Differential Games & iLQG

Neuro-fuzzy Control Project

National Technical University of Athens
Electrical and Computer Engineering

Kordas Nikolaos
Registration Number: 03121032

Abstract

Contents

1 Introduction **2**
1.1 Background . . . 2
1.1.1 LQR Control Problem . . . 2
1.1.2 General Sum Differential Games . . . 3
1.2 The iLQG Algorithm . . . 7
1.2.1 Problem Formulation . . . 7
1.2.2 Local Approximation . . . 8
1.3 Computational Complexity . . . 9

2 Result Reproduction **9**
2.1 Three-player Intersection . . . 9
2.1.1 Problem Description . . . 10
2.1.2 System Dynamics . . . 10
2.1.3 Cost Functions . . . 10
2.1.4 Our Scenario and Results . . . 10

3 Experiments **11**
3.1 Human-Robot Game 1 . . . 11
3.2 Human-Robot Game 2 . . . 13

4 Discussion and Limitations **14**
4.1 Simulation Implementation . . . 14
4.2 Limitations . . . 14

1 Introduction

1.1 Background

1.1.1 LQR Control Problem

The Linear Quadratic Regulator problem is an optimal control problem concerning dynamic systems with linear dynamics and quadratic cost functions. The objective is to find the optimal control u that minimizes the cost J [4].

- A linear time-varying system:

$$\dot{x} = A(t)x + B(t)u,\ x(t_0) = x_0,\ x \in \mathbb{R}^n,\ u \in \mathbb{R}^m$$

- Quadratic cost function:

$$J(x,u,t) = \int_{t_0}^{t_f} (x^T(t)Qx(t) + u^T(t)Ru(t))dt + x^T(t_f)Q_f x(t_f),$$

$$Q = Q^T \geq 0,\ Q_f = Q_f^T \geq 0,\ R = R^T > 0$$

  The running cost's matrices Q and R penalize the size of the state and the control effort respectively.

We derive the necessary conditions for the minimization of J by applying Pontryagin's Maximum Principle. This yields that the optimal control is a linear state feedback law of the form:

$$u^*(t) = -R^{-1}(t)B^T(t)P(t)x^*(t)$$

where P must be a solution of the matrix differential equation (Riccati differential equation (RDE)):

$$\dot{P}(t) = -P(t)A(t) - A^T(t)P(t) - Q(t) + P(t)B(t)R^{-1}(t)B^T(t)P(t),\ P(t_f) = Q_f$$

A special case of the LQR problem described above is the infinite-horizon LQR problem. Now, we assume that A, B, Q and R are constant (transient phenomena have subsided) and as $t_f \to \infty$ the terminal cost becomes negligible, thus $Q_f = 0$. So, the dynamics of the control system become time-invariant and the cost function becomes $J(x,u,t) = \int_{t_0}^{\infty}(x^T(t)Qx(t)+u^T(t)Ru(t))dt$. The optimal solution is given by:

1. $u^*(t) = -R^{-1}B^T P x^*(t)$
2. The limit $P = \lim_{t_f \to \infty} P(t_0, t_f)$ of the solution of the RDE is a constant matrix that satisfies the Algebraic Riccati Equation (ARE):

$$PA + A^T P + Q - PBR^{-1}B^T P = 0$$

3. The optimal cost is: $J(u^*) = x_0^T P x_0$
4. The closed-loop system $\dot{x}^* = (A - BR^{-1}B^T P)x^*$ is exponentially stable, as long as the couple $(A, \sqrt{Q})$ is detectable.

The theoretical foundation of the LQR problem is necessary for the understanding of dynamic environments featuring multiple agents. In classical optimal control, our objective is to minimize a unique cost function. In contrast, in multi-player games, the optimization process is coupled. Actions of one agent influence the perfrormance and the state of the others. This transforms the Riccati equation to a system of coupled Riccati equations. Their solutions lead to a Nash equilibrium point, not a minimum. This framework is the basis of the iLQG algorithm which utilizes LQR iteratively by approximating non-linear dynamics as LQ problems.

1.1.2 General Sum Differential Games

Before developing a theoretical framework in the context of general-sum differential games, it is advisable to examine several simpler cases.

Zero-sum Matrix Games

To analyze multi-agent interactions, we first examine two-player zero-sum games, which are effectively represented in bi-matrix form [1]. Let P_1 and P_2 be the participants, choosing from m and n available strategies, respectively. The game is defined by a pair of matrices (A, B), where each entry $(a_{ij} = J_1(u_1, u_2), b_{ij} = J_2(u_1, u_2))$ represents the cost incurred by P_1 and P_2 for a specific strategy pair (i, j). In a zero-sum game, the players' interests are diametrically opposed, satisfying the condition $B = -A$. Consequently, for every outcome, the sum of their costs is $J_1(u_1, u_2) + J_2(u_1, u_2) = 0$, for every strategy pair (u_1, u_2) implying that any advantage gained by one agent results in an equivalent loss for the other.

P_1/P_2	a	b
c	$2,\ -2$	$0,\ 0$
d	$4,\ -4$	$-4,\ 4$

(a) Game 1

P_1/P_2	a	b
c	$-2,\ 2$	$0,\ 0$
d	$2,\ -2$	$-2,\ 2$

(b) Game 2

Table 1: Examples of zero-sum games

In table 1 above, matrices A and B are merged into a single cost matrix C. We suppose that P_1 can play either c or d and P_2 can play either a or b. Both P_1 and P_2 try to minimize their cost functions.
In the game in table 1a, P_2 should rationally play strategy a because it yields a lower cost for P_2 regardless of whatP_1 plays. Knowing P_2 is rational and will play a, P_1 plays strategy c every time, choosing the lesser cost. Even if P_2 were not guaranteed to play a, P_1's security strategy is to play c because that is the strategy that ensures the smallest cost (2). This "rational" solution is called a *saddle-point* solution.
In our second example (1b), there is no saddle-point solution. P_1's security strategy is to play c (this strategy minimizes his maximum loss - 0). P_2's security strategies are both a and b (same smallest maximum loss). However, if P_2 plays securely (b) and P_1 knows this, P_1 has and incentive to switch to b. Conversely, if P_1 plays c (security strategy) and P_2 knows this, P_2 has to choose b from his security strategies. Thus, unlike the first game, there is not a Nash equilibrium.
A formal definition of Nash equilibrium can be found in [6].

Suppose a game with players. If $J_1(u_1, ..., u_N), ..., J_N(u_1, ..., u_N)$ are their respective cost functions, then the strategy set $\{u_1^*, ..., u_N^*\}$ is a Nash equilibrium if for every $i = 1, ..., N$:

$$J_i(u_1^*, ..., u_{i-1}^*, u_i, u_{i+1}^*, ..., u_N^*) \geq J_i(u_1^*, ..., u_N^*)$$

According to [6], all Nash equilibria are equivalent in a zero-sum game. Thus, every Nash equilibrium's strategy set is interchangeable.
In zero-sum games, players' interests are diametrically opposed—one's gain is the other's loss—so there is no possibility of mutual gain. Consequently, the primary solution concept of interest is a Nash (or minimax) equilibrium, where each player's strategy is optimal given the other's.

Nonzero-sum Matrix Games

In this case, the sum of the cost functions is not zero for every possible outcome dictated by the strategies selected.

P_1/P_2	Not Betray	Betray
Not Betray	2, 2	10, 1
Betray	1, 10	5, 5

Table 2: Prisoner's Dilemma

Table 2 is the famous game Prisoner's Dilemma in bi-matrix form. Here, the cost functions correspond to years in prison for each player (P_1 and P_2). The only Nash equilibrium solution is $(u_1, u_2) = (\text{Betray}, \text{Betray})$. But, it is not optimal in the sense that $(u_1, u_2) = (\text{Not Betray}, \text{Not Betray})$ yields a better result for both players. This illustrates the possibility of mutual interest in nonzero-sum games. A solution produced by the coalition of all the players is called pareto-optimal solution [6].
In addition, in nonzero-sum games, every Nash equilibrium strategy is not equivalent to every other (cost-wise).

While the preceding discussion of zero-sum and nonzero-sum games provides a foundational understanding of strategic interaction in discrete-time, finite-action settings, many real-world problems involve dynamics, continuous time, and evolving states. To model strategic decision-making in such contexts—where players' actions influence not only immediate payoffs but also future states and opportunities—we extend the framework to differential games. These games generalize optimal control theory to multi-agent settings, incorporating differential equations to describe state evolution and enabling the analysis of continuous-time strategic behavior over a horizon. In what follows, we introduce the core formulation of differential games, including dynamics, cost functionals, information structures, and solution concepts such as open-loop and feedback Nash equilibria.

General Differential Games

In General Differential Games [6], the formulation resembles that of classical optimal control problems. I a nonzero-sum, N-player differential game, each player i seeks to select a strategy (control) u_i so as to minimize an individual cost function of the form:

$$J_i = K_i(x(t_f), t_f) + \int_0^{t_f} L_i(x, u_1, ..., u_N)dt$$

The optimization's constraints are defined by the game's dynamics:

$$\dot{x} = f(x, u_1, ..., u_N, t),\ x(t_0) = x_0$$

Such a problem may include holonomic and inequality constraints on the state x and controls u_i. The players' strategies interact through both the dynamics and the cost functionals, leading to a strategic dynamic optimization problem.

A key distinction from classical optimal control problems is that, in differential games, one must specify the solution concept that characterizes rational strategic interaction among players. Unlike optimal control where a single decision-maker minimizes a cost functional, a differential game requires equilibria such as a Nash equilibrium or outcomes that are Pareto-optimal, depending on whether the setting is non-cooperative or cooperative [6]. Here, it is assumed that every player knows the value of the state vector, the system parameters and its cost functions, but not the rival players' strategies.

This project is about a special class of differential games where the system is linear and the cost functions are quadratic. These are known as Linear Quadratic Games. Due to their structural simplicity, LQ games often admit closed-form or numerically efficient solutions, while still capturing fundamental trade-offs such as control effort versus state regulation and strategic interaction among players.

Linear Quadratic Games

The cost function of each player i is of the form:

$$J_i = \frac{1}{2}(x^T S_{if} x)_{t=t_f} + \frac{1}{2}\int_{t_0}^{t_f} (x^T Q x + \sum_{j=1}^{N} u_j^T R_{ij} u_j)dt$$

Player i should choose a control strategy $u_i = \Psi_i(x, t)$ to minimize that cost function subject to the constraints imposed by the system's dynamics:

$$\dot{x} = Ax + \sum_{j=1}^{N} B_j u_j$$

- *Nash Equilibrium Solutions:* This type of solution is secure against arbitrary and attempts by single players to unilaterally alter their strategy. The reason lies withing Nash Equilibrium definition. One can observe that a player can only lose by deviating from his control strategy. Thus, those solutions are particularly interesting in games where cooperation is impossible or difficult to enforce.

  For our LQ game, the terminal and running cost for each player i are:

$$K_i(x(t_f), t_f) = \frac{1}{2}(x^T S_{if} x)_{t=t_f}$$

$$L_i(x, u_1, ..., u_N) = x^T Q x + \sum_{j=1}^{N} u_j^T R_{ij} u_j$$

  One can obtain the necessary conditions for a Nash equilibrium solution by applying Pontryagin's maximum principle. Those are listed in [6]:

1. $\dot{x} = f(x, u_1, ..., u_N, t),\ x(t_0) = x_0$
2. $\dot{\lambda}_i^T = -\frac{\partial H_i(x,u_1,...,u_N,t,\lambda_i)}{\partial x} - \sum_{j=1,j\neq i}^{N} \frac{\partial H_i}{\partial u_j}\frac{\partial \Psi_i}{\partial x}(x,t)$
3. $\lambda_i^T(x(t_f)) = \frac{\partial K_i(x(t_f),t_f)}{x(t_f)}$
4. $u_i = \Psi_i(x,t) = \arg\min_{u_i'} H_i(x, t, \Psi_1, ..., \Psi_{i-1}, u_i', \Psi_{i+1}, ..., \Psi_N, \lambda_i)$

The Hamiltonian of each player in an LQ game is:

$$H_i(x,t,u,\lambda_i^T) = L_i(x,u,t) + \lambda_i^T f(x,u,t) = x^T Q x + \sum_{j=1}^{N} u_j^T R_{ij} u_j + \lambda_i^T \left(Ax + \sum_{j=1}^{N} B_j u_j \right)$$

The application of necessary condition 4 yields:

$$\frac{\partial H_i}{\partial u_i} = R_{ii} u_i + B_i^T \lambda_i$$

$$\frac{\partial H_i}{\partial u_i} = 0 \implies u_i^* = -R_{ii}^{-1} B_i^T \lambda_i$$

Then, we apply condition 2 and get:

$$u_i^* = R_{ii}^{-1} B_i^T P_i x$$

where P_i is a solution of the coupled Riccati equation:

$$\dot{P}_i = -P_i A - A^T P_i - Q_i - \sum_{j=1}^{N} (P_j B_j R_{jj}^{-1} R_{ij} R_{jj}^{-1} B_j^T P_j - P_i B_j R_{jj}^{-1} B_j^T P_j - P_j B_j R_{jj}^{-1} B_j^T P_i)$$

$$P_i(t_f) = P_{if}$$

For $N = 1$, this is the RDE equation that was presented in our LQR Control Problem section (corresponds to 1 player).

- *Minimax Controls:* Paper [6] states that this is the solution of a two-player, zero-sum LQ game where the opponent chooses strategy trying to maximize J_i. The minimax control for player i is:

$$\bar{u}_i = -R_{ii}^{-1} B_i^T \bar{P}_i x$$

where

$$\dot{\bar{P}}_i = -\bar{P}_i A - A^T \bar{P}_i - Q_i + \bar{P}_i \sum_{j=1}^{N} B_j R_{ij}^{-1} B_j^T \bar{P}_i,\ R_{ii} > 0,\ R_{ij} < 0,\ j \neq i$$

This strategy's goal is to guarantee minimum cost against the worst possible set of strategies the other players can choose. It shall be used when one is not sure if their opponents will play their Nash controls.

- *Non-inferior Controls:* This type of solutions are preferable if a negotiated solution can be reached and enforced. For LQ games the non-inferior controls are:

$$\hat{u}_i(\mu) = -\left[\sum_{j=1}^{N} \mu_j R_{ji}\right]^{-1} B_i^T \hat{P}(\mu)x$$

where

$$\dot{\hat{P}}(\mu) = -\hat{P}A - A^T\hat{P} - \sum_{j=1}^{N} Q_j + \hat{P}\sum_{i=1}^{N} B_i \left[\sum_{j=1}^{N} \mu_j R_{ji}\right] B_i^T \hat{P},\ \hat{P}(\mu, t_f) = \sum_{i=1}^{N} \mu_i P_{if}$$

$$\sum_{j=1}^{N} \mu_i = 1, \mu_i \geq 0$$

1.2 The iLQG Algorithm

The iLQG algorithm is a method for solving general-sum differential non-linear games based on iterative Linear Quadratic Regulator (iLQR). The main idea is the iterative solution of approximate LQ games around a local trajectory.

1.2.1 Problem Formulation

Based on [3] the N-player finite horizon, general-sum differential game we focus on and will experiment with is defined by non-linear dynamics

$$\dot{x} = f(t, x, u_1, ..., u_N)$$

and cost functionals

$$J_i(u_1, ..., u_N) = \int_0^T g_i(t, x(t), u_1, ..., u_N)dt$$

where $x \in \mathbb{R}^n$ is the system state and $u_i \in \mathbb{R}^{m_i}$ is the control input of player i.
A strategy γ_i is a function that determines what player i is going to play any given moment in $t \in [0, T]$ according to their surroundings.

$$u_i(t) = \gamma_i(t, x(t))$$

Namely, a strategy is a function $\gamma_i : [0, T] \times \mathbb{R}^n \to \mathbb{R}^{m_i}$. Our objective is to compute optimal feedback Nash equilibrium strategies γ_i^*, such that no player can reduce their cost J_i by unilaterally deviating from their strategy. Formally, this is expressed by the inequalities:

$$J_i(\gamma_1^*; ...; \gamma_{i-1}^*; \gamma_i^*; \gamma_{i+1}^*; ...; \gamma_N^*) \leq J_i(\gamma_1^*; ...; \gamma_{i-1}^*; \gamma_i; \gamma_{i+1}^*; ...; \gamma_N^*),\ \forall i \in \{1, ..., N\}$$

Finding a global Nash equilibrium in a non-linear system is computationally intractable. Therefore, we adopt the iterative Linear Quadratic (iLQ) approach, which repeatedly constructs local linear-quadratic approximations of the dynamics and costs, enabling efficient numerical solution of the game in the neighborhood of a nominal trajectory.

1.2.2 Local Approximation

This section describes the initialization and the first steps of each iteration of the algorithm proposed in [3].

Given an initial feedback strategy $\{\gamma_i^0\}$ for each player i, and an initial state $x(0)$, the algorithm performs forward integration and yields a reference trajectory ξ^k. This trajectory describes the evolution of both the system and the players for $t \in [0, T]$. Specifically, it is defined as:

$$\xi^k = \{\hat{x}(t), \hat{u}_{1:N}(t)\}$$

where

- $\hat{x}(t)$ is the trajectory of system's state
- $\hat{u}_{1:N}(t)$ are the control inputs of the players $i = 1, ..., N$ every moment.

This forward integration is implemented in three steps:

1. Initialization $x(0)$
2. At each time instant t, the players determine their control inputs according to the feedback law $\hat{u}_i(t) = \gamma_i^k(t, x(t))$.
3. We solve the differential equation $\dot{x} = f(x, u_{1:N}, t)$ using a numerical method (here Runge-Kutta).

After the initialization, the algorithm's next step is to obtain a Jacobian linearization of the dynamics f about trajectory ξ^k [5].

$$\dot{\delta}x(t) \approx A(t)\delta x(t) + \sum_{i\in[N]} B_i(t)\delta u_i(t)$$

where $A(t)$ is the Jacobian $D_{\hat{x}} f(t, \hat{x}(t), \hat{u}_{1:N}(t))$ and $B_i(t)$ is the Jacobian $D_{\hat{u}_i} f(t, \hat{x}(t), \hat{u}_{1:N}(t))$
A quadratic approximation of the quadratic cost is also obtained:

$$g_i(t, x(t), u_{1:N}(t)) \approx g_i(t, \hat{x}(t), \hat{u}_{1:N}(t)) + \frac{1}{2}\delta x(t)^T(Q_i(t)\delta x(t) + 2l_i(t)) +$$
$$+\frac{1}{2}\sum_{j\in[N]} \delta u_j(t)^T(R_{ij}(t)\delta u_j(t) + 2r_{ij}(t))$$

where $l_i(t) = \nabla_{\hat{x}} g_i$, $r_{ij} = \nabla_{\hat{u}_j} g_i$ and matrices Q_i and R_{ij} are Hessians $D^2_{\hat{x}\hat{x}} g_i$ and $D^2_{\hat{u}_j\hat{u}_j} g_i$. The mixed partials (Hessians) are omitted because they rarely appear in cost functions of practical interest.
The linearized dynamics and the quadratic approximation of the cost together constitute a Linear–Quadratic (LQ) game, analogous to the formulation described in the background section. Consequently, its solution can be obtained by solving a system of coupled Riccati differential equations, as presented in [1].
After each iteration, we update our proposed strategies:

$$\gamma_i^k(t, x(t)) = \hat{u}_i(t) - P_i^k(t)\delta x(t) - \eta\alpha_i^k(t)$$

where P_i^k is the solution of the coupled Riccati differential equation at iteration k (LQ games' Nash equilibrium feedback control was $u_i^* = R_{ii}^{-1} B_i^T P_i x$) and α_i^k is an affine term related to minimization of Hamiltonian $\left(\frac{\partial H_i}{\partial u_i}\right)$. Parameter η is a step size that we choose performing line search. Its purpose is to improve the algorithm's convergence.

The structure of the proposed algorithm is summarized in the flowchart below.

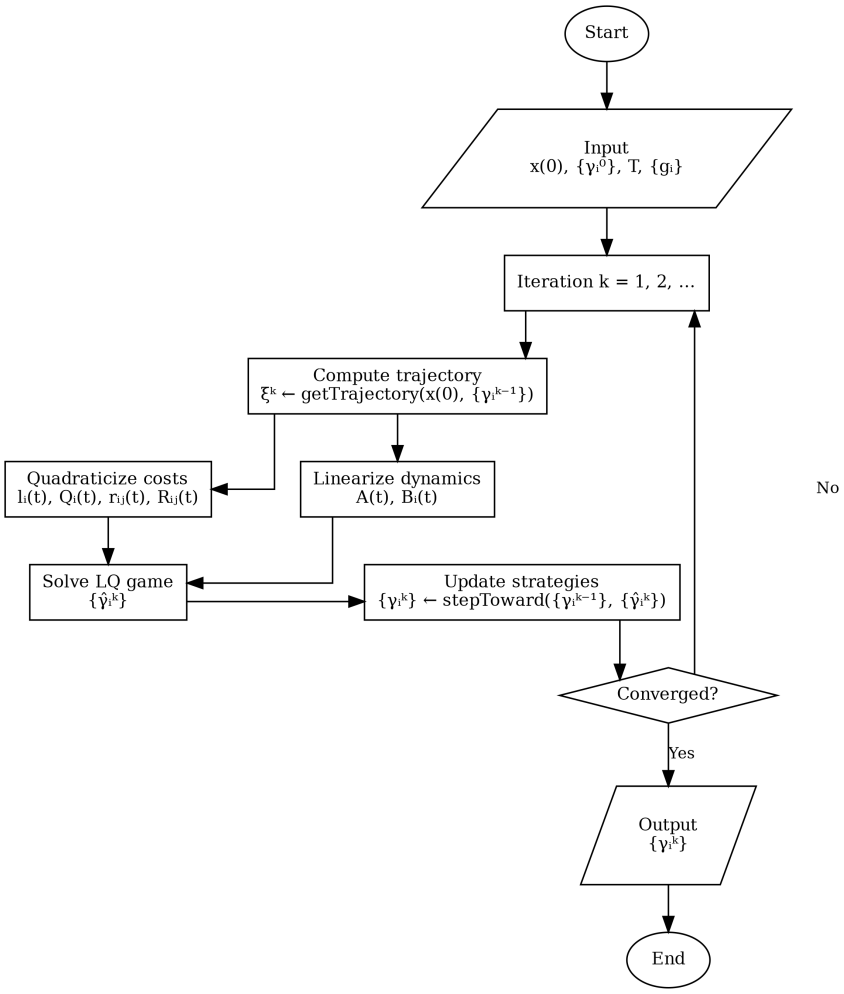

Figure 1: iLQG Flowchart

1.3 Computational Complexity

According to [3], linearization and cost approximation require $O(n^2)$ calculation at each timestep. The coupled Riccati differential equations solution has complexity $O(n^3)$.

2 Result Reproduction

2.1 Three-player Intersection

In this section, we are going to reproduce paper's [3] results using a similar scenario to the three player intersection.

2.1.1 Problem Description

The scenario consists of 2 cars and a pedestrian that must cross and intersection avoiding collisions. The problem's time horizon is $T = 5s$ and the time discretization is $\Delta t = 0.1s$.

2.1.2 System Dynamics

The three cars' dynamics are modeled after kinematic bicycle model:

$$\dot{p}_{x,i} = v_i \cos(\theta_i), \quad \dot{p}_{y,i} = v_i \sin(\theta_i), \quad \dot{\theta}_i = \frac{v_i \tan(\phi_i)}{L_i}, \quad \dot{\phi}_i = \psi_i, \quad \dot{v}_i = a_i$$

where $p_{x,i},\ p_{y,i}$ are each car's position coordinates, θ_i is each car's heading angle, u_i the linear velocity and ϕ_i the front wheel's steering angle. L_i is the inter-axle distance, and input $u_i = (\psi_i, \alpha_i)$ is the front wheel angular rate and longitudinal acceleration, respectively

The pedestrian is modeled after the kinematic unicycle model:

$$\dot{p}_x = v \cos(\theta),\ \dot{p}_y = v \sin(\theta),\ \dot{\theta} = \omega,\ \dot{v} = a$$

where $p_x,\ p_y$ are the pedestrian's position coordinates, θ their heading angle and u their linear velocity. Pedestrian's control input is $u_{ped} = [\omega, \alpha]^T$ (angular velocity and longitudinal acceleration).

Thus, our system is described by a 14-D state space model. There are 5 state variables for each car and 4 for the pedestrian.

2.1.3 Cost Functions

Each player's cost is a weighted sum of penalties for:

- *Lane Center and Boundary:* Penalty for deviation from the center and exit of the lane
- *Nominal Speed and Speed Bounds:* Enforcement of velocity $u_{ref,i}$ and respect of speed limits
- *Proximity Cost:* Avoid close proximity of the players, because that can result to a collision.

2.1.4 Our Scenario and Results

In our simulation we defined a reduced system with two players. Those agents' dynamics adhere to bicycle kinematic model (they are cars). Thus, our system is 10-D. The two lanes are straight lines for implementation simplicity. The objective is crossing the intersection without collisions.

Our simulation was implemented utilizing [2], a software library in Julia developed to solve games described by paper [3].
There, one can defined the system's dynamics as presented in the code snippet from file *intersection.jl* below:

```
using iLQGames
import iLQGames: dx, xyindex

nx, nu, dt, game_horizon = 10, 4, 0.1, 500
L1, L2 = 2.5, 2.9 # wheelbases
```

```
struct Intersection <: ControlSystem{dt, nx, nu} end

dx(cs::Intersection, x, u, t) = SVector(
    # Car1 (Compact) dynamics - Bicycle kinematic model
    # [px, py, θ, v, ψ]
    x[4]cos(x[3]),
    x[4]sin(x[3]),
    x[4]*tan(x[5])/L1,
    u[2],
    u[1],
    # Car2 (SUV) dynamics - Bicycle kinematic model
    # [px, py, θ, v, ψ]
    x[9]cos(x[8]),
    x[9]sin(x[8]),
    x[9]*tan(x[10])/L2,
    u[4],
    u[3]
)

# Specify players' positions
xyindex(::Intersection) = ((1,2), (6,7))
```

The cost functions that we implemented the ones presented in [3]. Our simulation parameters are:

- $d_{prox} = 0.5$
- $t_{goal} = 2.0$
- $d_{lane} = 1.5$

Our cars have different size specified through their wheelbase, 2.5 and 2.9 meters respectively.

Here is their resulting trajectory:

The corresponding gif file illustrates that there is no collision between our agents. Our algorithm indeed converges to a Nash equilibrium solving this simple game.

3 Experiments

In the previous section we proved that our algorithm is functional and reproduced a reduced result from paper [3]. Now we designed 2 more experiments to demonstrate the effect of Nash equilibria on our players' strategies and examine the robustness of our method.

3.1 Human-Robot Game 1

We consider a three player game. One of the players is a robot (blue agent) and the other two are human (red and green). Our players' dynamics are described by the following model:

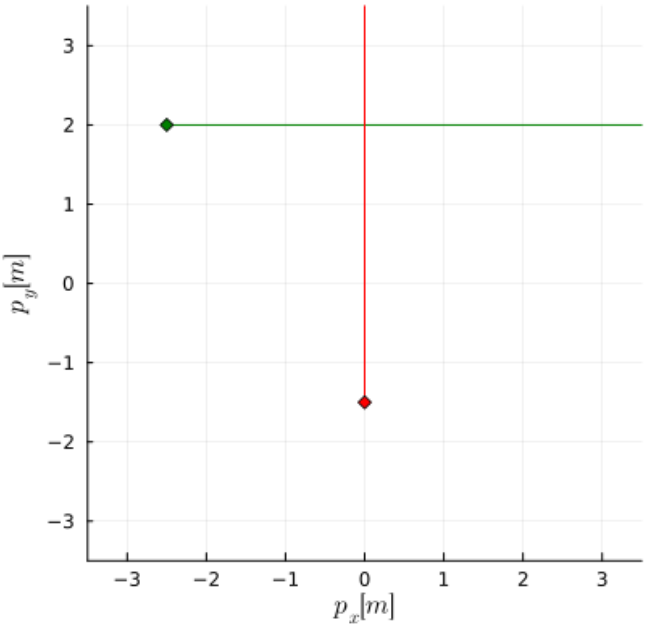

Figure 2: Two player intersection

$$\dot{p}_{x,i} = v_i \cos\theta_i,\ \dot{p}_{y,i} = v_i \sin\theta_i,\ \dot{u}_i = \alpha_i,\ \dot{\theta}_i = \omega_i$$

and our control input is:

$$u_i = [\alpha_i, \omega_i]$$

The players' initial positions are:

$$p_{0,1} = [-3, -3],\ p_{0,2} = [-3, 3],\ p_{0,3} = [3, -3]$$

and their goal positions are:

$$p_{g,1} = [0, 0],\ p_{g,2} = [3, -3],\ p_{g,3} = [-3, 3]$$

The cost function penalize distance from the target and proximity to other players. Specifically our cost functions are (for every player):

- $\mathbb{1}\{d_{ij} < d_{prox}\}(d_{prox} - d_{ij})^2$
- d_{goal}^2
- $u_i^T R u_i$

where d_{ij} is the distance between two different players and d_{goal} is the distance between the player and their goal. We are using a proximity threshold, $d_{prox} = 0.5$.

Suppose that we linearize the dynamics and solve a classic optimal time control problem. Our solution is given by the bang-bang principle. Since every player's distance from the center is the same, they would collide. Our algorithm, utilizes the proximity cost and the Nash equilibrium calculation results to the avoidance of the collision. In fact, the robot slows down waiting both human to cross paths (without colliding) and then it reaches its destination. Here is a static plot of our results:

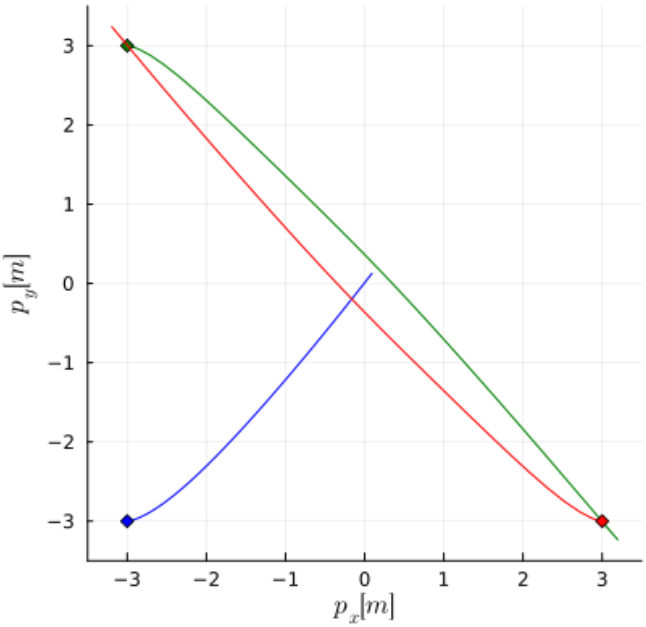

Figure 3: Robot-human experiment 1

3.2 Human-Robot Game 2

We confirmed that iLQG algorithm converges for two different scenarios. In the second one, we demonstrated the potential of our method. Now we are going to examine its robustness.
In this experiment, the players are characterized by identical dynamics and costs. However, at three random time instances, three random additional accelerations are introduced, distinct from the pre-calculated control input. The experiment's result is the following:

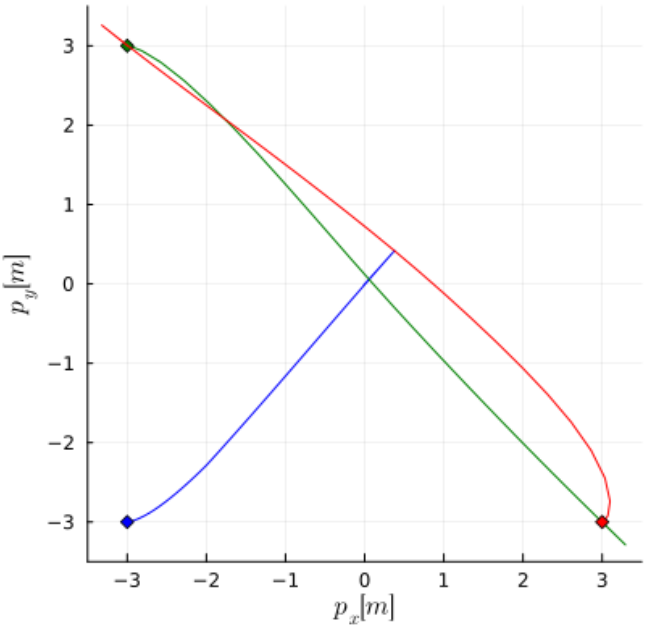

Figure 4: Robot-human experiment 2

This scenario showcases that iLQG can handle cases were there are unpredictable actions from the players. It computes different appropriate control inputs and manages to converge to a new

Nash equilibrium despite the receding time horizon.

4 Discussion and Limitations

4.1 Simulation Implementation

Our simulations were implemented in Julia. There is another software package in C++ developed by the authors. We opted for the first one because it offers a simpler interface for the definition of general differential games different than the ones from the paper. Our simulations' code and gifs can be found at https://github.com/orion-3464/Differential-Games-iLQG.

4.2 Limitations

According to [3] there is no theoretical guarantee of convergence to a Nash equilibrium from arbitrary initializations. Moreover, the dynamics' linearization may perform poorly in extreme cases of non-linearity and big disturbances.

References

[1] Tamer Başar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory, 2nd Edition*. Society for Industrial and Applied Mathematics, 1998.

[2] David Fridovich-Keil, Ellis Ratner, and Lasse Peters. iLQGames.jl: A julia package for solving multi-player differential games, 2020. Software available from https://github.com/JuliaGameTheoreticPlanning/iLQGames.jl.

[3] David Fridovich-Keil, Ellis Ratner, Lasse Peters, Anca D. Dragan, and Claire J. Tomlin. Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games, 2020.

[4] Daniel Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, Princeton, NJ, 2012.

[5] Richard M. Murray. Cds 101: Introduction to feedback control systems. `https://www.cds.caltech.edu/~murray/courses/cds101/fa02/caltech/pph.html`, 2002. Accessed: 2025-01-17.

[6] Alan W. Starr and Y. C. Ho. Nonzero-sum differential games. *Journal of Optimization Theory and Applications*, 3:184–206, 1969.