Numerical Studies of Hybrid Adaptive Control for a class of Autonomous Robots and Vehicles

by

Arturo Freydig Avila

B.S., University of Colorado Boulder, 2020

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Electrical, Computer and Energy Engineering

2020

Committee Members:

Prof. Jorge I. Poveda

Prof. Morteza Lahijanian

Prof. Xudong Chen

Freydig Avila, Arturo (M.S., Electrical Engineering)

Numerical Studies of Hybrid Adaptive Control for a class of Autonomous Robots and Vehicles

Thesis directed by Prof. Jorge I. Poveda

The work presented in this thesis studies the numerical performance of a class of new hybrid adaptive algorithms applied to autonomous navigation. Hybrid adaptive control is used to find the source of some signal which the vehicle can locally measure, while evading multiple obstacles at unknown positions. The source-seeking methods proposed achieve convergence to an ϵ-neighborhood of the extremum values while evading multiple obstacles at unknown positions under the presence of arbitrary small perturbations, overcoming limitations imposed by the topological obstructions induced by the obstacles.

We explore the use of both point-mass and nonholonomic unicycle dynamics to model the vehicle or robot using hybrid adaptive feedback law based on a signal (localization) function. Furthermore, we explore the use of data-enabled source seeking based on concurrent learning. We also discuss convergence guarantees for these real-time optimization methods, even under the presence of arbitrary small perturbations.

Lastly, real-time optimization numerical studies are shown to demonstrate the use of both hybrid adaptive feedback law and data-enabled extremum seeking controllers. Moreover, experimental results under the presence of adversarial signals are shown to numerically illustrate both controller's robustness.

Dedication

To my loving family.

Acknowledgements

Firstly, I would like to express my deepest gratitude to my parents for their unconditional and countless support throughout every challenge I have faced in my life and for helping me pursuit my interests. I will always be indebted with them, they are my motivation.

Additionally, I cannot express how thankful I am to have Prof. Jorge I. Poveda as my supervisor. His guidance, patience, immense knowledge, and ambition were central to the completion of this thesis. I sincerely appreciate his time and support; his dedication and commitment were truly inspirational.

I would like to extend my sincere thanks to my second thesis reader and committee member Prof. Morteza Lahijanian, I sincerely appreciate the time and effort he invested into revising this thesis. Likewise, I would like to thank my second committee member Prof. Xudong Chen his insightful comments and time. I also wish to thank Prof. Lucy Pao for encouraging to pursuit a thesis and helping me navigate graduate school. Likewise, I would like to thank my undergraduate research advisor Prof. Alan Mickelson for introducing me to academic research and always making himself available to help me.

Furthermore, I would like to thank my labmates and friends for their patience and willingness to always help me no matter how busy they were. Lastly, I would like to thank my aunt Lourdes Bernal for pushing me to attend graduate school.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

This thesis seeks to study the numerical performance of a class of new hybrid adaptive algorithms developed for the safe and robust navigation of a class of autonomous robots and vehicles. Using theoretical guarantees and numerical results we show that the new hybrid adaptive algorithms allow a vehicle to localize an unknown source under the presence of multiple obstacles. The main idea behind the hybrid feedback law is to measure a source (or localization) signal function to implement a switching rule that allows the vehicle to always implement, on average, a gradient ascend law over the environment evading multiple obstacles. The signal function is a quadratic Lyapunov function plus a barrier function, which tells the vehicle when to switch states. A quadratic form is not strictly required as shown in [6, 85], but it is used to simplify the analysis.

1.1 Motivation

In applications where point-cloud maps, GPS (global positioning system), and inertial navigation are too expensive or nonexistent, source seeking is a viable option for coordinated motion control, which is rapidly growing interest. Some interesting applications are underwater, cave, under ice, and urban navigation. In the case of sound source seeking a potential application is home automation, where a robot provided a sound impulse (e.g., a phone is ringing or human voice) needs to figure out the a trajectory to navigate from its starting location to the source of the sound [22]. Another potential application is the

effective underwater localization of a contaminant leakage, such an oil spill in the presence of turbulent flows or vibrations, which source seeking underwater drones can follow using sensors [2, 51] or when sensor measurements are intermittent and sporadic in areas with high Reynolds numbers [9, 72] turbulent medium source tracking methods can be implemented [30, 91]. While sonars can listen to contaminant leakages [7], source seeking drones can identify the exact leakage source [30]. This could help avoid enormous environmental disasters and save operators' lives, such as the 2010 Deepwater Horizon oil spill [10, 34]. Besides an oil spill, the same idea can be applied to radioactive dispersal [30], natural gas leaks [25], and methane leaks [8]. Additionally, light, heat, moisture, biological and chemical agent concentrations, magnetic fields, and electromagnetic radiation are also potential source signal examples. Companies such as Kiwibot [73] and TurtleBot [11] or projects like IceFin [82], L3Harris [13], and AURORA [20] illustrate that source seeking is currently being implemented in different applications, as in scalar field mapping [44], chemical plume tracing [47], and path planning [49].

To formalize environment-independent autonomous navigation, safety and robustness are critical. As showed in [88], source seeking oil spill behavior is affected by perturbations. Model-free adaptive control, neural networks, reinforcement learning, and model predictive control all present interesting ideas but lack, in some cases, robustness. As proposed in [60] hybrid adaptive control can achieve robust model-free autonomous navigation. Furthermore, as showed in [54] hybrid control can be used to robustly navigate while avoiding multiple objects. Considering the aforementioned facts, this thesis focuses on the numerical performance of new hybrid adaptive algorithms to achieve robust and safe environment-independent autonomous navigation in the presence of multiple unknown obstacles.

1.2 Literature Review

In recent years Real-time Optimization (RTO) has gain popularity as a control methodology given that as environmental constraints, energy costs, and complexity of system pro-

cesses increases the need for model-free optimization consequently increases. Most adaptive control schemes are developed for set-points or known reference trajectories, model-based control. Hence, when optimizing objective (signal) functions where the parameters or the desired states are unknown, RTO can be used — particularly, RTO by Extremum-Seeking Control (ESC). The goal of RTO is to find the set of controller variables which lead to near-optimal operation and the task of ESC is to maximize or minimize an objective function by real-time automated tuning of system parameters. ESC uses an external excitation signal, called *dither*, to numerically compute the gradient, which has advantages related to disturbance rejection [31].

ESC is not new, it first appeared in 1922 [45] where it was proposed as a method for maintaining maximum power transfer from a transmission line to a tram car. The proposed optimization framework proposed started getting attention in the 1940s, especially in the USSR [38, 39] where it was intensely developed. In the 1960s ESC branched into two directions, RTO and model-based adaptive control methods [52]. But it was until 1971 [50] where a formal analysis of ESC started, with the first stability analysis results based on Lyapunov theory. However, until 2000 the first general stability analysis [43] was published for stable dynamic systems with unknown output functions. The stability analysis is based on classic averaging and singular perturbation analysis to show that solutions of the closed-loop system converge to a small neighborhood of the extremum of the output function [78].

After the rigorous stability analysis was published [43], a resurgence in the field of ESC started to show promising results in smooth source seeking algorithms for autonomous navigation. Continuous-time algorithms [19, 97], discrete-time algorithms based on finite differences [42] and Newton-Raphson [12], and stochastic algorithms [69] were developed to provide local stability results in source seeking.

However, when obstacles are introduced to the environment, these create topological obstructions preclude the robust stabilization of the source of the signal by using a continuous feedback law [60, 76]. When using smooth time-invariant feedback, it is always possible to

find arbitrarily small adversarial signals e and a solution x to $\dot{x} = f_p(x + e)$ acting on the states (or vector field), such that a set of initial conditions x_0, can be rendered locally stable as noted in [69, 76]. In general, controllers based on navigation functions under the presence of obstacles, there will be at least one saddle point (critical point) in the state-space where smooth feedback can no longer guarantee extremum convergence given that adversarial signals can always render local stability see chapter 6.1.2 from [76].

In 2018, a hybrid source seeking algorithm showed that by introducing *barrier functions* to the localization or navigation function the state-space can be divided into sets which enclose the saddle points (critical points) induced by the presence of an obstacle and by using switching logic the vehicle can *jump* between sets to only *flow* in areas where the controller converges. These results opened the door to the use of Hybrid Dynamical Systems (HDS) in source seeking algorithms to achieve robust semi-global practical stabilization.

HDS is a popular framework to model systems that combine continuous-time and discrete-time dynamics. Numerous dynamical systems combine continuous-time and discrete-time behaviors. For instance, synchronizing clocks, biological systems (fireflies), systems that follow Newton's second law with instantaneous changes in velocity and momentum due to collisions, embedded systems, digital and analog components, and modern control algorithms all fit into the HDS framework class [28]. In 2012, a rigorous book on modeling, stability, and robustness of HDS [26] was published, which allowed to extend standard results in control theory to the setting of HDS. Since then, the field has been rapidly growing and finding multiple interesting applications like autonomous navigation under the presence of multiple objects [54].

Furthermore, most model-free adaptive control optimization techniques such as, ESC require a constant persistence of excitation (PE). In some cases when using sufficiently rich recorded data this PE condition can be relaxed as shown in [36]. Nonetheless, algorithms using current and past data during the seeking process to eliminate the PE condition have been explored only recently in [62, 63]. The technique called DES (Data-Enabled Extremum

Seeking) shows promising results using concurrent learning to guarantee convergence to a neighborhood of a convex optimization problem where the mathematical form of the signal function is unknown [63].

Additionally, there has been significant results in the collision avoidance navigation field implementing reinforcement learning [5], neural networks [35, 37], and model predictive control [92] (another RTO method [1]). However, these methods are not model-free methods, meaning that the environment must be known. Large deep neural networks using simultaneous localization and mapping (SLAM), high definition maps (HD maps), point-cloud mapping, or vision have become widely popular in companies aiming to achieve level four or level five [32] autonomous navigation meaning full self-driving such as Waymo (Google self-driving car), Aurora, Tesla, Lyft Level 5, and Uber ATG [14].

In summary, source seeking has been a common research area over the last two decades. The model-free algorithms have done important work using classic averaging and singular perturbation theory to guarantee stability but so far, they have not been applied nor studied in settings with multiple obstacles. Model-based controllers have achieved impressive results but require significant and constant data to learn the environment. However, the work presented in [60] and [54] shows that using a hybrid framework these topological obstructions introduced by obstacles can be evaded. Additionally [62, 63] shows that concurrent learning can be used to eliminate the PE condition in ESC.

1.3 Problem Statement

Based on the ideas presented in [54, 60, 61] could the results be extended to create a *model-free multiple obstacle avoidance source seeking algorithm*? Furthermore, could the results be replicated using *nonholonomic unicycle dynamics*? Lastly, could a *data-enabled extremum seeking method* based [62, 63], be introduced to achieve the same proposed multiple obstacle avoidance navigation?

1.4 Thesis Contributions

Primary contributions of this thesis are as follows:

(1) A numerical study that shows that results from [60] can be extended to multiple obstacles using point-mass dynamics.

(2) Replicated the point-mass Hybrid Adaptive Controller results using nonholonomic dynamics.

(3) Designed a novel multiple obstacle avoidance Hybrid Data-Enabled Extremum Seeking Controller.

1.5 Thesis Organization

The thesis is organized as follows: Chapter 2 introduces the Hybrid Dynamical System (HDS) framework and motivates its use. Chapter 3 formally presents Real-time Optimization using Averaged-Based Extremum Seeking. Additionally, it demonstrates the stability results of classic averaging and perturbation theory first introduced in [43]. Chapter 4 discusses the use of Data-Enabled Extremum Seeking Control for Real-time Optimization using concurrent learning. Chapter 5 showcases the main results of the thesis, implementing the techniques described in previous chapters by introducing the hybrid adaptive algorithms. Additionally, this chapter shows numerical studies demonstrating the thesis results. Lastly, chapter 6 concludes this research by summarizing the contributions and discussing open questions.

Chapter 2

Hybrid Dynamical Systems (HDS)

A dynamical system is normally classified as either a continuous-time or a discrete-time system. However, not every dynamical system perfectly fits such categories. There are numerous dynamical systems that exhibit characteristics typical of both continuous-time and discrete-time systems. For instance, circuits with both analog and digital components. These dynamical systems can be classified as Hybrid Dynamical Systems (HDS). Then, an HDS is a dynamical system that exhibits characteristics of both, continuous-time and discrete-time dynamical systems.

2.1 HDS Framework

The model of an HDS first proposed in 2004 [27] is represented in the following form:

$$\mathcal{H} \begin{cases} x \in C & \dot{x} \in F(x) \\ x \in D, & x^+ \in G(x), \end{cases} \tag{2.1}$$

where the state of the system is represented by x, which can change according to the differential inclusion $\dot{x} \in F(x)$ while in the set C, and it can change according to a difference inclusion $x^+ \in G(x)$ while in the set D. Furthermore, $\dot{x}$ represents the velocity of state x and x^+ represents the value of the state after an instantaneous change [26].

The behavior represented by the differential inclusion is referred to as *flow*, the behavior represented by the difference inclusion is referred to as *jumps*. The name of the four objects

in (2.1) are:

- $C \subset \mathbb{R}^n$ is called the flow set,
- a set-valued mapping $F : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ with $C \subset \text{dom}F$ is called the flow map,
- $D \subset \mathbb{R}^n$ is called the jump set,
- a set-valued mapping $G : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ with $D \subset \text{dom}G$ is called the jump map,

where the flow set, C, and the jump set, D, are subsets of an $n-$dimensional Euclidean space $\mathbb{R}^n$. We use double arrows ($\rightrightarrows$) to denote a set-valued mapping M, where we can regard a set-valued map as a correspondence or a multifunction that maps a point in $\mathbb{R}^n$ to a subset of $\mathbb{R}^n$. Set-valued mapping are a generalization of single-valued mappings. The HDS is represented by the notation $\mathcal{H} = (C, F, D, G)$.

Note that, when the flow set is equal to $\mathbb{R}^n$ and the jump set is the empty set this corresponds to a purely continuous-time system and when the jump set is equal to $\mathbb{R}^n$ with an empty flow set we obtain a purely discrete-time system [26].

2.1.1 Hybrid Time Domain

In continuous-time systems, solutions are parameterized by time, $t \in \mathbb{R}_{\geq 0}$, and in discrete-time systems, solutions are parameterized by discrete steps or jumps, $j \in \mathbb{N}$. HDS are parameterized by *hybrid time domains*, which is defined as: a subset $E \subset \mathbb{R}_{\geq 0} \times \mathbb{N}$ is a compact hybrid time domain if:

$$E = \bigcup_{j=0}^{J-1} ([t_j, t_{j+1}], j),$$

for some finite sequence of times $0 = t_0 \leq t_1 \leq t_2 \leq ... \leq t_J$. It is a hybrid time domain if for all $(T, J) \in E$, $E \cap ([0, T] \times 0, 1, ..., J)$ is a compact hybrid domain. Definition 2.3 from [26]. Figure 2.1 shows an example of a hybrid time domain E.

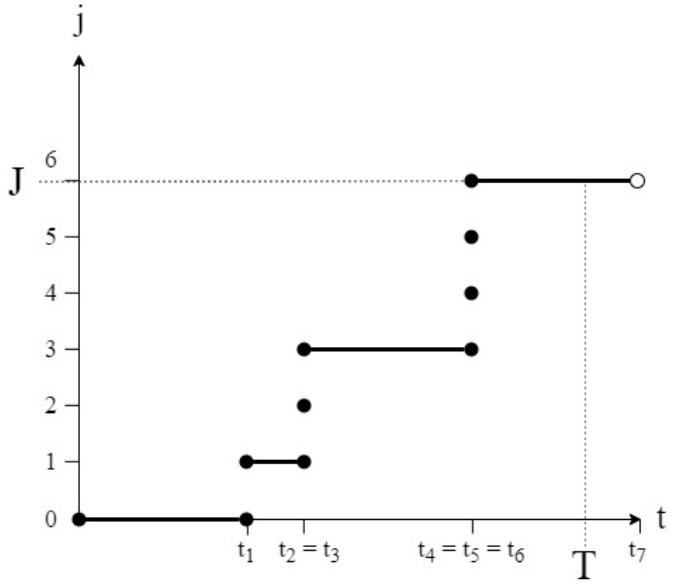


Figure 2.1: Hybrid Time Domain E

2.2 Switching Systems

Switching systems are continuous-time systems given by a family of differential equations, where the differential equation that governs the evolution of the state, is determined by a switching rule [26]. Let Q be a set where for each $q \in Q$, a function $f_q : \mathbb{R}^n \to \mathbb{R}^n$. When the switching signal σ, which takes values in Q, remains constant, the variable z evolves continuously according to the differential equation $\dot{z} = f_\sigma(z)$. However, when a switching signal occurs, it can be described by $\sigma^+ \in Q$ and the switched closed-loop system can be modeled as:

$$\mathcal{H} \begin{cases} z \in \mathbb{R}^n, q \in Q & \dot{z} \in f_q(z) \\ z \in \mathbb{R}^n, q \in Q & q^+ \in Q, \end{cases} \tag{2.2}$$

and the state $\mathbf{x}$ equals:

$$\begin{pmatrix} z \\ q \end{pmatrix} \in \mathbb{R}^{n+1},$$

where q remains constant during flows and z remains constant during jumps. Following (2.1) the switching system is modeled as:

$$\begin{pmatrix} \dot{z} \\ \dot{q} \end{pmatrix} \in F(x) := \begin{pmatrix} f_q(z) \\ 0 \end{pmatrix}, \quad \begin{pmatrix} z^+ \\ q^+ \end{pmatrix} \in G(x) := \begin{pmatrix} z \\ Q \end{pmatrix},$$

while $C = D = \mathbb{R}^n \times Q$ [26]. Note that, $Q = 3 - q$ is commonly used in switching systems when Q has only two possible values denoted $q = 1$ and $q = 2$.

2.3 HDS Example

As mentioned before, HDS can model numerous types of dynamical systems. In several cases opting for an HDS framework makes the stability and robustness analysis simpler. As it is with switching systems where a classic solution to the differential equation $\dot{z} = f_\sigma(z)$ consists of a piecewise constant function σ taking values in Q and a continuous and piecewise differentiable function z, with σ being right continuous and having left limits (CÀDLÀG function) that satisfy $\dot{z} = f_{\sigma(t)}(z(t))$ at all times t except the *switching instances* [26]. It is more general to consider an HDS switching system modeled by (2.2) and analyze it using Lyapunov theory, as shown in Theorem 3.18 (Sufficient Lyapunov Conditions) in [26]. One simple yet interesting example that illustrates the HDS framework is a bouncing ball. Consider a point-mass bouncing vertically ball on a horizontal surface. Where the state x is described as:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2,$$

where x_1 represents the height above the horizontal surface and x_2 represents the vertical velocity. Let the ball flow when the point-mass is above the surface or when it is at the surface and the velocity points up:

$$C = \{x \in \mathbb{R}^2 : x_1 > 0 \text{ or } x_1 = 0, x_2 \geq 0\}.$$

Let the flow map be defined as:

$$f(x) = \begin{pmatrix} x_2 \\ -\gamma \end{pmatrix} \in \mathbb{R}^2,$$

where $-\gamma$ is the acceleration due to gravity ($9.81 \frac{m}{s^2}$ for simulation purposes). Impacts happen when the point-mass is on the surface with negative velocity, hence the jump set is:

$$D = \{x \in \mathbb{R}^2 : x_1 = 0, x_2 < 0\},$$

and, lastly, the jump map is given by some $\lambda \in (0, 1)$:

$$f(x) = \begin{pmatrix} 0 \\ -\lambda x_2 \end{pmatrix} \in \mathbb{R}^2.$$

In the proposed model for initial conditions other than $x_0 = (0, 0)$, every jump is followed by a period of flow (either the ball is going up or down) meaning that consecutive jumps are not allowed to happen. Proposition 2.10 in [26] is satisfied showing the basic existence of solutions, meaning that the initial points, ξ with $\xi_1 > 0$ are in the interior of C and f is continuous on the interior of C, and let:

- (VC) $\xi \in \bar{C} \cup D$, if $\xi \in D$ or there exists $\varepsilon > 0$ and an absolutely continuous function $z : [0, \varepsilon] \to \mathbb{R}^n$ such that $z(0) = \xi$, $\dot{z}(t) \in F((z(t))$ for almost all $t \in [0, \varepsilon]$ and $z(t) \in C$ for all $t \in (0, \varepsilon]$, then there exists a nontrivial solution to, ϕ to $\mathcal{H}$ with $\phi(0, 0) = \xi$ [26].

The case where the initial condition $x_0 = (0, 0)$ is problematic given that it leads to *Zeno behavior* (i.e., pure jumps), applying a Krasovskii regularization (see Definition 2.12 in [77]) to the bouncing ball system Zeno behavior can be analyzed, see Example 8.5 in [26].

For the initial points ξ with $\xi_1 = 0$ and $\xi_2 > 0$, $f(\xi)$ points into the interior of C and this suggests that (VC) hold. Thus, from the initial point $\xi = 0$, there is the obvious solution $\phi(t, 0) = 0 \; \forall t \in \mathbb{R}_{\geq 0}$. Furthermore, also from Proposition 2.10 in [26] maximal solutions are complete, and every solution is bounded which can be shown using Lyapunov theory, see Example 3.19 in [26] for details.

Hence, given initial conditions ξ with $\xi_1 \geq 0$ and $\xi_2 \neq 0$ the first jump of the solution ϕ from ξ corresponds to the ball's first bounce:

$$t_1 = \frac{\xi_2 + \sqrt{\xi_2^2 + 2\gamma\xi_1}}{\gamma},$$

for $\lambda < 1$, $\sup_t \text{dom } \phi < \infty$.

Through numerical simulations letting $\lambda = 0.8$, $x_1 = 15$, and $x_2 = 0$ we find Figure 2.2 where we see that both x_1 and x_2 eventually converge to 0, as one would expect when bouncing a ball (it will not bounce indefinitely).

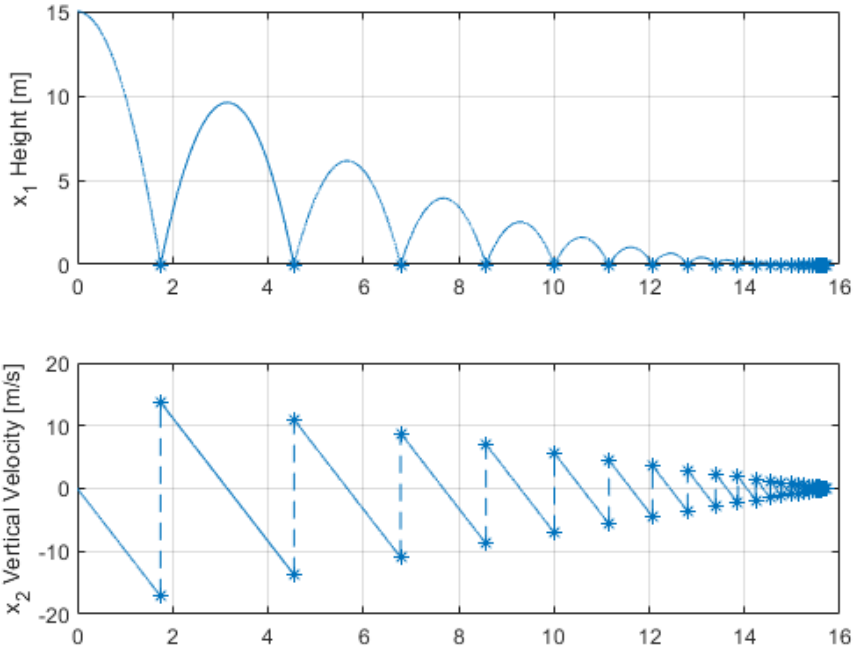


Figure 2.2: A Solution to the Bouncing Ball Example Starting at $x_0 = (15, 0)$

Chapter 3

Real-time Optimization using Averaging-Based Extremum Seeking Control

Extremum Seeking Control (ESC) is a model-free type of adaptive control where the input/output map is optimized, without any explicit knowledge other than that it exists, that it has at least a local extremum, and that we can measure it. ESC is a gradient based optimization technique, which relies on an appropriate exploration via a dither (or excitation signal), which could also be stochastic [80, 81], to provide with an approximate gradient, and consequently find the extremum. Using classic averaging and singular perturbation techniques the behavioral characteristics of the ESC are derived. The overall goal is to supplement a dynamical system to drive the input/output pair $[x, y]$ to an ϵ-neighborhood of the extremum pair $[x^*, y^*]$ [84].

3.1 Extremum Seeking Control (ESC) Analysis

Consider the classic ESC widely popular in literature [43, 96, 24] from Figure 3.1 where:

- ω is the oscillation frequency in $[\frac{rad}{s}]$
- a is the oscillation amplitude in $[rad]$ $(a << \hat{u})$
- h determines the cutoff frequency from the high-pass or washout filter
- ϵ scales the integrator gain that determines signal $\hat{u}$ $(\omega >> \epsilon)$

We can further simplify this general ESC by removing the low and high pass filter which are not essential to the stability analysis as mentioned in Chapter 2.3 of [24]. Now, we are left with Figure 3.2 where the only design parameters are the oscillation amplitude a and ω the oscillation frequency of the dither signal. We assume the dither signal is not correlated to the system noise, which could cause the algorithm to incorrectly update u [84].

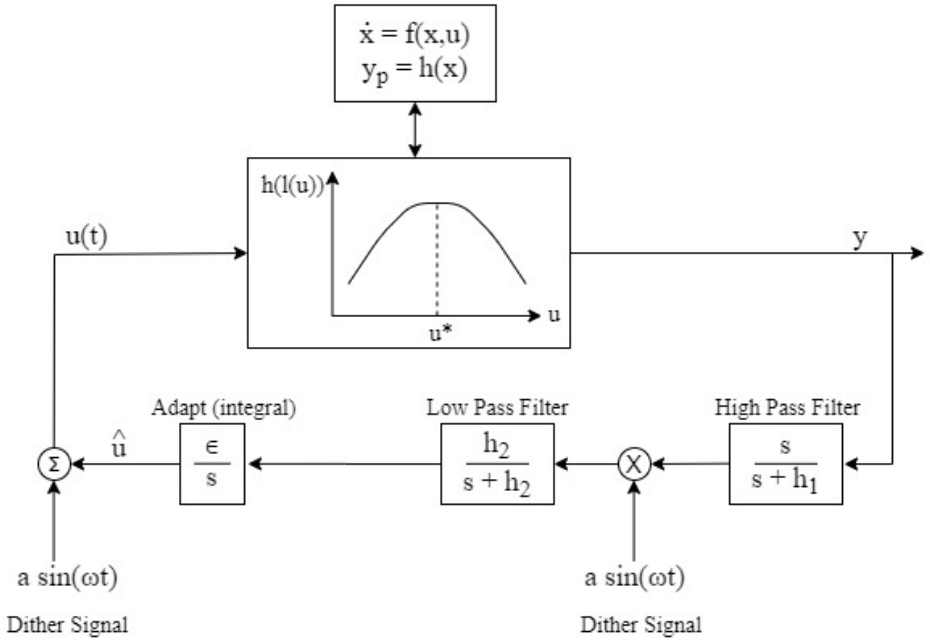


Figure 3.1: Classic Extremum Seeking Algorithm for Unknown Output Function, adapted from [85]

In its simpler form, the ESC equations can be summarized as:

$$\dot{x} = f(x, \hat{u} + asin(\omega t)) \tag{3.1}$$

$$\dot{\hat{u}} = \frac{\epsilon}{a} h(x) sin(\omega t). \tag{3.2}$$

3.1.1 Classic Averaging and Singular Perturbation Theory

Assume that both $\hat{u}$ and the dither signal are slow compared to the system dynamics. By applying singular perturbation theory as explained in Chapter 11 of [41] and following

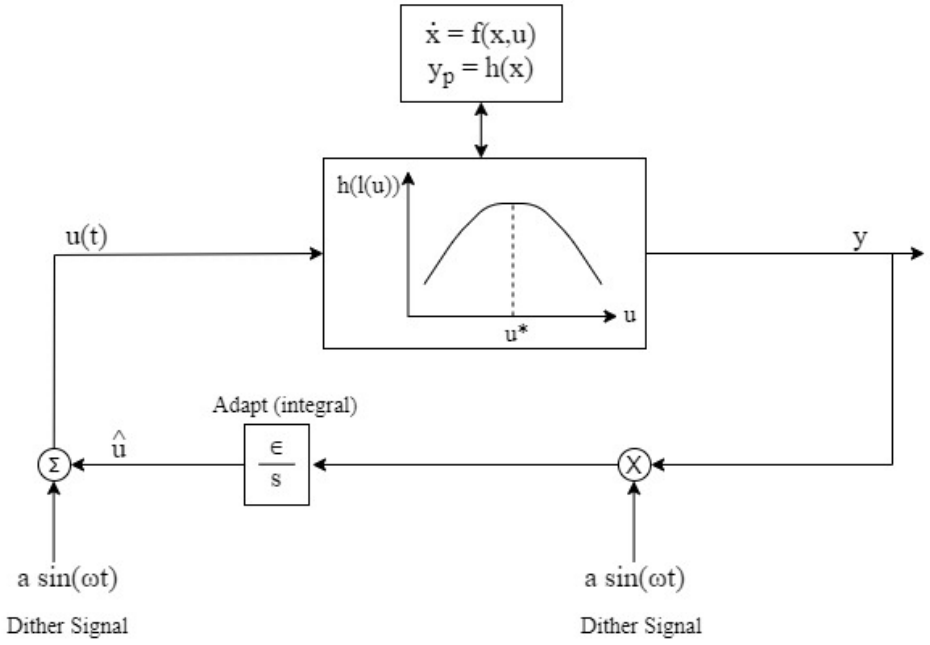


Figure 3.2: Simplified Extremum Seeking Algorithm for Unknown Output Function, adapted from[85]

the ideas explained in [55, 60, 85, 84] we can approximate x as:

$$x(t) = l(\hat{u} + asin(\omega t)) + \xi(t) + \mu(t), \tag{3.3}$$

where $\xi(t)$ captures the transients in the x-system, the fastest dynamics, and converges quickly to zero as $t \to \infty$ and the $\mu(t)$-term is small for small $a\omega$ and ϵ. From the approximation (3.3) for x and eliminating the fast-time response from the slow-learning dynamics we obtain equation (3.4) which is in three different time scales. The ξ-term represents the fast-dynamics (captures the transients in the x-system), the dither signal and disturbance are the *regular*-dynamics or *medium-fast* dynamics and the presented dynamics in (3.4) are the slow-dynamics.

$$\dot{\hat{u}} = \frac{\epsilon}{a}[(h(l(\hat{u} + asin(\omega t)) + \xi(t) + \mu(t))]sin(\omega t), \tag{3.4}$$

note that (3.4) is now in perturbation standard form described in Chapters 1.6 and 2.1 from [74] and we can proceed to apply averaging theory following Chapter 10 from [41]. Let the

average dynamics to be described by:

$$\dot{u}_{avg} = \epsilon \lim_{S\to\infty} \frac{1}{aS} \int_0^S (h(l(u + asin(\omega\tau)) + \xi(\tau) + \mu(\tau)))sin(\omega\tau)d\tau. \tag{3.5}$$

Thus, (3.5) is simplified to:

$$\dot{u}_{avg} = \frac{\epsilon}{aT} \int_0^T (h(l(u + asin(\omega\tau)))sin(\omega\tau))d\tau, \tag{3.6}$$

where $T = \frac{2\pi}{\omega}$, applying Taylor expansion to 3.6:

$$h(l(u + asin(\omega\tau)) \approx h(l(u)) + a\nabla h(l(u))sin(\omega t) + O(a^2), \tag{3.7}$$

note that (3.7) is approximately proportional to the steep descent direction $h_{avg}(u, a) = \frac{1}{2}\nabla h(l((u)) + O(a)$. Then, for a small a and assuming $h(l(u))$ in (3.6) is integrable, then $h(l(u + asin(\omega\tau))$ exist and approximates the gradient [84].

The analysis assumes a point-mass model, however as shown in Chapter 3.3 of [24] and in [95] classic averaging and singular perturbation theory can still be used to analyze the system dynamics, where:

$$\dot{x}_c = vcos(\theta)$$

$$\dot{y}_c = vsin(\theta)$$

$$\dot{\theta} = \omega_0.$$

Chapter 4

Data-enabled Extremum Seeking (DES)

As previously mentioned, the Extremum Seeking (ES) goal is to optimize a response map when it is not available and we can only rely on measurements taken in real-time where the controller operates in a slower-time scale by adjusting an ϵ-gain. This approach depends on an appropriate exploration via the dither [59]. The response map is defined as: the steady-state relationship between the input parameters of a plant and its performance output (i.e., objective or cost function). Hence, ES is constrained by the *exploration versus exploitation* dilemma [29], because of the use of an external dither signal to guarantee enough exploration on the objective function and to facilitate optimization making classical ES algorithms depend on persistence of excitation (PE) [66].

A way to create an ES algorithm that dispenses with the PE condition is to incorporate recorded past data into the feedback controller [15]. In the case of ES control, we already have large amounts of recorded data, hence online learning and real-time optimization that incorporates recorded data have become widely popular in the context of experience replay [53], interactive learning [58], deep learning [89], and reinforcement learning [36]. However, in the context of ES it still not ubiquitous [62]. Nonetheless, as shown in [62, 63], the use of sufficiently rich stored data to synergistically combine concurrent learning and ES has shown promising results.

4.1 Convergence Analysis

Concurrent learning requires both, real-time and recorded measurement data; thus, to guarantee convergence a response map approximation analysis for both is required. Then, by combining both approximation requirements, convergence can be guaranteed. To achieve a useful response map approximation, we implement a shallow neural network with a radial basis activation function.

We start by considering the dynamical system:

$$\dot{\theta} = f(\theta, z) \tag{4.1}$$

$$y = h(z), \tag{4.2}$$

where $f : \mathbb{R}^s \times \mathbb{R}^n \to \mathbb{R}^s$ is a Lipschitz continuous function, $h : \mathbb{R}^s \times \mathbb{R}^n$ is a C^2 output function, $\theta \in \mathbb{R}^s$; where a C^k function means a function with k continuous derivatives. The state is restricted to evolve in a compact set $\Theta \subset \mathbb{R}^s$ and $z \in \mathbb{R}^n$ is the input, which is restricted to evolve in a pre-defined compact set $\mathcal{F} \subset \mathbb{R}^n$. We assume that the functions f and h are unknown [62].

Assume that there exists a continuous function $l : \mathbb{R}^n \times \mathbb{R}^s$ satisfying $l(\mathcal{F}) \subset \Theta$ such that:

$$(\theta, z) \in \Theta \times \mathcal{F}, \begin{cases} \dot{\theta} = f(\theta, z) \\ \dot{z} = 0, \end{cases} \tag{4.3}$$

generates complete solutions for every initial condition and renders Uniform Global Asymptotic Stability (UGAS) to the set $H := \{(\theta, x), \in \mathbb{R}^{s+n} : \theta = l(z), z \in \mathcal{F}$.

- Where a set $\mathcal{A} \subset \mathbb{R}^n$ is said to be uniformly globally stable (UGS) to (2.1) if there exists a function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ where α is zero at zero, continuous, strictly increasing, and unbounded — known as $\alpha \in \mathcal{K}_\infty$; such that any solution ϕ to (2.1) satisfies $|\phi(t, j)|_{\mathcal{A}} \leq \alpha(|\phi(0, 0)|_{\mathcal{A}})$, $\forall (t, j) \in dom(\phi)$. Furthermore, the set $\mathcal{A} \subset \mathbb{R}^n$ is

uniformly globally pre-attractive (UGpA) to (2.1) if for each $\epsilon > 0$ and $r > 0$, there exists $T > 0$ such that for any solution ϕ to (2.1),

- $|\phi(t,j)|_{\mathcal{A}} \leq r \Rightarrow \phi(t,j)|_{\mathcal{A}} \leq \epsilon$, $\forall (t,j) \in \text{dom}(\phi, t+j \geq T)$. Lastly, the he set $\mathcal{A} \subset \mathbb{R}^n$ is UGAS to (2.1) if it is both UGS and UGpA [67].

Here the response map is defined as $\phi(z) := h(l(z), z)$ where h is the output function and l the mapping generated by (4.3). Assume that ϕ is a convex, continuously differentiable function on an open set $\mathcal{F} \subset D$. Furthermore, $\mathcal{F}$ is compact, convex, and nonempty [62].

Under these assumptions the problem we are trying to solve is, a real-time convex optimization:

$$\begin{aligned} &\text{minimize} \quad \phi(z) \\ &\text{subject to} \quad z \in \mathcal{F}. \end{aligned} \tag{4.4}$$

Every solution to z^* in our optimization problem is also a solution to the Variational Inequality (VI):

$$(z - z^*)\nabla\phi(z^*) \geq 0, \quad \forall z \in \mathcal{F}. \tag{4.5}$$

where $\nabla\phi(z^*)$ is the gradient of $\phi(z^*)$, Theorem 6.12 in [71].

The constrained convex ES problem (4.4) can also be framed as:

$$\mathcal{A} := \{z^* \in K : (z - z^*)\nabla\phi(z^*) \geq 0, \ \ \forall x \in \mathcal{F}\}. \tag{4.6}$$

By our earlier assumption that $\mathcal{F}$ is compact, convex, and nonempty and by letting $K \subset \mathbb{R}^n$ is compact, convex, and $\phi : K \to \mathbb{R}^n$ is continuous, then $\mathcal{A}$ is nonempty and compact [21].

Note that, we again can use a feedback-based data-driven algorithm designed to converge to the solutions of (4.5) to solve the convex optimization problem (4.4), like ES.

4.1.1 Response Map and Gradient Approximation

Let, (4.7) be an approximation to the response map, $\phi(z)$:

$$\phi(z) = b(z)^T w^* + \epsilon(z), \quad \forall z \in \mathcal{F}, \tag{4.7}$$

where ϵ is a continuously differentiable approximation error, $w^* \in \mathbb{R}^p$ is a vector of ideal weights, and b is a continuously differentiable pre-defined vector of basis functions. Where the set of ideal weights is defined as:

$$\mathbb{W}_{\phi,\mathcal{F}} := \{w^* \in \mathbb{R}^p : |f(z) - w^{*T}\phi(z)| \leq \epsilon(z), z \in \mathcal{F}\}, \tag{4.8}$$

which we assume to be compact. Note that b in (4.7) can be any admissible activation function such as sigmoid and rectified linear unit (ReLu) functions [4, 90]. Nonetheless, we opt to use radial basis functions, which are commonly used to approximate functions [93] [62]. From (4.7) and smoothness, we can compute $\nabla\phi(z)$:

$$\nabla\phi(z) = \nabla b(z)^T w^* + \nabla\epsilon(z), \quad \forall z \in \mathcal{F}, \tag{4.9}$$

where ∇b is the Jacobian matrix of b and by the Weierstrass High-order Approximation Theorem, theorem 2.4.13 from [18] the approximation errors ϵ and $\nabla\epsilon$ converge to zero as the number of basis p increases. This approximation method is commonly known as a *shallow neural network* or a *single-layer neural network* [62]. To solve (4.4), let $\hat{\phi}(z)$ approximate to the response map (4.7) defined as:

$$\hat{\phi}(z) := b(z)^T \hat{w}, \tag{4.10}$$

where $\hat{w} \in \mathbb{R}^p$ is an auxiliary state. The approximation error of the response map can be defined as:

$$\begin{aligned} e_{ss}(z) &:= \hat{\phi}(z) - \phi(z) \\ &= b(z)^T\hat{w} - b(z)^T w^* - \epsilon(z) \\ &= b(z)^T(\hat{w} - w^*) - \epsilon(z). \end{aligned} \tag{4.11}$$

Similarly, the approximation error of the gradient can be defined as: $\nabla e_{ss}(z) := \nabla b(z)^T(\hat{w} - w^*) - \nabla\epsilon(z)$. This shows that if $\hat{w} - w^* = 0$ the approximation errors, $\epsilon(z)$ and $\nabla\epsilon(z)$ will be of order $O(\sigma)$ where $\sigma > 0$ can be made arbitrarily small by increasing the dimension of the basis vector. To minimize the estimation error by using a gradient descent law, a PE

condition needs to be satisfied by the basis function to achieve convergence; meaning that there exists a $T > 0$ and $\gamma > 0$ such that:

$$\int_{t}^{t+T} b(\tau)b(\tau)^T d\tau \geq \gamma I, \tag{4.12}$$

for all $t \geq 0$. Note that in practical conditions is difficult a priori the satisfaction of (4.12) [62].

4.1.2 Data-driven Response Map Approximation

To dispense with the PE condition [40] let us consider the use of recorded data, where the recorded data is a finite sequence of inputs/outputs $\{(z_k, y_k)\}_{k=1}^{J}$ where $k = 1, 2, ..., J$ and $z_k := z(t_k)$. Obtaining a class of Data-enabled Extremum Seeking Dynamics (DES) that use concurrently real-time and recorded data of the input/output system. Now, let $b(z_k)$ be the basis function evaluated at the measured data-points and we can now define the steady-state estimation error using the recorded data:

$$\begin{aligned} e_{ss}(z_k) &:= \hat{\phi}(z_k) - \phi(z_k) \\ &= (\hat{w} - w^*)^T b(z_k) - \epsilon(z_k). \end{aligned} \tag{4.13}$$

Just as before, we need to satisfy an a priori condition to guarantee convergence. Now the data must be (γ, J)-*sufficiently rich* (SR); which is defined as:

$$\sum_{k=1}^{J} x_k x_k^T \geq \gamma I_p, \tag{4.14}$$

where $\gamma > 0$ [62]. By defining a matrix data, D:

$$D := [x_1, x_2, ..., x_k, ..., x_J] \in \mathbb{R}^{p \times J},$$

the inequality (4.14) can be expressed as:

$$\sum_{k=1}^{J} x_k x_k^T = DD^T \in \mathbb{R}^{p \times J}, \tag{4.15}$$

since the rank(DD^T) = rank(D^T) (see appendix A in [79]) then the inequality (4.15) holds if the rank$(D) = p$. Which means that by verifying the rank of the matrix data, D, a priori we can satisfy the inequality [62].

4.1.3 Convergence Result

The DES dynamics can now be described by:

$$\begin{aligned} \dot{\hat{w}} &= \frac{\varepsilon_1}{\varepsilon_2} F_{\hat{w}}(\hat{w}, z, y) \\ \dot{z} &= \varepsilon_1 F_z(g, z), \end{aligned} \tag{4.16}$$

where $\hat{w}$ is the same auxiliary state defined in (4.10), $g = \nabla b(z)^T \hat{w}$ and the function $F_{\hat{w}}$ equals:

$$F_{\hat{w}} := -\alpha_1 \Psi(z)(\hat{\phi}(z) - y) - \alpha_2 \sum_{k=1}^{J} \Psi(z_k)(\hat{\phi}(z_k) - y_k), \tag{4.17}$$

for $\alpha_1, \alpha_2 > 0$, where $\Psi(y)$ equals:

$$\Psi(y) := \frac{b(y)}{(1 + b(y)^T b(y))^2}. \tag{4.18}$$

The pairs (z, y) and (z_k, y_k) correspond to the real-time and recorded measurement data, respectively and function (4.18) is a normalized function see chapter 8.5.5 in [33] and [63, 90].

Function $F_{\hat{w}}$ (4.17), has two components: a real-time measurement driven output, which can be seen as normalized gradient descent to minimize the squared error, and a recorded error driven used to relax the PE condition [62].

To solve the ES optimization problem, the input/output data, $(\{z_k, y_k)\}_{k=1}^{J}$, must be *consistent*, meaning that it must satisfy:

$$|y(t_k) - h(l(z(t_k)), z(t_k))| \leq \frac{\widetilde{p}}{J}, \tag{4.19}$$

for each $\widetilde{p} > 0$ and for all $k \in \{1, 2, 3, ..., J\}$. Which means the data point y_k is a measurement of the dynamics, (4.1), output that is $\widetilde{p}$-close to a steady-state condition imposed by z_k:

$$z \in \mathcal{F}, \quad \dot{z} = F_z(\nabla \phi(z), z), \tag{4.20}$$

where (4.20) satisfies:

- F_z is locally Lipschitz continuous,
- the set $\mathcal{A}$ is UGAS,
- for each bounded continuous function $d : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, and each $z(0) \in \mathcal{F}$ the perturbed system $F_z(\nabla\phi(z) + d, z)$ generates complete solutions [62].

An ideal algorithm candidate that satisfies every assumption is Lipschitz Projected Gradient Descent [23, 94]:

$$\dot{z} = -z + P_{\mathcal{F}}(z - \nabla\phi(z)), \tag{4.21}$$

for $z \in \mathcal{F}$ where $P_{\mathcal{F}}(z) := \arg\min_{y \in P_{\mathcal{F}}} |z - y|$. where (4.21) is used in chapter 5 to create the hybrid DES controller [62].

Chapter 5

Main Results

Using the concepts discussed in chapters 2, 3, and 4 we can now build the proposed algorithms and showcase the numerical results.

Following the analysis from chapter 3.2 and the ideas presented in [55, 60, 74, 85], let the obstacle-free adaptive control dynamics be:

$$u_x = a\omega\mu_2 + k\xi_x \tag{5.1}$$

$$u_y = -a\omega\mu_1 + k\xi_y \tag{5.2}$$

$$\dot{\xi} = \bar{\omega}(\frac{2\mu}{a}J(\widetilde{p} + a\mu) - \xi) \tag{5.3}$$

$$\dot{\widetilde{p}} = \sigma\xi \tag{5.4}$$

$$\frac{\bar{\omega}}{\omega}\dot{\mu}_1 = \mu_2 \tag{5.5}$$

$$\frac{\bar{\omega}}{\omega}\dot{\mu}_2 = -\mu_1, \tag{5.6}$$

where $k = \sigma\bar{\omega}$, $\bar{\omega}$ is a scaling constant, J is a quadratic cost function, $\widetilde{p} = [x - a\mu_1, y - a\mu_2]^T$, $\xi = [\xi_x, \xi_y]^T$, μ_1 and μ_2 are the dither signals generated via a dynamic oscillator, and $\frac{\bar{\omega}}{\omega}$ is sufficiently small. Now, by averaging the dynamic system (5.3) along the solutions of (5.5 and 5.6), and, as we did previously in chapter 3.2.1, we perform a Taylor expansion of the cost function, J, to obtain:

$$J(\widetilde{p} + a\mu) = J(\widetilde{p}) + a\mu^T\nabla J(\widetilde{p}) + e_r,$$

where e_r is of order $O(a^2)$ and knowing that:

$$\int_0^{\frac{2\pi\bar{\omega}}{\omega}} \mu_i(t)dt = 0$$

$$\frac{\omega}{2\pi\bar{\omega}} \int_0^{\frac{2\pi\bar{\omega}}{\omega}} \mu_i(t)^2 dt = \frac{1}{2}$$

$$\int_0^{\frac{2\pi\bar{\omega}}{\omega}} \mu_i(t)\mu_j(t)dt = 0 \ \ \forall i \neq j.$$

Then, the averaged system becomes:

$$\dot{\xi}_{avg} = \bar{\omega}(\nabla J(\widetilde{p}_{avg}) + e_R - \xi_{avg})$$

$$\dot{\widetilde{p}}_{avg} = \sigma \cdot \xi_{avg},$$

where e_{ravg} is of order $O(a)$. If we let $\sigma = 0$ in $\dot{\widetilde{p}}_{avg}$, by linearity the averaged system of the *fast* dynamics, the system has an equilibrium point at $\nabla J(\widetilde{p}_{avg}) + e_{ravg}$ which is exponentially stable. Now, to obtain *slow* dynamics let $\nabla J(\widetilde{p}_{avg}) + e_{ravg}$ replace ξ_{avg} to obtain the *slow* dynamics:

$$\dot{\widetilde{z}} = \nabla \cdot J(\widetilde{p}) + e_r,$$

and, as mentioned in chapter 3.2.1, adjusting the tuning parameters ($\frac{\bar{\omega}}{\omega}$, σ, a) the dynamical system in (5.1-5.6) approximates, on average, in the slowest time scale a gradient system. Now, if we let $e_r = 0$ and assume the cost function, J, is smooth and it has an strict global maximum and for each $\alpha \in \mathbb{R}$ the set $\{(x, y) : J(x, y) \geq \alpha\}$ is compact and contains no points where $\nabla J(x, y) = 0$ other than the strict global maximum point, $[x^*, y^*]^T$ (note that these conditions are met for a quadratic function); the unperturbed gradient system guarantees Uniform Global Asymptotic Stability (UGAS) of the unique extremum point via the Lyapunov function $J(\widetilde{z}^*) - J(\widetilde{z})$, where $\widetilde{z}^* := [x^*, y^*]^T$ [60].

Furthermore, using the continuity of $\nabla J(\cdot)$ and the UGAS property by lemma 7.20 from [26] for $a \to 0^+$ system, $\dot{\widetilde{z}}$, is SGP-AS (Semi-globally Practically-Asymptotically Stable) [87]

and by Theorem 1 from [26] our dynamical system (5.1-5.6) is also SGP-AS. This means that the proposed feedback law can be tuned to guarantee convergence of the vehicle to any ϵ-neighborhood of $[x^*, y^*]^T$ from any compact set K of initial conditions [60]. However, when we introduce obstacles and even if we introduce potential functions to J to *push* the solutions away from saddle points it may be problematic. Because the inclusion of obstacles introduces at least one saddle point (critical point) [60, 68, 83] where the smooth time-invariant feedback system could locally converge in the presence of arbitrarily small adversarial signals, $e(t)$.

To illustrate this, consider a divided state-space in three sets $\mathcal{K}_1, \mathcal{K}_2$, and $\mathcal{K}$ using the closed-loop time-invariant controller, $\dot{\widetilde{z}}$, with $\widetilde{z} = 0$ and where $\dot{\widetilde{z}}(\cdot)$ is locally bounded and where there exists at least one Carathéodory complete solution $\widetilde{z}$, satisfying $\widetilde{z}(t) \in \mathcal{K}_1 \setminus \mathcal{K}$, $\forall\ t \in [0, T]$ where $T > 0$ of $\dot{\widetilde{z}}$ [60]. By adding at least one obstacle this introduces a topological obstruction, which guarantees the existence of a curve $\mathcal{K}_1$ or $\mathcal{K}_2$ such that for initial conditions on each side of $\mathcal{K}_1$ or $\mathcal{K}_2$ the trajectories of the vehicle approach the set $\mathcal{K}$ either from above or below the obstacle. This implies that it is possible to find arbitrarily small signals, $e(t)$, acting on the states of the vehicle such that some of the trajectories of the closed-loop system will remain in the neighborhood of the line $\mathcal{K}_1$ or $\mathcal{K}_2$ and consequently, will not converge to the set $\mathcal{K}$ by Theorem 6.5 from [76] see [60].

Therefore, as previously stated and following the ideas introduced in [54, 60] to guarantee the stability properties even in the presence of small adversarial signals, $e(t)$, a hybrid feedback law is required. This hybrid feedback law partitions the state-space and adds a switching state $q \in \{1, 2\}$ by modifying our cost function J to a localization function J_q, which is defined as:

$$J_q(x) = (x_1 - x_{t1})^2 + (x_2 - x_{t2})^2 + B(d) \tag{5.7}$$

$$B(d_q) = \begin{cases} (d_q - \rho)^2 ln(\frac{1}{d_q}), & d_q \in [0, \rho] \\ 0, & d_q > \rho, \end{cases} \tag{5.8}$$

where $\rho \in (0, 1]$ is a threshold variable, $d_q(x_1, x_2) = |[x_1, x_2]^T|^2_{\mathbb{R}^2}$ is a distance mapping

function, which maps a position to a squared value of its distance, and $q \in \{1, 2\}$ is our switching logic variable. The function d_q, with $\bar{d} = x_2 - r_2 + d_s$, is defined as:

- When $q = 1$, and a tolerance constant $ds \in \mathbb{N}$, is greater than $(x_1 - r_1)^2 + (x_2 - r_2)^2$ (where r_1 and r_2 are the location coordinates of the detected obstacle) d_q is defined as:

$$d_1 = \begin{cases} \frac{r_1 + r_2 - x1 - x2 - ds}{\sqrt{2}}, & \bar{d} < r_1 - x_1, x1 - r_1 < x_2 - r_2 + d_s \\ \frac{x_1 - r_1 + r_2 - x_2 - d_s}{\sqrt{2}}, & \bar{d} > r_1 - x_1, x_1 - r_1 > x_2 - r_2 + d_s \\ \sqrt{(x_1 - r_1)^2 + (x_2 - r_2)^2} - d_s, & \textit{otherwise} \end{cases}$$

- When $q = 2$ and $ds > (x_1 - r_1)^2 + (-x_2 + r_2)^2$, d_q is defined as:

$$d_2 = \begin{cases} \frac{r_1 - r_2 - x1 + x2 - ds}{\sqrt{2}}, & -\bar{d} < r_1 - x_1, x1 - r_1 < r_2 - x_2 + d_s \\ \frac{x_1 - r_1 - r_2 + x_2 - d_s}{\sqrt{2}}, & -\bar{d} > r_1 - x_1, x_1 - r_1 > r_2 - x_2 + d_s \\ \sqrt{(x_1 - r_1)^2 + (r_2 - x_2)^2} - d_s, & \textit{otherwise} \end{cases}$$

- When $q = 1$ and $ds \leq (x_1 - r_1)^2 + (x_2 - r_2)^2$ or $q = 2$ and $ds > (x_1 - r_1)^2 + (r_2 - x_2)^2$, d_q, d_q is defined as: $d_q = 0$

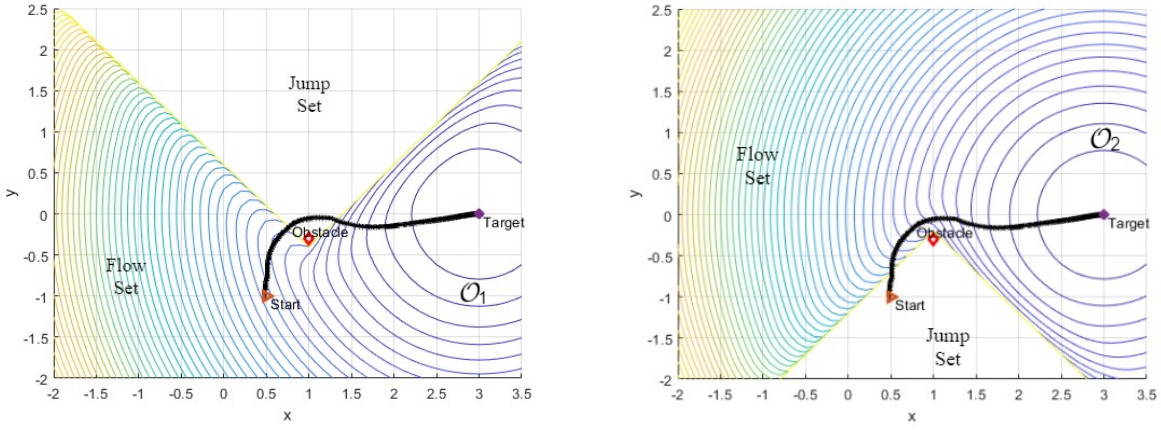
- For any other case, $d_q = 0$

Note that $[x, y] = [x_1, x_2]$ for the rest of the chapter.

5.1 Hybrid Adaptive Framework

By dividing the state-space using a barrier function we introduce the sets $\mathcal{O}_1$ and $\mathcal{O}_2$ as shown in Figures 5.1a and 5.1b. Note that the *Jump Sets* from Figures 5.1a and 5.1b are not just the empty (white) areas, but also include points close to the barrier functions. The idea is to construct a box around the obstacle and project adjacent sides of the box to divide the space in two parts. By doing this, $\mathcal{O}_1 \cup \mathcal{O}_2$ covers $\mathbb{R}^2$ entirely apart from of the

box that includes the detected obstacle. Then, using switching logic defined in C and D the vehicle switches from its current logic state, q. We can see from Figure 5.1a and 5.1b how the state-space is divided in two, flow and jump sets and we can tell from these figures when the controller triggered the switch logic and *jumped* to logic state $q = 2$, where the flow set is defined and then the jumps back to $q = 1$ and successfully converge to the global maximum, $[x^*, y^*]$.



(a) Space $\mathcal{O}_1$, including points near (5.8)

(b) Space $\mathcal{O}_2$, including points near (5.8)

Figure 5.1: Barrier Functions Space Division, $\mathcal{O}_1$ and $\mathcal{O}_2$

An important question is how to detect obstacles at unknown locations. To accomplish this, we attach a sensor to the vehicle of radius c, which can detect obstacles of radius b when the vehicle is close enough to an obstacle. The variable p stores the number of obstacles detected and r stores the location of the detected obstacle. If more than one obstacle is detected the controller will not flow until just one obstacle is detected, the radius c will be multiplied by a small variable $\eta \in (0, 1]$ to reduce the radius size until one obstacle is detected. Moreover, the number of obstacles detected is given by the cardinality of the set $P(x, c)$ which is defined as:

$$P(x, c) = \{i \in \mathcal{N} : (x + c\mathbb{B} \cap (r_i + b_0\mathbb{B}) \neq 0\} \tag{5.9}$$

where $\mathbb{B}$ denotes the unit ball in Euclidean space centered at the origin and $\mathcal{N}$ denotes the set $\{1, 2, 3, ..., \mathcal{N}\}$ where $\mathcal{N} \in \mathbb{N}$ [54].

The jump set is defined as: $D = D_{1a} \cup D_{1b} \cup D_2 \cup D_3$, where D_{1a} defines the regions for where the vehicle has detected an obstacle when no previously obstacles were detected: $D_{1a} = \{X \in \mathbb{Z} : |P(x, c)| = 1, p = 1\}$ where $r^+ = r$ and $p^+ = |P(x, c)|$ (5.9). D_{1b} is the set defined where multiple obstacles are detected simultaneously: $D_{1b} = \{X \in \mathbb{Z} : |P(x, c)| \geq 2\}$, where $p^+ = |P(x, \eta c)|$ (5.9) and $c^+ = \eta c$. D_2 is the set where the logic state q is updated where we introduce two design parameters $\mu > 1$ and $\lambda > 0$ to trigger jumps as: $D_2 = \{X \in \mathbb{Z} : J_q(x, r, p, c) \geq (\mu - \lambda) J_{3-q}(x, r, p, c), p \geq 1\}$ and as mentioned we employ the switch logic $Q = 3 - q$ to update q. Lastly, D_3 is the set where the vehicle has successfully evaded the obstacles after the use of the model-free hybrid adaptive controller defined as: $D_3 = \{X \in \mathbb{Z} : |P(x, c_0)| = 0, p \geq 1\}$ where $p^+ = |P(x, c)|$ (5.9) and $c^+ = c_0$ [54].

The flow set is defined as $C = C_1 \cup C_2 \cup C_3$, where C_1 denotes the region where no obstacles are present, defined as: $C_1 = \{X \in \mathbb{Z} : x \in \mathbb{R}^2, |P(x, c)| = 0, p = 0, c = c_0\}$. C_2 is the set where a single obstacle is detected: $C_2 = \{X \in \mathbb{Z} : J_q(x, r, p, c) \leq \mu J_{3-q}(x, r, p, c), |P(x, c)| = 1, p \geq 0\}$, and C_3 is the set where multiple obstacles are detected resulting in the reduction of the radius of detection region, c, defined as:$C_3 = \{X \in \mathbb{Z} : x \in \mathbb{R}^2, |P(x, c_0)| \geq 1, |P(x, c)| = 0, p \geq 1\}$.

Putting these definitions together the flow and jump sets are defined as:
Flow Set,

$$
\begin{aligned}
C &= C_1 \cup C_2 \cup C_3 \\
C_1 &= \{X \in \mathbb{Z} : x \in \mathbb{R}^2, |P(x, c)| = 0, p = 0, c = c_0\} \\
C_2 &= \{X \in \mathbb{Z} : J_q(x, r, p, c) \leq \mu J_{3-q}(x, r, p, c), |P(x, c)| = 1, p \geq 0\} \\
C_3 &= \{X \in \mathbb{Z} : x \in \mathbb{R}^2, |P(x, c_0)| \geq 1, |P(x, c)| = 0, p \geq 1\}
\end{aligned}
\tag{5.10}
$$

Jump Set,

$$\begin{aligned} D &= D_{1a} \cup D_{1b} \cup D_2 \cup D_3 \\ D_{1a} &= \{X \in \mathbb{Z} : |P(x,c)| = 1, p = 1\} \\ D_{1b} &= \{X \in \mathbb{Z} : |P(x,c)| \geq 2\} \\ D_2 &= \{X \in \mathbb{Z} : J_q(x,r,p,c) \geq (\mu - \lambda) J_{3-q}(x,r,p,c), p \geq 1\} \\ D_3 &= \{X \in \mathbb{Z} : |P(x,c_0)| = 0, p \geq 1\} \end{aligned} \tag{5.11}$$

The flow and jump sets in the algorithms implemented are the same, given that the jump rules do not change if the dynamics change. We still follow the same partitioning and switch logic. However, the jump and flow maps do change, depending on the algorithm and model dynamics, given that the *flow instructions* vary, and the jump variable updates varies depending on the defined variables. Furthermore, the proposed switching rule in chapter 2 is implemented, $Q(q) := 3 - q$.

Additionally, to numerically analyze the algorithms' robustness we introduce a perturbation $e_i = \epsilon s_i$, where $i \in \{1, 2\}$. By definition ϵ is a small number and s_i is defined as:

$$s_1 = \begin{cases} x - 0.6, & (x - 0.6) \leq 0.15 \quad \text{or} \quad (x - 0.6) \geq -0.15 \\ 0, & \textit{otherwise}, \end{cases} \tag{5.12}$$

$$s_2 = \begin{cases} y, & (y - 0.6) \leq 0.05 \quad \text{or} \quad (y - 0.6) \geq -0.05 \\ 0, & \textit{otherwise}, \end{cases} \tag{5.13}$$

5.1.1 Model-Free Hybrid Adaptive Controller

Now, implementing the ideas mentioned in chapter 2, 3, and 5.1 we can build the hybrid real-time optimization algorithm using point-mass and nonholonomic unicycle dynamics.

5.1.1.1 Point-Mass Dynamics

Using point-mass dynamics to model the vehicle, see Figure 5.2, the dynamics are quite similar to (5.1 - 5.6) with the exception that for the hybrid model-free controller we require the use of the localization function, J_q instead of the cost function J. Furthermore, we include the perturbation signals to the x and y signal and update the sensor radius and detection variables.

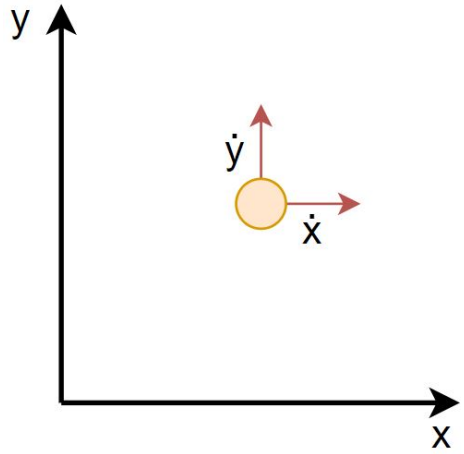


Figure 5.2: Point-mass Dynamics Model

Hence, $\mathcal{H}_{pm}$ is defined as:

- Flow Map, $F(x)_{pm}$, $\dot{\mu}_1 = -\omega sin(\omega t)$, $\dot{\mu}_2 = \omega cos(\omega t)$, $\dot{\xi}_1 = -\bar{\omega}(x_1 - \frac{2sin(\omega t)}{a} J_q)$, $\dot{\xi}_2 = -\bar{\omega}(x_2 - \frac{2cos(\omega t)}{a} J_q)$, $\dot{x}_1 = a\omega cos(\omega t) + kx_1 - \epsilon s_1$, $\dot{x}_2 = -a\omega sin(\omega t) + kx_2 - \epsilon s_2$, $\dot{q} = 0$, $\dot{p} = 0$, $\dot{c} = 0$
- Jump Map, $G(x)_{pm}$: $\mu_1^+ = \mu_1$, $\mu_2^+ = \mu_2$, $\xi_1^+ = \xi_1$, $\xi_2^+ = \xi_2$, $x_1^+ = x_1$, $x_2^+ = x_2$, $q^+ = 3 - q$, $p^+ = p$, $c^+ = c$
- Flow Set is defined in (5.10)
- Jump Set is defined in (5.11)

5.1.1.2 Nonholonomic Dynamics

The nonholonomic unicycle dynamics are modeled as:

$$\dot{x}_c = vcos(\theta), \quad \dot{y}_c = vsin(\theta), \quad \dot{\theta} = \omega_0, \tag{5.14}$$

where $[x_c, y_c]$ is the center, θ the orientation, $\dot{\theta}$ is the angular velocity of Figure 5.3, v is the forward velocity input, and ω_0 the angular velocity input. To simply the analysis and equations we assume that the sensor of the nonholonomic model is exactly at the center of the vehicle, which would make r from Figure 5.3 equal to zero.

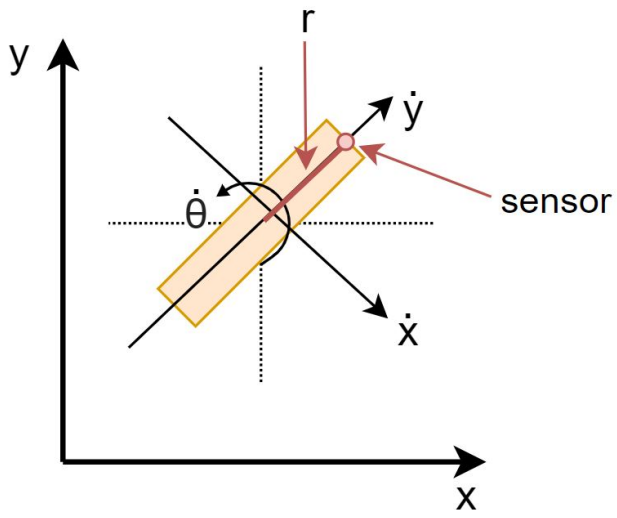


Figure 5.3: Nonholonomic Unicycle Dynamics Model

Nonholonomic systems, are systems which have constraints, which are nonintegrable into positional constrains. In the case of (5.14) this means the position of the unicycle is given by a pair of coordinates (x, y) and an additional coordinate; the orientation of the unicycle, which is specified by an angle θ. For any position, the unicycle can be pointed in any direction, θ. Thus, there is no constraint of the form $f(x, y, \theta, t) = 0$, which means that there is a constraint that: $\dot{x} = v(\cos(\theta x)+\sin(\theta y))$ or $\dot{y} = \dot{x}\tan(\theta)$. These are nonholonomic constraints, so we cannot reduce the number of variables in the system [86]. The physical implication of this system is that the orientation will keep oscillating, something that is

visible in the trajectory of the numerical studies, chapter 5.2.2.

To analyze the nonholonomic system we need to drive the averaging results following the same ideas we implemented when derived the point-mass dynamics. We can still, guarantee convergence of the vehicle to any ϵ-neighborhood of $[x^*, y^*]$ from any compact set K of initial conditions, however as previously mentioned the orientation will keep oscillating changing the trajectory shape as shown in [95].

Thus, changing the flow and jump maps to adjust the nonholonomic dynamics using the localization function, and including a perturbation we find that $\mathcal{H}_{nh}$:

- Flow Map, $F(x)_{nh}$: $\dot{\mu}_1 = -\omega sin(\omega t)$, $\dot{\mu}_2 = \omega cos(\omega t)$, $\dot{\xi}_1 = -\bar{\omega}(x_1 - \frac{2sin(\omega t)}{a} J_q)$, $\dot{\xi}_2 = -\bar{\omega}(x_2 - \frac{2cos(\omega t)}{a} J_q)$, $\dot{\theta} = \frac{\omega}{5}$, $\dot{x}_1 = -a\omega(cos(\omega t)cos(\frac{t\omega}{k_1}) + \alpha\dot{\theta}cos(\omega t)sin(\frac{t\omega}{k_1})) + kx_1 - \epsilon s_1$, $\dot{x}_2 = a\omega - cos(\omega t)sin(\frac{t\omega}{k_1}) + \alpha\dot{\theta}cos(\omega t)sin(\frac{t\omega}{k_1}) + kx_2 - \epsilon s_2$, $\dot{q} = 0$, $\dot{p} = 0$, $\dot{c} = 0$
- Jump Map, $G(x)_{nh}$: $\mu_1^+ = \mu_1$, $\mu_2^+ = \mu_2$, $\xi_1^+ = \xi_1$, $\xi_2^+ = \xi_2$, $\theta^+ = \theta$, $x_1^+ = x_1$, $x_2^+ = x_2$, $q^+ = 3 - q$, $p^+ = p$, $c^+ = c$
- Flow Set is defined in (5.10)
- Jump Set is defined in (5.11)

5.1.2 DES Controller

As mentioned in chapter 4, we can use Projected Lipschitz Gradient Descent, Lipschitz-PGD, to find the dynamics required to implement the DES algorithm, Figure 5.4.

Gradient Descent (GD) is a standard way to solve unconstrained optimization functions, GD iterates the following equation until a stopping condition is met:

$$z_{k+1} = z_k - y_k \nabla\phi(z_k),$$

in our case, the stopping condition would be to find the extremum values of the localization function, J_q. However, for a constrained situation, like the one we are solving, a projection is required.

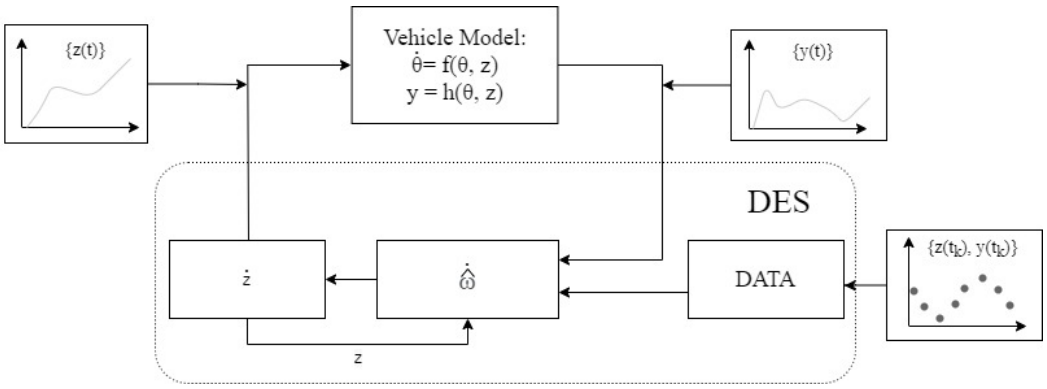


Figure 5.4: DES Algorithm Block Diagram, adapted from [62]

Projected Gradient Descent (PGD) is a common technique to solve constrained optimization problems. Consider the set $\mathcal{F}$, starting from an initial point $z_0 \in \mathcal{F}$, PGD iterates the equation:

$$z_{k+1} = P_{\mathcal{F}}(z_k - y_k \nabla\phi(z_k)),$$

where $P_{\mathcal{F}}$ is a Euclidean Lipschitz projection operator $P_{\mathcal{F}}(z) := argmin_{y \in P_{\mathcal{F}}|z-y|}$, given the point z_0 the projection aims to find a point $z \in \mathcal{F}$ which is *near* to z_0 [63, 62].

Now, if ϕ is a convex and a Lipschitz function (the gradient is bounded above) the PGD convergence rate is of $O(\frac{1}{\sqrt{k}})$ and if ϕ is additionally a smooth function, the rate of convergence will be the same as GD, $O(\frac{1}{k})$ [3]. As shown in [62] when $\nabla\phi : \mathbb{R}^n \to \mathbb{R}^n$ is a smooth and strongly monotone mapping and the assumption that $\mathcal{F}$ is compact, convex, and nonempty holds, Lipschitz-PGD results in:

$$\dot{z} = -z + P_{\mathcal{F}}(z - \nabla\phi(z)),$$

for $z \in \mathcal{F}$ where $P_{\mathcal{F}}(z) := \arg\min_{y \in P_{\mathcal{F}}}|z-y|$. Which renders exponentially stable to the set of solutions of the variation inequality (VI) (4.5) which is a singleton under strong monotonicity of ϕ and render UGAS to the compact set $\mathcal{A}$ provided $\nabla\phi : \mathbb{R}^n \to \mathbb{R}^n$ is monotone, under theorem 3 in [23].

By applying the DES algorithm and using Lipschitz-GPD we obtain the following $\mathcal{H}_{DES}$ set:

- Flow Map, $F(x)_{nh}$: $\dot{x}_1 = 5(-x + P_{\mathcal{F}\infty} - \epsilon s_1)$, $\dot{x}_2 = 5(-y + P_{\mathcal{F}\in} - \epsilon s_2)$, $\dot{w}[1:n] = -\gamma(D_{real} + D_{measured})$, $\dot{q} = 0$, $\dot{p} = 0$, $\dot{c} = 0$
- Jump Map, $G(x)_{DES} = x_1^+ = x_1$, $x_2^+ = x_2$, $w^+[1:n] = w[1:n]$, $q^+ = 3 - q$, $p^+ = p$, $c^+ = c$, where $n = \frac{(i+1)(i+2)}{2}$ and i is equal to the polynomial order of the response map.
- Flow Set is defined in (5.10)
- Jump Set is defined in (5.11)

Where $P_{\mathcal{F}}$ equals:

$$P_{\mathcal{F}} = \begin{bmatrix} min(\frac{r_{ns}}{\sqrt{p_1^2+p_2^2}}) & 1 \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$$

$$p_1 = x - \nabla\hat{\phi}_1$$

$$p_2 = y - \nabla\hat{\phi}_2,$$

where $\hat{\phi}$ is the estimation of the localization function's gradient, (4.9) and r_{ns} is the navigation set radius, D_{real} are the real-time component dynamics equal to (4.18) times (4.13 and $D_{measured}$ are the data-driven component dynamics equal to (4.18) times (4.11) [63, 62].

The DES controller was tested with four different approximation functions: quadratic, polynomial 4^{th} order, 6^{th} order, and 8^{th} order. The only change that this introduced was increasing the amount of weights, vector of basis function Jacobian matrix of the basis function, b. Moreover, as expected the higher the degree order the slower the controller converged.

5.2 Numerical Studies

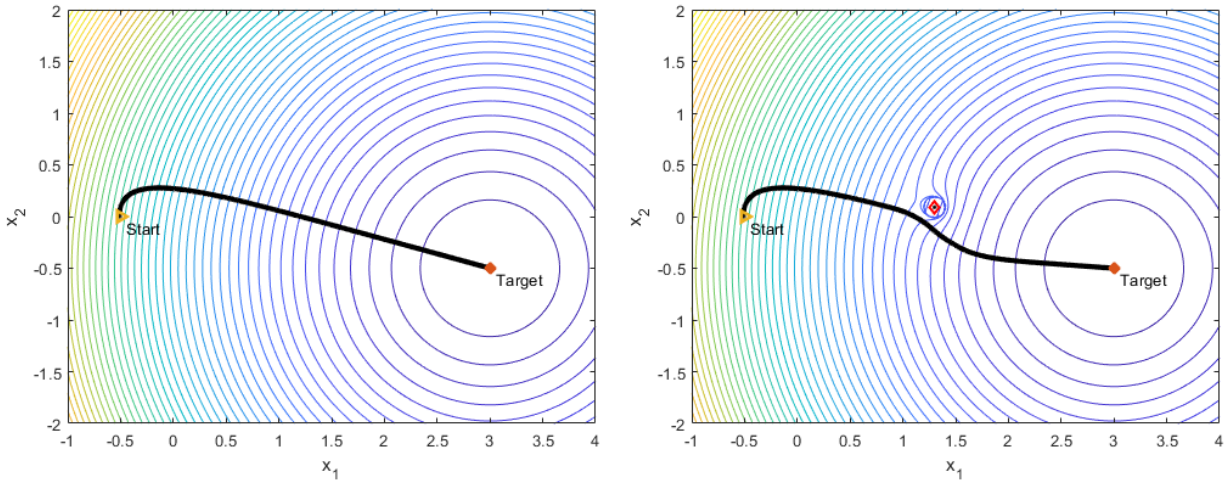
To validate the Hybrid Dynamical Adaptive Controllers, $\mathcal{H}_{pm}$, $\mathcal{H}_{nh}$, and $\mathcal{H}_{DES}$ presented in this chapter we used the Hybrid Equation (HyEQ) Toolbox in MATLAB [75] which facilities the simulation of hybrid dynamical systems.

The typical values used for every constant variable mentioned throughout the analysis are in Table A.1.

5.2.1 Hybrid Adaptive Controller: Point-Mass Dynamics

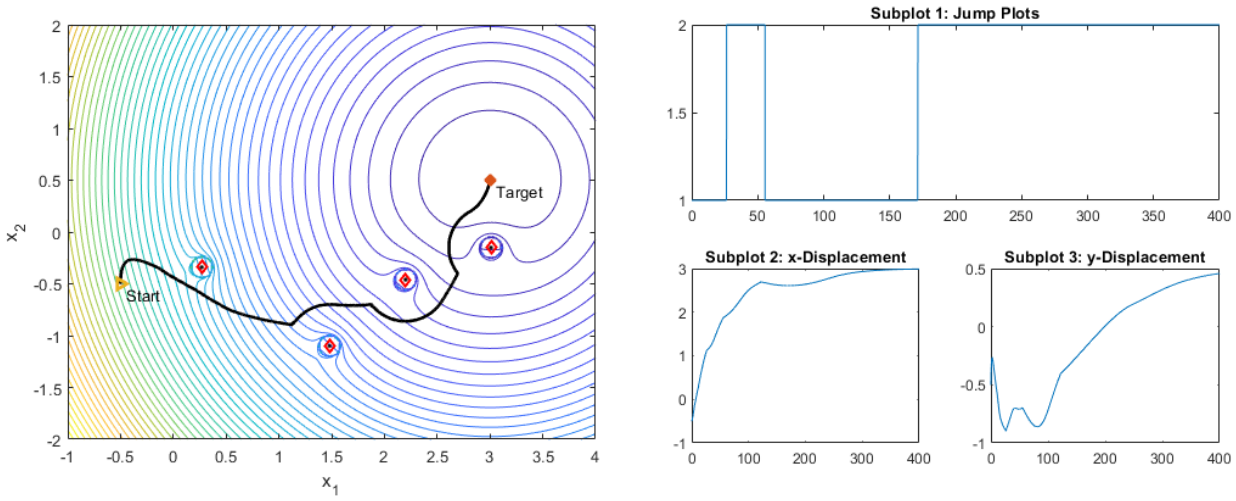
From $\mathcal{H}_{pm}$ and using the values defined in Table A.1, we proceed to numerically analyze the controller with point-mass dynamics. First, we analyze the source seeking control under the presence of no obstacles to guarantee convergence. From Figure 5.5a we see that under the presence of no obstacles the algorithm behaves just as a classical extremum seeking controller.

To truly test the obstacle avoidance property of the controller, we add one Figure 5.5b and multiple obstacles Figure 5.6a. From Figure 5.6b we can see the switching mechanism described in chapter 5.1 properly jumping between logical states to achieve convergence under the presence of one or multiple obstacles.



(a) Source Seeking (b) Single Obstacle Avoidance

Figure 5.5: No Obstacle Source Seeking and Single Obstacle Avoidance



(a) Multiple Obstacle Avoidance

(b) Jump Plots and Gradient Descent

Figure 5.6: Multiple Obstacle Avoidance with its Jump and $[x, y]$ Optimization Plots

5.2.1.1 Point-mass Dynamics Perturbation

Another important numerical analysis is under the presence of an ϵ-perturbation where recall that we define the perturbation or disturbance signal as: $e(t) = \epsilon s_i(t)$, where $i \in \{1, 2\}$ and s_i is defined in (5.12) and (5.13). For a perturbation signal with $\epsilon = 0.1$ Figure 5.7 and Figure 5.8a is the path before adding an obstacle in the presence of $\epsilon = 0.33$ and Figure 5.8b is after the presence of a single obstacle. Note that the convergence results are unaffected even though the path trajectory is different. However, when $\epsilon > 0.34$ we start seeing convergence issues. By adjusting ω, k, and a we can find results that converge, however for these combinations of ω, k, and a that an $\epsilon \leq 0.34$ is the upper limit. However, note that $\epsilon = 0.33$ is considered a large adversarial perturbation. Common ϵ-perturbation sizes are of 0.01, 0.05, and 0.1 see [68, 83].

5.2.2 Hybrid Adaptive Controller: Nonholonomic Unicycle Dynamics

From $\mathcal{H}_{nh}$ and using the values defined in Table A.1, we proceed to numerically analyze the controller with nonholonomic unicycle dynamics. Again, first we analyze the source

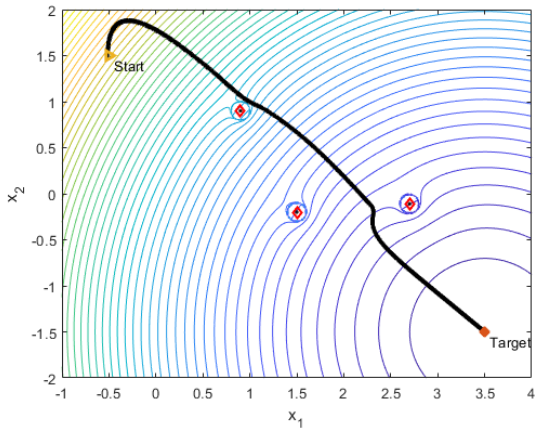
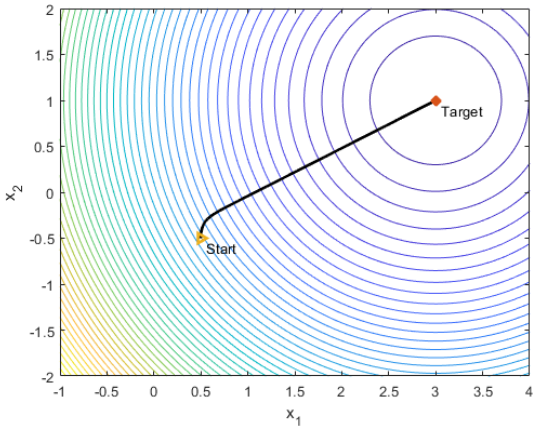
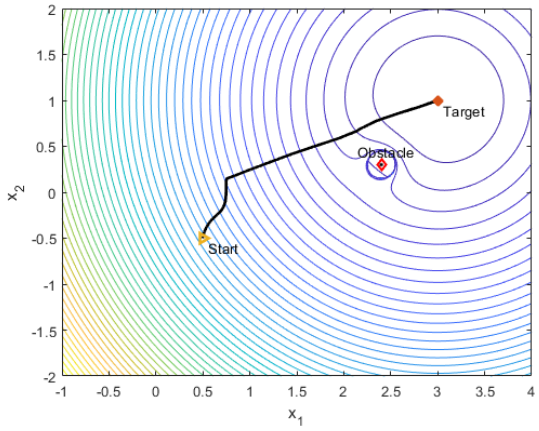


Figure 5.7: Multiple Obstacle Avoidance with $\epsilon = 0.1$



(a) No Obstacle Perturbation Path



(b) Single Obstacle Avoidance with $\epsilon = 0.33$

Figure 5.8: Point-mass Dynamics under the Presence of an Adversarial Signal

seeking control under the presence of no obstacles to guarantee convergence, see Figure 5.9.

Now, we jump to a multiple obstacle analysis Figure 5.10a and Figure 5.11a, where we can see that the controller successfully evades four obstacles directly in the vehicle's path. Furthermore, we see both jump plots in Figure 5.10b and Figure 5.11b where the controller performs two and four successful jumps between logic states to avoid any collisions.

Interestingly, as expected, the nonholonomic unicycle dynamic model's trajectory is different from the point-mass dynamic model, given that the orientation keeps oscillating making the vehicle's trajectory wider than the point-mass model.

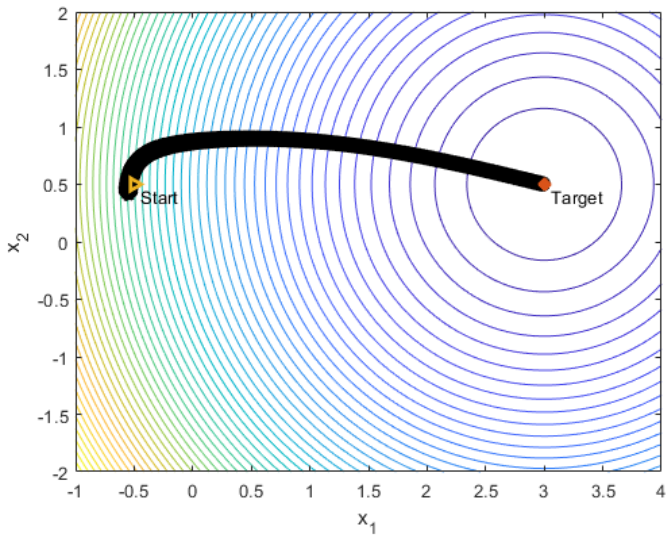
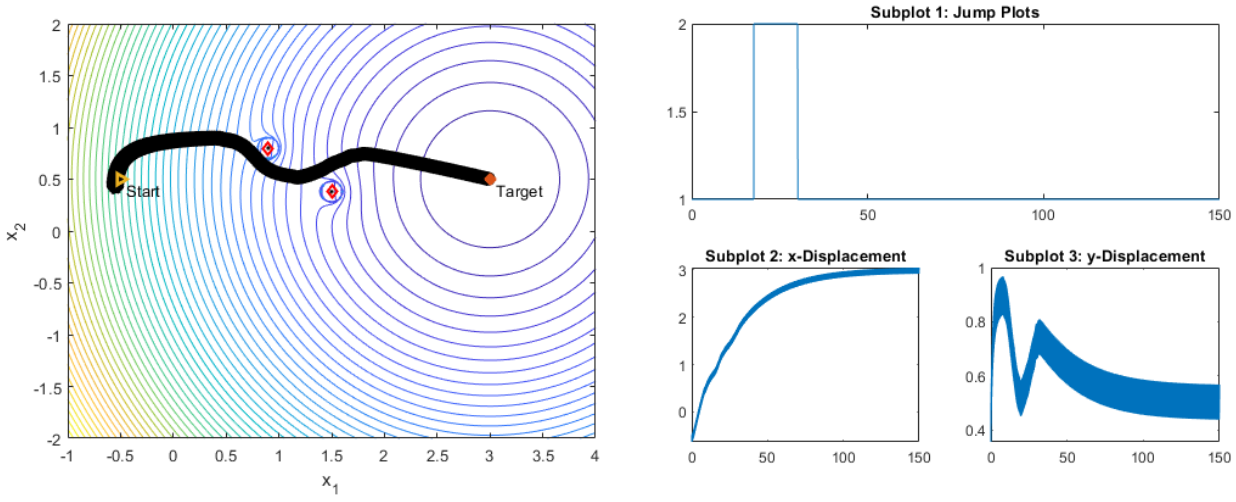


Figure 5.9: Source Seeking



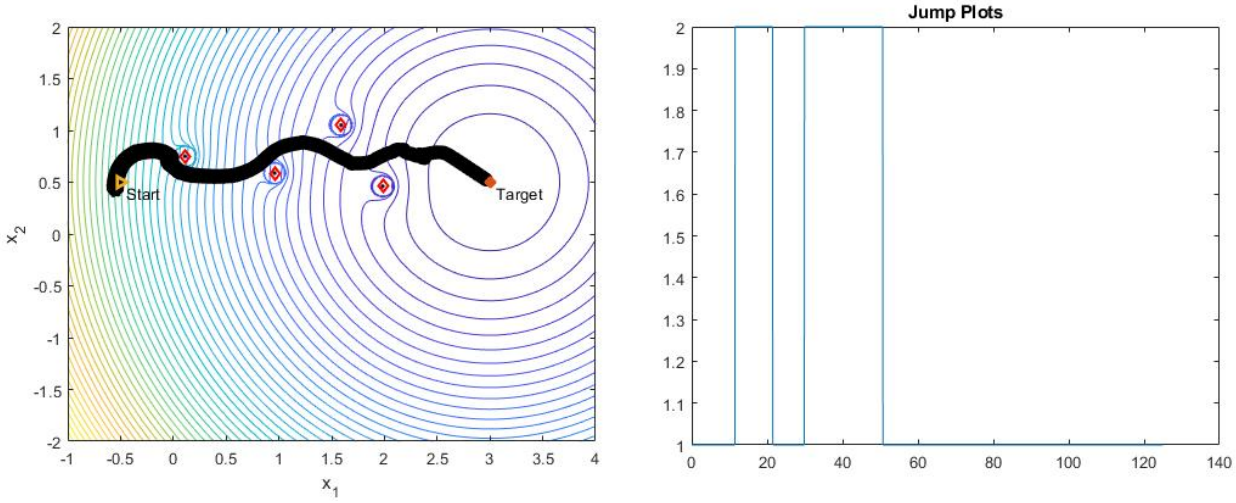
(a) Double Obstacle Avoidance

(b) Jump Plots and Gradient Descent

Figure 5.10: Multiple Obstacle Avoidance with its Jump and $[x, y]$ Optimization Plots

5.2.2.1 Nonholonomic Dynamics Perturbation

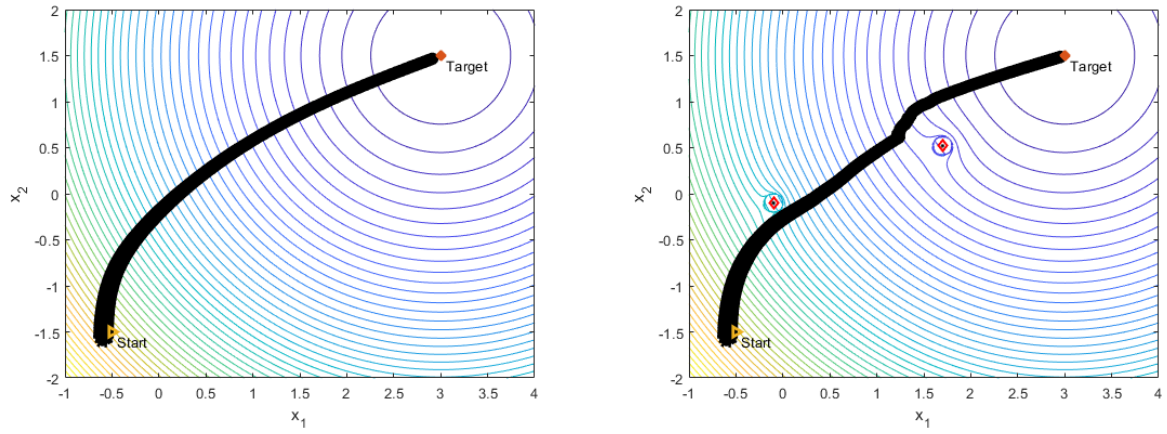
As shown with the point-mass dynamics, the nonholonomic unicycle dynamics can also successfully converge in the presence of adversarial signals. In this case, an ϵ-perturbation



(a) Multiple Obstacle Avoidance (b) Corresponding Jump Plot

Figure 5.11: Multiple Obstacle Avoidance of Nonholonomic Unicycle Dynamics

defined in (5.12) and (5.13) is applied. Under a perturbation signal with $\epsilon = 0.1$ Figure 5.12a shows convergence in the presence of no obstacles. Figure 5.12b shows how the controller successfully converges under the presence of two obstacles.



(a) No Obstacle Perturbation Path (b) Double Obstacle Avoidance with $\epsilon = 0.1$

Figure 5.12: Nonholonomic Unicycle Dynamics under the Presence of an Adversarial Signal

Furthermore, Figure 5.13 shows that the convergence results are unaffected even though the path trajectory is different under the presence of an $\epsilon = 1.05$. We start seeing convergence

issues when $\epsilon > 1.1$. By adjusting ω, k, and a we can find results that converge, however for these combinations of ω, k, and a that an $\epsilon \leq 1.05$ is the upper limit.

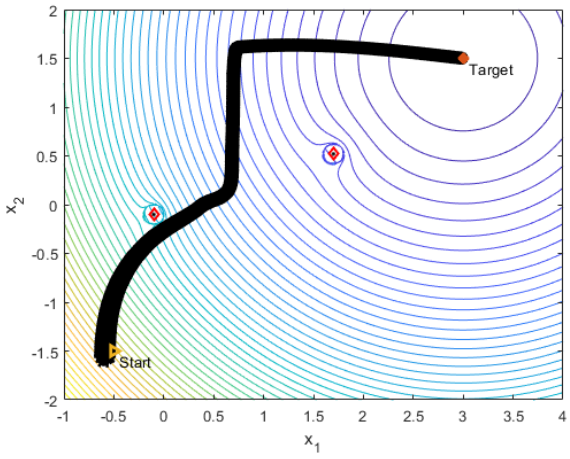


Figure 5.13: Double Obstacle Avoidance with $\epsilon = 1.05$

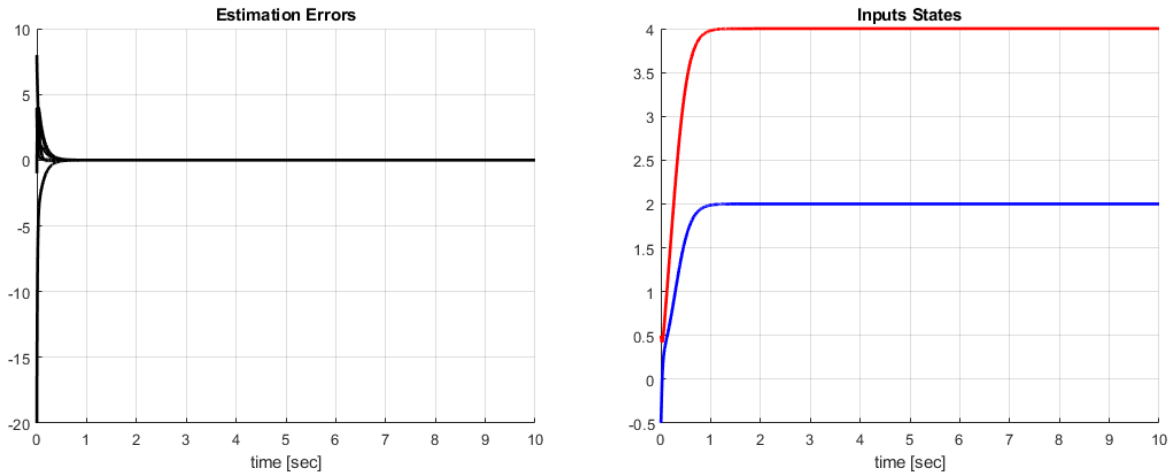
5.2.3 DES Controller

Lastly we analyze the numerical results of $\mathcal{H}_{DES}$, even though we expect similar results to the previous numerical studies the DES controller does not require the persistence of existence (PE) condition, meaning that the path should be a straight line with no oscillations.

We start the analysis with a quadratic response map, $\phi(z)$ and a state-space with no obstacles, by satisfying the a priori (γ, J)-Sufficiently Rich condition by showing that $\text{rank}(D) = p$, which in this case $\text{rank}(D) = 6$. Then as shown in Figure 5.14a and Figure 5.14b the parameter estimation errors approach zero as $t \to \infty$ and the input states approach the extremum values $[x, y] \to [x^*, y^*]$ as $t \to \infty$.

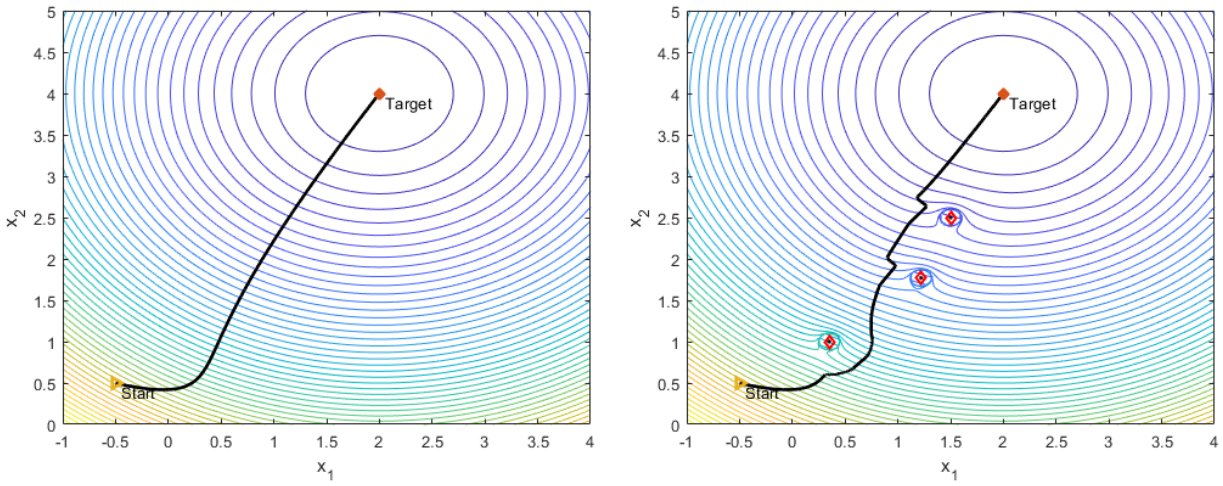
Consequently, the vehicle optimizes the response map as shown in Figure 5.15a. Adding obstacles to the state-space yield the same results with the hybrid DES algorithm optimizing the response map as shown in Figure 5.15b.

Furthermore, the algorithms still optimize the response map when using 4^{th}-order, 6^{th}-order, and 8^{th}-order parametric approximations of the potential field [Figure 5.17a - 5.18a].



(a) Parameter Estimation Error (b) Input States Gradient Descent

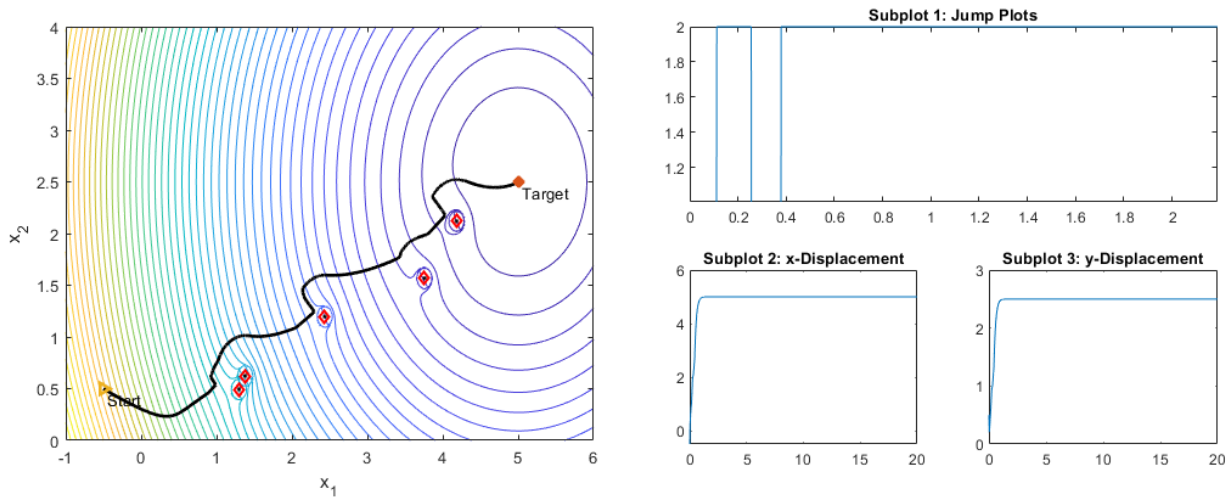
Figure 5.14: Error and State Convergence



(a) Source Seeking (b) Triple Obstacle Avoidance

Figure 5.15: DES Quadratic Function No Obstacle and Double Obstacle Avoidance

Note, that as the polynomial order increases the response map becomes *flat*, which makes the Lipschitz-PGD algorithm slower. Refer to Table 5.1 for a time comparison among response maps using the same numerical analysis and variable constants (performance test under the same conditions) Note that as simulation complexity increases, the time gaps become wider.



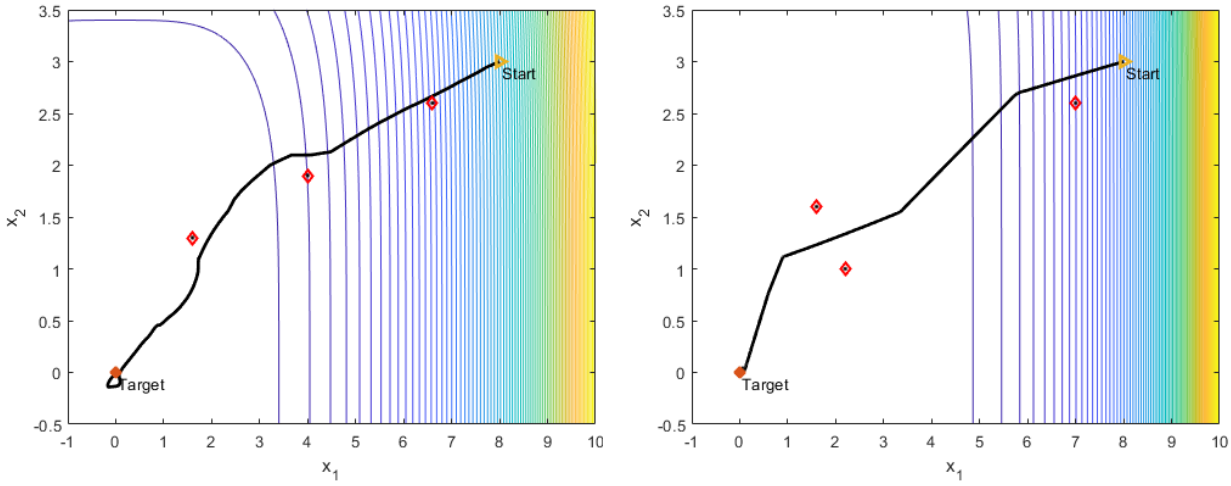
(a) Multiple Obstacle Avoidance (b) Jump Plot and Gradient Descent

Figure 5.16: DES Quadratic Function Multiple Obstacle Avoidance

However, we can see that as $t \to \infty$ even for the 8^{th}-order polynomials the state variable reaches the extremum $[x, y] \to [x^*, y^*]$ see Figure 5.18b.

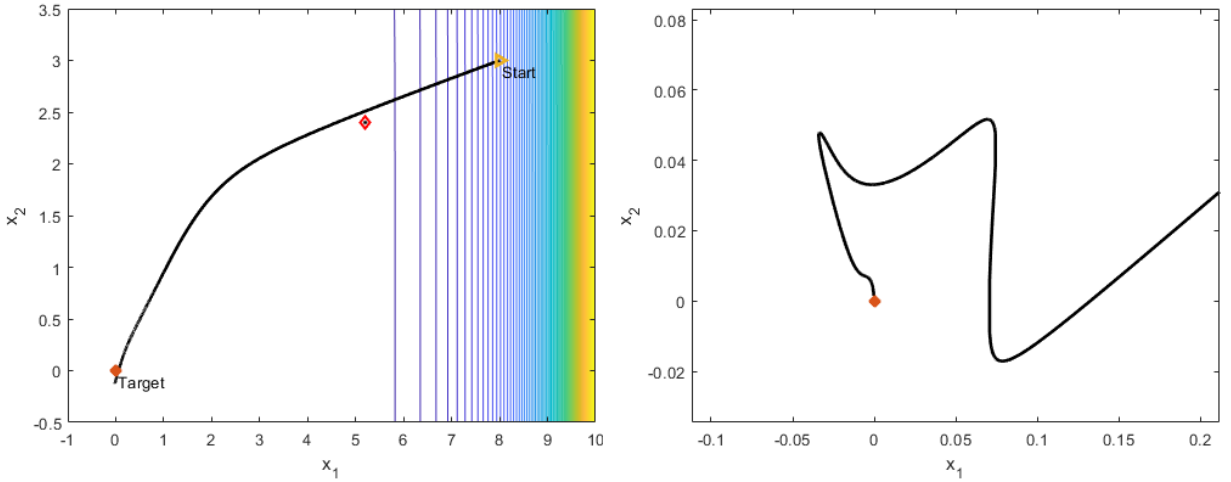
Response Map, $\phi(z)$	*Time [s]*
2^{nd} Order Polynomial Function	6.25
4^{th} Order Polynomial Function	17.65
6^{th} Order Polynomial Function	23.68
8^{th} Order Polynomial Function	38.18

Table 5.1: Convergence Time Comparison



(a) 4^{th}-Order Multiple Obstacle Avoidance (b) 6^{th}-Order Multiple Obstacle Avoidance

Figure 5.17: Nonholonomic Unicycle Dynamics for 4^{th} and 6^{th} Response Map



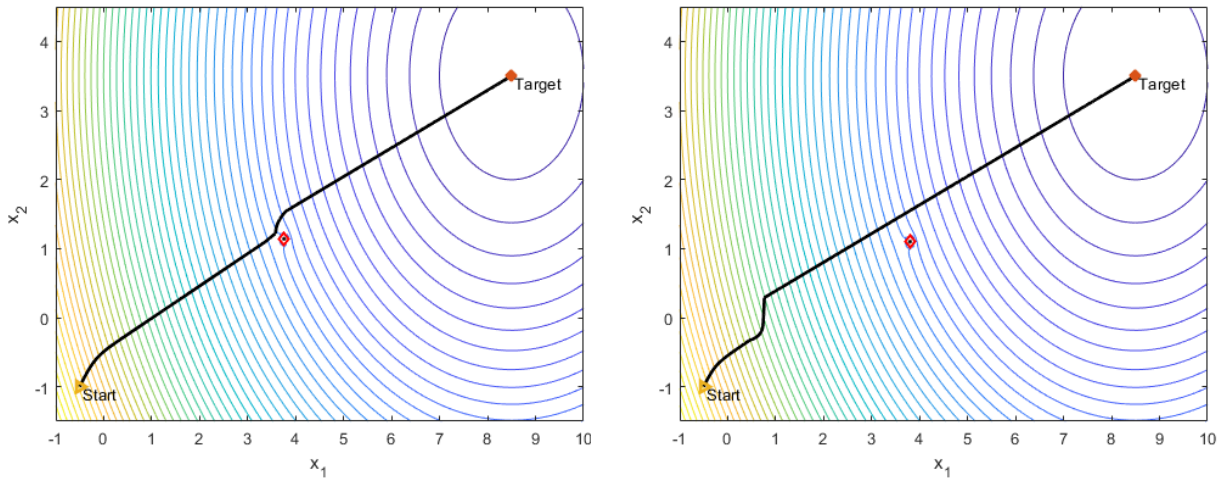
(a) DES 8^{th}-Order Obstacle Avoidance (b) DES 8^{th}-Order Extremum Trajectory

Figure 5.18: DES 8^{th}-Order Function Results

5.2.3.1 DES Perturbation

As expected, we can see from Figures 5.19 that the algorithm optimization results hold even under the presence of a perturbation. The upper limit varies depending on the

obstacle's locations and the value of γ. For Figure we found an upper limit at $\epsilon = 86.1$, this is a far larger ϵ value compared to the previous numerical results which is due to the data-driven nature of the controller, which takes into account the added adversarial signal.



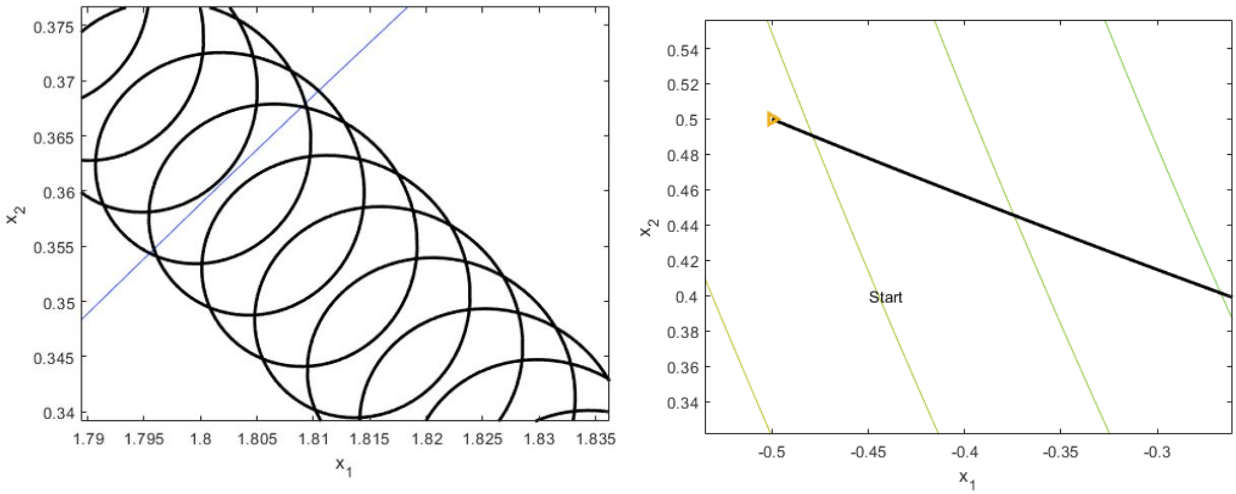
(a) Convergence without an Adversarial Signal (b) Convergence with an added Adversarial Signal

Figure 5.19: DES Controller Single Obstacle Avoidance Adversarial Signal Convergence Comparison

5.2.4 Controller Trajectory Shape Comparison

As previously mentioned, the three Real-time Optimization (RTO) algorithms have different trajectory shapes. This is due to the PE condition and the dynamic model used. Figure 5.20 and Figure 5.21 shows the three shapes exhibit by the controllers. Note, that the nonholonomic unicycle shape varies as the tunable constant variables are modified.

Note that the trajectory shape of Figure 5.21 is similar to the one shown in [86], where both are *triangular-like* shapes even though the variable parameters are different.



(a) Point-mass Dynamics Trajectory Shape

(b) DES Controller Trajectory Shape

Figure 5.20: Model-free Controller Trajectory Shape

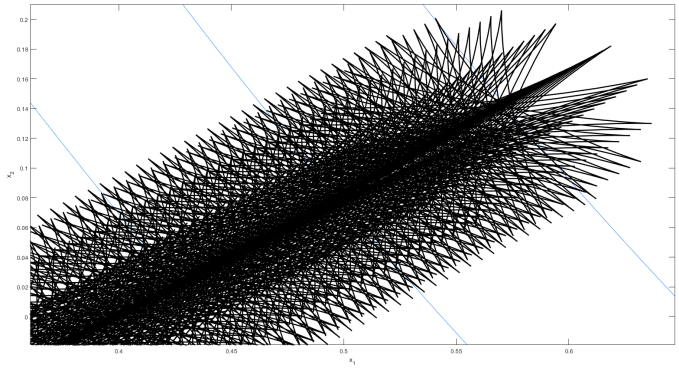


Figure 5.21: Nonholonomic Dynamics Trajectory Shape

Refer to A.2 for additional numerical simulation results, which illustrate path adjustments under the increasing presence of unknown obstacles in the state-space.

Chapter 6

Conclusion and Open Questions

6.1 Conclusion

Starting with the fundamental concepts of Real-time Optimization using Extremum Seeking, Hybrid Dynamical Systems, and Data-Enabled Extremum Seeking this thesis presented a numerical study that showed that the results from [60] can be extended to multiple obstacles maintaining the same hybrid model-free behavior, furthermore the results were also extended to nonholonomic unicycle dynamics where we saw how the trajectory of the vehicle was affected as the orientation cannot be stabilized; hence, the trajectory keeps rotating. Additionally, we implemented a Hybrid Data-Enabled Extremum Seeking (DES) algorithm to obtain the same previous results as with the Model-free Adaptive Control algorithms, but now with no persistence of excitation (PE) condition. This was clear in the numerical studies where the trajectory was a clean line with no oscillations, which are noticeable in controllers with PE. Lastly, we showed that the DES controller can use higher-order parametric approximations of the potential field not just quadratic, with the disadvantage that the computation time increases.

The results were all presented numerically with theoretical foundations. Both in the presence of no adversarial signals and in the presence of small adversarial signals. The simulations showed how similar yet different the three algorithms are, which have similar Hybrid Dynamical System frameworks but significantly different trajectory paths. Additionally, the jumps were also plotted to understand how the vehicle evaded obstacles.

6.2 Open Questions

An important question is could these algorithms achieve similar results experimentally using a robot, like the turtle bot? Based on the experimental results of [61] there is no apparent reason why these algorithms would not physically work, however it still must be tested.

Another interesting question, is how would the algorithms behave with dynamical obstacles [98]? We cannot always assume the obstacles will remain static, hence extending this work to dynamical obstacles would be extremely interesting.

Furthermore, as shown when using higher-order polynomial functions with the DES controller, the computational power required increased significantly, which consequently increased the time to converge by a factor of approximately five (from the quadratic polynomial to the 8^{th}-Order). Hence, it is logical to ask if these higher-order polynomials could benefit from momentum or acceleration [56, 57, 64, 65], and if so, how much?

Additionally, extending the dynamic results to 3-D dynamics [16, 17] would be interesting to be a step closer to full physical 3-D vehicle dynamics with aerospace and defense applications. Lastly, the work presented was all deterministic, it would be interesting to recreate this work following an stochastic approach [48, 70] to compare both, the deterministic and stochastic hybrid controller and perhaps then implement a controller with a combined deterministic and stochastic approach.

Bibliography

[1] Veronica Adetola and Martin Guay. Integration of real-time optimization and model predictive control. Journal of Process Control, 20(2):125–133, 2010.

[2] X. Ai, K. You, and S. Song. A source-seeking strategy for an autonomous underwater vehicle via online field estimation. In 2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), pages 1–6, 2016.

[3] Andersen Ang. Projected gradient algorithm. Mathematique et recherche operationnelle UMONS, 2019.

[4] Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. arXiv preprint arXiv:2005.00817, 2020.

[5] Dennis Barrios Aranibar and Pablo Javier Alsina. Reinforcement learning-based path planning for autonomous robots. In EnRI-XXIV Congresso da Sociedade Brasileira de Computaç ao, volume 10, 2004.

[6] Kartik B. Ariyur and Miroslav Krstić. Real Time Optimization by Extremum Seeking Control. John Wiley & Sons, Inc., USA, 2003.

[7] Job Bello, Peter Eriksen, and Pawel Pocwiardowski. Oil leak detections with a combined telescopic fluorescence sensor and a wide band multibeam sonar. In International Oil Spill Conference Proceedings, volume 2017, pages 1559–1573. International Oil Spill Conference, 2017.

[8] Ana Belluscio. What lies beneath antarctic ice. Nature, 2010.

[9] Howard C Berg and Edward M Purcell. Physics of chemoreception. Biophysical journal, 20(2):193–219, 1977.

[10] Jonny Beyer, Hilde C. Trannum, Torgeir Bakke, Peter V. Hodson, and Tracy K. Collier. Environmental effects of the deepwater horizon oil spill: A review. Marine Pollution Bulletin, 110(1):28 – 51, 2016.

[11] Sol Boucher. Obstacle detection and avoidance using turtlebot platform and xbox kinect. Department of Computer Science, Rochester Institute of Technology, 56, 2012.

[12] Lara Briñón-Arranz, Luca Schenato, and Alexandre Seuret. Distributed source seeking via a circular formation of agents under communication constraints. IEEE Transactions on Control of Network Systems, 3(2):104–115, 2015.

[13] Hunter C Brown and Jesse Rodocker. Cooperative underwater survey with an l3harris iver uuv and strategic robotic systems fusion rov. In Oceans 2019 MTS/IEEE Seattle, pages 1–5. IEEE, 2019.

[14] Jason Zisheng Chang. Training neural networks to pilot autonomous vehicles: Scaled self-driving car. 2018.

[15] Girish Chowdhary and Eric Johnson. Concurrent learning for convergence in adaptive control without persistency of excitation. In 49th IEEE Conference on Decision and Control (CDC), pages 3674–3679. IEEE, 2010.

[16] Jennie Cochran, Nima Ghods, and Miroslav Krstic. 3d nonholonomic source seeking without position measurement. In 2008 American Control Conference, pages 3518–3523. IEEE, 2008.

[17] Jennie Cochran, Antranik Siranosian, Nima Ghods, and Miroslav Krstic. Gps denied source seeking for underactuated autonomous vehicles in 3d. In 2008 IEEE International Conference on Robotics and Automation, pages 2228–2233. IEEE, 2008.

[18] Richard M Dudley. Real analysis and probability. CRC Press, 2018.

[19] Hans-Bernd Dürr, Miloš S Stanković, Dimos V Dimarogonas, Christian Ebenbauer, and Karl Henrik Johansson. Obstacle avoidance for an extremum seeking system using a navigation function. In 2013 American Control Conference, pages 4062–4067. IEEE, 2013.

[20] Alberto Elfes, Samuel S Bueno, Marcel Bergerman, Josué Ramos Jr, Sérgio Bittencourt Varella Gomes, et al. Project aurora: development of an autonomous unmanned remote monitoring robotic airship. Journal of the Brazilian Computer Society, 4(3), 1998.

[21] Francisco Facchinei and Jong-Shi Pang. Finite-dimensional variational inequalities and complementarity problems. Springer Science & Business Media, 2007.

[22] Chuang Gan, Yiwei Zhang, Jiajun Wu, Boqing Gong, and Joshua B Tenenbaum. Look, listen, and act: Towards audio-visual embodied navigation. arXiv preprint arXiv:1912.11684, 2019.

[23] Xing-Bao Gao. Exponential stability of globally projected dynamic systems. IEEE Transactions on Neural Networks, 14(2):426–431, 2003.

[24] Nima Ghods. Extremum seeking for mobile robots. PhD thesis, UC San Diego, 2011.

[25] Nima Ghods and Miroslav Krstić. Extremum seeking with very slow or drifting sensors. In 2009 American Control Conference, pages 1946–1951. IEEE, 2009.

[26] R Goebel, RG Sanfelice, and AR Teel. Hybrid dynamical systems: Hybrid dynamical systems: Modeling, stability, and robustness, 2012.

[27] Rafal Goebel, Joao Hespanha, Andrew R Teel, Chaohong Cai, and Ricardo Sanfelice. Hybrid systems: generalized solutions and robust stability. IFAC Proceedings Volumes, 37(13):1–12, 2004.

[28] Rafal Goebel, Ricardo G Sanfelice, and Andrew R Teel. Hybrid dynamical systems. IEEE control systems magazine, 29(2):28–93, 2009.

[29] Anil K Gupta, Ken G Smith, and Christina E Shalley. The interplay between exploration and exploitation. Academy of management journal, 49(4):693–706, 2006.

[30] H. Hajieghrary, A. F. Tomás, and M. A. Hsieh. An information theoretic source seeking strategy for plume tracking in 3d turbulent fields. In 2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pages 1–8, 2015.

[31] MAM Haring. Extremum-seeking control for steady-state performance optimization of nonlinear plants with periodic steadystate outputs. Technical Report DC2011. 029, Eindhoven University of Technology, 2011.

[32] SAE international. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. SAE International,(J3016), 2016.

[33] Petros Ioannou. Robust adaptive control. In American control conference, number 21, pages 1574–1578, 1984.

[34] X. Jin and A. Ray. Coverage control of autonomous vehicles for oil spill cleaning in dynamic and uncertain environments. pages 2594–2599, 2013.

[35] Il-Kyun Jung, Ki-Bum Hong, Suk-Kyo Hong, and Soon Chan Hong. Path planning of mobile robot using neural network. In ISIE'99. Proceedings of the IEEE International Symposium on Industrial Electronics (Cat. No. 99TH8465), volume 3, pages 979–983. IEEE, 1999.

[36] Rushikesh Kamalapurkar, Justin R Klotz, and Warren E Dixon. Concurrent learning-based approximate feedback-nash equilibrium solution of n-player nonzero-sum differential games. IEEE/CAA journal of Automatica Sinica, 1(3):239–247, 2014.

[37] Ashraf A Kassim and BVK Vijaya Kumar. Path planning for autonomous robots using neural networks. Journal of Intelligent Systems, 7(1-2):33–56, 1997.

[38] VV Kazakevich. Technique of automatic control of different processes to maximum or to minimum. Avtorskoe svidetelstvo,(USSR Patent), (66335), 1943.

[39] VV Kazakevich. On extremum seeking. Moscow High Technical University, 1944.

[40] Hassan K Khalil. Adaptive output feedback control of nonlinear systems represented by input-output models. IEEE Transactions on Automatic Control, 41(2):177–188, 1996.

[41] Hassan K Khalil and Jessy W Grizzle. Nonlinear systems, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

[42] Sei Zhen Khong, Ying Tan, Chris Manzie, and Dragan Nešić. Multi-agent source seeking via discrete-time extremum seeking control. Automatica, 50(9):2312–2320, 2014.

[43] Miroslav Krstić and Hsin-Hsiung Wang. Stability of extremum seeking feedback for general nonlinear dynamic systems. Automatica, 36(4):595–601, 2000.

[44] Hung Manh La and Weihua Sheng. Distributed sensor fusion for scalar field mapping using mobile sensor networks. IEEE Transactions on cybernetics, 43(2):766–778, 2013.

[45] Maurice Leblanc. Sur l'electrification des chemins de fer au moyen de courants alternatifs de frequence elevee. Revue générale de l'électricité, 12(8):275–277, 1922.

[46] Shuai Li, Ruofan Kong, and Yi Guo. Cooperative distributed source seeking by multiple robots: Algorithms and experiments. IEEE/ASME Transactions on mechatronics, 19(6):1810–1820, 2014.

[47] Wei Li, Jay A Farrell, Shuo Pang, and Richard M Arrieta. Moth-inspired chemical plume tracing on an autonomous underwater vehicle. IEEE Transactions on Robotics, 22(2):292–307, 2006.

[48] Shu-Jun Liu and Miroslav Krstic. Stochastic source seeking for nonholonomic unicycle. Automatica, 46(9):1443–1453, 2010.

[49] Shuang Liu, Dong Sun, and Changan Zhu. Coordinated motion planning for multiple mobile robots along designed paths with formation requirement. IEEE/ASME transactions on mechatronics, 16(6):1021–1031, 2010.

[50] John C Luxat and Leonard H Lees. Stability of peak-holding control systems. IEEE Transactions on Industrial Electronics and Control Instrumentation, (1):11–15, 1971.

[51] F. Mandić, N. Mišković, and I. Lončar. Underwater acoustic source seeking using time-difference-of-arrival measurements. IEEE Journal of Oceanic Engineering, 45(3):759–771, 2020.

[52] SM Meerkov. Asymptotic methods for investigating a class of forced states in extremal systems. Automation and Remote Control, 28(12):1916–1920, 1967.

[53] H. Modares, F. L. Lewis, and M. Naghibi-Sistani. Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks. IEEE Transactions on Neural Networks and Learning Systems, 24(10):1513–1525, 2013.

[54] V. Muthukumaran, R. G. Sanfelice, and G. H. Elkaim. A hybrid control strategy for autonomous navigation while avoiding multiple obstacles at unknown locations. In 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), pages 1042–1047, 2019.

[55] Dragan Nešić. Extremum seeking control: Convergence analysis. European Journal of Control, 15(3-4):331–347, 2009.

[56] Daniel E Ochoa, Jorge I Poveda, César A Uribe, and Nicanor Quijano. Hybrid robust optimal resource allocation with momentum. In 2019 IEEE 58th Conference on Decision and Control (CDC), pages 3954–3959. IEEE, 2019.

[57] Daniel E Ochoa, Jorge I Poveda, César A Uribe, and Nicanor Quijano. Robust optimization over networks using distributed restarting of accelerated dynamics. IEEE Control Systems Letters, 5(1):301–306, 2020.

[58] David H Owens and Jari Hätönen. Iterative learning control—an optimization paradigm. Annual reviews in control, 29(1):57–70, 2005.

[59] Dobrivoje Popovic. Topics in extremum seeking. PhD thesis, University of California, Santa Barbara, 2004.

[60] J. I. Poveda, M. Benosman, A. R. Teel, and R. G. Sanfelice. A hybrid adaptive feedback law for robust obstacle avoidance and coordination in multiple vehicle systems. In 2018 Annual American Control Conference (ACC), pages 616–621, 2018.

[61] J.I. Poveda, M. Benosman, A. R. Teel, and R.G. Sanfelice. Robust coordinated hybrid source seeking with obstacle avoidance in multi-vehicle autonomous systems,. submitted, 2019.

[62] J.I. Poveda, M. Benosman, and K. Vamvoudakis. Data-enabled extremum seeking: A cooperative concurrent learning-based approach,. under review, 2020.

[63] JI Poveda, K Vamvoudakis, and M Benosman. Dees: A class of data-enabled robust feedback algorithms for real-time optimization. IFAC-PapersOnLine, 52(16):670–675, 2019.

[64] Jorge I Poveda and Na Li. Inducing uniform asymptotic stability in non-autonomous accelerated optimization dynamics via hybrid regularization. In 2019 IEEE 58th Conference on Decision and Control (CDC), pages 3000–3005. IEEE, 2019.

[65] Jorge I Poveda and Na Li. Robust hybrid zero-order optimization algorithms with acceleration via averaging in continuous time. arXiv preprint arXiv:1909.00265, 2019.

[66] Jorge I Poveda, Kyriakos G Vamvoudakis, and Mouhacine Benosman. Codes: Cooperative data-enabled extremum seeking for multi-agent systems. In 2019 IEEE 58th Conference on Decision and Control (CDC), pages 2988–2993. IEEE, 2019.

[67] Christophe Prieur, Andrew R Teel, and Luca Zaccarian. Relaxed persistent flow/jump conditions for uniform global asymptotic stability. IEEE Transactions on Automatic Control, 59(10):2766–2771, 2014.

[68] Yao Qin, Nicholas Frosst, Sara Sabour, Colin Raffel, Garrison Cottrell, and Geoffrey Hinton. Detecting and diagnosing adversarial images with class-conditional capsule reconstructions. arXiv preprint arXiv:1907.02957, 2019.

[69] Eduardo Ramírez-Llanos and Sonia Martínez. Constrained source seeking for mobile robots via simultaneous perturbation stochastic approximation. In 2016 IEEE 55th Conference on Decision and Control (CDC), pages 6851–6856. IEEE, 2016.

[70] E. Ramírez-Llanos and S. Martínez. Stochastic source seeking for mobile robots in obstacle environments via the spsa method. IEEE Transactions on Automatic Control, 64(4):1732–1739, 2019.

[71] R Tyrrell Rockafellar and Roger J-B Wets. Variational analysis, volume 317. Springer Science & Business Media, 2009.

[72] Robin Andrew Russell. Locating underground chemical sources by tracking chemical gradients in 3 dimensions. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), volume 1, pages 325–330. IEEE, 2004.

[73] Carolyn Said. Kiwibots win fans at uc berkeley as they deliver fast food at slow speeds. San Fransisco Chronicle, 2019.

[74] Jan A Sanders, Ferdinand Verhulst, and James Murdock. Averaging methods in nonlinear dynamical systems, volume 59. Springer, 2007.

[75] Ricardo Sanfelice. Hybrid equations toolbox v2.04. MATLAB Central File Exchange, 2020.

[76] Ricardo G Sanfelice. Robust hybrid control systems. University of California, Santa Barbara, 2007.

[77] Ricardo G Sanfelice, Rafal Goebel, and Andrew R Teel. Generalized solutions to hybrid dynamical systems. ESAIM: Control, Optimisation and Calculus of Variations, 14(4):699–724, 2008.

[78] Alexander Scheinker. Extremum seeking for stabilization. PhD thesis, UC San Diego, 2012.

[79] George AF Seber and Alan J Lee. Linear regression analysis, volume 329. John Wiley & Sons, 2012.

[80] James C Spall. Implementation of the simultaneous perturbation algorithm for stochastic optimization. IEEE Transactions on aerospace and electronic systems, 34(3):817–823, 1998.

[81] James C Spall et al. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE transactions on automatic control, 37(3):332–341, 1992.

[82] Anthony Spears, Michael West, Matthew Meister, Jacob Buffo, Catherine Walker, Thomas Riley Collins, Ayanna Howard, and Britney Schmidt. Under ice in antarctica: The icefin unmanned underwater vehicle development and deployment. IEEE Robotics & Automation Magazine, 23(4):30–41, 2016.

[83] Shayan Taheri, Milad Salem, and Jiann-Shiun Yuan. Razornet: Adversarial training and noise training on a deep neural network fooled by a shallow neural network. Big Data and Cognitive Computing, 3(3):43, 2019.

[84] Ying Tan, William H Moase, Chris Manzie, Dragan Nešić, and IMY Mareels. Extremum seeking from 1922 to 2010. In Proceedings of the 29th Chinese control conference, pages 14–26. IEEE, 2010.

[85] Ying Tan, Dragan Nešić, and Iven Mareels. On non-local stability properties of extremum seeking control. Automatica, 42(6):889–903, 2006.

[86] Flip Tanedo. Notes on non-holonomic constraints. P3318: Analytical Mechanics, 2013.

[87] Andrew R Teel, Joan Peuteman, and Dirk Aeyels. Semi-global practical asymptotic stability and averaging. Systems & control letters, 37(5):329–334, 1999.

[88] Avi Turgeman and Herbert Werner. Cooperative 3-d source seeking using a swarm of auvs with application to a numerical oil spill model. 2017.

[89] Kyriakos G Vamvoudakis, Panos J Antsaklis, Warren E Dixon, João P Hespanha, Frank L Lewis, Hamidreza Modares, and Bahare Kiumarsi. Autonomy and machine intelligence in complex systems: A tutorial. In 2015 American Control Conference (ACC), pages 5062–5079. IEEE, 2015.

[90] Kyriakos G Vamvoudakis and Frank L Lewis. Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. Automatica, 46(5):878–888, 2010.

[91] Massimo Vergassola, Emmanuel Villermaux, and Boris I Shraiman. 'infotaxis' as a strategy for searching without gradients. Nature, 445(7126):406–409, 2007.

[92] Xiaohua Wang, Vivek Yadav, and SN Balakrishnan. Cooperative uav formation flying with obstacle/collision avoidance. IEEE Transactions on control systems technology, 15(4):672–679, 2007.

[93] Yue Wu, Hui Wang, Biaobiao Zhang, and K-L Du. Using radial basis function networks for function approximation and classification. ISRN Applied Mathematics, 2012, 2012.

[94] YS Xia and J Wang. On the stability of globally projected dynamical systems. Journal of Optimization Theory and Applications, 106(1):129–150, 2000.

[95] Chunlei Zhang, Daniel Arnold, Nima Ghods, Antranik Siranosian, and Miroslav Krstić. Source seeking with non-holonomic unicycle without position measurement and with tuning of forward velocity. Systems & control letters, 56(3):245–252, 2007.

[96] Chunlei Zhang and Raúl Ordóñez. Extremum-seeking control and applications: a numerical optimization-based approach. Springer Science & Business Media, 2011.

[97] Chunlei Zhang, Antranik Siranosian, and Miroslav Krstić. Extremum seeking for moderately unstable systems and for autonomous vehicle target tracking without position measurements. Automatica, 43(10):1832–1839, 2007.

[98] Rui Zou, Vijay Kalivarapu, Eliot Winer, James Oliver, and Sourabh Bhattacharya. Particle swarm optimization-based source seeking. IEEE Transactions on Automation Science and Engineering, 12(3):865–875, 2015.

Appendix A

Variable Values and Supplementary Numerical Results

A.1 Simulation Variable Values and Descriptions

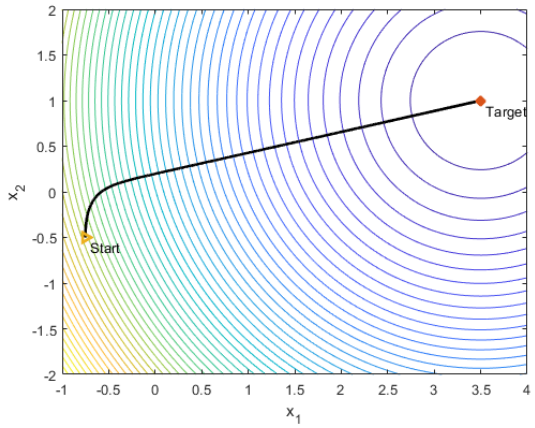
Name	*Value*	*Description*
r_1 & r_2	(0, 1.75)	Array of Obstacle(s) Location Values
x_t	(3, 0)	Target Location (extremum point, $[x^*, y^*]$
x_1, x_2	(-0.5, 0.5)	Vehicle's Starting
q	1	Starting Switching Logic
θ	0.1	Orientation Initial Value (only used in 5.2.2)
p	0	Initial Value of Obstacles Detected
c	0.1	Vehicle's Sensor Detection Radius
ξ_x, ξ_y	(0,0)	Fast Dynamics Initial Conditions (used in 5.2.1 & 5.2.2)
μ_1, mu_2)	(0,0)	Dither Signal Initial Value (used in 5.2.1 & 5.2.2)
num_{obs}	2	Number of Existing Obstacles
b_o	0.1	Initial Obstacle(s) Ball Radius
c_o	0.1	Initial Vehicle's Sensor Radius
η	0.9	Radius c Reduction Variable
ρ	1	Barrier Function Tuning Variable
ω	1000	Dither Signal Frequency

$\bar{\omega}$	1	Amplitude of Fast Dynamics (used in 5.2.1 & 5.2.2)
a	0.01	Dynamics Amplitude (used in 5.2.1 & 5.2.2)
k	0.002	Fast Dynamics Amplitude Coefficient (used in 5.2.1 & 5.2.2)
ϵ	0.15	Perturbation Size
T_s	0.1	Level Set Thickness
α	0.4	Nonholonomic Amplitude (only used in 5.2.2)
k_1	8	Nonholonomic Frequency Reduction Factor (only used in 5.2.2)
γ	20	DES Amplitude Constant (only used in 5.2.3)
w	0	Matrix of Weights (only used in 5.2.3)

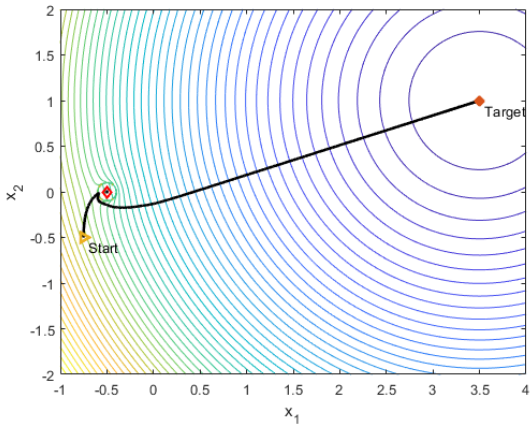
Table A.1: Constant Variables Typical Simulation Values

A.2 Supplementary Numerical Results

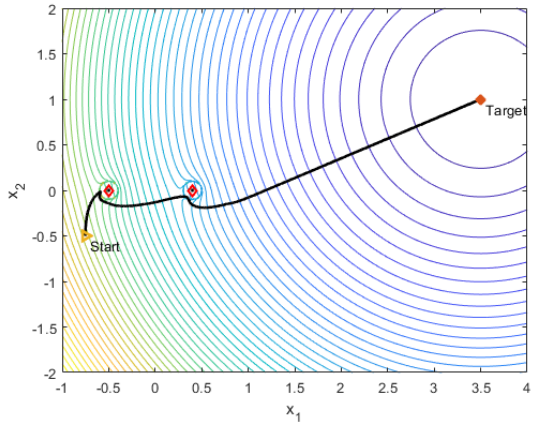
This section illustrates the path adjustments the algorithms perform as the number of obstacles increases. The numerical results presented showcase how the algorithms behave starting from no obstacles present to numerous (more than six) obstacles in the state-space. Where Figure A.1 shows the trajectory results for the point-mass dynamics of the hybrid adaptive controller, Figure A.2 illustrates the results for the holonomic unicycle dynamics, and Figure A.3 corresponds to the DES controller using a quadratic map function.



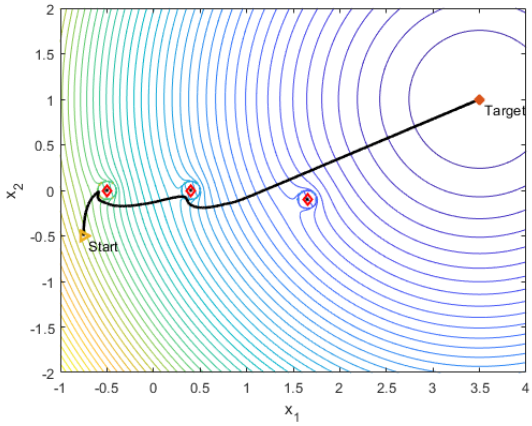
(a) No Obstacles Present



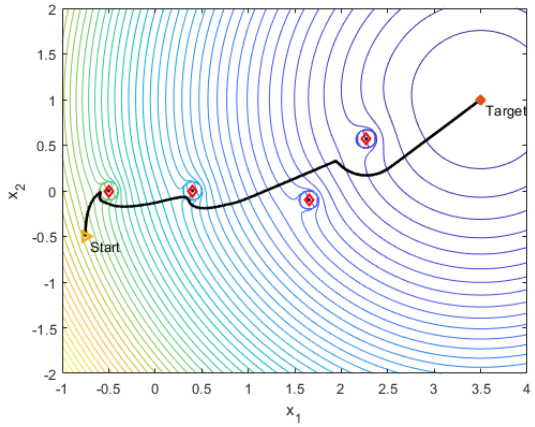
(b) Single Obstacles Present



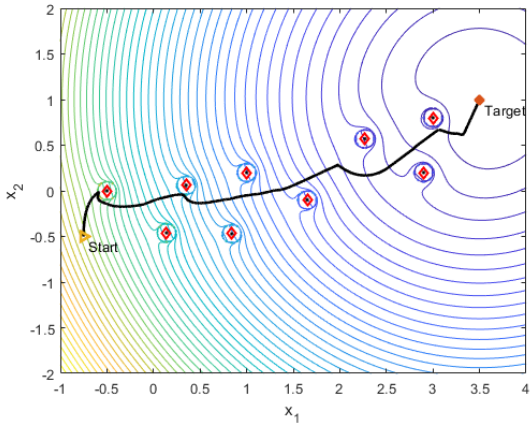
(c) Two Obstacles Present



(d) Three Obstacles Present

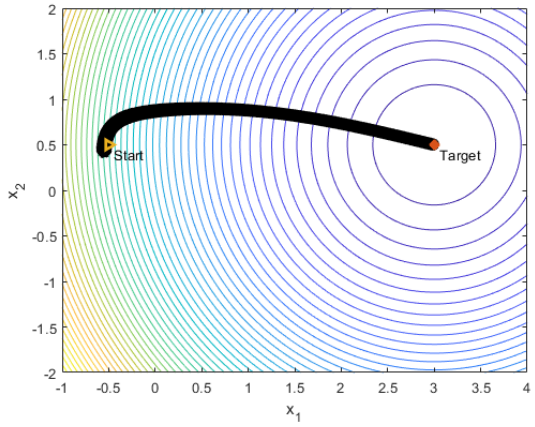


(e) Four Obstacles Present

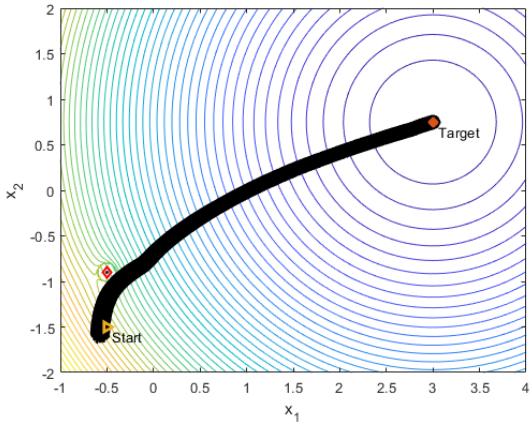


(f) Multiple Obstacles Present

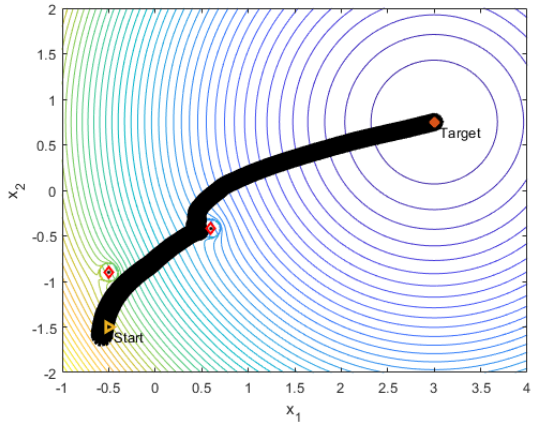
Figure A.1: Point-mass Dynamics Hybrid Adaptive Law Path Variation under a perturbation, $\epsilon = 0.01$



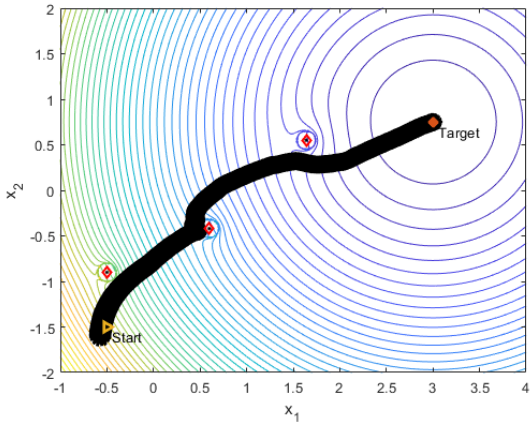
(a) No Obstacles Present



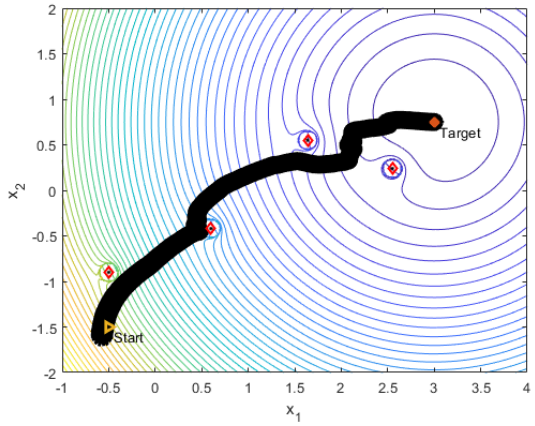
(b) Single Obstacles Present



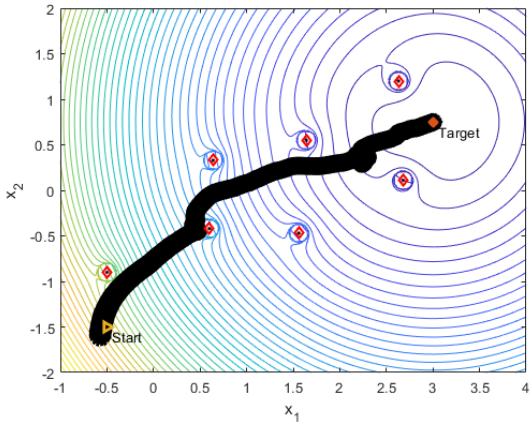
(c) Two Obstacles Present



(d) Three Obstacles Present

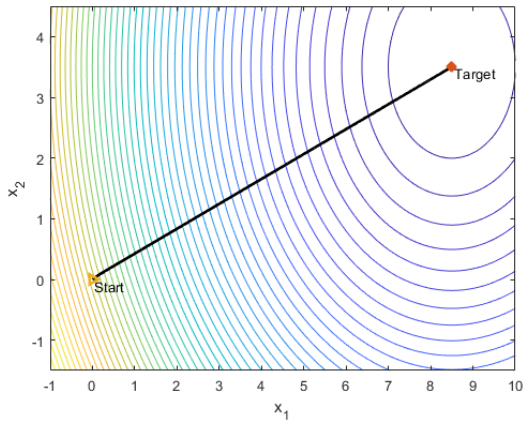


(e) Four Obstacles Present

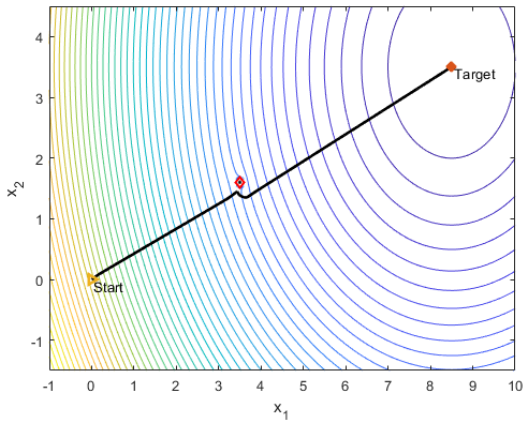


(f) Multiple Obstacles Present

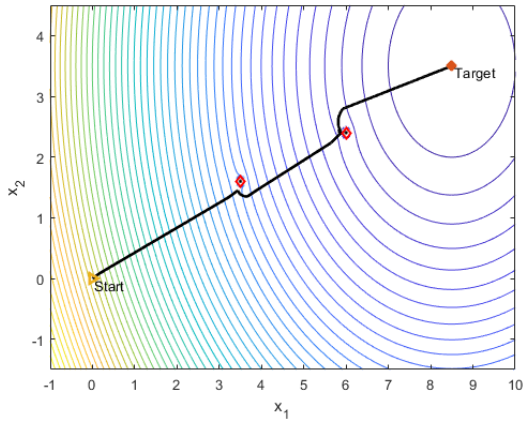
Figure A.2: Nonholonomic Unicycle Dynamics Hybrid Adaptive Law Path Variation



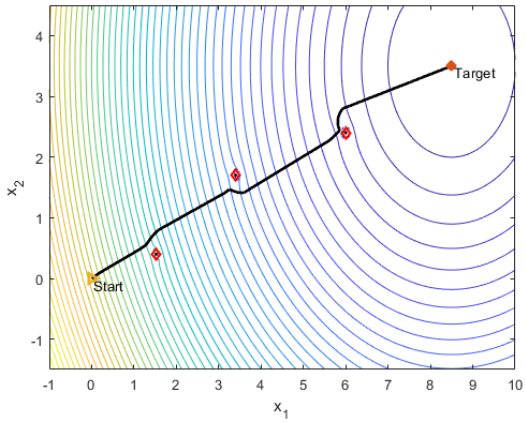
(a) No Obstacles Present



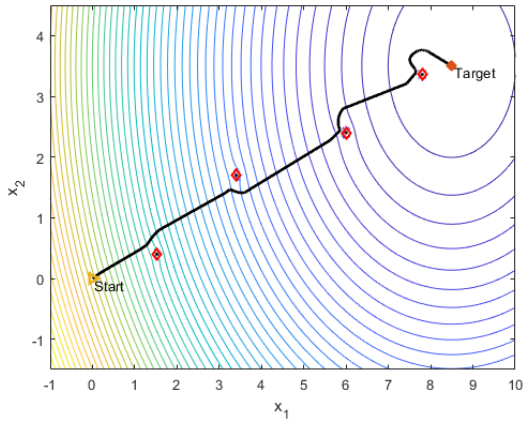
(b) Single Obstacles Present



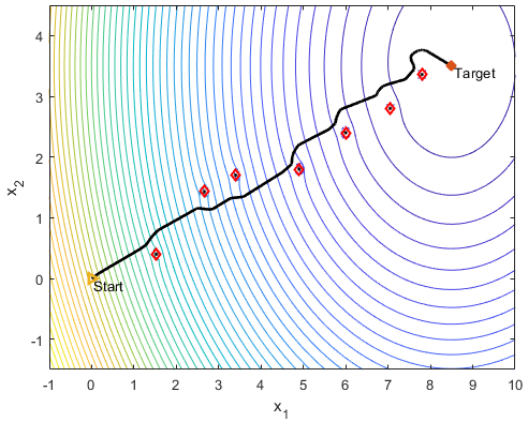
(c) Two Obstacles Present



(d) Three Obstacles Present



(e) Four Obstacles Present



(f) Multiple Obstacles Present

Figure A.3: DES Controller Path Variation