Stable, Efficient Solutions for Differential Games with Feedback Linearizable Dynamics

David Fridovich-Keil, Vicenç Rubies Royo, and Claire J. Tomlin
EECS, University of California, Berkeley
Berkeley, CA 94720
{dfk, vrubies, tomlin}@eecs.berkeley.edu

Abstract

Iterative linear-quadratic (ILQ) methods are widely used for nonlinear optimal control. Recent work has applied similar methodology in the setting of general-sum differential games. Here, ILQ methods are capable of finding local Nash equilibria in interactive motion planning problems in real-time. As in most iterative procedures, however, convergence and runtime can be sensitive to initial conditions and hyperparameter choices. In this paper, we focus our attention on a broad class of dynamical systems which are feedback linearizable, and exploit this structure to improve both algorithmic reliability and runtime. We showcase our method in three traffic scenarios, and results confirm that our method converges significantly more often and more quickly than was possible without exploiting system structure.

1 Introduction

In robotics, a wide variety of decision making problems, including low-level control, motion planning, and task planning, are often best expressed as optimal control problems. Dynamic game theory—the study of games played over time—provides a natural extension of optimal control to the multi-agent setting. For example, vehicles at an intersection (e.g., Fig. 1a) mutually influence one another as they attempt to balance making forward progress in a desired direction while avoiding collision. Abstractly, dynamic games provide each agent, or "player," a separate input to the system, and allow each player to have a different cost function which they wish to optimize.

We shall consider dynamic games played in continuous time, or differential games. Recently, the community has shown renewed interest in dynamic and differential games, with a variety of new approximate algorithms for identifying locally optimal play. For example, Sadigh et al. [7] optimize the behavior of a self-driving car while accounting for the reaction of a human driver. Wang et al. [10] demonstrate a real-time iterative best response algorithm for planning competitive trajectories in a 6-player drone racing game. Building upon the earlier sequential linear-quadratic method of [6] and the well-known iterative linear-quadratic regulator (ILQR) [5, 4], our own prior work [2] solves warm-started 3-player differential games in under 50 ms each.

In this paper, we extend and improve upon our previous work [2] by exploiting the feedback linearizable structure in a broad class of dynamical systems, including quadcopters and the planar unicycle and bicycle models commonly used to model automobiles. We establish theoretical equivalence between the solutions identified using our new algorithm and those which do not exploit the feedback linearizable structure. By exploiting this structure, however, our algorithm is able to take much larger steps at each iteration and generally converge to an equilibrium more quickly and more reliably than was previously possible. Our work facilitates a host of interesting future directions in the field, including learning costs for each player and studying the topology of local Nash equilibria.

This work is under review for ICRA 2020; a preprint is available online [3].

Abstract for: Smooth Games Optimization and Machine Learning Workshop (NeurIPS 2019), Vancouver, Canada.

2 Problem Formulation

We consider N-player, general-sum differential games with control-affine dynamics. That is, we presume that the game state $x \in \mathbb{R}^n$ evolves as

$$\dot{x} = f(x) + \sum_{i=1}^{N} g_i(x)u_i \ , \tag{1}$$

where $u_i \in \mathbb{R}^{k_i}$ is the control input of player i. In our examples (Section 4), x will be the concatenated states of multiple subsystems, but this is not strictly necessary. We assume that (1) is full state feedback linearizable [8, Chapter 9]. That is, there exist outputs $y = h(x) \in \mathbb{R}^k$ (where $k = \sum_i k_i$) with well-defined vector relative degree $(r_1, \dots, r_k), \sum_i r_i = n$, such that y and finitely many of its time derivatives evolve linearly as a function of some auxiliary inputs $z_i \in \mathbb{R}^{k_i}$, i.e.,

$$[y_1^{(r_1)}, \dots, y_k^{(r_k)}]^T = z := [z_1^T, \dots, z_N^T]^T, \text{ for control} \tag{2}$$

$$u_i := u_i(x, z_i) = M^{-1}(x)\big(z_i - m(x)\big). \tag{3}$$

$M(x)$ is called the *decoupling matrix*, and $m(x)$ is called the *drift term*. We shall denote the states of this linearized system ξ, which evolve as $\dot{\xi} = A\xi + Bz$ for readily-derivable A, B. For further information on linearization by state feedback, we direct the reader to [8, Chapter 9].

Next, we suppose that each player i wishes to minimize a running cost ℓ_i over finite time horizon T:

$$J_i(u_1, \dots, u_N) := \int_0^T \ell_i(t, x, u_1, \dots, u_N)dt \ . \tag{4}$$

We require ℓ_i to be C^2 in $x, u_j, \forall j$, uniformly in time t, and note that ℓ_i may also be expressed in terms of linearized coordinates, overloaded as $\ell_i(t; \xi; \dots)$. Player i's total cost J_i then depends explicitly upon each player's control input signal $u_i(\cdot)$ and implicitly upon the initial condition $x(0)$.

Finally, we presume that each player i has access to the state x at every time t, but *not* other players' control inputs $u_j, j \neq i$, i.e. $u_i(t) \equiv \gamma_i\big(t, x(t)\big)$ for some measurable function $\gamma_i : [0, T] \times \mathbb{R}^n \to \mathbb{R}^{k_i}$. We shall denote the set of such functions Γ_i. For clarity, we shall also overload the notation of costs $J_i(\gamma_1; \dots; \gamma_N) \equiv J_i\big(\gamma_1(\cdot, x(\cdot)), \dots, \gamma_N(\cdot, x(\cdot))\big)$. Thus equipped with dynamics (1), costs (4), and this memoryless information pattern we seek *local* Nash equilibria of the game.

3 Method

Like the original iterative LQ game algorithm [6, 2], we proceed from a set of initial strategies γ_i for each player—understood now to map from (t, ξ) to z_i—and iteratively refine them by solving LQ approximations. Our main contribution, therefore, lies in the transformation of the game itself into the coordinates ξ, z_i which correspond to feedback linearized dynamics. As shown in Section 4, solving in the transformed coordinates is much more reliable and converge at least as quickly.

We begin at the given initial condition $\xi(0)$ for the linearized system and strategies γ_i^0 for each player. Note that these strategies define control laws *for the linearized system*, i.e. $z_i(t) \equiv \gamma_i\big(t, \xi(t)\big)$.

At each iteration, we first integrate the linearized dynamics (2) forward to obtain the current operating point $\mu = (\hat{\xi}(\cdot), \{\hat{z}_i(\cdot)\})$. Then, we compute a quadratic approximation to each player's running cost in terms of the variations $\delta\xi := \xi - \hat{\xi}$ and $\delta z_j := z_j - \hat{z}_j$

$$\ell_i(t; \xi; \dots) - \ell_i(t; \hat{\xi}; \dots) \approx l_i(t)^T \delta\xi + \frac{1}{2}\Big(\delta\xi^T Q_i(t)\delta\xi + \sum_{j=1}^{N} \delta z_j^T R_{ij}(t)\delta z_j\Big), \tag{5}$$

using the chain rule to differentiate through the diffeomorphic change of coordinates $\xi = \lambda(x)$ and compute the terms l_i, Q_i and R_{ij} for each player.

Equipped with linear dynamics (2) and quadratic costs (5), the solution of the resulting general-sum LQ game is given by a set of coupled Riccati differential equations, which may be derived from the first order necessary conditions of optimality for each player [1, Chapter 6]. In practice, we

numerically solve these equations in discrete-time using a time step of Δt. If a solution exists at the p^{th} iteration, it is known to take the form $\gamma_i^p(t,\xi) \equiv \hat{z}_i(t) - P_i^p(t)\big(\xi(t) - \hat{\xi}(t)\big) - \alpha_i^p(t)$ for matrix $P_i^p(t)$ and vector $\alpha_i^p(t)$ [1, Corollary 6.1].

We cannot simply use these strategies at the $(p+1)^{\text{th}}$ iteration or we risk diverging, however, without further assumptions on the curvature and convexity of running costs ℓ_i. In fact, these costs are generally nonconvex when expressed in terms of ξ and z_j, which necessitates some care in updating strategies. To address this issue, we follow a common practice in the ILQR and sequential quadratic programming literature (e.g., [9]) and introduce a step size parameter $\eta \in (0, 1]$:

$$\gamma_i^p(t,\xi) = \hat{z}_i(t) - P_i^p(t)\big(\xi(t) - \hat{\xi}(t)\big) - \eta\alpha_i^p(t). \tag{6}$$

Observe that, taking $\eta = 0$ and recalling that $\xi(0) = \hat{\xi}(0)$, we recover the previous open-loop control signal $\gamma_i^p(t,\xi) = \hat{z}_i, \forall t \in [0,T]$. Taking $\eta = 1$, we recover the LQ solution from this iteration. As is common in the literature, we perform a backtracking linesearch on η, starting with initial value η_0 and terminating when the trajectory that results from (6) satisfies a trust region constraint at level ϵ. In our experiments, we use an L_∞ constraint, i.e. $\|\xi(t) - \hat{\xi}(t)\|_\infty < \epsilon, \forall t$, and also check to ensure that M^{-1} exists at each time.

We conclude this section with several remarks about the theoretical soundness of our approach and the overall impact of exploiting feedback linearization.

Remark 1 *(Criterion for Convergence to Local Nash Equilibrium) Suppose that our algorithm converges to strategies $\{\gamma_i^*\}$. Then, from [2, Theorem 1] and presuming the invertibility of M we have that if Hessians $D_{xx}\big(\ell_i(t)\big), D_{u_iu_j}\big(\ell_i(t)\big) \succeq 0, \forall t$ at convergence, then the* open-loop *controllers defined by $z_i(t) = \hat{z}_i^*(t)$ comprise a local Nash equilibrium in open-loop strategies for the original system. That is, taking*

$$u^*(t) = M^{-1}\Big(\lambda\big(\hat{\xi}^*(t)\big)\Big)\Big(\hat{z}^*(t) - m\big(\lambda(\hat{\xi}^*(t))\big)\Big)$$

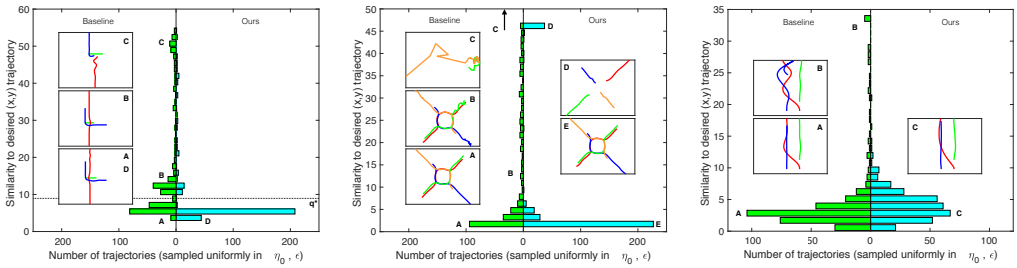
we obtain a local Nash equilibrium in open loop strategies for the game.

Remark 2 *(Benefits of Feedback Linearization) In comparison to the non-feedback linearizable case, the linearized dynamics* (2) *are trajectory- (and hence iteration-) independent. That is, in the non-feedback linearizable case [2], each iteration begins by constructing a Jacobian linearization of dynamics* (1)*; this is superfluous in our case. As a consequence, large changes in auxiliary input z between iterations—which lead to large changes in state trajectory—are trivially consistent with the feedback linearized dynamics* (2)*. By contrast, a large change in control u may take the nonlinear dynamics* (1) *far away from the previous Jacobian linearization, which causes the algorithm from [2] to be fairly sensitive to step size η and trust region size ϵ.*

Remark 3 *(Drawbacks of Feedback Linearization) While many systems of interest (e.g., manipulators, cars, and quadrotors) and feedback linearizable, this is not true of all systems. Additionally, there are two major drawbacks of our algorithm. First, we must take care to avoid singularities (regions in which M^{-1} does not exist), especially when constructing the costs. Second, and more importantly, the transformed costs $\ell_i(t;\xi;\dots)$ may have much more varied, extreme curvature than the original costs $\ell_i(t,x,\dots)$. In some cases, this can make our approach sensitive to linesearch parameters η_0 and ϵ, even offsetting the benefits from Remark 2. We defer further discussion and empirical study for Section 4.*

4 Results

To showcase the benefits of our feedback linearization-based approach, we study the empirical sensitivity of solutions to the initial step size η_0 and trust region size ϵ hyperparameters from Section 3. We first consider a three-player intersection example and compare the strategies identified by our approach with those identified on the original dynamics [2]. Here, two cars modeled with bicycle dynamics with 6 state dimensions and a pedestrian modeled as a unicycle with 4 state dimensions navigate an intersection. We place quadratic penalties on each player's distance from the appropriate lane center and from a fixed goal location, as well as on the difference between speed v and a fixed nominal speed $\bar{v}$. Players are penalized quadratically within a fixed distance of each other.



(a) Intersection (b) Roundabout (c) Overtaking

Figure 1: Comparison of the proposed algorithm with the state of the art [2] for three interactive traffic scenarios. Histograms (left, baseline; right, ours) show that our method is more numerically stable and converges more frequently. Labelled insets show a typical trajectory for the associated bin, and the dotted horizontal line in (a) shows a sample threshold q^* to distinguish (un)successful trials in the intersection problem.

In order to assess the quality of a trajectory $\mu = (\xi, z_1, \ldots, z_N)$ generated by a particular algorithm, we define the metric $q(\mu, \tilde{\mu}) := \max_{t\in[0,T]} \|\xi(t) - \tilde{\xi}(t)\|_{2,(p_x,p_y)}$. Here, we take $\tilde{\mu}$ to be the equilibrium trajectory which that algorithm ideally converges to. The norm measures Euclidean distance only in the position dimensions (p_x, p_y). Trajectories that diverge or converge to unreasonable solutions yield high values for q, while trajectories that closely match $\tilde{\mu}$ incur low values. We fix the initial conditions and cost weights identically for both algorithms. Fig. 1a shows histograms of solution quality q for 324 random pairs (η_0, ϵ) for each algorithm. We observe that solving the game using feedback linearization converges more reliably than solving it for the original system. Moreover, for converged trajectories with low q-value, the average computation time was 0.3982 ± 0.3122 s (mean $\pm$ standard deviation) for our method and 0.8744 ± 0.9582 s for the baseline.

Unfortunately, as per Remark 3, in some cases the cost landscape gets much more complicated when expressed in linearized system coordinates ξ, z_i. For example, a simple quadratic penalty on a single player's speed difference from nominal $\bar{v}$ is nonconvex and non-smooth near the origin when expressed as a function of linearized system state ξ: $(v - \bar{v})^2 \iff (\bar{v} - \|(\dot{p}_x, \dot{p}_y)\|_2)^2$. Consequences vary; the effect is negligible in the intersection example from Fig. 1a, but it is more significant in the roundabout example shown in Fig. 1b, where cars must slow down before turning into the roundabout. Fortunately, in practical settings of interest it is typically straightforward to design smooth, semantically equivalent costs explicitly as functions of the linearized system coordinates ξ. For example, we can replace this nominal speed cost with a time-varying quadratic penalty in that player's position (p_x, p_y): $(v - \bar{v})^2 \implies \|\big(p_x(t) - \bar{p}_x(t), p_y(t) - \bar{p}_y(t)\big)\|_2^2$, where $\big(\bar{p}_x(\cdot), \bar{p}_y(\cdot)\big)$ defines the point on the lane center a distance $\bar{v}t$ from the initial condition.

We demonstrate the effectiveness of this substitution in two examples—merging into a roundabout, and overtaking a lead vehicle. Results for the roundabout merging and overtaking examples are shown in Figures 1b and 1c, respectively. We see that, again, our approach converges more frequently than the method of [2]. Moreover, when successful, the average computational time in the roundabout example was 0.2797 ± 0.1274 s for our method and 0.4244 ± 0.5259 s for the baseline. Runtimes for the overtaking example were 0.5112 ± 0.3228 s (ours) and 0.4417 ± 0.4142 s (baseline). Runtimes for our approach cluster more tightly around the mean, indicating a more reliable convergence rate.

5 Conclusion

We have presented a novel algorithm for identifying local Nash equilibria in differential games with feedback linearizable dynamics by repeatedly solving LQ games in the linearized system coordinates, rather than in the original system coordinates. By working with the linearized system, our algorithm becomes less sensitive to parameters such as initial step size and trust region size, which often leads it to converge more quickly. Our method is fully general, i.e. any cost expressed in terms of nonlinear system coordinates may also be expressed in terms of linearized coordinates, which implies sufficient conditions for fixed points of our algorithm to be local Nash equilibria. In some cases transforming costs in this way makes the cost landscape more complicated, though it is often possible to design semantically equivalent replacement costs directly in the linearized coordinates. We test our method in a variety of competitive traffic scenarios, where experiments confirm the computational stability and efficiency of our approach. These results bode well for future research related to learning costs for each player and studying the topology of local Nash equilibria.

References

[1] Tamer Başar and Geert Jan Olsder. *Dynamic Noncooperative Game Theory*.

[2] David Fridovich-Keil, Ellis Ratner, Anca D Dragan, and Claire J Tomlin. Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. *arXiv preprint arXiv:1909.04694*, 2019.

[3] David Fridovich-Keil, Vicenc Rubies-Royo, and Claire J Tomlin. An iterative quadratic method for general-sum differential games with feedback linearizable dynamics. *arXiv preprint arXiv:1910.00681*, 2019.

[4] David H Jacobson and David Q Mayne. Differential dynamic programming. 1970.

[5] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO*, pages 222–229, 2004.

[6] H Mukai, A Tanikawa, I Tunay, IN Katz, H Schättler, P Rinaldi, IA Ozcan, GJ Wang, L Yang, and Y Sawada. Sequential linear quadratic method for differential games. In *Proc. 2nd DARPA-JFACC Symposium on Advances in Enterprise Control*, pages 159–168. Citeseer, 2000.

[7] Dorsa Sadigh, Shankar Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science & Systems*.

[8] Shankar Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer, 1999.

[9] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE, 2014.

[10] Zijian Wang, Riccardo Spica, and Mac Schwager. Game theoretic motion planning for multi-robot racing. In *Distributed Autonomous Robotic Systems*, pages 225–238. Springer, 2019.