A Sequential Quadratic Programming Approach to the Solution of Open-Loop Generalized Nash Equilibria for Autonomous Racing

Edward L. Zhu and Francesco Borrelli

Abstract—Dynamic games can be an effective approach for modeling interactive behavior between multiple competitive agents in autonomous racing and they provide a theoretical framework for simultaneous prediction and control in such scenarios. In this work, we propose DG-SQP, a numerical method for the solution of local generalized Nash equilibria (GNE) for open-loop general-sum dynamic games for agents with nonlinear dynamics and constraints. In particular, we formulate a sequential quadratic programming (SQP) approach which requires only the solution of a single convex quadratic program at each iteration. The three key elements of the method are a non-monotonic line search for solving the associated KKT equations, a merit function to handle zero sum costs, and a decaying regularization scheme for SQP step selection. We show that our method achieves linear convergence in the neighborhood of local GNE and demonstrate the effectiveness of the approach in the context of head-to-head car racing, where we show significant improvement in solver success rate when comparing against the state-of-the-art PATH solver for dynamic games. An implementation of our solver can be found at [https://github.com/zhu-edward/DGSQP](https://github.com/zhu-edward/DGSQP).

Index Terms—Game Theory, Generalized Nash Equilibrium, Sequential Quadratic Programming, Autonomous Racing.

I. INTRODUCTION

Autonomous racing has steadily gained attention as an area of research due to the unique combination of challenges that it presents [1]. Like many widely studied autonomous navigation problems, racing scenarios typically involve multiple agents with no information sharing who are subject to direct competition and safety related constraints. However, what sets racing apart is the aggressive and direct competition which necessitates performance at the limit of speed, tire friction, thrust, etc. in order to achieve victory against one’s opponents. As such, we are faced with the challenge in prediction and planning for autonomous racing where, we need to solve for interactive multi-agent behavior arising from direct competition under shared constraints, while subject to performance limiting factors such as friction limits from nonlinear tire models.

Approaches to tackling these challenges fall broadly into data-driven and model-based approaches, which have both seen impressive successes. In [2], autonomous agents were trained using deep reinforcement learning (DRL) in a video game environment and achieved superior performance in head-to-head races against human players. While this approach was certainly able to learn complex vehicle dynamics and highly expressive multi-agent interaction models, it required days of training and, as with most RL-based methods, is dependent on large quantities of data to reach super-human performance in a simulation environment. In [3], DRL was used to achieve super-human performance in real-world drone racing. However, due to the sparsity of close-proximity interactions in drone races, no interaction models were used in the solution. On the other hand, [4], [5] used model-based approaches in a predict-then-plan architecture to achieve super-human performance in full scale race cars at speeds of up to 270 km/h. In these works, an upstream prediction module provides behavior forecasts of the opponents, which are then treated as constant during the downstream planning phase when an action plan is formulated for the ego vehicle (EV) via optimal control techniques which leverage explicit nonlinear vehicle models. These approaches have the benefit of being highly transparent, which is crucial for deployment on expensive and safety critical hardware, but can be limited in expressiveness in terms of interactivity between the opponent predictions and ego plan. Our work seeks to address this limitation for model-based approaches by producing highly interactive solutions to the prediction and planning problem for autonomous racing while maintaining transparency through the use of explicit vehicle models.

Recent work has posed the prediction and planning problem as an equilibrium finding problem for a dynamic game [6], [7], [8]. These approaches search for equilibria over the joint trajectory space of the EV and its opponents, which allows for simultaneous prediction and planning, as they are not conditioned on a static prior EV plan and can in fact solve over the space of multi-agent interactions. Such methods are based on the theory of non-cooperative dynamic games [9] and have been applied to a wide variety of trajectory optimization tasks for multi-agent robotic systems [10], [11], [12], [13], [14], [15]. Two solution or equilibrium concepts are common for dynamic games, namely the Stackelberg and Nash equilibria, which make different assumptions about the information structure of the game. A Stackelberg equilibrium can be found for a game with an explicit leader-follower hierarchy [16], whereas a Nash equilibrium models the case when agents make their decisions simultaneously. Our work focuses on the selection of Nash equilibria, which we believe to be a good fit for modeling the behavior of ego-centric agents in racing scenarios where no a priori structure is imposed on the order of the interactions and where agents can have zero-sum terms in their objective functions, which describe direct
competition with their opponents.

Motivated by the considerations above, this paper focuses on finding generalized Nash equilibria (GNE) \cite{17} for open-loop dynamic games with state and input constraints with a numerically robust solver. We build on our previous conference paper \cite{6} and present three main contributions. The first is the DG-SQP solver, which is a novel iterative approach for finding GNE of a discrete-time open-loop dynamic game based on sequential quadratic programming (SQP). In particular, the method is able to account for nonlinear game dynamics and constraints on both the game state and agent actions. The three key elements of the method are a non-monotonic line search for solving the associated KKT equations, a merit function to handle zero sum costs, and a decaying regularization scheme for the SQP step selection. The second contribution is a novel application of model predictive contouring control in the context of dynamic games which is used to approximate Frenet-frame kinematics for improved numerical robustness. Finally, we perform an extensive simulation study comparing the performance of our solver to the state-of-the-art PATH solver based on mixed complementary problems \cite{18}.

We show comparable performance to the PATH solver, in terms of success rate, on dynamic games formulated using exact Frenet-frame kinematics and significant improvements when the approximate formulation is used.

We remark that the broader objective of our research extends to leveraging these open-loop dynamic game solutions within feedback-based systems. While the focus of this paper is the robustness of the proposed open-loop GNE solver, solver robustness in itself is not the ultimate goal. Rather, it is critical property which allows for effective integration of the solver and/or its solutions into closed-loop systems. This transition from open to closed-loop application of GNE solutions from our proposed solver has previously appeared in \cite{6}.

II. RELATED WORK

A. Numerical Solvers for Nash Equilibria of Dynamic Games

For the solution of GNE for dynamic games, many numerical approaches have been proposed. \cite{12} takes a differential dynamic programming approach to obtain a linear quadratic approximation of the dynamic game and its associated feedback Nash equilibrium. This is built upon in \cite{8} to address stochastic games over belief spaces, which takes into account the effect of noisy measurements on state estimation in the game dynamics. However, these approaches are unable to explicitly account for inequality constraints and instead must include them in the cost function via barrier functions. This can obfuscate the meaning of the game as the cost measures both performance and constraint violation. \cite{15}, \cite{19} propose an approach for the subclass of potential dynamic games which was shown to be computationally efficient. However, by requiring that the dynamic game be described by a single potential function over all agents, we note that direct competition, in the form of zero-sum terms in the agents’ objective functions, cannot be captured in this approach. \cite{7}, \cite{20}, \cite{21} formulate a method akin to block coordinate descent called Sensitivity Enhanced Iterative Best Response (SE-IBR) where the optimal control problems for each agent, which are coupled in the dynamic game setting, are decoupled but augmented with a term in the objective function which captures the sensitivity of the TV’s solution w.r.t. the EV’s solution. Agents then improve their solutions in a sequential manner while holding the behavior of all other agents fixed. It was shown that fixed points of this algorithm correspond to GNE. However, each iteration of the method requires the solution of the same number of optimization problems as there are agents and can be slow to converge in practice. In contrast, our proposed solver only requires the solution of a single convex optimization problem at each iteration. \cite{11} proposes a solver based on the augmented Lagrangian method. The solver, called ALGAMES, shows good performance when compared to \cite{12}. However, we show in prior work that ALGAMES appears to struggle with convergence in the context of car racing where more complex dynamics and environments are introduced \cite{6}. Our proposed solver is perhaps most similar to \cite{23} which also leverages SQP for the computation of feedback Nash equilibria, but does not investigate its local behavior. Compared to \cite{23}, we also introduce an approximation scheme which improves solver convergence in racing scenarios. Finally, we note that the problem of finding GNE for open-loop dynamic games can be formulated as a mixed complementarity problem for which the PATH solver is the state-of-the-art \cite{18}. This solver was used to great effect in multiple recent works to compute GNE for multi-agent navigation tasks \cite{13}, \cite{14}. As such, the PATH solver will be the main target of our comparisons, where we show, through a numerical study, that our proposed solver achieves greater success rate in the context of head-to-head racing scenarios.

B. Game-Theoretic Methods in Autonomous Racing

Due to its competitive nature, autonomous racing has been a popular test bed for game-theoretic methods of prediction and planning and many of the previously mentioned methods have been applied in this context. \cite{8} formulates the racing problem as a stochastic dynamic game which can take into account state and measurement uncertainty when solving for a Nash equilibrium. However, this approach is unable to explicitly incorporate safety-critical track boundary and collision avoidance constraints, which are crucial in racing. \cite{24} poses the problem as a constrained potential game. This results in good computational efficiency for game-theoretic prediction and planning in a real-time, model predictive game play (MPGP) manner, but importantly precludes zero-sum components in the agents’ cost functions by definition of the potential game. This makes it difficult to capture direct competition between the agents and instead the approach relies on heuristics and mode switching in order to induce competitive behavior. \cite{7}, \cite{22}, \cite{20}, \cite{21} leverage the aforementioned SE-IBR algorithm to approximately solve a dynamic game in a MPGP manner and use the approximate GNEs as high-level plans for a tracking policy in hardware races. However, this work, like all of the ones discussed so far in this section, have only shown GNE solution results for linear models such as the integrator model or simple nonlinear models such as the kinematic bicycle
model. The only exception to this is in [25] where a bimatrix game was defined over the incurred costs of sampled trajectory rollouts for two vehicles each using the dynamic bicycle model. The trajectory pair corresponding to the NE of this bimatrix game was then chosen as the opponent prediction and EV plan. This approach can be considered as a zeroth order method for the solution of GNEs as it only requires evaluations of the components of the dynamic game. (This is in contrast to [11, 13] and our approach, which leverage gradients of the game components). As such, it provides a simple framework for leveraging high fidelity vehicle models. However, it is straightforward to see that the effectiveness of this approach largely depends on the number of trajectory samples taken and it is unclear if the trajectory pair corresponding to the NE of the bimatrix game is a GNE of the dynamic game.

### III. Problem Formulation

Consider an $M$-agent, finite-horizon, discrete-time, general-sum, open-loop, dynamic game whose state is characterized by the joint dynamical system:

$$x_{k+1} = f(x_k, u_k),$$

where $x_k^i \in \mathcal{X}^i$ and $u_k^i \in \mathcal{U}^i$ are the state and input of agent $i$ at time step $k$ and

$$x_k = [x_k^i, \ldots, x_k^M] \in \mathcal{X}^i \times \cdots \times \mathcal{X}^M = \mathcal{X} \subseteq \mathbb{R}^n$$

$$u_k = [u_k^i, \ldots, u_k^M] \in \mathcal{U}^i \times \cdots \times \mathcal{U}^M = \mathcal{U} \subseteq \mathbb{R}^m,$$

are the concatenated states and inputs of all agents. In this work, we will use the notation $x_k^i$ and $u_k^i$ to denote the collection of states and inputs for all but the $i$-th agent.

Each agent $i$ attempts to minimize its own cost function, which is comprised of stage costs $l_k^i$ and terminal cost $l_N^i$, over a horizon of length $N$:

$$J^i(x, u^i) = \sum_{k=0}^{N-1} l_k^i(x_k, u_k^i) + l_N^i(x_N)$$  \hspace{1cm} (2a)$$

$$= J^i(u^1, \ldots, u^M, x_0),$$  \hspace{1cm} (2b)$$

where $x = \{x_0, \ldots, x_N\}$ and $u_i = \{u_{0}^i, \ldots, u_{N-1}^i\}$ denote state and input sequences over the horizon. Note that the cost in (2a) for agent $i$ depends on its own inputs and the joint state, which can capture dependence on the behavior of the other agents. We arrive at (2b) by recursively substituting in the dynamics (1) to the cost function, which are naturally a function of the open-loop input sequences for all agents. The agents are additionally subject to $n_c$ constraints

$$C(u^1, \ldots, u^M, x_0) \leq 0,$$  \hspace{1cm} (3)$$

which can be used to describe individual constraints as well as coupling between agents and where we have once again made the dependence on the joint dynamics implicit. For the sake of brevity, when focusing on agent $i$, we omit the initial state $x_0$ and write the cost and constraint functions as $J^i(u^i, u^{i-})$ and $C(u^i, u^{i-})$. Let us now define the conditional constraint set

$$\mathcal{U}^i(u^{i-}) = \{u^i | C(u^i, u^{i-}) \leq 0\},$$

which can be interpreted as a restriction of the joint constraint set for agent $i$ given some $u^{i-}$. We make the following assumption about the functions and constraint sets.

**Assumption 1.** The sets $\mathcal{X}^i$ and $\mathcal{U}^i$ are compact and the functions $f$, $J^i$, and $C$ are three times continuously differentiable on $\mathcal{X}^i$ and $\mathcal{U}^i$ for all $i \in \{1, \ldots, M\}$.

### A. Generalized Nash Equilibrium

We define the constrained dynamic game as the tuple:

$$\Gamma = (N, \mathcal{X}, \mathcal{U}, f, (J^i)^{M}_{i=1}, C).$$  \hspace{1cm} (4)$$

For such a game, a GNE [17] is attained at the set of feasible input sequences $u = \{u^i\}_{i=1}^M$ which minimize (2) for all agents $i$. Formally, we define this solution concept as follows:

**Definition 1.** A generalized Nash equilibrium (GNE) for the dynamic game $\Gamma$ is the set of open-loop solutions $u^* = \{u^{i*}\}_{i=1}^M$ such that for each agent $i$:

$$J^i(u^{i*}, u^{-i*}) \leq J^i(u^i, u^{i*}), \forall u^i \in \mathcal{U}^i(u^{-i*}).$$

If the condition holds only in some local neighborhood of $u^{i*}$, then $u^{i*}$ is denoted as a local GNE.

In other words, at a local GNE, agents cannot improve their cost by unilaterally perturbing their open-loop solution in a locally feasible direction. Furthermore, it was shown in [23, Theorem 2.2] that the local GNE for agent $i$ can be obtained equivalently by solving the following constrained finite horizon optimal control problems (FHOCP):

$$u^{i*, u^{-i,*}} = \arg \min_{u^i} J^i(u^i, u^{-i,*})$$

subject to $C(u^i, u^{-i,*}) \leq 0,$

where $u^{-i,*}$ correspond to local GNE solutions for the other agents. Note that we are assuming uniqueness of the local GNE of $\Gamma$. This assumption will be made formal in the next section. A distinct advantage of using (5) to model agent interactions is that a dynamic game allows for a direct representation of agents with competing objectives as the $M$ objectives are considered separately instead of being summed together, which is typical in cooperative multi-agent approaches [24].

### IV. An SQP Approach to Dynamic Games

In this section, we propose a method which iteratively solves for open-loop local GNE of dynamic games using sequential quadratic approximations. In particular, we will derive the algorithm and present guarantees on local convergence, which is based on established SQP theory [27]. We begin by defining the Lagrangian functions for the $M$ coupled FHOCPs in (5):

$$\mathcal{L}^i(u^i, u^{-i,*}, \lambda) = J^i(u^i, u^{-i,*}) + C(u^i, u^{-i,*})^\top \lambda,$$

where we have again omitted the dependence on the initial state $x_0$ for brevity. As in [11], we require that the Lagrange multipliers $\lambda \geq 0$ are equal over all agents, i.e $\lambda_i = \lambda_j = \lambda$, $\forall i, j \in \{1, \ldots, M\}$. Since the multipliers reflect the sensitivity of the optimal cost w.r.t. constraint violation, this can be
interpreted as a requirement for parity in terms of constraint violation for all agents. Under this condition, the GNE from (5) are also known as normalized Nash equilibria [23].

A direct consequence of writing the constrained dynamic game in the coupled nonlinear optimization form of (5) is that, subject to regularity conditions, solutions of (5) must satisfy the KKT conditions below:

\( \nabla u_i L^i(u^{i*}, u^{-i*, \lambda^*}) = 0, \forall i = 1, \ldots, M, \) (6a)

\( C(u^{1*}, \ldots, u^{M*}) \leq 0, \) (6b)

\( C(u^{1*}, \ldots, u^{M*})^\top \lambda^* = 0, \) (6c)

\( \lambda^* \geq 0. \) (6d)

We therefore propose to find a local GNE as a solution to the KKT system in an iterative fashion starting from an initial guess for the primal and dual solution, which we denote the KKT system (6) in an iterative fashion starting from an initial guess for the primal and dual solution, which we denote as \( u_0 \) and \( \lambda_0 \geq 0 \) respectively, and taking steps \( p_q^i \) and \( \lambda_q \), at iteration \( q \), to obtain the sequence of iterates:

\[ u_{q+1}^i = u_q^i + p_q^i, \lambda_{q+1} = \lambda_q + p_q^\lambda, \] (7)

In particular, we form a quadratic approximation of (6a) and linearize the constraints in (6b) about the primal and dual solution at iteration \( q \) in a SQP manner [29] as follows:

\[ L_q = \begin{bmatrix}
\nabla^2 & \nabla u^2 & \cdots & \nabla u^M \\
\nabla u^1 & \nabla u^2 & \cdots & \nabla u^M \\
\vdots & \vdots & \ddots & \vdots \\
\nabla u^1 & \nabla u^M & \cdots & \nabla u^M
\end{bmatrix},
\]

\[ h_q = \begin{bmatrix}
\nabla u^1 J_q^1 \\
\cdots \\
\nabla u^1 J_q^N
\end{bmatrix},
\]

\[ G_q = \begin{bmatrix}
\nabla u^1 C_q \\
\cdots \\
\nabla u^1 C_q
\end{bmatrix},
\]

\[ B_q = \text{proj}_{\geq 0}(L_q + L_q^\top)/2 + \epsilon I,
\] (8)

where the subscript \( q \) indicates that the corresponding quantity is evaluated at the primal and dual iterate \( u_q \) and \( \lambda_q \). Here, \( \epsilon \geq 0 \) is a regularization coefficient, \( I \) is the identity matrix of appropriate size, and

\[ \text{proj}_{\geq 0}(X) = \arg \min_{Y \succeq 0} \| X - Y \|_F. \] (9)

denotes the operation which projects the symmetric matrix \( X \in \mathbb{S}^n \) onto the positive semi-definite cone, where \( \| \cdot \|_F \) denotes the Frobenius norm. We note that the semidefinite program in (9) admits the closed-form solution

\[ \text{proj}_{\geq 0}(X) = \sum_{i=1}^n \max\{0, s_i\} v_i v_i^\top, \]

where the \( s_i \) and \( v_i \) denote the \( i \)-th eigenvalue and eigenvector of \( X \) respectively. Using the approximation in (8), we solve for the step in the primal variables via the following convex quadratic program (QP):

\[ p_q^u = \arg \min_{p^u} \frac{1}{2} p^u^\top B_q p^u + h_q^\top p^u \] (10a)

subject to \( C_q + G_q p_q^u \leq 0. \) (10b)

where we denote \( p^u = [p^1, \ldots, p^M]^\top \). Denote the Lagrange multipliers corresponding to the solution of (10) as \( d_q \). We then define the step in the dual variables as

\[ p_q^\lambda = d_q - \lambda_q, \] (11)

which maintains nonnegativity of the dual iterates given \( \lambda_0 \geq 0 \). Finally, we note that in contrast to the approach used in [7], which requires the solution of \( M \) optimization problems, our SQP procedure requires the solution of only a single QP at each iteration.

### A. Local Behavior of Dynamic Game SQP

We make the following assumptions about the primal and dual solutions of (5):

**Assumption 2.** Solutions \( \{u^{i*}\}_{i=1}^M \) and \( \lambda^* \) of (5) satisfy the following, for each \( i \in \{1, \ldots, M\} \):

- \( \lambda^* \perp C(u^{i*}, u^{-i*}) \) and \( \lambda^*_j = 0 \iff C_j(u^{i*}, u^{-i*}) < 0 \) for \( j = 1, \ldots, n_c \).

- The rows of the Jacobian of the active constraints at the local GNE, i.e. \( \nabla u \tilde{C}(u^{i*}, u^{-i*}) \), are linearly independent.

- \( d^\top \nabla^2 u \tilde{L}^i(u^{i*}, u^{-i*}, \lambda^*) d > 0, \forall d \neq 0 \) such that \( \nabla u \tilde{C}(u^{i*}, u^{-i*}, \lambda^*)^\top d = 0 \).

The first assumption is strict complementary slackness, the second is the linear independence constraint qualification (LICQ), and the third states that the Hessian of the Lagrangian function is positive definite on the null space of the active constraint Jacobians at the solution. It is straightforward to see that By standard optimization theory, Assumption 2 and Assumption 2 together constitute necessary and sufficient conditions for a primal and dual solution of (5) for agent \( i \) to be locally optimal and unique given \( u^{-i*} \). When these conditions hold for the solutions over all agents, it was shown in [23] that satisfaction of the requirements for a unique local GNE follow immediately. Note that, as in [27] and [23], Assumption 2 is standard and can be verified a posteriori.

To analyze the local behavior of the iterative procedure as defined by (7), (8), and (10), let us assume that \( u_0 \) and \( \lambda_0 \) are close to the optimal solution and the subset of active constraints at the local GNE, which we denote as \( C \), with Jacobian \( \tilde{G} \), is known and constant at each iteration \( q \). Therefore, for the purposes of this section, we can replace the inequality constraint in (10) with the equality constraint:

\[ \tilde{C}_q + \tilde{G}_q p^u = 0. \] (12)

We refer to the QP constructed from (10) and (12) as EQP.

In the traditional derivation of the SQP procedure [27], [29], it was shown that under the aforementioned assumptions, the SQP step computed is identical to a Newton step for the corresponding KKT system. The SQP step therefore inherits the quadratic convergence rate of Newton’s method in a local neighborhood of the optimal solution [27] Theorem 3.1]. However, in the case of dynamic games, the equivalence between the SQP procedure and Newton’s method is no longer exact since the matrix \( L_q \) is not symmetric in general. To see this, let us first state the joint KKT system for the equality constrained version of (5):

\[ F(u^{1*}, \ldots, u^{M*}, \lambda^*) = \begin{bmatrix}
\nabla u^1 \tilde{L}^1(u^{1*}, \ldots, u^{M*}, \lambda^*) \\
\vdots \\
\nabla u^M \tilde{M}^M(u^{1*}, \ldots, u^{M*}, \lambda^*)
\end{bmatrix} = 0, \] (13)
For the system of equations [13], the Newton step at iteration \( q \) is the solution of the linear system:
\[
\begin{bmatrix}
L_q & G_q^T \\
G_q & 0
\end{bmatrix}
\begin{bmatrix}
p_q^u \\
p_q^\lambda
\end{bmatrix} = - \begin{bmatrix}
h_q + G_q^T \lambda_q \\
C_q
\end{bmatrix}.
\number
(14)
\]
On the other hand, by the first order optimality conditions for EQP, we have that the SQP step must satisfy
\[
\begin{bmatrix}
B_q & G_q^T \\
G_q & 0
\end{bmatrix}
\begin{bmatrix}
p_q^u \\
p_q^\lambda
\end{bmatrix} = - \begin{bmatrix}
h_q + G_q^T \lambda_q \\
C_q
\end{bmatrix}.
\number
(15)
\]
When the matrix \( L_q \) is positive definite and \( \epsilon = 0 \), [15] and [14] are equivalent. This corresponds to the special case of potential games [30]. However, this is not true in general for our SQP step, which implies that we cannot inherit the quadratic convergence of Newton’s method. Instead, the SQP step from [10a] and [12] can be seen as a symmetric approximation to the Newton step. As such, we establish guaranteed local linear convergence for our SQP procedure via established theory for SQP with approximate Hessians. Before proving the main result of this section, let us first define the bounded deterioration property, which essentially requires that the distance between a matrix and its approximations are bounded.

**Definition 2.** A sequence of matrix approximations \( \{B_q\} \) to \( L^* \) for the SQP method is said to have the property of bounded deterioration if there exist constants \( \alpha_1 \) and \( \alpha_2 \) independent of \( q \) such that:
\[
\|B_{q+1} - L^*\| \leq (1 + \alpha_1 \sigma_q)\|B_q - L^*\| + \alpha_2 \sigma_q,
\]
where \( \sigma_q = \max\{\|u_{q+1} - u^*\|, \|u_q - u^*\|, \|\lambda_{q+1} - \lambda^*\|, \|\lambda_q - \lambda^*\|\} \), and \( L^* \) denotes the matrix \( L \) in [3] evaluated at the solution \( u^*, \lambda^* \).

**Theorem 1.** Consider the dynamic game defined by [4]. Let Assumptions [7] and [2] hold. Then there exist positive constants \( \epsilon_1 \) and \( \epsilon_2 \) such that if
\[
\|u_0 - u^*\| \leq \epsilon_1, \quad \|B_0 - L^*\| \leq \epsilon_2,
\]
and \( \lambda_0 = -(G_qG_q^T)^{-1}G_qh_0 \), then the sequence \( \{u_q, \lambda_q\} \) generated by the SQP procedure [7] and [10] converges linearly to \( (u^*, \lambda^*) \).

**Proof.** We obtain the result by showing that the conditions of [27] Theorem 3.3] are satisfied. The first condition requires that the approximations \( B_q \) are positive definite on the null space of \( G_q \). Since \( B_q \) is constructed by projecting \( L_q \) into the positive definite cone, this condition is satisfied trivially.

The second condition requires that the sequence \( \{B_q\} \) satisfies the bounded deterioration property from Definition 2. To show this, we begin with the following derivation:
\[
\|B_{q+1} - L^*\| = \|B_{q+1} - B_q + B_q - L^*\| \\
\leq \|B_q - L^*\| + \|B_{q+1} - B_q\|.
\]
From the above, it can be seen that \( \{B_q\} \) satisfies the bounded deterioration property with \( \alpha_1 = 0 \) if there exists \( \alpha_2 \) such that \( \|B_{q+1} - B_q\| \leq \alpha_2 \sigma_q \). We will show this via the mean value theorem, where we first observe that by Assumption 1, \( L(u_q, \lambda_q) \) is continuous and differentiable in \( u_q \) and \( \lambda_q \).

The smoothness of \( B(u_q, \lambda_q) \) is therefore dependent on the smoothness of the projection operator in [9]. It was established in [31] that for parametric strongly convex conic optimization problems, there exists a unique, continuous, and differentiable mapping from the parameters to the optimal solution of the problem. The projection operator [9] is a semi-definite program which admits a unique solution [32]. Therefore, we may conclude that \( B(u_q, \lambda_q) \) is continuous and differentiable in \( u_q \) and \( \lambda_q \). This allows us to establish the following inequalities:
\[
\|B_{q+1} - B_q\| = \|B_{q+1} - B^* + B^* - B_q\| \\
\leq \|B_{q+1} - B^*\| + \|B_q - B^*\| \\
= \|B_{q+1} - B(u^*, \lambda_{q+1}) + B(u^*, \lambda_{q+1}) - B^*\| \\
+ \|B_q - B(u^*, \lambda_q) + B(u^*, \lambda_q) - B^*\| \\
\leq \|B_{q+1} - B(u^*, \lambda_{q+1})\| \\
+ \|B(u^*, \lambda_{q+1}) - B^*\| \\
+ \|B_q - B(u^*, \lambda_q)\| \\
+ \|B(u^*, \lambda_q) - B^*\|.
\]
Since \( B \) is continuous and differentiable, by the mean value theorem:
\[
\|B_{q+1} - B(u^*, \lambda_{q+1})\| + \|B(u^*, \lambda_{q+1}) - B^*\| \\
+ \|B_q - B(u^*, \lambda_q)\| + \|B(u^*, \lambda_q) - B^*\| \\
\leq \beta_1 \|u_{q+1} - u^*\| + \beta_2 \|\lambda_{q+1} - \lambda^*\| \\
+ \beta_3 \|u_q - u^*\| + \beta_4 \|\lambda_q - \lambda^*\| \\
\leq \max(\beta_1, \beta_2, \beta_3, \beta_4) \sigma_q,
\]
where
\[
\begin{align*}
\beta_1 &= \sup_{t \in [0, 1]} \|\nabla_u B(tu_{q+1} + (1 - t)u^*, \lambda_{q+1})\|, \\
\beta_2 &= \sup_{t \in [0, 1]} \|\nabla_B B(u^*, t\lambda_{q+1} + (1 - t)\lambda^*)\|, \\
\beta_3 &= \sup_{t \in [0, 1]} \|\nabla_u B(tu_q + (1 - t)u^*, \lambda_q)\|, \\
\beta_4 &= \sup_{t \in [0, 1]} \|\nabla_B B(u^*, t\lambda_q + (1 - t)\lambda^*)\|.
\end{align*}
\]
This satisfies the bounded deterioration property with \( \alpha_2 = \max(\beta_1, \beta_2, \beta_3, \beta_4) \). Given Assumption 2 holds, linear convergence of the SQP iterations follows directly as a consequence of [27] Theorem 3.3].

**V. NUMERICAL ROBUSTNESS OF DYNAMIC GAME SQP**
We have shown that our proposed SQP approach exhibits linear convergence when close to a local GNE. However, as is commonly seen with numerical methods for nonlinear optimization, a naive implementation of the procedure defined by [7], [8], and [10] often performs poorly due to overly aggressive steps leading to diverging iterates. In this section, we introduce multiple practical considerations which will help address this problem in practice. Namely, a merit function, a non-monotonic line search method, and a decaying regularization scheme. These components will be used to determine how much of the SQP step \( p_q^u \) and \( p_q^\lambda \) can be taken to make progress towards a local GNE while remaining in a region about the current iterate where the SQP approximation [10] is
valid. We will show through examples that these additions are crucial in improving the numerical robustness of the proposed SQP approach.

A. A Novel Merit Function

Merit functions are commonly used in conjunction with a backtracking line search technique as a mechanism for measuring progress towards the optimal solution and limiting the step size taken by an iterative solver [29]. This is important as the quality of the QP approximation [10] may be poor when far away from the iterate, which could result in divergence of the iterates if large steps are taken.

In traditional constrained optimization, merit functions typically track a combination of cost value and constraint violation. In the context of dynamic games, this is not as straightforward as the agents may have conflicting objectives and a proposed step may result in an increase in the objectives of some agents along with a decrease in others’. We also cannot just simply sum the objectives as minimizers of the combined cost function may not be local GNE. We therefore propose the following merit function:

\[
\phi(u, \lambda, s; \mu) = \frac{1}{2} \left\| \nabla_u L^1(u, \lambda) \right\|^2 + \mu \|C(u) - s\|_1,
\]

and define \(\gamma(u, \lambda) = (1/2)\|\nabla L(u, \lambda)\|^2\). The slack variable \(s = \min(0, C(u))\) is defined element-wise such that \(C - s\) captures violation of the inequality constraints and we define the step \(p^s = C(u) + G(u)p^u - s\). Compared to the merit function from [23], which only included the first term of (16), the novelty of our’s is the \(l^1\) norm term, whose purpose will be described shortly. Instead of measuring the agent objectives, our merit function tracks the first order optimality conditions in addition to constraint violation. It should be easy to see that the merit function attains a minimum of zero at any local GNE. However, we note that this merit function is not exact [29] since the first order conditions are only necessary for optimality.

Since we would like the sequence of iterates to converge to the minimizers of \(\phi\), it follows that at each iteration we would like take a step in a descent direction of \(\phi\). As such, we will now, in a manner similar to [29], analyze the directional derivative of the proposed merit function in the direction of the steps computed from [10] and describe a procedure for choosing the parameter \(\mu\) and the corresponding conditions such that a descent in \(\phi\) is guaranteed.

Since the \(l^1\) norm is not differentiable everywhere, we begin by taking a Taylor series expansion of \(\gamma\) and \(C - s\) for \(\alpha \in (0, 1)\):

\[
\phi(u + \alpha p^u, \lambda + \alpha p^\lambda, s + \alpha p^s; \mu) - \phi(u, \lambda, s; \mu) \\
\leq \alpha \nabla_{u, \lambda, s} \gamma \left[ \begin{array}{c} p^u \\ p^\lambda \\ p^s \end{array} \right] + \mu \|C - s + \alpha(Gp^u - p^s)\|_1 \\
- \mu \|C - s\|_1 + \beta \alpha^2 \|p\|_2^2 \\
= \alpha \left( \nabla_{u, \lambda, s} \gamma \left[ \begin{array}{c} p^u \\ p^\lambda \\ p^s \end{array} \right] - \mu \|C - s\|_1 \right) + \beta \alpha^2 \|p\|_2^2,
\]

where the term \(\beta \alpha^2 \|p\|_2^2\) bounds the second derivative terms for some \(\beta > 0\). Following a similar logic, we can obtain the bound in the other direction:

\[
\phi(u + \alpha p^u, \lambda + \alpha p^\lambda, s + \alpha p^s; \mu) - \phi(u, \lambda, s; \mu) \\
\geq \alpha \left( \nabla_{u, \lambda, s} \gamma \left[ \begin{array}{c} p^u \\ p^\lambda \\ p^s \end{array} \right] - \mu \|C - s\|_1 \right) - \beta \alpha^2 \|p\|_2^2.
\]

Dividing the inequality chain by \(\alpha\) and taking the limit \(\alpha \to 0\), we obtain the directional derivative:

\[
D(\phi(u, \lambda, s; \mu), p^u, p^\lambda) = \nabla_{u, \lambda, s} \gamma \left[ \begin{array}{c} p^u \\ p^\lambda \end{array} \right] - \mu \|C - s\|_1
\]

From (17), it should be clear that given \(C - s \neq 0\), there exists a value for \(\mu > 0\) such that the directional derivative is negative. As such, we propose the following expression to compute \(\mu\), given some \(\rho \in (0, 1)\):

\[
\mu \geq \left( \nabla_{u, \lambda, s} \gamma \left[ \begin{array}{c} p^u \\ p^\lambda \end{array} \right] \right) / ((1 - \rho)\|C - s\|_1),
\]

which results in \(D(\phi(u, \lambda, \mu), p^u, p^\lambda) \leq -\rho \mu \|C - s\|_1\).

In the case when \(C - s = 0\), we unfortunately cannot guarantee that the directional derivative will always be negative. To see this, let us analyze the first term of (17):

\[
\nabla_{u, \lambda} \gamma \left[ \begin{array}{c} p^u \\ p^\lambda \end{array} \right] = \nabla L^T \left[ \begin{array}{cc} L & G^T \end{array} \right] \left[ \begin{array}{c} p^u \\ p^\lambda \end{array} \right]
\]

\[
= \nabla L^T [B + R G^T] \left[ \begin{array}{c} p^u \\ p^\lambda \end{array} \right]
\]

\[
= -\nabla L^T \nabla L + \nabla L^T R p^u,
\]

where \(R\) denotes the residual matrix i.e. \(R = L - B\) and we arrive at the third equality by plugging in for the stationarity condition from [10]. From (19), it should be immediately apparent that when the residual matrix is large and dominates the first term, it is possible for the directional derivative to be positive. This can be interpreted as a condition on how well the positive definite \(B\) actually approximates the original stacked Hessian matrix \(L\). For dynamic games where the agents have highly coupled and differing objectives, i.e. when \(L\) is highly non-symmetric, it is reasonable to expect that the approximation would suffer and that we may not be able to achieve a decrease in the merit function. For this reason, we utilize a non-monotone strategy for the line search step, which will be discussed in the following.

B. A Non-Monotone Line Search Strategy

Line search methods are used in conjunction with merit functions to achieve a compromise between the goals of making rapid progress towards the optimal solution and keeping
the iterates from diverging. This is done by finding the largest step size \( \alpha \in (0, 1] \) such that the following standard decrease condition is satisfied [27]:

\[
\phi(u_q + \alpha p^u_q, \lambda_q + \alpha p^\lambda_q, s_q + \alpha p^s_q; \mu) \\
\leq \phi(u_q, \lambda_q, s_q; \mu) + \zeta \alpha D(\phi(u_q, \lambda_q, s_q; \mu), p^u_q, p^\lambda_q),
\]

where \( \zeta \in (0, 0.5) \). However, since our merit function is not exact, the line search procedure can be susceptible to poor local minima which do not correspond to local GNE. Moreover, there is evidence that requiring monotonic decrease in the merit function at each iteration may actually impede the solution process [33]. We therefore include in our approach a non-monotone approach to line search called the watchdog strategy [34]. Instead of insisting on sufficient decrease in the merit function at every iteration, this approach allows for relaxed steps to be taken for a certain number of iterations where increases in the merit function are allowed. The decrease requirement (20) is then enforced after the prescribed number of relaxed iterations or if the step size exceeds some given threshold. The rationale behind this strategy is that we can use the relaxed steps as a way to escape regions where it is difficult to make progress w.r.t. the merit function. The algorithm we implement is based on the non-monotone stabilization scheme presented in [18] and involves the taking of \( d \)-steps and \( m \)-steps, where \( d \)-steps correspond to relaxed steps (\( \alpha = 1 \)) which disregard the merit decrease condition and \( m \)-steps which enforce merit decrease through an Armijo backtracking linesearch on \( \alpha \) in (20). Furthermore, the method stores the list of iterates which satisfy the merit decrease condition as checkpoints. This allows for a watchdog step to reset the iterate to a checkpoint in the event that an \( m \)-step was unsuccessful and to perform a backtracking linesearch from there.

C. A Decaying Regularization Scheme

The regularization parameter \( \epsilon \) introduced in [3] is a common mechanism used in iterative Newton-based procedures to control the size of \( p^u \). However, the selection of the regularization parameter is non-trivial in many cases as the convergence behavior of the iterative procedure can be highly sensitive to the value of \( \epsilon \). Often, it is observed that iterates can diverge if \( \epsilon \) is too small, whereas progress is slow if \( \epsilon \) is too big. To address this challenge, one method which has seen success is that of decaying regularization which given some initial setting of \( \epsilon \) gradually decreases its value over iterations of the algorithm [11], [35], [36]. In this work, we take a similar approach to updating the values of \( \epsilon \), but importantly, integrate it into the non-monotone line search procedure. Specifically, for some initial regularization value \( \epsilon_0 \geq 0 \) and decay multiplier \( \eta \in (0, 1] \), we impose the following update rule:

\[
\epsilon_{q+1} = \begin{cases} 
\epsilon_q & \text{if } d \text{-step}, \\
\eta \epsilon_q & \text{if } m \text{-step}, 
\end{cases}
\]

where the regularization value only decays after a step which satisfies the merit function decrease condition is found. The reason for this is to avoid reducing the regularization strength when potentially spurious relaxed \( d \)-steps are taken and the iterate is reset to a checkpoint.

VI. THE DYNAMIC GAME SQP ALGORITHM

By combining the elements discussed in the previous section, we arrive at the dynamic game SQP (DG-SQP) algorithm, which is presented in Algorithm 1. The algorithm requires as input initial guesses of open-loop input sequences for each agent and the initial regularization value \( \epsilon_0 \). Line 2 initializes the primal and dual iterates, where the dual variables are initialized as the least squares solution to (6a). Lines 3 to 14 perform the SQP iteration which has been described in Sections [IV] and [V]. An iterate is said to have converged to a local GNE if it satisfies the KKT conditions described in (6) up to some user specified tolerance. Namely, for some given \( \rho_1, \rho_2, \rho_3 > 0 \), we require the conditions \( \|\nabla \mathcal{L}(u_q, \lambda_q)\|_\infty \leq \rho_1, \|C(u_q)\|_\infty \leq \rho_2, |\lambda_q^T C(u_q)| \leq \rho_3 \) be satisfied in order for the algorithm to terminate successfully. The algorithm outputs the open-loop strategies for the \( M \) agents and the corresponding Lagrange multipliers.

VII. FRENET-FRAME GAMES FOR AUTONOMOUS RACING

This section presents the second main contribution of this paper. We formulate the autonomous racing problem as a
dynamic game using ideas from contouring control \[37\], \[38\] and leverage the solver presented in the previous section to compute the solution to such game. Numerical studies in Section \[VIII\] will show that with this formulation, we achieve significantly higher success rates when solving for GNE of racing dynamic games.

We consider the class of racing dynamic games where the race track is represented as a parametric path \( \tau : [0, L] \rightarrow \mathbb{R}^2 \), which is assumed to be twice differentiable and of total length \( L \). In particular, when given an arclength \( s \in [0, L] \), \( \tau \) returns the inertial \( x,y \) position of the path and we may write the path tangent angle at \( s \) is \( \Phi(s) = \arctan(\tau_y'(s)/\tau_x'(s)) \). In the case where \( \tau \) forms a closed circuit we have that \( \tau(0) = \tau(L) \), \( \tau'(0) = \tau''(L) \), \( \tau''(0) = \tau''(L) \), where \( \tau' \) and \( \tau'' \) are the element-wise first and second derivatives of \( \tau \). Given a path \( \tau \), we may then transform an inertial frame position \( p = (x, y) \) and yaw angle \( \psi \) into a path-relative pose as follows:

\[
\begin{align*}
    s(p) &= \arg\min_s \|\tau(s) - p\|_2, \quad (22a) \\
    e_y(p) &= -\sin(\Phi(s(p))) \cos(\Phi(s(p)))(p - \tau(s(p))), \quad (22b) \\
    e_{\psi}(p, \psi) &= \psi - \Phi(s(p)),
\end{align*}
\]

which are the path progress, the lateral displacement from the path, and the heading deviation from the path tangent corresponding to \( p \) and \( \psi \) respectively. These quantities, which are illustrated in Figure \[13\] are known as the coordinates of a Frenet reference frame \[39\] and they enable a straightforward expression of quantities which are important to the formulation of the autonomous racing problem as a dynamic game \[17\]. First, we may express the difference in track progress between two agents as \( s(p^i) - s(p^j) \). Maximization of this quantity in the game objective \( J^i \) captures the direct competition between agents \( i \) and \( j \) in racing. Furthermore, using the lateral deviation in \( (22b) \), we may easily express track boundary constraints for agent \( i \) as simple bounds on \( e_y(p^i) \), which are then included in the game constraints \( C^i \) \[3\]. Repeated evaluation of the potentially non-linear optimization problem (NLP) in \( (22a) \) can be avoided by directly describing the evolution of a vehicle’s kinematics in the Frenet reference frame \[39\]. In this work, we consider two vehicle models which can be expressed this way. The first is the Frenet-frame kinematic bicycle model which, for vehicle \( i \), has the state and input vector:

\[
\begin{align*}
    x^i_{\text{kin}} &= [v, s, e_y, e_{\psi}]^\top \in \mathcal{X}^i_{\text{kin}}, \quad (23a) \\
    u^i_{\text{kin}} &= [a, \delta_f]^\top \in \mathcal{U}^i_{\text{kin}}, \quad (23b)
\end{align*}
\]

where \( v \) and \( a \) are the velocity and acceleration of vehicle \( i \) along its direction of travel and \( \delta_f \) is the steering angle of the front wheel. The second is the Frenet-frame dynamic bicycle model which, for vehicle \( i \), has the state and input vector:

\[
\begin{align*}
    x^i_{\text{dyn}} &= [v_x, v_y, \omega, s, e_y, e_{\psi}]^\top \in \mathcal{X}^i_{\text{dyn}}, \quad (24a) \\
    u^i_{\text{dyn}} &= [a_x, \delta_f]^\top \in \mathcal{U}^i_{\text{dyn}}, \quad (24b)
\end{align*}
\]

where \( v_x \) and \( v_y \) are the longitudinal and lateral velocity, \( \omega \) is the yaw rate, and \( a_x \) is the longitudinal acceleration of vehicle \( i \). The following functions describe the state evolution of the kinematic or dynamic bicycle models and expressions for both can be found in the Appendix.

\[
\begin{align*}
    x^{i,\ast}_{\text{kin}} &= f^i_{\text{kin}}(x^i_{\text{kin}}, u^i_{\text{kin}}) \quad (25) \\
    x^{i,+}_{\text{dyn}} &= f^i_{\text{dyn}}(x^i_{\text{dyn}}, u^i_{\text{dyn}}) \quad (26)
\end{align*}
\]

These can then be concatenated to construct the game dynamics in \( (1) \):

\[
    f_\ast = \text{vec}(f^1_\ast, \ldots, f^M_\ast),
\]

where the subscript \( \ast \in \{\text{kin, dyn}\} \) indicates the vehicle model. A Frenet-frame dynamic game with either the kinematic or dynamic bicycle model can therefore be defined as:

\[
    \Gamma_\ast = (N, X_\ast, U_\ast, f_\ast, \{J^i_\ast\}_{i=1}^M, C_\ast).
\]

Direct expression of the vehicle kinematics in a Frenet reference frame offers two benefits in the formulation of racing tasks. The first is that it allows for the expression of progress maximization as a linear function of the state variable \( s \). The second is that it allows for track boundary constraints to be imposed as simple bounds on the state variable \( e_y \) \[40\], \[5\], \[41\]. However, this comes at the cost of introducing singularities at the centers of curvature (as seen in the numerator of \( (13) \)) which can cause numerical instabilities especially on tracks with tight turns. Furthermore, Frenet-frame kinematics also complicate the expression of obstacle avoidance constraints in multi-agent settings, which will be seen in our description of the racing scenarios in Section \[VIII\].

Our formulation, which applies ideas from contouring control \[37\], \[38\], \[42\] in the context of dynamic games, approximates the evolution of \( s \) and \( e_y \) without introducing singularities, and results in a dynamic game with inertial kinematics, where the approximations of \( s \) and \( e_y \) can be leveraged in a similar manner as before for simple description of racing tasks.

Let us first define the inertial-frame counterparts of the kinematic and dynamic bicycle models for vehicle \( i \), which have as state and input vectors:

\[
\begin{align*}
    \bar{x}^i_{\text{kin}} &= [\bar{v}, \bar{x}, \bar{y}, \bar{\psi}]^\top \in \bar{\mathcal{X}}^i_{\text{kin}}, \quad (29a) \\
    \bar{u}^i_{\text{kin}} &= [\bar{a}, \bar{\delta}_f]^\top \in \bar{\mathcal{U}}^i_{\text{kin}}, \quad (29b)
\end{align*}
\]

and

\[
\begin{align*}
    \bar{x}^i_{\text{dyn}} &= [\bar{v}_x, \bar{v}_y, \bar{\omega}, \bar{x}, \bar{y}, \bar{\psi}]^\top \in \bar{\mathcal{X}}^i_{\text{dyn}}, \quad (30a) \\
    \bar{u}^i_{\text{dyn}} &= [\bar{a}_x, \bar{\delta}_f]^\top \in \bar{\mathcal{U}}^i_{\text{dyn}}, \quad (30b)
\end{align*}
\]

respectively. The dynamics functions describe the state evolution of the inertial-frame kinematic or dynamic bicycle models and expressions for both are again included in the Appendix.

\[
\begin{align*}
    \bar{x}^{i,\ast}_{\text{kin}} &= \bar{f}^i_{\text{kin}}(\bar{x}^i_{\text{kin}}, \bar{u}^i_{\text{kin}}) \quad (31) \\
    \bar{x}^{i,+}_{\text{dyn}} &= \bar{f}^i_{\text{dyn}}(\bar{x}^i_{\text{dyn}}, \bar{u}^i_{\text{dyn}}) \quad (32)
\end{align*}
\]

We begin the derivation of our approximation by augmenting the state and input vectors for vehicle \( i \) with the variables \( \bar{s}^i \) and \( \bar{v}^i \) to obtain

\[
    \bar{x}_\ast = (\bar{x}_\ast, \bar{s}^i) \in \bar{\mathcal{X}}^i_\ast, \quad \bar{u}_\ast = (\bar{u}^i_\ast, \bar{v}^i) \in \bar{\mathcal{U}}^i_\ast,
\]

where we use the subscript \( \ast \in \{\text{kin, dyn}\} \) to indicate that this augmentation may be performed on the state and input...
vectors for either the kinematic and dynamic bicycle models. The variable \( \overline{s}_i \) represents our approximation of the progress \( s \) for vehicle \( i \) along a given path \( \tau \) and evolves according to simple integrator dynamics:

\[
\overline{s}_{i,k+1} = \overline{s}_{i,k} + \Delta t \cdot \overline{v}_{i,k},
\]

with initial condition \( \overline{s}_{i,0} = s(p_0^i) \), where \( \overline{v}_i \) is a decision variable that can be thought of as an approximate arc speed for agent \( i \). We denote the augmented dynamics function as

\[
\hat{\mathbf{f}}_s = \text{vec}(\hat{f}_s^1, \ldots, \hat{f}_s^M).
\]

Now, given an inertial position \( p_i \) and approximate progress \( \overline{s}_i \), we define the lag error for vehicle \( i \) as follows:

\[
e_i(p_i, \overline{s}_i) = \left[ -\cos \overline{\Phi}(\overline{s}_i), -\sin \overline{\Phi}(\overline{s}_i) \right] (p_i - \tau(\overline{s}_i)).
\]

Specifically, the lag error approximates the difference between \( \overline{s}_i \) and \( s(p_i) \). We note that the complement of the lag error can be written as \( e_c(p_i, \overline{s}_i) = [\sin \overline{\Phi}(\overline{s}_i), \cos \overline{\Phi}(\overline{s}_i)](p_i - \tau(\overline{s}_i)) \). This term is known as the contouring error and it approximates the lateral deviation of the vehicle from the path at \( \overline{s}_i \). It is straightforward to see through the illustration in Figure 1a that when \( \overline{s} \) is in a local neighborhood of \( s(p) \) then \( e_l \approx 0 \) implies that \( \overline{s} \approx s(p) \) and \( e_c \approx e_c(p) \). However, if \( \overline{s} \) is far from \( s(p) \), as in the case of Figure 1b then it is entirely possible that the difference between \( \overline{s} \) and \( s(p) \) is arbitrarily large despite \( e_l = 0 \). Therefore, in order to for \( \overline{s} \) to be a good approximation, we must not only drive \( e_l \) to zero, but we must also attempt to keep \( \overline{s} \) close to \( s(p) \). This leads us to make the following two modifications to the game objectives and constraints.

First, to minimize the lag error, we introduce it into the augmented game objective for agent \( i \) as follows:

\[
\hat{J}_s^i(\overline{u}_s^i, \overline{u}_{s}^{i-1}) = J_s^i(\overline{u}_s^i, \overline{u}_{s}^{i-1}) + \sum_{k=1}^{N} q_k e_i(p_k^i(\overline{u}_s^i), \overline{s}_k^i(\overline{v}^i))^2,
\]

where \( J_s^i \) is the original game objective for agent \( i \) in (28), \( q_k \gg 0 \) is a weight on the lag error cost, and \( p_k^i(\cdot) \) and \( \overline{s}_k^i(\cdot) \) denote functions which, given the input sequences \( \overline{u}_s^i \) and \( \overline{v}^i \), roll out the dynamics function \( \hat{f}_s^i \) up to time step \( k \) and return the \( x,y \)-position and approximate progress respectively. For brevity, we have omitted the explicit dependence of \( J_s^i \) on \( \overline{s}_i \); the path progress corresponding to the state \( \overline{s}_i,0 \). This can be easily computed via (22a) as \( \overline{s}_0 = s(p_0) \). Furthermore, under a slight abuse of notation, we replace the arguments of the original Frenet-frame game objectives in (28) (i.e. the first term in (38)) with the augmented input sequences to reflect that any terms in \( J_s^i \) which originally depended on \( s^i \) and \( s^{i-1} \) should be replaced with the approximations \( \overline{s}_i \) and \( \overline{s}_{i-1} \). Second, to keep \( \overline{s}_k^i \) close to \( s(p_k^i) \), we impose the simple bounds on the approximate arc speed input: \( |\overline{v}_i^i| \leq \max_v \). This helps in avoiding undesirable local minima of the lag error term in (38) which correspond to poor approximations with \( \overline{s}_k^i \) as shown in Figure 1b. We denote the concatenation of the arcspeed bounds for all agents \( i \in \{1, \ldots, M\} \) and time steps \( k \in \{1, \ldots, N\} \) as \( C_v(\overline{v}^1, \ldots, \overline{v}^M) \).

Next, we turn to the track boundary constraints which can no longer be expressed as simple bounds on \( e_y \). Instead, we may conveniently use the contouring error \( e_c \) in Figure 1a as an approximation to \( e_y \) to define the following constraints for agent \( i \):

\[
-w^-(\overline{s}_k^i(\overline{v}^i)) \leq e_c(p_k^i(\overline{u}_s^i), \overline{s}_k^i(\overline{v}^i)) \leq w^+(\overline{s}_k^i(\overline{v}^i)),
\]

where \( w^+, w^- : [0, L] \rightarrow \mathbb{R}_{++} \) are functions which, for a value of \( s \), return the distance from \( \tau(s) \) to the track boundaries in the directions normal to \( \tau \) at \( s \). Finally, we define the augmented constraint function \( \hat{C}_s \) as the concatenation of the original constraints of the dynamic game \( C_s \) in (28) with the arc speed bounds:

\[
\hat{C}_s(\overline{u}_s^1, \ldots, \overline{u}_s^M) = \text{vec}(C_s(\overline{u}_s^1, \ldots, \overline{u}_s^M), C_v(\overline{v}^1, \ldots, \overline{v}^M)),
\]

where through a similar abuse of notation as before, we replace the arguments of the original Frenet-frame game constraints \( C_s \) with the augmented input sequences to reflect that the original track boundary constraints in \( e_y \) should be replaced with \( w^\pm \). Combining the elements discussed above, we define the following dynamic game:

\[
\hat{\Gamma}_s = (\mathcal{N}, \hat{X}_s, \hat{U}_s, \hat{F}_s, \hat{J}_s|_{1 \leq i \leq M}, \hat{C}_s),
\]

which can be viewed as an approximation to the racing dynamic game \( \Gamma_s \) in (28), which uses the exact Frenet-frame kinematics when defining the game dynamics, agent objective, and constraint functions. Table 1 summarizes the components of each of the exact and approximate dynamic games which will be solved in the following numerical studies.

| | Dynamics | Objectives | Constraints |
|---|---|---|---|
| \( \Gamma_{kin} \) | \( \hat{\Gamma}_{kin} \) | \( \hat{J}_{kin}^M \) | \( \hat{C}_{kin} \) |
| \( \Gamma_{dyn} \) | \( \hat{\Gamma}_{dyn} \) | \( \hat{J}_{dyn}^M \) | \( \hat{C}_{dyn} \) |

VIII. NUMERICAL STUDY

In the following numerical studies, we examine the performance of our DG-SQP solver in solving for open-loop GNE of three head-to-head racing scenarios, which are defined in Part A. In Part B, we examine the effect of the proposed merit function and non-monotone line search strategy on the success rate of the DG-SQP solver via an ablation study on these components. In Part C, we investigate the effect of the proposed decaying regularization scheme through a sensitivity study on the regularization value and decay rate. In Part D, we compare the performance of our DG-SQP solver against the PATH solver from [18] and show improved performance when solving both the exact and approximate dynamic games \( \Gamma_s \) and \( \hat{\Gamma}_s \). Our
DG-SQP solver was implemented in Python and for the PATH solver, we use the Julia implementation PATHSolver.jl for comparison, but call it through a custom Python wrapper in order for us to easily pass identical definitions of the following dynamic games to it. Our implementation can be found at https://github.com/zhu-edward/DG-SQP.

A. Scenario Description

We perform our numerical studies on three head-to-head racing scenarios which were constructed to showcase the performance of our DG-SQP solver with different vehicle models and track geometries.

1) Scenario 1: In this scenario, we use the L-shaped track shown in the left plot of Figure 2 where the vehicles travel in the counter clockwise direction from the red start line. We model both vehicles using the kinematic bicycle model \( f_{\text{kin}} \) with parameters that correspond to a 1/10th scale RC car. The discrete time dynamics are obtained via 4th order Runge-Kutta discretization with a time step of \( \Delta t = 0.1 \text{s} \). The Frenet-frame, or exact, dynamic game \( \Gamma_{\text{kin}} \) is defined with the game dynamics \( f_{\text{kin}} = \text{vec}(f_{\text{kin}}^1, f_{\text{kin}}^2) \) and objectives:

\[
\begin{align*}
J_{\text{kin}}^1(u^1, u^2) & = \sum_{k=0}^{N-1} \left( \| u_k^1 \|_{R^1} + \| \Delta u_k^1 \|_{R^1} + q_1^e s_k^e(u^1) + q_1^l s_k^l(u^1) \right), \\
J_{\text{kin}}^2(u^2, u^1) & = \sum_{k=0}^{N-1} \left( \| u_k^2 \|_{R^2} + \| \Delta u_k^2 \|_{R^2} + q_2^e s_k^e(u^2) + q_2^l s_k^l(u^2) \right),
\end{align*}
\]

where \( \Delta u_k^i = u_k^i - u_{k-1}^i, \ u_{-1}^i \) is the input from the previous time step, \( R^i, P^i \geq 0 \) are weights on the quadratic input and input rate penalties, \( q_1^e, q_2^e > 0 \) are weights on the competition costs, and \( \| x \|_{A_k} = x^T A_k x \). The constraints \( C_{\text{kin}} \) are defined as follows:

\[
\begin{align*}
& u_l \leq u_k^i \leq u_u, \ i = 1, 2 \quad (42a) \\
& -w/2 \leq e_{y,k}^i(u^i) \leq w/2, \ i = 1, 2 \quad (42b) \\
& \| p_k^i(u^i) - p_k^i(u^2) \| \geq (r^1 + r^2)^2, \quad (42c)
\end{align*}
\]

which consist of input limits (42a), track boundary constraints (42b), and collision avoidance constraints (42c). In the above, \( u_l \) and \( u_u \) are the lower and upper bounds on the input, \( w \) is the track width, and \( r^1, r^2 \) are the radii of the circular collision buffers for the two vehicles. When implementing the collision avoidance constraint for the exact dynamic game \( \Gamma_{\text{kin}} \), the mapping \( p_k^i(u^i) \), which returns the inertial \( x-y \) position of agent \( i \) at time step \( k \), is written as:

\[
p_k^i(u^i) = \tau(s_k^i(u^i)) + \frac{e_{y,k}^i(u^i)}{\| \tau'(s_k^i(u^i)) \|_2} \left[ -\tau_y'(s_k^i(u^i)) \right].
\]

For the approximate dynamic game \( \hat{\Gamma}_{\text{kin}} \), we construct the game dynamics as \( f_{\text{kin}} = \text{vec}(f_{\text{kin}}^1, f_{\text{kin}}^2) \). The game objectives \( J_{\text{kin}}^i \) are obtained by replacing any occurrences of \( s_k^i(\cdot) \) with \( \hat{s}_k^i(\cdot) \) in the expressions for \( J_{\text{kin}}^i \) above. The game constraints are constructed as \( C_{\text{kin}} = \text{vec}(C_{\text{kin}}, C_{\text{vec}}) \) where \( e_{y,k}^i(u^i) \) is replaced with \( e_{\hat{y}}(p_k^i(u^i), \hat{s}_k^i(v^i)) \) in the original track boundary constraints (42b) in \( C_{\text{kin}} \). When implementing the collision avoidance constraint for the approximate dynamic game \( \hat{\Gamma}_{\text{kin}} \), the mapping \( p_k^i(u^i) \) simply extracts elements corresponding to \( x \) and \( y \) from the state vector of vehicle \( i \) at time \( k \). Finally, for this scenario, given an initial condition, we warm start the solver using trajectories obtained via a PID controller which attempt to maintain the speed \( v \) and lateral deviation \( e_y \) as specified in the initial condition.

2) Scenario 2: This scenario is identical to the previously described Scenario 1 in all aspects except for the game dynamics and warm start strategy, where instead, the vehicle is modeled using the dynamic bicycle model [44] and the tire forces are described via Pacejka tire models [44] with parameters that correspond to a 1/10th scale RC car. \( \Gamma_{\text{dy}} \) and \( \hat{\Gamma}_{\text{dy}} \) are therefore constructed in a similar manner as Scenario 1. Warm starting of the solvers is done with trajectories computed from an optimal control problem which attempts to track a pre-computed race line (the red line in the left plot of Figure 2) for the two vehicles but does not consider collision avoidance.

3) Scenario 3: In this scenario, we use a segment of the Austin F1 track corresponding to the first hairpin turn, as shown in the right plot of Figure 2 and again model the vehicles using the dynamic bicycle model. Model parameters are chosen which match those of a full scale race car. The agent objectives in this scenario remain identical to Scenario 1 and the constraints are also the same, with the exception of the the track boundary constraints which, due to the variable track width, are now written as:

\[
-w_{-}(s_k^i(u^i)) \leq e_{y,k}^i(u^i) \leq w_{+}(s_k^i(u^i)),
\]

where \( w_{-}, w_{+} : [0, L] \rightarrow \mathbb{R}_{++} \) are functions which return the width of the track in the directions normal to the parametric path at the given arclength \( s \). We also use the same warm start strategy as in Scenario 2 with the race line shown in the right plot of Figure 2.

B. An Ablation Study on the Merit Function and Non-Monotone Line Search Strategy

We first demonstrate the value of our proposed merit function and non-monotone line search strategy, by performing an ablation study on these components for the dynamic game described in Scenario 1 with a horizon of \( N = 25 \). We compare convergence results from the full DG-SQP approach.
presented in Algorithm 1 with two ablative cases. In the first, we replace the merit function \((\hat{\mu}, \hat{\sigma})\) with the sum of all of the agents’ objectives:

\[
\phi(u, s; \mu) = \sum_{i=1}^{M} J^i(u^1, \ldots, u^M) + \mu \|C(u) - s\|_1, \tag{43}
\]

which is a standard choice in nonlinear optimization approaches [29]. In the second, we do not use the non-monotone line search strategy and instead employ a standard backtracking line search which enforces the merit decrease condition \((20)\) at every iteration. The results are shown in Table II where GNE are solved for using DG-SQP and the two ablative cases. This is done over 100 randomly sampled joint initial conditions on the L-shaped track in Figure 2 where the vehicles start within 1.2 car lengths of each other and are traveling at velocities that exhibit at most a 25% difference. We count the number of trials where the solvers reached convergence with thresholds \(\rho_1, \rho_2, \rho_3 = 10^{-4}\). From the results, it can be clearly seen that certainly both the proposed merit function and the non-monotone line search strategy have a positive impact on the numerical robustness of the solver. In particular, we observe a 5.3% and 62.8% reduction in solver success rate when the two components are ablated. Comparing between the two components, it appears that the non-monotone line search has a larger impact on solver performance. We believe that this is due to the globalizing effect that the non-monotone scheme has on the solution process, which was shown in [18] for the PATH solver. While we do not prove the same property for the DG-SQP solver, it is likely that the non-monotone scheme can have a similar effect.

C. A Sensitivity Study on the Decaying Regularization Scheme

Next, we investigate the sensitivity of our approach to the regularization values and decay rates, by performing a sensitivity study over a grid of these values for the same two-player car racing dynamic game as the ablation study.

In particular, we again randomly sample 100 joint initial conditions and count the number of trials where the solver reached convergence for values of \(\epsilon_0 \in \{0, 0.1, 1, 10, 1000\}\) and \(\eta \in \{0.5, 0.65, 0.8, 0.95, 1\}\). Note that \(\epsilon_0 = 0\) corresponds to no regularization (in which case the value of the decay rate becomes irrelevant), and \(\eta = 1\) corresponds to no regularization decay, which corresponds to standard regularization schemes with a constant regularizer. The results of this study are shown in Figure 3 where we may observe 1) the importance of regularization, as none of the trials reached convergence when no regularization is used (first column), and 2) the sensitivity of solver performance to the choice of a constant \(\epsilon_0\), where the solver was only able to achieve a nontrivial success rate for a single setting of \(\epsilon_0\) (last row). Next, by looking along the rows of Figure 3 it can be seen that decaying regularization helps to alleviate the sensitivity of solver convergence to the choice in \(\epsilon_0\) as we observe nontrivial success rates over a range of magnitudes for \(\epsilon_0\). Finally, by looking along the columns of Figure 3 we see that while some settings of decay rate can certainly lead to better success rates, it appears that a fairly large range of values are sufficient to stabilize the solution process to achieve high success rates of >80%. While our analysis does not provide formal performance guarantees w.r.t. selection of \(\epsilon_0\) and \(\eta\), our sensitivity study suggests that large initial regularization values and slow decay are crucial to achieve high success rates for our solver.

D. Solver and Model Comparison for Racing Tasks

The PATH solver [18] is considered to be the state-of-the-art in solving for GNE of open-loop dynamic games and has been used effectively in many multi-agent navigation tasks [13], [14]. However, these works have primarily demonstrated its performance on games with inertial kinematic models. We now compare its performance against our DG-SQP approach for the exact and approximate games \(\Gamma_k\) and \(\hat{\Gamma}_k\) in the head-to-head racing scenarios described above.

1) Scenario 1: For this scenario, we randomly sample 500 collision-free initial conditions where the agents start within 1.2 car lengths of each other and are traveling at velocities that exhibit at most a 25% difference. For each initial condition, we solve for GNE of \(\Gamma_k\) and \(\hat{\Gamma}_k\) with horizon lengths of \(N \in \{15, 20, 25\}\) using both the PATH and DG-SQP solvers. Convergence thresholds are again set to \(\rho_1, \rho_2, \rho_3 = 10^{-4}\). The results are shown in Figure 4 where we first observe that for the exact dynamic game \(\Gamma_k\) our DG-SQP solver shows better performance when compared against the PATH solver, achieving higher success rates for all three horizon lengths. However, it is clear that both solvers struggle with finding a solution for longer horizon dynamic games. We note from the first column of Figure 4 that for the exact dynamic game \(\Gamma_k\), the majority of the failure cases for the PATH and DG-SQP solvers occur at the entrance of the chicane. Due to the many tight turns, this section of the track is where the numerical shortcomings of the Frenet-frame dynamics, as described in Section VII can be especially apparent. Next, we examine the performance of the solvers on the approximate dynamic
game $\hat{\Gamma}_{\text{kin}}$. Here, we observe significant improvements in success rate for the DG-SQP solver over all horizon lengths, with a success rate of over 85% in the case of $N = 25$. This is an $\sim 87x$ improvement when compared to the success rate of solving the exact dynamic game $\Gamma_{\text{kin}}$ with the same DG-SQP solver. When solving $\hat{\Gamma}_{\text{kin}}$ with the PATH solver, we see improvements over its performance with the exact dynamic game for the longer horizons of $N = 20$ and $N = 25$. However, we again see that DG-SQP achieves superior performance, with a $\sim 6x$ improvement compared to the PATH solver in the case of $N = 25$.

2) Scenario 2: For this scenario, we randomly sample 500 collision-free initial conditions about a pre-computed race line (the red line in the left plot of Figure 2) where the agents start within 1.2 car lengths of each other and at velocities which are within $\pm 0.75$ m/s of the race line velocity at the sampled $s_0$. We compare the success rates of solving for GNE from these initial conditions for the exact and approximate dynamic games $\Gamma_{\text{dyn}}$ and $\hat{\Gamma}_{\text{dyn}}$. This is done again for the set of horizon lengths of $N \in \{15, 20, 25\}$. The results are summarized in Figure 5 where we observe similar trends to the results from Scenario 1 in the case of the exact dynamic game $\Gamma_{\text{dyn}}$, which shows that our DG-SQP solver achieves similar or superior performance to PATH for all three values of horizon lengths. Turning our attention now to the approximate dynamic game, we again see that our DG-SQP solver achieves significantly higher success rates when solving $\hat{\Gamma}_{\text{dyn}}$, with an over $3x$ improvement over the case of $\Gamma_{\text{dyn}}$ with DG-SQP and $N = 25$. Interestingly, we observe a rather severe decrease in performance by the PATH solver when the dynamic bicycle model is used in the formulation of the approximate game. An example failure case is shown in Figure 6, where despite warm starting the solver with a feasible initial guess, the terminal iterate is infeasible. In contrast, a combination of the regularization scheme and the explicit enforcement of (linearized) constraints when computing the primal and dual step leads to the success of the DG-SQP solver. This encourages small steps especially at the beginning of the solution process and helps to maintain the feasibility of the iterates. On the other hand, we note that the PATH solver does not have any explicit regularization scheme or constraint enforcement when computing the primal and dual step at each
iterate, and instead only relies on a non-monotone line search scheme for dampening the iterates.

3) Scenario 3: Using a similar procedure to Scenario 2, we randomly sample 100 joint initial conditions about a precomputed raceline (the red line in the right plot of Figure 2) where the agents start within 3 car lengths of each other and at velocities which are within \( \pm 7.5 \) m/s of the race line velocity at the sampled \( s_q \). The success rates of solving for GNE of the approximate dynamic game \( \hat{\Gamma}_{\text{dyn}} \) from these initial conditions are shown in Figure 7 where we observe that, in the case of \( N = 25 \), our DG-SQP solver is able to achieve a success rate of 77% of the sampled initial conditions and out-performs the PATH solver by over 3x.

E. Comparing GNE From the Exact and Approximate Dynamic Games

We have just shown using three scenarios that significantly higher success rates can be achieved when solving for GNE of the approximate dynamic games \( \hat{\Gamma}_* \) with the DG-SQP solver. However, it should be clear that as our approximation scheme modifies the components of the exact dynamic game \( \Gamma_* \), we are therefore solving for the GNE of a related, but nevertheless different dynamic game. The question that naturally arises is then how the GNEs from the approximate dynamic game \( \hat{\Gamma}_* \) compare with those from the exact one \( \Gamma_* \). In order to answer this question, we now examine the solutions from Scenario 2 with \( N = 15 \). Specifically, compute the pairwise mean squared error (MSE) between the GNE from two comparison cases. The first compares the GNE of \( \Gamma_{\text{dyn}} \) for the samples where both PATH and DG-SQP were successful. The normalized MSE for this case is defined as follows:

\[
\frac{1}{N} \sum_{k=0}^{N} \sum_{i=1}^{M} \left\| u_{\text{DGSQP},k}^{i,*} - u_{\text{PATH},k}^{i,*} \right\|_2^2 \cdot D^{-1}.
\]

The second case compares the GNE of \( \Gamma_{\text{dyn}} \) and \( \hat{\Gamma}_{\text{dyn}} \) from DG-SQP, where we only consider samples where DG-SQP was able to successfully solve for GNE of both the exact and approximate dynamic games. The normalized MSE for this case is defined as follows:

\[
\frac{1}{N} \sum_{k=0}^{N} \sum_{i=1}^{M} \left\| \hat{u}_{\text{DGSQP},k}^{i,*} - u_{\text{PATH},k}^{i,*} \right\|_2^2 \cdot D^{-1},
\]

where \( D = \text{diag}(u_{\text{u}}) \) normalizes the errors using the input upper bound \( u_{\text{u}} \) from \([22a]\) and \( \hat{u}_{\text{DGSQP},k}^{i,*} \) denotes the first two elements of the GNE solution at time step \( k \) (recall that \( \hat{u} \) is the augmented input vector which includes the approximate arc speed \( \hat{v} \)). The distribution of these errors are shown in Figure 8 where we may first observe from the top histogram that, when solving the exact Frenet-frame dynamic game, the DG-SQP and PATH solvers, for the most part, produce the same solutions with a minimum, median, and mean MSE of \( 1.57 \times 10^{-6} \), \( 1.01 \times 10^{-5} \), and \( 2.00 \times 10^{-2} \) respectively. This serves as a baseline to illustrate the correctness of our approach. Looking at the bottom histogram, we see that although there are certainly outliers, the majority of the GNE from the approximate dynamic game are close to those from the exact dynamic game when the same solver is used. In this comparison case, we observe a minimum, median, and mean MSE of \( 2.54 \times 10^{-3} \), \( 3.05 \times 10^{-2} \), and \( 3.22 \times 10^{-2} \) respectively.

We now turn to specific examples from the histograms to examine the qualitative differences in the GNE from the two comparison cases in Figures 9 and 10. In particular, we visualize the three GNE pairs with the smallest MSE (“Best” column), largest MSE (“Worst” column), and the three pairs with a MSE that is closest to the mean value (“Average” column).

In Figures 10, we first compare the position traces arising from the GNE of the exact dynamic game \( \Gamma_{\text{dyn}} \) from the DG-SQP and PATH solvers. In the “Best” column, we observe that the GNE from the two solvers are identical. We note that in this comparison case, these examples are indicative of the majority of successful GNEs as evidenced by the minimum and median MSEs. In the “Average” and “Worst” columns, we observe that some differences arise especially towards the end of the game horizon. These instances correspond to the solvers finding different local GNE. However, we note that in...
evaluate (37) with \( \Phi = 0 \), which leads to \( e_l(p, \hat{s}) = r_x(\hat{s}) - x \). From the “Average” column, we see that differences between the GNE are minor and that importantly, the GNE of the approximate dynamic game still capture the competitive nature of the racing scenario. Furthermore, we note that as in the previous comparison, the outcome remains the same despite small perturbations to the GNE trajectories between the exact and approximate dynamic games. This holds true even for the examples in the “Worst” column despite larger discrepancies between the GNE of the exact and approximate dynamic games.

**IX. LIMITATIONS AND FUTURE WORK**

In this work, we have measured the performance of our DG-SQP solver against the state-of-the-art PATH solver using the metric of success rate and showed significant improvements in racing scenarios especially when approximate dynamics are used. Another metric which we have not discussed is that of solution time. This is especially important if we would like to use our DG-SQP solver in a real-time MPGP manner like in \([13]\) and \([14]\). However, we note that main shortcoming of our approach at this time is computational efficiency, with solutions taking up to two minutes for some of the successful trials with \( N = 25 \) in Scenario 3. This is due to two reasons. The first is that our solver requires the solution of a sequence of constrained quadratic programs, which can be computationally demanding especially when compared to the PATH solver, which at its core, solves a sequence of linear systems. Secondly, our solver is implemented in Python, which is again significantly slower than the C++ implementation of the PATH solver. Though we can certainly implement our solver in a different language, we anticipate that it would still be difficult to achieve real-time solutions for the long-horizon racing problems presented here due to the solution of QPs. As such, an important direction of future work is to improve the computational efficiency of our DG-SQP solver, which would allow us to leverage its solutions for real-time racing experiments. One possible approach could be to improve the quality of the initial guess through supervised learning of GNE as in \([45]\) and \([46]\). By warm starting the solver with the a neural network, which predicts GNE for a given joint initial condition, this would potentially reduce the number of solver iterations required to reach convergence. An alternative approach would be to leverage the numerical robustness of our DG-SQP solver to generate an extensive dataset of GNE from various initial conditions and track environments. This data could then be used to learn a game-theoretic value function which can be integrated in an optimal control scheme to aid in long-term strategic planning without the need for online solutions from our solver.

**X. CONCLUSIONS**

In this work, we have presented DG-SQP, an SQP approach to the solution of generalized Nash equilibria for open-loop dynamic games. We show that the method exhibits local linear convergence in the neighborhood of GNE and present several practical improvements to the vanilla SQP algorithm.
including a non-monotone line search strategy with a novel merit function and decaying regularization scheme. We further present an approximation scheme to Frenet-frame dynamic games which can be used to improve the performance of our solver in racing scenarios. We conducted an extensive numerical study on various head-to-head racing scenarios with both kinematic and dynamic vehicle models and different race tracks. This study showed that our DG-SQP solver outperforms the state-of-the-art PATH solver in terms of success rate when applied to interactive racing scenarios.

REFERENCES

[1] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, “Autonomous vehicles on the edge: A survey on autonomous vehicle racing,” IEEE Open Journal of Intelligent Transportation Systems, 2022.

[2] P. Wurm, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs et al., “Outrunning champion gran turismo drivers with deep reinforcement learning,” Nature, vol. 602, no. 7896, pp. 223–228, 2020.

[3] E. Kaufmann, L. Bauersfeld, A. Loquecchio, M. Müller, V. Koltun, and D. Scaramuzza, “Champion-level drone racing using deep reinforcement learning,” Nature, vol. 620, no. 7962, pp. 982–987, 2023.

[4] J. Betz, T. Betz, F. Fent, M. Geisslinger, A. Heilmeier, L. Hermansdorfer, T. Kavanu, A. Dragan, P. Karle, and N. Lienkamp et al., “Tum autonomous motorsport: An autonomous racing software for the indy autonomous challenge,” Journal of Field Robotics, vol. 40, no. 4, pp. 783–809, 2023.

[5] C. Jung, A. Finazzi, H. Seong, D. Lee, S. Lee, B. Kim, G. Gang, S. Han, and D. H. Shim, “An autonomous system for head-to-head race: Design, implementation and analysis; team kaist at the indy autonomous challenge,” arXiv preprint arXiv:2303.09463, 2023.

[6] E. L. Zhu and F. Borrelli, “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.

[7] 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.

[8] 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.

[9] T. Başar and G. J. Olsder, Dynamic noncooperative game theory. SIAM, 1999.

[10] F. Laine, D. Fridovich-Keil, C.-Y. Chiu, and C. Tomlin, “Multi-hypothesis interactions in game-theoretic motion planning,” in 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021, pp. 8016–8023.

[11] S. Le Cleac’h, M. Schwager, Z. Manchester et al., “Algames: A fast solver for constrained dynamic games,” in Proceedings of Robotics: Science and Systems, 2020.

[12] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin, “Efficient iterative linear-quadratic approximations for nonlinear multiplayer general-sum differential games,” in 2020 IEEE international conference on robotics and automation (ICRA). IEEE, 2020, pp. 1475–1481.

[13] X. Liu, L. Peters, and J. Alonso-Mora, “Learning to play trajectory games against opponents with unknown objectives,” IEEE Robotics and Automation Letters, 2023.

[14] L. Peters, A. Bajcsy, C.-Y. Chiu, D. Fridovich-Keil, F. Laine, L. Ferranti, and J. Alonso-Mora, “Contingency games for multi-agent interaction,” arXiv preprint arXiv:2304.05483, 2023.

[15] T. Kavanu, A. Yaraneri, and N. Mehr, “Potential ilqp: A potential-minimizing controller for planning multi-agent interactive trajectories,” in 17th Robotics: Science and Systems, RSS 2021. MIT Press Journals, 2021.

[16] J. F. Fisac, E. Bronstein, E. Stefanosson, S. Dadjie, S. S. Sastry, and A. D. Dragan, “Hierarchical game-theoretic planning for autonomous vehicles,” in Proceedings on Robotics and Automation (ICRA). IEEE, 2019, pp. 9590–9596.

[17] F. Facchinei and C. Kanzow, “Generalized nash equilibrium problems,” Annals of Operations Research, vol. 175, no. 1, pp. 177–211, 2010.
Edward Zhu received his B.S. in Mechanical Engineering from Villanova University, PA, USA, in 2015 and his M.S. in Mechanical Engineering from the University of California, Berkeley, CA, USA. He is currently a Ph.D. candidate in Mechanical Engineering at the University of California, Berkeley, CA, USA. His research interests include leveraging model-based optimal control and game-theoretic methods for prediction and planning in multi-agent environments.

Francesco Borrelli received the Laurea degree in computer science engineering from the University of Naples Federico II, Naples, Italy, in 1998, and the Ph.D. degree from ETH-Zurich, Zurich, Switzerland, in 2002. He is currently an Associate Professor with the Department of Mechanical Engineering, University of California, Berkeley, CA, USA. He is the author of more than 100 publications in the field of predictive control and author of the book Constrained Optimal Control of Linear and Hybrid Systems (Springer-Verlag). His research interests include constrained optimal control, model predictive control and its application to advanced automotive control and energy efficient building operation.

Dr. Borrelli received the 2009 National Science Foundation CAREER Award and the 2012 IEEE Control System Technology Award. In 2008, he became the Chair of the IEEE Technical Committee on Automotive Control.

APPENDIX

Here, we describe the kinematic and dynamic bicycle models which are used to formulate the dynamic games in Section VII.

The Frenet-frame kinematic bicycle model has the state and input vectors:

\[ x_{\text{kin}} = [v, s, e_y, e_\psi]^\top, \quad u_{\text{kin}} = [a, \delta]^\top, \]

and the continuous-time dynamics are written as:

\[ \dot{v} = a, \]
\[ \dot{s} = v \cos(e_\psi + \beta(\delta))/(1 - e_y \kappa(s)), \]
\[ \dot{e}_y = v \sin(e_\psi + \beta(\delta)), \]
\[ \dot{e}_\psi = v \sin(\beta(\delta))/(l' - \kappa(s)v \cos(e_\psi + \beta(\delta))/(1 - e_y \kappa(s)), \]

where \( \beta(\delta) = \arctan((\tan \delta \cdot U/(l' + l'')) \) is the side slip angle, \( l', l'' \) are the distance from the center of mass to the front and rear axles respectively, and \( \kappa(s) = (\tau_{y}^p(s)^{2} - \tau_{y}^{f}(s)^{2} + \tau_{y}^{r}(s)^{2})^{3/2} \) is the curvature of the path \( \tau \) at a given \( s \).

The Frenet-frame dynamic bicycle model has the state and input vectors:

\[ x_{\text{dyn}} = [v_x, v_y, \omega, s, e_y, e_\psi] \]
\[ u_{\text{dyn}} = [a_x, \delta]^\top, \]

and the continuous-time dynamics are written as:

\[ \dot{v}_x = a_x - \frac{1}{m} F_y^f \sin \delta - c_d v_x + \omega v_y, \]
\[ \dot{v}_y = \frac{1}{m} (F_x^f + F_y^f \cos \delta) - \omega v_x, \]
\[ \dot{\omega} = \frac{1}{I_z} (l' F_y^f \cos \delta - l'' F_y^r), \]
\[ \dot{s} = (v_x \cos(e_\psi) - v_y \sin(e_\psi))/(1 - e_y \kappa(s)), \]
\[ \dot{e}_y = v_x \sin(e_\psi) + v_y \cos(e_\psi), \]
\[ \dot{e}_\psi = \omega - \kappa(s)(v_x \cos(e_\psi) - v_y \sin(e_\psi))/(1 - e_y \kappa(s)), \]

where \( m \) and \( I_z \) are the mass and yaw moment of inertia of the vehicle and \( c_d \) is an aerodynamic drag coefficient. The lateral tire forces are modeled using a simplified Pacejka tire model [44]:

\[ F_y^f = D^f \sin(C^f + \arctan(B^f \alpha^f)) \]
\[ F_y^r = D^r \sin(C^r + \arctan(B^r \alpha^r)) \]

where \( B, C, \) and \( D \) are parameters obtained experimentally and \( \alpha^f \) and \( \alpha^r \) are the slip angles of the front and rear tires respectively:

\[ \alpha^f = - \arctan \left( \frac{\omega l_f + v_y}{v_x} \right) + \delta \]
\[ \alpha^r = \arctan \left( \frac{\omega l_r - v_y}{v_x} \right) \].

The inertial-frame kinematic bicycle model has the state and input vectors:

\[ \bar{x}_{\text{kin}} = [v, x, y, \psi]^\top, \quad \bar{u}_{\text{kin}} = [a, \delta]^\top, \]
and the continuous-time dynamics are written as:
\[
\begin{align*}
\dot{v} &= a, \\
\dot{x} &= v \cos(\beta(\delta) + \psi), \\
\dot{y} &= v \sin(\beta(\delta) + \psi), \\
\dot{\psi} &= \frac{v \sin(\beta(\delta))}{l},
\end{align*}
\]

The inertial-frame dynamic bicycle model has the state and input vectors:
\[
\bar{x}_{\text{dyn}} = [v_x, v_y, \omega, x, y, \psi]^T, \quad \bar{u}_{\text{dyn}} = [a_x, \delta]^T,
\]

and the continuous-time dynamics are written as:
\[
\begin{align*}
\dot{v}_x &= a_x - \frac{1}{m} F_f^f \sin \delta - c_d v_x + \omega v_y, \\
\dot{v}_y &= \frac{1}{m} (F_r^y + F_f^f \cos \delta) - \omega v_x, \\
\dot{\omega} &= \frac{1}{I_z} (l_f^f F_f^f \cos \delta - l_r^r F_r^y), \\
\dot{x} &= v_x \cos \psi - v_y \sin \psi, \\
\dot{y} &= v_x \sin \psi + v_y \cos \psi, \\
\dot{\psi} &= \omega.
\end{align*}
\]

We obtain the discrete-time Frenet-frame dynamics \(f_{\text{kin}}, f_{\text{dyn}}\), and inertial-frame dynamics \(\bar{f}_{\text{kin}}, \bar{f}_{\text{dyn}}\) via 4-th order Runge-Kutta discretization with a time step of \(\Delta t = 0.1s\).