University of Cincinnati

Date: 7/2/2024

I, Shivam Bajpai, hereby submit this original work as part of the requirements for the degree of Master of Science in Aerospace Engineering.

It is entitled:

Investigating the Performance of Different Controllers in Optimized Path Tracking in Robotics: A Lie Bracket System and Extremum Seeking Approach.

Student's name: **Shivam Bajpai**

This work and its defense approved by:

Committee chair: Sameh Eisa, Ph.D.

Committee member: Abhinav Sinha, Ph.D.

Committee member: Shaaban Abdallah, Ph.D.

48204

Investigating the Performance of Different Controllers in Optimized Path Tracking in Robotics: A Lie Bracket System and Extremum Seeking Approach

A thesis submitted to the
Graduate School
of the University of Cincinnati
in partial fulfillment of the
requirements for the degree of

Master of Science
in the Department of Aerospace Engineering and Engineering Mechanics
of the College of Engineering and Applied Sciences

by

Shivam Bajpai
University of Cincinnati

Committee Chair and Advisor:
Assistant Prof. Sameh A. Eisa
Committee Members:
Prof. Shaaban A Abdallah, Assistant Prof. Abhinav Sinha

UNIVERSITY OF CINCINNATI

Abstract

Department of Aerospace Engineering and Engineering Mechanics
College of Engineering and Applied Sciences

Master of Science

by Shivam Bajpai

Autonomous vehicles are a hot topic in control theory and are utilised in different fields such as industries, aerospace and robotics. Trajectory-tracking is one of the crucial features of autonomous vehicles which involves two major steps: generating a reference trajectory and tracking it. In many cases, the reference trajectory is generated by using an objective function where we want vehicles to move towards the extremum (maximum/minimum) of the objective function which is a terminal position of the trajectory. We performed some simulations and experiments using traditional controllers including the Proportional-Derivative Controller (PDC), Model-Predictive Controller (MPC), and Pure Pursuit Controller (PPC). These controllers while showing some degree of desirable vs. undesirable behaviour, they are model-based controllers that require a mathematical expression of an objective function. Additionally, they show difficulties and they have other limitations. In this thesis, we make a case for the utilization of extremum-seeking control (ESC) systems which are model-free, real-time, adaptive systems. We revisit some works that have been done regarding the classic ESC (C-ESC) structure. The primary part of this thesis is where we provide the simulations and experimental works using control-affine ESC (CA-ESC) systems which have been used rarely in experimental environments in literature. Particularly, we utilized single-integrator and unicycle dynamics CA-ESC structures and conducted simulations and experiments. Additionally, we propose a novel amended CA-ESC design (for both single-integrator and unicycle dynamics) by adopting some developments that took place in this area in recent years. Our proposed design consists of a Geometric-Based Extended Kalman Filter (GEKF) for gradient estimation and adaptation law for attenuation oscillations and better convergence rate. We performed simulations to show the effectiveness of our proposed design.

"Research means that you don't know, but are willing to find out"

- Charles F. Kettering

Acknowledgements

When I started my MS, my journey was different and challenging for me since before this I did my undergraduate from India. I struggled a lot during my first semester as I was not familiar with software like MATLAB and the programming language Python, and the other students were efficient in them. I often questioned myself whether coming to the USA for my MS was the right decision or not. However, from the second semester onward, everything started to improve and I began to enjoy my MS journey. In this regard, I would like to thank the people who have supported me throughout the journey.

I am grateful to my family for their immense support emotionally and financially without which I would not completed my journey. I am thankful to my mother Mrs Geeta Bajpai, and my father Mr Rajendra Narayan Bajpai my elder sisters, and their husbands. I know how hard it was for them to send me here far away from them, and motivating me each day. I am indebted to them for all their support and motivation.

I express my special gratitude to my advisor Dr. Sameh A. Eisa for giving me the opportunity to work with him. I am thankful for his support, guidance, and valuable time throughout my journey. I still remember in the first semester when I informed him that I didn't have sufficient skills in MATLAB, and Python, then he gave me time to improve and provided materials to help me excel.

I am thankful to my graduate school at the University of Cincinnati for providing an amazing study and work environment. A special thanks to my committee members- Dr. Shaaban A Abdallah and Dr. Abhinav Sinha for their valuable comments and feedback.

I would like to thank my group members Sameer, Ahmed, Hesham, and Benjamin at the MDCL lab. It was a wonderful experience to work together filled with a lot of fun. I am thankful to each one of them for their valuable input to my research work, which had a significant impact on my work.

I would like to give credit to Ahmed for editing all the experimental videos.

I would also like to express my gratitude to Rohith and Matthew for sharing their experimental and theoretical knowledge and experience and for helping us in setting up the lab. I am thankful to undergraduate students (Alex, Ariana and Paulo) for their help in conducting experiments. Lastly, I am thankful to all my roommates, friends, and family who are here and in India for all their help, guidance and support at each step of my MS journey. I am grateful to all of them for being the valuable part of my part.

Contents

Abstract i

Acknowledgements iv

List of Figures viii

List of Tables xiii

1 Introduction 1
- 1.1 Autonomous Vehicles . . . 1
- 1.2 Motivation . . . 2
- 1.3 Literature Review . . . 5
  - 1.3.1 Proportional – Integral – Derivative Controller (PIDC) . . . 5
  - 1.3.2 Pure Pursuit Controller (PPC) . . . 6
  - 1.3.3 Model Predictive Controller (MPC) . . . 7
- 1.4 Thesis Contribution . . . 8
- 1.5 Thesis Layout . . . 9

2 Model-based Solutions with Simulations and Experiments 10
- 2.1 Introduction . . . 10
- 2.2 Path Planning Under Consideration . . . 11
- 2.3 Differential-Drive Model for the Robot . . . 15
- 2.4 Controllers . . . 17
  - 2.4.1 Proportional-Integral-Derivative Controller (PIDC) . . . 17
  - 2.4.2 Model Predictive Controller (MPC) . . . 18
  - 2.4.3 Pure Pursuit Controller (PPC) . . . 21
- 2.5 Experimental Setup . . . 23
  - 2.5.1 Hardware . . . 24
    - 2.5.1.1 Turtlebot3 (TB3) . . . 24
    - 2.5.1.2 Motion Capturing System (MCS) . . . 24
  - 2.5.2 Software . . . 25
    - 2.5.2.1 MATLAB . . . 25
    - 2.5.2.2 Motive . . . 25
- 2.6 Results . . . 26
  - 2.6.1 Simulations . . . 26

2.6.1.1 Proportional-Derivative Controller (PDC) . . . 27
2.6.1.2 Pure Pursuit Controller (PPC) . . . 29
2.6.1.3 Model Predictive Controller (MPC) . . . 31
2.6.2 Experiments . . . 33
2.6.2.1 Pure Pursuit Controller (PPC) . . . 33
2.6.2.2 Model Predictive Controller (MPC) . . . 35
2.7 Conclusion . . . 37

3 Revisiting Model-Free Source Seeking in Robots via Classical Extremum Seeking 38
3.1 Introduction . . . 38
3.2 Extremum Seeking Control (ESC) System . . . 39
3.2.1 Classical Extremum-Seeking Control (C-ESC) . . . 41
3.3 Results . . . 44
3.3.1 Simulation Results . . . 44
3.3.1.1 1-D Scheme . . . 44
3.3.1.2 2-D Navigation . . . 48
3.3.2 Experimental Results . . . 52
3.3.2.1 With known Objective Function . . . 52
3.3.2.2 With unknown Objective Function (Light Source-Seeking) 54
3.4 Conclusion . . . 56

4 Model-free Source Seeking by a Robot: The Case of Control Affine-ESC (CA-ESC) 57
4.1 Introduction . . . 57
4.2 Control Affine Extremum-Seeking Control (CA-ESC) . . . 58
4.3 Results . . . 60
4.3.1 Unicycle Dynamics . . . 60
4.3.1.1 Simulation Results . . . 61
4.3.1.2 Experimental Result . . . 64
4.3.1.3 With Known Objective Function . . . 64
4.3.1.4 With unknown Objective Function . . . 66
4.3.2 Single-Integrator Dynamics . . . 68
4.3.2.1 Simulation Results . . . 69
4.3.2.2 Experimental Results . . . 71
4.3.2.3 With Known Objective Function . . . 71
4.3.2.4 Light Source-Seeking with Single-Integrator Dynamics . . 73
4.4 Improving Convergence Rate and Attenuating Oscillations by Geometric-Based Extended Kalman Filter . . . 75
4.4.0.1 Chen-Fliess Functional Expansion . . . 75
4.5 Main Results . . . 76
4.5.1 Method for Attenuation of Oscillations for ESC systems . . . 77
4.5.2 Geometric-Based Extended Kalman Filter (GEKF) for gradient and Lie Bracket Estimation in Control Affine Extremum-Seeking Control (CA-ESC) Systems . . . 78
4.6 Simulation Results . . . 82
4.6.1 Unicycle Dynamics . . . 82
4.6.2 Single-Integrator Dynamics . . . 88

4.7 Conclusion 95

5 Summary and Future Work **96**
5.1 Future Work 96

Bibliography **98**

List of Figures

1.1 Depiction of trajectory tracking problem in xy plane. The black dotted is the reference trajectory and the red is the actual trajectory tracked by the differential-drive robot. . . . 2

2.1 Three different trajectories generated by LBS corresponding to the equations. The blue trajectory corresponds to map 1 (2.8), the red trajectory refers to map 2 (2.9), and the green trajectory is of (2.10). The"star" sign refers to the starting position, and the "stop" sign refers to the terminal position. . . . 15
2.2 A depiction of differential-drive robot in XY plane. . . . 16
2.3 PID control law design. . . . 18
2.4 MPC model design consists of reference path, optimizer, model, and a feedback loop of current states of the system. . . . 19
2.5 PPC. . . . 20
2.6 A detailed description of the PPC algorithm in which the green trajectory represents the reference trajectory, and the differential-drive robot is in blue. . . . 23
2.7 Modeling, Dynamics, and Control Lab (MDCL). 1. TB3 inside the testing area, 2. A light source, 3. Motion capturing system (MCS). . . . 24
2.8 A depiction of turtlebot3 (TB3) burger with its components: LIDAR, Raspberry pi 3B+, OpenCR, Dynamixel motors. . . . 25
2.9 A generalized structure of trajectory-tracking model design for simulations. 26
2.10 (a) . . . 28
2.11 (b) . . . 28
2.12 (c) . . . 28
2.13 (d) . . . 28
2.14 (e) . . . 28
2.15 Simulation results of PDC. (a) x state, (b) y state, (c) Linear velocity, (d) Angular velocity, and (e) Trajectory of the system using PDC. . . . 28
2.16 (a) . . . 30
2.17 (b) . . . 30
2.18 (c) . . . 30
2.19 (d) . . . 30
2.20 (e) . . . 30
2.21 Simulation results of PPC. (a) x state, (b) y state, (c) Linear velocity, (d) Angular velocity, and (e) Trajectory of the system using PPC. . . . 30
2.22 (a) . . . 32
2.23 (b) . . . 32
2.24 (c) . . . 32

2.25 (d) . . . 32
2.26 (e) . . . 32
2.27 Simulation results of PPC. (a) x state, (b) y state, (c) Linear velocity, (d) Angular velocity, and (e) Trajectory of the system using MPC. . . . 32
2.28 (a) . . . 34
2.29 (b) . . . 34
2.30 (c) . . . 34
2.31 (d) . . . 34
2.32 Experimental results of PPC. (a) x state, (b) y state, (c) Trajectory of the TB3 using PPC and (d) Angular velocity. . . . 34
2.33 (a) . . . 36
2.34 (b) . . . 36
2.35 (c) . . . 36
2.36 (d) . . . 36
2.37 Experimental results of MPC. (a) x state, (b) y state, (c) Trajectory of the TB3 using MPC, and (d) Angular velocity. . . . 36

3.1 Simplified ESC system diagram consisting of blocks of Perturbation signal, System, and objective function. . . . 40
3.2 Generalized representation of the classical structure of extremum seeking controller (C-ESC). This structure of ESC consists of four major steps. First is the modulation step in which a perturbation signal is added to a nominal value of the tuning parameter. In the second step, the system responds and it leads to a change in an objective function (leads to a new measurement). In the third step, the measurement of the objective function gets multiplied with a demodulation signal and in the final step, the demodulation is integrated and updates the tuning parameter. . . . 43
3.3 (a) Generalized and modified designs C-ESC. The generalized C-ESC is the top figure, and the modified C-ESC design is the bottom figure which works well with both cases: in the case of a slow sensor ($\epsilon > 0$) and the case of a pure integrator when $\epsilon = 0$. . . . 46
3.4 (a) . . . 47
3.5 (b) . . . 47
3.6 (c) . . . 47
3.7 (d) . . . 47
3.8 Simulation results of modified C-ESC with slow sensor dynamics. (a) Objective function, (b) Position estimate of the leak, (c) Sensor reading, and (d) HPF. . . . 47
3.9 ES design for 2-D navigation with slow sensor . . . 50
3.10 (a) . . . 51
3.11 (b) . . . 51
3.12 (c) . . . 51
3.13 (d) . . . 51
3.14 (e) . . . 51
3.15 (f) . . . 51
3.16 (g) . . . 51
3.17 (h) . . . 51

3.18 Simulation results of modified C-ESC for 2-D navigation with slow sensor. (a) Objective function, (b) Planar trajectory of the system (c) HPF, (d) Sensor reading, (e) x state (f) y state, and (g) Control input for x state (h) Control input for y state. 51
3.19 (a) 53
3.20 (b) 53
3.21 (e) 53
3.22 (d) 53
3.23 (e) 53
3.24 (f) 53
3.25 Experimental results of modified C-ESC for 2-D navigation for known objective function. (a) Objective function, (b) Planar trajectory of TB3, (c)x state (d) y state, (e) HPF, and (f) Angular velocity. 53
3.26 (a) 55
3.27 (b) 55
3.28 (c) 55
3.29 (d) 55
3.30 Source Seeking Experiment using a light source. 55

4.1 The generalized CA-ESC design 60
4.2 Unicycle Dynamics CA-ESC Structure. 62
4.3 (a) 63
4.4 (b) 63
4.5 (c) 63
4.6 (d) 63
4.7 (e) 63
4.8 (f) 63
4.9 Simulation results of unicycle dynamics using a known mathematical expression of the objective function to access the measurements: (a) x state, (b) y state, (c) Objective function (d) Planar trajectory of the system, (e) HPF, and (f) Linear velocity. 63
4.10 (a) 65
4.11 (b) 65
4.12 (c) 65
4.13 (d) 65
4.14 (e) 65
4.15 (f) 65
4.16 Experimental results of unicycle dynamics using a known mathematical expression of the objective function to access the measurements: (a) x state, (b) y state, (c) Objective function (d) Planar trajectory of the system, (e) HPF, and (f) Angular velocity. 65
4.17 (a) 67
4.18 (b) 67
4.19 (c) 67
4.20 Light source seeking experimental results using unicycle design. (a) x state (b) y state and (c) Trajectory of TB3. 67
4.21 Single-Integrator Dynamics CA-ESC Structure. 68
4.22 (a) 70

4.23 (b) . . . 70
4.24 (c) . . . 70
4.25 (d) . . . 70
4.26 (e) . . . 70
4.27 (f) . . . 70
4.28 (f) . . . 70
4.29 Simulation results of single-integrator dynamics using a known mathematical expression of the objective function to access the measurements: (a) x state, (b) y state, (c) Objective function (d) Planar trajectory of the system, (e) and (f) Represent the control input in the respective states and (g) HPF. . . . 70
4.30 (a) . . . 72
4.31 (b) . . . 72
4.32 (c) . . . 72
4.33 (d) . . . 72
4.34 (e) . . . 72
4.35 (f) . . . 72
4.36 Experimental results of single-integrator dynamics using a known mathematical expression of the objective function to access the measurements: (a) x state, (b) y state, (c) Objective function (d) Planar trajectory of TB3, (e) HPF, and (f) Angular velocity. . . . 72
4.37 (a) . . . 74
4.38 (b) . . . 74
4.39 (c) . . . 74
4.40 Light source seeking experimental results using single-integrator design. (a) x state (b) y state and (c) Trajectory of TB3. . . . 74
4.41 CA-ESC structure with GEKF and adaptation law. . . . 78
4.42 Proposed Design of CAESC with GEKF and Adaptation law for unicycle dynamics . . . 85
4.43 (a) . . . 87
4.44 (b) . . . 87
4.45 (c) . . . 87
4.46 (d) . . . 87
4.47 (e) . . . 87
4.48 Simulation results of our proposed design with unicycle dynamics using a known mathematical expression of the objective function in which the blue one represents the results using traditional single-integrator design found in the literature, and red one represents results using our design: (a) x state, (b) y state, (c) Objective function (d) Angular velocity and (e) Trajectory of the system. . . . 87
4.49 Proposed Design of CAESC with GEKF and Adaptation law . . . 89
4.50 (a) . . . 94
4.51 (b) . . . 94
4.52 (c) . . . 94
4.53 (d) . . . 94
4.54 (e) . . . 94
4.55 (f) . . . 94

4.56 Simulation results of our proposed design using a known mathematical expression of the objective function to access the measurements, in which the blue one represents the results using traditional single-integrator design mentioned in the literature, and red one represents results using our design: (a) x state, (b) y state, (c) Planar trajectory of the system (d) Objective function, (e) and (f) Represent the control input in the respective states. 94

List of Tables

2.1 PDC tuning parameters for simulations. 27
2.2 PPC simulation tuning parameters. 29
2.3 MPC simulation tuning parameters. 31
2.4 PPC tuning parameters (for experiment). 33
2.5 MPC tuning parameters (experiment). 35

3.1 Comparison between traditional Controllers and ESC. 40
3.2 ESC parameters for one-dimensional case. 45
3.3 ESC parameters for 2D navigation scheme (simulation). 49
3.4 ESC parameters for 2D navigation scheme (experiment). 52
3.5 ESC parameters for 2D navigation scheme (source seeking experiment) . . 54

4.1 CA-ESC parameters (for simulation). 63
4.2 CA-ESC parameters (for known objective function experiment). 65
4.3 CA-ESC parameters (for light source seeking experiment). 66
4.4 CA-ESC parameters (for simulation). 69
4.5 CA-ESC parameters (for known objective function experiment). 71
4.6 CA-ESC parameters (for light source seeking experiment). 73
4.7 CA-ESC simulation parameters for unicycle dynamics. 86
4.8 ESC and adaptation law parameters for simulations. 92

Chapter 1

Introduction

1.1 Autonomous Vehicles

Autonomous vehicles have gained popularity due to their utilization in industries, aerospace, and military robotics [1, 2]. Capable of moving from their current state to their desired state, they are capable of navigating through environments that are dangerous for humans such as where toxic gases and high temperatures are present [3]. This technology is a hot topic in control theory due to its potential to enhance the transportation system's efficiency by reducing the load from the driver, augmenting fuel efficiency, and decreasing the chances of traffic-related casualties [4, 5].

One of the primary features of an autonomous vehicle/robot is to track a reference trajectory. The trajectory-tracking method computes the control command that drives a robot to follow the reference trajectory [6]. It involves two steps: generating a trajectory and tracking it. The trajectory is characterized as either a sequence of consecutive reference points or by a set of geometrical primitives like lines or curves [7]. The trajectory-tracking problem has been a focal point of research for the last few decades.

It is considered one of the important features of controlling the vehicle. It requires gradient information in case of a pre-defined trajectory or real-time trajectory tracking. Among the methods to conduct trajectory tracking, there are those that depend on having access to kinematics for characterizing the trajectory and there are those that build their characterization of the trajectory based on the geometry of the environment.

In this work, the particular path that we are concerned with is generated via the Lie-Bracket System (LBS) [8–10]. While we explain LBS in chapter 2, we want the reader here to know that LBSs are gradient-like systems in which they go from an initial point to a terminal point of a path that is characterized by an objective function. The terminal point is an extremum (maxima/minima) point of the objective function

1.2 Motivation

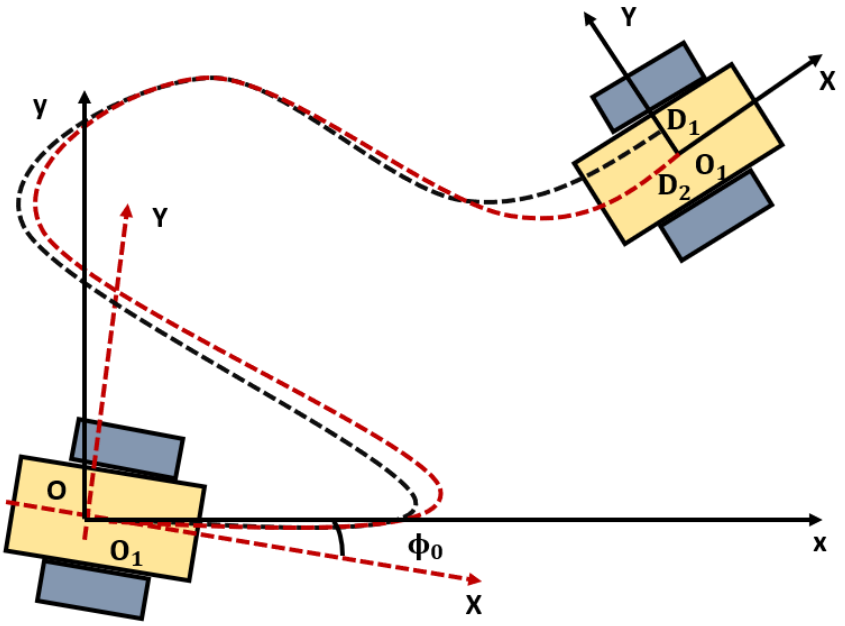

FIGURE 1.1: Depiction of trajectory tracking problem in xy plane. The black dotted is the reference trajectory and the red is the actual trajectory tracked by the differential-drive robot.

Consider the figure 1.1, which demonstrates a trajectory-tracking scenario. In the figure, a differential drive robot is presented which has body axes as (X, Y), whereas global axes as (x, y). The initial angle between the global and body axes is ϕ_0. The generated reference trajectory from point "O" to point "D_1" is presented by the black dotted curve, and the actual trajectory that is tracked by the robot is presented by the red dotted curve. It can be observed from the figure that the controller drove the robot from its initial point (O_1) to its near-desired position (the actual final point is D_2). However, from the figure, we can explicitly observe that the controller is ineffective to compute optimal control inputs for the robot so that it can follow the trajectory smoothly and perfectly.

Numerous controllers have been deployed in the works mentioned in the literature to address the trajectory tracking issue. Despite the efficiency demonstrated by the controllers in resolving the problem, there are still some limitations that persist with them. The limitations are discussed in the following points.

- **Optimization**: There are two steps in path-tracking: path planning, and path tracking. Generating a trajectory for reference is a crucial step. There are different techniques to generate it. However, generating an optimal path is important to enhance the performance.
- **Information**: Traditional controllers need information about the reference trajectory and a known mathematical expression of an objective function to compute the control inputs for the robot to drive. However, there are some cases where the objective function is unknown (i.e. sensor data), in such cases traditional controllers are ineffective.

- **Computational Complexity**: Some controllers are computationally complex. For example, model predictive controller (MPC) [11] is considered one of the advanced controllers that not only minimizes the difference between the current states of the robot, and the reference states but also solves online optimization problems. Nevertheless, it is computationally complex which makes it less desirable.

- **Flexibility**: Traditional controllers may not be very flexible in some cases where additional constraints are introduced. For instance, if there is an obstacle on the path, to avoid it, the controller gives the command to the robot which may drive it away from its reference trajectory and it may become hard to follow the desired path back. Consequently, the accuracy of the controller reduces in tracking the path.

- **Model-based**: Traditional controllers are model-based controllers which means that their performance relies on the model. To be a highly efficient controller, the model needs to be accurate. However, getting an accurate model is difficult to obtain [4]. If the model is not accurate then it would have an adverse effect on the performance of the controller.

To overcome the limitations of traditional controllers, we move forward to the use of the extremum-seeking scheme. Extremum seeking control (ESC) system is a control system that is used to determine and maintain the extremum (maximum/minimum) of a function [12–15]. ESC is a model-free optimization technique, which estimates the local gradient of an objective function without the explicit knowledge of mathematical expression, however, there are other kinds of ESC that involve higher-order derivative information (i.e. hessian) [16]. In fact, we are motivated in this thesis to revisit C-ESC

experiments such as in [17], and conduct experiments with new concepts of CA-ESC and recently developed techniques of attenuating oscillations [8, 9, 18–20]

1.3 Literature Review

1.3.1 Proportional – Integral – Derivative Controller (PIDC)

In the paper [3], the authors address the problem of trajectory tracking of the system using the Particle Swarm Optimization PID (PSO-PID) controller. They use two PIDCs to control the position and orientation of the system. PSO technique is utilized to get optimal values of the tuning parameters of the controller (K_P, K_D, K_I) and it demonstrates good results in a short time as compared to other optimization techniques. In [7], a trajectory-tracking controller based on a robust PID algorithm is proposed by the authors. They provide a method that uses a simple linearized model of the mobile robot and demonstrates the performance and robustness of the controller experimentally. The robust tuning of the controller is done using a simple and intuitive parameter, allowing for a good compromise between performance and robustness. A special case of PIDC (PI controller) is used to resolve two different tasks: "going to target", and "Following a trajectory". in [21]. The authors consider the "Khepera II" platform for the controller to adjust the motor velocities. The provided method is efficient in accomplishing both tasks, with a simple design and fewer tuning parameters. In the literature [22], a fuzzy-PID controller is provided by the researchers to solve the trajectory-tracking problem. The Fuzzy controller is utilized to find the optimal values of PID tuning parameters. The comparative results between the fuzzy-PID controller and the classical PID show that the fuzzy-PID controller has a better performance than the classical PID for a mobile robot with high stability, and performance.

1.3.2 Pure Pursuit Controller (PPC)

The authors in [6] mention a novel method that heuristically selects a look-ahead point by considering the relationship between a vehicle and a path. By exploiting the look-ahead point, the vehicle can stably converge to the desired path and track it effectively. The results of the proposed method show that the vehicle tracks the desired path more accurately than using standard PPC, and verify the effectiveness of the method experimentally using an autonomous vehicle [6]. In literature [23], the authors tackle the problem of PPC ineffectiveness in tracking a path at high speeds. They propose a model predictive active yaw control implementation of pure pursuit path tracking that accommodates the vehicle's steady-state lateral dynamics to improve tracking performance at high speeds. They conduct tests for three different paths from low to high speeds. The results show that the idea of implementing a receding horizon strategy for pure pursuit tracking improves their performance significantly [23]. The paper [24] provides an algorithm to track a trajectory based on two fuzzy controllers to adjust the speed and foresight distance in the pure pursuit method simultaneously. The effectiveness of the proposed algorithm is illustrated by simulation and field experiments on different trails and shows that the system tracked the trajectories of both continuous and discontinuous curvatures more effectively than the standard pure pursuit algorithm. On the other side, field experiments indicate that the provided algorithm has very small distance errors when tracking the reference path [24].

1.3.3 Model Predictive Controller (MPC)

In [1], the authors propose a new solution to the trajectory tracking problem for input-constrained differential-driver robots. The technique is designed by properly leveraging two main ingredients: 1) a feedback linearization (FL) technique and 2) a Receding Horizon Control (RHC) framework. The controller has a unique capability to deal with state-dependent input constraints acting on the feedback linearized vehicle model while ensuring recursive feasibility [1]. The controller's effectiveness is validated experimentally on a mobile robot by comparing the control performance with different schemes [1]. In the literature [25], the authors tackle the problem of trajectory tracking by using an Explicit-MPC. The MPC solution is calculated offline and expressed as a piece-wise affine function of the current state of a differential-drive mobile robot. This approach is restricted to a linear model. The simulation results show a good performance and verified it experimentally. [25]. In the work [26], authors present a framework for real-time, full-state feedback, unconstrained, Nonlinear-MPC (N-MPC) that combines trajectory optimization and tracking control in a single, unified approach. The method used an iterative optimal control algorithm, known as Sequential Linear Quadratic (SLQ), in a MPC setting to solve the underlying nonlinear control problem and simultaneously derive the optimal feed-forward and feedback terms. The performance of the approach is validated on two different hardware platforms: AscTec firefly hexacopter and the ball balancing robot Razer [26]. The results show that the MPC problem can be solved in a few milliseconds, even for time horizons of several seconds, indicating that the approach can scale well to more complex systems [26].

1.4 Thesis Contribution

Our contributions in this thesis are specified as follows -

- Investigate the performance of traditional controllers (a special form of PIDC (PDC), PPC, and MPC) in addressing trajectory-tracking problems when the trajectory is meant to take the system from a given point to an extremum of an objective function and conducting simulations in MATLAB/Simulink. Reference trajectory is generated using the concept of LBS as mentioned [8–10].
- Conduct experiments by deployment of a real-world robot (Turtlebot3 (TB3)) [27]. Deployment is done in real-time.
- Literature review and study of C-ESC then conduct simulations and experiments for 1-D and 2-D cases using C-ESC with a known mathematical expression of objective functions [17].
- Conduct an experiment by C-ESC for source-seeking for an unknown objective function. For this experiment, a light source is used.
- We conduct a literature review and study CA-ESC. We also discuss the 2-D navigation model of CA-ESC for single-integrator design and unicycle dynamics [8, 9, 18, 28].
- We conduct simulations and experiments using both models (single-integrator and unicycle dynamics models).
- We propose our novel proposed design (for both single-integrator and unicycle dynamics models) with attenuating oscillations and a better convergence rate. Said design is based on using the concept of Geometric-Based Extended Kalman Filter (GEKF) and adaptation law for attenuating oscillations [19, 20].

- We perform simulations using our proposed design to show the efficiency of our designs in attenuating the oscillations by comparing its performance with traditional single-integrator and unicycle dynamics designs mentioned in literature [8].

1.5 Thesis Layout

The work in this thesis is structured as follows: Chapter 2 introduces the concept of LBS for generating a reference trajectory. In this chapter, we investigate the performance of traditional controllers by performing simulations, and then verify it experimentally. In Chapter 3, we do the literature review and study of C-ESC. We show the simulation results and experimental results with known mathematical expressions of an objective function and also perform a light source-seeking experiment in which we only have access to light sensor measurement to show the effectiveness of the C-ESC design mentioned in [17]. We study and do the literature review of CA-ESC in chapter 4. In this chapter, we utilize the single-integrator and unicycle dynamics models with CA-ESC to conduct simulations and experiments in real-time. Moreover, we propose our novel single-integrator and unicycle CA-ESC designs with attenuating oscillations and better convergence rate. The designs are based on using the concepts of GEKF and adaptation law for attenuating oscillations. Our proposed design consists of traditional CA-ESC coupled with GEKF and adaptation law that guarantees the attenuation of oscillations. We summarise our observations and talk about our next plans to proceed with our research in chapter 5.

Chapter 2

Model-based Solutions with Simulations and Experiments

2.1 Introduction

In this chapter, we investigate the performance of three different traditional controllers ((a special form of Proportional-Integral-Derivative Controller (PD Controller (PDC))), Pure Pursuit Controller (PPC), and Model Predictive Controller (MPC)) in addressing the trajectory-tracking problem. Trajectory-tracking is considered one of the important features of controlling the vehicle. It requires gradient information, however, it is one of the challenges in control theory to drive the vehicle so that it follows the trajectory with high accuracy. The efficiency of tracking it highly relies on the controller.

In this chapter, we have worked in two steps: in step 1, a reference trajectory is generated by using LBS which is based on the gradient information (in terms of known mathematical expression), and in step 2, we perform simulations on MATLAB/Simulink using all controllers mentioned in the previous paragraph. Moreover, we conduct real-time

experiments by deploying the controllers on real hardware TB3 to verify the simulation results.

We present the path planning algorithm in section 2.2, using LBS. Differential-drivel model is presented in section 2.3. Analysis and study of traditional controllers are manifested in section 2.4. We discuss experimental setup, and results in sections 2.5, and 2.6. We provide a conclusion of this chapter based on the simulation and experimental results in section 2.7.

2.2 Path Planning Under Consideration

Gradient systems play an important role in different areas of science since they are utilized for analyzing the stability of the dynamical system, optimization, and control theory in designing the controller to track a desired trajectory. A gradient system is defined as the system's dynamics in terms of gradient function. Mathematically, it can be defined as [29–32]:

$$\dot{\mathbf{x}} = -k\nabla V(\mathbf{x}), \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^n$, which is a state variable of the system, k is any positive constant, $V(\mathbf{x})$ is the potential function that is continuous and twice differentiable, and $\nabla V(\mathbf{x})$ is the gradient of the potential function. The negative sign indicates the system is moving toward the minimum of the function V, which occurs when $\nabla V = 0$ [30, 32]. The trajectory follows the path of negative V. The stability of the system can be determined by using the Lyapunov function. The condition for the system to be stable is that the

dot product of the derivative of the state with the gradient of the potential function should be negative [30, 32].

$$\dot{\mathbf{x}}.\nabla V(\mathbf{x}) < 0. \tag{2.2}$$

From (2.1), and (2.2),

$$-k\nabla V(\mathbf{x}).\nabla V(\mathbf{x}) < 0, \tag{2.3}$$

$$-k\nabla^2 V(\mathbf{x}) < 0, \tag{2.4}$$

from the above equation, we can say the Lyapunov stability criteria is satisfied.

A lie bracket can be defined as a mathematical operator that describes how one vector field changes on the flow of another vector field. Let two vector fields $\mathbf{g}$ and $\mathbf{f}$ on the manifold then lie derivative of $\mathbf{g}$ with respect to vector $\mathbf{f}$ can be defined as:

$$L_{\mathbf{f}}\mathbf{g} = \frac{\partial \mathbf{g}}{\partial x}\mathbf{f} - \frac{\partial \mathbf{f}}{\partial x}\mathbf{g} = [\mathbf{f}, \mathbf{g}]. \tag{2.5}$$

The expression $L_{\mathbf{f}}\mathbf{g}$ represents the Lie bracket, $\frac{\partial \mathbf{g}}{\partial x}\mathbf{f}$ represents the flow of the gradient of vector $\mathbf{g}$ in the direction of $\mathbf{f}$. Similarly, $\frac{\partial \mathbf{f}}{\partial x}\mathbf{g}$ represents the gradient flow of vector $\mathbf{f}$ in the direction of $\mathbf{g}$.

Notice that if we replace the function V with an objective function f in (2.1), which we want our robotics system to track, then the system becomes ascending or descending based on the gradient. One can note that LBS works similarly to the gradient system

(2.1). A gradient system can be written in terms of LBS. Using gradient information, this system moves along a gradient to go from the initial point to the extremum (minimum/maximum) of an objective function. For instance, consider an objective function f and a scalar positive constant k. The lie derivative will be:

$$L_f k = \frac{\partial k}{\partial x} f - \frac{\partial f}{\partial x} k = [f, k], \tag{2.6}$$

$$L_{fk} = -k \nabla f, \tag{2.7}$$

from the above equation, we can essentially say that a gradient system equation (2.1) can be written as a LBS. Note $\frac{\partial k}{\partial x} f = 0$ since k is a positive constant.

Generating a trajectory for the vehicle to be followed using the controller is one of the essential steps in the trajectory-tracking algorithm. Many factors can affect the trajectory-tracking algorithm, such as curvature, obstacles, etc., which make the trajectory-tracking algorithm challenging. We utilized the LBS from one of the significant literatures [8]. The authors addressed the multi-agent problem, in which they defined three different objective functions for three different agents. The following objective functions are:

$$f^a(\bar{x}) = -\frac{1}{2}(x_1^a - 1)^2 - \frac{1}{2}(x_2^a - 1)^2 + x_1^{b^2} + x_2^{b^2} - 10 + e^{(-x_1^{c^2} - x_2^{c^2})}, \tag{2.8}$$

$$f^b(\bar{x}) = -\frac{1}{2}(x_1^b + 1)^2 - \frac{1}{2}(x_2^b + 1)^2 - 10 + \sin(x_1^a + x_2^a), \tag{2.9}$$

$$f^c(\bar{x}) = -\frac{1}{2}(x_1^c + 1)^2 - \frac{3}{2}(x_2^c - 1)^2 - 10, \tag{2.10}$$

where the parameters x_j^i represent the 2-D states (x *and* y) of the system, where $i = a,\ b,\ and\ c$ are the functions of the respective map and $j = 1,\ or\ 2$ represents the system's state. Here is one important point to mention that it is not mandatory to use any specific objective function from the above equations, however for simulation and experiments, we fixed the objective function provided by equation 2.8.

The corresponding LBS states are given as [8]:

$$\dot{z^i}_1 = \frac{1}{2}(c^i \alpha^i \nabla_{z_1^i} f^i(\bar{z}) - c^{i^2} \nabla_{z_2^i} f^i(\bar{z})(f^i(\bar{z}) - z_e^i h)) \tag{2.11}$$

$$\dot{z^i}_2 = \frac{1}{2}(c^i \alpha^i \nabla_{z_2^i} f^i(\bar{z}) + c^{i^2} \nabla_{z_1^i} f^i(\bar{z})(f^i(\bar{z}) - z_e^i h)) \tag{2.12}$$

$$\dot{z^i}_e = -z^i{}_e h^i + f^i(\bar{z}), \tag{2.13}$$

z_1, and z_2 are states of the system, z_e is the state of the filter, f is the objective function, h is the high pass filter (HPF) value c, and α are the constants.

Consider figure 2.1, which depicts the LBS trajectories of all the objective functions that we discussed. Trajectory 1 (in blue) corresponds to the objective function (2.8), similarly trajectories 2 (in red), and 3 (in green) correspond to objective functions (2.9) and (2.10). The initial conditions are $[2, -2, -2, 2, -1, 2.5]$, and the final coordinates of the trajectories are $[1, 1, -1, -1, -1, 1]$. Throughout the thesis, the "star" sign depicts

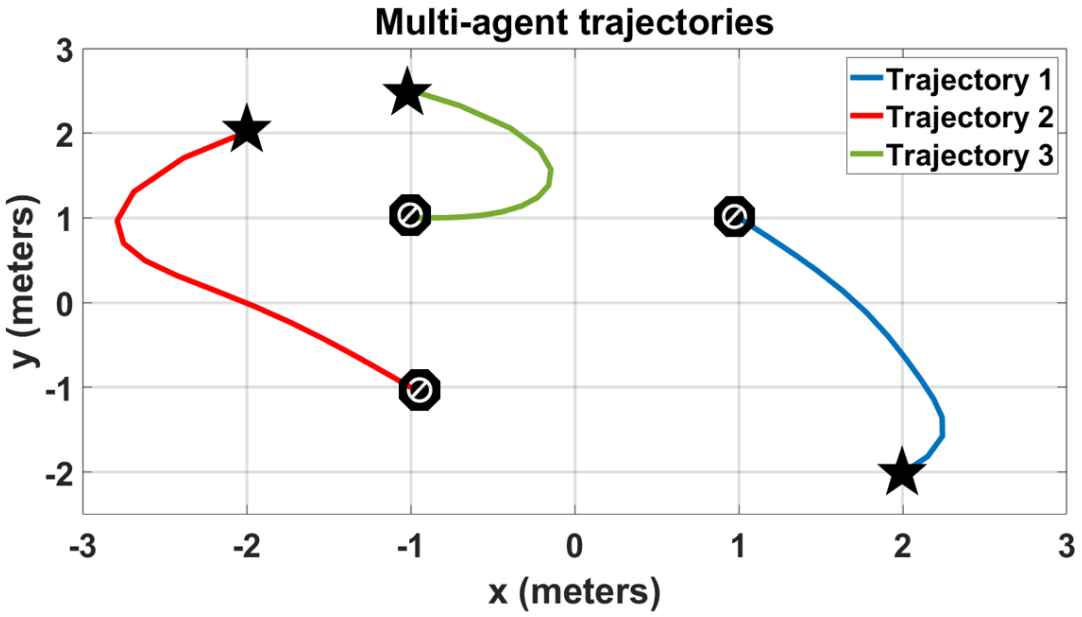

FIGURE 2.1: Three different trajectories generated by LBS corresponding to the equations. The blue trajectory corresponds to map 1 (2.8), the red trajectory refers to map 2 (2.9), and the green trajectory is of (2.10). The"star" sign refers to the starting position, and the "stop" sign refers to the terminal position.

the initial position of the system/robot, and the "end" sign corresponds to the terminal or final point of the system/robot.

2.3 Differential-Drive Model for the Robot

Consider the figure 2.2, in which a differential-drive robot is presented, having linear, and angular velocities as v, and ω. The global, and body axes are represented by (X, Y), and (X_B, Y_B). The robot is orientated at an angle θ w.r.t the horizontal axis. The left, and right wheel velocities are $\dot{\phi}_L$ and $\dot{\phi}_R$.

The model of the differential drive robot is represented by the following equations [33, 34]:

$$\dot{x} = v\cos\theta, \tag{2.14}$$

$$\dot{y} = v \sin\theta, \tag{2.15}$$

$$\dot{\theta} = \omega, \tag{2.16}$$

where x, y, and θ are the states of the robot. The expressions for v and ω are given as:

$$v = (\dot{\phi}_R + \dot{\phi}_L)\frac{r}{2}, \tag{2.17}$$

$$\omega = (\dot{\phi}_R - \dot{\phi}_L)\frac{r}{b}, \tag{2.18}$$

where r and b are the wheel radius and wheelbase of the robot, $\dot{\phi}_R$ is the right wheel velocity, and $\dot{\phi}_L$ is the left wheel velocity.

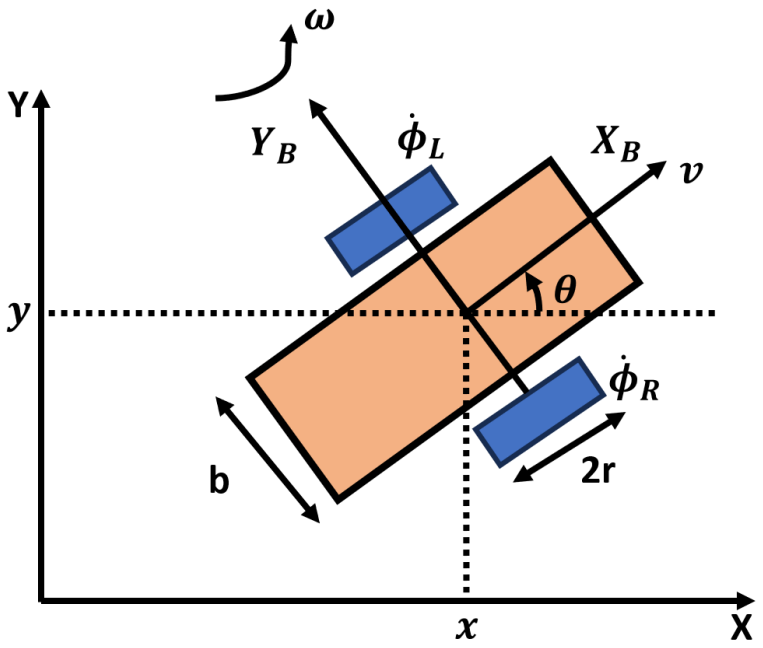

FIGURE 2.2: A depiction of differential-drive robot in XY plane.

2.4 Controllers

2.4.1 Proportional-Integral-Derivative Controller (PIDC)

PIDC is one of the most robust and popular controllers in control systems. The controller's objective is to minimize the error between the reference value and the measured value from the dynamical system and then gives the output [35, 36]. It ensures the system remains as close as possible to the reference value. As its name suggests, it has three components which are proportional, integral and derivative. The proportional component develops an output signal that is proportional to the error. The integral term creates a signal proportional to the accumulation of error measurements over the previous time intervals, and the derivative component creates a signal proportional to the rate of change of an error [35, 36]. The PIDC is depicted in figure 2.3. The generalized PID law is given by:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}, \tag{2.19}$$

where the $u(t)$ is the output, k_p, k_i k_d are the gains. $e(t)$ is the error in the system.

In our work, we have used a special form of PIDC, PDC. The error terms in the respective states are provided by the equations 2.20, 2.21, and 2.22. Moreover, the control law of the PDC is provided in the equation 2.23:

$$e(y) = y(r) - y, \tag{2.20}$$

$$e(x) = x(r) - x, \tag{2.21}$$

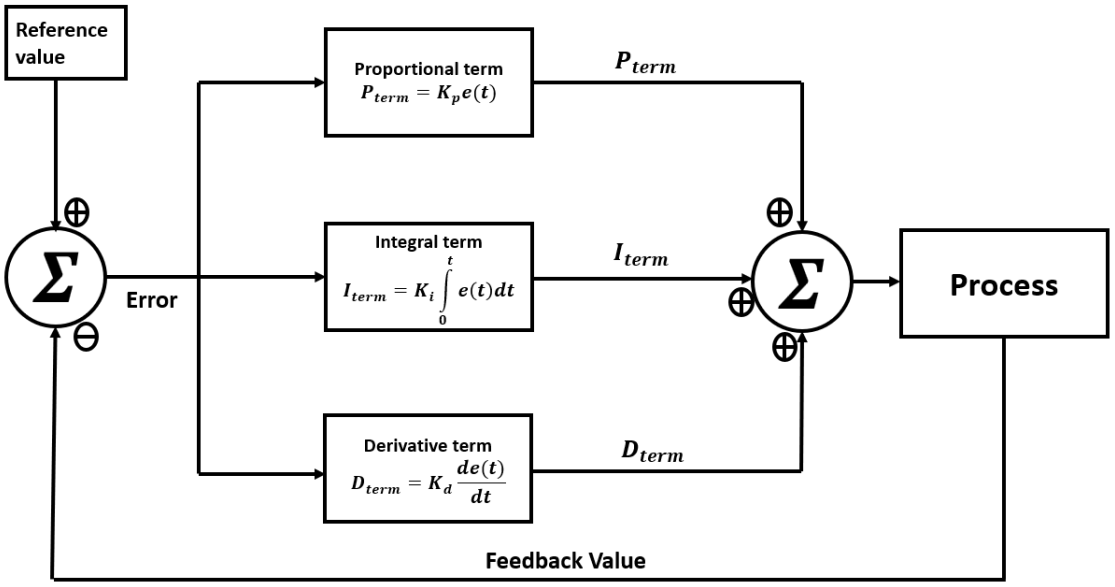

FIGURE 2.3: PID control law design.

$$e(\theta) = \arctan \frac{e(y)}{e(x)}, \tag{2.22}$$

$$u(\theta) = K_p e(\theta) + k_d \frac{d}{dt}(e(\theta)), \tag{2.23}$$

where $y(r)$ and $x(r)$ are reference states at time t, whereas x and y are the true states of the robot at time t. $e(y)$, $e(x)$ and $e(\theta)$ are the errors in the states of the robot. Note that $u(\theta)$ is an error representation of the robot's orientation.

2.4.2 Model Predictive Controller (MPC)

MPC is considered as one of the most successful and popular advanced control methods that predict the future behavior of the controlled system over a finite time horizon and computes an optimal control input by minimizing an objective function. MPC optimizes the performance of the system with governing constraints, that represent physical, economic, and environmental limits of the system [11]. It efficiently handles the system

constraints and deals with Multiple Input and Multiple Output (MIMO) systems. The comprehensive MPC design is depicted in fig 2.4.

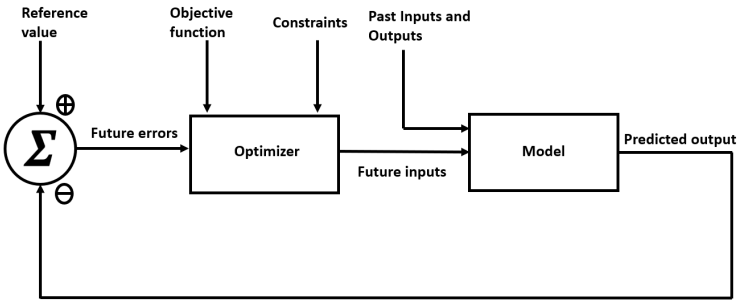

FIGURE 2.4: MPC model design consists of reference path, optimizer, model, and a feedback loop of current states of the system.

The generalized form of MPC for a linear, and discrete-time system is given as:

$$\mathbf{x}[k+1] = \mathbf{x}[k] + u[k], \tag{2.24}$$

where $\mathbf{x}$ is the state vector, and u is the control action.

The objective of the MPC is to minimize the objective function:

$$J = \sum_{k=0}^{N_p-1} \mathbf{x}^2[k] + \rho u^2[k] + \mathbf{x}^2[N_p], \tag{2.25}$$

where $\rho > 0$ is the weight, J is an objective function.

An unconstrained MPC problem gives an analytical solution, while a constrained problem is solved by a numerical solution, and the problem is stated as the discrete state space:

The discrete state space equations are presented as [11]:

$$x_{k+1} = Ax_k + Bu_k, \tag{2.26}$$

$$y_k = Cx_k + Du_k, \tag{2.27}$$

the linear-quadratic objective function is given as:

$$J = \sum_{k=0}^{N_P} (\hat{y} - r)^T Q(\hat{y} - r) + \nabla u^T R \nabla u, \tag{2.28}$$

where r is the output reference value, $\hat{y}$ is the predicted process output, ∇u is the predicted change in the control value $\nabla u = u_k - u_{k-1}$, Q is the output error weight matrix, J is the objective function, and R is the control weight matrix.

The optimal future control input is achieved at $\frac{\partial J}{\partial u} = 0$.

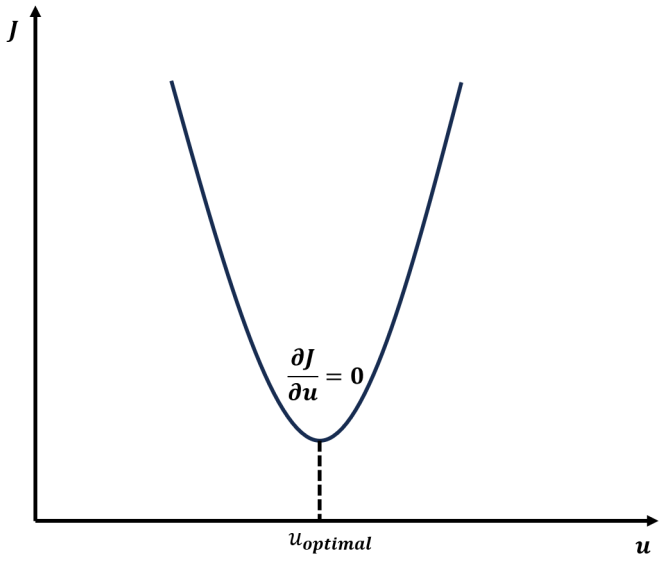

FIGURE 2.5: PPC.

2.4.3 Pure Pursuit Controller (PPC)

A PPC is a geometric trajectory tracking controller that tracks the reference trajectory using the geometry of the vehicle kinematics and the reference trajectory. This is known as lateral vehicle control. PPC has only one tuning parameter which is look-ahead distance. Look-ahead distance is how far away along the path the robot should look from the current position to command the angular velocity [37]. The look-ahead point can be far away, and the control input is computed in real-time, which makes it to work efficiently in case of a non-smooth trajectory or when the trajectory is defined by using waypoints [6]. It reduces the error between the heading and the lateral of the vehicle by controlling the heading angle of the vehicle [24].

Figure 2.6, depicts a bicycle model in which the current position of the rear wheel axle of the vehicle is at A (A_x, A_y) [24]. The target point is set at B (B_x, B_y). The distance between the current position and the target point is look-ahead distance L_d. R represents the radius around the instantaneous center of rotation (ICR) that is at O, and δ is the steering angle [24].

In isosceles triangle ΔOAB:

$$\psi_1 + \psi_2 + \psi_3 = 180^\circ, \tag{2.29}$$

since OA, and OB are equal to R,

$$\psi_2 = \psi_3 = (\pi/2 - \theta), \tag{2.30}$$

from equations 2.27, and 2.28,

$$\psi_1 = 2\theta. \tag{2.31}$$

Applying sine rule in ΔOAB:

$$AB/\sin(\psi_1) = OA/\sin(\psi_3), \tag{2.32}$$

which yields to,

$$L_d/\sin(2\theta) = R/\sin(\pi/2 - \theta), \tag{2.33}$$

$$R = L_d/2\sin(\theta), \tag{2.34}$$

the curvature of the reference trajectory is given as:

$$c = 2\sin(\theta)/L_d, \tag{2.35}$$

for the case of simplified two-wheeled vehicle model, the steering angle is given as:

$$\delta = arc\tan(cL), \tag{2.36}$$

$$\delta = arc\tan(2L\sin\theta/L_d), \tag{2.37}$$

from the above equation of the steering angle, we can observe that the steering angle is relying on the L_d. Therefore, L_d is a crucial parameter of PPC that should be tuned carefully to drive the vehicle on the predefined trajectory effectively.

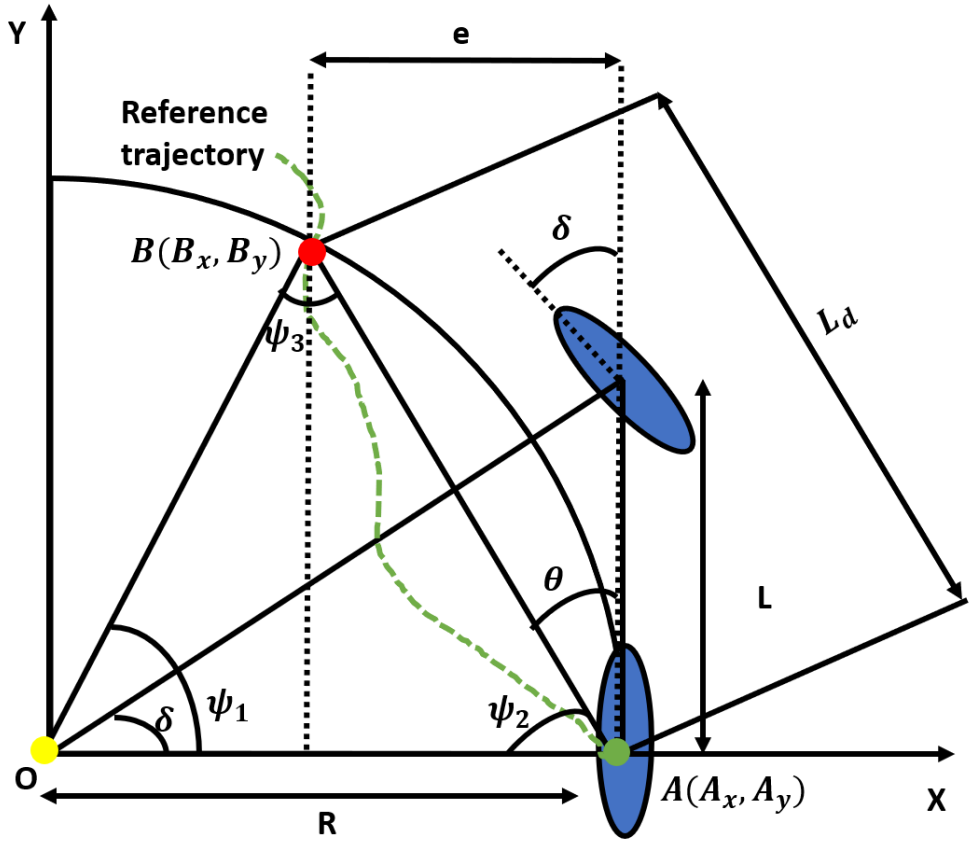

FIGURE 2.6: A detailed description of the PPC algorithm in which the green trajectory represents the reference trajectory, and the differential-drive robot is in blue.

2.5 Experimental Setup

In this section, we discuss the hardware and software that we use to perform the experiments. We perform all the experiments in our Modeling, Dynamics, and Control Lab (MDCL) [38], the experimental setup is provided in figure 2.7.

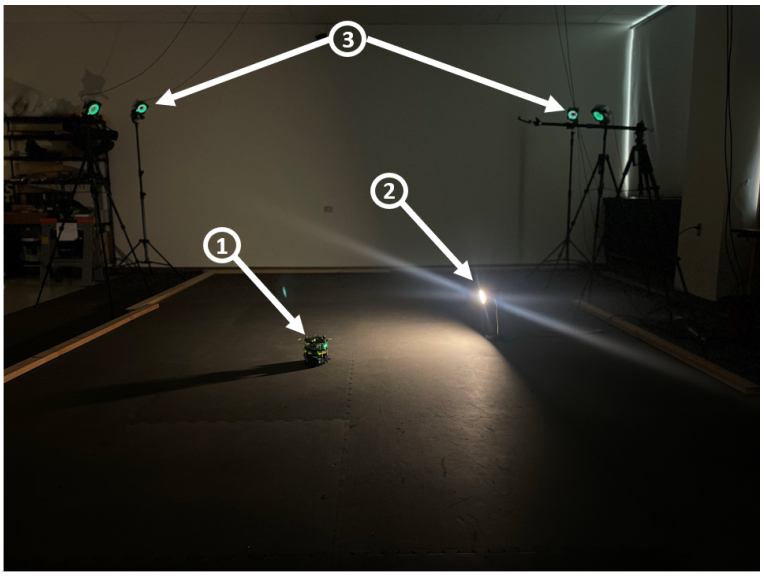

FIGURE 2.7: Modeling, Dynamics, and Control Lab (MDCL). 1. TB3 inside the testing area, 2. A light source, 3. Motion capturing system (MCS).

2.5.1 Hardware

2.5.1.1 Turtlebot3 (TB3)

TB3 is a small, Robot Operating System (ROS)-based, differential drive robot [27]. It is widely used for educational, and research purposes. It consists of Raspberry Pi 3B+ as its microcomputer, a LIDAR sensor to measure the distance, dynamixel motors to drive the wheels, and OpenCR supplies power to the robot.

2.5.1.2 Motion Capturing System (MCS)

We exploit MCS to get the real-time data of TB3 to perform experiments. MCS detects the rigid body (using at least three markers to create a rigid body), tracks its real-time pose, and streams it to the local network [39, 40].

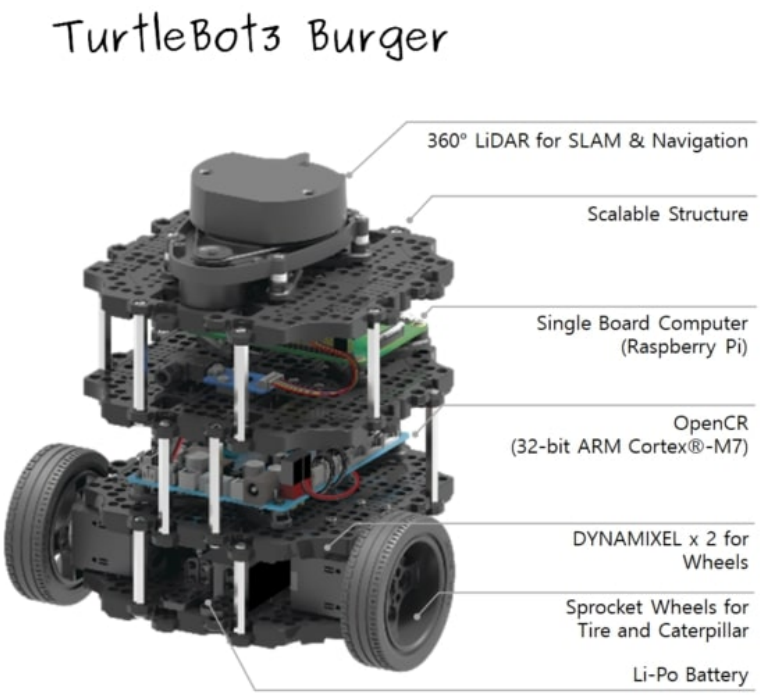

FIGURE 2.8: A depiction of turtlebot3 (TB3) burger with its components: LIDAR, Raspberry pi 3B+, OpenCR, Dynamixel motors.

2.5.2 Software

2.5.2.1 MATLAB

We develop controllers design on MATLAB/Simulink, to conduct simulations, and establish communication between the TB3, and MATLAB through a local network. To carry out the experiments, we utilize different toolboxes -ROS toolbox to send control input commands to TB3, Navigation toolbox to use PPC, Raspberry Pi support toolbox, and OptitrackToolbox to collect the real-time pose of the robot from motive software [41].

2.5.2.2 Motive

Motive software is a licensed software of optitrack. It works with cameras to track and capture the motion of markers and sends the data to the network with high accuracy

[39].

2.6 Results

We present simulation and experimental results using the controllers mentioned above in this chapter. We utilize MATLAB/Simulink software to conduct simulation results, and then deploy the controllers on TB3 to perform real-time experiments.

2.6.1 Simulations

The generalized model design for all the controllers to conduct simulations is shown in fig 2.9. In the figure, the controller block represents the respective controller (PDC, PPC, and MPC), the model corresponds to the dynamics of the system, and in the feedback loop, we get the states of the system. For our work, we get the x, and y states in the feedback. The initial value of the states is $(2, -2)$, and the terminal point is $(1, 1)$. The linear velocity and angular velocity of the system are computed by the controller. The wheel radius and wheelbase values are 0.033 m, and 0.178 m.

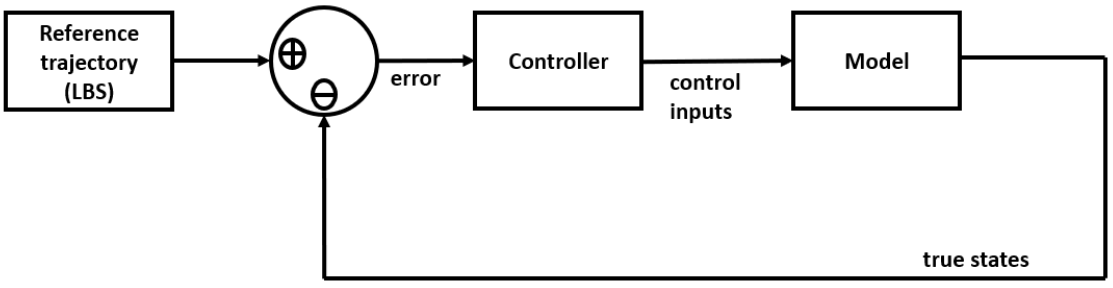

FIGURE 2.9: A generalized structure of trajectory-tracking model design for simulations.

2.6.1.1 Proportional-Derivative Controller (PDC)

The tuning parameters of the PDC are mentioned in table 2.1 (from 2.10 to 2.14), and the simulation results are provided in figure 2.15. The planar trajectory of the system is plotted in figure 2.14. The states of the system (x, y) are depicted in 2.10, and 2.11. The control inputs (linear and angular velocities) are shown in 2.12, and 2.13. The system's trajectory is depicted in 2.14. From the states and trajectory graphs, it can be explicitly observed that during the first 15 seconds, the PDC is unable to compute the proper control inputs for the system to follow the trajectory, after which the system starts to follow it as we can observe it from the states plots. Moreover, we have used the upper bound $(0.2\ m/s)$ in linear velocity.

Parameters	**Values**
Simulation time (seconds)	50
K_p	0.5
K_d	0.001

TABLE 2.1: PDC tuning parameters for simulations.

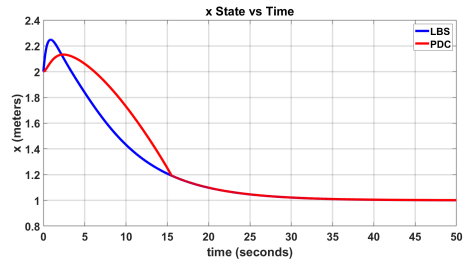

FIGURE 2.10: (a)

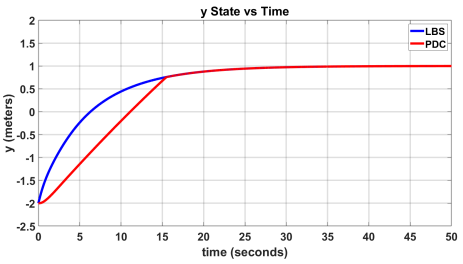

FIGURE 2.11: (b)

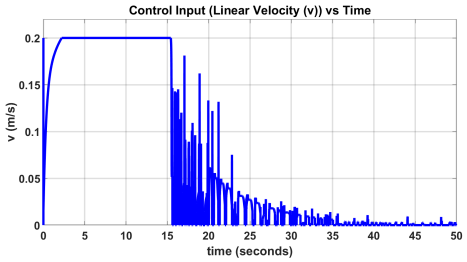

FIGURE 2.12: (c)

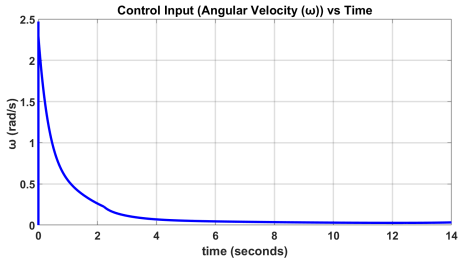

FIGURE 2.13: (d)

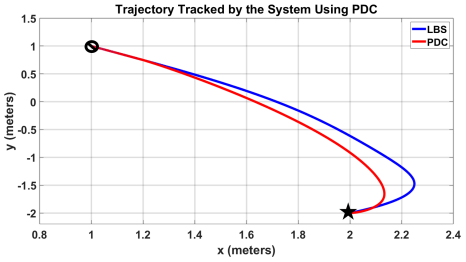

FIGURE 2.14: (e)

FIGURE 2.15: Simulation results of PDC. (a) x state, (b) y state, (c) Linear velocity, (d) Angular velocity, and (e) Trajectory of the system using PDC.

2.6.1.2 Pure Pursuit Controller (PPC)

Parameters	Values
Desired Linear Velocity (m/s)	0.15
Maximum Angular Velocity (rad/sec)	2.84
Look-ahead Distance	4.5

TABLE 2.2: PPC simulation tuning parameters.

The simulation parameters are provided in the table 2.4 and the plots of the simulation results of the system using PPC are shown in figure 2.21 (from 2.16 to 2.20). States graphs are shown in figures 2.16 and 2.17. Figure 2.20 depicts the planar trajectory graph of the system. The control inputs in figures 2.18 and 2.19. From the trajectory graph in 2.20, it is clear that the system doesn't follow the trajectory well, especially near the curvature as there is a significant difference between the trajectory of the system and of the reference trajectory. From the states graph, PPC doesn't look effective at all in addressing the trajectory-tracking problem. PPC computes constant desired linear velocity and computes angular velocity based on the error (difference between the states of the system and the trajectory).

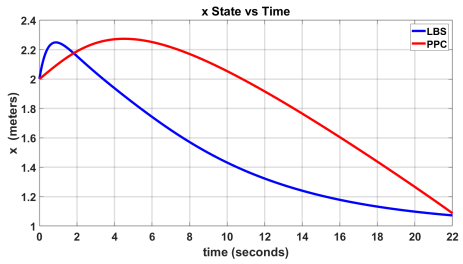

FIGURE 2.16: (a)

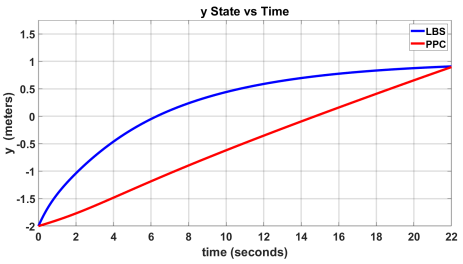

FIGURE 2.17: (b)

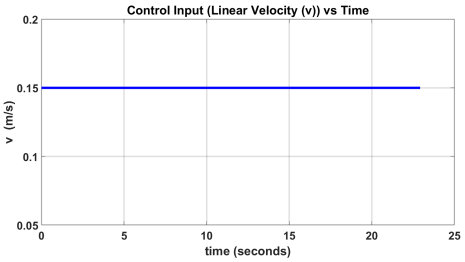

FIGURE 2.18: (c)

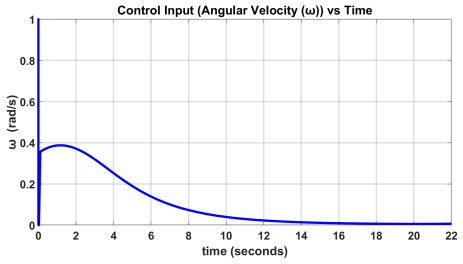

FIGURE 2.19: (d)

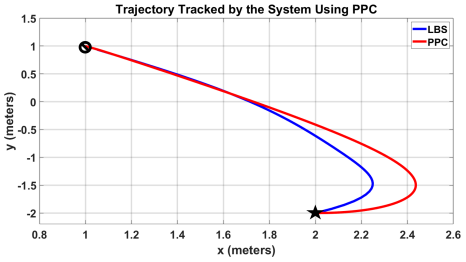

FIGURE 2.20: (e)

FIGURE 2.21: Simulation results of PPC. (a) x state, (b) y state, (c) Linear velocity, (d) Angular velocity, and (e) Trajectory of the system using PPC.

2.6.1.3 Model Predictive Controller (MPC)

The simulation results are provided in figure 2.27 (from 2.22 to 2.26) and the tuning parameters are given in the table 2.3. States plots are shown in figures 2.22, and 2.23. The control inputs are plotted in figures 2.24, and 2.25. The system's trajectory is provided in 2.26. From the simulation results, we can observe that the MPC performed well in addressing the trajectory-tracking problem and outperformed PDC and PPC. From the trajectory plot, the system tracks the LBS trajectory with minimal deviation. The effectiveness of the controller can be verified from the states graph, as the system's states follow the reference states well, however, there is a minute steady state error in the x state.

$$Q = \begin{bmatrix} x_m & 0 & 0 \\ 0 & y_m & 0 \\ 0 & 0 & \theta_m \end{bmatrix} \tag{2.38}$$

$$R = \begin{bmatrix} v_m & 0 \\ 0 & \omega_m \end{bmatrix} \tag{2.39}$$

Parameters	Values
P	20
x_m	115
y_m	300
θ_m	0
v_m	10
ω_m	300

TABLE 2.3: MPC simulation tuning parameters.

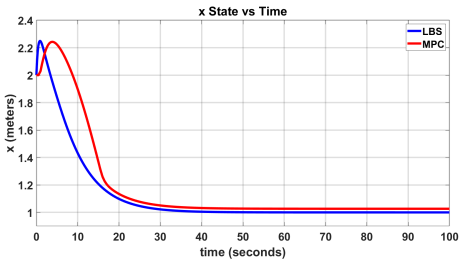

FIGURE 2.22: (a)

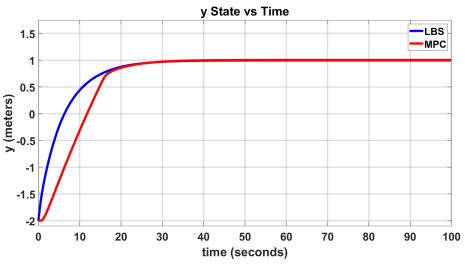

FIGURE 2.23: (b)

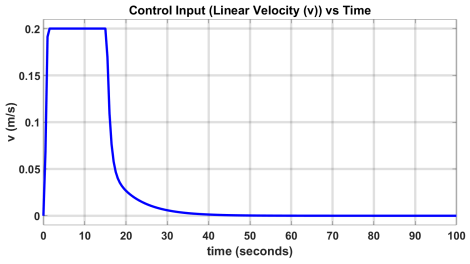

FIGURE 2.24: (c)

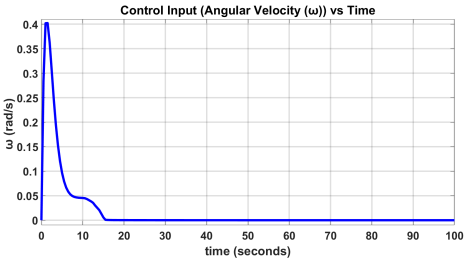

FIGURE 2.25: (d)

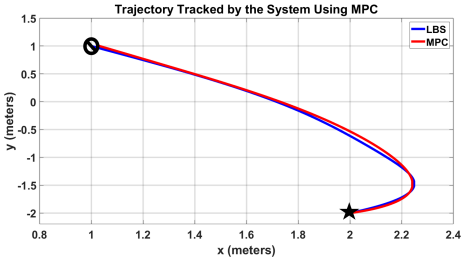

FIGURE 2.26: (e)

FIGURE 2.27: Simulation results of PPC. (a) x state, (b) y state, (c) Linear velocity, (d) Angular velocity, and (e) Trajectory of the system using MPC.

2.6.2 Experiments

In this subsection, we discuss the experimental results of real-time trajectory-tracking using the TB3 robot and then compare the performance of MPC, and PPC in terms of their efficiency in tracking the trajectory. The real-time states (x, y) are obtained by MCS and then sent to the controllers as feedback. For the feasibility of the experiments, we maintain a constant linear velocity of 0.22 m/s.

2.6.2.1 Pure Pursuit Controller (PPC)

The experimental results of the controller are presented in figure 2.32 (from 2.28 to 2.31) and the tuning parameters are given in the table 2.4. The states graph is provided in figures 2.28 and 2.29. The trajectory plot is shown in figure 2.30. The controller's output (angular velocity) is plotted in 2.31. As we have observed in the simulation results using PPC, we got similar experimental results. From the trajectory graph, TB3 doesn't follow the trajectory at the curvature. The states graph shows that the PPC didn't compute the prper control inputs for TB3.

Parameters	**Values**
Desired Linear Velocity (m/s)	0.22
Maximum Angular Velocity (rad/sec)	2.84
Look-ahead Distance	3

TABLE 2.4: PPC tuning parameters (for experiment).

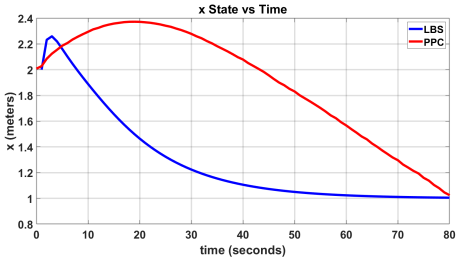

FIGURE 2.28: (a)

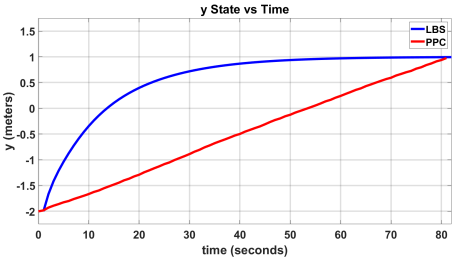

FIGURE 2.29: (b)

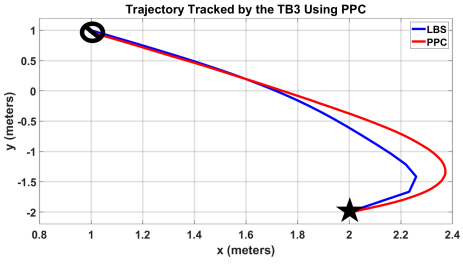

FIGURE 2.30: (c)

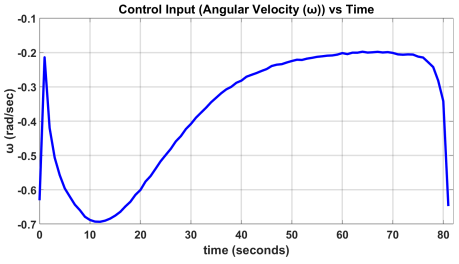

FIGURE 2.31: (d)

FIGURE 2.32: Experimental results of PPC. (a) x state, (b) y state, (c) Trajectory of the TB3 using PPC and (d) Angular velocity.

2.6.2.2 Model Predictive Controller (MPC)

The tuning parameters for the controller are provided in the table 2.5 and the experimental results of the controller are presented in figure 2.37 (from 2.33 to 2.36). The states of the robot are depicted in figures 2.33, and 2.34. Control input is depicted in figure 2.36. From the trajectory plot, TB3 follows the LBS trajectory quite well with some oscillations in its trajectory and in control input. However, the states plot depicts that the states of the TB3 didn't follow the reference states effectively. There are some oscillations in the control input (angular velocity).

Parameters	**Values**
P	10
x_m	50
y_m	100
θ_m	-0.8
v_m	50
ω_m	75

TABLE 2.5: MPC tuning parameters (experiment).

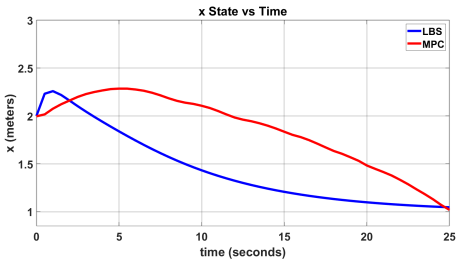

FIGURE 2.33: (a)

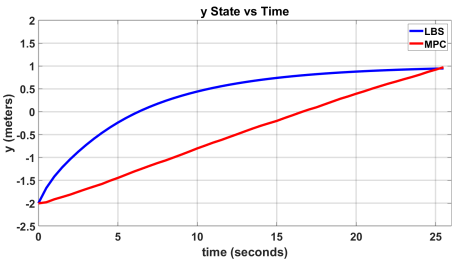

FIGURE 2.34: (b)

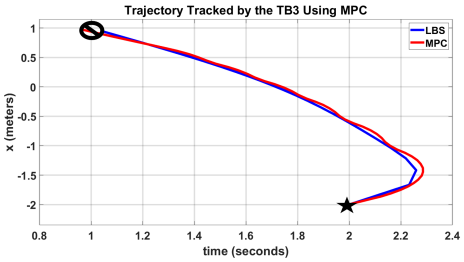

FIGURE 2.35: (c)

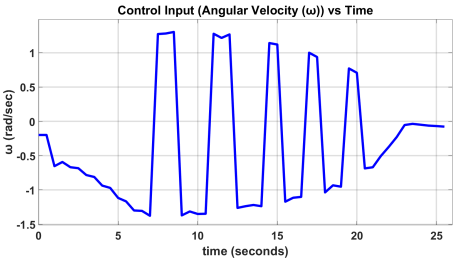

FIGURE 2.36: (d)

FIGURE 2.37: Experimental results of MPC. (a) x state, (b) y state, (c) Trajectory of the TB3 using MPC, and (d) Angular velocity.

2.7 Conclusion

In this chapter, we attempted to address the problem of trajectory-tracking using three different traditional controllers: PDC, PPC, and MPC. We introduced the concept of LBS and generated the reference trajectory using it for the system/robot to follow. We performed some simulations using the traditional controllers mentioned above to conduct a comparative study of their performance. From the simulation results, it is clear that the PDC and PPC were quite ineffective in driving the system on the reference trajectory, especially near the curvature of the lie bracket trajectory. On the other side, MPC outperformed PPC and PDC significantly. Additionally, we conducted experiments in real-time to verify our simulation results of PPC and MPC. In the experiment also, MPC outperformed the PPC as we observed from the experimental results.

Chapter 3

Revisiting Model-Free Source Seeking in Robots via Classical Extremum Seeking

3.1 Introduction

In this chapter, we discuss extremum-seeking control (ESC) systems and their application. Moreover, we go through its classic form (C-ESC), and how it works. We perform simulations and experiments using C-ESC. We conduct the simulation for the 1-D case in which we have a tuning parameter (θ) and then we move to the 2-D navigation case where we have tuning parameters (x, y). Using the 2-D model design, we conduct simulations and experiments in real-time. For the experiment, we show the results using a known objective function and discuss the source-seeking experiment that we perform using a light source, in which we do not have access to the mathematical expression of an objective function, however, we do have access to its measurements.

This chapter covers the introduction of the ESC systems in 3.2. Then, we discuss C-ESC structure 3.2.1. Following that, we perform and talk about simulation, and experimental results under the section 3.3. In the end, we collect all the observations and conclude the chapter in section 3.4.

3.2 Extremum Seeking Control (ESC) System

ESC systems are adaptive control systems that aim to optimize the input-output map of a nonlinear dynamical system by estimating the gradient of an unknown objective function [12, 14, 42, 43]. ESC systems do not require any mathematical expression of the objective function, just access to its measurements is sufficient [20]. They operate in real-time. Different schemes of ESC systems have been proposed to address optimization problems [44, 45], and recently, interest is growing in solving extremum-seeking problems with constraints [45]. ESC schemes utilize the measurement of an objective function relying on the real-time value of the objective function, perturbation incurred by time-varying excitation signal on the system. This allows the ESC scheme to extract the information to estimate the gradient [18].

Autonomous vehicles are equipped with Global Positioning System (GPS) or Inertial Navigation System (INS). However, they are not reliable. GPS doesn't work in many areas including under ice, water, and in caves, on the other hand, the INS systems' cost is high [46]. On the other side, ESC systems are capable of working in different and challenging environments effectively and they can solve the seeking problem based on the local measurements of the signal, and it doesn't require any mathematical expression of the signal.

Traditional controllers (PDC, PPC, and MPC) have been widely used in different applications mentioned in the literature, however, in some cases, they possess some limitations which affect their effectiveness or make them undesirable. Some of them are mentioned in the table 3.1.

	Traditional Controllers (PDC, PPC, MPC)	**ESC System**
Information: Mathematical expression of an objective function	Required	Not necessary (access to measurements is enough)
Computation Complexity and real-time	Some are computationally complex	Generally not computationally complex
Model-Based	Required	Model-free

TABLE 3.1: Comparison between traditional Controllers and ESC.

A simple picture that explains how the ESC scheme works is presented in [47] is provided in the figure 3.1.

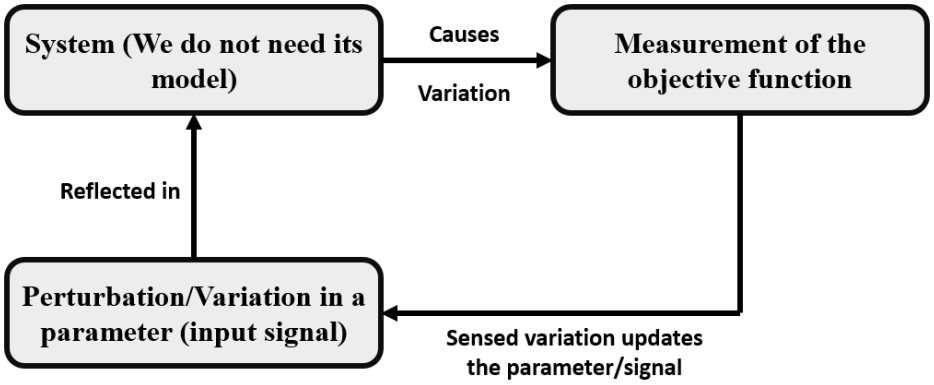

FIGURE 3.1: Simplified ESC system diagram consisting of blocks of Perturbation signal, System, and objective function.

ESC systems are not new in literature, it has been used for a very long time [45]. In literature [48], the authors used ESC to maintain the maximum power transfer from an electrical transmission line to a tram car. During World War 2, there was significant research activity in Russia using ES [49]. ESC systems were a hot research topic in the 1950s and 1960s [45]. The first assessment of the stability of a classic extremum-seeking feedback scheme was provided by Krstic and Wang [50].

ESC systems have a wide range of applications. In bioinspiration work [51], the authors provide a link between dynamic soaring (flight technique of birds for long- duration with very little energy as wind is utilized to gain lift), and ESC. ESC in this paper captures the biological behavior of soaring birds. In [52, 53], authors used ESC with Kalman filter (KF) to get the optimal position of the follower aircraft with respect to leading aircraft to reduce the drag in formation flight. In [54] authors present the experimental results with extremum-seeking adaptive algorithms to control combustion instability suitable for the reduction of acoustic pressure oscillations in gas turbines over a large range of operating conditions.

3.2.1 Classical Extremum-Seeking Control (C-ESC)

The most common structure of ESC in literature is C-ESC [15]. It is generally utilized for a nonlinear dynamical system subjected to a time-invariant objective function. The design of C-ESC is shown in figure 3.2. The parameter θ is what C-ESC updates to reach the extremum of the objective function. C-ESC structure consists of four primary steps [55]. In the first step, which is the modulation step, a perturbation signal- generally a sinusoidal signal- is added to the nominal value of the tuning parameter $\hat{\theta}$ (θ estimate). We initiate with a guess value of θ. The updated or generated value of θ is fed to the system dynamics $\dot{\mathbf{x}} = f(\mathbf{x}, a(\mathbf{x}, \theta))$. In the second step, the system responds based on the signal and it causes a change in an objective function (leads to a new measurement). In the third step, the measurement of the objective is multiplied with a demodulation signal (again generally a sinusoidal signal is used). The output of the demodulation signal decides whether to increase or decrease the $\hat{\theta}$. In the final step, the demodulated value is integrated and updates the tuning parameter $\hat{\theta}$. The High Pass Filter (HPF)

and Low Pass Filter (LPF) are optional filters and can be used or discarded based on the requirements or application.

The corresponding equations for C-ESC are as follows:

$$
\begin{aligned}
\dot{x} &= f(x, a(x, \theta)) \\
J &= g(x, \theta) \\
\theta &= \hat{\theta} + a \sin(\omega t) \\
\dot{\hat{\theta}} &= K\xi \\
\dot{\xi} &= -\omega_l \xi + \omega_l (J - \eta) b \sin(\omega t - \phi_{phase}) \\
\dot{\eta} &= -\omega_h \eta + \omega_h J
\end{aligned}
\tag{3.1}
$$

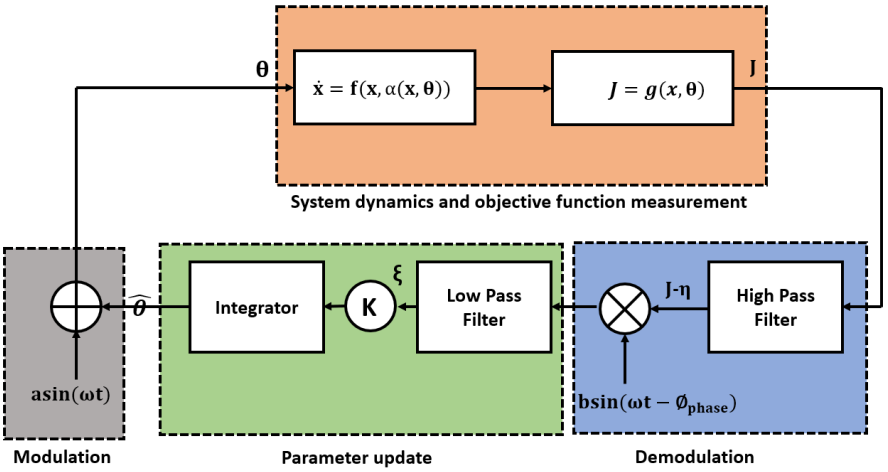

FIGURE 3.2: Generalized representation of the classical structure of extremum seeking controller (C-ESC). This structure of ESC consists of four major steps. First is the modulation step in which a perturbation signal is added to a nominal value of the tuning parameter. In the second step, the system responds and it leads to a change in an objective function (leads to a new measurement). In the third step, the measurement of the objective function gets multiplied with a demodulation signal and in the final step, the demodulation is integrated and updates the tuning parameter.

3.3 Results

We introduce a modified C-ESC design from [17] for slow sensors and we replicate their simulation results. For the extension of our work, we perform some experiments with known and unknown objective functions (we only have access to the measurements of the objective function) to verify the design in real-time.

3.3.1 Simulation Results

3.3.1.1 1-D Scheme

The generalized and modified C-ESC for the one-dimensional case is represented in figure 3.3. This design works with very slow sensors. These sensors are used to determine the hazardous gases and their dynamics are governed by a first-order system [17]:

$$G_{sensor}(s) = \frac{b}{s+\epsilon}, \tag{3.2}$$

where b, and ϵ are the constants that depend on the sensor.

The transformation of generalized to modified C-ESC is done by swapping the integrator and the multiplication by $\sin(\omega t)$. This is carried out by using the integration by parts [17]:

$$\int_0^t \eta(\tau)\sin(\omega\tau)\,d\tau = \sin(\omega t)\int_0^t \eta(\tau)\,d\tau - \omega\int_0^t \cos(\omega\tau)\int_0^\tau \eta(\sigma)\,d\sigma\,d\tau. \tag{3.3}$$

By utilizing this observation, the authors transform the generalized C-ESC design to the modified C-ESC, where the guiding idea is the sensor is a pure integrator when $\epsilon = 0$, and the modified version also works for $\epsilon > 0$. The mathematical expression of the objective function for the simulation is given as:

$$f(\theta) = \frac{\delta^*}{1 + P_\theta(\theta - \theta^*)^2}, \tag{3.4}$$

where P_θ is a constant having value 0.5, $\delta^* = 250$. From the equation 3.4, the maximum value of the nonlinear objective function (J^*) is 250 at $\theta = \theta^*$ where $\theta^* = 0$. The values of b, and ϵ are 0.037 and 0.046. The ESC parameters are given in the table 3.2.

ESC Parameters	**Values**
a	0.2
ω	30
k	10
h	1

TABLE 3.2: ESC parameters for one-dimensional case.

The simulation results for the one-dimensional case are represented in figure 3.8 (from 3.4 to 3.7). The objective function graph is depicted in figure 3.4. Figure 3.5 represents the position estimate ($\hat{\theta}$) of the gas leak. The sensor reading and HPF plots are plotted in 3.6, and 3.7. The objective function ($f(\theta)$) converges to its extremum (maximum) as position estimate ($\hat{\theta}$) moves to $\theta^\star$.

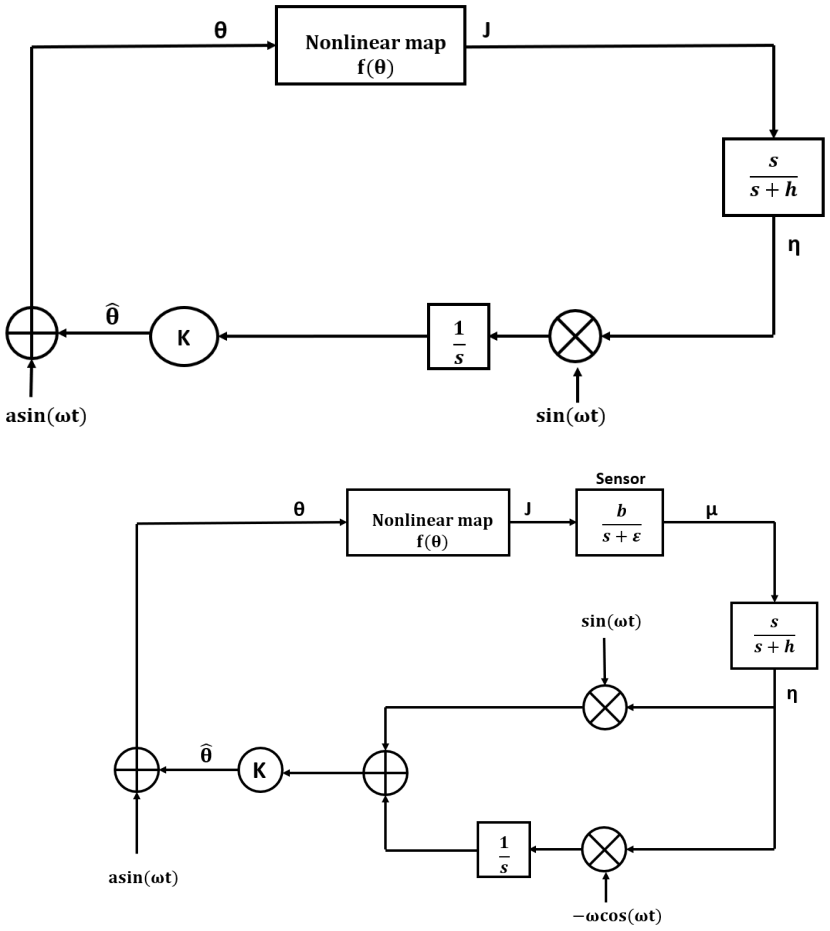

FIGURE 3.3: (a) Generalized and modified designs C-ESC. The generalized C-ESC is the top figure, and the modified C-ESC design is the bottom figure which works well with both cases: in the case of a slow sensor ($\epsilon > 0$) and the case of a pure integrator when $\epsilon = 0$.

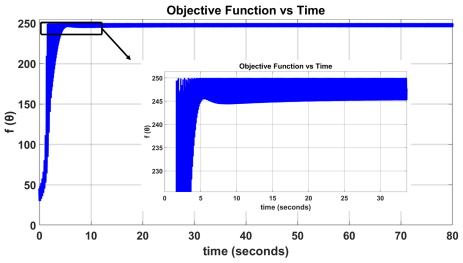

FIGURE 3.4: (a)

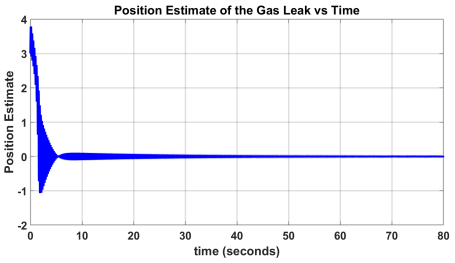

FIGURE 3.5: (b)

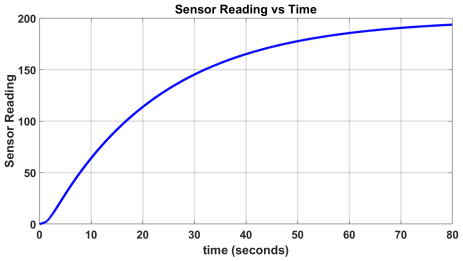

FIGURE 3.6: (c)

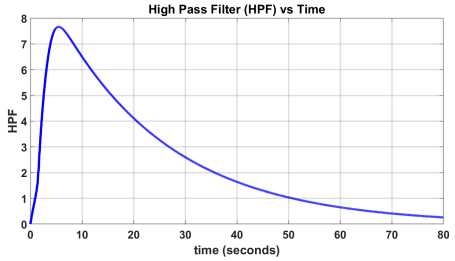

FIGURE 3.7: (d)

FIGURE 3.8: Simulation results of modified C-ESC with slow sensor dynamics. (a) Objective function, (b) Position estimate of the leak, (c) Sensor reading, and (d) HPF.

3.3.1.2 2-D Navigation

In this section, we discuss a vehicle modeled as a 2D point mass with a slow sensor as per provided in literature [17]. The C-ESC design is presented in figure 3.9. In the design, two ESC laws are used for each state (x, y). Both laws have the phase difference of 90°. [17, 56]. To get an integrator at the input of the objective function, a term $= \frac{s}{s}$ is inserted between the gain and the perturbation signal. The term $= \frac{1}{s}$ is moved downstream in the signal flow direction and after the perturbation input, which results in a differentiation of perturbation, and an integrator has appeared at the input of the objective function. The differentiator term s from the term $\frac{s}{s}$ is replaced with an approximate differentiator, i.e., a washout filter $\frac{s}{s+d}$ [17].

The control inputs for the vehicle are:

$$\dot{x}(t) = u_x(t), \tag{3.5}$$

$$\dot{y}(t) = u_y(t), \tag{3.6}$$

where $u_x(t)$ and $u_y(t)$ are the two control inputs and the control laws are:

$$u_x(t) = a\omega cos(\omega t) + K_x\xi_x, \tag{3.7}$$

$$u_y(t) = -a\omega sin(\omega t) + K_y\xi_y, \tag{3.8}$$

The mathematical expression of the objective function for the simulation is given as:

$$f(x,y) = \frac{\delta^*}{1 + p_x(x - x^*)^2 + p_y(y - y^*)^2}, \tag{3.9}$$

where p_x, and p_y are constants with values of 1 and 0.5, $\delta^* = 250$, initial position of the vehicle is at $(1, 1)$, and $(x^*, y^*) = (0, 0)$. The values of b, and ϵ of slow sensor dynamics are 0.037 and 0.046. The ESC parameters are provided in the table 3.3.

ESC Parameters	**Values**
ω	20
a	0.1
k_x	0.75
k_y	0.75
d_x	0.2
d_y	0.2
h	0.5

TABLE 3.3: ESC parameters for 2D navigation scheme (simulation).

The results of the 2D navigation simulation are shown in figure 3.18 (from 3.10 to 3.17). Figure 3.10 represents the objective function ($f(x, y)$) plot. The trajectory of the system is given in figure 3.11. The HPF and sensor reading graphs are provided in the figures 3.12, and 3.13. The states (x, y) of the vehicle vs time are plotted in figures 3.14, and 3.15. Control inputs for both states are provided in figures 3.16 and 3.17

The objective function converges to its maximum value in approximately 40 seconds. The states move towards $(x^\star, y^\star)$ and then oscillate around it. The system converges to the extremum from its initial position and then revolves around it.

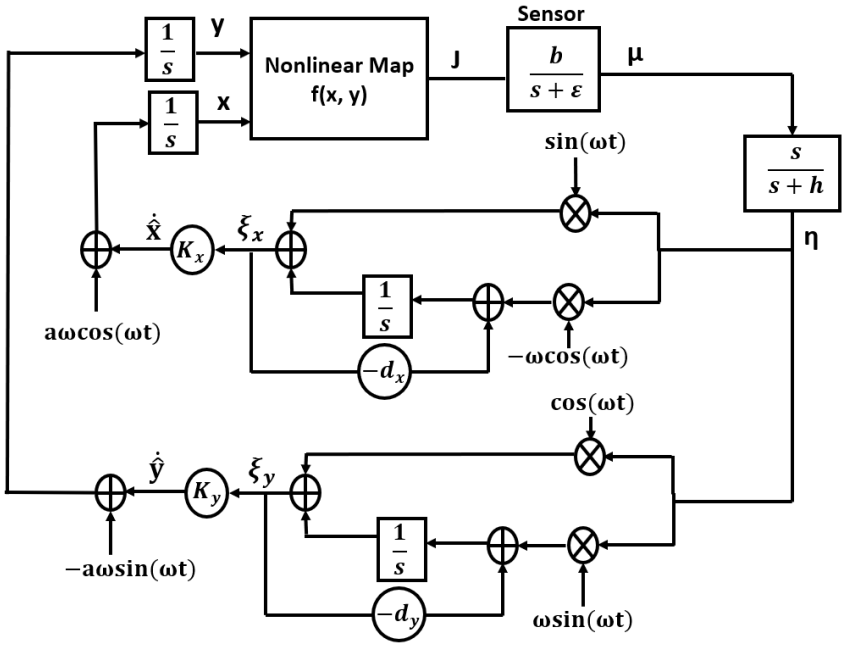

FIGURE 3.9: ES design for 2-D navigation with slow sensor

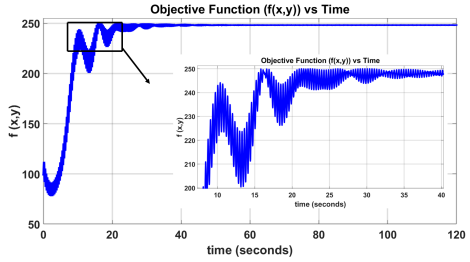

FIGURE 3.10: (a)

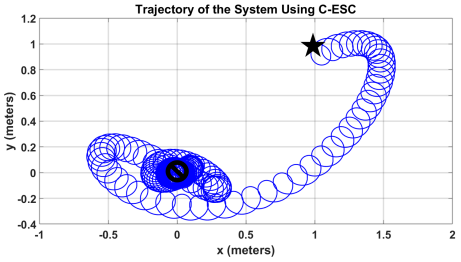

FIGURE 3.11: (b)

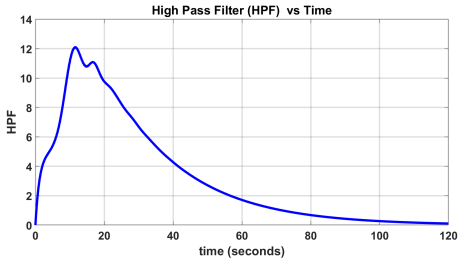

FIGURE 3.12: (c)

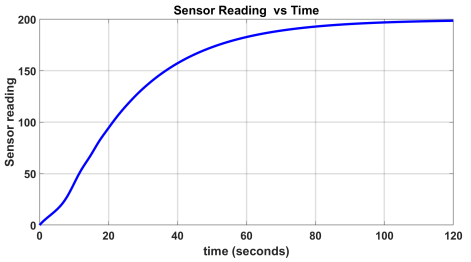

FIGURE 3.13: (d)

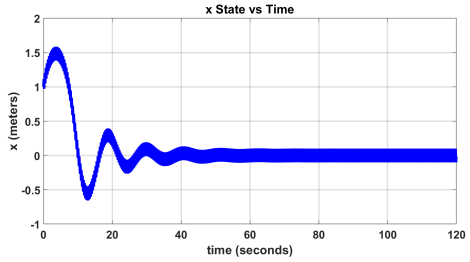

FIGURE 3.14: (e)

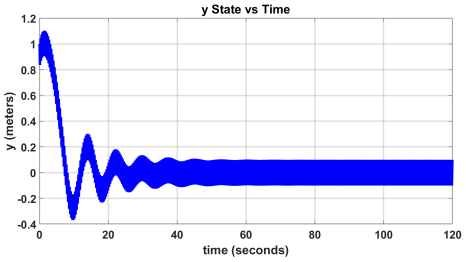

FIGURE 3.15: (f)

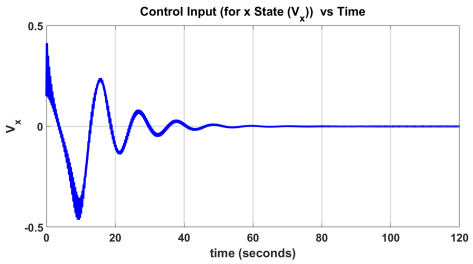

FIGURE 3.16: (g)

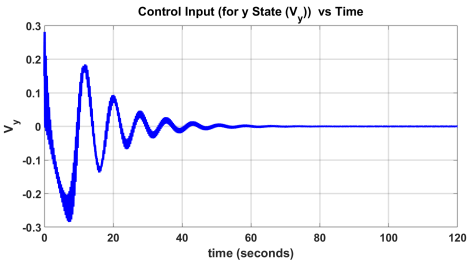

FIGURE 3.17: (h)

FIGURE 3.18: Simulation results of modified C-ESC for 2-D navigation with slow sensor. (a) Objective function, (b) Planar trajectory of the system (c) HPF, (d) Sensor reading, (e) x state (f) y state, and (g) Control input for x state (h) Control input for y state.

3.3.2 Experimental Results

3.3.2.1 With known Objective Function

We present the experimental results that we performed by deploying the C-ESC 2D scheme on TB3 in real-time.

For the experiment, we utilized the same mathematical expression of the objective function as for the simulation of 2D navigation given in (3.9). The initial position of TB3 is $(2.4, -1)$. The values of the parameters in the objective function are as $p_x = 1$, $p_y = 1$, $\delta^* = 250$, and $(x^*, y^*) = (1, 1)$. The control inputs to the TB3 are linear velocity (v) and angular velocity (ω). We kept the constant linear velocity 0.22 m/s, which is its maximum linear velocity. The ESC parameters are mentioned in the table 3.4.

ESC Parameters	Values
ω	30
a	1.5
k_x	0.5
k_y	0.5
d_x	1
d_y	1
h	1

TABLE 3.4: ESC parameters for 2D navigation scheme (experiment).

The experimental results are shown in figure 3.25 (from 3.19 to 3.24). The objective function $(f(x, y))$ is plotted in the figure 3.19. The planar graph that depicts the trajectory of TB3 is presented in 3.20. The states of the TB3 are presented in figures 3.21, and 3.22. The HPF and control input are shown in figures 3.23, and 3.24. The objective function converges to its maximum value in around 240 seconds, as the states (x, y) of the TB3 move towards $(x^\star, y^\star)$. From the trajectory plot, it is clear that after reaching the extremum position, TB3 revolves around it. The experimental video is provided in [57].

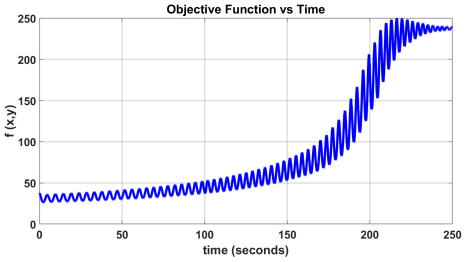

FIGURE 3.19: (a)

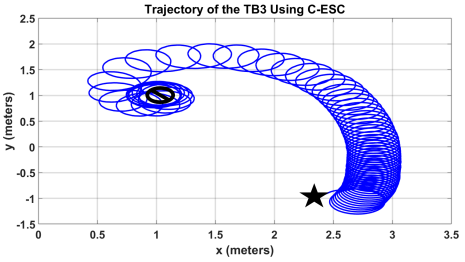

FIGURE 3.20: (b)

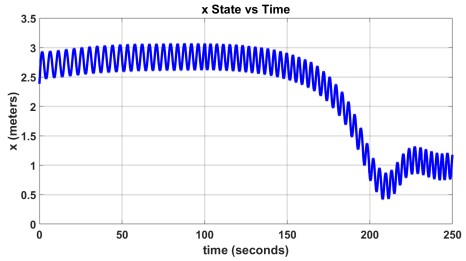

FIGURE 3.21: (e)

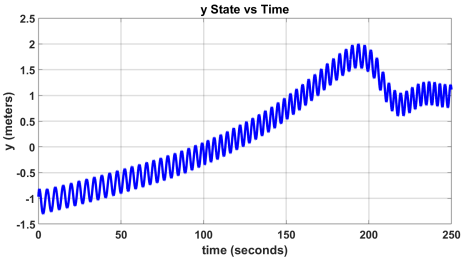

FIGURE 3.22: (d)

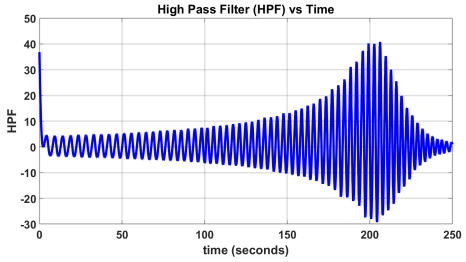

FIGURE 3.23: (e)

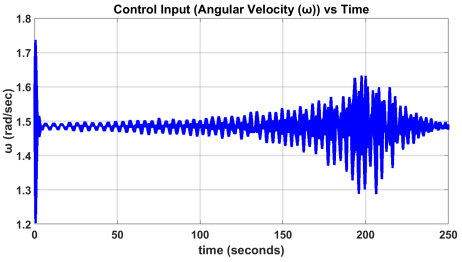

FIGURE 3.24: (f)

FIGURE 3.25: Experimental results of modified C-ESC for 2-D navigation for known objective function. (a) Objective function, (b) Planar trajectory of TB3, (c)x state (d) y state, (e) HPF, and (f) Angular velocity.

3.3.2.2 With unknown Objective Function (Light Source-Seeking)

Here we discuss the 2nd experiment in which we conduct a source-seeking experiment, using a light source. The source represents the maximum intensity of the signal. For the light sensor experiment, we fixed a light sensor on the TB3, and two arduinos - one is connected to TB3, and the other is connected to the Simulink system - are utilized to stream the light sensor data from the robot to Simulink through a local network. The C-ESC parameters are provided in the table 3.5. The experimental video is provided in [58].

ESC Parameters	**Values**
ω	30
a	2.5
k_x	1
k_y	1
d_x	1
d_y	1
h	1

TABLE 3.5: ESC parameters for 2D navigation scheme (source seeking experiment)

FIGURE 3.26: (a)

FIGURE 3.27: (b)

FIGURE 3.28: (c)

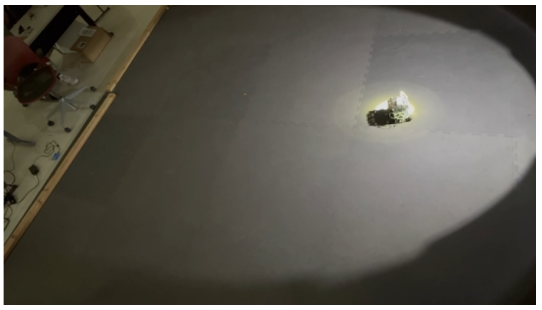

FIGURE 3.29: (d)

FIGURE 3.30: Source Seeking Experiment using a light source.

3.4 Conclusion

In this chapter, we revisited C-ESC structure using robotics in literature [17]. We utilized the modified C-ESC structure from [17], and replicated their simulation results for the 1-D scheme and 2-D navigation scheme. We verified the effectiveness of the design by conducting real-time experiments on real hardware (TB3 robot). In the first experiment, we utilized the mathematical expression of the objective function and performed real-time experiment. In the second experiment, we performed a source-seeking experiment, in which we used light as a source, TB3 detected the light source using a light sensor and sought the source.

Chapter 4

Model-free Source Seeking by a Robot: The Case of Control Affine-ESC (CA-ESC)

4.1 Introduction

In this chapter, we study the control affine structure of ESC (CA-ESC). In literature, there are two forms of ESC systems based on the linearity of control input in the structure: (a) the classic structure (C-ESC), and (b) the control-affine structure (CA-ESC), however, most experimental works utilized the C-ESC, and very few experimental works used the CA-ESC. CA-ESC systems often deal with problems expressed in control-affine formulation [20]. In control-affine form, the objective function is not modulated but it gets multiplied by the input signal, resulting in the objective function being comprised within the vector fields of the CA-ESC system [20].

The chapter is organized as follows: in section 4.2, we study the CA-ESC systems. We present the simulations and experimental results for unicycle and single-integrator dynamics in sections 4.3.1 and 4.3.2. In section 4.4 we propose a novel amended CA-ESC design for attenuation of oscillations and a better convergence rate using the concept of Geometric-Based Extended Kalman Filter (GEKF) for gradient estimation. We show the effectiveness of our proposed design by conducting simulations in the section 4.6.

4.2 Control Affine Extremum-Seeking Control (CA-ESC)

The generalized CA-ESC systems can be defined by the state space representation as [8, 20]:

$$\dot{\mathbf{x}} = \mathbf{b_d}(t, \mathbf{x}) + \sum_{i=1}^{m} \mathbf{b_i}(t, \mathbf{x})\sqrt{\omega}u_i(t, \omega t) \tag{4.1}$$

where $\mathbf{x}(t_0) = \mathbf{x_0} \in \mathbb{R}^n$ and $\omega \in (0, \infty)$. $\mathbf{x}$ is the state space vector, $\mathbf{b_d}$ is the drift vector field, u_i are the control inputs, m is the number of control inputs, while the vector $\mathbf{b_i}$ are the control vector fields. The CA-ESC system in (4.1) can be approximated and characterized by LBS [8, 20]:

$$\dot{\mathbf{z}} = \mathbf{b_d}(t, \mathbf{z}) + \sum_{\substack{i=1 \\ j=i+1}}^{m} [\mathbf{b_i}, \mathbf{b_j}](t, \mathbf{z})\nu_{j,i}(t), \tag{4.2}$$

where

$$\nu_{j,i}(t) = \frac{1}{T}\int_0^T u_j(t, \theta)\int_0^\theta u_i(t, \tau)d\tau d\theta. \tag{4.3}$$

The operation $[\cdot,\cdot]$ donates Lie bracket operation applied to two vector fields $\mathbf{b_i}, \mathbf{b_j}$: $\mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ with $\mathbf{b_i}(t,\cdot), \mathbf{b_j}(t,\cdot)$ being continuously differentiable, and is defined as:

$$[\mathbf{b_i}, \mathbf{b_j}](t,\mathbf{x}) := \frac{\partial \mathbf{b_j}(t,\mathbf{x})}{\partial \mathbf{x}} \mathbf{b_i}(t,x) - \frac{\partial \mathbf{b_i}(t,\mathbf{x})}{\partial \mathbf{x}} \mathbf{b_j}(t,\mathbf{x}). \tag{4.4}$$

To ensure the well-posedness of the systems described by equations 4.1 and 4.2 and to accurately approximate the ESC system with the corresponding LBS presented in [8, 28], certain assumptions must be adopted. The assumptions for 4.1 and 4.2 are:

A1. $\mathbf{b_i} \in C^2 : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n, i = 0, ..., m.$

A2. For every compact set $\mathscr{C} \subseteq \mathbb{R}^n$, there exist $A_1, ..., A_6 \in [0, \infty)$ such that $|\mathbf{b_i}(t,\mathbf{x})| \leq A_1, |\frac{\partial \mathbf{b_i}(t,\mathbf{x})}{\partial t}|$
$\leq A_2, |\frac{\partial \mathbf{b_i}(t,\mathbf{x})}{\partial \mathbf{x}}| \leq A_3, |\frac{\partial^2 \mathbf{b_j}(t,\mathbf{x})}{\partial t \partial \mathbf{x}}| \leq A_4,$
$|\frac{\partial [\mathbf{b_j}, \mathbf{b_k}](t,\mathbf{x})}{\partial \mathbf{x}}| \leq A_5, |\frac{\partial [\mathbf{b_j}, \mathbf{b_k}](t,\mathbf{x})}{\partial t}| \leq A_6$ for all $\mathbf{x} \in \mathscr{C}, t \in \mathbb{R}, i = 0, ..., m;\ j = 1, ..., m;\ k = j, ..., m.$

A3. $u_i : \mathbb{R} \times \mathbb{R} \to \mathbb{R}, i = 1, ..., m$ are measurable functions. Moreover, for every $i = 1, ..., m$ there exist constants $N_i, M_i \in (0, \infty)$ such that $|u_i(t_1, \theta) - u_i(t_2, \theta)| \leq N_i |t_1 - t_2|$ for all $t_1, t_2 \in \mathbb{R}$ and such that $sup_{t,\theta \in \mathbb{R}} |u_i(t,\theta)| \leq M_i$. Similarly, $u_i(t,\cdot)$ is T-periodic, i.e. $u_i(t, \theta + T) = u_i(t,\theta)$, and has zero average, i.e. $\int_0^T u_i(t,\tau) d\tau = 0$, with $T \in (0, \infty)$ for all $t, \theta \in \mathbb{R}, i = 1, ..., m$.

These assumptions are essential for the analysis and existence of solutions for the ESC system 4.1. A1 ensures the smoothness of the system's vector fields, including those involving the objective function. A2 guarantees the boundedness of the expressions involving the vector fields $\mathbf{b_i}$ $(i = 1, ..., m)$. A3 ensures T-periodicity and zero average

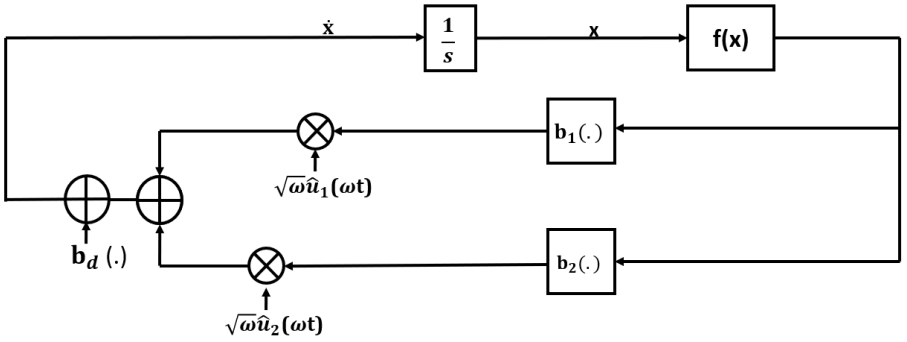

FIGURE 4.1: The generalized CA-ESC design

for the control inputs. The simplified CA-ESC structure is provided in the figure 4.1. In the figure, we have two control inputs ($\hat{u}_1(\omega t)$, and $\hat{u}_2(\omega t)$), two control vector fields (b_1 and b_2), and $b_d = 0$.

4.3 Results

In this section, we discuss the CA-ESC (with unicycle and single-integrator dynamics) structure mentioned in [8]. Moreover, we perform simulations and experiments.

4.3.1 Unicycle Dynamics

Consider the figure 4.2. In the figure, we have the unicycle dynamics with CA-ESC. For the simulations and experiments, the control input (linear velocity v) is computed from the CA-ESC, whereas the angular velocity is constant [8]. We have used a sinusoidal wave as a perturbation signal. Moreover, there is a HPF in the design to reduce noise.

The dynamics of the unicycle are given as:

$$\dot{x} = u\cos(\theta(t)), \tag{4.5}$$

$$\dot{y} = u \sin(\theta(t)), \tag{4.6}$$

$$\dot{\theta} = v, \tag{4.7}$$

where the u is the forward velocity computed from the ESC,

$$u = (c(f(x,y) - x_e h))\sqrt{\omega} sin(\omega t) + a\sqrt{\omega} cos(\omega t). \tag{4.8}$$

From equations 4.5, 4.6, 4.7, and 4.8, the control law for each state and HPF are provided as:

$$\dot{x} = ((c(f(x,y) - x_e h))\sqrt{\omega} sin(\omega t) + a\sqrt{\omega} cos(\omega t)) cos(\Omega t) \tag{4.9}$$

$$\dot{y} = ((c(f(x,y) - x_e h))\sqrt{\omega} sin(\omega t) + a\sqrt{\omega} cos(\omega t)) sin(\Omega t) \tag{4.10}$$

$$\dot{x_e} = -x_e h + f(x,y), \tag{4.11}$$

where $\theta(t) = \Omega t$.

4.3.1.1 Simulation Results

For the simulations, the expression of objective function $J(x,y)$ is chosen as:

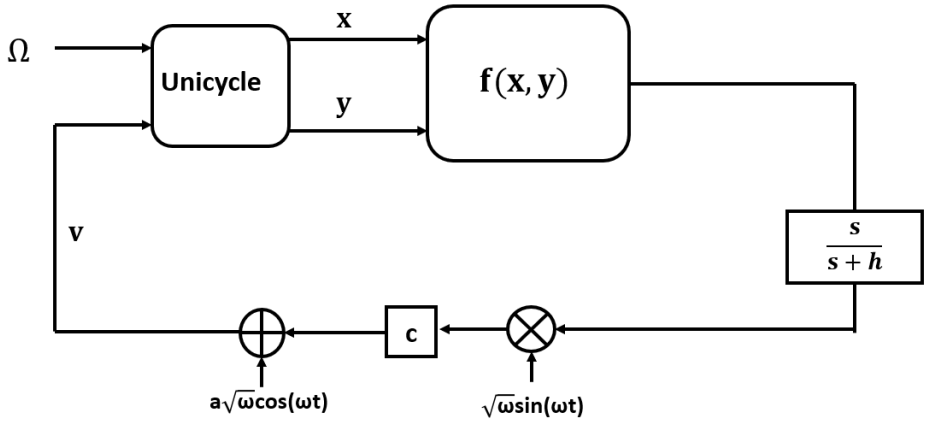

FIGURE 4.2: Unicycle Dynamics CA-ESC Structure.

$$J(x, y) = 10 - 0.5(x - x^*)^2 - 1.5(y - y^*)^2. \tag{4.12}$$

,

It is clear from the equation 4.12, the maximum value of the objective function ($J(x, y)$) is 10, at (x^*, y^*). The value of (x^*, y^*) is $(1, 1)$.

The simulation results of unicycle dynamics are shown 4.9 (from 4.3 to 4.8) and the CA-ESC parameters value are provided in the table 4.1. The states (x, y) graphs are 4.3 and 4.4. The objective function is provided in figure 4.5. The trajectory of the system is 4.6. The high pass filter (HPF), and the control input (linear velocity) plots, are in the figures 4.7 and 4.8.

From the objective function plot, the $J(x, y)$ reaches its extremum (maximum) value in around 50 seconds as states converge to (x^*, y^*) and then oscillate around it. From the trajectory of the system, we can observe that as the system reaches its extremum position, it revolves around it. The angular velocity for this simulation is set at 3 rad/sec.

Parameters	Values
ω	80
c	0.3
a	1
h	1

TABLE 4.1: CA-ESC parameters (for simulation).

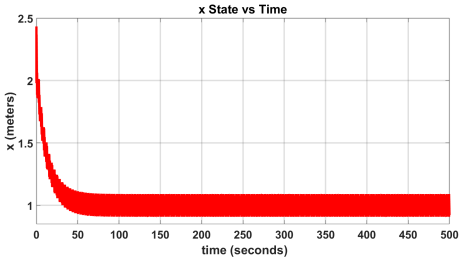

FIGURE 4.3: (a)

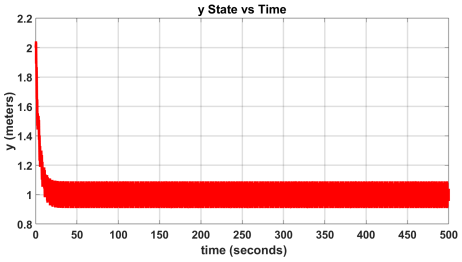

FIGURE 4.4: (b)

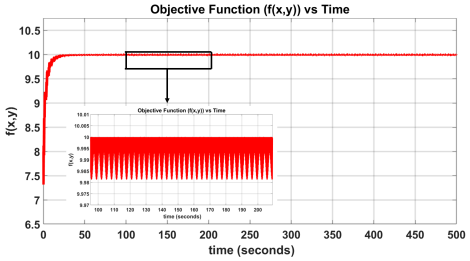

FIGURE 4.5: (c)

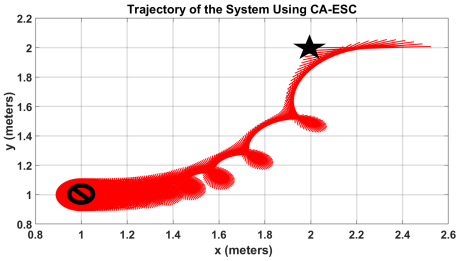

FIGURE 4.6: (d)

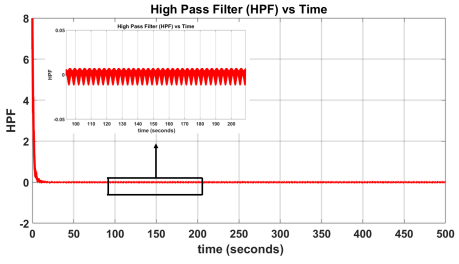

FIGURE 4.7: (e)

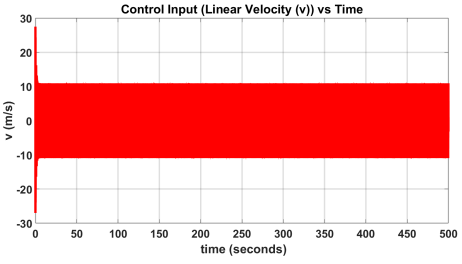

FIGURE 4.8: (f)

FIGURE 4.9: Simulation results of unicycle dynamics using a known mathematical expression of the objective function to access the measurements: (a) x state, (b) y state, (c) Objective function (d) Planar trajectory of the system, (e) HPF, and (f) Linear velocity.

4.3.1.2 Experimental Result

4.3.1.3 With Known Objective Function

In this section, we discuss the observations of the experiment that we conducted using unicycle dynamics. For this experiment, we have access to the mathematical expression of the objective function. For feasibility, we used the same objective function as in the simulation which is given as:

$$J(x, y) = 10 - 0.5(x - x^*)^2 - 1.5(y - y^*)^2, \tag{4.13}$$

where (x^*, y^*) is set as $(1, 1)$.

The experimental results are provided in figure 4.16 (from 4.10 to 4.15) and the CA-ESC parameters are provided in the table 4.2. The states (x, y) vs time plots are plotted in 4.10 and 4.11. The objective function graph is presented in 4.12. The trajectory of TB3 is depicted in 4.13. The HPF and control input (linear velocity) graphs are shown in 4.14 and 4.15.

From the objective function graph, the value of $J(x, y)$ reaches its extremum (maximum) in around 1900 seconds. The states graph shows that the states of TB3 converge to (x^*, y^*) and then oscillate around it. TB3 starts from its initial position $(2, 2)$ and moves to its terminal position which is the extremum of the objective function. The experimental video is provided in [59].

Parameters	Values
ω	80
c	5
a	1
h	1

TABLE 4.2: CA-ESC parameters (for known objective function experiment).

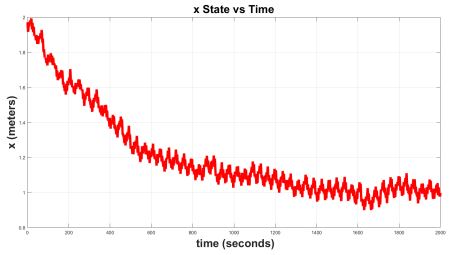

FIGURE 4.10: (a)

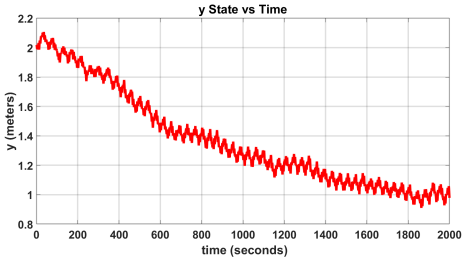

FIGURE 4.11: (b)

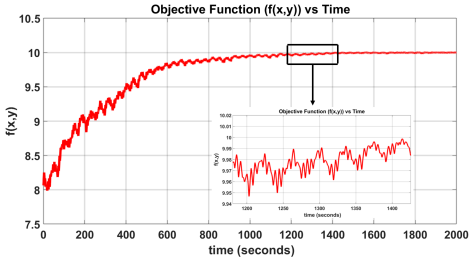

FIGURE 4.12: (c)

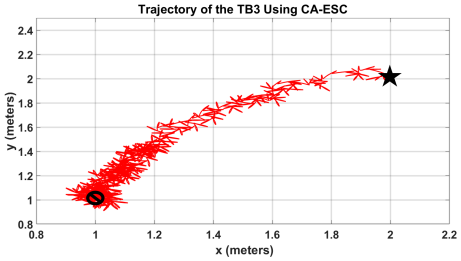

FIGURE 4.13: (d)

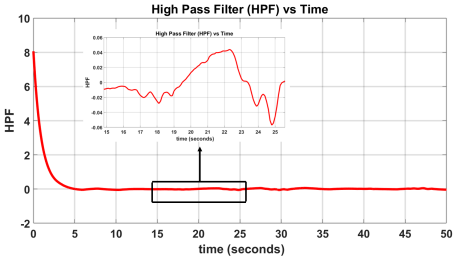

FIGURE 4.14: (e)

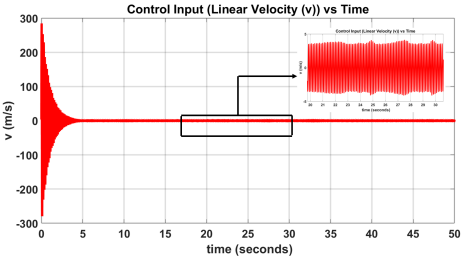

FIGURE 4.15: (f)

FIGURE 4.16: Experimental results of unicycle dynamics using a known mathematical expression of the objective function to access the measurements: (a) x state, (b) y state, (c) Objective function (d) Planar trajectory of the system, (e) HPF, and (f) Angular velocity.

4.3.1.4 With unknown Objective Function

In this subsection, we discuss the light source-seeking experiment using unicycle dynamics. For this experiment, we used the same setup as in the experiment using C-ESC.

The experimental results are provided in figure 4.20 (from 4.17 to 4.19) and the tuning parameters are given in the table 4.3. The states graphs are presented in figures 4.17 and 4.18. The trajectory of TB3 is shown in figure 4.19.

The initial position of TB3 is $(1.6, 0.5)$ and the maximum intensity of the light in the plane is approximately at $(1, 1.7)$. From the trajectory plot, we can observe that TB3 moves towards the extremum position (maximum intensity). The states of the TB3 converge close to their (x^*, y^*) and then oscillate around it. The experimental video is provided in [60].

Parameters	**Values**
ω	100
c	1
a	0.55
h	6

TABLE 4.3: CA-ESC parameters (for light source seeking experiment).

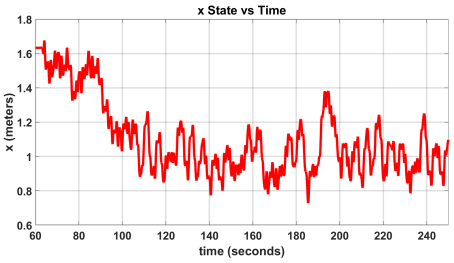

FIGURE 4.17: (a)

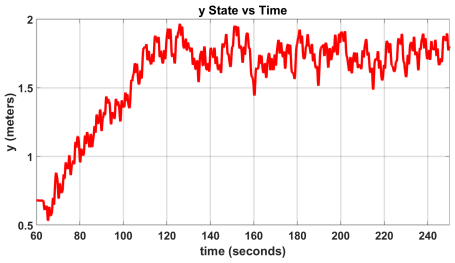

FIGURE 4.18: (b)

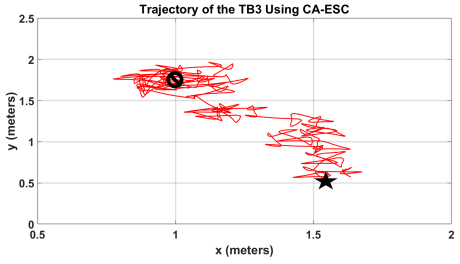

FIGURE 4.19: (c)

FIGURE 4.20: Light source seeking experimental results using unicycle design. (a) x state (b) y state and (c) Trajectory of TB3.

4.3.2 Single-Integrator Dynamics

Here, we have a similar situation as in 4.3.1. Figure 4.21 represents the single-integrator dynamics CA-ESC structure. There are two ESC loops for each state (x, and y).

The dynamics of the system are:

$$\dot{x} = c_x(f(x,y) - x_e h)\sqrt{\omega}sin(\omega t) + a_x\sqrt{\omega}cos(\omega t), \tag{4.14}$$

$$\dot{y} = -c_y(f(x,y) - x_e h)\sqrt{\omega}cos(\omega t) + a_y\sqrt{\omega}sin(\omega t), \tag{4.15}$$

$$\dot{x_e} = -x_e h + f(x,y) \tag{4.16}$$

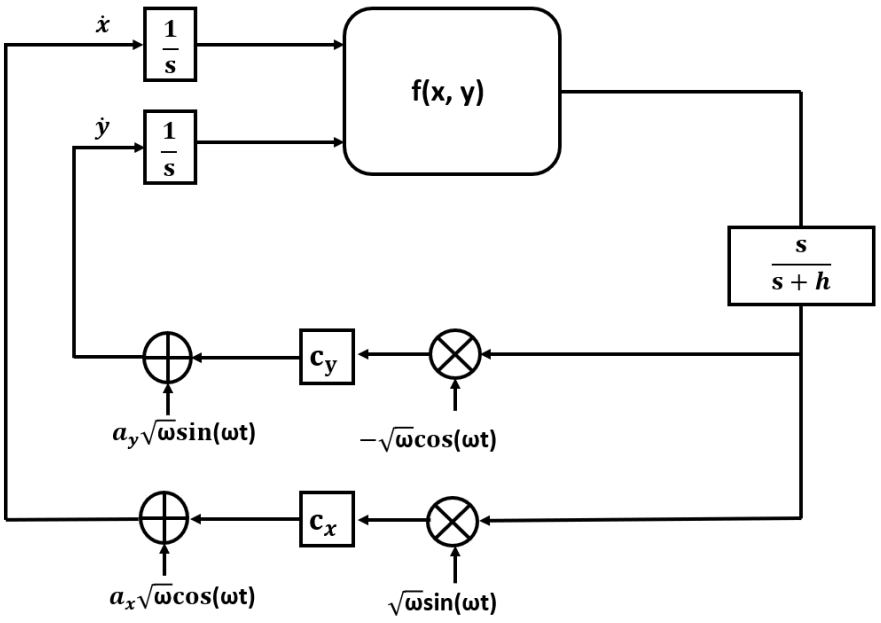

FIGURE 4.21: Single-Integrator Dynamics CA-ESC Structure.

4.3.2.1 Simulation Results

We chose the same objective function as in the unicycle dynamics case provided:

$$J(x, y) = 10 - 0.5(x - x^*)^2 - 1.5(y - y^*)^2, \tag{4.17}$$

,

where (x^*, y^*) is equal to $(1, 1)$.

The simulation results are provided in figure 4.29 (from 4.22 to 4.28) and the CA-ESC parameters are given in the table 4.4. The states plots are presented in 4.22 and 4.23. The objective function and the system trajectory are shown in figures 4.24 and 4.25. The control inputs for each state are provided in 4.26 and 4.27. The HPF graph is shown in figure 4.28.

From the objective function plot, the $J(x, y)$ converges to its maximum value in around 40 seconds as the states of the system converge to (x^*, y^*) and then oscillate around it. The trajectory of the system shows that the system revolves around its extremum position once it reaches there.

Parameters	**Values**
ω	80
c	0.3
a_x	1
a_y	1
h	1

TABLE 4.4: CA-ESC parameters (for simulation).

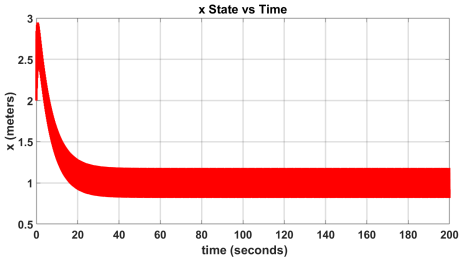

FIGURE 4.22: (a)

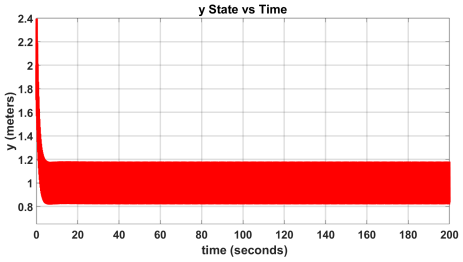

FIGURE 4.23: (b)

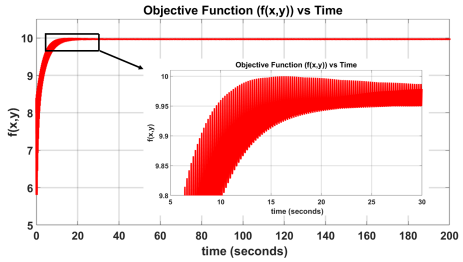

FIGURE 4.24: (c)

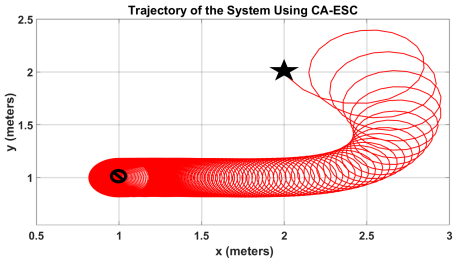

FIGURE 4.25: (d)

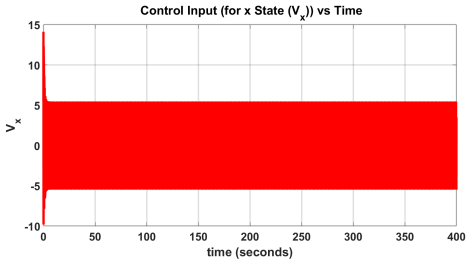

FIGURE 4.26: (e)

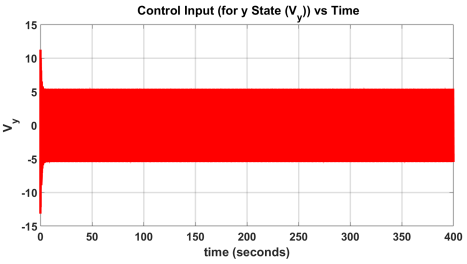

FIGURE 4.27: (f)

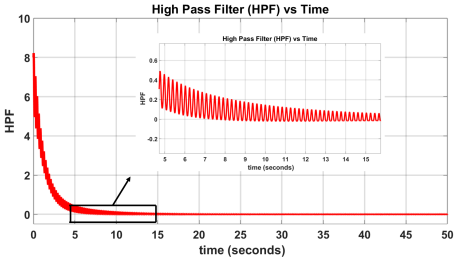

FIGURE 4.28: (f)

FIGURE 4.29: Simulation results of single-integrator dynamics using a known mathematical expression of the objective function to access the measurements: (a) x state, (b) y state, (c) Objective function (d) Planar trajectory of the system, (e) and (f) Represent the control input in the respective states and (g) HPF.

4.3.2.2 Experimental Results

4.3.2.3 With Known Objective Function

Here, we discuss the results of the experiment we performed using single-integrator dynamics. For this experiment, we utilize the same mathematical expression as we have used earlier in simulations and experiments, which is given by:

$$J(x, y) = 10 - 0.5(x - x^*)^2 - 1.5(y - y^*)^2, \tag{4.18}$$

where (x^*, y^*) is equal to $(1, 1)$.

The experimental results are presented in figure 4.36 (from 4.30 to 4.35) and the CA-ESC parameter values are given in the table 4.5. The objective function and the trajectory of TB3 are shown in figures 4.32 and 4.33. The HPF and angular velocity are plotted in figures 4.34 and 4.35. For the experiment, we used constant linear velocity with a value of 0.22 m/s.

The objective function ($J(x, y)$) converges to its maximum value in around 2800 seconds. TB3 reaches its extremum position and revolves around it. The states of the TB3 reach (x^*, y^*) and then oscillate around it. The video is provided in [61].

Parameters	**Values**
ω	30
c	0.5
a	0.137
h	0.005

TABLE 4.5: CA-ESC parameters (for known objective function experiment).

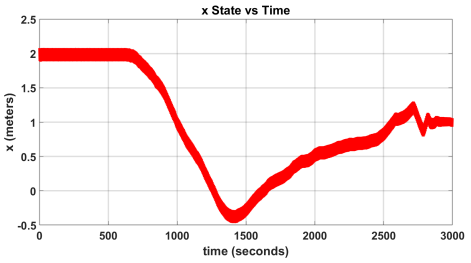

FIGURE 4.30: (a)

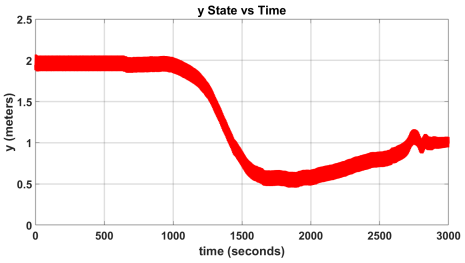

FIGURE 4.31: (b)

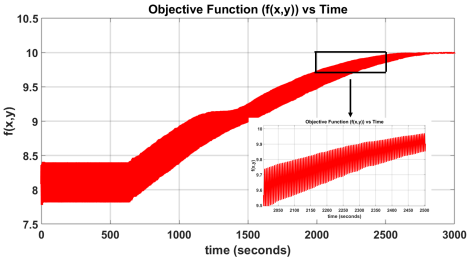

FIGURE 4.32: (c)

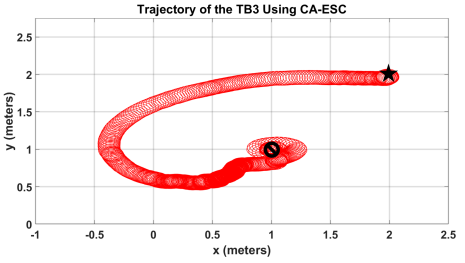

FIGURE 4.33: (d)

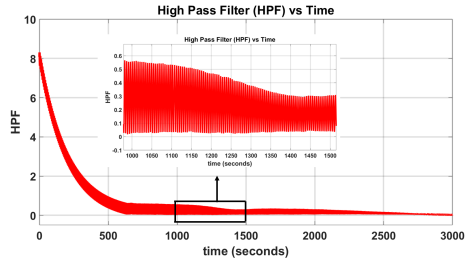

FIGURE 4.34: (e)

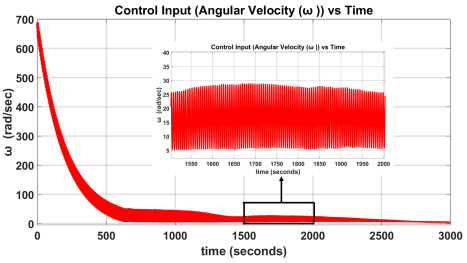

FIGURE 4.35: (f)

FIGURE 4.36: Experimental results of single-integrator dynamics using a known mathematical expression of the objective function to access the measurements: (a) x state, (b) y state, (c) Objective function (d) Planar trajectory of TB3, (e) HPF, and (f) Angular velocity.

4.3.2.4 Light Source-Seeking with Single-Integrator Dynamics

We perform a light source-seeking experiment using single-integrator dynamics with CA-ESC structure. We have only access to the measurement of the light intensity.

The experimental results are provided in figure 4.40 (from 4.37 to 4.39) and the CA-ESC parameters are provided in the table 4.6. The states graphs are presented in figures 4.37 and 4.38. The trajectory of TB3 is shown in figure 4.39.

The initial position of TB3 is $(0, 3)$, and the maximum light intensity is at approximately $(1.3, 1)$. From the trajectory graph, it is observable that despite the presence of noises and sensitivities to the environment, TB3 moves towards the extremum (here towards maximum light intensity). The states of the TB3 converge close to $(1.3, 1)$ and then oscillate around it. The experimental video is provided in [62].

Parameters	**Values**
ω	30
c	1
a	0.027
h	1.5

TABLE 4.6: CA-ESC parameters (for light source seeking experiment).

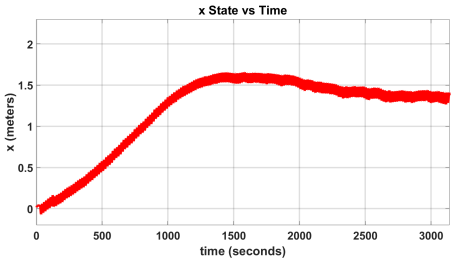

FIGURE 4.37: (a)

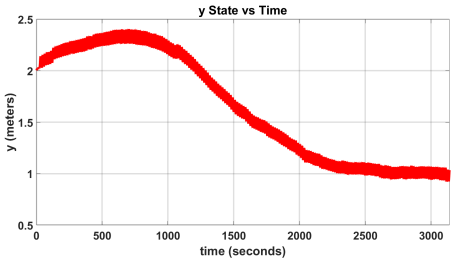

FIGURE 4.38: (b)

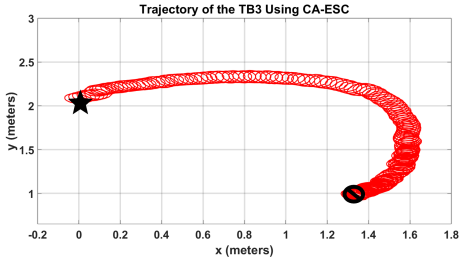

FIGURE 4.39: (c)

FIGURE 4.40: Light source seeking experimental results using single-integrator design. (a) x state (b) y state and (c) Trajectory of TB3.

4.4 Improving Convergence Rate and Attenuating Oscillations by Geometric-Based Extended Kalman Filter

In previous sections, we have discussed the CA-ESC systems and performed some simulations and experiments using single-integrator and unicycle dynamics. Even though CA-ESC performed well in driving the system/TB3 towards the extremum of the objective function, there are persisting oscillations in the system/robot after reaching the extremum which is undesirable in some applications.

To address this issue, we provide the method to attenuate the oscillations of the system by using the concept of GEKF [28], to get the estimation of the gradient of an objective function. Additionally, we propose our amended CA-ESC design comprised of generalized simple CA-ESC, GEKF, and adaptation law to attenuate the oscillation and then we perform some simulations (using single-integrator and unicycle dynamics) to demonstrate the effectiveness of our proposed design.

4.4.0.1 Chen-Fliess Functional Expansion

The Chen-Fliess functional expansion [63] offers a representation of the input-output behavior of a nonlinear system governed by control-affine differential equation (4.1) and the associated output function $y = h(\mathbf{x})$, $y \in \mathbb{R}$. Before discussing the Chen-Fliess expansion, we first need to define the concept of the iterated integral of a set of functions. Consider a fixed time value T and real-valued piecewise continuous functions $u_1, ..., u_m$ defined on the interval [0, T]. Now, iterated integral for each multi-index $(i_k, ..., i_0)$ is defined as

$$\int_0^t d\xi_{i_k} ... d\xi_{i_0} = \int_0^t d\xi_{i_k}(\tau) \int_0^t d\xi_{i_{k-1}} ... \int_0^t d\xi_{i_0}, \tag{4.19}$$

where $0 \leq t \leq T$; $\xi_0(t) = t$; $\xi_i(t) = \int_0^t u_i(\tau)d\tau$ for $1 \leq i \leq m$. Now, Chen-Fliess expansion (4.20) provides the evolution of the output $y(t)$ for a short time $t \in [0, T]$ as

$$y(t) = h(\mathbf{x}_0) + \sum_{k=0}^{\infty} \sum_{i_0,\ldots,i_k=0}^{m} L_{\mathbf{b}_{i_0}} \ldots L_{\mathbf{b}_{i_k}} h(\mathbf{x}_0) \int_0^t d\xi_{i_k} \ldots d\xi_{i_0}, \tag{4.20}$$

where $L_{\mathbf{b}}h$ is the Lie derivative of h along $\mathbf{b}$, i.e. $L_{\mathbf{b}}h = \nabla h \cdot \mathbf{b}$.

4.5 Main Results

In this section, we go through the method to attenuate oscillation in the CA-ESC systems explained in [28]. The method involves the estimation of the gradient, for which we use GEKF presented in literature [28]. The generalized class we consider is similar to the one discussed in [9], encompassing most of the significant CA-ESC systems documented in literatures [8, 9, 64–67]. This generalized structure represents a drift-less system that consists of two control inputs or dither signals, governed by a generalized dynamical equation expressed as follows:

$$\dot{\mathbf{x}} = \sum_{i=1}^{n} \left(b_{1i}(f(\mathbf{x}))\sqrt{\omega}u_{1i} + b_{2i}(f(\mathbf{x}))\sqrt{\omega}u_{2i} \right) e_i, \tag{4.21}$$

where $f \in C^2 : \mathbb{R}^n \to \mathbb{R}$ is the objective function, e_i denotes the i^{th} unit vector in $\mathbb{R}^n$, b_{1i} and b_{2i} are the vector fields associated with control inputs u_{1i} and u_{2i}, and ω is the frequency of input signal . The corresponding LBS for this multi-variable case, similar to [9], can be found as

$$\dot{\mathbf{z}} = -\sum_{i=1}^{n} \nu_{2i,1i} \frac{\partial f(\mathbf{z})}{\partial z_i} b_{fi}(f(\mathbf{z}))e_i, \tag{4.22}$$

with some $b_{fi} : \mathbb{R} \to \mathbb{R}$.

4.5.1 Method for Attenuation of Oscillations for ESC systems

The LBS provided in equation 4.22 gives an approximation that captures and characterizes the behavior of the ESC system in 4.21. It is evident that, except for the gradient components, the LBS derived in 4.22 can be obtained by utilizing measurements of the objective function $f(\mathbf{x})$. Consequently, the estimation of the LBS relies on the gradient of the objective function. This leads to the concept of the estimated LBS, denoted as $\hat{\mathbf{z}}$, which can be defined as follows:

$$\dot{\hat{\mathbf{z}}} = -\sum_{i=1}^{n}(\nu_{2i,1i}\frac{\partial f(\mathbf{z})}{\partial z_i}b_{0i}(f(\mathbf{z})) + \eta_i(t))e_i = \mathbf{J}(t,\mathbf{z}). \tag{4.23}$$

The estimated LBS in 4.23 is similar to 4.22 but with the addition of an error term $\eta_i(t)$ that accounts for the inaccuracies introduced during the estimation process. In literature [28], an additional assumption imposed on $\eta_i(t)$:

A4. $\eta_i(t) : \mathbb{R} \to \mathbb{R}, i = 1, ..., n$ is measurable function and there exist constants $\theta_0, \epsilon_0 \in (0, \infty)$ such that $|\eta_i(t_2) - \eta_i(t_1)| \leq \theta_0|t_2 - t_1|$ for all $t_1, t_2 \in \mathbb{R}$ and $\sup_{t\in\mathbb{R}}|\eta_i(t)| \leq \epsilon_0$. Furthermore, $\lim_{t\to\infty}\eta_i(t) = 0$.

We now utilize an ESC structure that couples the control-affine ESC in 4.21 with an adaptation law for the amplitude of the control input which depends on the estimated LBS as:

$$\dot{\mathbf{x}} = \sum_{i=1}^{n}\left(b_{1i}(f(x))\sqrt{\omega}a_i(t)\hat{u}_{1i} + b_{2i}(f(x))\sqrt{\omega}a_i(t)\hat{u}_{2i}\right)e_i, \tag{4.24}$$

$$\dot{\mathbf{a}} = \sum_{i=1}^{n}\left(-\lambda_i(a_i(t) - J_i(t,\mathbf{x}))\right)e_i, \tag{4.25}$$

where $a_i \in \mathbb{R}$ is the amplitude of the input signal, $\lambda_i > 0 \in \mathbb{R}$ is a tuning parameter, and $u_i = a_i \hat{u}_i$. The design proposed in [28] is shown in figure 4.41. The unshaded area is the same as the generalized CA-ESC system ($b_d = 0$) in [9]. The above-mentioned result for

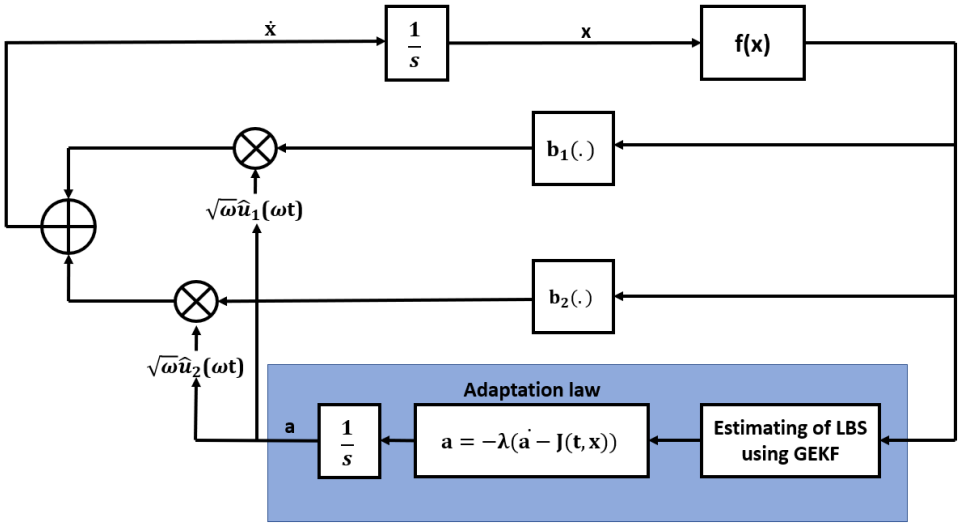

FIGURE 4.41: CA-ESC structure with GEKF and adaptation law.

attenuation of oscillation in this subsection depends on the knowledge of the estimated LBS. So, in the next subsection, we describe the method to obtain the estimated LBS using a Geometric-based Kalman Filter (GEKF).

4.5.2 Geometric-Based Extended Kalman Filter (GEKF) for gradient and Lie Bracket Estimation in Control Affine Extremum-Seeking Control (CA-ESC) Systems

The use of estimation theory tools for estimating the gradient of an unknown objective function is a well-defined concept in the literature on ESC. Since the system's gradient changes over time, the states of the estimation model need to be updated. The Kalman Filter (KF) works well for this purpose, as they propagate over time [28]. However, these methods are confined to traditional ESC structures like C-ESC, for more details, refer to the literature [28]. The KF cannot be applied accurately to CA-ESC systems

since it is not natural to them. Here, we introduce a GEKF which is a natural technique that accurately approximates the gradient of an objective function of CA-ESC systems. This technique depends on the Chen-Fliess series. To show the utility of the Chen-fliess series, we rewrite the equation 4.21 as:

$$\dot{\mathbf{x}} = \mathbf{b}_1\bar{u}_1 + \mathbf{b}_2\bar{u}_2, \tag{4.26}$$

with $\bar{u}_1 = \sqrt{\omega}u_1$ and $\bar{u}_2 = \sqrt{\omega}u_2$. We set the output of the system 4.26 as the objective function $y = f(\mathbf{x})$. Then, writing the Chen-Fliess series expansion 4.20 for the output $y|_{t_2}$ and truncating after first-order terms, we get

$$y|_{t_2} = y|_{t_1} + L_{\mathbf{b}_1}y|_{t_1}\int_{t_1}^{t_2}\bar{u}_1 d\tau + L_{\mathbf{b}_2}y|_{t_1}\int_{t_1}^{t_2}\bar{u}_2 d\tau + O(\Delta t^2), \tag{4.27}$$

where $t_2 = t_1 + \Delta t$, Δt is a short time period/step, and $O(\Delta t^2)$ is the remainder of the expansion. The symbol $(\cdot)|_t$ denotes the evaluation of a given term at time t. As per the Chen-Fliess series 4.20, the assumptions A1-A4 guarantee the mathematical posedness of 4.27. Recall from 4.21 that $\mathbf{b}_1$ and/or $\mathbf{b}_2$ are functions of the output $y = f(\mathbf{x})$. While any control input satisfying assumptions A3 can be used here, we consider a sinusoidal input control-affine ESC system such that $u_1 = \cos(\omega t)$ and $u_2 = \sin(\omega t)$. Then,

$$\begin{aligned}\int_{t_1}^{t_2}\bar{u}_1 d\tau &= \int_{t_1}^{t_2}\sqrt{\omega}\cos(\omega t)d\tau = \frac{1}{\sqrt{\omega}}\left(\sin(\omega t_2) - \sin(\omega t_1)\right)\\ &= \frac{2}{\sqrt{\omega}}\cos(\omega t)\sin\left(\frac{\omega\Delta t}{2}\right) = K\cos(\omega t),\end{aligned} \tag{4.28}$$

where $t = (t_1 + t_2)/2$ and $K = (2/\sqrt{\omega})\sin(\omega\Delta t/2)$. Similarly,

$$\begin{aligned}\int_{t_1}^{t_2} \bar{u}_2 d\tau &= \int_{t_1}^{t_2} \sqrt{\omega}\sin(\omega t) d\tau = -\frac{1}{\sqrt{\omega}}\left(\cos(\omega t_2) - \cos(\omega t_1)\right) \\ &= \frac{2}{\sqrt{\omega}}\sin(\omega t)\sin\left(\frac{\omega\Delta t}{2}\right) = K\sin(\omega t).\end{aligned} \tag{4.29}$$

So, the equation 4.27 can be presented as:

$$y|_{t_2} = y|_{t_1} + L_{\mathbf{b}_1} y|_{t_1} K\cos(\omega t) + L_{\mathbf{b}_2} y|_{t_1} K\sin(\omega t) + O(\Delta t^2). \tag{4.30}$$

Here, the expression describes how the objective function value can be calculated at future time t_2, depending on the control inputs, objective function value and Lie derivatives of the objective function calculated at given time t_2.

Now, we utilize the equation 4.30 and continuous-discrete extended KF to formulate GEKF mentioned in [28] which applies to CA-ESC systems. GEKF effectively works in real-time and is considered as a discrete-continuous extended KF that requires: (i) a measurement equation, which is the discrete part that relates the parameter to be estimated (i.e., the gradient of the objective function, $\nabla f(x, y)$ in our case) with measurements, we have access to (e.g., measurements of the objective function $f(x, y)$ in our case); and (ii) the propagation model, which is the continuous part that governs the filter propagation continuously with time between the measurements [28, 47]. As discussed earlier, LBSs are approximations of the CA-ESC systems and they are formulated in terms of gradient. The states of the GEKF are given as [28]:

$$\bar{\mathbf{X}} = \begin{bmatrix} [\bar{\mathbf{x}}_1]_{n\times 1} \\ [\bar{\mathbf{x}}_2]_{n\times 1} \\ [\bar{x}_3]_{1\times 1} \end{bmatrix} = \begin{bmatrix} -\sum_{i=1}^{n} \alpha_i \nu_{2i,1i} \frac{\partial f(\mathbf{x})}{\partial x_i} b_{fi}(f(\mathbf{x}))e_i \\ \dot{\bar{\mathbf{x}}}_1 \\ f(\mathbf{x})|_{t_1} \end{bmatrix}, \tag{4.31}$$

where $\bar{\mathbf{x}}_1$ is proportional to the right hand side of the LBS 4.22 and α_i is a proportionality constant. Between every two successive measurements, we assume $\bar{\mathbf{x}}_1$ has a constant derivative given by $\bar{\mathbf{x}}_2$. In addition, the state $\bar{x}_3$ is taken as the measured value of the objective function which is updated during the measurement update. The propagation equation for the state $\bar{\mathbf{X}}$ is given by:

$$\dot{\bar{\mathbf{X}}} = \begin{bmatrix} \bar{\mathbf{x}}_2 \\ \mathbf{0}_{n\times 1} \\ 0 \end{bmatrix} + \mathbf{\Omega}, \tag{4.32}$$

where $\mathbf{\Omega}$ represents the process noise. It is assumed to be Gaussian white noise, $\mathbf{\Omega} \sim N(0, \mathbf{Q})$. The Jacobian matrix A associated with the state dynamics can be obtained using the equation 4.32 as:

$$A = \begin{bmatrix} \mathbf{0}_{n\times n} & \mathbf{I}_{n\times n} & \mathbf{0}_{n\times 1} \\ \mathbf{0}_{n\times n} & \mathbf{0}_{n\times n} & \mathbf{0}_{n\times 1} \\ \mathbf{0}_{1\times n} & \mathbf{0}_{1\times n} & \mathbf{0}_{1\times 1} \end{bmatrix}, \tag{4.33}$$

where $\mathbf{0}_{n\times n}$ and $\mathbf{I}_{n\times n}$ represent a zero matrix and an identity matrix of size $n\times n$, respectively. Now, the measurement update, 4.30, obtained using the Chen-Fliess expansion,

can be written as:

$$\begin{aligned} y|_{t_2} &= \mathbf{h}(\bar{\mathbf{X}}) + \nu(t) \\ &= y|_{t_1} + L_{\mathbf{b}_1} y|_{t_1} K\cos(\omega t) + L_{\mathbf{b}_2} y|_{t_1} K\sin(\omega t) + \nu(t) \\ &= f(\mathbf{x})|_{t_1} + (\nabla f(\mathbf{x})\cdot\mathbf{b}_1)|_{t_1} K\cos(\omega t) + (\nabla f(\mathbf{x})\cdot\mathbf{b}_2)|_{t_1} K\sin(\omega t) + \nu(t) \end{aligned} \tag{4.34}$$

where the remainder terms are assumed Gaussian measurement noise $\nu(t) \sim N(0, r)$. Furthermore, the process noise and the measurement noise are assumed to be uncorrelated. The algorithm of GEKF is provided in Algorithm 1.

Algorithm 1 Geometric-based Continuous-discrete Extended Kalman Filter

Initialize $\bar{\mathbf{X}}$ with some initial values.
choose a sample rate as an output T_{out} less than the sensors rates (used for measurements).
For each sample time T_{out}:
for $i = 1$ to N do (Prediction Step)
$\bar{\mathbf{X}} = \bar{\mathbf{X}} + (T_{out}/N)\dot{\bar{\mathbf{X}}}$
$\mathbf{P} = \mathbf{P} + \frac{T_{out}}{N}(\mathbf{AP} + \mathbf{PA} + \mathbf{Q})$,
once the measurement is received i then
(Measurement step)
$\mathbf{C} = \frac{\partial \mathbf{h}}{\partial \bar{\mathbf{X}}}(\bar{\mathbf{X}})$
$\mathbf{L} = \mathbf{PC}^T(\mathbf{R} + \mathbf{CPC}^T)^{-1}$
$\mathbf{P} = (\mathbf{I} - \mathbf{LC})\mathbf{P}$
$\bar{\mathbf{X}} = \bar{\mathbf{X}} + \mathbf{L}(\mathbf{y}[n] - \mathbf{h}(\bar{\mathbf{X}}))$
end if

4.6 Simulation Results

4.6.1 Unicycle Dynamics

In this section, we propose our novel and amended design of the simple unicycle CA-ESC structure case depicted in figure 4.42 with the following control law similar to [8]:

$$\dot{\mathbf{x}} = f(\mathbf{x})\sqrt{\omega}u_1(t) + a\sqrt{\omega}u_2(t) \tag{4.35}$$

where $\mathbf{x} \in \mathbb{R}$ is the state vector, u_1 is $\sin(\omega t)$, and u_2 is $\cos(\omega t)$. The LBS corresponding to equation 4.35 is given as:

$$\dot{z} = \frac{[f, 1]}{2} = -\frac{\delta f}{\delta z} = J \tag{4.36}$$

For the purpose of simulation, the objective function is given as:

$$J(x, y) = 10 - 0.5(x - x^*)^2 - 1.5(y - y^*)^2, \tag{4.37}$$

where x^*, and y^* are constants with values $(1, 1)$. The states of the filter are given as:

$$\bar{\mathbf{X}} = \begin{bmatrix} \bar{\mathbf{x}}_1 \\ \bar{\mathbf{x}}_2 \\ \bar{x}_3 \end{bmatrix} = \begin{bmatrix} -\frac{\alpha \nabla f(x)|_{t_1}}{2} \\ \dot{\bar{\mathbf{x}}}_1 \\ f(\mathbf{x})|_{t_1} \end{bmatrix}, \tag{4.38}$$

where α is constant and $\alpha = K = (2\sqrt{\omega})sin(\omega \Delta t/2)$. Now, the propagation equation as per 4.32 is given as:

$$\dot{\bar{\mathbf{X}}} = \begin{bmatrix} \bar{\mathbf{x}}_2 \\ 0 \\ 0 \end{bmatrix} + \mathbf{\Omega}, \tag{4.39}$$

and the Jacobian matrix A associated with the propagation equation can be obtained following 4.33

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.40}$$

Now, we utilize 4.34 for measurement update, we and find the Lie derivatives as $L_{\mathbf{b}_1}|t_1 = (\Delta f(x) f(x))|_{t_1}$, $L_{\mathbf{b}_2}|t_1 = (\alpha \Delta f(x))|_{t_1}$. The measurement update equation can be presented as:

$$\begin{aligned} f(x)\Big|_{t_2} &= f(x)\Big|_{t_1} + (\Delta f(x) f(x))\Big|_{t_1} K\cos(\omega t) + (\alpha \Delta f(x))\Big|_{t_1} K\sin(\omega t) + \nu(t) \\ &= \bar{x}_3 - 2\bar{x}_3\bar{x}_1\cos(\omega t) - 2a\bar{x}_1\sin(\omega t) + \nu(t) \end{aligned} \tag{4.41}$$

Our proposed design consists of simple CA-ESC, GEKF for gradient estimation, and adaptation law for attenuating oscillations and a better convergence rate is provided in figure 4.42. In the design, HPF is an optional filter, and it can be avoided in the design. The control law for each state and HPF are provided as:

$$\dot{x} = ((cf(x,y) - x_e h)\sqrt{\omega} sin(\omega t) + a\sqrt{\omega} cos(\omega t)) cos(\Omega t) \tag{4.42}$$

$$\dot{y} = ((cf(x,y) - x_e h)\sqrt{\omega} sin(\omega t) + a\sqrt{\omega} cos(\omega t)) sin(\Omega t) \tag{4.43}$$

$$\dot{x_e} = -x_e h + f(x, y), \tag{4.44}$$

These state equations are the same as that we discussed in the subsection 4.3.1. The adaptation law is given as:

$$\dot{a} = -\lambda(a - J(t, x, y)) \tag{4.45}$$

In equation 4.45, the λ is a tuning parameter, $J(t, x, y)$ is the estimation of the RHS of LBS, and a is the amplitude of the modulation signal.

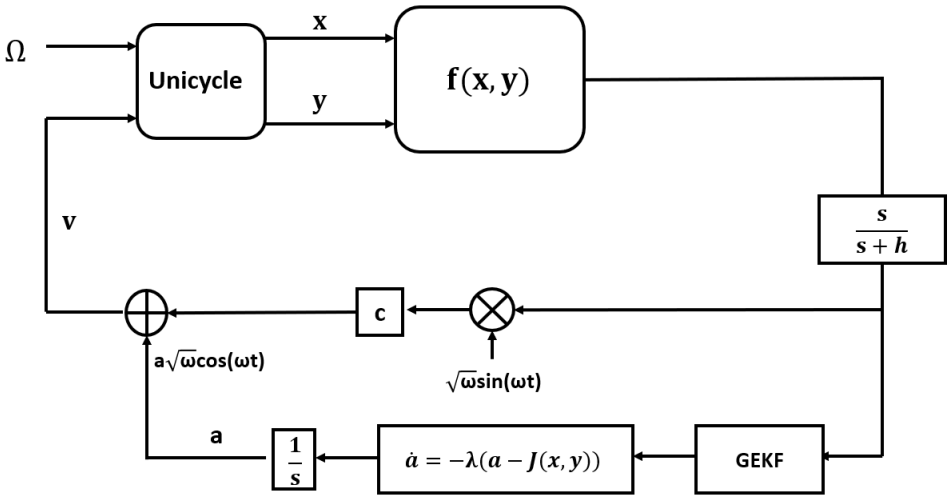

FIGURE 4.42: Proposed Design of CAESC with GEKF and Adaptation law for unicycle dynamics

The simulation results for Unicycle dynamics are provided in figure 4.48, and the CA-ESC parameters are provided in the table 4.7. For the unicycle case, we set the angular velocity (Ω) of the system as 3 rad/sec, and linear velocity is computed from the CA-ESC. The GEKF parameters are set as process noise ($Q = 0.005I$), measurement noise ($R = 0.01$), and sample time $T_s = 0.1$ $seconds$. The λ is set as 0.008, and the initial value of a is set as 1.

Parameters	Values
ω	80
c	0.3
a	1
Ω	3
h	1

TABLE 4.7: CA-ESC simulation parameters for unicycle dynamics.

Here we present the simulation results of our proposed design in figure 4.48 (from 4.43 to 4.47). To show the effectiveness of our proposed design in killing the oscillations and a better convergence rate, we plotted the two results (our proposed design is in red, and the design mentioned in [8] is in blue) for comparison. The states of the system are plotted in figures 4.43 and 4.44. The objective function is shown in figure 4.45. The trajectory of the system is provided in figure 4.47. The control input (linear velocity) is shown in figures and 4.46.

From the trajectory plot, there are attenuating oscillations in the trajectory of the system by using our proposed design (in red), while there are persisting oscillations in the system by using the design (in blue) mentioned in literature [8]. We can observe the same from the states and control input plots. States of the system converge to their optimal point with attenuating oscillations by using the proposed design. The control input converges to zero with reducing oscillations by using our proposed design (in red), on the other side there are continuous oscillations in the control input by using the design (in blue) mentioned in the literature [8].

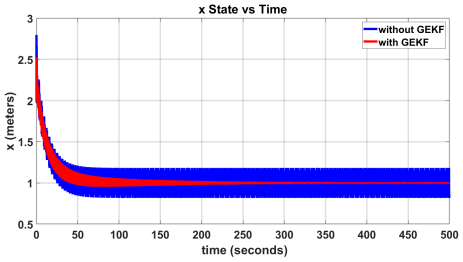

FIGURE 4.43: (a)

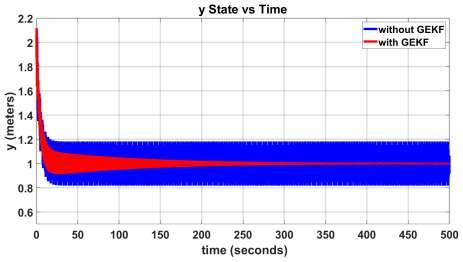

FIGURE 4.44: (b)

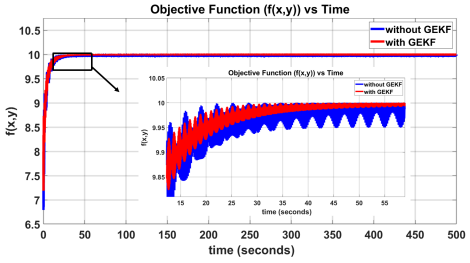

FIGURE 4.45: (c)

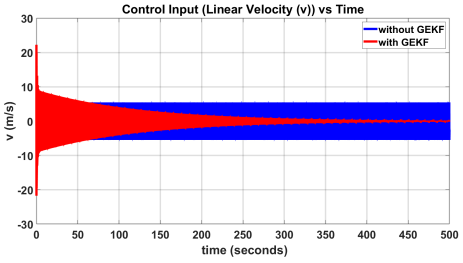

FIGURE 4.46: (d)

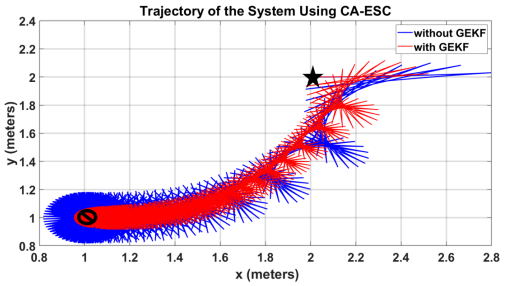

FIGURE 4.47: (e)

FIGURE 4.48: Simulation results of our proposed design with unicycle dynamics using a known mathematical expression of the objective function in which the blue one represents the results using traditional single-integrator design found in the literature, and red one represents results using our design: (a) x state, (b) y state, (c) Objective function (d) Angular velocity and (e) Trajectory of the system.

4.6.2 Single-Integrator Dynamics

In this subsection, we present our novel and amended design of a simple single-integrator CA-ESC structure which stabilizes a system/TB3 about the extremum with attenuated oscillations. In literature, single-integrator designs have been used (i.e., [8]). We introduce a novel single-integrator design based on the design mentioned in [20]. Our proposed design of a single-integrator CA-ESC with attenuating oscillations is shown in figure 4.49. The proposed design which operates in a planar mode (x and y coordinates) can be defined as:

$$\dot{x} = c_x\, f(x,y)\sqrt{\omega}a_x u_1(\omega t) + \sqrt{\omega}u_2(\omega t), \tag{4.46}$$

$$\dot{y} = -c_y\, f(x,y)\sqrt{\omega}a_y u_2(\omega t) + \sqrt{\omega}u_1(\omega t), \tag{4.47}$$

$$\dot{a}_x = -\lambda_x(a_x - J_x(x,y)), \tag{4.48}$$

$$\dot{a}_y = -\lambda_y(a_y - J_y(x,y)), \tag{4.49}$$

where control inputs are $u_1(\omega t) = \sin(\omega t)$, $u_2(\omega t) = \cos(\omega t)$, $f(x,y)$ is the objective function, ω, c_x and $_y$ are the frequency and gains, the amplitude of the input signals for x and y states are $a_x, a_y \in \mathbb{R}$, respectively, $\lambda_x, \lambda_y > 0 \in \mathbb{R}$ are the tuning parameters of adaptation law, and $J_x(x,y)$ and $J_y(x,y)$ are the estimation of the right-hand side of LBS such that [20]:

$$J_x(x,y) = \alpha_1 \nabla_{z_x} f(x,y) + \eta_1(t), \tag{4.50}$$

$$J_y(x,y) = \alpha_2 \nabla_{z_x} f(x,y) + \eta_2(t), \tag{4.51}$$

with α_1, α_2 are constants and η_1, η_2 are the estimation errors of J_x, J_y respectively.

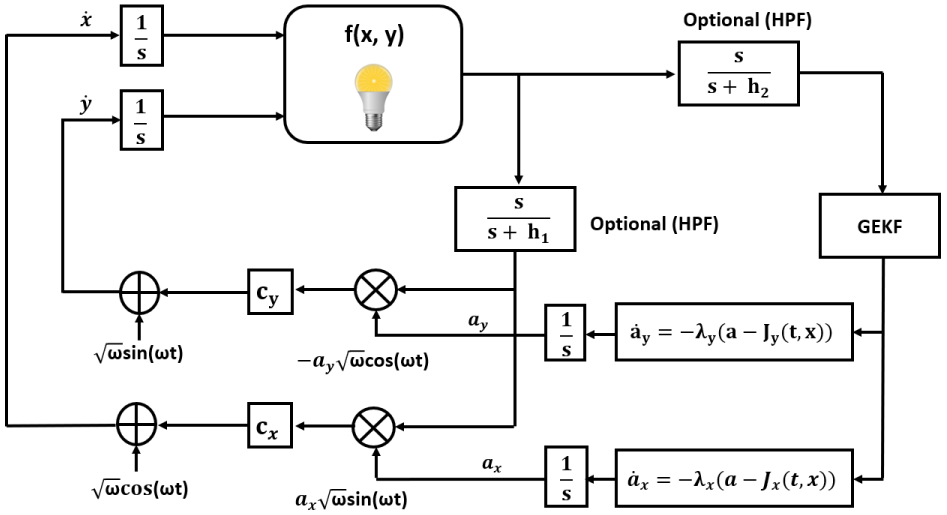

FIGURE 4.49: Proposed Design of CAESC with GEKF and Adaptation law

The GEKF states are formulated as follows:

$$\bar{\mathbf{X}} = \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \bar{x}_3 \\ \bar{x}_4 \\ \bar{x}_5 \end{bmatrix} = \begin{bmatrix} \frac{K}{2} \nabla_x f(x,y)|_{t_1} \\ \frac{K}{2} \nabla_y f(x,y)|_{t_1} \\ \dot{\bar{x}}_1 \\ \dot{\bar{x}}_2 \\ f(x,y)|_{t_1} \end{bmatrix}_{5\times 1}, \tag{4.52}$$

where $\bar{x}_1$ and $\bar{x}_2$ are proportional to the gradient components we are estimating. Following the guidelines provided in [20], we take $K = 2/\sqrt{\omega}\sin(\omega\Delta t/2)$ where Δt is a short time step between the measurements. Now, following a constant velocity propagation model similar to [20], we get:

$$\dot{\bar{\mathbf{X}}} = \begin{bmatrix} \bar{\mathbf{x}}_3 \\ \bar{\mathbf{x}}_4 \\ 0 \\ 0 \\ 0 \end{bmatrix}_{5\times 1} + \boldsymbol{\Omega}, \tag{4.53}$$

where $\boldsymbol{\Omega}$ represents the process noise as a random variable, $\mathbf{Q}$ is the covariance matrix for process noise (system noise) and the Jacobian matrix $\mathbf{A}$ associated with the state dynamics is obtained as

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}_{5\times 5}. \tag{4.54}$$

For the measurement updates, The Chen-Flies series expansion that we discussed earlier in 4.4.0.1 and also provided in [20]. The Chen-Flies expansion can be written as follows:

$$f(\mathbf{x},\mathbf{y})|_{t_2} = f(\mathbf{x},\mathbf{y})|_{t_1} + (\nabla f(\mathbf{x},\mathbf{y}) \cdot \mathbf{b}_1)|_{t_1} K\cos(\omega t) + (\nabla f(\mathbf{x},\mathbf{y}) \cdot \mathbf{b}_2)|_{t_1} K\sin(\omega t) + \nu(t), \tag{4.55}$$

where

$$\mathbf{b}_1 = \left[c_x\sqrt{\omega} f(x,y); \quad a_x\sqrt{\omega}\right], \tag{4.56}$$

$$\mathbf{b}_2 = \left[-c_y\sqrt{\omega} f(x,y); \quad a_y\sqrt{\omega}\right]. \tag{4.57}$$

where $t_2 = t_1 + \Delta t$. The remaining terms follow a Gaussian distribution as measurement noise denoted as $\nu(t) \sim N(0, R)$, and R is the covariance matrix associated with noise measurement. It is further assumed that the process noise and the measurement noise are uncorrelated. Now, the measurement update equation is [47]:

$$\begin{aligned} f(\mathbf{x},\mathbf{y})|_{t_2} &= \mathbf{h}(\bar{\mathbf{X}}) + \nu(t) \\ &= f(\mathbf{x},\mathbf{y})|_{t_1} \\ &\quad + \Big(c_x f(\mathbf{x},\mathbf{y})\nabla_x f(\mathbf{x},\mathbf{y}) + a_x \nabla_y f(\mathbf{x},\mathbf{y})\Big) K\cos(\omega t) \\ &\quad + \Big(a_y \nabla_x f(\mathbf{x},\mathbf{y}) - c_y f(\mathbf{x},\mathbf{y})\nabla_y f(\mathbf{x},\mathbf{y})\Big) K\sin(\omega t) \\ &\quad + \nu(t) \\ &= \bar{x}_5 + 2c_x\bar{x}_1\bar{x}_5\cos(\omega t) + 2a\bar{x}_2\cos(\omega t) \\ &\quad + 2a\bar{x}_1\sin(\omega t) - 2c_y\bar{x}_2\bar{x}_5\sin(\omega t) + \nu(t), \end{aligned} \tag{4.58}$$

where $t = (t_1 + t_2)/2$. The Jacobian matrix associated with the measurement update equation is given by $\mathbf{C}$ matrix:

$$\mathbf{C} = \begin{bmatrix} 2c\bar{x}_5 \cos(\omega t) + 2a_y \sin(\omega t) \\ 2a_x \cos(\omega t) - 2c\bar{x}_5 \sin(\omega t) \\ 0 \\ 0 \\ 1 + 2c\bar{x}_1 \cos(\omega t) - 2c\bar{x}_2 \sin(\omega t) \end{bmatrix}_{5\times 1} \tag{4.59}$$

The simulation results using our proposed design and the design mentioned in the literature [8] are provided in figure 4.56 (from 4.50 to 4.55), and the tuning parameters are provided in the table 4.8. The trajectory and the objective functions are plotted in figures 4.52 and 4.53. The states of the system are provided in figures 4.50 and 4.51. The control inputs (linear and angular velocities) are depicted in figures 4.54 and 4.55.

Parameters	**Values**
ω	30
c	0.3
a_x	1
a_y	1
λ_x	0.015
λ_y	0.0995
h_1	1
h_1	1
Q	$0.05I$
R	0.5

TABLE 4.8: ESC and adaptation law parameters for simulations.

The utilized mathematical expression of the objective function for the simulation is given as:

$$f(x,y) = 10 - \frac{1}{2}(x-1)^2 - \frac{3}{2}(y-1)^2 \tag{4.60}$$

where an extremum (maximum) is $f^* = 10$ at $(x, y) = (1, 1)$.

From the trajectory plot, we can observe that as the system moves towards the extremum of the objective function, there is a significant reduction in the oscillations by using our proposed design (in red), while there are persistent oscillations in the system by using the design mentioned in the literature [8]. We can verify the effectiveness of our proposed design in attenuating the oscillations from the states and control inputs graphs. As the states of the system converge to $(1, 1)$, the oscillations in the states reduce. The control input also converges nearly zero with attenuating oscillations, on the other side, there are continuous oscillations in the control input (in blue) by using the design mentioned in the literature.

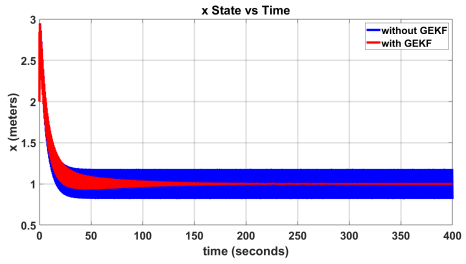

FIGURE 4.50: (a)

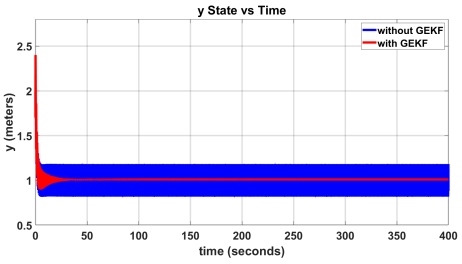

FIGURE 4.51: (b)

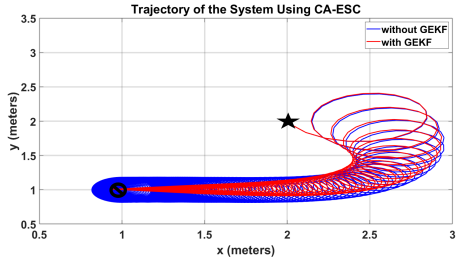

FIGURE 4.52: (c)

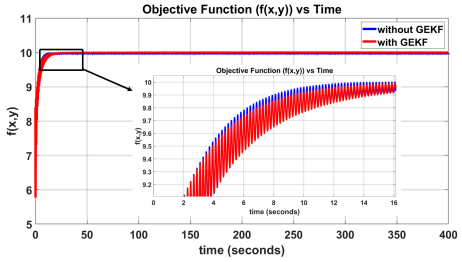

FIGURE 4.53: (d)

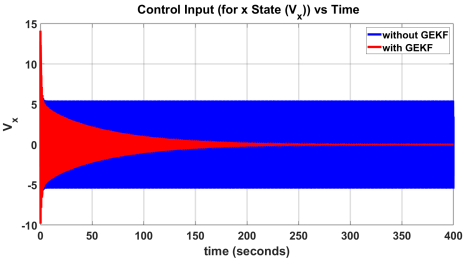

FIGURE 4.54: (e)

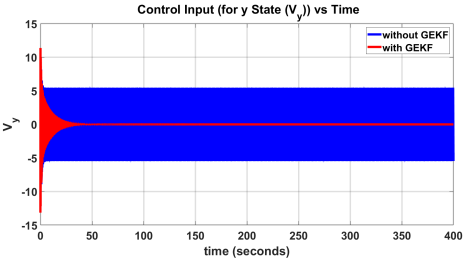

FIGURE 4.55: (f)

FIGURE 4.56: Simulation results of our proposed design using a known mathematical expression of the objective function to access the measurements, in which the blue one represents the results using traditional single-integrator design mentioned in the literature, and red one represents results using our design: (a) x state, (b) y state, (c) Planar trajectory of the system (d) Objective function, (e) and (f) Represent the control input in the respective states.

4.7 Conclusion

In this chapter, we discuss the CA-ESC systems. We conducted simulations using CA-ESC structure with unicycle and single-integrator dynamics presented in [8]. To verify the effectiveness of CA-ESC systems, we performed real-time experiments. We carried out two types of experiments: one in which we utilized the mathematical expression of an objective function and in second which was light source seeking in which we had only access to the measurement of an unknown objective function. We had some observations based on the simulations and experimental results. One of the aspects of ESC is the persistence of oscillations despite the system reaching its extremum (maximum/minimum) which is sometimes undesirable in real applications. We then introduced our novel proposed CA-ESC design with GEKF and adaptation law to attenuate the oscillations as the system converges to the extremum. We showed the efficiency of our proposed amended CA-ESC design by conducting simulations using both single-integrator and unicycle dynamics and then compared the results with the design presented in [8]. From the results, we concluded that our proposed design is significantly efficient in attenuating the oscillations and with a better convergence rate.

Chapter 5

Summary and Future Work

In this thesis, in chapter 2 we addressed the trajectory-tracking problem using traditional controllers: PDC, PPC, and MPC. For the reference trajectory, we used the LBS which generates the optimal trajectory by utilizing the gradient of an objective function. We discussed the ESC system and its classic form (C-ESC) in chapter 3. We conducted simulations and experiments (for both known and unknown objective functions). We discussed the CA-ESC in chapter 4 and performed some simulations and experiments based on the design provided in [8]. Moreover, we introduced our novel CA-ESC design with GEKF and adaptation law for attenuating the oscillations in the system.

5.1 Future Work

The simulation of our proposed design showed promising results. Now, we explore the potential of our proposed CA-ESC design in real applications. We performed experiments using our design for single-integrator dynamics and the results are promising, even our work [47] got accepted in the European Control Conference (ECC) 2024 [47].

Our design performed well in both experiments (with a mathematical expression of an objective function and light source seeking). In the light source-seeking experiment, although there were more noises and sensitivities to the environment, our design performed well. Currently, we are performing experiments using our design with unicycle dynamics. Moreover, a theoretical analysis is required to provide crucial conditions and proof that the utilization of GEKF with CA-ESC systems improves the convergence rate. We will implement a Low Pass Filter (LPF) in our design to improve the performance of our design and then we will use our design to demonstrate its effectiveness in addressing the multi-agent problem. Additionally, we will incorporate the object avoidance algorithm in our proposed design. We will then implement our design on different systems including UAVs, flapper drones, etc.

Bibliography

[1] Cristian Tiriolo, Giuseppe Franzè, and Walter Lucia. A receding horizon trajectory tracking strategy for input-constrained differential-drive robots via feedback linearization. *IEEE Transactions on Control Systems Technology*, 31(3):1460–1467, 2022.

[2] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[3] Turki Y Abdalla and AA Abdulkarem. Pso-based optimum design of pid controller for mobile robot trajectory tracking. *International Journal of Computer Applications*, 47(23):30–35, 2012.

[4] Zejiang Wang, Xingyu Zhou, and Junmin Wang. Extremum-seeking-based adaptive model-free control and its application to automated vehicle path tracking. *IEEE/ASME Transactions on Mechatronics*, 27(5):3874–3884, 2022.

[5] Chao Huang, Chen Lv, Peng Hang, and Yang Xing. Toward safe and personalized autonomous driving: Decision-making and motion control with dpf and cdt techniques. *IEEE/ASME Transactions on Mechatronics*, 26(2):611–620, 2021.

[6] Joonwoo Ahn, Seho Shin, Minsung Kim, and Jaeheung Park. Accurate path tracking by adjusting look-ahead point in pure pursuit method. *International journal of automotive technology*, 22:119–129, 2021.

[7] Julio E Normey-Rico, Ismael Alcalá, Juan Gómez-Ortega, and Eduardo F Camacho. Mobile robot path tracking using a robust pid controller. *Control Engineering Practice*, 9(11):1209–1214, 2001.

[8] Hans-Bernd Dürr, Miloš S Stanković, Christian Ebenbauer, and Karl Henrik Johansson. Lie bracket approximation of extremum seeking systems. *Automatica*, 49 (6):1538–1552, 2013.

[9] Victoria Grushkovskaya, Alexander Zuyev, and Christian Ebenbauer. On a class of generating vector fields for the extremum seeking problem: Lie bracket approximation and stability properties. *Automatica*, 94:151–160, 2018.

[10] Sameer Pokhrel and Sameh A Eisa. Higher order lie bracket approximation and averaging of control-affine systems with application to extremum seeking. *arXiv preprint arXiv:2310.07092*, 2023.

[11] Mohamed H Darwish, Ahmed A Elgohary, Marwan M Gomaa, Ahmed M Kaoud, Ahmed M Ashry, and Haitham E Taha. A comparative assessment of the performance of pid and mpc controllers: A uav altitude hold autopilot case study. In *AIAA SCITECH 2022 Forum*, page 1519, 2022.

[12] Alexander Scheinker. 100 years of extremum seeking: A survey. *Automatica*, 161: 111481, 2024.

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

[14] Kartik B Ariyur and Miroslav Krstic. *Real-time optimization by extremum-seeking control*. John Wiley & Sons, 2003.

[15] Miroslav Krstic and Hsin-Hsiung Wang. Stability of extremum seeking feedback for general nonlinear dynamic systems. *Automatica*, 36(4):595–602, 2000.

[16] Christophe Labar, Emanuele Garone, Michel Kinnaert, and Christian Ebenbauer. Newton-based extremum seeking: A second-order lie bracket approximation approach. *Automatica*, 105:356–367, 2019.

[17] Nima Ghods. *Extremum seeking for mobile robots*. University of California, San Diego, 2011.

[18] Victoria Grushkovskaya, Simon Michalowsky, Alexander Zuyev, Max May, and Christian Ebenbauer. A family of extremum seeking laws for a unicycle model with a moving target: theoretical and experimental studies. In *2018 European Control Conference (ECC)*, pages 1–6. IEEE, 2018.

[19] Sameer Pokhrel and Sameh A Eisa. Gradient and lie bracket estimation of extremum seeking systems: A novel geometric-based kalman filter and relaxed time-dependent stability condition. *International Journal of Control, Automation and Systems*, 21 (12):3839–3849, 2023.

[20] Sameer Pokhrel and Sameh A Eisa. Control-affine extremum seeking control with attenuating oscillations: A lie bracket estimation approach. In *2023 Proceedings of the Conference on Control and its Applications (CT)*, pages 133–140. SIAM, 2023.

[21] Donia Ben Halima Abid, Najah Yousfi Allagui, and Nabil Derbel. Navigation and trajectory tracking of mobile robot based on kinematic pi controller. In *2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pages 252–256. IEEE, 2017.

[22] Kinam Lee, Dae-Yeong Im, Bongwoo Kwak, Young-Jae Ryoo, et al. Design of fuzzy-pid controller for path tracking of mobile robot with differential drive. *International Journal of Fuzzy Logic and Intelligent Systems*, 18(3):220–228, 2018.

[23] Mohamed Elbanhawi, Milan Simic, and R Jazar. Receding horizon lateral vehicle control for pure pursuit path tracking. *Journal of Vibration and Control*, 24(3): 619–642, 2018.

[24] Hua Wang, Xi Chen, Yu Chen, Baoming Li, and Zhonghua Miao. Trajectory tracking and speed control of cleaning vehicle based on improved pure pursuit algorithm. In *2019 Chinese control conference (CCC)*, pages 4348–4353. IEEE, 2019.

[25] Ivan Maurović, Mato Baotić, and Ivan Petrović. Explicit model predictive control for trajectory tracking with mobile robots. In *2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 712–717. IEEE, 2011.

[26] Michael Neunert, Cédric De Crousaz, Fadri Furrer, Mina Kamel, Farbod Farshidian, Roland Siegwart, and Jonas Buchli. Fast nonlinear model predictive control for unified trajectory optimization and tracking. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1398–1404. IEEE, 2016.

[27] ROBOTIS. Turtlebot3 emanual. Online. Retrieved from https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/.

[28] Sameer Pokhrel, Ahmed A Elgohary, and Sameh Eisa. Extremum seeking by multi-agent vehicles and uavs with no steady state oscillation using a geometric-based kalman filtering. In *AIAA SCITECH 2024 Forum*, page 0724, 2024.

[29] Earl A Codington and Norman Levinson. Theory of ordinary differential equations. *New York*, 1955, 1955.

[30] Morris W Hirsch, Stephen Smale, and Robert L Devaney. *Differential equations, dynamical systems, and an introduction to chaos*. Academic press, 2012.

[31] David Ruelle. *Elements of differentiable dynamics and bifurcation theory*. Elsevier, 2014.

[32] Edwin KP Chong, Wu-Sheng Lu, and Stanislaw H Żak. *An Introduction to Optimization: With Applications to Machine Learning*. John Wiley & Sons, 2023.

[33] Gregor Klancar, Andrej Zdesar, Saso Blazic, and Igor Skrjanc. *Wheeled mobile robotics: from fundamentals towards autonomous systems*. Butterworth-Heinemann, 2017.

[34] Siciliano Bruno and Khatib Oussama. Springer handbook of robotics. *Bruno Siciliano, Oussama Khatib*, 2008.

[35] Karl Johan Åström and Tore Hägglund. *Advanced PID control*. ISA-The Instrumentation, Systems and Automation Society, 2006.

[36] William Y Svrcek, Donald P Mahoney, and Brent R Young. *A real-time approach to process control*. John Wiley & Sons, 2014.

[37] R Craig Coulter et al. *Implementation of the pure pursuit path tracking algorithm*. Carnegie Mellon University, The Robotics Institute, 1992.

[38] University of Cincinnati. Uc aeem mdcl website. https://sites.google.com/view/uc-aeem-mdcl/home. Reseach Lab.

[39] Optitrack. Motion capturing system. Online. Retrieved from https://optitrack.com/.

[40] Giovanni Lavezzi, Nathan J Stang, and Marco Ciarcià. Start: A satellite three axis rotation testbed. *Micromachines*, 13(2):165, 2022.

[41] kutzer/OptiTrackToolbox. Matlab toolbox for optitrack natnet sdk. Online. Retrieved from https://www.mathworks.com/matlabcentral/fileexchange/55675-kutzer-optitracktoolbox.

[42] Paul Frihauf, Shu-Jun Liu, and Miroslav Krstic. A single forward-velocity control signal for stochastic source seeking with multiple nonholonomic vehicles. *Journal of Dynamic Systems, Measurement, and Control*, 136(5):051024, 2014.

[43] Jorge I Poveda and Nicanor Quijano. Shahshahani gradient-like extremum seeking. *Automatica*, 58:51–59, 2015.

[44] D Nesić, Ying Tan, William H Moase, and Chris Manzie. A unifying approach to extremum seeking: Adaptive schemes based on estimation of derivatives. In *49th IEEE conference on decision and control (CDC)*, pages 4625–4630. IEEE, 2010.

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

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

[47] Shivam Bajpai, Ahmed A Elgohary, and Sameh A Eisa. Model-free source seeking by a novel single-integrator with attenuating oscillations and better convergence: Robotic experiments. *arXiv preprint arXiv:2311.04330*, 2023.

[48] 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.

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

[50] Hsin-Hsiung Wang and Miroslav Krstic. Extremum seeking for limit cycle minimization. *IEEE Transactions on Automatic control*, 45(12):2432–2436, 2000.

[51] Sameh A Eisa and Sameer Pokhrel. Analyzing and mimicking the optimized flight physics of soaring birds: A differential geometric control and extremum seeking system approach with real time implementation. *SIAM Journal on Applied Mathematics*, pages S82–S104, 2023.

[52] David F Chichka, Jason L Speyer, and CG Park. Peak-seeking control with application to formation flight. In *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No. 99CH36304)*, volume 3, pages 2463–2470. IEEE, 1999.

[53] Benjamin Moidel, Ahmed A Elgohary, Shivam Bajpai, and Sameh Eisa. Reintroducing the formation flight problem via extremum seeking control. In *AIAA SCITECH 2024 Forum*, page 2317, 2024.

[54] Andrzej Banaszuk, Youping Zhang, and Clas A Jacobson. Adaptive control of combustion instability using extremum-seeking. In *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, volume 1, pages 416–422. IEEE, 2000.

[55] Sameer Pokhrel and Sameh A Eisa. A novel hypothesis for how albatrosses optimize their flight physics in real-time: an extremum seeking model and control for dynamic soaring. *Bioinspiration & Biomimetics*, 18(1):016014, 2022.

[56] Ximing T Zhang, Darren M Dawson, Warren E Dixon, and Bin Xian. Extremum-seeking nonlinear controllers for a human exercise machine. *IEEE/ASME Transactions on mechatronics*, 11(2):233–240, 2006.

[57] MDCL. Cesc experiment for known objective function. https://drive.google.com/file/d/11sY23NKBz3jJLs99AzJyZbaojh05Mrif/view?usp=drive_link, 2024. Accessed: 2024-06-20.

[58] MDCL. Cesc experiment for light source seeking. https://drive.google.com/file/d/1FA7ahWMXfy0zX5lIknIl676c8SaG9ctO/view?usp=sharing, 2024. Accessed: 2024-06-20.

[59] MDCL. Caesc with unicycle dynamics experiment for known objective function. https://drive.google.com/file/d/1Vlb7fw8wINcFpEzV4jcgOv28LSqFQwDk/view?usp=sharing, 2024. Accessed: 2024-06-20.

[60] MDCL. Caesc with unicycle dynamics experiment for light source seeking. https://drive.google.com/file/d/1m2dF-nyzId8unqav4c93LdLeNVtqDhfi/view?usp=sharing, 2024. Accessed: 2024-06-20.

[61] MDCL. Caesc with single integrator dynamics experiment for known objective function. https://drive.google.com/file/d/1AdFJO-LZ5uzHRZvQ6LRlniIkwPJoiX-K/view?usp=sharing, 2024. Accessed: 2024-06-20.

[62] MDCL. Caesc with single integrator dynamics experiment for light source seeking. https://drive.google.com/file/d/1cTWs4mf2SP6JOnjOTrm-Tk2BPi1Sf0zM/view?usp=sharing, 2024. Accessed: 2024-06-20.

[63] Alberto Isidori. *Nonlinear control systems: an introduction*. Springer, 1985.

[64] 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.

[65] Alexander Scheinker and Miroslav Krstić. Extremum seeking with bounded update rates. *Systems & Control Letters*, 63:25–31, 2014.

[66] Raik Suttner and Sergey Dashkovskiy. Exponential stability for extremum seeking control systems. *IFAC-PapersOnLine*, 50(1):15464–15470, 2017.

[67] Alexander Scheinker and David Scheinker. Bounded extremum seeking with discontinuous dithers. *Automatica*, 69:250–257, 2016.