A Preview of Open-Loop and Feedback Nash Trajectories in Racing Scenarios*

Matthias Rowold[1]

Abstract— **Trajectory planning for autonomous race cars poses special challenges due to the highly interactive and competitive environment. Prior work has applied game theory as it provides equilibria for such non-cooperative dynamic problems. This contribution introduces a framework to assess the suitability of the Nash equilibrium for racing scenarios. To achieve this, we employ a variant of iLQR, called iLQGame, to find trajectories that satisfy the equilibrium conditions for a linear-quadratic approximation of the original game. In particular, we are interested in the difference between the behavioral outcomes of the open-loop and the feedback Nash equilibria and show how iLQGame can generate both types of equilibria. We provide an overview of open problems and upcoming research, including convergence properties of iLQGame in racing games, cost function parameterization, and moving horizon implementations.**

I. Introduction

The complexity of considering interactions between autonomous vehicles and their interactions with human agents presents a significant challenge in trajectory planning. In established sequential methods, the autonomous vehicle of interest – hereafter referred to as the ego vehicle – is concerned with predicting the most likely trajectories of all relevant agents to react with a collision-free trajectory. This is often sufficient, as applications have shown. However, sequential approaches neglect the reciprocal nature of scenarios, meaning the other agents respond to the executed motion of the ego vehicle, creating a bidirectional interdependence. Planning approaches that consider and incorporate this mutual dependency are categorized as interaction-aware.

Interaction-aware approaches promise to generate trajectories with a lesser degree of conservatism compared to sequential approaches. This means they are performant and human-like even in environments with rapidly increasing prediction uncertainties that, under a sequential approach, would lead to overly cautious trajectories. By leveraging the knowledge that other agents react to the ego vehicle, including collision avoidance, interaction-aware approaches can influence the other agents' behaviors to a certain extent to achieve more progressive and less risk-averse behaviors. An application-oriented goal of interaction-aware planning is to generate behaviors that are seamlessly integrateable into traffic scenarios like lane changes [1], ramp merges [2], or crosswalks [3], [4]. In addition to traffic, autonomous racing is another domain that heavily relies on interactions. In racing, strategies like overtaking, blocking, and faking are common, requiring anticipating the opponent's reaction to the ego-trajectory to be successful and safe. A major distinction to traffic scenarios is that the desired behavior in racing is usually competitive, i.e., non-cooperative.

Interaction-aware planning approaches employ multi-agent planning with a joint cost function, partially observable Markov desicion processess, reinforcement learning, and game-theoretical concepts. The latter seems especially fitting for autonomous racing since game theory provides concepts for non-cooperative behaviors in environments where the agents cannot communicate. Furthermore, the objective of each agent is similar and known. The goal is to maximize speed and be ahead of the opponents, in contrast to traffic scenarios with a wide range of objectives. Given that game-theoretic concepts require assumptions about the cost functions that govern the agent's decisions, their use for racing seems appropriate.

In this and the following work, we will analyze the suitability of a game-theoretic concept, the Nash equilibrium, for trajectory planning in autonomous racing. Our focus lies on two types of Nash equilibria: the open-loop and the feedback equilibria. Both types have been considered in previous work, but they have not been compared regarding their behavioral outcomes.

A. Related Work

Most game-theoretic planning approaches in traffic and racing scenarios are concerned with finding trajectories that fulfill the requirements of a Nash equilibrium. At a Nash equilibrium, no agent, in the following called player, has an incentive to alter its strategy unilaterally. Depending on the information structure of the formulated game, one obtains either the open-loop or the feedback solution. Each player has to commit to a sequence of control inputs at the beginning of the game for an open-loop solution. In contrast, for a feedback solution, the players look for strategies that allow them to react to the current state in each stage of the game. A more detailed introduction to these concepts will follow in Section III.

We categorize the existing approaches for game-theoretic trajectory planning into the following groups:

1) Offline policy generation: Fisac et al. [5] discretize the state space and determine the optimal policy for a feedback Stackelberg equilibrium offline via dynamic programming. This policy can be applied efficiently only, but the offline calculations suffer from the curse of dimensionality, so only a few players and coarse discretizations are possible. Bhargav

*This work was not supported by any organization

[1]Matthias Rowold is with the Chair of Automatic Control, Department of Mechanical Engineering, TUM School of Engineering and Design, Technical University of Munich, 85748 Garching matthias.rowold@tum.de

et al. [6] perform extensive offline computations as well. However, they do not solve for equilibria, but policies with a high probability of successful overtaking for different race track positions.

Zheng et al. [7] formulate racing as a two-player zero-sum game in extensive form and determine the optimal strategy via counterfactual regret minimization.

2) Sampling-based: Liniger and Lygeros [8] formulate bi-matrix games by sampling trajectory candidates for two players. Solving these games for a Nash equilibrium results in open-loop trajectories. Feedback is introduced when the planning is performed with a receding horizon.

3) Iterative best response (IBR): In IBR approaches, the players optimize their trajectories alternately while keeping all other players' trajectories fixed. If this algorithm converges, no player is incentivized to alter its decision, making it a Nash equilibrium. Sensitivity-enhanced algorithms have been proposed in [9]–[11] for drone and vehicle racing. Since the trajectories are optimized as a whole, the result is an open-loop equilibrium.

4) Differential dynamic programming (DDP): DDP [12] is a trajectory optimization method that iteratively performs backward- and forward passes to refine the trajectory. During the backward pass, an incremental control law is generated based on second-order approximations of the cost and dynamics along a nominal trajectory. The forward pass updates the nominal trajectory based on the incremental control law. Using only a first-order approximation of the dynamics results in iLQR [13].

Fridovich-Keil et al. [3] transfer this iterative procedure to dynamic games. They approximate each player's cost function with a second-order tailor expansion and linearize the dynamics. The result is a linear quadratic game for which – like for time-discrete linear-quadratic regulators (LQRs) – analytic solutions exist [14]. If this algorithm, called iLQGame, converges, a Nash equilibrium to a local approximation of the game is found. Since iLQGame provides feedback strategies for each player, the solution constitutes a feedback Nash equilibrium. Similarly, Schwarting et al. [15] solve a quadratic game in the backward pass to compute incremental feedback laws for the players. However, they plan in belief space, making it a multi-player variant of iterative linear-quadratic Gaussian control.

Kavuncu et al. [16] show that their used cost function constitutes a potential game so that the problem can be reformulated as a conventional optimal control problem (OCP). Using iLQR to solve the OCP, they find an open-loop Nash equilibrium.

5) First-order optimality condition: ALGames by Le Cleac'h et al. [2] solve a root-finding problem to fulfill the first-order optimality condition of a Nash equilibrium. They enforce constraints with an augmented Lagrangian method and obtain a local open-loop Nash equilibrium with reported superior computation times compared to iLQGame. Zhu and Borrelli [17] develop an sequential quadratic programming variant to find a Nash equilibrium as a solution to the Karush–Kuhn–Tucker conditions. As in [2], the algorithm, if it converges, finds an open-loop equilibrium.

II. Scope

Some of the above approaches are compared regarding their calculation times [2] and convergence success rates [17]. We, however, are interested in their behavioral outcome and performance in racing scenarios. We focus on the comparison of open-loop and feedback solutions since the two types of equilibria can lead to entirely different solutions, as shown in Starr and Ho [18].

With this contribution, we propose a framework to assess both concepts in racing scenarios. We identify the iLQGame approach as a suitable method for finding open-loop and feedback Nash equilibria. Although only the feedback case is analyzed in [3], an adaption allows the approach to find open-loop equilibria. The adaption does not require altering the cost function or changing the fundamental working of the algorithm so that differences in the solutions due to different cost functions can be ruled out. This ensures the comparability of different solutions caused by the type of equilibrium.

In the following, we will first provide game-theoretic preliminaries and introduce the two types of equilibria. Section IV explains the iLQGame algorithm, and Section V formulates our racing game with its dynamics and the players' cost functions. In Section VI, we will show exemplary results of open-loop and feedback trajectories to illustrate the necessity of a more detailed examination.

III. Game-Theoretic Preliminaries

The dynamics describing the propagation of the joint state $\boldsymbol{x}_k$ for a dynamic game with N players is given by:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_k\left(\boldsymbol{x}_k, \boldsymbol{u}_k^1, \dots, \boldsymbol{u}_k^N\right), \tag{1}$$

where $\boldsymbol{x}_k \in \mathcal{X} = \mathbb{R}^n$ and $\boldsymbol{u}_k^i \in \mathcal{U}^i = \mathbb{R}^m$. We consider K-stage games with an initial state $\boldsymbol{x}_0$, where the stage cost for player $i \in \mathcal{N} = \{1, 2, \dots, N\}$ depends on player i's control inputs $\boldsymbol{u}_k^i \in \boldsymbol{u}^i = \left\{\boldsymbol{u}_0^i, \boldsymbol{u}_1^i, \dots, \boldsymbol{u}_{K-1}^i\right\}$ and the state $\boldsymbol{x}_k$. The sequence of states depends on $\boldsymbol{u}^i$ and the control inputs of all other players, which is often expressed with the index $-i$. Hence, the total cost of player i depends on the initial state and all players' inputs:

$$J^i\left(\boldsymbol{x}_0, \boldsymbol{u}^i, \boldsymbol{u}^{-i}\right) = \sum_{k=0}^{K-1} g_k^i\left(\boldsymbol{x}_k, \boldsymbol{u}_k^i\right) + g_K^i\left(\boldsymbol{x}_K\right). \tag{2}$$

A strategy $\gamma^i = \left\{\boldsymbol{\gamma}(\cdot)_0^i, \boldsymbol{\gamma}(\cdot)_1^i, \dots, \boldsymbol{\gamma}(\cdot)_{K-1}^i\right\}$ of the strategy space $\Gamma^i = \left\{\Gamma_0^i, \Gamma_1^i, \dots, \Gamma_{K-1}^i\right\}$ determines the control inputs for each stage k, depending on the available information to player i. The cost functional (2) expressed with strategies is:

$$J^i\left(\boldsymbol{x}_0, \gamma^i, \gamma^{-i}\right) = \sum_{k=0}^{K-1} g_k^i\left(\boldsymbol{x}_k, \boldsymbol{\gamma}_k^i(\cdot)\right) + g_K^i\left(\boldsymbol{x}_K\right). \tag{3}$$

We omit the dependency on $\boldsymbol{x}_0$ for brevity in the following. An N-tuple of strategies $\left\{\gamma^{i*} \in \Gamma^i; i \in \mathcal{N}\right\}$ constitutes a Nash equilibrium if:

$$\forall i \in \mathcal{N} : J^i\left(\gamma^{i*}, \gamma^{-i*}\right) \leq J^i\left(\gamma^i, \gamma^{-i*}\right) \tag{4}$$

Loosely speaking, no player can improve its outcome at a Nash equilibrium by unilaterally altering its strategy.

The domain and codomain of the functions in the strategy space depend on the information structure of the game [14]. The two information structures we consider lead to the following two types of equilibria:

1) Open-loop Nash equilibrium: In the open-loop case, all players observe the initial state $\boldsymbol{x}_0$ and generate a sequence of control inputs in a single act. This means, the strategy at stage k in (3) is a constant function with $\boldsymbol{\gamma}_k^i(\cdot) \in \Gamma_k^i = \mathcal{U}^i$. A Nash equilibrium $\left\{\gamma^{i*} \in \Gamma^i; i \in \mathcal{N}\right\}$ therefore directly translates to the players' input sequences $\left\{\boldsymbol{u}^{i*} = \gamma^{i*}; i \in \mathcal{N}\right\}$. A forward simulation of (1) beginning with $\boldsymbol{x}_0^* = \boldsymbol{x}_0$ provides the corresponding open-loop state trajectory $\left\{\boldsymbol{x}_{k+1}^*; k \in \{0, 1, \ldots, K-1\}\right\}$.

The open-loop problem in discrete time can be seen as a static infinite game, i.e., a game with infinite possible control input sequences of which one has to be chosen at the first and only stage $k = 0$ [14].

2) Feedback Nash equilibrium: If the players know the current state $\boldsymbol{x}_k$, they can react to it and are not bound to an initially set sequence of control inputs. A feedback strategy $\gamma_k^i : \mathcal{X} \to \mathcal{U}^i$ maps the state to a control input $\boldsymbol{u}_k^i$ so that the control inputs at a stage k corresponding to a Nash equilibrium are: $\left\{\boldsymbol{u}_k^{i*} = \gamma_k^{i*}(\boldsymbol{x}_k); i \in \mathcal{N}\right\}$. Such strategies can be calculated via dynamic programming, i.e., by working backward for k from K to 0 and determining a Nash equilibrium for each static sub-game from stage k to $k+1$.

In OCPs, which correspond to games with $N = 1$ and only one cost-functional J^1, the trajectory obtained by simulating (1) and applying the feedback solution in each stage coincides with the open-loop solution. This, however, does not apply to Nash equilibria of non-zero-sum games with $N > 1$, even in the absence of disturbances or other unpredictable inputs. Starr and Ho [18] provide an illustrative example of this phenomenon and further examinations.

IV. Solving Discrete-Time Dynamic Games

The iLQGame approach in [3] generates time-variant linear feedback laws for the players, yielding a feedback Nash equilibrium solution. However, iLQGame can be adapted to generate an open-loop solution as done in the supplementary material of [3][1]. In the following, we recapitulate the procedure which is summarized in Algorithm IV.

Beginning with an initial state $\boldsymbol{x}_0$ and an initial guess for each player's control input sequence $\hat{\boldsymbol{u}}^i$, the initial nominal trajectory $\hat{\boldsymbol{x}}$ is obtained with (1).

1) Linearization of the dynamics: A linearization along the nominal trajectory provides the dynamic and input matrices for each time step and player. The resulting linear time-variant system is:

$$\begin{aligned} \Delta\boldsymbol{x}_{k+1} &= \boldsymbol{A}_k \Delta\boldsymbol{x}_k + \sum_{i=1}^{N} \boldsymbol{B}_k^i \Delta\boldsymbol{u}_k^i \text{ with} \\ \Delta\boldsymbol{x}_k &= \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k \text{ and } \Delta\boldsymbol{u}_k^i = \boldsymbol{u}_k - \hat{\boldsymbol{u}}_k^i. \end{aligned} \tag{5}$$

2) Quadratization of the cost function: As in iLQR, the stage cost is approximated by a second-order Taylor series:

$$\begin{aligned} g_k^i\left(\Delta\boldsymbol{x}_k, \Delta\boldsymbol{u}_k^1, \ldots, \Delta\boldsymbol{u}_k^N\right) \approx\ & g_k^i\left(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{u}}_k^1, \ldots, \hat{\boldsymbol{u}}_k^N\right) + \\ & \underbrace{\left(\nabla_{\mathbf{x}} g_k^i|_{\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{u}}_k^1, \ldots, \hat{\boldsymbol{u}}_k^N}\right)^\top}_{\boldsymbol{q}_k^i} \Delta\boldsymbol{x}_k + \\ & \frac{1}{2}\Delta\boldsymbol{x}_k^\top \underbrace{\left(\nabla_{\mathbf{x}}^2 g_k^i|_{\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{u}}_k^1, \ldots, \hat{\boldsymbol{u}}_k^N}\right)}_{\boldsymbol{Q}_k^i} \Delta\boldsymbol{x}_k + \\ & \underbrace{\left(\nabla_{\mathbf{u}^i} g_k^i|_{\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{u}}_k^1, \ldots, \hat{\boldsymbol{u}}_k^N}\right)^\top}_{\boldsymbol{r}_k^{ii}} \Delta\boldsymbol{u}_k^i + \\ & \frac{1}{2}\Delta\boldsymbol{u}_k^{i\top} \underbrace{\left(\nabla_{\mathbf{u}^i}^2 g_k^i|_{\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{u}}_k^1, \ldots, \hat{\boldsymbol{u}}_k^N}\right)}_{\boldsymbol{R}_k^{ii}} \Delta\boldsymbol{u}_k^i. \end{aligned} \tag{6}$$

Here, we omit the mixed second-order terms since they do not appear in our cost function. Using the notation in (6) and omitting the constant term, the total cost for player i is approximated by:

$$\begin{aligned} J^i \propto \frac{1}{2} \sum_{k=0}^{K-1} \Bigg[& \left(\Delta\boldsymbol{x}_k^\top \boldsymbol{Q}_k^i + 2\boldsymbol{q}_k^{i\top}\right) \Delta\boldsymbol{x}_k + \\ & \sum_{j=1}^{N} \left[\left(\Delta\boldsymbol{u}_k^{j\top} \boldsymbol{R}_k^{ij} + 2\boldsymbol{r}^{ij\top}\right)\right] \Delta\boldsymbol{u}_k^j \Bigg] + \\ & \frac{1}{2}\Delta\boldsymbol{x}_K^\top \boldsymbol{Q}_K^i \Delta\boldsymbol{x}_K + \boldsymbol{q}_K^{i\top} \Delta\boldsymbol{x}_K. \end{aligned} \tag{7}$$

In our case, the mixed terms $\boldsymbol{R}_k^{ij}$ and $\boldsymbol{r}_k^{ij}$ with $i \neq j$ will be 0. The cost structure (7) and the linear dynamics (5) constitute a LQ game that approximates the original game locally around the current nominal trajectory $\hat{\boldsymbol{x}}$.

Algorithm 1 iLQGame

Input: $\boldsymbol{x}_0$, initial trajectory $\hat{\boldsymbol{u}}^i$ and $\hat{\boldsymbol{x}}$
Output: Nash equilibrium trajectory $\boldsymbol{u}^{i*}$ and $\boldsymbol{x}^*$
while not converged **do**
  for $k \in \{0, 1, \ldots, K\}$ **do**
    $\boldsymbol{A}_k, \boldsymbol{B}_k^i \leftarrow \textsc{Linearize}(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{u}}_k^1, \ldots, \hat{\boldsymbol{u}}_k^N)$
    $\boldsymbol{Q}_k^i, \boldsymbol{q}_k^i, \boldsymbol{R}_k^{ii}, \boldsymbol{r}_k^{ii} \leftarrow \textsc{Quadratize}(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{u}}_k^1, \ldots, \hat{\boldsymbol{u}}_k^N)$
  $\boldsymbol{K}_k^i, \boldsymbol{k}_k^i \leftarrow \textsc{SolveLQGame}(A_k, B_k^i, Q_k^i, q_k^i, R_k^{ii}, r_k^{ii})$
  for $k \in \{0, 1, \ldots, K\}$ **do**
    $\hat{\boldsymbol{u}}_k^{i,\text{new}} \leftarrow \textsc{UpdateInput}(\hat{\boldsymbol{x}}_k^{\text{new}}, \boldsymbol{K}_k^i, \boldsymbol{k}_k^i)$
    $\hat{\boldsymbol{x}}_{k+1}^{\text{new}} = \boldsymbol{f}_k\left(\hat{\boldsymbol{x}}_k^{\text{new}}, \hat{\boldsymbol{u}}_k^{1,\text{new}}, \ldots, \hat{\boldsymbol{u}}_k^{N,\text{new}}\right)$
  $\hat{\boldsymbol{u}}^i \leftarrow \hat{\boldsymbol{u}}^{i,\text{new}}$, $\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}}^{\text{new}}$
$\boldsymbol{u}^{i*} \leftarrow \hat{\boldsymbol{u}}^i$, $\boldsymbol{x}^* \leftarrow \hat{\boldsymbol{x}}$

[1] https://github.com/HJReachability/ilqgames

3) Solving the LQ game: For the LQ game above, an analytical solution for the strategy of a feedback Nash equilibrium exists and has the linear affine form $\gamma_k^{i*}(\Delta \boldsymbol{x}_k) = -\boldsymbol{K}_k^i \Delta \boldsymbol{x}_k - \boldsymbol{k}_k^i$ [14]. The elements in the matrices $\boldsymbol{K}_k^i$ and vectors $\boldsymbol{k}_k^i$ are obtained by solving the following systems of linear equations [3], [14]:

$$\begin{aligned}\left(\boldsymbol{R}_k^{ii} + \boldsymbol{B}_k^{i\top} \boldsymbol{P}_{k+1}^i \boldsymbol{B}_k^i\right) \boldsymbol{K}_k^i + \boldsymbol{B}_k^{i\top} \boldsymbol{P}_{k+1}^i \cdot \\ \sum_{j=1, j\neq i}^{N} \left[\boldsymbol{B}_k^j \boldsymbol{K}_k^j\right] = \boldsymbol{B}_k^{i\top} \boldsymbol{P}_{k+1}^i \boldsymbol{A}_k \end{aligned} \tag{8a}$$

$$\begin{aligned}\left(\boldsymbol{R}_k^{ii} + \boldsymbol{B}_k^{i\top} \boldsymbol{P}_{k+1}^i \boldsymbol{B}_k^i\right) \boldsymbol{k}_k^i + \boldsymbol{B}_k^{i\top} \boldsymbol{P}_{k+1}^i \cdot \\ \sum_{j=1, j\neq i}^{N} \left[\boldsymbol{B}_k^j \boldsymbol{k}_k^j\right] = \boldsymbol{B}_k^{i\top} \boldsymbol{p}_{k+1}^i + \boldsymbol{r}_k^{ii} \end{aligned} \tag{8b}$$

As for linear-quadratic OCPs in discrete time, the matrices $\boldsymbol{P}_k^i$ and vectors $\boldsymbol{p}_k^i$ can be obtained by a recursion, which is given in Appendix VII-A.

For the open-loop Nash equilibrium, the strategy $\gamma_k^{i*}(\cdot) = -\boldsymbol{k}_k^i$ does not depend on the current state and can be obtained with [3], [14]:

$$\boldsymbol{k}_k^i = -\boldsymbol{R}_k^{ii^{-1}} \left[\boldsymbol{B}_k^{i\top} \left(\boldsymbol{M}_{k+1}^i \Delta \boldsymbol{x}_{k+1} + \boldsymbol{m}_{k+1}^i\right) + \boldsymbol{r}_k^{ii}\right] \tag{9a}$$

$$\begin{aligned}\Delta \boldsymbol{x}_{k+1} = \boldsymbol{\Lambda}_k^{-1} \Bigg[\boldsymbol{A}_k \Delta \boldsymbol{x}_k - \\ \sum_{j=1}^{N} \boldsymbol{B}_k^j \boldsymbol{R}_k^{jj^{-1}} \left(\boldsymbol{B}_k^{j\top} \boldsymbol{m}_{k+1}^j + \boldsymbol{r}_k^{jj}\right) \Bigg]. \end{aligned} \tag{9b}$$

Again, $\boldsymbol{M}_{k+1}^i$ and $\boldsymbol{m}_{k+1}^i$ are obtained by a recursion given in Appendix VII-B. Since both recursions proceed backward from K to 0, this step is often called the backward pass.

4) Update Trajectory: The forward pass updates the control inputs and nominal state trajectory according to the generated strategies. Due to the linearization of the dynamics and quadratization of the cost function along the trajectory, the obtained strategies are additive to the control inputs of the previous iteration. In the open-loop case, $\boldsymbol{K}_k^i$ is set to $\boldsymbol{0}$, and in the feedback case, it is applied on the difference from the previous iteration. Beginning with $\hat{\boldsymbol{x}}_0^{\text{new}} = \hat{\boldsymbol{x}}_0 = \boldsymbol{x}_0$ the forward pass is:
For k from 1 to K:

$$\hat{\boldsymbol{u}}_k^{i,\text{new}} = \hat{\boldsymbol{u}}_k^i - \boldsymbol{K}_k^i \left(\hat{\boldsymbol{x}}_k^{\text{new}} - \hat{\boldsymbol{x}}_k\right) - \eta \boldsymbol{k}_k^i \tag{10a}$$

$$\hat{\boldsymbol{x}}_{k+1}^{\text{new}} = \boldsymbol{f}_k\left(\hat{\boldsymbol{x}}_k^{\text{new}}, \hat{\boldsymbol{u}}_k^{1,\text{new}}, \ldots, \hat{\boldsymbol{u}}_k^{N,\text{new}}\right). \tag{10b}$$

The scalar parameter $0 < \eta \leq 1$ can be interpreted as a step size and is usually chosen much smaller than 1 to account for large deviations from the nominal trajectories where the approximations (5) and (7) do not hold. With the new trajectory, the above sequence of linearization, quadratization, backward pass, and forward pass repeat until the algorithm converges.

Fridovich-Keil et al. [3] point out that the resulting trajectory is not necessarily a Nash equilibrium of the original game. Instead, it represents a strategy that satisfies the conditions for a Nash equilibrium for a sequence of local approximations of the game.

V. Racing Game

A. Vehicle model and game dynamics

Each player is modeled by a point mass following [19] where the state includes the progress s, velocity V, lateral displacement n, relative orientation χ towards the track's reference line with the curvature $\kappa(s)$, and the longitudinal and lateral accelerations a_x and a_y. The control input vector includes the jerks in longitudinal and lateral directions: $\boldsymbol{u}^\top = \begin{bmatrix} j_\text{x} & j_\text{y} \end{bmatrix}$. The time-continuous nonlinear dynamics of player i are given by:

$$\dot{\mathbf{x}}^i = \begin{bmatrix} \dot{s}^i \\ \dot{V}^i \\ \dot{n}^i \\ \dot{\chi}^i \\ \dot{a}_\text{x}^i \\ \dot{a}_\text{y}^i \end{bmatrix} = \tilde{\mathbf{f}}^i(\mathbf{x}^i, \mathbf{u}^i) = \begin{bmatrix} \frac{V^i \cos(\chi^i)}{1 - n^i \kappa(s^i)} \\ a_\text{x}^i \\ V^i \sin(\chi^i) \\ \frac{a_\text{y}^i}{V^i} - \kappa(s^i) \frac{V^i \cos(\chi^i)}{1 - n^i \kappa(s^i)} \\ j_\text{x}^i \\ j_\text{y}^i \end{bmatrix}. \tag{11}$$

Since racing cars often operate at the handling limits, it is important to constrain the accelerations to obtain feasible trajectories. Similar to [19], we approximate the velocity-dependent gg-diagrams by diamonds with a maximum positive acceleration $a_\text{x} \leq a_{\text{x,max}}(V)$ and a maximum combined radius $\rho(V)$:

$$\sqrt{a_\text{x}^2 + a_\text{y}^2} \leq \rho(V). \tag{12}$$

The joint state vector of the game is a concatenation of N player state vectors:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{x}}^1 \\ \vdots \\ \dot{\boldsymbol{x}}^N \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{f}}^1(\mathbf{x}, \mathbf{u}^i) \\ \vdots \\ \tilde{\mathbf{f}}^N(\mathbf{x}, \mathbf{u}^N) \end{bmatrix} = \tilde{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u}^1, \ldots, \boldsymbol{u}^N) \tag{13}$$

B. Cost function

LQR approaches naturally do not consider state and input constraints. Chen et al. [20] realize constraints in iLQR through the cost function and introduce barrier functions. Quadratic cost terms for constraint violations in [3] show good results regarding convergence and robustness of the iLQGame algorithm. Our stage costs, including the constraints, are:

$$g_k^i = \boldsymbol{u}_k^{i\top} \boldsymbol{R}^i \boldsymbol{u}_k^i + \tag{14a}$$

$$\sum_{j=1, j\neq i}^{N} c_\text{c}^i \left(e^{1 - \left(\frac{s_k^i - s_k^j}{l_\text{veh}}\right)^2 - \left(\frac{n_k^i - n_k^j}{w_\text{veh}}\right)^2} \right)^2 + \tag{14b}$$

$$\mathbf{1}\left\{n_k^i \geq w_{\text{tr,l/r}}(s_k^i)\right\} c_\text{w}^i \left(n_k^i - w_{\text{tr,l/r}}(s_k^i)\right)^2 + \tag{14c}$$

$$\mathbf{1}\left\{a_{\text{x},k}^i \geq a_{\text{x,max}}(V_k^i)\right\} c_{a_\text{x}}^i \left(a_{\text{x},k}^i - a_{\text{x,max}}(V_k^i)\right)^2 + \tag{14d}$$

$$\mathbf{1}\left\{\sqrt{a_\text{x}^2 + a_\text{y}^2} \geq \rho(V)\right\} c_a^i \left(\sqrt{a_\text{x}^2 + a_\text{y}^2} - \rho(V)\right)^2. \tag{14e}$$

As in [3], we use the operator $\mathbf{1}\{\cdot\}$ which becomes 1 if the condition holds, and 0 otherwise. The term (14a) regularizes the jerk as in [19] and (14b) introduces a coupling between players by penalizing collisions. As player i and j

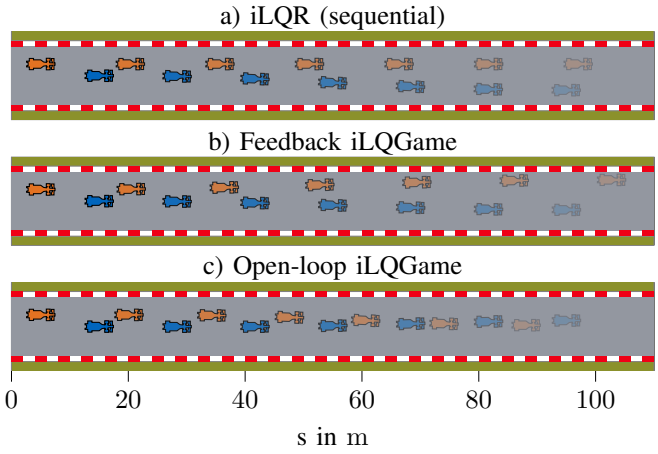

Fig. 1. Results for iLQR, open-loop iLQGame, and feedback iLQGame: All three methods use the same cost parameters and are initialized with $\hat{\boldsymbol{u}}^i = \mathbf{0}$.

come closer, the term increases with longitudinal and lateral distances weighted differently.

All other stage cost terms implement soft constraints with the weights c_{w}, $c_{a_{\mathrm{x}}}$, c_a, c_{c}. (14c) enforces the track boundaries with the track widths to the left and right $w_{\mathrm{tr,l/r}}$. Note that none of the state constraints depend on the inputs in $\boldsymbol{u}_k^i$, as this would result in mixed second-order terms in (6).

The terminal costs introduce another coupling and should provide the incentive to drive fast and to be ahead at the end of the planning horizon:

$$g_K^i = -s_K^i + c_{\mathrm{g}}^i \sum_{j=1, j\neq i}^{N} s_K^j \tag{15}$$

The first term penalizes little progress and the second term with the weight c_{g} should incentivize defending or blocking maneuvers. A similar terminal cost for racing is used in [9].

VI. Exemplary Results

This section provides examples to demonstrate the capability of iLQGame to consider interactions in racing scenarios and to motivate comparing the two types of Nash equilibria. The considered scenario in Figures 1 and 2 includes the ego vehicle ($i = 1$, blue) with a maximum velocity of $20\,\mathrm{m/s}$ and the opponent ($i = 2$, orange) with $25\,\mathrm{m/s}$. The opponent approaches the ego vehicle with $23\,\mathrm{m/s}$ and a lateral displacement of $2\,\mathrm{m}$. For the following results, we initialize the control input sequences with $\hat{\boldsymbol{u}}^i = \mathbf{0}$.

Figure 1 a) shows the trajectories obtained with a sequential approach. The opponent vehicle is predicted assuming a constant velocity and lateral displacement. With the fixed prediction, the iLQGame algorithm reduces to iLQR, and the resulting trajectory swerves to the right to avoid collisions. This scenario highlights the importance of interaction-aware planning since the observed yielding behavior is not desirable in competitive racing.

The feedback solution is shown in Figure 1 b). The right swerving maneuver of the ego vehicle occurs to a lesser

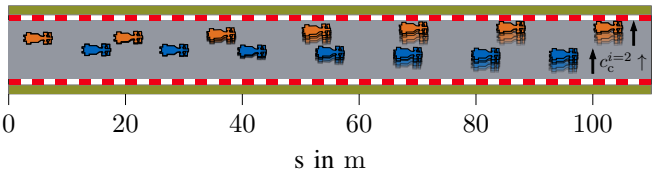

Fig. 2. Variation of $c_{\mathrm{c}}^{i=2}$ for feedback iLQGame: A greater ratio of $c_{\mathrm{c}}^{i=2}/c_{\mathrm{c}}^{i=1}$ causes the ego vehicle to be more aggressive, forcing the opponent to swerve further to the left.

degree due to its awareness that the opponent is also trying to avoid collisions. Increasing the collision cost weight of the opponent $c_{\mathrm{c}}^{i=2}$ as shown in Figure 2 results in a greater leveraging of the opponent's reaction so that the ego vehicle can maintain its course. The choice of $c_{\mathrm{c}}^{i=2} > c_{\mathrm{c}}^{i=1}$ can be justified assuming that the trailing vehicle bears a greater responsibility to avoid collisions.

Figure 1 c) shows the trajectories of the open-loop solution. The players' behaviors significantly differ from the feedback solution, and the ego vehicle performs a blocking maneuver. However, we want to point out that when the open-loop iLQGame algorithm is initialized with the feedback solution, it terminates after the first iteration, yielding the same outcome.

VII. Outlook

The examples in Section VI illustrate iLQGame's capability to consider interactions in racing scenarios. The algorithm converges to different solutions in the open-loop and feedback cases when initialized with identical input sequences. However, when initialized differently, both concepts can yield the same solution. This phenomenon is consistent with the non-uniqueness of Nash equilibria in dynamic nonzero-sum games. Therefore, the convergence property of iLQGame should be further examined in future work. The investigation should include the influence of the initialization and of the step size η.

Our current analysis is limited to one planning step, whereas planning algorithms are usually applied with a moving horizon. The algorithm's initialization is then based on the solution from the previous planning step. Future work should assess the outcomes regarding performance and safety when the two types of equilibria are used with a moving horizon. The analyses should also consider more complex race tracks and the case in which the opponent employs a sequential approach to evaluate the robustness when exposed to a non-interaction-aware player. Ultimately, the analyses should conclude whether iLQGame is suited for racing scenarios and whether an open-loop or a feedback solution should be preferred.

As indicated in Figure 2, the cost parameterization influences the ego vehicle's aggressiveness. Further experiments should determine reasonable racing parameterizations and identify possibly online adjustable parameters to gain an advantage during a race while maintaining safe behaviors. These parameters may depend, e.g., on the current position relative to the opponent, i.e., whether the ego vehicle is leading or trailing.

REFERENCES

[1] M. Schmidt, C. Manna, J. H. Braun, C. Wissing, M. Mohamed, and T. Bertram, "An Interaction-Aware Lane Change Behavior Planner for Automated Vehicles on Highways Based on Polygon Clipping," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1876–1883, 2019.

[2] S. Le Cleac'h, M. Schwager, and Z. Manchester, "ALGAMES: a fast augmented Lagrangian solver for constrained dynamic games," *Autonomous Robots*, vol. 46, no. 1, pp. 201–215, 2022.

[3] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin, "Efficient Iterative Linear-Quadratic Approximations for Nonlinear Multi-Player General-Sum Differential Games," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1475–1481.

[4] L. Crosato, H. P. H. Shum, E. S. L. Ho, and C. Wei, "Interaction-Aware Decision-Making for Automated Vehicles Using Social Value Orientation," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1339–1349, 2023.

[5] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical Game-Theoretic Planning for Autonomous Vehicles," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9590–9596.

[6] J. Bhargav, J. Betz, H. Zheng, and R. Mangharam, "Track based Offline Policy Learning for Overtaking Maneuvers with Autonomous Racecars."

[7] H. Zheng, Z. Zhuang, J. Betz, and R. Mangharam, "Game-theoretic Objective Space Planning."

[8] A. Liniger and J. Lygeros, "A Noncooperative Game Approach to Autonomous Racing," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 884–897, 2020.

[9] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager, "Game-Theoretic Planning for Self-Driving Cars in Multivehicle Competitive Scenarios," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1313–1325, 2021.

[10] Z. Wang, R. Spica, and M. Schwager, "Game Theoretic Motion Planning for Multi-robot Racing," in *Distributed Autonomous Robotic Systems*, ser. Springer Proceedings in Advanced Robotics, N. Correll, M. Schwager, and M. Otte, Eds. Cham: Springer International Publishing, 2019, vol. 9, pp. 225–238.

[11] R. Spica, E. Cristofalo, Z. Wang, E. Montijano, and M. Schwager, "A Real-Time Game Theoretic Planner for Autonomous Two-Player Drone Racing," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1389–1403, 2020.

[12] D. Q. Mayne, "A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[13] W. Li and E. Todorov, "Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems," in *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics*. SciTePress - Science and and Technology Publications, 2004, pp. 222–229.

[14] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed., ser. Classics in applied mathematics. Philadelphia, Pa.: SIAM Soc. for Industrial and Applied Mathematics, 1999, vol. 23.

[15] W. Schwarting, A. Pierson, S. Karaman, and D. Rus, "Stochastic Dynamic Games in Belief Space," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 2157–2172, 2021.

[16] T. Kavuncu, A. Yaraneri, and N. Mehr, "Potential iLQR: A Potential-Minimizing Controller for Planning Multi-Agent Interactive Trajectories," in *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, 2021.

[17] E. L. Zhu and F. Borrelli, "A Sequential Quadratic Programming Approach to the Solution of Open-Loop Generalized Nash Equilibria," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3211–3217.

[18] A. W. Starr and Y. C. Ho, "Further properties of nonzero-sum differential games," *Journal of Optimization Theory and Applications*, vol. 3, no. 4, pp. 207–219, 1969.

[19] M. Rowold, L. Ögretmen, U. Kasolowsky, and B. Lohmann, "Online Time-Optimal Trajectory Planning on Three-Dimensional Race Tracks," in *2023 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2023, pp. 1–8.

[20] J. Chen, W. Zhan, and M. Tomizuka, "Constrained iterative LQR for on-road autonomous driving motion planning," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–7.

APPENDIX

The derivations of the following recursions (16) and (18) without linear cost terms are given in [14]. The supplementary material to [3] (https://github.com/HJReachability/ilqgames/tree/master/ derivations) provides the extensions with linear cost terms.

A. Recursion for the feedback equilibrium

Beginning with $\boldsymbol{P}_K^i = \boldsymbol{Q}_K^i$ and $\boldsymbol{p}_K^i = \boldsymbol{q}_K^i$:
For k from $K-1$ to 0:

$$\boldsymbol{F}_k = \boldsymbol{A}_k - \sum_{j=1}^{N} \boldsymbol{B}_k^j \boldsymbol{K}_k^j, \quad \boldsymbol{\beta}_k = -\sum_{j=1}^{N} \boldsymbol{B}_k^j \boldsymbol{k}_k^j \tag{16a}$$

$$\boldsymbol{P}_k^i = \boldsymbol{Q}_k^i + \boldsymbol{F}_k^\top \boldsymbol{P}_{k+1}^i \boldsymbol{F}_k + \sum_{j=1}^{N} \boldsymbol{K}_k^{j\top} \boldsymbol{R}_k^{ij} \boldsymbol{K}_k^j \tag{16b}$$

$$\begin{aligned} \boldsymbol{p}_k^i = \boldsymbol{q}_k^i + \boldsymbol{F}_k^\top \left(\boldsymbol{p}_{\boldsymbol{k+1}}^{\boldsymbol{i}} + \boldsymbol{P}_{k+1}^i \boldsymbol{\beta}_k\right) + \\ \sum_{j=1}^{N} \left[\boldsymbol{K}_k^{j\top} \boldsymbol{R}_k^{ij} \boldsymbol{k}_k^j - \boldsymbol{K}_k^{j\top} \boldsymbol{r}_k^{ij}\right]. \end{aligned} \tag{16c}$$

It should be noted, that for $N = 1$ and $\boldsymbol{q}_k = \boldsymbol{r}_k = 0$, (16b) together with (8a) simplify to the *difference Riccati equation* (dropping index i):

$$\begin{aligned} \boldsymbol{P}_k = \boldsymbol{Q}_k + \boldsymbol{A}_k^\top \boldsymbol{P}_{k+1} \boldsymbol{A}_k - \left(\boldsymbol{A}_k^\top \boldsymbol{P}_{k+1} \boldsymbol{B}_k\right) \cdot \\ \left(\boldsymbol{R}_k + \boldsymbol{B}_k^\top \boldsymbol{P}_{k+1} \boldsymbol{B}_k\right)^{-1} \left(\boldsymbol{B}_k^\top \boldsymbol{P}_{k+1} \boldsymbol{A}_k\right), \end{aligned} \tag{17}$$

which is well known from LQRs in discrete time.

B. Recursion for the open-loop equilibrium

Beginning with $\boldsymbol{M}_K^i = \boldsymbol{Q}_K^i$ and $\boldsymbol{m}_K^i = \boldsymbol{q}_K^i$:
For k from $K-1$ to 0:

$$\boldsymbol{\Lambda}_k = \mathbb{I} + \sum_{j=1}^{N} \boldsymbol{B}_k^j \boldsymbol{R}_k^{jj^{-1}} \boldsymbol{B}_k^{j\top} \boldsymbol{M}_{k+1}^j \tag{18a}$$

$$\begin{aligned} \boldsymbol{m}_k^i = \boldsymbol{A}_k^\top \Bigg[\boldsymbol{m}_{k+1}^i - \boldsymbol{M}_{k+1}^i \boldsymbol{\Lambda}_k^{-1} \cdot \\ \sum_{j=1}^{N} \boldsymbol{B}_k^j \boldsymbol{R}_k^{jj^{-1}} \left(\boldsymbol{B}_k^{j\top} \boldsymbol{m}_{k+1}^j + \boldsymbol{r}_k^{jj}\right) \Bigg] + \boldsymbol{q}_k^i \end{aligned} \tag{18b}$$

$$\boldsymbol{M}_k^i = \boldsymbol{Q}_k^i + \boldsymbol{A}_k^\top \boldsymbol{M}_{k+1}^i \boldsymbol{\Lambda}_k^{-1} \boldsymbol{A}_k. \tag{18c}$$