8, 34], the probability of a specific optimality gap $r \geq 0$ is approximately on the order of:

$$\begin{aligned} P(\|\xi' - \xi^\star\| \leq r) &\gtrsim 1 - O(e^{-\beta\Delta(r)}) - O(e^{-MN}) \\ \Delta(r) &= \inf\left[J\left(\xi'_u, \xi^\star_w\right) - J\left(\xi^\star_u, \xi'_w\right)\right] \end{aligned} \tag{21}$$

after MN iterations of the nested Metropolis-Hastings samplers, where $\xi^\star$ is the optimal trajectory. If the cost function is convex in ξ_u and concave in ξ_w (e.g., if the system has quadratic cost and

separable dynamics), then the relation simplifies to:

$$P\left(\|\xi' - \xi^\star\| \leq r\right) \gtrsim 1 - 2e^{-\beta r^2/2} - O(e^{-MN}) \tag{22}$$

With sufficient N, M, and β, MCLQ will approach the stationary distributions in Equations (15) and (17) and therefore produce the robust robot policy shown in Equation (7). Now, we consider three cases for the underlying environment: if the dynamics are linear and the cost is quadratic, if the dynamics are nonlinear, and if the system is non-linear-quadratic.

- **LQ Systems:** For a linear-quadratic system the DARE presented in Equation (11) finds the equilibrium point between the two agents, if it exists. This means that — for our MCLQ method — the initial trajectory is the most robust robot policy, and Monte Carlo sampling is unable to improve upon this initial guess.
- **Non-Linear Systems with Quadratic Cost:** If the system is non-linear but has quadratic cost, then MCLQ improves upon the original LQ approximation *proportional to the error in the dynamics linearization.* The resulting error follows Equation (22). As an example, for a separable 4-dimensional system evaluated over a control horizon of 10 steps, the probability that the trajectory produced by MCLQ is within one unit of the optimal trajectory approaches 99% with $\beta = 10$ as the total number of iterations in the stochastic search grows.
- **Non-Linear Non-Quadratic Systems:** If the system is not linear-quadratic, then MCLQ improves upon the LQ approximation with a probability shown in Equation (21). If the error in the LQ approximation is large, then the sampler will improve upon the initial trajectory with probability proportional to $1 - \exp(-\beta\Delta(r))$.

5 SIMULATIONS

In theory, our proposed MCLQ method moves towards robust, game-theoretic behaviors while offering real-time performance. Here we test our theoretical claims by comparing our approach to exact Hamilton-Jacobi methods and tractable LQ approximations. More specifically, we perform controlled experiments where simulated agents interact in three environments: point-mass, driving, and manipulator (see Figure 2). Each environment contains two agents. The simulated robot is attempting to complete its task (e.g., reaching a goal) while simulated humans move in its proximity. The robot must avoid collisions with these simulated humans — even when the humans take unexpected or noisy actions.

Independent Variables. We vary the robot's controller across four levels. To test LQ approximations, we include **DARE-nA** [38] and **ILQGames** [13] approaches. These methods iteratively approximate the dynamics and cost as a LQ system and solve for the resulting Nash Equilibrium. We used the official repositories for **DARE-nA** and **ILQGames** when applicable. To test exact solutions, we next solve for the true Nash Equilibrium (**NE**) using the Bellman Equation. Finally, we evaluate our **MCLQ** approach from Section 4.

Environments. Below we describe the three simulated environments. All simulations were performed on an AMD Ryzen 7000 Series 5 CPU with multithreading enabled.

Point-Mass. Here the simulated human and robot move in a 2D plane. The initial positions and velocities of both agents are randomized; across an interaction of $T = 30$ timesteps, the robot attempts to reach a fixed goal location. Both agents have linear dynamics such that $x^{t+1} = x^t + u^t + w^t$, and the robot's cost is quadratic (considering both distance to goal and distance to human). Because this environment is a linear-quadratic game, we anticipate that LQ approximations should find the Nash Equilibrium.

Driving. Our driving environment is taken from the iLQ baseline [13]. As before, one simulated human and robot move in a 2D plane with randomized initial configurations. The dynamics of each

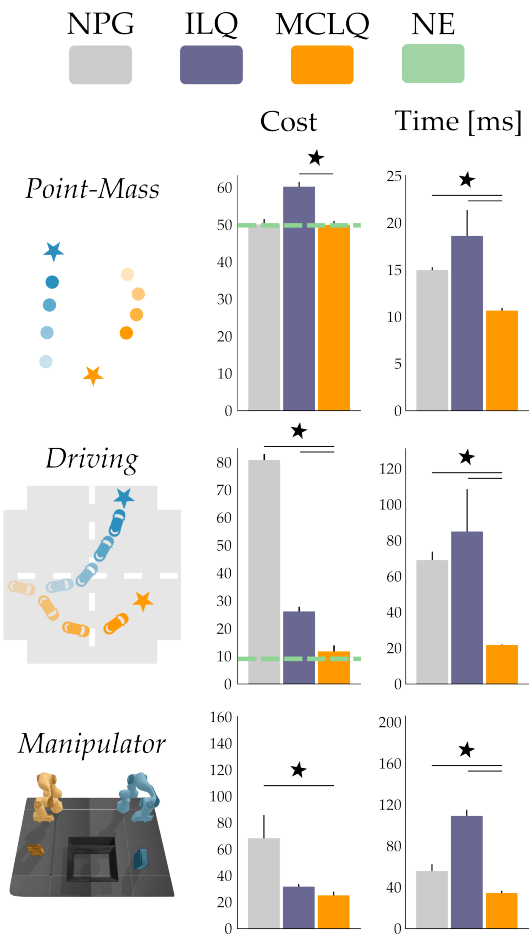

Fig. 2. Simulation results across *point-mass*, *driving*, and *manipulator* environments. (Left) We plot the cost and (Right) computation time averaged over 100 simulations. Computation time is the number of milliseconds per robot action (normalized by the number of timesteps per trajectory). In non-LQ settings the computation time for NE is prohibitively high; e.g., in *driving* the NE computation time exceeded one hour. We did not calculate NE in the 26-dimensional *manipulator* environment. Error bars show standard deviation and an $*$ denotes statistical significance.

agent follow a nonlinear bicycle model. Here the robot's cost function is not quadratic:

$$j(x, u, w) = \|x_{\mathcal{R}}^t - g\|^2 + \eta \exp\left(\frac{\|x_{\mathcal{R}}^t - x_{\mathcal{H}}^t\|^2}{-\eta}\right) + u' R_u u$$

where g is the goal position the robot is trying to reach, $x_{\mathcal{R}}$ is the robot's position, $x_{\mathcal{H}}$ is the human's location, and η scales the cost of approaching the human. Each interaction lasts a total of $T = 30$ timesteps.

Manipulator. Our final environment contains two 7-DoF robot arms in PyBullet. We treat one of these arms as the simulated human, and the other as the robot. Both agents are trying to pick up and place objects within a shared workspace. The dynamics of the arms and the task are nonlinear, and the system state $x \in \mathcal{X}$ is 26-dimensional (including manipulators and objects). We leverage a

cost function similar to *driving*. The robot moves to reach, grasp, and place items while avoiding collisions with the other arm. We separate this task into two separate sub-tasks: *reaching* and *placing*, each with separate cost functions. Settings with grasping violate the smooth dynamics assumptions of Section 3. To still use LQ approximations for the environment, the sub-task is automatically switched when the *reaching* sub-task's cost is below a certain threshold.

Simulated Human. We simulate humans as bounded-rational agents that noisily optimize their cost function J_H:

$$\begin{aligned} \pi_{\mathcal{R}}\left(w^t \mid x^t\right) &\propto \exp\left(-\alpha \cdot J_H(x^t, w_i^t)\right) \\ \text{s.t. } x^{t+1} &= f(x^t, u^{t_0}, w^t) \end{aligned} \tag{23}$$

Increasing $\alpha \to \infty$ causes the human to always take the optimal action, while decreasing $\alpha \to 0$ causes the human to act randomly [12]. In our simulations we set $\alpha = 7.5$. When computing the cost for future timesteps the simulated human assumes the robot will repeat its most recent action u^{t_0} (e.g., the robot will keep moving with the same velocity). In each environment the human had a task that was independent of the robot's objective — for instance, in *driving* the human tried to reach their own goal position.

Results. The results from our first simulation are summarized in Figure 2. Across all three environments, MCLQ achieved costs that were closest to the upper value of the game (**NE**) while also requiring the least amount of computation time. In *point-mass* we highlight that all methods reached similar cost; this matches our expectations, because *point-mass* was an LQ system. By contrast, in non-LQ systems (*driving* and *manipulator*) the LQ approximations made by DARE-nA and ILQ fell short, leading to suboptimal performance. Computing the exact solution with NE was time-consuming and not always feasible: indeed, in the 26-dimension *manipulator* task, we were unable to compute the true NE because of the high-dimensional and continuous state-action space.

Adjusting Safety Margins. As discussed in Section 4.3, one feature of our approach is the designer-selected safety margin. MCLQ robots reason over the worst-case human action within bounds λ. By increasing λ in Equation (19) the designer makes the robot more risk-averse (i.e., the robot considers larger deviations from the nominal human model). Conversely, decreasing λ makes the robot more risk-seeking (i.e., the robot increasingly relies on its human model). Within our simulations from Figure 2 we left this value of λ fixed at risk-neutral behavior. Now we explore how increasing and decreasing λ changes the robot's performance in a modified *point-mass* environment.

Our extended *point-mass* environment includes one to ten humans that are navigating to goal positions. The robot seeks to reach to its own goal while avoiding these randomly generated agents. To find an initial plan, the robot is equipped with a nominal human model — it assumes each human will move in a straight line towards their goal. In practice, however, our simulated humans deviate from this model when nosily optimizing their cost function. Figure 3 plots the robot's performance as a function of λ. When λ increases the robot becomes more risk-averse: the MH sampler accounts for a wider range of adversarial human actions. This causes the robot agent to stay farther from humans (reducing collisions), but also leads to longer paths (increasing distance). Conversely, lower values of λ cause to risk-seeking behavior where the robot relies on its human model. If humans stick to this model, the robot avoids collisions while minimizing distance. But when humans deviate, the number of collisions increase. Overall, this simulation supports our theoretical description of λ and demonstrates how designers can leverage MCLQ to tune the safety margin.

We further test how designers can change the human model within Equations (15) and (16) by comparing the performance of MCLQ across different human models. We leverage the noisily-optimal human model from Equation (23) with both varying levels of rationality $\alpha \in \{2.5, 10.0, 20.0\}$

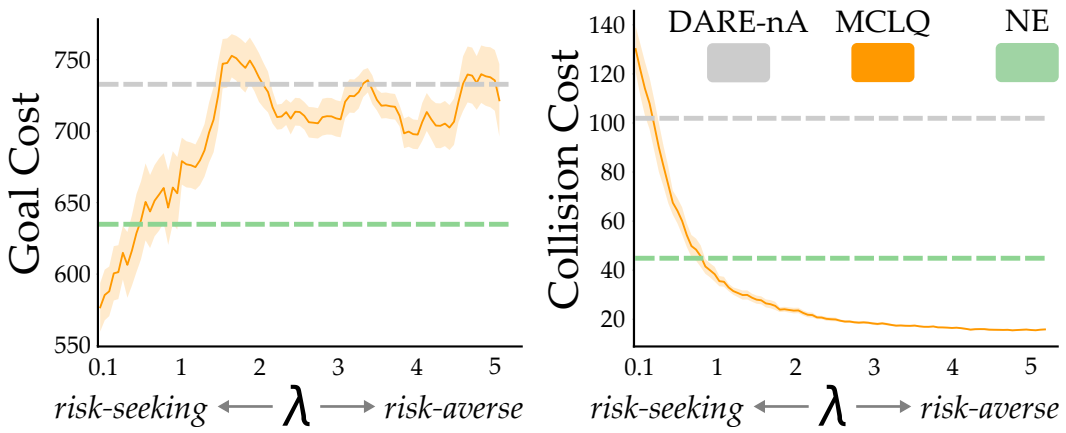

Fig. 3. Simulation results for a modified point-mass environment where we adjust the safety margin λ in MCLQ. Increasing λ causes the MCLQ robot to consider a wider range of worst case human actions, resulting in more conservative behavior. Conversely, decreasing λ causes the MCLQ robot to increasingly rely on its nominal human model. Unlike LQ approximations, our proposed method gives designers the flexibility to tune λ and adjust the safety margin. In practice, this results in risk-averse robots that avoid all possible collisions but reach the goal more slowly, or risk-seeking robots that allow some potential collisions and rapidly move to the goal.

and cost functions $J_H \in \{J_H^1, J_H^2, J_H^3\}$:

$$J_H^1(x,u,w) = \|x_H^N - g_H\|_{P_H}^2 + \sum_{t=0}^{N-1} \|x_H^t - g_H\|_{Q_H}^2 + \|w^t\|_{R_H}^2 \tag{24}$$

$$J_H^2(x,u,w) = \|x_H^N - g_H\|_{P_H}^2 + \sum_{t=0}^{N-1} \|x_H^t - g_H\|_{Q_H}^2 + \eta \exp\left(\frac{\|x_R^t - x_H^t\|_2^2}{-\eta}\right) + \|w^t\|_{R_H}^2 \tag{25}$$

$$J_H^3(x,u,w) = \|x_H^N - g_H\|_{P_H}^2 + \sum_{t=0}^{N-1} \|x_H^t - x_R^t\|_{Q_H}^2 + \|w^t\|_{R_H}^2 \tag{26}$$

$$\text{s.t.} \quad x^{t+1} = f(x^t, u^t, w^t), \quad u^t \sim \pi_{\mathcal{R}}(x^t) \tag{27}$$

where Q_H, R_H, and P_H are fixed, time-invariant matrices and $\pi_{\mathcal{R}}$ is our proposed method for selecting robust robot actions. These cost functions cause the human to move towards the goal while minimizing their action size, but they also ensure that the human follows one of three general behaviors: ignore the robot's position (24), avoid the robot's position (25), and follow the robot (26). Intuitively, each of these cost functions could be a reasonable approach for predicting the human's behavior, and designers may choose any of these (or other similar functions). Our goal is to demonstrate that even when the human model is incorrect — and the human follows another underlying behavior pattern — our approach to robust interact still provides a safety measure.

As expected, when the actual human aligns with the behavior that MCLQ is modeling, the number of average collisions decreases (Figure 4). Notably, when the predicted human model is more pessimistic (such as in Equation (26)), MCLQ tends to avoid the human more at the expense of goal convergence. We test MCLQ with a safety margin $\lambda = 1.0$ according to Equation (19) and a non-quadratic cost function similar to the *Driving* environment. Notably, the iterative LQ

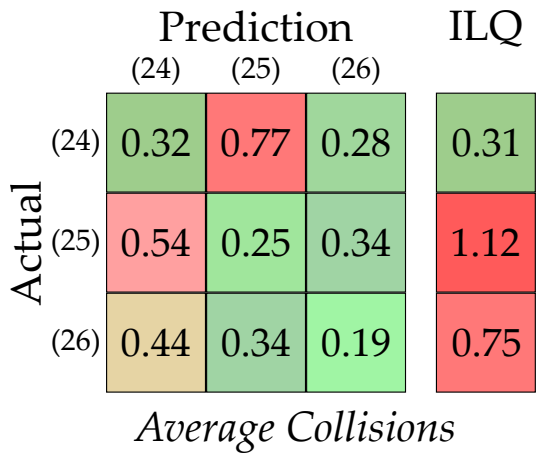

Fig. 4. Generally, when the human model aligns with the actual human behavior, MCLQ avoids worst-case scenarios (here, collisions). The annotations (24), (25), and (26) correspond to the human models in Equations (24-26). Here we define a collision as an instance where the human is within 5 units of the robot. The results are averaged across 1089 trials with 9 variations of human models in the point-mass environment. Note that values along the diagonal tend to be the lowest, indicating that when the actual and modeled motions of the human model align, the robot maintains safety. Overall, when a pessimistic human model is used (such as Equation (26)), the robot tends to avoid the human more frequently. In this experiment, we define a "collision" as a state where the human and robot agents are within 5 units of each other. The initial configuration of the agents is randomly sampled from $U[-100, 100]$. Similar to *Point Mass*, each agent has two degrees of freedom (i.e., $\mathcal{X} = \mathbb{R}^4$ and $\mathcal{U} = \mathcal{W} = \mathbb{R}^2$) and each interaction lasts for $T = 100$ timesteps.

approximation baseline has similar performance for Equation (24), but fails for complex behaviors that are more difficult to approximate.

6 USER STUDY

Our simulations from Section 5 support our theoretical analysis, and suggest that MCLQ converges towards maximally-robust behaviors while minimizing computation time. We next evaluate our method in a real-world setting with $N = 24$ in-person human participants. Here users walk around a room to complete an assembly task while an autonomous drone circles that room to inspect the parts (see Figure 1). The robot modifies its high-level trajectory to avoid getting to close to the human workers. We compare two *real-time* methods for safe interactions that adjust the drone's behavior: ILQ [13] and MCLQ. We selected ILQ here because it was the best performing real-time baseline from our simulations.

Experimental Setup. Participants interacted with a Crazyflie 2.1+ (Bitcraze) during the assembly task. The participant's objective was to construct a Lego tower at the central station. The blocks needed for building that tower were scattered in three other stations located along the perimeter of the workspace. Accordingly, users needed to move back and forth through the workspace to acquire blocks and build their tower. The drone's objective was to monitor the workspace during the task by completing as many revolutions around the central station as possible. As the drone moved around the central station it repeatedly intersected with the human worker: here the drone should take actions that avoid the moving participant. Participants interacted with the drone separately (i.e., the drone interacted with one user at a time).

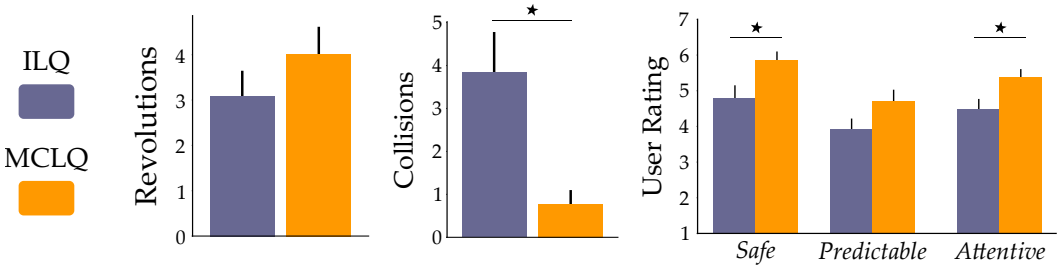

Fig. 5. Results from our user study in Section 6. Participants walked around a room to assemble a tower; a drone completed revolutions around the same workspace to monitor the human's progress (also see Figure 1). (Left) The average number of revolutions the drone completed and the average number of collisions. Here "collisions" occurred when the drone was within 0.5 meters of the human. The proposed MCLQ algorithm adjusts the robot's behavior to increase safety (fewer collisions) while also enhancing performance (more revolutions). (Right) After interacting with each algorithm participants answered survey questions about how safe, predictable, and attentive the robot was. Ratings suggest that participants perceived MCLQ to be a safer system. Error bars show standard deviation and an $*$ denotes statistical significance ($p < .05$).

Participants and Procedure. We recruited 24 participants (6 female, age 24.5 ± 4.5) from our university community for this user study. Of the 24 participants, 8 had not used drones and 5 did not have experience with robotics. Participants received monetary compensation for their time and provided informed written consent according to university guidelines (IRB #23-1237).

We leveraged a between-subjects design where every participant interacted with ILQ for five minutes and MCLQ for five minutes. To prevent the participants from always going back and forth between the same stations, we instructed users to move according to three movement patterns: clockwise, counter-clockwise, and random. Both the order of the methods and movement patterns were counterbalanced (i.e., half of the participants started with MCLQ). Participants were never told which algorithm the drone was using.

Dependent Measures — Objective. We using tracking devices (VIVE) to measure the states and actions of the drone and human at each timestep of the interaction. To assess objective performance, we considered two metrics. For our first metric (*collisions*) we counted the number of times the drone was within a radius of 0.5m from the human. Our second metric (*revolutions*) is the number of revolutions the drone completed within the five minute trial. Lower values for *collisions* indicate that the drone is maintaining safety, while higher values for *revolutions* show that the drone is performing the task more efficiently.

Dependent Measures — Subjective. After interacting with each control algorithm, participants completed a 7-point Likert scale survey. This survey assessed the user's subjective preferences along three multi-item scales. We asked users:

(1) If they felt *safe* during the interaction,
(2) If they thought the drone was *attentive* to their position,
(3) If they thought the drone's movements were *predictable*.

Hypotheses. We had two hypotheses for this user study:

> **H1. MCLQ** *will modify the drone's actions to reduce the number of collisions and improve subjective feelings of safety.*
>
> **H2. MCLQ** *will complete a similar number of revolutions as the iLQ baseline.*

Results. The results from our in-person user study are summarized in Figure 5. To assess **H2**, we measured the number of revolutions that each method completed per interaction: a higher number indicates more efficient performance. One-way ANOVA tests show that the performance difference between the methods is trending towards significance ($F(2, 86) = 1.50$, $p = 0.14$), where MCLQ completes more revolutions than ILQ. This suggests that the safety improvements made by MCLQ are not coming at the expense of overly conservative behavior — the MCLQ drone is still performing its high-level task of monitoring the workspace.

Given that the drone is completing a similar number of revolutions with each method, the key question becomes the *safety* of the robot's behavior. We analyzed **H1** along two levels: objective safety (maintaining a minimum distance between agents) and subjective safety (the user's perception of the system). For objective safety, ANOVA tests revealed that MCLQ leads to significantly fewer collisions than ILQ ($F(2, 86) = 4.16$, $p < 0.001$). The participants' responses to our Likert scale survey align with these objective results: users perceived the MCLQ robot to be safer ($F(2, 24) = 2.33$, $p < 0.05$) and more attentive ($F(2, 24) = 2.37$, $p < 0.05$). In addition, users thought that drones following the MCLQ algorithm had more predictable actions, with differences trending towards significance ($F(2, 24) = 1.84$, $p = 0.073$). In their free response comments, participants stated that with the MCLQ drone they "*felt safer*" and that the drone was "*more reactive and predictable*."

7 CONCLUSION

In this paper we presented a real-time safety approach for human-robot interaction. Our approach ensures that the robot is robust to noisy and unexpected human behaviors within designer-specified bounds. To achieve this game-theoretic safety with tractable computation, we combined linear-quadratic approximations with stochastic local searches. Our theoretical and empirical analysis showed the resulting MCLQ algorithm converges towards robust robot policies while providing flexible and efficient implementation. Across multiple simulations and a user study, we observed that MCLQ advances both objective and subjective safety measures when compared to state-of-the-art control alternatives.

Limitations. MCLQ is a step towards control policies that are robust to human disturbances. Since the stochastic search is performed on the underlying system directly, MCLQ is not restricted to systems that can be approximated as linear-quadratic. One key assumption in our work is that the objective function and state dynamics can be evaluated quickly. The inner loop of the double Metropolis-Hastings sampler is somewhat expensive: every perturbation of the original trajectory mandates recalculating the dynamics and cost. Performing this recalculation quickly is not always feasible, such as in fluid dynamics or with visuomotor policies.

REFERENCES

[1] Tianjiao An, Xinye Zhu, Mingchao Zhu, Bing Ma, and Bo Dong. 2023. Fuzzy logic nonzero-sum game-based distributed approximated optimal control of modular robot manipulators with human-robot collaboration. Neurocomputing 543 (2023), 126276.

[2] Christophe Andrieu and Éric Moulines. 2006. On the ergodicity properties of some adaptive MCMC algorithms. The Annals of Applied Probability 16, 1 (2006), 1462–1505.

[3] Andrea Bajcsy, Somil Bansal, Eli Bronstein, Varun Tolani, and Claire J Tomlin. 2019. An efficient reachability-based framework for provably safe autonomous navigation in unknown environments. In IEEE Conference on Decision and Control. 1758–1765.

[4] Somil Bansal, Andrea Bajcsy, Ellis Ratner, Anca D Dragan, and Claire J Tomlin. 2020. A Hamilton-Jacobi reachability-based framework for predicting and analyzing human motion for safe planning. In IEEE International Conference on Robotics and Automation. 7149–7155.

[5] Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. 2017. Hamilton-Jacobi reachability: A brief overview and recent advances. In IEEE Annual Conference on Decision and Control. 2242–2253.

[6] Somil Bansal and Claire J Tomlin. 2021. Deepreach: A deep learning approach to high-dimensional reachability. In 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 1817–1824.

[7] Tamer Başar and Geert Jan Olsder. 1998. Dynamic Noncooperative Game Theory. SIAM.

[8] Siddhartha Chib and Edward Greenberg. 1995. Understanding the metropolis-hastings algorithm. The american statistician 49, 4 (1995), 327–335.

[9] Christian M Chilan and Bruce A Conway. 2020. Optimal nonlinear control using Hamilton–Jacobi–Bellman viscosity solutions on unstructured grids. Journal of Guidance, Control, and Dynamics (2020).

[10] Benjamin A Christie and Dylan P Losey. 2024. LIMIT: Learning interfaces to maximize information transfer. ACM Transactions on Human-Robot Interaction 13, 4 (2024), 1–26.

[11] Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. 2019. Bridging hamilton-jacobi safety analysis and reinforcement learning. In 2019 International Conference on Robotics and Automation (ICRA). IEEE, 8550–8556.

[12] David Fridovich-Keil, Andrea Bajcsy, Jaime F Fisac, Sylvia L Herbert, Steven Wang, Anca D Dragan, and Claire J Tomlin. 2020. Confidence-aware motion prediction for real-time collision avoidance. The International Journal of Robotics Research 39, 2-3 (2020), 250–265.

[13] David Fridovich-Keil, Ellis Ratner, Lasse Peters, Anca D Dragan, and Claire J Tomlin. 2020. Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In IEEE International Conference on Robotics and Automation. 1475–1481.

[14] Joshua Hoegerman and Dylan Losey. 2023. Reward learning with intractable normalizing functions. IEEE Robotics and Automation Letters 8, 11 (2023), 7511–7518.

[15] Kai-Chieh Hsu, Haimin Hu, and Jaime F Fisac. 2023. The safety filter: A unified view of safety-critical control in autonomous systems. Annual Review of Control, Robotics, and Autonomous Systems 7 (2023).

[16] Haimin Hu, David Isele, Sangjae Bae, and Jaime F Fisac. 2024. Active uncertainty reduction for safe and efficient interaction planning: A shielding-aware dual control approach. The International Journal of Robotics Research 43, 9 (2024), 1382–1408.

[17] Haimin Hu, Zixu Zhang, Kensuke Nakamura, Andrea Bajcsy, and Jaime F Fisac. 2023. Deception game: Closing the safety-learning loop in interactive robot autonomy. arXiv preprint arXiv:2309.01267 (2023).

[18] Rufus Isaacs. 1999. Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization. Courier Corporation.

[19] Frank Jiang, Glen Chou, Mo Chen, and Claire J Tomlin. 2016. Using neural networks to compute approximate and guaranteed feasible Hamilton-Jacobi-Bellman PDE solutions. arXiv preprint arXiv:1611.03158 (2016).

[20] Morgan Jones and Matthew M Peet. 2020. Polynomial approximation of value functions and nonlinear controller design with performance bounds. arXiv preprint arXiv:2010.06828 (2020).

[21] Kushal Kedia, Atiksh Bhardwaj, Prithwish Dan, and Sanjiban Choudhury. 2024. Interact: Transformer models for human intent prediction conditioned on robot actions. In 2024 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 621–628.

[22] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. The annals of mathematical statistics 22, 1 (1951), 79–86.

[23] Forrest Laine, David Fridovich-Keil, Chih-Yuan Chiu, and Claire Tomlin. 2023. The computation of approximate generalized feedback Nash equilibria. SIAM Journal on Optimization 33, 1 (2023), 294–318.

[24] Tenavi Nakamura-Zimmerer, Qi Gong, and Wei Kang. 2020. A causality-free neural network method for high-dimensional Hamilton-Jacobi-Bellman equations. In American Control Conference. 787–793.

[25] Tenavi Nakamura-Zimmerer, Qi Gong, and Wei Kang. 2021. Adaptive deep learning for high-dimensional Hamilton–Jacobi–Bellman equations. SIAM Journal on Scientific Computing 43, 2 (2021), A1221–A1247.

[26] Youngim Nam and Cheolhyeon Kwon. 2024. Active inference-based planning for safe human-robot interaction: Concurrent consideration of human characteristic and rationality. IEEE robotics and automation letters 9, 8 (2024), 7086–7093.

[27] Sagar Parekh, Lauren Bramblett, Nicola Bezzo, and Dylan P Losey. 2025. Using high-level patterns to estimate how humans predict a robot will behave. In 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 16947–16954.

[28] Sagar Parekh and Dylan P Losey. 2023. Learning latent representations to co-adapt to humans. Autonomous Robots 47, 6 (2023), 771–796.

[29] Shahabedin Sagheb, Sagar Parekh, Ravi Pandya, Ye-Ji Mun, Katherine Driggs-Campbell, Andrea Bajcsy, and Dylan P Losey. 2025. A unified framework for robots that influence humans over long-term interaction. arXiv preprint arXiv:2503.14633 (2025).

[30] Maurice Sion. 1958. On general minimax theorems. (1958).

[31] Oliver Slumbers, David Henry Mguni, Stefano B Blumberg, Stephen Marcus Mcaleer, Yaodong Yang, and Jun Wang. 2023. A game-theoretic framework for managing risk in multi-agent systems. In International Conference on Machine Learning. 32059–32087.

[32] Alan Wilbor Starr and Yu-Chi Ho. 1969. Nonzero-sum differential games. Journal of Optimization Theory and Applications 3 (1969), 184–206.

[33] Ran Tian, Liting Sun, Andrea Bajcsy, Masayoshi Tomizuka, and Anca D Dragan. 2022. Safety assurances for human-robot interaction via confidence-aware game-theoretic human models. In IEEE International Conference on Robotics and Automation. 11229–11235.

[34] Luke Tierney. 1994. Markov chains for exploring posterior distributions. the Annals of Statistics (1994), 1701–1728.

[35] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature methods 17, 3 (2020), 261–272.

[36] Kim P Wabersich, Andrew J Taylor, Jason J Choi, Koushil Sreenath, Claire J Tomlin, Aaron D Ames, and Melanie N Zeilinger. 2023. Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems. IEEE Control Systems Magazine 43, 5 (2023), 137–177.

[37] Mingyu Wang, Negar Mehr, Adrien Gaidon, and Mac Schwager. 2020. Game-theoretic planning for risk-aware interactive agents. In IEEE/RSJ International Conference on Intelligent Robots and Systems.

[38] Jiduan Wu, Anas Barakat, Ilyas Fatkhullin, and Niao He. 2023. Learning zero-sum linear quadratic games with improved sample complexity. In IEEE Conference on Decision and Control. 2602–2609.