

# Online Learning of Feasible Strategies in Unknown Environments

Santiago Paternain and Alejandro Ribeiro

**Abstract**—An environment is defined as a set of constraint functions that vary arbitrarily over time. An agent wants to select feasible actions that keep all the constraints negative, but must do so causally. I.e., the dynamical system that determines actions is such that only their time derivatives can depend on the current constraints. An environment is said viable if there exists an action that can satisfy the constraints for all times. The fit of a trajectory is defined as a vector that integrates the constraint violations over time and is used to measure the extent to which a policy succeeds in learning feasible actions. An online saddle point controller is proposed to control fit and shown to do so under minimal technical conditions. The online saddle point controller pushes actions along a linear combination of the constraint negative gradients and dynamically adapts the coefficients of this linear combination to find appropriate weightings. Concepts are illustrated throughout with the problem of a shepherd that wants to stay close to all sheep in a herd. Numerical experiments show that the controller allows the shepherd to do so.

## I. INTRODUCTION

A shepherd wants to stay close to a herd of sheep. The movement of the sheep is unknown a priori and arbitrary, perhaps strategic. However, they stay in a configuration at which it is possible for the shepherd to stay within a prescribed distance of all of them. The shepherd observes the sheep movement and responds to this online information through a causal dynamical system. This paper shows that an online version of the saddle point algorithm of Arrow and Hurwicz [1] succeeds in keeping the shepherd close to all sheep. More generically, we consider an agent that operates in an environment that we define as a time varying function of the agents actions – the distance between the sheep and the shepherd. We further define a viable environment as one in which the constraints are satisfiable over time – whatever the sheep do, the shepherd can position himself close to all of them – and the fit as the accumulation of the constraint violation over time – the integrals of the distance of the shepherd to each sheep. The online saddle point controller that we introduce here achieves bounded fit. The agent learns a feasible trajectory even if constraint functions are unknown ex ante but revealed ex post.

Designing a controller to let an agent satisfy a number of convex constraints is related to the problem of letting the agent minimize a convex cost. The latter is a canonical problem that can be solved with a gradient descent controller that pushes the agent along the negative gradient of the cost

function; see e.g., [2]. These costs can represent natural constraints or artificial potentials, can be even allowed to be stochastic [3]–[5], and are common methodologies to solve, e.g., navigation problems [6], [7]. The problem of satisfying a number of constraints can be formally posed as the determination of a saddle point of a Lagrangian function which can be found with the saddle point algorithm of Arrow and Hurwicz [1]. This algorithm interprets each constraint as a separate potential and descends on a linear combination of their negative gradients. The coefficients of these linear combinations are multipliers that adapt dynamically so as to push the agent to a region where all constraints are satisfied. Saddle point algorithms and variations are popular in several application domains, see e.g., [8].

The novelty of this work is to consider constraints that are unknown a priori and can change arbitrarily over time. This is related to the minimization of unknown convex costs that vary arbitrarily over time, in which case the problem can be formulated in the language of regret [9], [10]. In regret formulations, agents operate online by selecting plays that incur a cost selected by nature. The cost functions are revealed to the agent ex post and used to adapt subsequent plays. Regret is defined as the accumulation over time of the loss difference between the online learner and a offline learner to which cost functions have been revealed beforehand. It is remarkable that an online version of gradient descent is able to find plays whose regret grows at a sublinear rate [11], [12] – therefore suggesting vanishing per-play penalties of online plays with respect to the clairvoyant play.

We formulate online learning of strategies that are feasible with respect to an unknown and arbitrarily varying environment in the language of fit, which we define as a vector that accumulates the violation of each constraint over time (Section II). We propose to control fit growth with the use of an online saddle point controller that moves along a linear combination of the negative gradients of the instantaneous constraints. The coefficients of these linear combinations are adapted dynamically as per the instantaneous constraint functions as well (Section III). This online saddle point controller is a generalization of (offline) saddle point in the same sense that an online gradient controller generalizes (offline) gradient descent. We show that if there exists a viable strategy that can satisfy the environmental constraints at all times, the online saddle point controller achieves bounded fit (Theorem 1). Throughout the paper we illustrate concepts with the problem of a shepherd that has to stay close to a herd of sheep (Section II-B). A numerical analysis of this problem closes the paper (Section IV).

Work in this paper is supported by NSF CCF-1017454, NSF CCF-0952867 and ONR N00014-12-1-0997. The authors are with the Department of Electrical and Systems Engineering, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104. Email: {spaternain, aribeiro}@seas.upenn.edu.

**Notation.** A multivalued function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  is defined by stacking the components functions, i.e.,  $f := [f_1, \dots, f_m]^T$ . The notation  $\int f(x)dx := [\int f_1(x)dx, \dots, \int f_m(x)dx]^T$  represents a vector stacking each individual integral. An inequality  $x \leq y$  between vectors of equal dimension  $x, y \in \mathbb{R}^n$  is interpreted componentwise. An inequality  $x \leq c$  between a vector  $x = [x_1, \dots, x_n]^T \in \mathbb{R}^n$  and a scalar  $c \in \mathbb{R}$  means that  $x_i \leq c$  for all components of  $x$ .

## II. VIABILITY AND FEASIBILITY

We consider a continuous time environment in which an agent selects an action that results in a time varying set of penalties. Using  $t$  to denote time and  $x \in X \subseteq \mathbb{R}^n$  to denote the agent's action, the penalties incurred at time  $t$  are given by the value  $f(t, x)$  of the vector function  $f : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^m$ . We interpret the vector penalty function  $f$  as a definition of the environment. Our interest in this paper is in situations where the agent is faced with an environment  $f$  and must choose an action  $x \in X$  – or perhaps a trajectory  $x(t)$  – that guarantees nonpositive penalties  $f(t, x(t)) \leq 0$  for all times  $t$  not exceeding a time horizon  $T$ . Since the existence of this trajectory depends on the specific environment, we start by defining a viable environment as one in which it is possible for the agent to select an action with nonpositive penalty for times  $0 \leq t \leq T$  as we formally specify next.

**Definition 1 (Viable environment).** We say that a given environment  $f : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^m$  is viable over the time horizon  $T$  for an agent that selects actions  $x \in X$  if there exists an action  $x^\dagger \in X$  such that

$$f(t, x^\dagger) \leq 0, \quad \text{for all } t \in [0, T]. \quad (1)$$

An action  $x^\dagger$  satisfying (1) is said feasible and the set  $X^\dagger := \{x^\dagger : f(t, x^\dagger) \leq 0, \text{ for all } t \in [0, T]\}$  is termed the feasible set of actions.

In subsequent definitions and analyses we require integrability of the environment  $f$  as well as convexity with respect to  $x$  as we formally state next.

**Assumption 1.** The function  $f(t, x)$  is integrable with respect to  $t$  in the interval  $[0, T]$ .

**Assumption 2.** The function  $f(t, x)$  is convex with respect to  $x$  for all times  $t \in [0, T]$ .

If the environment  $f$  is known beforehand, the question of finding an action in a viable environment that gives non positive penalties reduces to finding a solution to the following constrained convex optimization program:

$$\begin{aligned} \min_{x \in X} \quad & 0 \\ \text{s.t.} \quad & f(t, x) \leq 0, \text{ for all } t \in [0, T]. \end{aligned} \quad (2)$$

A number of algorithms are known to solve this problem. Here, we consider the problem of adapting a strategy  $x(t)$  when the function  $f(t, x)$  is arbitrary and revealed causally. I.e., we want to choose the action  $x(t)$  using observations of viability  $f(t, x)$  in the open interval  $[0, t)$ . This implies

that  $f(t, x(t))$  is not observed before choosing  $x(t)$ . The action  $x(t)$  is chosen ex ante and the corresponding viability  $f(t, x(t))$  is incurred ex post.

### A. Fit

We evaluate the performance of trajectories  $x(t)$  through the concept of fit. We define the fit of the trajectory  $x(t)$  as the accumulated value of the penalties  $f(t, x(t))$  incurred for times  $t \in [0, T]$ ,

$$\mathcal{F}_T := \int_0^T f(t, x(t)) dt. \quad (3)$$

The fit  $\mathcal{F}_T$  can be interpreted as a performance loss associated with online causal operation as opposed to offline clairvoyant operation. In the latter, since the function  $f$  is known, an action that entails a non positive cumulative penalty can always be selected for a viable environment. If the fit  $\mathcal{F}_T$  is positive in a viable environment we are in a situation in which, had the environment  $f$  be known a priori, we could have selected an action  $x^\dagger$  with  $f(t, x^\dagger) \leq 0$ . The fit measures how far the trajectory  $x(t)$  comes from achieving that goal. This observation motivates the definition of strongly feasible trajectories.

**Definition 2 (Strong Feasibility).** Given an environment  $f : \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^m$  and a trajectory  $x(t)$  we say that the trajectory  $x(t)$  is strongly feasible in the environment if the fit  $\mathcal{F}_T$  is bounded for all  $T$ . I.e., if there exists a constant vector  $C$  such that for all times  $T$  it holds,

$$\mathcal{F}_T := \int_0^T f(t, x(t)) dt \leq C. \quad (4)$$

**Remark 1 (Not every trajectory is strongly feasible).**

Notice that in definition (4) we are considering the integral of a measurable function in a finite interval, hence the integral will always be bounded by a constant. Yet the constant could be dependent on the time horizon  $T$ , in which case the trajectory is not strongly feasible because the bound is not independent of  $T$ .

Having the fit satisfy  $\mathcal{F}_T \leq C$  irrespectively of  $T$  is an indication that that  $x(t)$  approaches the feasible set of actions  $X^\dagger$ , so that the integral stops growing. This need not be true as it is possible to achieve bounded fit by having  $f(t, x(t))$  oscillate around 0. In general, the possibility of having small fit by a trajectory that does not approach  $X^\dagger$  is a limitation of the concept of fit. Alternatively, we can think of the feasible offline policy  $x^\dagger$  as fixing a budget for the accumulated cost across time  $\int_0^T f(t, x(t)) dt \leq 0$ . A strongly feasible online policy meets that budget within a constant factor  $C$  – perhaps by overspending at some times and underspending at some other times.

In section III we provide an algorithm for solving the problem of finding strongly feasible trajectories according to Definition 2. Before that, we clarify concepts with the introduction of an example.

### B. The shepherd problem

Consider a target tracking problem in which an agent – the shepherd – follows a group of  $m$  targets – the sheep. Specifically, let  $z(t) = [z_1(t), z_2(t)]^T \in \mathbb{R}^2$  denote the position of the shepherd at time  $t$ . To model smooth paths for the shepherd introduce a polynomial parameterization so that each of the position components  $z_k(t)$  can be written as

$$z_k(t) = \sum_{j=0}^{n-1} x_{kj} p_j(t), \quad (5)$$

where  $p_j(t)$  are polynomials that parameterize the space of possible trajectories. The action space of the shepherd is then given by the vector  $x = [x_{10}, \dots, x_{1,n-1}, x_{20}, \dots, x_{2,n-1}]^T \in \mathbb{R}^{2n}$  that stacks the coefficients of the parameterization in (5).

Further define  $y_i(t) = [y_{i1}(t), y_{i2}(t)]^T$  as the position of the  $i$ th sheep at time  $t$  for  $i = 1, \dots, m$  and introduce a maximum allowable distance  $r_i$  between the shepherd and each of the sheep. The goal of the shepherd is to find a path  $z(t)$  that is within distance  $r_i$  of sheep  $i$  for all sheep. This can be captured by defining an  $m$ -dimensional environment  $f$  with each component function  $f_i$  defined as

$$f_i(t, x) = \|z(t) - y_i(t)\|^2 - r_i^2 \quad \text{for all } i = 1..m. \quad (6)$$

That the environment defined by (6) is viable means that it is possible to select a vector of coefficients  $x$  so that the shepherd's trajectory generated by (5) stays close to all sheep for all times. To the extent that (5) is a loose parameterization – we can approximate arbitrary functions with sufficiently large index  $n$  –, this simply means that the sheep are sufficiently close to each other at all times. E.g., if  $r_i = r$  for all times, viability is equivalent to having a maximum separation between sheep smaller than  $2r$ .

Trajectories  $x(t)$  differ from actions in that they are allowed to change over time, i.e., the constant values  $x_{kj}$  in (5) are replaced by the time varying values  $x_{kj}(t)$ . A feasible trajectory  $x(t)$  means that the shepherd is repositioning to stay close to all sheep. In this case we apply the usual caveat that small fit may be achieved with stretches of underachievement following stretches of overachievement.

### III. SADDLE POINT CONTROLLER

Given an environment  $f(t, x)$  verifying assumptions 1 and 2 we set our attention towards the problem of designing a controller that gives origin to feasible trajectories. As already noted, when the environment is known beforehand the problem of finding such trajectories is a constrained convex optimization problem, which we can solve using the saddle point algorithm of Arrow and Hurwicz [1]. Following this idea, let  $\lambda \in \Lambda = \mathbb{R}_+^m$ , be a multiplier and define the time-varying Lagrangian associated with the online problem as

$$\mathcal{L}(t, x, \lambda) = \lambda^T f(t, x). \quad (7)$$

Saddle point methods rely on the fact that for a constrained convex optimization problem, a pair is a primal-dual optimal

solution if and only if the pair is a saddle point of the Lagrangian associated with the problem; see e.g. [13]. Since optimality is not of interest for us, any point in the feasible set is an optimal solution of the convex optimization problem. Hence, finding a saddle point is the equivalent of finding a feasible point. The main idea of the algorithm is then to generate trajectories that descend in the opposite direction of the gradient of the Lagrangian with respect to  $x$  and that ascend in the direction of the gradient with respect to  $\lambda$ . To avoid restricting attention to functions that are differentiable with respect to  $x$ , we introduce the notion of subgradient that we formally define next.

**Definition 3 (Subgradient).** *Let  $g : X \rightarrow \mathbb{R}$ , be a convex function where  $X \subset \mathbb{R}^n$ . Then  $g_x$  is a subgradient of  $g$  at a point  $x \in X$  if:*

$$g(y) \geq g(x) + g_x(x)^T(y - x) \quad \text{for all } y \in X \quad (8)$$

Subgradients are defined at all points for convex functions. At the points where the function  $f$  is differentiable the subgradient and the gradient coincide. For vector functions  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  we group the subgradients of each component into a subgradient matrix  $f_x(x) \in \mathbb{R}^{n \times m}$  that we define as

$$f_x(x) = [ f_{1,x}(x) \quad f_{2,x}(x) \quad \cdots \quad f_{m,x}(x) ] \quad (9)$$

where  $f_{i,x}(x)$  is a subgradient of  $f_i(x)$  as per Definition 3. Since the Lagrangian is differentiable with respect to  $\lambda$ , we denote by  $\mathcal{L}_\lambda(t, x, \lambda) = f(t, x)$  the derivative of the Lagrangian with respect to  $\lambda$ . On the other hand, since the function  $f(\cdot, x)$  is convex the subgradient with respect to  $x$  always exist, let then denote by  $\mathcal{L}_x(t, x, \lambda)$  the subgradient of the Lagrangian with respect to  $x$ . Since the action must always be selected from the set  $X$  and the multipliers must be in the non negative orthant, we need to project the solutions over those sets. With this goal in mind, we define a projected dynamical system over a closed set; see [14], [15].

**Definition 4 (Projected dynamical system).** *Let  $X \in \mathbb{R}^n$  be a closed convex set:*

**Projection of a point.** *For any  $z \in \mathbb{R}^n$ , there exists a unique element in  $X$ , denoted  $P_X(z)$  such that:*

$$P_X(z) = \arg \inf_{y \in X} \|y - z\|. \quad (10)$$

**Projection of a vector at a point.** *Let  $x \in X$  and  $v \in \mathbb{R}^n$ , we define the projection of  $v$  over the set  $X$  at the point  $x$ ,  $\Pi_X(x, v)$  as:*

$$\Pi_X(x, v) = \lim_{\delta \rightarrow 0^+} (P_X(x + \delta v) - x)/\delta. \quad (11)$$

*The projection of a vector at a point over a set is equivalent to project the vector over the smallest cone containing the set  $X$  with vertex at the point  $x$ .*

**Projected dynamical system.** *Given a closed convex set  $X$  and a vector field  $F(t, x)$  which takes elements from  $X$  into  $\mathbb{R}^n$  the projected differential equation associated with  $X$  and  $F$  is defined to be:*

$$\dot{x}(t) = \Pi_X(x, F(t, x)). \quad (12)$$

If the point  $x$  is in the interior of  $X$  then the projection is not different from the original vector field i.e.  $\Pi_X(x, F(t, x)) = F(t, x)$ . If the point  $x$  is in the border of  $X$  and the vector field is pointing outside the set, then the projection is just the component of the vector field that is tangential to the set  $X$  at the point  $x$ . Let's consider for instance the case where the set  $X$  is a box in  $\mathbb{R}^n$ . Let  $X = [a_1, b_1] \times \dots \times [a_n, b_n]$  where  $a_1 \dots a_n$  and  $b_1 \dots b_n$  are real numbers. Then for each component of the vector field we have:

$$\Pi_X(x, F(t, x))_i = \begin{cases} 0 & \text{if } x_i = a_i \text{ and } F(t, x)_i < 0, \\ 0 & \text{if } x_i = b_i \text{ and } F(t, x)_i > 0, \\ F(t, x)_i & \text{otherwise.} \end{cases} \quad (13)$$

Using the notions of subgradient and projected dynamical system we can present the online saddle point controller. For that matter introduce the gain  $\varepsilon$  and define a controller that descends in the direction of the Lagrangian subgradient with respect to the action  $x$ ,

$$\dot{x} = \Pi_X(x, -\varepsilon \mathcal{L}_x(x_t, \lambda_t)) = \Pi_X(x, -\varepsilon f_x(t, x)\lambda), \quad (14)$$

and that ascends in the direction of the Lagrangian gradient with respect to the multiplier  $\lambda$

$$\dot{\lambda} = \Pi_\Lambda(\varepsilon \mathcal{L}_\lambda(t, x, \lambda)) = \Pi_\Lambda(\lambda, \varepsilon f(t, x)). \quad (15)$$

An important observation regarding (14) - (15) is that the environment is observed locally in space and causally in time. The values of the environment constraints and its subgradients are observed at the current trajectory position  $x(t)$  and the values of  $f(t, x(t))$  and  $f_x(t, x(t))$  affect the derivatives of  $x(t)$  and  $\lambda(t)$  only.

A block diagram for the controller in (14) - (15) is shown in Figure 1. The controller operates in an environment to which it inputs at time  $t$  an action  $x(t)$  that results in a penalty  $f(t, x(t))$ . The value of this penalty and its subgradient  $f_x(t, x(t))$  are observed and fed to the multiplier and action feedback loops, respectively. The action feedback loop behaves like a weighted gradient descent controller. We move in the direction given by a linear combination of the different constraint subgradients  $f_i(t, x(t))$  weighted by their corresponding multipliers  $\lambda_i(t)$ . Intuitively, this pushes  $x(t)$  towards satisfying the constraints. However, the question remains of how much weight to give to each constraint. This is the task of the multiplier feedback loop. When constraint  $i$  is violated we have  $f_i(t, x(t)) > 0$ . This pushes the multiplier  $\lambda_i(t)$  up, thereby increasing the force  $\lambda_i(t)f_i(t, x(t))$  pushing  $x(t)$  towards satisfying the constraint. If the constraint is satisfied, we have  $f_i(t, x(t)) < 0$ , the multiplier  $\lambda_i(t)$  being decreased, and the corresponding force decreasing. The more that constraint  $i$  is violated, the faster we increase the multiplier, and the more we increase the force that pushes  $x(t)$  towards satisfying  $f_i(t, x(t)) < 0$ . If the constraint is satisfied, the force is decreased and may eventually vanish altogether if we reach the point of making  $\lambda_i(t) = 0$ .

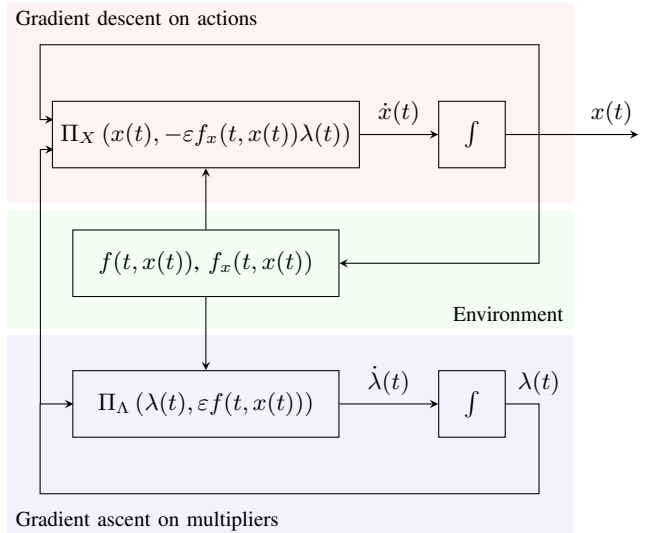


Fig. 1: Block diagram of the saddle point controller. The main structure is composed by two interconnected feedback loops. Once, that action  $x$  is selected at time  $t$ , we measure the corresponding values of  $f(t, x)$  and  $f_x(t, x)$ .

#### A. Strongly feasible trajectories

This section presents a bound on the fit of the trajectories  $x(t)$  generated by the saddle point controller defined by (14) and (15). This bound ensures that the trajectories  $x(t)$  are strongly feasible in the sense of Definition 2. To state the result consider an arbitrary fixed action  $\bar{x} \in X$  and an arbitrary multiplier  $\bar{\lambda} \in \Lambda$  and define the energy function

$$V_{\bar{x}, \bar{\lambda}}(x, \lambda) = \frac{1}{2} (\|x - \bar{x}\|^2 + \|\lambda - \bar{\lambda}\|^2). \quad (16)$$

We can then bound fit in terms of the initial value  $V_{\bar{x}, \bar{\lambda}}(x(0), \lambda(0))$  of the energy function for properly chosen  $\bar{x}$  and  $\bar{\lambda}$  as we formally state next.

**Theorem 1.** *Let  $f : \mathbb{R} \times X \rightarrow \mathbb{R}^m$ , satisfying assumptions 1 and 2, where  $X \subseteq \mathbb{R}^n$  is a closed convex set. If the environment is viable, then the controller defined by (14) and (15) gives origin to feasible trajectories  $x(t)$  for all  $T > 0$ . Furthermore the fit is bounded by:*

$$\mathcal{F}_{T, i} \leq (V_{x^\dagger, e_i}(x(0), \lambda(0))) / \varepsilon, \quad (17)$$

where  $x^\dagger$  belongs to the feasible set  $X^\dagger$ , and  $e_i$  with  $i = 1 \dots m$  are the vectors of the canonical base of  $\mathbb{R}^m$ .

*Proof.* See Appendix.  $\square$

Theorem 1 assures that if an environment is viable for an agent that selects actions over a set  $X$ , the controller defined by (14) and (15) gives origin to a trajectory  $x(t)$  that is strongly feasible in the sense of Definition 2. This result is not trivial, since the function  $f$  that defines the environment is observed causally and can change arbitrarily over time. In particular, the agent could be faced with an adversarial environment that changes the function  $f$  in a way that makes the value of  $f(t, x(t))$  larger. The caveat is that the choice of the function  $f$  must respect the viability condition that

there exists a feasible action  $x^\dagger$  such that  $f(t, x^\dagger) \leq 0$  for all  $t \in [0, T]$ . This restriction still leaves significant leeway for strategic behavior. E.g., in the shepherd problem of Section II-B we can allow for strategic sheep that observe the shepherd's movement and respond separating as much as possible. The strategic action of the sheep are restricted by the condition that the environment remains viable, which in this case reduces to the not so stringent condition that the sheep stay in a ball of radius  $2r$  if all  $r_i = r$ .

Since the initial value of the energy function  $V_{x^\dagger, e_i}(x(0), \lambda(0))$  is the square of the distance between  $x(0)$  and  $x^\dagger$  plus a term that depends on the distance of the initial value of the multiplier with the multiplier  $e_i$ , the fit bound in (14) shows that the closer we start to the feasible set the smaller the accumulated constraint violation becomes. Likewise, the larger the gain  $\epsilon$ , the smaller the fit bound is. Theoretically, increasing  $\epsilon$  can make the fit bound arbitrarily small. This is not possible in practice because larger  $\epsilon$  entails trajectories with larger derivatives which cannot be implemented in systems with physical constraints. In the example of Section II-B the derivatives of the state  $x(t)$  control speed and acceleration. The physical limits of these quantities along with an upper bound on the gradient  $f_x(t, x)$  and an upper bound on the multiplier values of  $\lambda(t)$  can be used to estimate the largest allowable gain  $\epsilon$ . An upper bound in  $\lambda(t)$  has not been provided but can be obtained as a direct consequence of Theorem 1.

**Corollary 1.** *Under the hypothesis of Theorem 1, from the controller (14), (15) arises multipliers  $\lambda$  that are bounded for all time. In particular, for each  $i = 1..m$ , it holds*

$$0 \leq \lambda_i(t) \leq \lambda_i(0) + (V_{x^\dagger, e_i}(x(0), \lambda(0))) / \epsilon. \quad (18)$$

*Proof.* See Appendix.  $\square$

The bound in Corollary 1 ensures that action derivatives  $\dot{x}(t)$  remain bounded if the subgradients are. Further observe that increasing the gain  $\epsilon$  decreases the multiplier bound in (18). This means that action derivatives increase, at most, linearly with  $\epsilon$  and is not compounded by an increase in the values of the multipliers.

#### IV. NUMERICAL EXPERIMENTS

We evaluate performance of the saddle point algorithm defined by (14)-(15) in the solution of the shepherd problem introduced in Section II-B. We determine sheep paths using a perturbed polynomial characterization akin to the one in (5). Specifically, letting  $p_j(t)$  be elements of a polynomial basis, the path  $y_i(t) = [y_{i1}(t), y_{ik}(t)]^T$  followed by the  $i$ th sheep is given by the expression

$$y_{ik}(t) = \sum_{j=0}^{n_i-1} y_{ikj} p_j(t) + w_{ik}(t), \quad (19)$$

where  $k = 1, 2$  denotes different path components,  $n_i$  the total number of polynomials that parameterize the path followed by sheep  $i$ , and  $y_{ikj}$  represent the corresponding  $n_i$  coefficients. The noise terms  $w_{ik}(t)$  are Gaussian white with

zero mean, standard deviation  $\sigma$ , and chosen independently across components and sheep. Their purpose is to obtain more erratic paths.

To determine  $y_{ikj}$  we make  $w_{ik}(t) = 0$  in (19) and require all sheep to start at position  $y_i(0) = [0, 0]^T$  and finish at position  $y_i(T) = [1, 1]^T$ . A total of  $L$  random points  $\{\tilde{y}_l\}_{l=1}^L$  are then drawn independently and uniformly at random in the unit box  $[0, 1]^2$ . Sheep  $i = 1$  is required to pass through points  $\tilde{y}_l$  at times  $lT/(L+1)$ , i.e.,  $y_1(lT/(L+1)) = \tilde{y}_l$ . For each of the other sheep  $i \neq 1$  we draw  $L$  random offsets  $\{\Delta\tilde{y}_{il}\}_{l=1}^L$  uniformly at random from the box  $[-\Delta, \Delta]^2$  and require the  $i$ th sheep path to satisfy  $y_i(lT/(L+1)) = \tilde{y}_l + \Delta\tilde{y}_{il}$ . Paths  $y_i(t)$  are then chosen as those that minimize the path integral of the acceleration squared subject to the constraints of each individual path, i.e.,

$$\begin{aligned} y_i^*(t) = \operatorname{argmin} & \int_0^T \|\ddot{y}_i(t)\|^2 dt, \\ \text{s.t.} & y_i(0) = [0, 0]^T, \quad y_i(T) = [1, 1]^T, \\ & y_i(lT/(L+1)) = \tilde{y}_l + \Delta\tilde{y}_{il}, \end{aligned} \quad (20)$$

where, by construction  $\Delta\tilde{y}_{il} = 0$  for  $i = 1$ . The paths in (20) can be computed as solutions of a quadratic program [16]. We obtain the paths  $y_{ik}(t)$  by adding  $w_{ik}(t)$  to  $y_{ik}^*(t)$ .

In subsequent numerical experiments we consider  $m = 5$  sheep, a time horizon  $T = 1$ , and set the proximity constraint in (6) to  $r_i = 0.3$ . We use the standard polynomial basis  $p_j(t) = t^j$  in both, (5) and (19). The number of basis elements in both cases is set to  $n = n_i = 30$ . To generate sheep paths we consider a total of  $L = 3$  randomly chosen intermediate points, set the variation parameter to  $\Delta = 0.1$ , and the perturbation standard deviation to  $\sigma = 0.1$ . These problem parameters are such that the environment is most likely viable in the sense of Definition 1. We check that this is true by solving the offline feasibility problem. If the environment is not viable a new one is drawn before proceeding to the implementation of (14)-(15).

We emphasize that even if the complete trajectory of the sheep is known to us, the information is not used by the controller. The controller is only fed information of the position of the sheep at the current time, which it uses to evaluate the environment functions  $f_i(t, x)$  in (6) and their gradients  $f_{ix}(t, x)$ .

The system's behavior is illustrated in Figure 2 when the gain is set to  $\epsilon = 50$ . A qualitative examination of the sheep and shepherd paths shows that the shepherd succeeds in following the herd. A more quantitative evaluation is presented in Figure 3 where we plot the instantaneous constraint violation  $f_i(t, x(t))$  with respect to each sheep for the trajectories  $x(t)$  obtained from (14)-(15). Observe the oscillatory behavior that has the constraint violations  $f_i(t, x(t))$  hovering at around  $f_i(t, x(t)) = 0$ . When the constraints are violated, i.e., when  $f_i(t, x(t)) > 0$ , the saddle point controller drives the shepherd towards a position that makes him stay within  $r_i$  of all sheep. When a constraint is satisfied we have  $f_i(t, x(t)) < 0$ . This drives the multiplier  $\lambda_i(t)$  towards 0 and removes the force that pushes the

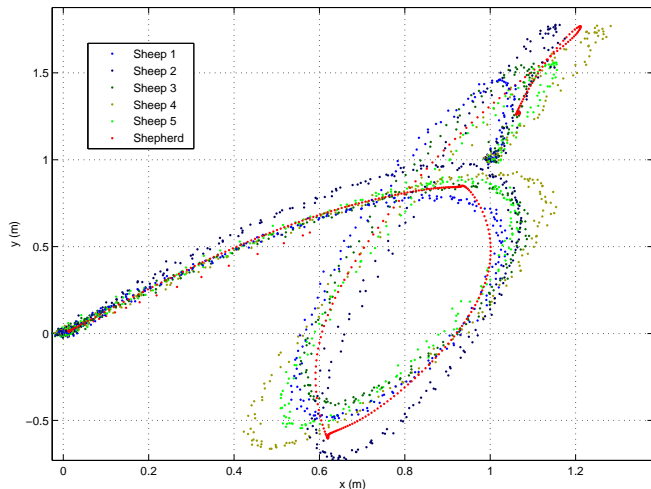


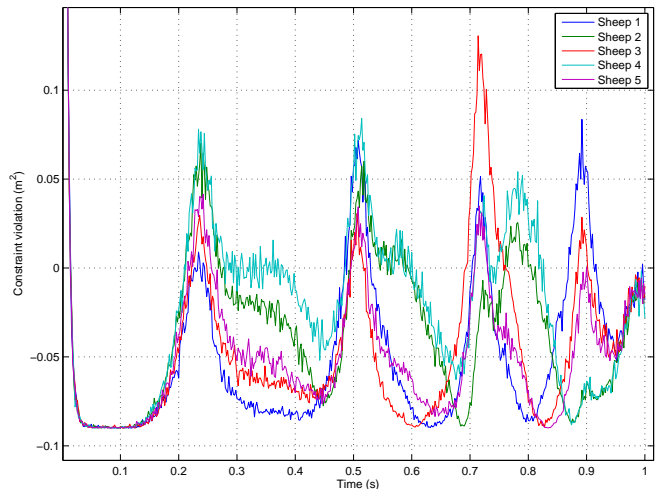
Fig. 2: Path of the sheep and the shepherd. The value of the gain of the saddle point controller is set to be  $\varepsilon = 50$ . We observe how the shepherd –in red– path is close to the path of all sheep. He succeeds to follow the herd.

shepherd towards the sheep (c.f. Figure 3). The absence of this force makes the constraint violation grow and eventually surpass the maximum tolerance  $f_i(t, x(t)) = 0$ . At this point the multipliers start to grow and, as a consequence, to push the shepherd back towards proximity with the sheep.

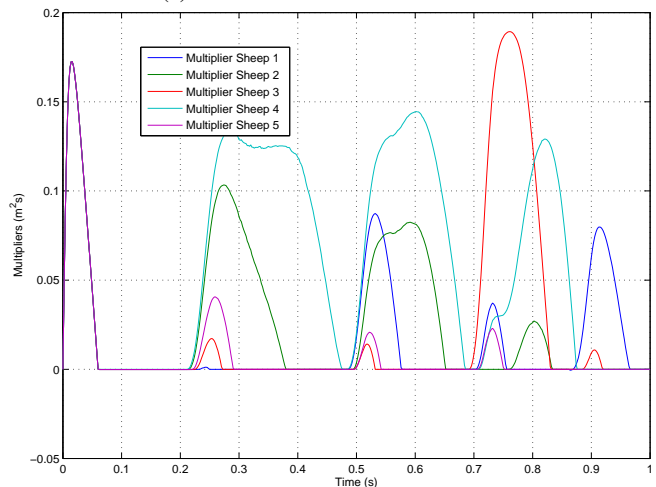
The behavior observed in Figure 3 does not contradict the result in Theorem 1 which gives us a guarantee on fit, not on instantaneous constraint violations. The components of the fit are shown in Figure 4a and they are indeed bounded. Thus, the trajectory is strongly feasible in the sense of Definition 2, even if the constraints are being violated at specific time instances. Further note that the regret is not only bounded but actually becomes negative. This is a consequence of the relatively large gain  $\varepsilon = 50$  which helps the shepherd to respond quickly to the sheep movements. The fit for a second experiment in which the gain is reduced to  $\varepsilon = 5$  is shown in Figure 4b. In this case the fit stabilizes at a positive value. This behavior is expected because reducing  $\varepsilon$  decreases the speed with which the shepherd can adapt to changes in the sheep paths. More to the point, the fit bound in Theorem 1 is inversely proportional to the gain  $\varepsilon$ . The paths and instantaneous constraints violations for  $\varepsilon = 5$  are qualitatively similar to the ones shown for  $\varepsilon = 50$  in figures 2 and 3.

## V. CONCLUSION

Throughout this work we have considered a continuous time environment in which an agent must select actions in order to satisfy the constraints imposed by the environment. A saddle point controller was designed and we showed that for a viable environment the trajectories that arise from it are strongly feasible. Furthermore, we showed that the controller gives origin to bounded action derivatives. In the last section we show how saddle points algorithms can be used in a tracking problem, in which we want to follow several targets. In this case the environment is given by the distance between the



(a) Instantaneous constraint value.



(b) Temporal evolution of the multipliers.

Fig. 3: Relation between instantaneous constraint value and multipliers  $\lambda$ . At the times in which the value of a constraint is positive, the corresponding multiplier increases. The latter entails a decrease of the value of the constraint function. Once the constraint function is negative the corresponding multiplier decreases.

agent and the targets to track. The results obtained are in concordance with what was proved in Theorem 1.

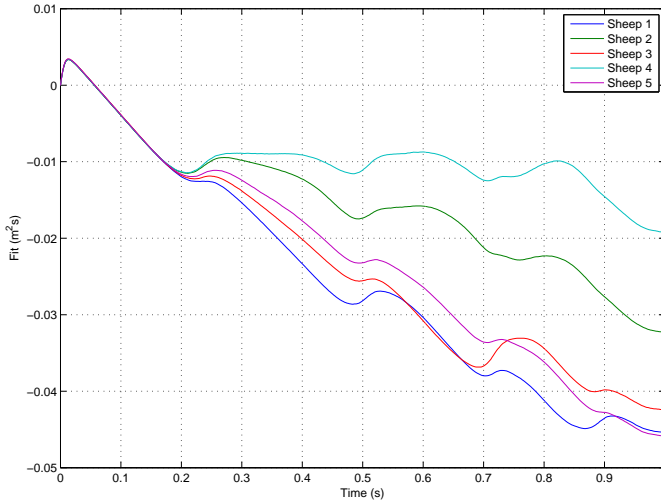
## APPENDIX

In order to develop proofs we need to define the concept of tangent cone and to state Lemmas 1 and 2.

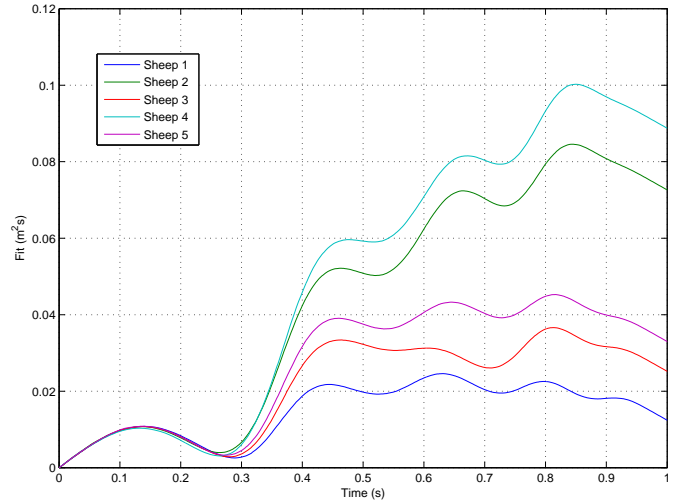
**Definition 5 (Tangent cone).** Let  $X \subset \mathbb{R}^n$  be a closed convex set. We define the tangent cone to  $X$  at  $x_0$  as:

$$T_X(x_0) = \overline{\cup_{\theta > 0, x \in X} (x - x_0)\theta} \quad (21)$$

The above union is over all the points of the set  $X$  and over all the positive reals  $\theta$ . Notice that the  $\cup_{\theta > 0} (x - x_0)\theta$  is the ray from  $x_0$  and intersecting the point  $x$ . Thus, the tangent cone is then the closure of the cone formed by all rays emanating from  $x_0$  and intersecting at least one point  $x \in X$  different from  $x_0$ .



(a) The gain in the saddle point controller is set to be  $\varepsilon = 50$ .



(b) The gain in the saddle point controller is set to be  $\varepsilon = 5$ .

Fig. 4: Fit  $\mathcal{F}_T$  for two different controller gains. Notice that both results are in concordance with Theorem 1. First of all the fit is bounded in the two experiments. Moreover, the larger the value of the gain  $\varepsilon$  the smaller the fit of the trajectory generated.

**Lemma 1.** For arbitrary  $\delta$  and  $v$  the projection over the set  $X$  can be written as:

$$P_X(x_0 + \delta v) = x_0 + \delta P_{T_X(x_0)}(v) + \mathcal{O}(\delta) \quad (22)$$

where  $\mathcal{O}(\delta)$  is a function such that  $\lim_{\delta \rightarrow 0} \mathcal{O}(\delta)/\delta = 0$

*Proof.* See [15] Lemma 4.6 page 300.  $\square$

**Corollary 2.** Let  $X \in \mathbb{R}^n$  be a closed convex set, let  $x_0 \in X$  and let  $v \in \mathbb{R}^n$ . Then the projection of  $v$  over the set  $X$  at  $x_0$  defined in (10) is:

$$\Pi_X(x_0, v) = P_{T_X(x_0)}(v) \quad (23)$$

*Proof.* The proof is trivial from lemma 1  $\square$

**Lemma 2.** Let  $X$  be a closed convex set and  $x_0 \in X$  and  $x \in X$ . Then:

$$(x_0 - x)^T \Pi_X(x_0, v) \leq (x_0 - x)^T v. \quad (24)$$

*Proof of Lemma 2.* Let's first consider the case in which  $x_0 \in \text{int}(X)$ . Then, for any  $v$  there exists a small enough  $\delta > 0$  such that  $x_0 + \delta v \in X$ . Hence  $P_X(x_0 + \delta v) = x_0 + \delta v$ , and then we have that  $P_X(x_0 + \delta v) - x_0 = \delta v$ . Thus  $\Pi_X(x_0, v) = v$  and then we have trivially that:

$$(x_0 - x)^T \Pi_X(x_0, v) = (x_0 - x)^T v \quad (25)$$

Let's now consider the case in which  $x_0$  is in the border of  $X$ , here two case are possible: either  $x_0 + \delta v \in T_X(x_0)$  for small enough  $\delta > 0$  or  $x_0 + \delta v \notin T_X(x_0)$  for all  $\delta > 0$ . Because of this distinction is that the result of Corollary 2 is important. In the first case we trivially have that:

$$\Pi_X(x_0, v) = P_{T_X(x_0)}(v) = v \quad (26)$$

And therefore (25) holds in this case as well. Finally, we are going to consider the case in which  $x_0 \in \partial X$  and  $x_0 + \delta v \notin T_X(x_0)$ . Because  $X$  is a convex set there exists a vector

$a \in \mathbb{R}^n$  with  $\|a\| = 1$  defining the supporting hyperplane  $\mathcal{H} = \{x \in \mathbb{R}^n : a^T(x - x_0) = 0\}$  at  $x_0$ . Since it is a supporting hyperplane, for all  $x \in X$  we have that:

$$a^T(x - x_0) \leq 0 \quad (27)$$

If the set is smooth at  $x_0$  then the border of the tangent cone at the point  $x_0$  is contained in the hyperplane  $\mathcal{H}$ , therefore  $\Pi_X(x_0, v) \subset \mathcal{H}$ . Thus,  $a^T \Pi_X(x_0, v) = 0$ . In this case we have as well that  $a^T v \geq 0$ , otherwise there must exist a  $\delta > 0$  such that  $x_0 + \delta v \in T_X(x_0)$ . On the other hand if there is a corner at  $x_0$  there are infinite supporting hyperplanes. One of them verifies that  $a^T v \geq 0$  and contains the border of the tangent cone, thus  $a^T \Pi_X(x_0, v) = 0$ . Finally, since  $\Pi_X(x_0, v)$  is the projection of  $v$  over the tangent cone, we have that:  $\Pi_X(x_0, v) = P_{T_X(x_0)}(v) = (a_\perp^T v) a_\perp$ , where  $a_\perp \in \mathbb{R}^n$  and verifies that  $a^T a_\perp = 0$  and  $\|a_\perp\| = 1$ . Projecting the vectors  $x_0 - x$  and  $v$  over  $a$  and  $a_\perp$ , we have:

$$(x_0 - x)^T v = (x_0 - x)^T a v^T a + (x_0 - x)^T a_\perp v^T a_\perp \quad (28)$$

Because of the above previous discussion (28) reduces to:

$$(x_0 - x)^T v = (x_0 - x)^T a v^T a + (x_0 - x)^T \Pi_X(x_0, v) \quad (29)$$

Finally using the fact that  $a$  is the vector director of a supporting hyperplane (27) and using the fact that  $v^T a \geq 0$  the following inequality holds:

$$(x_0 - x)^T v \geq (x_0 - x)^T \Pi_X(x_0, v) \quad (30)$$

Hence we have proved the lemma for all possible cases.  $\square$

**Proof of Theorem 1.** Consider action trajectories  $x(t)$  and multiplier trajectories  $\lambda(t)$  and the corresponding energy function  $V_{\bar{x}, \bar{\lambda}}(x(t), \lambda(t))$  in (16) for arbitrary given action  $\bar{x} \in X$  and multiplier  $\bar{\lambda} \in \Lambda$ . The derivative  $\dot{V}_{\bar{x}, \bar{\lambda}}(x(t), \lambda(t))$  of the energy with respect to time is then given by

$$\dot{V}_{\bar{x}, \bar{\lambda}}(x(t), \lambda(t)) = (x(t) - \bar{x})^T \dot{x}(t) + (\lambda(t) - \bar{\lambda})^T \dot{\lambda}(t). \quad (31)$$

If the trajectories  $x(t)$  and  $\lambda(t)$  follow from the saddle point dynamical system given by (14) and (15) we can substitute the action and multiplier derivatives by their corresponding values and reduce(31) to

$$\begin{aligned} \dot{V}_{\bar{x},\bar{\lambda}}(x(t), \lambda(t)) = & (x(t) - \bar{x})^T \Pi_X(x, -\varepsilon f_x(t, x(t))\lambda(t)) \\ & + (\lambda(t) - \bar{\lambda})^T \Pi_\Lambda(x, \varepsilon f(t, x(t))). \end{aligned} \quad (32)$$

Then, using the result of Lemma 2 for both  $X$  and  $\Lambda$ , the following inequality holds:

$$\begin{aligned} \dot{V}_{\bar{x},\bar{\lambda}}(x(t), \lambda(t)) \leq & \varepsilon(\bar{x} - x(t))^T f_x(t, x(t))\lambda(t) \\ & + \varepsilon(\lambda(t) - \bar{\lambda})^T f(t, x(t)). \end{aligned} \quad (33)$$

Notice that  $f(t, x)^T \lambda(t)$  is a convex function with respect to the action, therefore we can upper bound the inner product  $(\bar{x} - x(t))^T f_x(t, x(t))\lambda(t)$  by the quantity  $f(t, \bar{x})^T \lambda(t) - f(t, x(t))^T \lambda(t)$  and transform (33) into:

$$\begin{aligned} \dot{V}_{\bar{x},\bar{\lambda}}(x(t), \lambda(t)) \leq & \varepsilon(f(t, \bar{x}) - f(t, x(t)))^T \lambda(t) \\ & + \varepsilon(\lambda(t) - \bar{\lambda})^T f(t, x(t)). \end{aligned} \quad (34)$$

Notice that in the above equation the second and the third term are opposite. Thus, the above equation reduces to:

$$\dot{V}_{\bar{x},\bar{\lambda}}(x(t), \lambda(t)) \leq \varepsilon[\lambda^T(t)f(t, \underline{x}) - \bar{\lambda}^T f(t, x(t))]. \quad (35)$$

Rewriting the above expression and then integrating both sides with respect to the time from  $t = 0$  to  $t = T$  yields:

$$\varepsilon \int_0^T \bar{\lambda}^T f(t, x(t)) - \lambda^T(t) f(t, \bar{x}) dt \leq - \int_0^T \dot{V}_{\bar{x},\bar{\lambda}}(x(t), \lambda(t)) dt. \quad (36)$$

Integrating the right side of the above equation we obtain:

$$- \int_0^T \dot{V}_{\bar{x},\bar{\lambda}}(x(t), \lambda(t)) dt = V_{\bar{x},\bar{\lambda}}(x(0), \lambda(0)) - V_{\bar{x},\bar{\lambda}}(x(T), \lambda(T)), \quad (37)$$

and then using the fact that  $V_{\bar{x},\bar{\lambda}}(x(t), \lambda(t)) \geq 0$  for all  $t$  we have that:

$$- \int_0^T \dot{V}_{\bar{x},\bar{\lambda}}(x(t), \lambda(t)) dt \leq V_{\bar{x},\bar{\lambda}}(x(0), \lambda(0)). \quad (38)$$

Then, combining (36) and (38), we have that

$$\int_0^T \bar{\lambda}^T f(t, x(t)) - \lambda^T(t) f(t, \bar{x}) dt \leq (V_{x^\dagger, \bar{\lambda}}(x(0), \lambda(0))) / \varepsilon. \quad (39)$$

Since the environment is viable there exist a fixed action  $x^\dagger$  such that  $f(t, x^\dagger) \leq 0$  for all  $t \geq 0$ , then choosing  $\bar{x} = x^\dagger$ ,  $\lambda^T(t) f(t, x^\dagger) dt \leq 0 \forall t \in [0, T]$ , therefore the left hand side of (39) can be lower bounded, obtaining:

$$\bar{\lambda}^T \int_0^T f(t, x(t)) dt \leq (V_{x^\dagger, \bar{\lambda}}(x(0), \lambda(0))) / \varepsilon. \quad (40)$$

Finally, choosing  $\bar{\lambda} = e_i$  where  $e_i$  is the  $i$ -th element of the canonical base of  $\mathbb{R}^m$ , we have that for all  $i = 1..m$ :

$$\int_0^T f_i(t, x(t)) dt \leq (V_{x^\dagger, e_i}(x(0), \lambda(0))) / \varepsilon. \quad (41)$$

The left hand side of the above inequality is the  $i$ -th component of the fit. Since the  $m$  components of the fit

obtained by the trajectory generated by the saddle point algorithm are bounded for all  $T$ , the trajectory is feasible. Furthermore, we proved the upper bound of (17).  $\square$

**Proof of Corollary 1.** Since the trajectory of the multipliers is defined by  $\lambda(t) = \Pi_\Lambda(\lambda(t), f(t, x(t)))$ , it is clear that for all time  $t \in [0, T]$ , we have that  $\lambda(t) \geq 0$ . On the other hand, for all  $t \in [0, T]$ , we have that:  $\Pi_\Lambda(\lambda(t), f(t, x(t))) \geq f(t, x(t))$ . The latter is true because for any  $f(t, x(t))$  the projection is the same as the original vector unless  $\lambda_i(t) = 0$  and  $f_i(t, x(t)) < 0$  and in that case the projection is zero. Therefore we have that:

$$\int_0^\tau \dot{\lambda}(t) dt = \int_0^\tau \Pi_\Lambda(\lambda(t), f(t, x(t))) dt \leq \int_0^\tau f(t, x(t)) dt. \quad (42)$$

Notice that the right hand side of the above equation is the definition of the fit until time  $\tau$ . Hence we have that:

$$\lambda(\tau) - \lambda(0) \leq \mathcal{F}_\tau \quad (43)$$

Since we proved that the fit is bounded for all  $T$  it is also bounded for  $\tau$ . Furthermore, the bound for the  $i$ -th component is given by (17). Therefore  $\lambda(\tau)$  is bounded for all  $\tau \in [0, T]$  by the expression given in 18.  $\square$

## REFERENCES

- [1] K. J. Arrow and L. Hurwicz, *Studies in linear and nonlinear programming*. CA: Stanford University Press, 1958.
- [2] M. W. Hirsch, S. Smale, and R. L. Devaney, *Differential equations, dynamical systems, and an introduction to chaos*, vol. 60. Academic press, 2004.
- [3] S.-i. Azuma, M. S. Sakar, and G. J. Pappas, "Nonholonomic source seeking in switching random fields," in *Decision and Control (CDC), 2010 49th IEEE Conference on*, pp. 6337–6342, IEEE, 2010.
- [4] S.-i. Azuma, M. S. Sakar, and G. J. Pappas, "Stochastic source seeking by mobile robots," *Automatic Control, IEEE Transactions on*, vol. 57, no. 9, pp. 2308–2321, 2012.
- [5] N. Atanasov, J. Le Ny, N. Michael, and G. J. Pappas, "Stochastic source seeking in complex environments," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 3013–3018, IEEE, 2012.
- [6] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 5, pp. 501–518, 1992.
- [7] C. W. Warren, "Global path planning using artificial potential fields," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pp. 316–321, IEEE, 1989.
- [8] A. Chambolle and T. Pock, "A first-order primal-dual algorithm for convex problems with applications to imaging," *Journal of Mathematical Imaging and Vision*, vol. 40, no. 1, pp. 120–145, 2011.
- [9] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and Trends in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2011.
- [10] V. Vapnik, *The nature of statistical learning theory*. Springer, 2000.
- [11] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *ICML*, pp. 928–936, 2003.
- [12] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Machine Learning*, vol. 69, no. 2-3, pp. 169–192, 2007.
- [13] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1999.
- [14] M.-G. Cojocaru and L. Jonker, "Existence of solutions to projected differential equations in hilbert spaces," *Proceedings of the American Mathematical Society*, vol. 132, no. 1, pp. 183–193, 2004.
- [15] D. Zhang and A. Nagurny, "On the stability of projected dynamical systems," *J. Optim. Theory Appl.*, vol. 85, pp. 97–124, Apr. 1995.
- [16] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.