Trajectory Modeling, Estimation and Interception of a Thrown Ball using a Robotic Ground Vehicle

by

Nirangkush Das

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2018 by the
Graduate Supervisory Committee:

Armando A. Rodriguez, Co-Chair
Spring Berman, Co-Chair
Panagiotis Artemiadis

ARIZONA STATE UNIVERSITY

December 2018

ABSTRACT

Toward the ambitious long-term goal of developing a robotic circus, this thesis addresses key steps toward the development of a ground robot that can catch a ball. Toward this end, we examine nonlinear quadratic drag trajectories for a tossed ball. Relevant least square error fits are provided. It is also shown how a Kalman filter and Extended Kalman filter can be used to generate estimates for the ball trajectory. Several simple ball intercept policies are examined. This includes open loop and closed loop policies. It is also shown how a low-cost differential-drive research grade robot can be built, modeled and controlled. Directions for developing more complex xy planar intercept policies are also briefly discussed. In short, the thesis establishes a foundation for future work on developing a practical ball catching robot.

TABLE OF CONTENTS

Page

LIST OF FIGURES iv

CHAPTER

1 INTRODUCTION 1

1.1 Introduction and Motivation 1

1.2 Literature Survey 2

1.3 Contributions: Critical Questions to be addressed 4

1.4 Outline of Thesis 5

1.5 Summary and Conclusions 6

2 TRAJECTORY MODELING 7

2.1 Introduction and Overview 7

2.2 Parabolic Spherical Particle Trajectories: No Drag 8

2.3 Spherical Particle Trajectories: Including Linear Drag 9

2.4 Spherical Particle Trajectories: Including Quadratic Drag and Approximations 12

2.5 Summary and Conclusions 29

3 TRAJECTORY ESTIMATION 30

3.1 Introduction and Overview 30

3.2 Kalman Bucy Filtering (KBF) 30

3.3 Extended Kalman Filter (EKF) 34

3.4 Comparisons: KBF vs EKF 35

3.5 Summary and Conclusions 37

4 DIFFERENTIAL DRIVE GROUND ROBOTIC MODELING AND CONTROL 38

4.1 Introduction and Overview 38

CHAPTER Page

4.2 Description of Hardware 38
4.3 Description of Model 39
4.4 Speed Control of the Ground Vehicle 43
4.5 Summary and Conclusions 44
5 CONTROL LAWS FOR GROUND ROBOT INTERCEPTING A BALL 45
5.1 Introduction and Overview 45
5.2 Interception Control Laws 46
5.3 Robot Intercepting Ball: Simulation Results 48
5.4 Stability Proof 54
5.5 Summary and Conclusions 57
6 SUMMARY, CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH 58
6.1 Summary of Work 58
6.2 Directions for Future Research 59
REFERENCES 60
APPENDIX
A MATLAB CODE 63

LIST OF FIGURES

Figure Page

2.1 Variation of C_d with the Reynold's number for a sphere [28] 11

2.2 Range vs Initial Velocity: Golf Ball, Quadratic Drag Model 17

2.3 Range vs Initial Velocity, Golf Ball, Quadratic vs No Drag Model 18

2.4 Range vs Initial Angle: Golf Ball, Quadratic Drag Model 19

2.5 Range vs Initial Angle: Golf Ball, Quadratic vs No Drag Model 20

2.6 Quadratic Least Squares Fit of Range vs Initial Velocity 21

2.7 Quadratic Least Squares Fit of Range vs Initial Angle 22

2.8 Quadratic Least Squares Coefficients (v_0 fit) vs initial angle 23

2.9 Quadratic Least Squares Coefficients (θ_o fit) vs initial velocity 24

2.10 Quadratic LS Fit of Coefficients (v_0 fit) vs initial angle 25

2.11 Quadratic LS Fit of Coefficients (θ_o fit) vs initial velocity 26

2.12 Range vs Initial velocity: Original vs coefficients from $v_o - \theta_o$ fits 27

2.13 Range vs Initial velocity: Original vs coefficients from $\theta_o - v_o$ fits 28

3.1 x-displacement: Actual vs Estimated (KBF) 35

3.2 x-displacement: Actual vs Estimated (EKF) 35

3.3 y-displacement: Actual vs Estimated (KBF) 36

3.4 y-displacement: Actual vs Estimated (EKF) 36

3.5 Estimation Errors (KBF) 36

3.6 Estimation Errors (EKF) 36

4.1 The Differential Drive Ground Robot and model [1] 38

4.2 Outer-Loop Position-Direction Control System [1] 43

4.3 Inner-Loop Speed (v, ω) Speed Control System [1] 43

5.1 Diagram depicting the ball position A(x_b, y_b) and car position C$(x_c, 0)$.
α is the elevation angle while ϕ is the line of sight angle. 47

Figure Page

5.2 Car acceleration profile for $x_c(0)$ ranging from 0 to 50 m. 49

5.3 Car velocity profile for $x_c(0)$ ranging from 0 to 50 m. Velocity increases as car approaches the range. 49

5.4 Car position for $x_c(0)$ ranging from 0 to 50 m. Intercept occurs for any $x_c(0)$. In practice, acceleration would be constrained. 50

5.5 Ball trajectory for $V_o = 10\ m/s$ and $\theta_o = 45^o$ with no drag and quadratic drag with $\beta = 0.0085$. Range and Time of flight are $R = 9.55\ m$ and $t_R = 1.416\ s$. 50

5.6 Horizontal displacement of ball with time. 51

5.7 Horizontal velocity of ball with time. 51

5.8 Vertical displacement of ball with time. 51

5.9 Vertical velocity of ball with time. 51

5.10 Car accelerations for various $x_c(0)$ for ideal x_c equation 52

5.11 Car accelerations for various $x_c(0)$ for OAC control law 52

5.12 Car velocities for various $x_c(0)$ for ideal x_c equation 53

5.13 Car accelerations for various $x_c(0)$ for OAC control law 53

5.14 Car positions for various $x_c(0)$ for ideal x_c equation 53

5.15 OAC Car Positions: Final % Error wrt half car length. $T = 10^{-6}$s 53

5.16 Error wrt half car length for various $x_c(0)$ for ideal x_c equation 54

5.17 Error wrt half car length for various $x_c(0)$ for OAC control law 54

5.18 ϕ vs time for various initial positions of car (from Stability eqn 5.25) . . 56

5.19 ϕ vs time for various initial positions of car (90-α) 56

5.20 ϕ vs time for various initial positions of car (via inverse tangent) 56

5.21 α vs time for various initial positions of car 56

Chapter 1

INTRODUCTION

1.1 Introduction and Motivation

Robotics and Autonomous systems are a burgeoning filed. With the tremendous progress made in the field of Autonomous Vehicles, Reusable Rockets etc, it is imperative that control system design has to be almost perfect so that the systems are safe for use and also give us predictable results. Humans have been doing ballistic studies since many years ago. But the most progress to understand ballistics has only happened in the last 30 years [2] [3]. One example of ballistics is the missile technology where a missile successfully follows and intercepts a target with tremendous precision. Not much different from this problem is the ball catching problem. Humans have been catching balls since ages. It is most commonly seen in sports like Baseball and Cricket where the outfielders have to catch a strongly hit ball with bare hands. It is imperative that the catchers have to know where to detect the ball, how to follow a path to successfully intercept and ultimately grasp the fast moving ball. Researchers have tried to come up with numerous frameworks to properly understand how an expert outfielder actually catches the ball [4] [5] [6]. Many control design methods were explored [7] [8] [9] [10] [11] [12] [13] [14] [15]. This work is a sincere attempt in laying the foundations for building a low cost ground vehicle to successfully intercept and catch a tossed ball.

1.2 Literature Survey

There is a tremendous amount of literature for solving the ball catching problem. Various approaches were taken such as visual systems for a 3D catch [2], catching with a robotic arm [16], nonprehensile catching [17], catching complex nonspherical objects [18]. In an effort to shed light on the state of research on modeling ball trajectories, estimation and their interception, the following topically organized literature survey is offered. An effort is made below to highlight what technical papers/works are most relevant to this thesis. In short, the following works are most relevant for the developments within this thesis:

- [19] is the paper where the author tried to find analytical closed form solution to the motion of a particle with Quadratic Air Drag. Approximate expressions for low trajectories, short time, long time and exact implicit solutions were provided.

- The Exact time implicit solution to the Problem of a Projectile moving in a Medium with Quadratic Drag is given in terms of parametric description of the particle motion in [20].Both One dimensional and two dimensional motions were considered. An Exact Analysis of the maximum range of the projectile is provided and general properties of the exact parametric solution is discussed. The time Implicit Solution has quadratures of limited practical applicability. Ideal Projectile Trajectory can be approximated by a quadratic polynomial. This paper claims a Quadratic Drag Projectile can be approximated close to a Cubic Polynomial. Approximate solution while approximating with a Cubic Law is presented and validated.

- [21] uses Lambert W function to find the range of a projectile subject to lin-

ear resistant medium. It also derives projectile motion equations in quadratic resistant medium with low angle trajectories via Lambert W Function. [22] derives projectile motion equations in quadratic resistant medium with low angle trajectories, nonzero height via Lambert W function.

- Homotopy Analysis method applied to a projectile motion in a quadratic resistant medium to approach to a general analytical solution is shown in [23]. Solution is given in terms of Power Series. The Solution is compared with simulation results from Range-Kutta method for various angles of throw.

- [24] incorporated effects of Spin along with identifying other aerodynamic parameters of the projectile subject to a quadratic resistant medium. [25] gave analytical expressions in the form of a ratio of two series expansion. Reducing Coupled Nonlinear Differential equations to manageable ones are discussed. Effect of Side wind also added into the dynamics and precise parametric equations were developed in [26].

- Analytical explicit expressions are derived in [27] that accurately predict the maximum height, its arrival time as well as the flight range of the projectile at the highest ascent. The most significant property of the proposed formula is that they are not restricted to the initial speed and firing angle of the object, nor to the drag coefficient of the medium. In combination with the available approximations in the literature, it is possible to gain information about the flight and complete the picture of a trajectory with high precision, without having to numerically simulate the full governing equations of motion.

- This NASA web article [28] provides insights into how the Drag Coefficient of a Smooth and Rough Sphere varies with Reynold's Number.The flow patterns

vary from ideal attached flow to separated turbulent flow for specific thresholds of the Reynold's Number.

- All the parameters of a flight in a resistant medium eg. lift force, drag, force of buoyancy and coefficient of restitution are considered and an algorithm developed for a realistic simulation are shown in [29].

- [4] established the Linear Optical Trajectory (LOT) theory for a successful interception of the ball by a baseball outfielder in 3D. This paper also laid down the flaws in the previous Optical Angular Cancellation (OAC) law.

- [30] used OAC law to implement it on a ground vehicle with a ball dropped vertically down and got successful interception

1.3 Contributions: Critical Questions to be addressed

Within this thesis, the following fundamental questions are addressed. When taken collectively, the answers offered below, and details within the thesis, represent a useful contribution to researchers in the field. Moreover, it must be emphasized that answers to these questions are critical in order to move substantively toward the longer-term goal of designing a Robotic Ball catcher.

1. **Modeling the Drag.** How can we model trajectories subjected to Quadratic Drag reasonably and efficiently? Can such models be useful for catching a ball?

2. **Estimating the trajectory.** What methods can be used to effectively estimate ball trajectories for catching purposes?

3. **Catcher/Robot Design.** How can we build a low-cost research grade vehicle that can catch a ball?

4. **Catching Theories.** What are good catching strategies? What are their limitations?

5. **Catch Zone.** How can we quantify what a tough catch is? Can we effectively compute the catch zone?

6. **Control Law for Interception.** How can we effectively devise a Control law along with our knowledge of catching strategies to effectively intercept and catch a ball?

In short, this work provides a systematic approach to a learn, design and build a robotic ball catcher.

1.4 Outline of Thesis

The remainder of the thesis is organized as follows. In Chapter 2, we lay the foundation for modeling the trajectory of the projectile under various conditions of the medium. We start from the simple no drag model and gradually move to linear drag model culminating in the nonlinear quadratic drag model. In Chapter 3, we describe the estimation techniques and under what conditions each estimator is sufficient. We start with a simple linear Kalman Bucy estimator to a more complicated nonlinear discrete time Extended Kalman Filter. Chapter 4 contains an illustrative hardware description followed by control design of a differential drive ground robotic vehicle or our probable catcher. Chapter 5 proposes open loop and closed loop ball interception control policies to successfully intercept the ball. The results are illustrated with pros and cons of each control law. Finally, Chapter 6 summarizes the thesis and presents directions for future research.

1.5 Summary and Conclusions

In this chapter, an overview of the work presented in this thesis and the major contributions have been provided. A central contribution of the thesis is laying the foundation and theoretical frameworks for building a low cost research grade ground vehicle that can catch a tossed ball.

Chapter 2

TRAJECTORY MODELING

2.1 Introduction and Overview

Trajectory modeling is the first and foremost step of learning about the trajectory of a tossed ball. Trajectories of the ball has been considered in ideal drag conditions first that lays the foundations for better understanding the behavior of the ball in linear drag as well as the more complicated quadratic drag model. The quadratic drag model is considered closest to the realistic scenario of ball trajectories [19]. From here onwards, we will be studying the properties of a tossed ball in a resistant medium [19]-[31]. The following properties of the medium are considered:

$$\rho = 1.2041\ kg/m^3\ (Density\ of\ Air,\ STP\ at\ Sea\ Level) \tag{2.1}$$

$$g = 9.81\ m/s^2\ (Acceleration\ due\ to\ Gravity) \tag{2.2}$$

That being said, we consider the following parameters for the tossed ball. These parameters resemble that of a golf ball in normal atmospheric conditions.

$$m = 0.045\ kg\ (Mass\ of\ Sphere,\ Golf\ Ball) \tag{2.3}$$

$$r = 0.02\ m\ (Radius\ of\ Sphere,\ Golf\ Ball) \tag{2.4}$$

$$A = \pi r^2\ (Frontal\ Area\ of\ the\ Ball) \tag{2.5}$$

The projectile is considered to be tossed either by a thrower on the ground or by a human subject from a height. But for reasons of simplicity, we have considered all throws from the ground level. The following launch conditions are considered:

$$\theta_o = 45^\circ\ (Inital\ Launch\ Angle\ (degrees)) \tag{2.6}$$

$$v_o = 10\ m/s\ (Inital\ Launch\ Velocity\ (m/s)) \tag{2.7}$$

2.2 Parabolic Spherical Particle Trajectories: No Drag

Projectile motion in a resistant medium subjected to a no drag force is considered now. The following assumption is made.

$$F_D = 0 \tag{2.8}$$

The force balance in the x-axis yields

$$ma_x = 0 \tag{2.9}$$

$$a_x = 0 \tag{2.10}$$

The force balance in the Y-axis yields

$$ma_y = -mg \tag{2.11}$$

$$a_y = -g \tag{2.12}$$

From the equations above, we get

$$a_x = \frac{dv_x}{dt} = 0 \tag{2.13}$$

$$a_y = \frac{dv_y}{dt} = -g \tag{2.14}$$

Let us consider the following states

$$\begin{aligned} x_1 = x \quad x_2 = \dot{x} \\ x_3 = y \quad x_4 = \dot{y} \end{aligned} \tag{2.15}$$

Writing the state space equations give

$$\dot{x_1} = x_2 \tag{2.16}$$

$$\dot{x_2} = 0 \tag{2.17}$$

$$\dot{x_3} = x_4 \tag{2.18}$$

$$\dot{x_4} = -g \tag{2.19}$$

$$\dot{x} \;=\; Ax + Bu \qquad\qquad y \;=\; Cx + Du \tag{2.20}$$

with scalar input u, state vector $x = [\ x_1 \;\; x_2 \;\; x_3 \;\; x_4\ ]^T$, and scalar output y.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g \end{bmatrix} \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 \end{bmatrix} \tag{2.21}$$

2.3 Spherical Particle Trajectories: Including Linear Drag

Before diving into incorporating linear drag in the resistant medium, we need to have basic ideas about the Reynold's number. This section will also give an idea about when the linear drag model suffices in estimating the trajectory of a tossed ball correctly. It will also provide a substantial motivation on incorporating the quadratic drag model into our calculations.That being said, we will talk a little bit about the Reynold's number, then move onto defining the drag force and finally the drag coefficient. We will also talk about how to model the drag coefficient correctly, since that is of paramount importance in getting the drag models working perfectly. The reynold's number, also called particle reynold's number is "a dimensionless quantity that characterizes the nature of the surrounding medium when an object is moving through it" [32]. It is defined as the "ratio of inertial forces (resistant to change of motion) of the moving object to the viscous forces (heavy or gluey forces) of the fluid medium" [32]. "It quantifies the transition from laminar flow to turbulent flow

around the moving object" [32].

$$R_e = \frac{\text{Inertal Forces}}{\text{Viscous Forces}} \tag{2.22}$$

$$R_e = \frac{\rho v l}{\mu} \tag{2.23}$$

where R_e is the Reynold's number, ρ is density of air ($\frac{Kg}{m^3}$), v is the velocity of the golf ball ($\frac{m}{s}$) l is the characteristic length (m) and μ is the dynamic viscosity (Pa.s).

A high value of Reynold's Number (10^5) indicates that inertial forces dominate the viscous forces resulting in turbulent flow patterns. A low value of reynold's number (< 10) indicates that viscous force is in abundance and keeps the flow attached and Laminar.

That being said, we can now comfortable move onto understanding the drag force. The drag force is defined as a resistant force acting opposite to the direction of motion of an object moving in a fluid medium [28]

$$F_D = \frac{1}{2} C_D \rho A v^2 = \beta v^2 \tag{2.24}$$

where F_D is the drag force, C_D is the drag coefficient of the ball, ρ is the density of air ($\frac{Kg}{m^3}$), v is the velocity of the ball ($\frac{m}{s}$) and A is the projected area of the ball (m^2)

As evident from the drag force equation, the drag coefficient is the most important parameter that needs to be computed correctly to estimate the ball trajectories correctly. The drag coefficient C_D is a dimensionless number that characterizes all of the complex factors that affect Drag [28]. The drag coefficient depends on the reynold's number and the shape of the object [28]. The drag coefficient is mostly determined experimentally in a wind tunnel via the drag force equation [28]. Drag Coefficient of a Golf Ball is 0.45 [28].

The drag coefficient of a sphere varies with reynold's number as shown in the figure below Figure 2.1 [28].

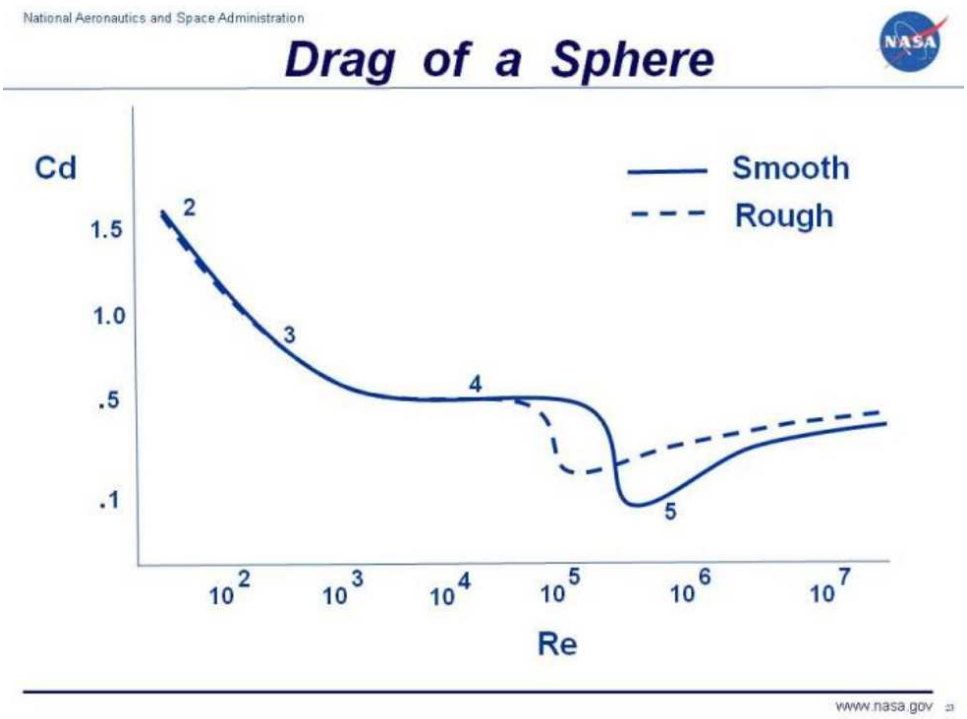

Figure 2.1: Variation of C_d with the Reynold's number for a sphere [28]

Figure 2.1 shows that around $Re = 10^2$

$$C_D \propto \frac{1}{R_e} \propto \frac{1}{v} \tag{2.25}$$

We would want to know under what conditions the model works and more importantly, when it doesn't. That being said, we computed the maximum velocity of the projectile under which the linear drag model is valid. The underlying idea is that withing the range of Reynold's Number when linear drag model is valid, we will find out the maximum velocity allowed. That would give us a fair idea about the advantages and limitations of the linear drag model.

$$R_e = \frac{\rho v l}{\mu} \tag{2.26}$$

where R_e is the Reynold's number, ρ is the density of air ($\frac{Kg}{m^3}$), v is the velocity of the ball ($\frac{m}{s}$), l is the characteristic length (m) and μ is the dynamic viscosity (Pa.s)

$$\rho = 1.225(\frac{kg}{m^3}) \text{ at sea level} \tag{2.27}$$

$$R_e = 10^2 \text{ for attached flow} \tag{2.28}$$

$$\rho = \text{Density of air } (\frac{Kg}{m^3}) \tag{2.29}$$

$$l = 0.04277(m) \tag{2.30}$$

$$\mu = 18.37 \times (10^{-6})(Pa.s) \tag{2.31}$$

solving the equation,

$$100 = \frac{1.225 * v * 0.04277}{18.37(10^{-6})} \tag{2.32}$$

$$v = 0.035\ ms^{-1} \text{ or } v = 0.115\ ft.s^{-1} \tag{2.33}$$

As evident, we are dealing with velocities varying from 5 m/s to 20 m/s . The linear drag model ceases to work at a meager velocity of 0.035 m/s. Hence, the linear drag model is insufficient in properly estimating the trajectory of a ball in a resistant medium.

2.4 Spherical Particle Trajectories: Including Quadratic Drag and Approximations

Projectile motion in a resistant medium subjected to a quadratic drag force. We make the following assumption [19].

$$F_D \propto -v^2 \tag{2.34}$$

$$F_D = -\beta v^2 \tag{2.35}$$

Force balance in the x-axis gives:

$$
\begin{aligned}
ma_x &= -\beta v^2 \cos\theta && (2.36)\\
a_x &= -\frac{\beta}{m} v^2 \cos\theta && (2.37)\\
a_x &= -\frac{\beta}{m} v(v\cos\theta) && (2.38)\\
a_x &= -\frac{\beta}{m} v v_x && (2.39)
\end{aligned}
$$

Force balance in the y-axis gives:

$$
\begin{aligned}
ma_y &= -mg - kv^2 \sin\theta && (2.40)\\
a_y &= -g - \frac{\beta}{m} v^2 \sin\theta && (2.41)\\
a_y &= -g - \frac{\beta}{m} v(v\sin\theta) && (2.42)\\
a_y &= -g - \frac{\beta}{m} v v_y && (2.43)
\end{aligned}
$$

Considering both the axes together gives:

$$
\begin{aligned}
a_x &= \frac{dv_x}{dt} = -\frac{\beta}{m} v v_x && (2.44)\\
a_y &= \frac{dv_y}{dt} = -\frac{\beta}{m} v v_y - g && (2.45)
\end{aligned}
$$

Let us consider the following states:

$$
\begin{aligned}
x_1 = x \quad x_2 = \dot{x}\\
x_3 = y \quad x_4 = \dot{y}
\end{aligned}
\qquad (2.46)
$$

Writing the State Space Equations gave:

$$
\begin{aligned}
\dot{x_1} &= x_2 && (2.47)\\
\dot{x_2} &= -\frac{\beta}{m} \sqrt{(x_2^2 + x_4^2)}\; x_2 && (2.48)\\
\dot{x_3} &= x_4 && (2.49)\\
\dot{x_4} &= -\frac{\beta}{m} \sqrt{(x_2^2 + x_4^2)}\; x_4 - g && (2.50)
\end{aligned}
$$

As is evident we get a system of coupled nonlinear differential equation which as of yet, doesn't have an analytical solution [23].

To override this problem, a simple combination of horizontal and vertical toss is merged along with quadratic drag to come up with a closed form analytical solution [33]. The concept of terminal velocity is introduced to help with the analytical solution [33]. The terminal velocity is essentially the state at which the weight and drag of a ball cancels out and the ball attains a steady speed. Henceforth, considering the vertical descent of our ball,

$$a = 0 \tag{2.51}$$

$$\text{Weight}(W) = \text{Drag Force}(F_D) = mg \tag{2.52}$$

$$W = F_D = mg \tag{2.53}$$

$$F_D = C_d \rho A v_t^2 \text{ (where } v_t \text{ is the terminal velocity)} \tag{2.54}$$

$$v_t = \sqrt{\frac{2mg}{C_d \rho A}} \tag{2.55}$$

Let us consider the ascent trajectory now

$$\text{Initial vertical velocity} = v_{y_0} \tag{2.56}$$

$$\text{Initial horizontal velocity} = v_{x_0} \tag{2.57}$$

$$F_{net} = -W - F_D \tag{2.58}$$

$$a = -g\left(1 + \frac{v}{v_t^2}\right) \tag{2.59}$$

$$\frac{v}{v_t} = \frac{v_{y_0} - v_{x_0} \tan(\frac{gt}{v_t})}{v_t + v_{y_0} \tan(\frac{gt}{V_t})} \tag{2.60}$$

$$y = \frac{v_t^2}{2g} \ln\left(\frac{v_{y_0}^2 + v_t^2}{v^2 + v_t^2}\right) \tag{2.61}$$

Now let us consider the horizontal aspect of the trajectory

$$
\begin{aligned}
F_{net} &= ma = -F_D && (2.62)\\
a &= \frac{-C_d \rho A u v^2}{2m} && (2.63)\\
a &= -\frac{g v^2}{v_t^2} && (2.64)\\
u &= \frac{v_t^2 v_{x_0}}{v_t^2 + g v_{x_0} t} && (2.65)\\
x &= \frac{v_t^2}{g} \ln\left(\frac{v_t^2 + g v_{x_0} t}{v_t^2}\right) && (2.66)
\end{aligned}
$$

From the equations above, we get analytical expressions for the horizontal displacement x and the vertical displacement y [33].

Least Square fit to Quadratic Drag model. Since no analytical closed form approximation is available in literature, we need suitable approximations to the problem of a ball trajectory in a quadratic resistant medium. In this section, we will be looking at suitable least square fits that would give us the best approximations. We approach this problem in two distinct pathways. The first is referred to as the $v_o - \theta_o$ least square fit while the second pathway is referred to as the $\theta_o - v_o$ least square fit. The steps are outlined to give an idea of what lies ahead.

$v_o - \theta_o$ **LS fit**

- $R(v_o, \theta_o)$: We plot Range vs v_o with θ_o as a parameter.
- We do LS quadratic fits for each θ_o
- The coefficients that we get depend on θ_o
- $R(v_o, a_i(\theta_o)) = a_2(\theta_o)v_o^2 + a_1(\theta_o)v_o + a_0(\theta_o)$

$\theta_o - v_o$ **LS fit**

- $R(\theta_o, v_o)$: We plot Range vs θ_o with v_o as a parameter.
- We do LS quadratic fits for each v_o
- The coefficients that we get depend on v_o
- $R(\theta_o, b_i(v_o)) = b_2(v_o)\theta_o^2 + b_1(v_o)\theta_o + b_0(v_o)$

Range vs Initial Velocity. The following plot show how the range varies with the initial velocities for three initial launch angles $\theta = 30^o, \theta = 45^o, \theta = 60^o$.

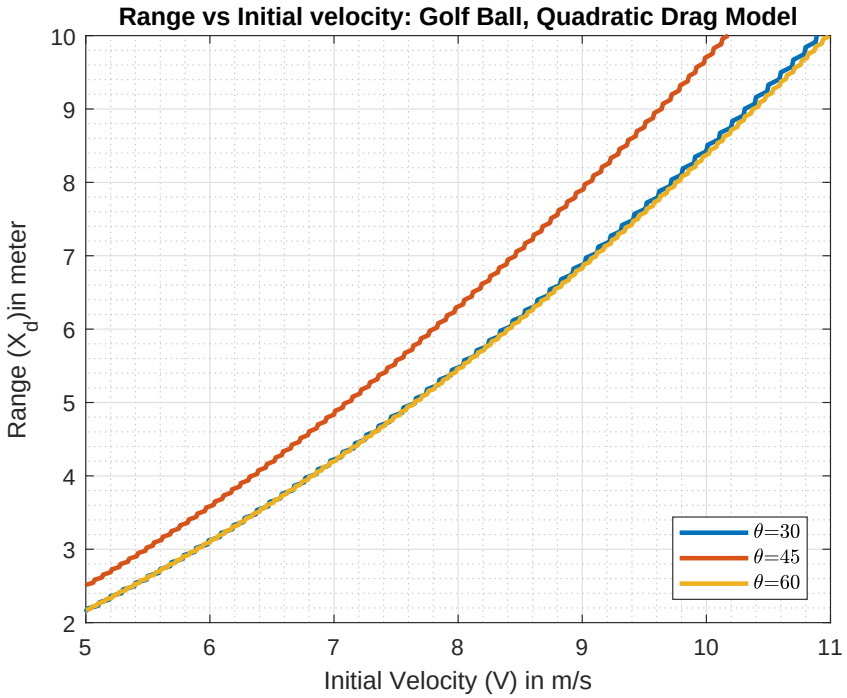

Figure 2.2: Range vs Initial Velocity: Golf Ball, Quadratic Drag Model

- Number of velocity points taken was 60000
- The initial velocity v_o was chosen to lie in [5,11] m/s
- MATLAB ode45 solver used to solved the 4 nonlinear differential equations arising from considering the Quadratic Drag Model of the Golf Ball.
- Step Size of Ode45 Solver considered = 0.0001. With this particular step size and 60000 Velocity points, time taken is around 30 mins.
- As expected, Range increases with higher Initial Velocity for a particular θ_o
- It should be noted that the observed (small) oscillations can be reduced by taking more points

Range vs Initial Velocity: Quadratic vs No Drag Model. We try to make necessary comparisons on how the trajectory changes with ideal drag to Quadratic drag.

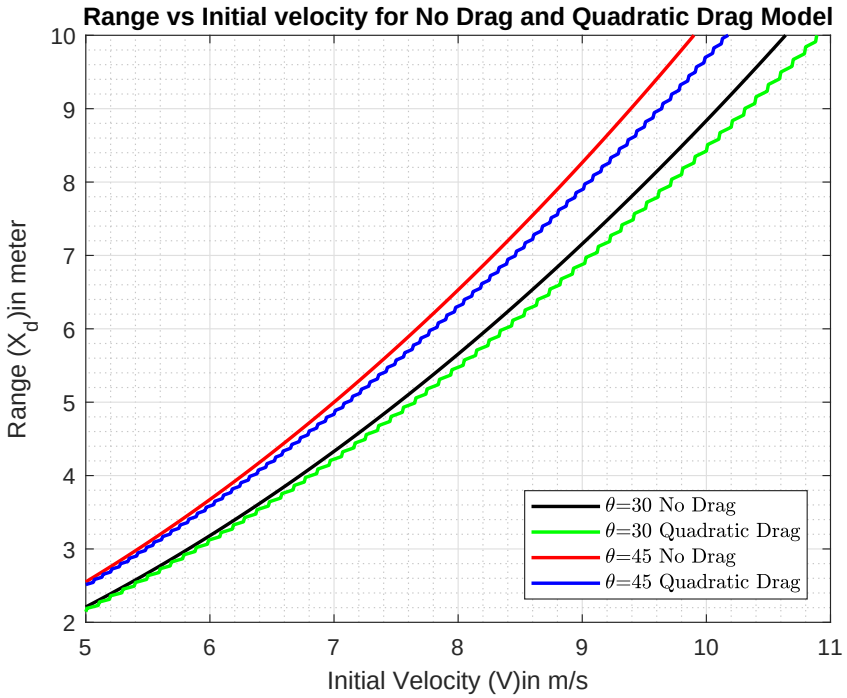

Figure 2.3: Range vs Initial Velocity, Golf Ball, Quadratic vs No Drag Model

- For Lower Initial Velocities e.g. at 5 m/s, Range is almost the same for both Quadratic and No Drag Model
- For Higher Initial Velocities e.g. at 10 m/s, Range is more for the No Drag Model.

Range vs Initial Angle: Quadratic Drag Model. We plot to understand how the range varies with the initial angle of launch for three particular velocities.

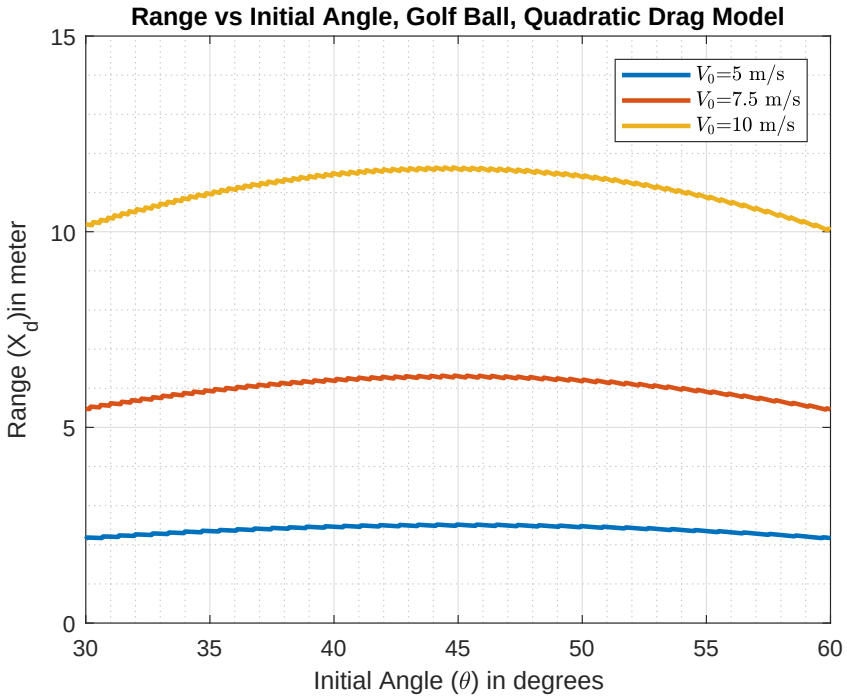

Figure 2.4: Range vs Initial Angle: Golf Ball, Quadratic Drag Model

- Number of angle points taken was 60000
- Range of initial angles were chosen to be $[30°, 60°]$
- MATLAB ode45 solver used to solved the 4 nonlinear differential equations arising from considering the Quadratic Drag Model of the Golf Ball.
- Step Size of Ode45 Solver considered = 0.0001. With this particular step size and 600 Velocity points, time taken is around 30 mins.
- For a particular Initial velocity, Range peaks at $\theta = 45°$.
- This peaking is more prominent at Higher Initial velocities e.g. at 10 m/s where we see a significant curvature than at 5 m/s

Range vs Initial Angle: Quadratic vs No Drag Model. We try to make necessary comparisons on how the trajectory changes with ideal drag to Quadratic drag.

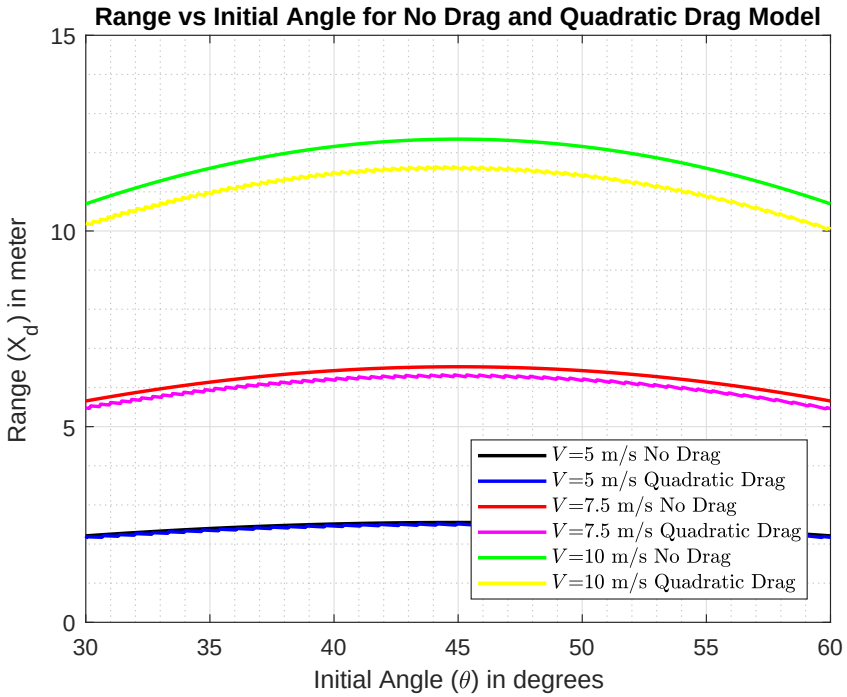

Figure 2.5: Range vs Initial Angle: Golf Ball, Quadratic vs No Drag Model

- Variation of Range is almost same for both Quadratic and No Drag Model, for low Initial Velocities e.g. 5m/s
- At Higher Velocities, Range is more for No Drag for any Initial Angle.
- The Curvature remains the same for both Quadratic and No Drag Model for Each Velocities.

Quadratic Least Squares Fit of Range vs Initial Velocity. We fit quadratic least sqaure curves on the range vs initial speed plot.

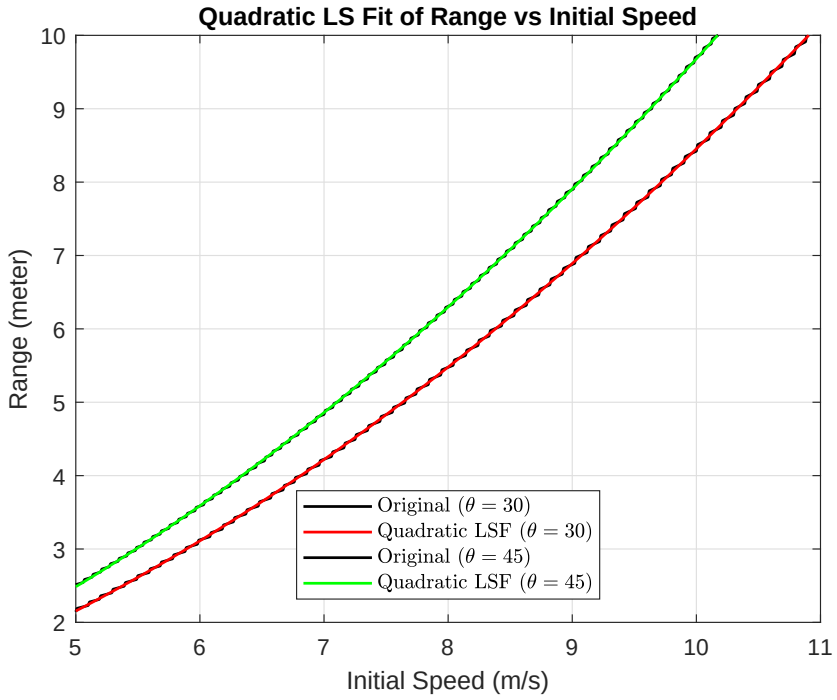

Figure 2.6: Quadratic Least Squares Fit of Range vs Initial Velocity

- Most of the fit curves overlap the original curves
- $\hat{R}(v_o, a_i(\theta_o)) = a_2(\theta_o)v_o^2 + a_1(\theta_o)v_o + a_0(\theta_o)$
- The Maximum Error is 1.36%

Quadratic Least Squares Fit of Range vs Initial Angle. We fit quadratic least sqaure curves on the range vs initial angle plot.

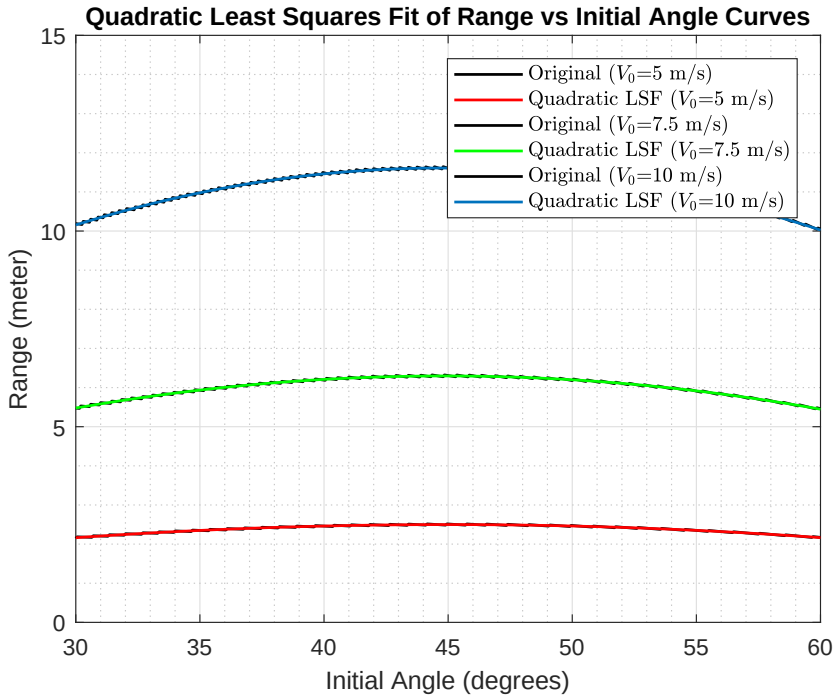

Figure 2.7: Quadratic Least Squares Fit of Range vs Initial Angle

- Most of the fit curves overlap the original curves
- $\hat{R}(\theta_o, b_i(v_o)) = b_2(v_o)\theta_o^2 + b_1(v_o)\theta_o + b_0(v_o)$
- The Maximum Error is 1.48%

Least Square fit: Coefficient Analysis. The coefficients that we got from fitting the quadratic drag plots are used for a second round of approximations. We follow the similar two approaches mentioned earlier, namely $v_o - \theta_o$ coefficients fit and $\theta_o - v_o$ coefficients fit. In the former we plot the coefficients as a function of θ_o while in the latter we plot the coefficients with v_o.

Quadratic Least Squares Coefficients (v_0 fit) vs initial angle. We plot the coefficients of the range vs initial velocity plot to see how they vary with the initial angle parameter.

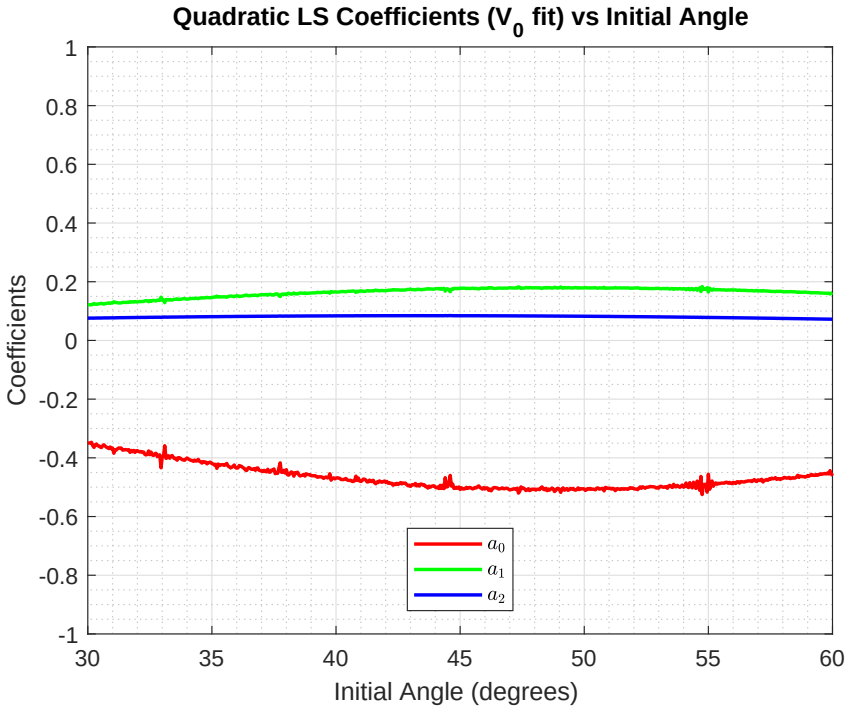

Figure 2.8: Quadratic Least Squares Coefficients (v_0 fit) vs initial angle

- The plot shows how the v_0 fit coefficients change as we vary the initial angle
- The uneven jumps in the curve can be minimized by either approximating with a an LS fit or increasing the number of data points.
- We will be doing fits for these coefficient curves

Quadratic Least Squares Coefficients (θ_o fit) vs initial velocity. We plot the coefficients of the range vs initial angle plot to see how they vary with the initial velocity parameter.

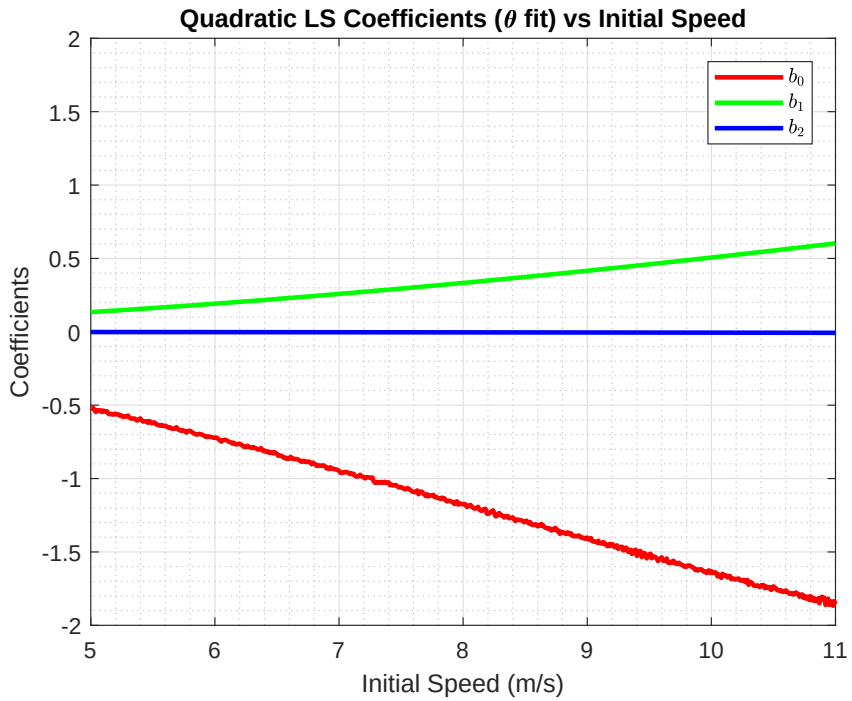

Figure 2.9: Quadratic Least Squares Coefficients (θ_o fit) vs initial velocity

- The plot shows how the θ_o fit coefficients change as we vary the initial velocity.
- We will be doing fits for these coefficient curves to properly understand whether they are quadratic or cubic in nature
- The coefficient b_2 seems to be zero for all initial velocities. Upon closer inspection its not.

Quadratic LS Fit of Coefficients (v_0 fit) vs initial angle. We perform a second quadratic least squares fit over the coefficient curve to model the behavior.

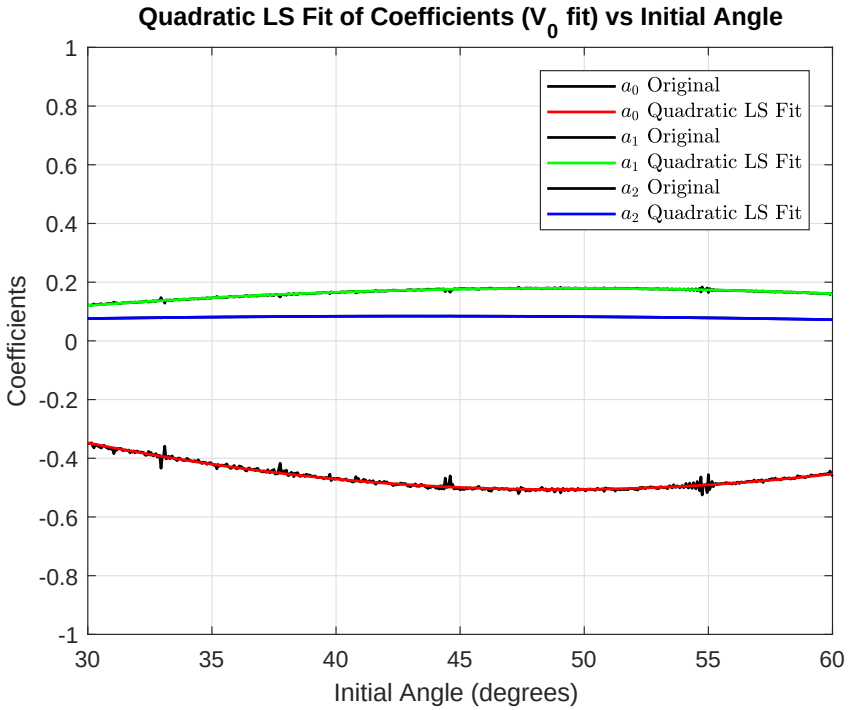

Figure 2.10: Quadratic LS Fit of Coefficients (v_0 fit) vs initial angle

$$a_0 = 0.0110\,\theta_o^2 - 0.1936\,\theta_o + 0.3454 \tag{2.67}$$

$$a_1 = -0.0040\,\theta_o^2 + 0.0701\,\theta_o - 0.1310 \tag{2.68}$$

$$a_2 = -0.0011\,\theta_o^2 + 0.0174\,\theta_o + 0.0172 \tag{2.69}$$

- The Quadratic Least Squares Fit gave a good approximation of how the v_0 fit coefficients vary with θ_o

Quadratic LS Fit of Coefficients (θ_o fit) vs initial velocity. We perform a second quadratic least squares fit over the coefficient curve to model the behavior.

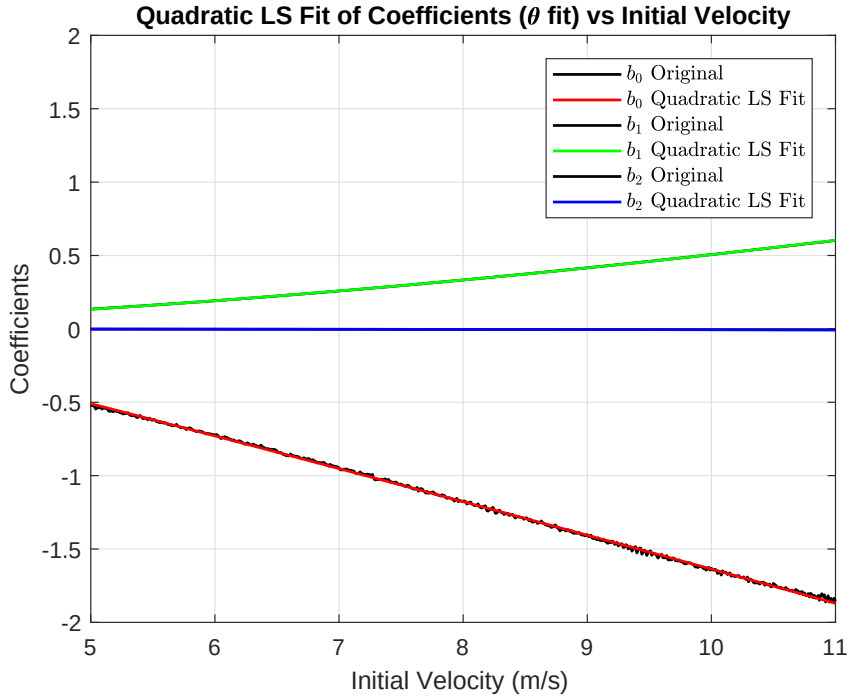

Figure 2.11: Quadratic LS Fit of Coefficients (θ_o fit) vs initial velocity

$$b_0 = -0.0014\, v_o^2 - 0.2051\, v_o + 0.5500 \tag{2.70}$$

$$b_1 = 0.0039\, v_o^2 + 0.0156\, v_o - 0.0431 \tag{2.71}$$

$$b_2 = -0.000045\, v_o^2 + 0.00016\, v_o + 0.00044 \tag{2.72}$$

- The Quadratic Least Squares Fit gave a good approximation of how the θ_o fit Coefficients vary with v_o
- As evident, b_2 is very close to 0 for any initial velocities. Hence it looks like Range is almost linearly dependent on the Initial Angle for a golf Ball with quadratic drag.

Range vs initial velocity: Original vs coefficients from $v_o - \theta_o$ fits

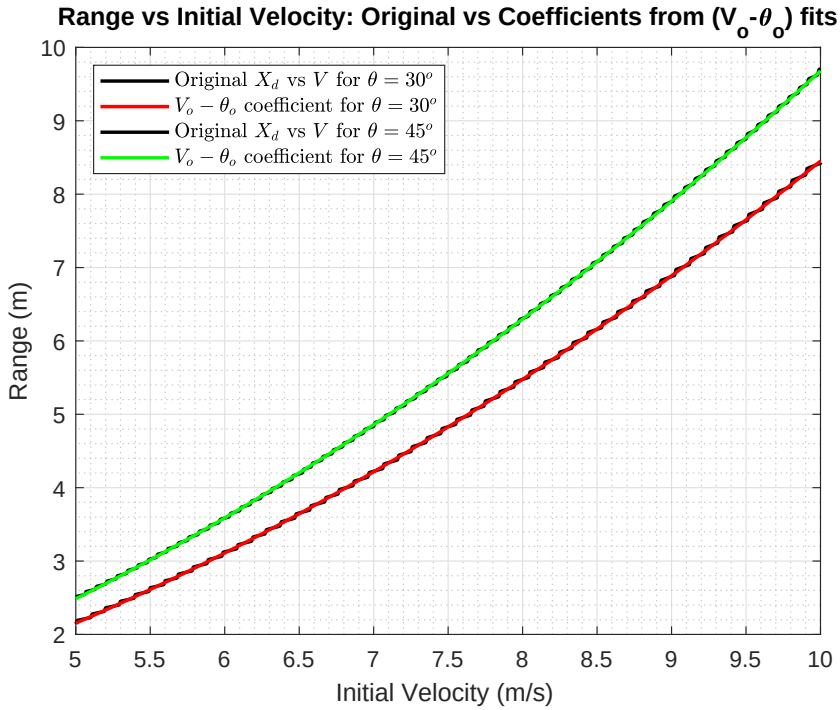

Figure 2.12: Range vs Initial velocity: Original vs coefficients from $v_o - \theta_o$ fits

- $\hat{\hat{R}}(v_o, \theta_o) = \hat{a}_2(\theta_o)v_o^2 + \hat{a}_1(\theta_o)v_o + \hat{a}_0(\theta_o)$
- The maximum error is 1.38%

Range vs Initial velocity: Original vs coefficients from $\theta_o - v_o$ fits

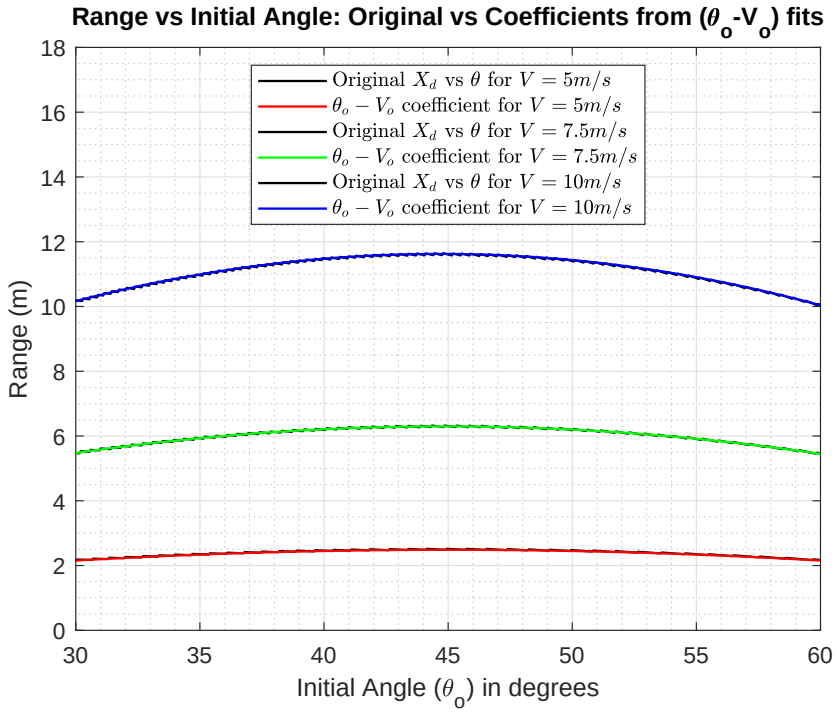

Figure 2.13: Range vs Initial velocity: Original vs coefficients from $\theta_o - v_o$ fits

- $\hat{\hat{R}}(\theta_o, v_o) = \hat{b}_2(v_o)\theta_o^2 + \hat{b}_1(v_o)\theta_o + \hat{b}_0(v_o)$
- The maximum error is 1.29%

2.5 Summary and Conclusions

We modeled the trajectories using various models of drag i.e. the ideal drag or no drag model, the linear drag model and finally concluding it with the quadratic drag model. We estimated the probable trajectories under each conditions. There is no analytical closed form solution for solving the coupled and nonlinear dynamical equations that we get when considering quadratic drag. Based on NASA archives, we find out suitable approximations for quadratic drag using vertical toss and horizontal toss. Finally, we approximated the quadratic drag trajectories using least squares fits and got very good approximations.

Chapter 3

TRAJECTORY ESTIMATION

3.1 Introduction and Overview

In this chapter, we estimate the trajectory of the model that we built in the last chapter. We start with the basics involving linear estimators like Kalman Bucy filtering and proceed to more complicated Extended Kalman Filters. We try to answer the essential questions like how perfect is the estimation, what is the cost associated and under what conditions do the Estimators fail.

3.2 Kalman Bucy Filtering (KBF)

In this section we motivate and precisely formulate the Kalman-Bucy Filtering problem, the ideas which are laid out comprehensively in [34].

System Dynamics. We begin by considering an LTI MIMO plant having the following dynamical model

$$\dot{x} = Ax + Bu + L\xi \qquad x(t_o) = x_o \tag{3.1}$$

$$y = Cx + Du + \theta \tag{3.2}$$

subject to three sources of uncertainty:

- *Initial Condition.* x_o is an uncertain initial condition. It is not known. Only *a priori* first and second order statistics are assumed to be available:

$$E(x_o) = m_o \tag{3.3}$$

$$E((x_o - m_o)(x_o - m_o)^T) = \Sigma_o \qquad \Sigma_o = \Sigma_o^T \geq 0 \tag{3.4}$$

- *Process Noise.* ξ is called the *process noise*. It is not accessible. It is assumed to be a WSS stochastic process. Only *a priori* first and second order statistics are assumed to be available:

$$E(\xi) = m_\xi \tag{3.5}$$

$$\Sigma_\xi(\tau) = I \ \delta(\tau) \tag{3.6}$$

- *Measurement Noise.* θ is the *measurement noise* or *sensor noise*. It is not accessible. It is assumed to be a WSS stochastic process. Only *a priori* first and second order statistics are assumed to be available:

$$E(\theta) = m_\theta \tag{3.7}$$

$$\Sigma_\theta(\tau) = \Theta \ \delta(\tau) \qquad \Theta = \Theta^T > 0 \tag{3.8}$$

Goal. The goal is to develop a state estimation structure to estimate the state x of the system given access to the input u, the output y, and statistical information regarding the three sources of uncertainty: the initial condition x_o, the process noise ξ, and the measurement noise.

State Estimation Error. We define the *state estimation error* as follows:

$$\tilde{x} \stackrel{\text{def}}{=} x - \hat{x}. \tag{3.9}$$

The immediate goal is to formulate an optimization problem that will result in a filter gain matrix H that will ensure stable state estimation error $\tilde{x}$ dynamics. Toward this end, we consider minimizing the following quadratic (rms state estimation error) cost functional:

$$J(\hat{x}) \stackrel{\text{def}}{=} \lim_{t_o \to -\infty} \sqrt{E(\ \tilde{x}(t)^T \ \tilde{x}(t)\ )} \tag{3.10}$$

where the limit implies that we have infinite past measurements. This is analogous to the infinite planning horizon (infinite upper limit) in the LQR problem.

Design Parameters. The matrices L and Θ are analogous to the matrices M and R in the LQR problem. L and Θ should be viewed as design parameters that we use to achieve design specifications; i.e. to obtain an acceptable filter gain matrix H . The following assumption will be made. The KBF problem and its solution are now provided.

KBF Problem Statement:

$$\min_{\hat{x}(u,y)} J(\hat{x}) \stackrel{\text{def}}{=} \lim_{t_o \to -\infty} \sqrt{E(\tilde{x}\tilde{x})} \tag{3.11}$$

Solution: The solution to this problem is provided by the following model based state estimation/observer structure:

$$\dot{\hat{x}} = Ax + Bu + Lm_\xi + H(y - \hat{y}) \qquad \hat{x}(t_o) = m_o \tag{3.12}$$

$$\hat{y} = C\hat{x} + Du + m_\theta \tag{3.13}$$

where

- $\hat{x}$ is the optimal estimate of the state x;
- $\hat{y}$ is an estimate of the (known) output y;
- m_o is the mean of the initial condition x_o; it represents our best *a priori* estimate for x_o;
- m_ξ is the mean of the process noise ξ; it represents our best *a priori* estimate for ξ;
- m_θ is the mean of the measurement noise θ; it represents our best *a priori* estimate for θ;
- the term $H(y - \hat{y})$ provides a feedback mechanism which will keep $\hat{x}$ close to x. The matrix $H \in \mathcal{R}^{n \times p}$ is called the *filter gain matrix*. It is analogous to the

control gain matrix G in the LQR problem .

where the *filter gain matrix* $H \in \mathcal{R}^{n \times p}$ is given by

$$H = \Sigma C^T \Theta^{-1} \tag{3.14}$$

and Σ is the unique symmetric (at least) positive semidefinite solution of the following *Filter Algebraic Riccati Equation (FARE)*:

$$0 = A\Sigma + \Sigma A^T + LL^T - \Sigma C^T \Theta^{-1} C\Sigma. \tag{3.15}$$

Nominal Stability: The state estimation error $\tilde{x}$ dynamics, defined by

$$\dot{\tilde{x}} = (A - HC)\tilde{x} + L(\xi - m_\xi) + H(\theta - m_\theta) \tag{3.16}$$

are exponentially stable; i.e. all eigenvalues of (A-HC) have negative real parts.

3.3 Extended Kalman Filter (EKF)

The Extended Kalman filter is widely used in controls related applications, especially in missile technologies. Since many of the real world systems are nonlinear, the Kalman Bucy filter discussed in the previous section fails to address the inherent nonlinearities. That being said, the Extended Kalman filter is the nonlinear companion of the Kalman Bucy filter that takes into account the inherent nonlinearities. It essentially takes a linearized estimate of the states at every step of the filter. The equations for EKF has been taken from [35]. We used a discrete time Kalman Filter to estimate the states of our projectile subjected to nonlinear quadratic drag force.

Consider the nonlinear discrete time model of any system being

$$x_k = f(x_{k-1}, u, k) + w_{k-1} \tag{3.17}$$

$$y_k = h(x_k, u_k, k) + v_k \tag{3.18}$$

Here, k denotes a discrete point in time, u_k is a vector of inputs, x_k is a vector of the actual states, which may be observable but not measured, y_k is a vector of the actual process outputs, w_k and v_k are process and output noises respectively; assumed to be zero mean Gaussian with covariance Q_k and R_k respectively. For the Extended Kalman Filter, the predictor step is given by

$$\hat{x_k}^- = f(\hat{x}_{k-1}, u_k, k) \tag{3.19}$$

$$P_k^- = F_{k-1} P_{k-1} F_{k-1}^T + Q_k \tag{3.20}$$

and the corrector step is given by,

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{3.21}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(\tilde{y}_k - h(\hat{x}_k^-, u_k, k)) \tag{3.22}$$

$$P_k = (I - K_k H_k) P_k^- \tag{3.23}$$

3.4 Comparisons: KBF vs EKF

The KBF and EKF simulations were used to estimate the x-displacement and y-displacement of the ball trajectory in ideal medium and quadratic resistant medium respectively, over a period of 2 seconds. The following initial conditions were chosen for both the filters. $x_{initial} = 8$ and $y_{initial} = -8$. Please note that in both the filters Θ and R_k, also called the measurement noise covariances are the design parameters and has been chosen as $0.02\ I_{2x2}$. We try to choose this parameter small to get faster convergence, but at the cost of an expensive sensor. For our launch conditions of $V_o = 10$ m/s and $\theta_o = 45^o$, we get the time of flight $t_R = 1.416$ s. Therefore, we would like a convergence to happen around 1/10th the flight time t_R i.e. 0.14 seconds. To achieve this, $\Theta, R_k = 0.02\ I_{2x2}$ has been chosen. Figure 3.1 and Figure 3.2 shows the tracking of the x-displacement of the ball. We get convergence in around 0.15 seconds which is a fairly good convergence time. Another thing to notice is that the EKF filter does and overshoot before it converges to the actual trajectory.

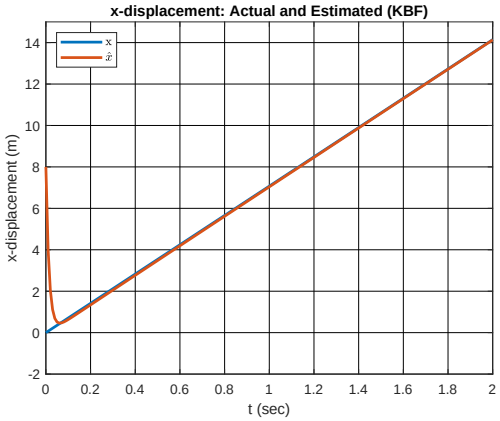

Figure 3.1: x-displacement: Actual vs Estimated (KBF)

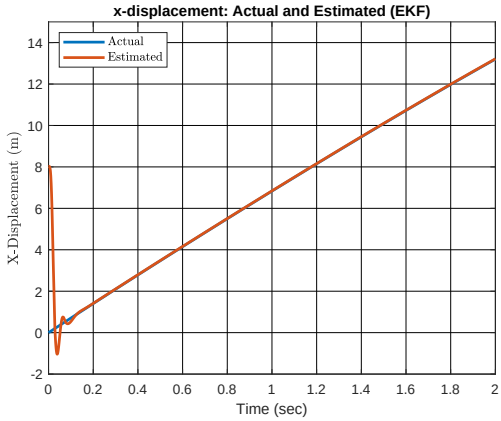

Figure 3.2: x-displacement: Actual vs Estimated (EKF)

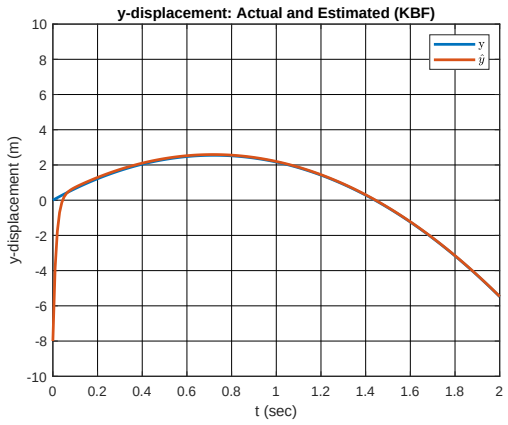

Figure 3.3: y-displacement: Actual vs Estimated (KBF)

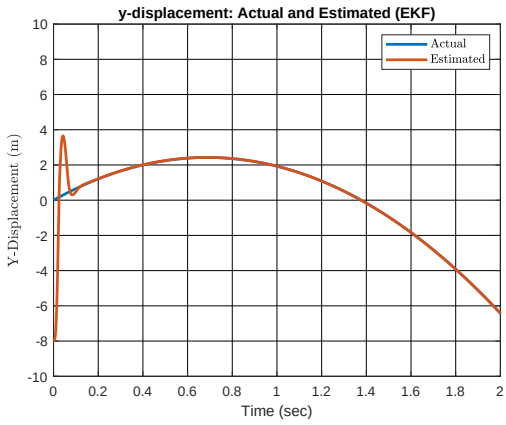

Figure 3.4: y-displacement: Actual vs Estimated (EKF)

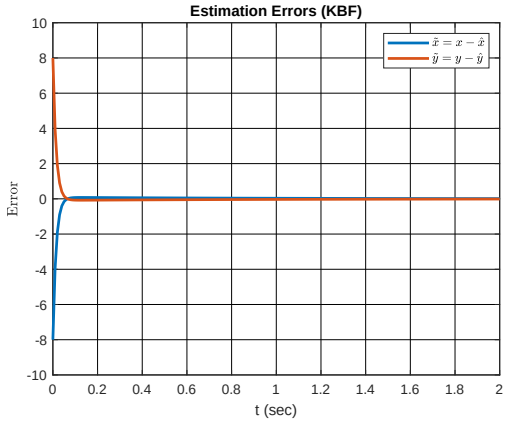

Figure 3.5: Estimation Errors (KBF)

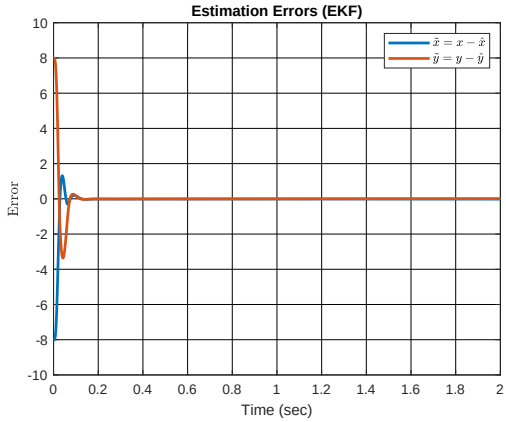

Figure 3.6: Estimation Errors (EKF)

Figure 3.3 and Figure 3.4 shows the tracking of the y-displacement of the ball. We get convergence in around 0.1 seconds. As in the previous plot, EKF filter does an overshoot before it converges to the actual trajectory. Finally, Figure 3.5 and Figure 3.6 shows the state estimation errors. The state estimation error converge to zero at approximately 0.10 s and 0.15s for the KBF and EKF respectively.

3.5 Summary and Conclusions

We used Linear model based and nonlinear model based trajectory estimation filters to estimate the projectile in a quadratic resistance medium. We started with the linear Kalman Bucy Filter and proceeded to the nonlinear discrete Extended Kalman Filter. These estimate errors go to zero faster than the system. These estimates are essential to account for any noises that might be present during sensing the ball, e.g. the camera sensing the ball might have some error. These filters help us clean up the noises and give us a correct idea about the trajectory of the ball. These estimation data will be used in properly understanding catching laws and coming up with an intercept control law in later chapters.

Chapter 4

DIFFERENTIAL DRIVE GROUND ROBOTIC MODELING AND CONTROL

4.1 Introduction and Overview

In this chapter, we will be discussing the modeling and control of a Differential Drive Ground Robot and it possibilities to become a catcher robot for our ball projectile. Trade studies and Control issues will be discussed in detail. The limitations will also be put forth. The hardware details will be discussed first, followed by the modeling of the vehicle. The chapter will be concluded with control design trade studies. In this chapter, the differential drive ground robot developed within [1] are used.

4.2 Description of Hardware

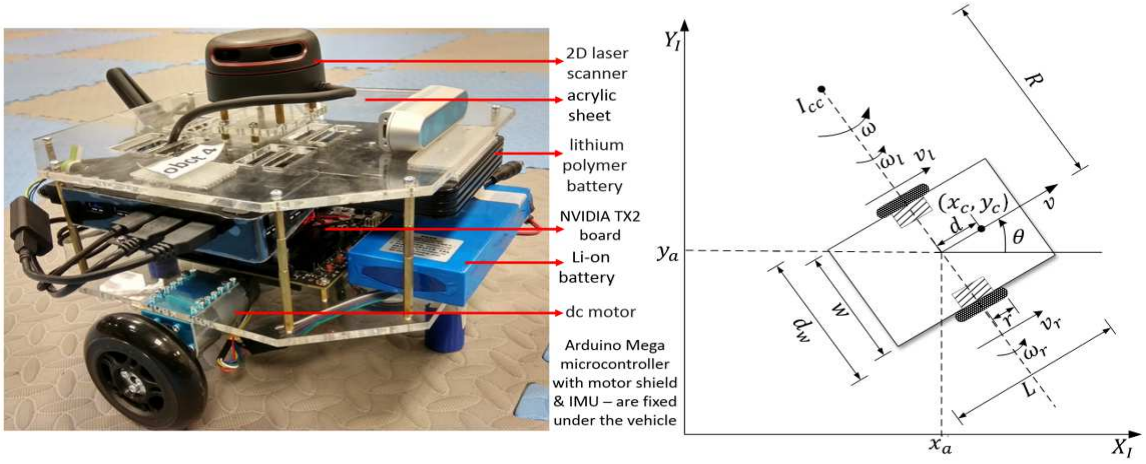

Figure 4.1: The Differential Drive Ground Robot and model [1]

The components of our differential drive ground robot [1] are as follows: 1) a vehicle frame with acrylic sheet on top and bottom which are connected by screws

with dimensions (0.28 m long, 0.28 m wide, 0.20 m high), 2) two brush DC motors having encoders in them (12V, 5 A stall, 1.2 N-m stall, 200 max rpm, 3200 counts per revolution), 3) one Arduino Mega microcontroller with one motor shield (20 kHz max PWM frequency, 12 A per channel) for inner-loop speed control, 4) an NVIDIA TX2 board for on-board high speed computations such as image processing and some outer loop functions, 5) an Intel real Sense Camera, 6) an inertial measurement unit (IMU) with 3 accelerometers and 3 gyros (16-bit digital output per axis, max 8000 samples per sec ADC sampling rate, 3.3V, 3.2 mA), 7) a lithium polymer battery to power the Nvidia TX2, Arduino Mega and all the sensors (18000 mAh), 8) a lithium-ion battery to power the motors (9800 mAh). 9) the vehicle weighs 3.4 kg, with an average acceleration of about 32 m/sec^2 and can achieve a top speed of about 1.6 m/sec in approximately 0.05 seconds.

4.3 Description of Model

Nonlinear Dynamical Model. In this thesis, it has been assumed that the vehicle is symmetrical about the longitudinal centerline (body x axis). This basically means that the axes of the 2 wheels pass through the vehicle's midpoint ($\frac{l}{2}$, body y axis) along length l of vehicle. Each wheel is assumed to be driven by identical dc motors. Throughout the thesis, we will use a nonlinear vehicle-motor-kinematic model. Each component of the model is now described.

Nonlinear Dynamical Vehicle Model. The following nonlinear dynamical model has been taken from [1] and is used in this thesis

$$\begin{bmatrix} I_w + \frac{r^2}{d_w^2}\left(\frac{1}{4}md_w^2 + I\right) & \frac{r^2}{d_w^2}\left(\frac{1}{4}md_w^2 I\right) \\ \frac{r^2}{d_w^2}\left(\frac{1}{4}md_w^2 I\right) & I_w + \frac{r^2}{d_w^2}\left(\frac{1}{4}md_w^2 + I\right) \end{bmatrix} \begin{bmatrix} \dot{\omega}_r \\ \dot{\omega}_l \end{bmatrix} = \begin{bmatrix} 0 & -\frac{r^2}{d_w} m_c d\omega \\ \frac{r^2}{d_w} m_c d\omega & 0 \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} + \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \tag{4.1}$$

Here, $\omega_{r,l}$ are the wheel angular speeds, v is the vehicle speed (along the vehicle centerline or the body axis), $\tau_{r,l}$ are the applied torques to the wheels by the motors,θ is the angle that the vehicle centerline makes with the horizontal; and $\omega = \dot{\theta}$ is the angular speed of the vehicle. Here, m is the vehicle mass, m_c is the vehicle chassis/platform mass (base plus components, excluding wheels and motors), I is the moment of inertia of the vehicle about its c.g., I_w is the moment of inertia of the wheel-motor combination about the wheel axle, r is the wheel radius, l is the vehicle length, w is the vehicle width, d_w is the distance between the two wheels and d is the distance along the vehicle centerline that the vehicle c.g lies in front of the wheel axles. Also, the vehicle mass is given by $m = m_c + 2m_w$ where $m_c = m_b + m_{comp}$, m_b is the mass of the vehicle base, m_{comp} is the mass of the components on the base and m_w is the mass of the wheel-motor combination. The vehicle moment of inertia is given by $I = I_c + m_c d^2 + \frac{1}{2}m_w d_w^2 + I_w$ where I_c is the moment of inertia of the vehicle chassis/platform (base plus components, excluding wheels and motors). Since we have assumed a rectangular vehicle chassis/platform, it follows that $I_c = \frac{1}{12}m_c(l^2 + w^2)$.

This model has the applied torques on each wheel (τ_r, τ_l) as inputs and the wheel angular speeds (ω_r, ω_l) as states. It should be noted that the model is nonlinear (with nonlinear $\omega\omega_r$ and $\omega\omega_l$ product terms) if and only if the vehicle center of gravity (c.g.) does not lie on the wheel axle ($d \neq 0$); i.e. the model is linear when $d = 0$ [1]. The

impact of this and other parameters are examined below.

Nonlinear Speed-Force-Torque Model. We can rewrite the nonlinear model with aggregate force $F = \frac{\tau_r+\tau_l}{r}$ and torque $\tau = d_w\frac{(\tau_r\tau_l)}{2r}$ as inputs and wheel angular speeds (ω_r, ω_l) as states. This makes the model discussed above better to visualize:

$$\left(m + \frac{2I_w}{r^2}\right)\dot{v} - m_c d\omega^2 = F \quad \left(I + \frac{d_w^2}{2r^2}I_w\right)\dot{\omega} + m_c d\omega v \quad = \tau \tag{4.2}$$

Here F and τ are control inputs for v and ω, respectively. Upon simple look the nonlinearities of ω^2 and ωv are observed. here also, it is pretty obvious that the model is nonlinear when $d \neq 0$. This model possesses the following equilibria: $F_{eq} = m_c d\omega_{eq}^2$, $\tau_{eq} = -m_c d v_{eq}\omega_{eq}$. From this, it follows that to maintain a constant speed v_{eq} (with $\omega_{eq} = 0$) requires $F_{eq} = \tau_{eq} = 0$.

Kinematic Model. The following Kinematic model can be simply visualized [1]. $v_x = \dot{x} = v\cos\theta$ is the horizontal speed of the vehicle, $v_y = \dot{x} = v\sin\theta$ is the vertical speed of the vehicle and $\dot{\theta} = \omega$ is the angular speed of the vehicle. In most of the literature that models mobile robot, this simple kinematic model is assumed to be sufficient to model the mobile robot.

Linear Speed-Force-Torque Model. The following linear speed-force-torque equation is obtained $P_{[F,\tau]\to[v,\omega]}$ (linearization done with (v_{eq}, ω_{eq})):

$$\begin{bmatrix}\dot{v}\\ \dot{\omega}\end{bmatrix} = \begin{bmatrix} 0 & \frac{2m_c d\omega_{eq}}{m+\frac{2I_w}{r^2}} \\ \frac{-m_c d\omega_{eq}}{I+\frac{1}{2}\left(\frac{d_w}{r}\right)^2 I_w} & \frac{-m_c d v_{eq}}{I+\frac{1}{2}\left(\frac{d_w}{r}\right)^2 I_w}\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} + \begin{bmatrix}\frac{1}{m+\frac{2I_w}{r^2}}, \frac{1}{I+\frac{d_w^2 I_w}{2r^2}}\end{bmatrix}\begin{bmatrix}F\\ \tau\end{bmatrix} \tag{4.3}$$

Here, we can see that if $d = 0$, then the system "A matrix" is zeroed and the model becomes a gain-integrator model. If $d \neq 0$ and $w_{eq} = 0$, then the model is observed to possess an integrator along with a real pole.

Identical Motor Models. In this thesis, it is assumed that the torques on each wheel (τ_r, τ_l) are created by similar or identical brushed DC motors with input voltages

(e_{a_r}, e_{a_l}). The motor model to be used is as follows:

$$\begin{aligned} e_{a_{r,l}} &= l_a \frac{di_a}{dt} + r_a i_a + e_b && (4.4)\\ e_b &= k_b k_g \omega_{r,l} && (4.5)\\ \tau_{r,l} &= k_t k_g i_a && (4.6)\\ I_w \dot{\omega}_{r,l} + \beta \omega_{r,l} &= \frac{\tau_{r,l}}{k_g^2}, && (4.7) \end{aligned}$$

where $e_{a_{r,l}}$ are the motor input voltages, i_a denotes armature current, e_b denotes back emf, $\omega_{r,l}$ are the angular wheel speeds, $\tau_{r,l}$ are the torques applied to the wheels.

State Space TITO Model. The following two-input two-output fourth order state space model $P_{[e \to \omega_{r,l}]}$ describes our differential drive vehicle near (v_{eq}, ω_{eq}):

$$\dot{x} = Ax + Bu \quad y = Cx + Du \tag{4.8}$$

where $u = [\ e_{a_r}\ e_{a_l}\ ]^T$, $x = [\ v\ \omega\ i_{a_r}\ i_{a_l}\ ]^T$, $y = [\ \omega_r\ \ \omega_l\ ]^T$,

$$A = \begin{bmatrix} \frac{-2\beta k_g^2}{mr^2+I_w} & \frac{2m_c\omega_{eq}d^2}{\hat{m}} & \frac{k_t k_g}{l_a r \hat{m}} & \frac{k_t k_g}{l_a r \hat{m}} \\ \frac{-m_c\omega_{eq}d}{\hat{I}} & \left(\frac{-m_c v_{eq} d}{\hat{I}} - \frac{d_w^2 k_g^2 \beta}{2r^2(\hat{I})}\right) & \frac{k_t k_g d_w}{2l_a r \hat{I}} & -\frac{k_t k_g d_w}{2l_a r \hat{I}} \\ -\frac{k_b k_g}{r} & -\frac{k_b k_g d_w}{2r} & -\frac{r_a}{l_a} & 0 \\ -\frac{k_b k_g}{r} & \frac{k_b k_g d_w}{2r} & 0 & -\frac{r_a}{l_a} \end{bmatrix} \tag{4.9}$$

$$B = \begin{bmatrix} 0_{2\times 2} \\ I_{2\times 2} \end{bmatrix}\ C = \begin{bmatrix} M^{-1} & 0_{2\times 2} \end{bmatrix}\ D = 0_{2\times 2}\ M = r \begin{bmatrix} 0.5 & 0.5 \\ \frac{1}{d_w} & \frac{1}{d_w} \end{bmatrix} \tag{4.10}$$

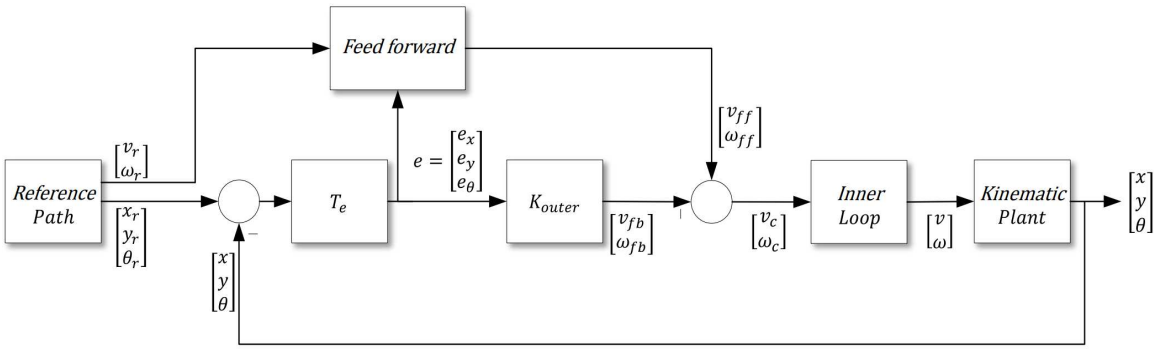

Figure 4.2: Outer-Loop Position-Direction Control System [1]

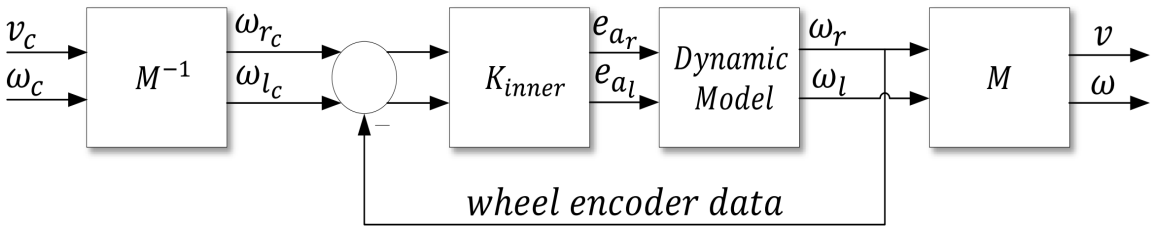

Figure 4.3: Inner-Loop Speed (v, ω) Speed Control System [1]

In the Figures 4.2-4.3, there is an inner-loop (v, ω) speed control system and an outer loop position control system. This refers to the map from commanded or desired speeds (v_r, ω_r) to actual speeds (v, ω) (Fig. 4.3). This refers to the map from commanded planar positions and direction (x_r, y_r, θ_r) to actual planar positions and direction (x, y, θ) (Fig. 4.2). The inner loop speed control system is much faster than the outer loop position control system, typically by an order of magnitude of 5.

Let us suppose that we want our differential drive ground robot to follow reference commands (x_r, y_r, θ_r) with a maximum frequency ω_o. Given this, we require the outer-loop (x, y, θ) position-direction control system to have a bandwidth of roughly $5\omega_o$ or

faster and an inner-loop bandwidth of approximately $25\omega_o$ or faster in order to get acceptable command following.

Inner Loop Speed Control. As discussed in [1], when the plant $P_{[e\to\omega_{r,l}]}$ is nearly decoupled, we can use a decentralized controller $K = kI_{2\times 2}$ where $k = \frac{g(s+z)}{s}\left[\frac{b}{s+b}\right]$ is a PI controller (with high frequency roll off) and $W = \frac{z}{s+z}$ is command pre-filter to address overshoot (not shown in Fig. 4.3). We chose $g = 5, z = 10, b = 200$. This structure is selected because it can be used to generate good controllers for a nearly decoupled $P_{[e\to\omega_{r,l}]} \approx pI_{2\times 2}$.

4.5 Summary and Conclusions

This chapter addressed modeling and control issues associated with differential drive ground vehicles and its potential to become a suitable catcher. A nonlinear model is used to capture the vehicle dynamics. Motor dynamics are also modeled. Trade studies are conducted to shed light on critical vehicle parameters and how they impact static properties, dynamic properties, control system design and overall vehicle design. In the next chapter, the interception control laws are discussed. Future work will involve implementing those laws in the differential drive ground robot.

Chapter 5

CONTROL LAWS FOR GROUND ROBOT INTERCEPTING A BALL

5.1 Introduction and Overview

Even novice baseball or cricket players appear to know virtually from the moment of bat contact where to run to catch a ball [4].To understand this task of intercepting a ball, we simplify the problem by assuming the ball and the interceptor car is in the same plane and the car just moves in a line along the x-axis. To make the car interceptor model simple, we assumed a double integrator model of the car. No acceleration constraint has been put on the car. We would like to understand how the interception works for a planar catch in a quadratic resistant medium. A successful interception is considered when the error between the ball position and the car position is equal or less than half car lengths at the ball drop point or interception zone. In previous chapters, we talked about modeling the trajectories, estimating them, discussed about the hardware and control tradeoffs. We simply laid down the foundations for building a research grade ground vehicle that successfully intercepts a ball. This chapter primarily lays down the foundation for a control law that can be employed with a simple double integrator model of a car. Open Loop and closed loop control policies are discussed, simulation results are presented and the necessity of continuous feedback is put forth. Interdependency of the initial ball launch conditions and initial car positions are provided. Essentially, we want to know under what conditions of initial velocity, angle and initial car positions gives us a successful interception.

5.2 Interception Control Laws

In this section, we discuss primarily an open loop and a closed loop control policy for intercepting a tossed ball in a quadratic resistant medium. We consider a simple double integrator model for the car/robot. We also assume that the car and ball trajectory is in the same plane and the car moves along a line. The open loop policy is based on perfect information of range and time of flight for a projectile motion in ideal or no drag medium. Upon gaining knowledge of the range and time of flight of the ball, we command the car to reach the range at or before the ball hits the ground.

Open Loop Policy. The no drag model for the ball is:

$$\ddot{x_b} = 0 \tag{5.1}$$

$$\ddot{y_b} = -g \tag{5.2}$$

The assumed model for our interceptor car is:

$$\ddot{x_c} = u \tag{5.3}$$

The range and time of flight information of the ball trajectory in a no drag medium:

$$R = \frac{V_o^2 sin2\theta_o}{g} \tag{5.4}$$

$$t_R = \frac{2V_o sin\theta_o}{g} \tag{5.5}$$

Now, assuming perfect knowledge of the range and time of flight, we command the car to get to the range R at the specified time t_R.

$$R = x_c(t_R) = x_c(0) + \frac{1}{2}ut_R^2 \tag{5.6}$$

$$\ddot{x_c} = u = \frac{-2[x_c(0) - R]}{t_R^2} = constant \tag{5.7}$$

Closed Loop Policy. The open loop policy bases itself on a perfect knowledge of the range and time of flight information, which in practice cannot be obtained easily.“

Baseball outfielders could derive the destination from an assumed projected parabolic trajectory, but research indicates that observers are very poor at using such a purely computational approach"[4]. "In addition, factors such as air resistance, ball spin and wind can cause trajectories to deviate from the parabolic ideal"[4]. This interception strategy doesn't demand the ball position, but only needs the elevation angle of the ball with respect to the interceptor or robot, which can be done by a camera mounted on the vehicle. This closed loop policy, also called Gaining Angle of Gaze or the Optical Acceleration Cancellation law (OAC), doesn't need any information regarding the initial ball launch velocity or angle nor any instantaneous speed information [4], [5], [6],[30].

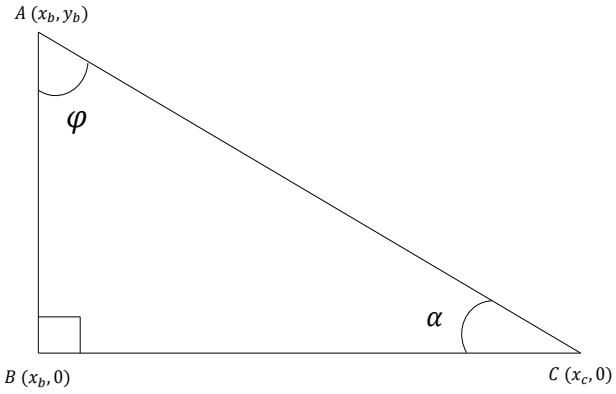

Figure 5.1: Diagram depicting the ball position A(x_b, y_b) and car position C$(x_c, 0)$. α is the elevation angle while ϕ is the line of sight angle.

Consider the diagram shown in Figure 5.1. Let the location of the ball at any time t is A(x_b, y_b), the location of the car/robot be C$(x_c, 0)$ and let α be the elevation angle while ϕ is the line of sight angle. "[4]. The same idea is applied to our interceptor

car. That being said, the quadratic drag model for the ball is:

$$\ddot{x_b} = -\beta v v_x \tag{5.8}$$

$$\ddot{y_b} = -\beta v v_y - g \tag{5.9}$$

The interceptor moves in a path so that the rate of change of the slope with respect to the ball trajectory remains constant. The control law to do the same is defined below [4], [30].

$$\frac{d}{dt}\left(tan\ \alpha\right) = constant \tag{5.10}$$

$$\frac{d}{dt}\left(\frac{AB}{CB}\right) = constant \tag{5.11}$$

$$\frac{d}{dt}\left(\frac{y_b}{x_c - x_b}\right) = constant \tag{5.12}$$

Intergrating the above equation, along with finding the necessary coefficients give us

$$x_c = \frac{y_b x_c(0)}{V_o sin(\theta_o) t} + x_b \tag{5.13}$$

Differentiating the above with respect to time gives us

$$\dot{x_c} = u = -\frac{y_b x_c(0)}{V_o sin(\theta_o)\ t^2} + \frac{\dot{y_b} x_c(0)}{V_o sin(\theta_o)\ t} + \dot{x_b} \tag{5.14}$$

where x_c is the position of the car, u is the controls applied, y_b is y component of the ball position, $x_c(0)$ is the initial car position, V_o is the initial launch velocity, θ_o is the initial launch angle and x_b is the x component of the ball position.

5.3 Robot Intercepting Ball: Simulation Results

Open Loop Policy. Constant acceleration is commanded for specific initial positions of the car $x_c(0)$. The car acceleration commanded is lesser when the initial car position $x_c(0)$ is close to the range i.e. $x_c(0)$ at 10 m has the lowest acceleration due

to proximity with the range at 9.55 m. In summary $\ddot{x_c}$ increases with larger $x_c(0)$ and smaller $t_R(v_o, \theta_o)$.

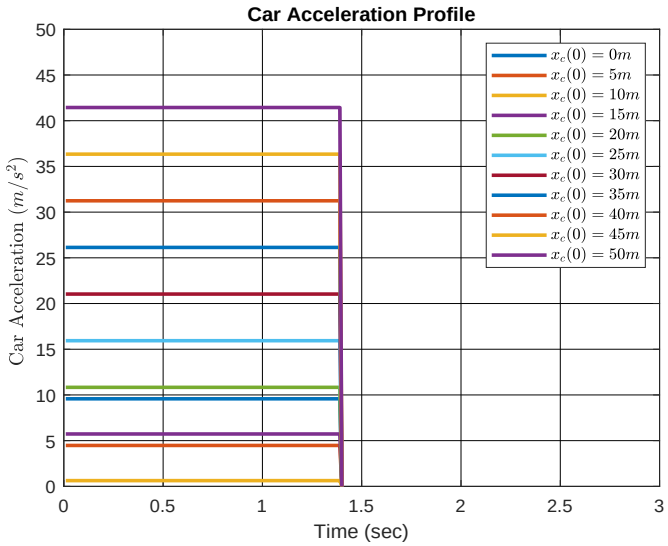

Figure 5.2: Car acceleration profile for $x_c(0)$ ranging from 0 to 50 m.

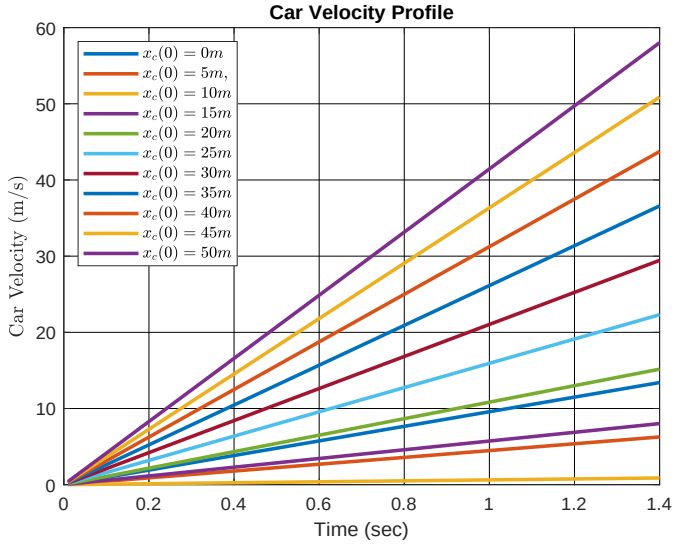

Figure 5.3: Car velocity profile for $x_c(0)$ ranging from 0 to 50 m. Velocity increases as car approaches the range.

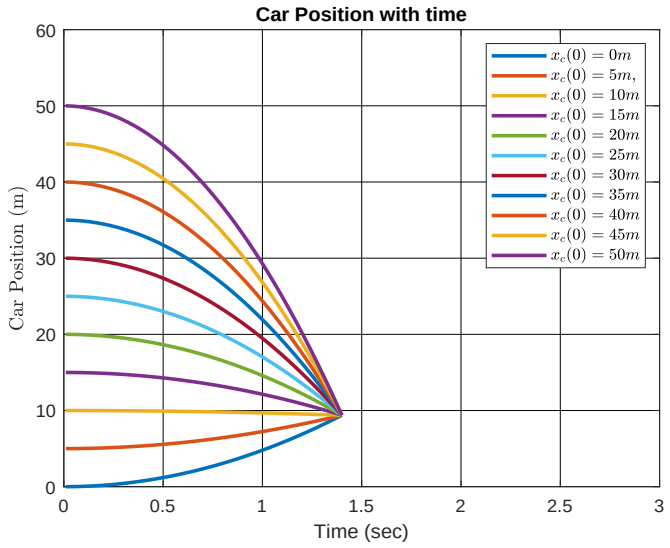

Figure 5.4: Car position for $x_c(0)$ ranging from 0 to 50 m. Intercept occurs for any $x_c(0)$. In practice, acceleration would be constrained.

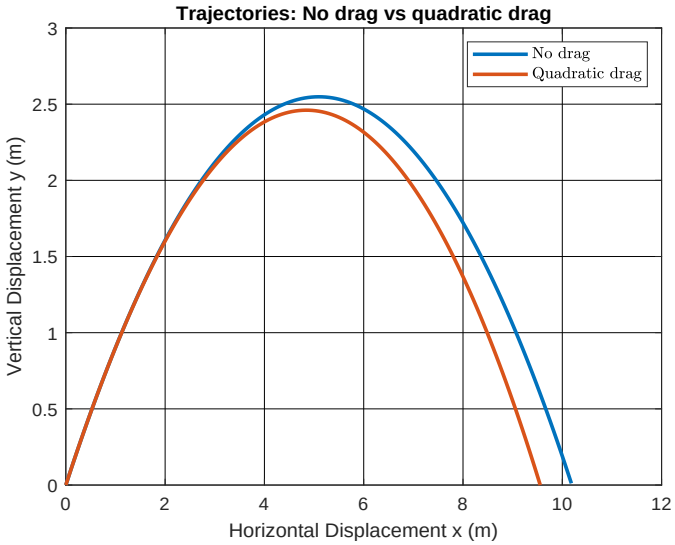

Figure 5.5: Ball trajectory for $V_o = 10\ m/s$ and $\theta_o = 45^o$ with no drag and quadratic drag with $\beta = 0.0085$. Range and Time of flight are $R = 9.55\ m$ and $t_R = 1.416\ s$.

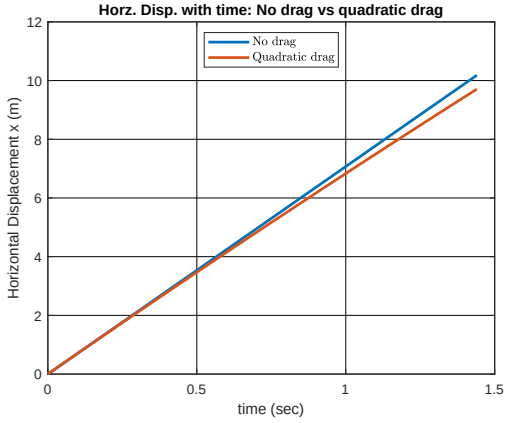

Figure 5.6: Horizontal displacement of ball with time.

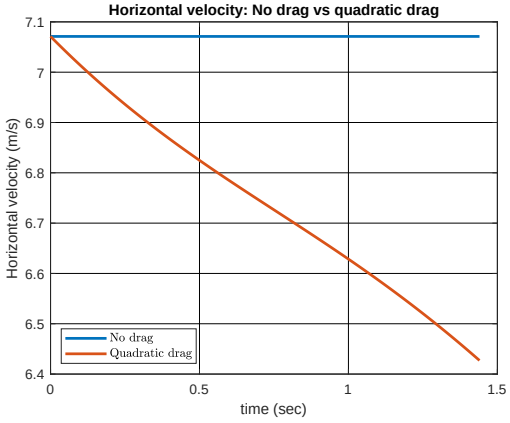

Figure 5.7: Horizontal velocity of ball with time.

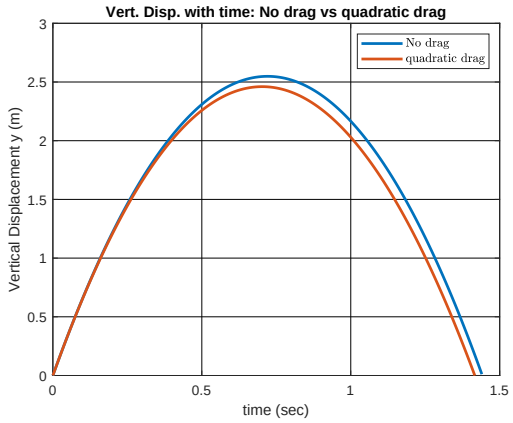

Figure 5.8: Vertical displacement of ball with time.

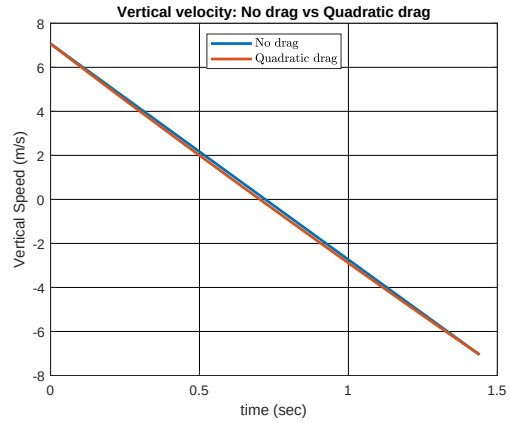

Figure 5.9: Vertical velocity of ball with time.

Closed Loop Policy This policy does not need knowledge of the initial launch conditions of the ball in a quadratic resistant medium. For simulation, $V_o = 10\ m/s$ and $\theta_o = 45^o$ are selected. Figure 5.5 shows the trajectory of the ball. The range and time of flight information obtained from the simulation are $R = 9.55\ m$ and $t_R = 1.416\ s$ respectively. These values are provided to give a perspective. These values are not used in the control policy for the car to intercept the ball. Additionally, Figure 5.6, Figure 5.7, Figure 5.8 and Figure 5.9 are provided to gain an idea regarding the displacement and velocities in each axis. The time step considered is 10^{-6} seconds so that numerical error is minimal. From this point onward, lets define Equation 2.13 as the ideal x_c equation and Equation 2.14 as the OAC control law. It makes sense to simulate and compare the results that we get from each equation. The left plots mean results from the ideal x_c equation and right plots mean results from OAC control law. Figure 5.10and Figure 5.11 depict the car accelerations. Car approaching from the

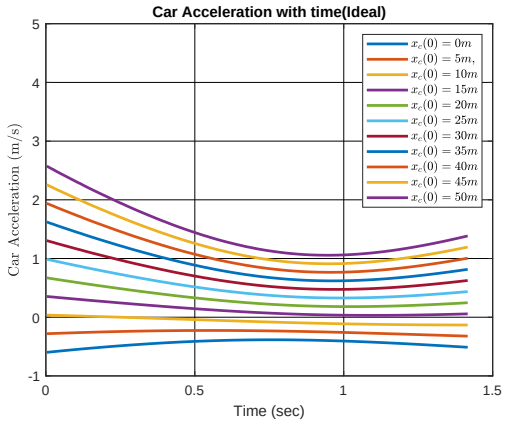

Figure 5.10: Car accelerations for various $x_c(0)$ for ideal x_c equation

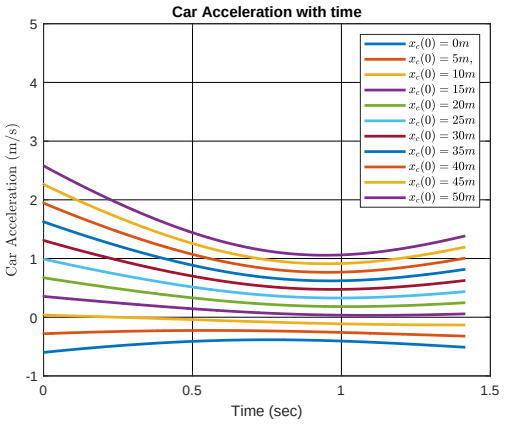

Figure 5.11: Car accelerations for various $x_c(0)$ for OAC control law

left of the range i.e. with $x_c(0) = 0,5$ m, the acceleration curve is concave down which means the acceleration decreases as it approaches the target. Car approaching from the right of the range i.e. with $x_c(0) = 10$ m and above, the acceleration curve

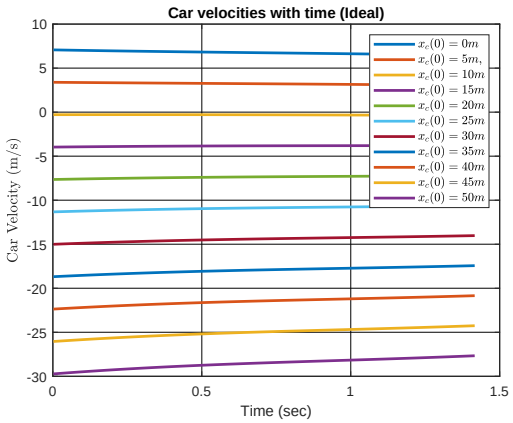

Figure 5.12: Car velocities for various $x_c(0)$ for ideal x_c equation

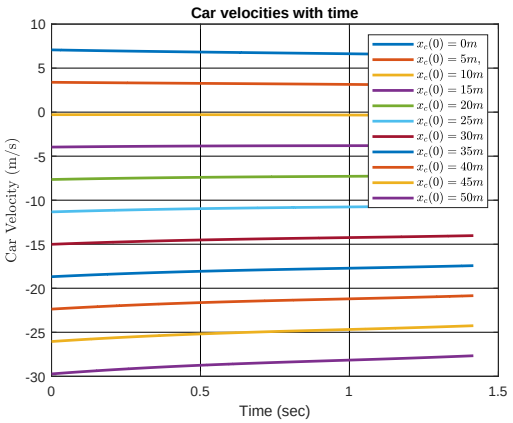

Figure 5.13: Car accelerations for various $x_c(0)$ for OAC control law

is concave up which means the acceleration increases as it approaches the target. Figure 5.12and Figure 5.13 gives us an idea about the commanded velocity. As it is evident from the plots, the car is commanded approximately constant velocity which increases in magnitude as $x_c(0)$ is further from the range. Figure 5.14and Figure 5.15

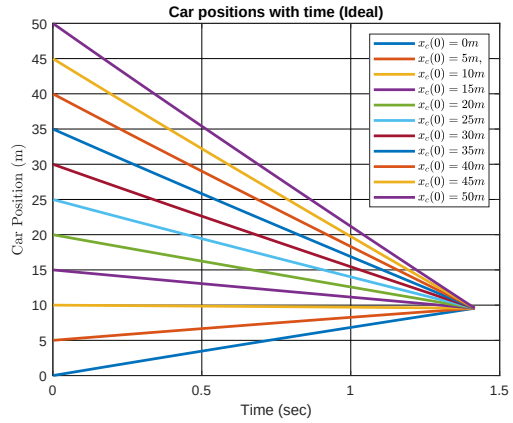

Figure 5.14: Car positions for various $x_c(0)$ for ideal x_c equation

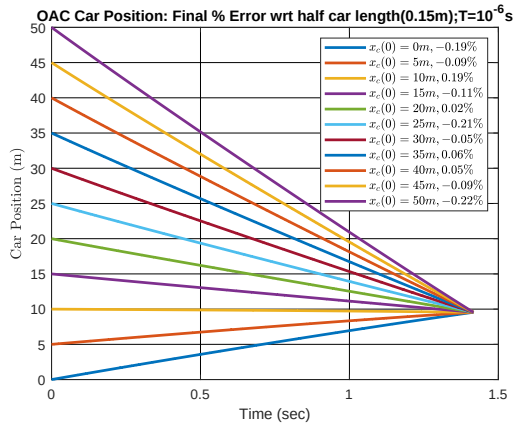

Figure 5.15: OAC Car Positions: Final % Error wrt half car length. $T = 10^{-6}$s

show the car positions for various $x_c(0)$. At this point we define the final error at t_R as $\frac{x_c - R}{0.5\ Car\ Length}$x100. The underlying idea is that we make sure an interception

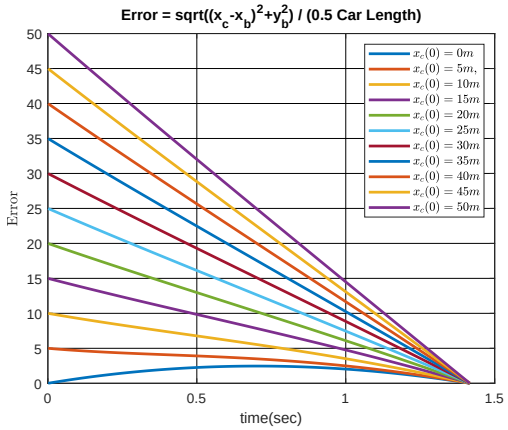

Figure 5.16: Error wrt half car length for various $x_c(0)$ for ideal x_c equation

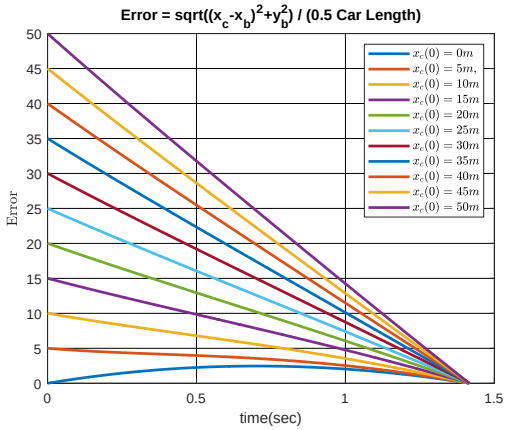

Figure 5.17: Error wrt half car length for various $x_c(0)$ for OAC control law

occurs only when this final error is below a certain limit, say 1/10 th car length. From Figure 5.15, the final error for each car positions are depicted. The maximum error turns out to be about 0.22 % which is well below the 1/10 th limit we want. hence it can be said that the car, irrespective of its initial position intercepts the ball successfully at t_R. To gain additional perspective, Figure 5.20and Figure 5.21 shows the Error with respect to half car lengths go to zero for all initial car positions at t_R. The upcoming section will analytical prove how the line of sight angle ϕ goes to zero upon interception.

5.4 Stability Proof

From Figure 5.1,

$$\phi = 90^o - \alpha \tag{5.15}$$

$$= 90^o - tan^{-1}\left(\frac{y_b}{x_c - x_b}\right) \tag{5.16}$$

Differentiating w.r.t time gives

$$\dot{\phi} = -\dot{\alpha} \tag{5.17}$$

$$= -\frac{(x_c - x_b)^2}{y_b^2 + (x_c - x_b)^2} \frac{d}{dt} (tan\ \alpha) \tag{5.18}$$

$$= -c\ sin^2\phi \tag{5.19}$$

where the constant c is defined as:

$$c = \frac{V_o sin\theta_o}{x_c(0)} \tag{5.20}$$

Rewriting above equations give us:

$$\dot{\phi} = -\frac{V_o sin\theta_o}{x_c(0)}\ sin^2\phi \tag{5.21}$$

For small angle approximations, the above equation can be written as:

$$\dot{\phi} = -\frac{V_o sin\theta_o}{x_c(0)}\ \phi^2 \tag{5.22}$$

Integrating the above gives:

$$\int_{\phi_o}^{\phi} \frac{d\phi}{\phi^2} = -\frac{V_o sin\theta_o}{x_c(0)} \int_0^t dt \tag{5.23}$$

$$\frac{1}{\phi_o} - \frac{1}{\phi} = -\frac{V_o sin\theta_o}{x_c(0)}\ t \tag{5.24}$$

$$\phi = \frac{\phi_o x_c(0)}{x_c(0) + \phi_o V_o sin\theta_o\ t} \tag{5.25}$$

$$\lim_{t\to\infty} \phi = 0 \tag{5.26}$$

From the above equation, ϕ is locally asymptotically stable.

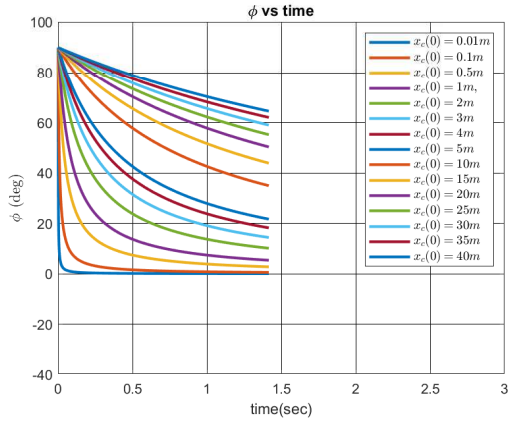

Figure 5.18: ϕ vs time for various initial positions of car (from Stability eqn 5.25)

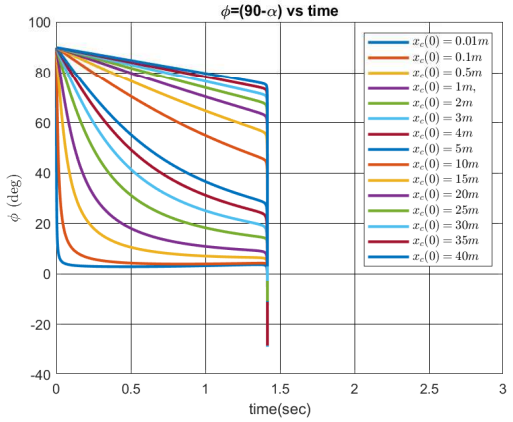

Figure 5.19: ϕ vs time for various initial positions of car (90-α)

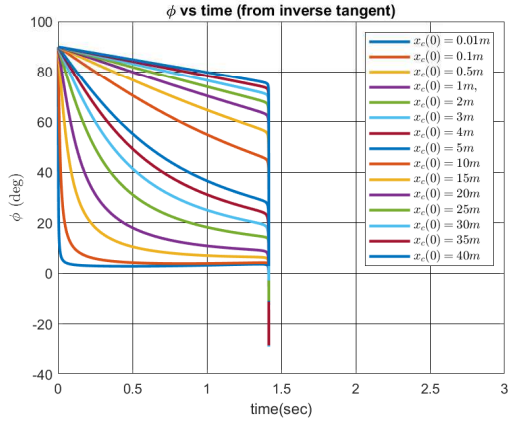

Figure 5.20: ϕ vs time for various initial positions of car (via inverse tangent)

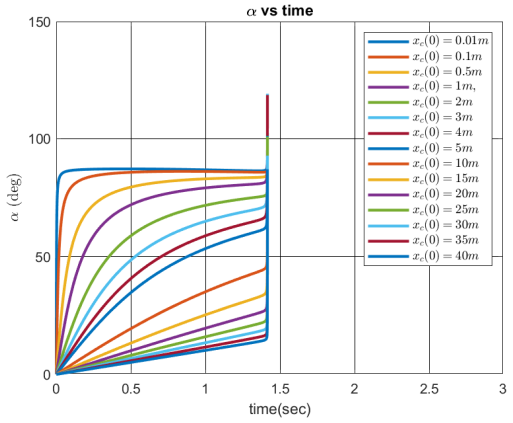

Figure 5.21: α vs time for various initial positions of car

5.5 Summary and Conclusions

We took ideas from observing how baseball or cricket outfielders intercept a ball. We proposed an open loop and a closed loop control policy to intercept the tossed ball by a car in the same plane of the trajectory. The open loop policy is based on the simple idea of sending the car to the range of the ball before the flight time. This is based on perfect information of the range and time of flight of the ball motion, which in practice is not helpful. The closed loop policy is based on the Optical Angular cancellation (OAC) law that doesn't need knowledge of initial launch conditions of the ball. It is showed how the control law based on OAC is effective in intercepting the ball successfully with an error margin of less than 1/10th the car length. A stability proof for the line of sight angle is also provided to solidify the claim.

Chapter 6

SUMMARY, CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH

6.1 Summary of Work

This thesis addressed control issues that are important to achieve the design and development of a robotic catcher. The following summarizes key themes within the thesis.

1. **Literature Survey.** A fairly comprehensive literature survey of relevant work was presented.

2. **Modeling the Trajectories.** The model of the trajectory of the tossed ball was presented in varying drag mediums. Analysis was done to show how the trajectory changes when the medium is ideally resistant or quadratically resistant.

3. **Estimating the Trajectories.** Every model has limitations and are subject to measurement noises that can give unpredictable results. To get correct measurements, results from linear model based and nonlinear model based estimators were put forth. Kalman Bucy Filter and Extended Kalman Filter were demonstrated.

4. **Hardware and Control.** The hardware to be used was described and suitable control trade offs were demonstrated.

5. **Control Law for Interception.** Open loop and closed loop interception Control Laws were described in details and validated with necessary simulations.

6.2 Directions for Future Research

Future work will involve each of the following:

- **Modeling.** The trajectory modeling will incorporate effects of spin, longitudinal wind and other forms of disturbances to make it a more realistic computation. More efforts towards realizing a closed form analytical solution for the projectile motion with quadratic drag problem.

- **Estimation.** Different types of noises and uncertainties will be taken into account to estimate the trajectories correctly. Nonlinear filters like particle filter will be explored and trade studies will be done.

- **Hardware.** Work on hardware will start with better components. Multiple processors will be used for computationally intense work; e.g. onboard optimization and decision making.

- **Catchers and Throwers** Multiple catchers and throwers will be built to realize a robotic circus.

REFERENCES

[1] A. Rodriguez, "Control and trajectory following for single and multiple non-holonomic differential drive robots: Critical design tradeoffs," in *Submitted to American Controls Conference, July 2019*. IEEE, 2019.

[2] U. Frese, B. Bauml, S. Haidacher, G. Schreiber, I. Schäfer, M. Hahnle, and G. Hirzinger, "Off-the-shelf vision for a robotic ball catcher," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2001, pp. 1623–1629.

[3] Z. Yu, Y. Liu, Q. Huang, X. Chen, W. Zhang, J. Li, G. Ma, L. Meng, T. Li, and W. Zhang, "Design of a humanoid ping-pong player robot with redundant joints," in *Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on*. IEEE, 2013, pp. 911–916.

[4] M. K. McBeath, D. M. Shaffer, and M. K. Kaiser, "How baseball outfielders determine where to run to catch fly balls," *Science*, vol. 268, no. 5210, pp. 569–573, 1995.

[5] E. Aboufadel, "A mathematician catches a baseball," *The American mathematical monthly*, vol. 103, no. 10, pp. 870–878, 1996.

[6] R. Mori and F. Miyazaki, "Examination of human ball catching strategy through autonomous mobile robot," *Transactions of the Society of Instrument and Control Engineers*, vol. 38, no. 6, pp. 543–548, 2002.

[7] K. Puttannaiah, "$\mathcal{H}^\infty$ control design via convex optimization: Toward a comprehensive design environment," M.S. Thesis, Arizona State University, Tempe, AZ, 2013.

[8] K. Puttannaiah, J. A. Echols, and A. A. Rodriguez, "A generalized $\mathcal{H}^\infty$ control design framework for stable multivariable plants subject to simultaneous output and input loop breaking specifications," in *IEEE American Control Conference, 2015*, July 2015, pp. 3310–3315.

[9] K. Puttannaiah, J. A. Echols, K. Mondal, and A. A. Rodriguez, "Analysis and use of several generalized $\mathcal{H}^\infty$ mixed sensitivity framework for stable multivariable plants subject to simultaneous output and input loop breaking specifications," in *Proceedings of the 54th IEEE Conference on Decision and Control, 2015*, Dec 2015, pp. 6617–6622.

[10] K. Puttannaiah, A. A. Rodriguez, K. Mondal, J. A. Echols, and D. G. Cartagena, "A generalized mixed-sensitivity convex approach to hierarchical multivariable inner-outer loop control design subject to simultaneous input and output loop breaking specifications," in *IEEE American Control Conference, 2016*, July 2016, pp. 5632–5637.

[11] J. A. Echols, K. Puttannaiah, K. Mondal, and A. A. Rodriguez, "Fundamental control system design issues for scramjet-powered hypersonic vehicles," in *AIAA Guidance, Navigation & Control Conference*. American Institute of Aeronautics and Astronautics, 2015. [Online]. Available: http://dx.doi.org/10.2514/6.2015-1760

[12] K. Puttannaiah, "A generalized $\mathcal{H}^\infty$ mixed sensitivity convex approach to multivariable control design subject to simultaneous output and input loop-breaking specifications," Ph.D. Dissertation, Arizona State University, Tempe, AZ, 2018.

[13] K. Puttannaiah, K. Mondal, and A. A. Rodriguez, "A generalized mixed sensitivity approach to hierarchical control design subject to specifications at several loop breaking points," in *2019 Annual American Control Conference (ACC)*, 2019, submitted.

[14] A. Sarkar, K. Puttannaiah, and A. A. Rodriguez, "Inner-outer loop based robust active damping for lcl resonance in grid-connected inverters using grid current feedback," in *2018 Annual American Control Conference (ACC)*, June 2018, pp. 6766–6771.

[15] D. G. Cartagena, K. Puttannaiah, and A. A. Rodriguez, "Modeling of a multicore processor thermal dynamics for development of dynamic thermal management controllers," in *IEEE American Control Conference, 2016*, July 2016, pp. 6917–6922.

[16] Y. Imai, A. Namiki, K. Hashimoto, and M. Ishikawa, "Dynamic active catching using a high-speed multifingered hand and a high-speed vision system," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, vol. 2. IEEE, 2004, pp. 1849–1854.

[17] G. Bätz, A. Yaqub, H. Wu, K. Kühnlenz, D. Wollherr, and M. Buss, "Dynamic manipulation: Nonprehensile ball catching," in *Control & Automation (MED), 2010 18th Mediterranean Conference on*. IEEE, 2010, pp. 365–370.

[18] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Transactions on Robotics*, vol. 30, no. EPFL-ARTICLE-198748, 2014.

[19] G. Parker, "Projectile motion with air resistance quadratic in the speed," *American Journal of Physics*, vol. 45, no. 7, pp. 606–610, 1977.

[20] J. C. Hayen, "Projectile motion in a resistant medium: Part i: exact solution and properties," *International journal of non-linear mechanics*, vol. 38, no. 3, pp. 357–369, 2003.

[21] E. W. Packel and D. S. Yuen, "Projectile motion with resistance and the lambert w function," *The College Mathematics Journal*, vol. 35, no. 5, pp. 337–350, 2004.

[22] C. H. Belgacem, "Analysis of projectile motion with quadratic air resistance from a nonzero height using the lambert w function," *Journal of Taibah University for Science*, vol. 11, no. 2, pp. 328–331, 2017.

[23] K. Yabushita, M. Yamashita, and K. Tsuboi, "An analytic solution of projectile motion with the quadratic resistance law using the homotopy analysis method," *Journal of Physics A: Mathematical and theoretical*, vol. 40, no. 29, p. 8403, 2007.

[24] Q. Jia, X. Li, J. Song, X. Gao, G. Chen, and H. Zhang, "Projectile motion aerodynamic parameter identification and simulation," in *Industrial Electronics and Applications (ICIEA), 2014 IEEE 9th Conference on.* IEEE, 2014, pp. 1872–1876.

[25] S. Ray and J. Fröhlich, "An analytic solution to the equations of the motion of a point mass with quadratic resistance and generalizations," *Archive of Applied Mechanics*, vol. 85, no. 4, pp. 395–414, 2015.

[26] V. Chistyakov and K. Malykh, "A precise parametric equation for the trajectory of a point projectile in the air with quadratic drag and longitudinal or side wind," in *Mechanics-Seventh Polyakhov's Reading, 2015 International Conference on.* IEEE, 2015, pp. 1–5.

[27] M. Turkyilmazoglu, "Highly accurate analytic formulae for projectile motion subjected to quadratic drag," *European Journal of Physics*, vol. 37, no. 3, p. 035001, 2016.

[28] "Drag of a Sphere: NASA," https://www.grc.nasa.gov/WWW/K-12/airplane/dragsphere.html, accessed: 2018-08-20.

[29] R. Sumukha, S. G. Koolagudi, V. Naresh, F. Afroz, and Y. A. Reddy, "Realistic golf flight simulation," in *Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on.* IEEE, 2016, pp. 2215–2219.

[30] K. Mundhra, A. Suluh, T. Sugar, and M. McBeath, "Intercepting a falling object: Digital video robot," in *ICRA*, 2002, pp. 2060–2065.

[31] R. Saucier, "Analytical approximations of projectile motion for linear and quadratic air drag," Tech. Rep., 2012.

[32] "Reynold's Number: NASA," https://www.grc.nasa.gov/WWW/BGH/reynolds.html, accessed: 2018-08-20.

[33] "Flight Equations with Drag: NASA," https://www.grc.nasa.gov/WWW/K-12/airplane/flteqs.html, accessed: 2018-08-20.

[34] A. A. Rodriguez, *Multivariable Control System Design.* Control3D LLC, 2010, vol. 1.

[35] "An Introduction to The Extended Kalman Filter," http://www.goddardconsulting.ca/extended-kalman-filter.html, accessed: 2018-08-20.

APPENDIX A

MATLAB CODE

```
%
%
%*************************************************************
%*************************************************************
%
% PURPOSE
%
% This routine examines projectile motion of a particle in a
% gravitational field subject to:
%
% 1. Quadratic (NonLinear) Drag Model
% 2. Linear Drag Model
%
%
% ASSUMPTIONS:
%
% Projectile: Golf ball
% Medium:     Air
%
%*************************************************************
%*************************************************************
%
% Nirangkush Das, A.A. Rodriguez, all rights reserved.
% 05-18-18
%


%
%*************************************************************
%*************************************************************
%
% Medium Properties
%

rho = 1.2041;         % Density of Air (kg/m^3)
% 20 deg C (68 deg F)
% Speed of Sound = 343.21 m/sec
% Characteristic specific impedance 413.3 Pa s/m

g   = 9.81;           % Acceleration due to gravity (m/s^2)
% at Earth surface

%
%*************************************************************
%*************************************************************
%
% Projectile Properties
```

```
%

m = 0.04593;          % Mass (kg)
d = 0.04277;          % Diameter of spherical projectile (m)
A = pi*d^2/4;         % Silhouette area (m^2)

Cd = [0 0.45 1];         % 3 Drag Coefficients
C  = 0.5*Cd*A*rho/m;  % Drag Force in gees
%Type of Surface = Recess Dimples


%
%*************************************************************
%*************************************************************
%
% Launch Conditions: Initial Velocity and Angle
%

V0    = 10;           % Initial launch Velocity (m/s)
beta0 = 45;           % initial launch angle (degrees)

%
%*************************************************************
%*************************************************************
%
% FLIGHT TIME ESTIMATE - TIME FOR SIMULATION
%
% ASSUMPTION: Used zero drag flight time
%             Provides upper bouncd on real-world flight time
%

tf   = 2*V0*sind(abs(beta0))/g;  % time of flight with No Drag
tvec = 0:0.01:tf;          % time vector



%
%*************************************************************
%*************************************************************
%*************************************************************
%*************************************************************
%*************************************************************
%*************************************************************
%
% 1. QUADRATIC (NONLINEAR) DRAG MODEL
%
%
```

```
% Initial Condition vector for Simulation
%

IC = [0;V0*cosd(beta0);0;V0*sind(beta0)];

%
% Form the Non-Linear function of the model for Simulation
%

f = @(t,a1,C)[a1(2);-C*sqrt(a1(2)^2 + a1(4)^2)* a1(2); a1(4);
-g-C*sqrt(a1(2)^2 + a1(4)^2)* a1(4)];

%
% This routine used ode45 to solve the system of Non-Linear equations.
%

for i=1:length(C)

[t,xa] = ode45(@(t,a1) f(t,a1,C(i)),[tvec],IC);

x(i,:)  = xa(:,1);       % Storing the results
vx(i,:) = xa(:,2);       % Storing the results
y(i,:)  = xa(:,3);       % Storing the results
vy(i,:) = xa(:,4);       % Storing the results


v(i,:) = sqrt(vx(i,:).^2+vy(i,:).^2); % Computing the
% particle and storing it


theta(i,:) = atand(vy(i,:)./vx(i,:)); % Computing the
% projectile with horizontal and storing it.
pause(0.05)

end
```

This part clears out negative values for x,y,. Also end values

```
x(y<0) = NaN;x(1,end)=NaN;
y(y<0) = NaN;y(1,end)=NaN;

%
% End values for v,vx,vy,theta needs to be NaN
% for getting discontinuous plot
%

vx(1,end)    = NaN;vx(2,end)=NaN;vx(3,end)=NaN;
```

```
vy(1,end)    = NaN;vy(2,end)=NaN;vy(3,end)=NaN;
theta(1,end) = NaN;theta(2,end)=NaN;theta(3,end)=NaN;
v(1,end)     = NaN;v(2,end)=NaN;v(3,end)=NaN;

%
%*************************************************************
%*************************************************************
%
% Particle Trajectory for Quadratic Drag Model
%

figure(10)
plot(x(1,:),y(1,:),'k',x(2,:),y(2,:),'r',x(3,:),y(3,:),'g')
title('Trajectory for Quadratic (Nonlinear) Drag Model')
xlabel('Horizontal Displacement x (m)')
ylabel('Vertical Displacement y (m)')
legend('$C_d=0$','$C_d=1$','$C_d=2$')
set(legend,'Interpreter','latex')
grid on
%axis([0 5 -5 5])

%*************************************************************
%*************************************************************
%*************************************************************
%*************************************************************
%*************************************************************
%*************************************************************
%*************************************************************
%*************************************************************
%
% 2. LINEAR DRAG MODEL
%
% Projectile conditions:
%
%       linear drag model
%       same launch conditions
%       same drag coefficients
%       Solved using lsim


v_tlin = g./C ;      % Form terminal velocity of the Linear System.

%
%
% Form the A,B,C,D matrix from the State Space representation
% System. There are 3 System Model Matrix A, since 3 values of
% Coefficients are considered. Matrices B,C,D remain the same.
%
```

```
a1=[0 1 0 0;
0 -g/v_tlin(1) 0 0;
0 0 0 1;
0 0 0 -g/v_tlin(1)];

a2=[0 1 0 0;
0 -g/v_tlin(2) 0 0;
0 0 0 1;
0 0 0 -g/v_tlin(2)];

a3=[0 1 0 0;
0 -g/v_tlin(3) 0 0;
0 0 0 1;
0 0 0 -g/v_tlin(3)];

b=[0;0;0;-g];
c=eye(4,4);
d=0*eye(4,1);

%
% Form the system for 3 different drag coefficients.
%

sys1=ss(a1,b,c,d);
sys2=ss(a2,b,c,d);
sys3=ss(a3,b,c,d);

%
% Create the input vector u.
%

u=ones(length(tvec),1);

%
% Simulate using lsim. The 'tvec' and 'IC' come from previ
% for Quadratic Drag.
%

[y1,t,x1]=lsim(sys1,u,tvec,IC);
[y2,t,x2]=lsim(sys2,u,tvec,IC);
[y3,t,x3]=lsim(sys3,u,tvec,IC);

%
% Compute the Instantaneous Velocity Vectors
%
```

```
v1=sqrt(x1(:,2).^2+x1(:,4).^2);
v2=sqrt(x2(:,2).^2+x3(:,4).^2);
v3=sqrt(x3(:,2).^2+x3(:,4).^2);

%
% Compute the Instantaneous Angle Vectors
%

theta1=atand(x1(:,4)./x1(:,2));
theta2=atand(x2(:,4)./x2(:,2));
theta3=atand(x3(:,4)./x3(:,2));


%****************************************************************
%****************************************************************
%****************************************************************
%****************************************************************
%
% This section has been commented out.
%


% for i=1:length(C)
%
% v_tlin(i,:) = g./C(i);          % Terminal Velocity
%
%
% % Computing the horizontal component of Velocity.
% vx_lin(i,:) = V0.*cosd(beta0).*exp(-g.*tvec./v_tlin(i));
%
% % Computing the vertical component of Velocity.
% vy_lin(i,:) = V0.*sind(beta0).*exp(-g.*tvec./v_tlin(i))-
%
% % Computing the horizontal Displacement.
% x_lin(i,:) = (V0*v_tlin(i)*cosd(beta0)/g).*(1-exp(-g.*t
%
% % Computing the vertical Displacement.
% y_lin(i,:) = v_tlin(i)/g*(V0*sind(beta0)+v_tlin(i)).*(1
%
%
% % Computing the Instantaneous Velocity of the Particle.
% v_lin(i,:) = sqrt(vx_lin(i,:).^2+vy_lin(i,:).^2);
%
% % Computing the Instantaneous Angle of projectile with
% theta_lin(i,:) = atand(vy_lin(i,:)./vx_lin(i,:));
%
% pause(0.05)
```

```
%
%
% end
%
% %
% % This part clears out negative values for x,y,.
% % Also end values need to be NaN for discotinuous plot
% %
%
% x_lin(y_lin<0) = NaN;x_lin(1,end)=NaN;
% y_lin(y_lin<0) = NaN;y_lin(1,end)=NaN;
%
% %
% % End values for v, vx, vy, theta needs to be NaN
% %
%
% vx_lin(1,end)    = NaN;vx_lin(2,end)=NaN;vx_lin(3,end)=NaN;
% vy_lin(1,end)    = NaN;vy_lin(2,end)=NaN;vy_lin(3,end)=NaN;
% theta_lin(1,end) = NaN;theta_lin(2,end)=NaN;theta_lin(3,en
% v_lin(1,end)     = NaN;v_lin(2,end)=NaN;v_lin(3,end)=NaN;



%
%*************************************************************
%*************************************************************
%*************************************************************
%*************************************************************
%

%
% Trajectories for both Quadratic and Linear Drag Models
%

figure(20)
plot(x(1,:),y(1,:),'k',x(2,:),y(2,:),'r',x(3,:),y(3,:),'g',
title('Trajectory: Quadratic (Solid) & Linear Drag (Dashed)')
xlabel('Horizontal Displacement x (m)')
ylabel('Vertical Displacement y (m)')
legend('$C_d=0$','$C_d=1$','$C_d=2$','$C_d=0$','$C_d=1$','$C_d=2$')
set(legend,'Interpreter','latex')
grid on

%
%
%*************************************************************
%*************************************************************
```

```
%
% Horizontal Displacement with time
% for both Quadratic and Linear Drag Models
%

figure(30)
plot(t',x(1,:),'k',t',x(2,:),'r',t',x(3,:),'g',t',x1(:,1),--')
title('Horz. Disp. with time: Quadratic(Solid) & Linear Drag(Dashed)')
xlabel('time (sec)')
ylabel('Horizontal Displacement x (m)')
legend('$C_d=0$','$C_d=1$','$C_d=2$','$C_d=0$','$C_d=1$','$C_d=2$')
set(legend,'Interpreter','latex')
grid on

%
%************************************************************
%************************************************************
%
% Vertical Displacement with time
% for both Quadratic and Linear Drag Models
%

figure(40)
plot(t',y(1,:),'k',t',y(2,:),'r',t',y(3,:),'g',t',x1(:,3)--')
title('Vert. Disp. with time: Quadratic(Solid) & Linear Drag(Dashed)')
xlabel('time (sec)')
ylabel('Vertical Displacement y (m)')
legend('$C_d=0$','$C_d=1$','$C_d=2$','$C_d=0$','$C_d=1$','$C_d=2$')
set(legend,'Interpreter','latex')
grid on

%
%************************************************************
%************************************************************
%
% Horizontal Component of Velocity with time
% for both Quadratic and Linear Drag Models
%

figure(50)
plot(t',vx(1,:),'k',t',vx(2,:),'r',t',vx(3,:),'g',t','g--')
title('Horizontal Velocity: Quadratic (Solid) & Linear Drag (Dashed)')
xlabel('time (sec)')
ylabel('Horizontal Velocity (m/s)')
legend('$C_d=0$','$C_d=1$','$C_d=2$','$C_d=0$','$C_d=1$','$C_d=2$')
set(legend,'Interpreter','latex')
grid on
```

```
%
%************************************************************
%************************************************************
%
% Vertical Component of Velocity with time
% for both Quadratic and Linear Drag Models
%

figure(60)
plot(t',vy(1,:),'k',t',vy(2,:),'r',t',vy(3,:),'g',t','g--')
title('Vertical Velocity: Quadratic (Solid) & Linear Drag (Dashed)')
xlabel('time (sec)')
ylabel('Vertical Velocity (m/s)')
legend('$C_d=0$','$C_d=1$','$C_d=2$','$C_d=0$','$C_d=1$','$C_d=2$')
set(legend,'Interpreter','latex')
grid on

%
%************************************************************
%************************************************************
%
% Instantaneous Velocity of Projectile versus time
% for both Quadratic and Linear Drag Models
%

figure(70)
plot(t',v(1,:),'k',t',v(2,:),'r',t',v(3,:),'g',t',v1(:,1),--')
title('Velocity: Quadratic (Solid) & Linear Drag (Dashed)')
xlabel('time (sec)')
ylabel('Velocity (m/s)')
legend('$C_d=0$','$C_d=1$','$C_d=2$','$C_d=0$','$C_d=1$','$C_d=2$')
set(legend,'Interpreter','latex')
grid on

%
%************************************************************
%************************************************************
%
% Instantaneous Angle of Velocity Projectile with horizontal versus
% for both Quadratic and Linear Drag Models
%

figure(80)
plot(t',theta(1,:),'k',t',theta(2,:),'r',t',theta(3,:),
title('Angle: Quadratic (Solid) & Linear Drag (Dashed)')
xlabel('time (sec)')
```

```
ylabel('Angle (degree)')
legend('$C_d=0$','$C_d=1$','$C_d=2$','$C_d=0$','$C_d=1$','$C_d=2$')
set(legend,'Interpreter','latex')
grid on


%
%*****************************************************************
%*****************************************************************
%*****************************************************************
%*****************************************************************
%*****************************************************************
%*****************************************************************
%*****************************************************************
%*****************************************************************
%
% 3. FLIGHT EQUATIONS WITH DRAG CONSIDERING VERTICAL AND HORIZONTAL
% (IDEAS AND EQUATIONS FROM NASA WEBPAGE)
%
% Projectile conditions:
%
%       Quadratic drag model
%       same launch conditions
%       same drag coefficients
%       Terminal velocity for Quadratic Drag employed
%       Solved using ode45


CdT = [0 1.4 2.8];         % 3 Drag Coefficients
CT  = 0.5*CdT*A*rho/m;  % Drag Force in gees


%
% Form Terminal velocity for Quadratic Drag Model
%
V_t=sqrt(g./CT);

%
% Form function for simulation with ode45
%

f = @(t,a1,V_t)[a1(2);-g*(a1(2)^2)/V_t^2; a1(4);
-g-g*(a1(4)^2)/V_t^2];


for i=1:length(V_t)
```

```
[t,Xa] = ode45(@(t,a1) f(t,a1,V_t(i)),[tvec],IC);

X(i,:)  = Xa(:,1);       % Storing the results fo
Vx(i,:) = Xa(:,2);       % Storing the results for
Y(i,:)  = Xa(:,3);       % Storing the results for
Vy(i,:) = Xa(:,4);       % Storing the results for

pause(0.05)

end



figure(90)
plot(x(1,:),y(1,:),'k',x(2,:),y(2,:),'r',x(3,:),y(3,:),'g',
title('Trajectory:Original Quad model(solid)vs Toss Mod
xlabel('Horizontal Displacement x (m)')
ylabel('Vertical Displacement y (m)')
legend('$C_d=0$','$C_d=1$','$C_d=2$','$C_d=0$','$C_d=1
set(legend,'Interpreter','latex')
grid on
```

Curve Fits for Quadratic and Linear Model on Balls.m

```
X3=X(2,:)';
Y3=Y(2,:)';

% Least Square Fit
H7=[ones(length(Y3),1),X3,X3.^2];
Astar7=zeros(3);
Ytilde7=zeros(length(Y3),3);
R7=zeros(length(Ytilde7),3);
Astar7=inv(H7'*H7)*H7'*Y3;
Ytilde7=H7*Astar7;
R7=Y3-Ytilde7;

% Cubic Polynomial Fit
H8=[ones(length(Y3),1),X3,X3.^2,X3.^3];
Astar8=zeros(4);
Ytilde8=zeros(length(Y3),4);
R8=zeros(length(Ytilde8),4);
Astar8=inv(H8'*H8)*H8'*Y3;
Ytilde8=H8*Astar8;
R8=Y3-Ytilde8;

% Weighted Least Squares Fit
v3=0:1/(length(Y3)-1):1;
```

```
W3=diag(v3);

H9=[ones(length(Y3),1),X3,X3.^2];
Astar9=zeros(3);
Ytilde9=zeros(length(Y3),3);
R9=zeros(length(Ytilde9),3);
Astar9=inv(H9'*W3*H9)*H9'*W3*Y3;
Ytilde9=H9*Astar9;
R9=Y3-Ytilde9;

plot(X3,Y3,'k',X3,Ytilde7,'r',X3,Ytilde8,'g',X3,Ytilde9,'m')
title('NASA Toss Model of Golf Ball (C_d=0.45) with Least
xlabel('Horizontal Displacement x (m)')
ylabel('Vertical Displacement y (m)')
legend('Original Toss Model','Quadratic LSF','Cubic LSF',
set(legend,'Interpreter','latex')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',0.8);
grid on
axis([0 11 0 3])
hold on

%
%
%*************************************************************
%*************************************************************
%
% PURPOSE
% This rountine examines how a Kalman Filter can be used to
% estimate the trajectory of a partcle subjected to Linear
%
% ASSUMPTION:
% zero mean process noise xi with unit intesity for x, vx, y, vy
% zero mean output noise with intensity mu for vx and vy
%
%*************************************************************
%*************************************************************
%
% A.A. Rodriguez, all rights reserved
% 05-18-18
%
%
% NonLinear Equations for Planar xy Motion of a Point Mass with
% Linear Drag = - beta v where v = sqrt( vx^2 + vy^2)
%
% Variables:
%               x1 = x  (m)
%               x2 = vx (m/sec)
```

```
%               x3 = y  (m)
%               x4 = vy (m/sec)
%
% x1_dot  =              x2;
% x2_dot  =  - \beta * x2;
% x3_dot  =              x4;
% x4_dot  =    - gravit_accel*1 - \beta * x4;


% Medium Properties
%

rho = 1.2041;         % Density of Air (kg/m^3)
% 20 deg C (68 deg F)
% Speed of Sound = 343.21 m/sec
% Characteristic specific impedance 413.3 Pa s/m

gravit_accel   = 9.81;           % Acceleration due to gravity
% at Earth surface

%
%*************************************************************
%*************************************************************
%
% Projectile Properties
%

m = 0.04593;          % Mass (kg)
d = 0.04277;          % Diameter of spherical projectile (m)
A = pi*d^2/4;         % Silhouette area (m^2)

Cd = 0.45;            % 3 Drag Coefficients
beta  = 0.5*Cd*A*rho/m;  % Drag Force in gees


%
%*************************************************************
%*************************************************************
%
%
% PLANT ICs
%
% xinit  = 5;       % RED
% vxinit = 1;       % GREEN
% yinit  = 40;      % BLUE
% vyinit = -5;      % MAGENTA
```

```
xinit  = 0;          % RED
vxinit = 7.071;      % GREEN
yinit  = 0;          % BLUE
vyinit = 7.071;      % MAGENTA
%

%
% ESTIMATOR ICs
%
% xhatinit  = -5;   % RED
% vxhatinit = 6;    % GREEN
% yhatinit  = 5;    % BLUE
% vyhatinit = 3;    % MAGENTA

xhatinit  = -1;   % RED
vxhatinit = 1.25*7.071;    % GREEN
yhatinit  = 1;    % BLUE
vyhatinit = 0.75*7.071;    % MAGENTA

%
%*************************************************************
%*************************************************************
%
% Form Initial Conditions for Plant and Estimator
%
xo = [ xinit vxinit yinit vyinit xhatinit vxhatinit yhati


%
%*************************************************************
%*************************************************************
%
% FORM TIME VECTOR
gravit_accel = 9.81; % accelearation due to gravity in m/sec
tinit  = 0;                     % Initial Time
tinc   = 0.01;                  % Time Increment
tfinal = 4;
%%tfinal = 1.001*(-vyinit - sqrt( vyinit^2 + 2*gravit_acce
t      = [tinit:tinc:tfinal];   % Time Vector


%
%*************************************************************
%*************************************************************
%
% Desured KF OPEN LOOP BW
%
```

```
Desired_KFBW = 4*5/(tfinal);   % Choose KF BW parameter mu
% to achieve this open loop KF BW
% Corresponds to a 5tau settling time of
% 0.25* tfinal; i.e.  5tau = 0.25 tfinal
% or Desired_KFBW = 1/tau = 4*5/tfinal


%
%************************************************************
%************************************************************
%
% NON IDEA IDEAL PARABOLIC PHYSICS TRAJECTORY - LINEAR DRAG
% Nonlinear Model becomes linear
%
% LINEAR PLANT/SYSTEM:      xdot = a x + b u + l xi
%                           y    = c x + d u + theta
%

gravit_accel = 9.81; % accelearation due to gravity in m/sec

a = [  0   1   0   0
0   -beta   0   0
0   0   0   1
0   0   0   -beta            ];
b = [  0   0   0  -gravit_accel ]'; % u = unit step
l =    eye(4);                      % State Process Noise Input Matrix
c = [  1   0   0   0
0   0   1   0            ]; % Selects x and y coordinates
d = [  0*ones(2,1)              ];

%
%************************************************************
%************************************************************
%
% KALMAN FILTER DESIGN VIA LQR DUALITY
%
statewm    = l*l';         % State Weighting Matrix (l*l')
% Must be symmetric and
% at least Positive Semidefinite

rho        = 0.02;         % Control Weighting Scalar
% Miust choose small enough so that KF
% is MUCH faster than flight time
% want convergence to be around tfinal/2 maximum
controlwm  = rho*eye(2,2); % Control Weighting Matrix
% Must be symmetric and Postivie Definite
```

```
g          = lqr(a', c', statewm, controlwm);
h          = g';


%
% NOTE: Selection of state weighting matrix assumes that
%       Process Noise Intensity Matrix = eye(4,4)
%
mu         = rho;          % Sensor Noise Intensity Scalar
sensintenm = controlwm';   % Sensor Noise Intensity Matrix
eig(a-h*c); % must be stable; real(eigenvalues) < 0
% requires (a,c) detectable or (a',c') stabilizable

%***************************************************************
%***************************************************************
%
% FORM PLANT AND KALMAN FILTER
%
% PLANT/SYSTEM:             xdot = a x + b u + l xi
%                           y    = c x + d u + theta
%
% MODEL BASED ESTIMATOR:    xhatdot = a xhat + bu + l
%                           yhat    = c xhat + du +   mtheta
%
% COMBINING THE ABOVE:
%
%
%
% | ytilde | = | y - yhat | = | c   -c | | x   |
%                                        | xhat|
%  or
%
% zdot   = m z + n u + p xi      +  pp mxi + q  (y - yhat)
% ytilde = r z       + s thetavec
%
ns = size(a)*[0 1 ]'; % Number of States
m = [ a       0*a
0*a     a   ];
n = [ b
b           ];
p = [ l
0*l         ];
q = [ 0*h
h           ];

r = [ c            -c  ];
s = eye(2,2);
```

```
%
%************************************************************
%************************************************************
%
% CLOSE MODEL-BASED ESTIMATOR LOOP ON OUTPUT ESTIMATION ERROR
%
% zdot   = m z + n u + p xivec + q ytilde
% ytilde = r z + s thetavec
%
% or
%
% zdot   = m z + n u + p xivec + q (r z + s thetavec)
% ytilde = r z + s thetavec
%
% or
%
% zdot   = (m + qr) z  + n u + p xivec + q * s  thetavec
% ytilde = r z         + 0 u              + s thetavec
%
% or
%
% zdot   = acl z  + n u + p xivec + q * s  thetavec
% ytilde = r z    + 0 u           + s thetavec
%

%
%************************************************************
%************************************************************
%
% EXTERNAL SIGNALS
%
% u = UNIT STEP
%
% xivec =  [ white noise with intensity 1;
%            0 ];
% thetavec = [ white noise with intensity mu
%               0 ];

%
%************************************************************
%************************************************************
%
% FORM u to ytilde
%
%
acl = m + q*r;
```

```
bcl = [n p q];
ccl = r;
dcl = [0*eye(2,5) eye(2,2)];
sysu2ytilde = ss(acl,n,r,0*eye(2,1));

%
%*************************************************************
%*************************************************************
%
% COMPUTE ESTIMATOR CLOSED LOOP POLES
%
eig(acl) % CONTAINS PLANT POLES AND
% CLPs of ESTIMATOR; CLPs of ESTIMATOR should be STABLE !

damp(eig(acl))



%
%*************************************************************
%*************************************************************
%
% FORM NOMINAL TIME RESPONSE
%
u = 1*ones(1,length(t));         % Unit Step

%
%*************************************************************
%*************************************************************
%
% Compute Nominal
%
%       Output Estimation Error and State Estimation Error
%       Plant States and Estimates
%
[ytilde,t,z] = lsim(sysu2ytilde, u, t,xo);

%
% Output Estimation Error
%
figure(8)
plot(t, ytilde(:,1),t, ytilde(:,2))
title('Output Estimation Error')
xlabel('t (sec)')
ylabel('$\tilde{y} = y - \hat y$','Interpreter','latex')

set(legend,'Interpreter','latex')
```

```
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on

%
% State Estimation Error
%
figure(10)
title('State Estimation Error')
xlabel('t (sec)')
ylabel('$\tilde{x} = x - \hat x$','Interpreter','late
set(legend,'Interpreter','latex')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on

%
% Plant States and Estimator States
%
figure(12)
title('Plant States (solid) and Estimator States (dashed)')
xlabel('t (sec)')
ylabel('x') x4$','Location','southwest')
set(legend,'Interpreter','latex')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1);
grid on

%
% % Plant State Estimates
% %
% figure(14)
% plot(t, z(:,1),'r', t, z(:,2), 'g', t, z(:,3), 'b',t
% title('Plant States')
% xlabel('t (sec)')
% ylabel('$\hat{x}$','Interpreter','latex')
% grid on
% %
% %
% %  Estimator States
% %
% figure(16)
% plot(t, z(:,5),'--r', t, z(:,6), '--g', t, z(:,7), '--b'
% title('Estimator States')
% xlabel('t (sec)')
% ylabel('$\hat{x}$','Interpreter','latex')
% grid on
```

```
% %

%
%*************************************************************
%*************************************************************
%
% Trajectory Slope
%
figure(18)
plot(t,  z(:,4)./z(:,2))
title('Trajectory Slope')
xlabel('t (sec)')
ylabel('dy/dx')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on

%
%*************************************************************
%*************************************************************
%
% Trajectory
%
figure(20)
plot(z(:,1), z(:,3))
title('Trajectory')
xlabel('x (m)')
ylabel('y (m)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on

%
%*************************************************************
%*************************************************************
%
% Trajectory Angle
%
figure(22)
plot(t,  (180/pi)*atan( z(:,4)./z(:,2)))
title('Trajectory Angle')
xlabel('t (sec)')
ylabel('\theta (deg)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on
```

```
%
%*************************************************************
%*************************************************************
%
% FORM FREQUENCY VECTOR
%
winit = 10^-2;   % Initial frequency
wfin  = 10^2;    % Final frequency
nfps  = 1000;    % Number of Frequency Points
w     = logspace(log10(winit), log10(wfin),nfps); % Vector of

%
%*************************************************************
%*************************************************************
%
% PLANT SINGULAR VALUES
%
figure(5)
plant = ss(a,b,c,d);
clsv  = sigma(plant,w); % PLANT
semilogx(w,20*log10(clsv))
title('Plant')
xlabel('w (rad/sec)')
ylabel('singular values (dB)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on

%
%*************************************************************
%*************************************************************
%
% PLANT SINGULAR VALUES - xi to y
%
figure(6)
xi2yplant = ss(a,l,c,0*ones(2,4));
xi2yclsv  = sigma(xi2yplant,w);        % PLANT - xi to y
semilogx(w,20*log10(xi2yclsv))
title('Plant: \xi to y')
xlabel('w (rad/sec)')
ylabel('singular values (dB)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on
```

```
%
%*************************************************************
%*************************************************************
%
% PLANT TRANSFER FUNCTION MATRIX
%
zpk(plant) % Plant Transfer Function Matrix

%
%*************************************************************
%*************************************************************
%
% SINGULAR VALUES - u to ytilde (SHOULD BE VERY VERY SMALL!!!)
%
figure(30)
clsv = sigma(ss(acl,bcl,ccl,dcl),w); % u to ytilde
semilogx(w,20*log10(clsv))
title('Nominal Frequency Response - Gravity to Estimation Error')
xlabel('w (rad/sec)')
ylabel('singular values (dB)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on

%
%*************************************************************
%*************************************************************
%
% ERROR DYNAMICS (Assume zero means for process and sensor noises)
%
% xtildedot = (a - h*c) xtilde + l xi - h theta
% ytilde = c xtilde + theta

%
%*************************************************************
%*************************************************************
%
% KF LOOP
%
figure(32)
kfloop    = ss(a, h, c, 0*ones(2,2) );    % KF LOOP
kfloop_sv = sigma(kfloop,w);
semilogx(w,20*log10(kfloop_sv))
title('KF Loop')
xlabel('\omega (rad/sec)')
ylabel('singular values (dB)')
h_line = findobj(gcf, 'type', 'line');
```

```
set(h_line, 'LineWidth',2);
grid on

zpk(kfloop) % Open Loop TFM for KFLOOP

%
figure(34)
kfloop    = ss(a, h, c, 0*ones(2,2) );    % KF LOOP
bode(kfloop,w)
title('KF Loop')
xlabel('$\omega$ (rad/sec)')
ylabel('singular values (dB)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on

%
%*************************************************************
%*************************************************************
%
% ALL MARGINS
%
allmargin(ss(a,h*[1 0]',[1 0]*c,0*ones(1,1)))

%
%*************************************************************
%*************************************************************
%
% KF SENSITIVITY
%
figure(36)
kfsen    = ss(a-h*c, h, -c, eye(2,2) );    % KF SENSITIVITY
kfsen_sv = sigma(kfsen,w);
semilogx(w,20*log10(kfsen_sv))
title('KF Sensitivity')
xlabel('\omega (rad/sec)')
ylabel('singular values (dB)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on

%
%*************************************************************
%*************************************************************
%
% KF COMPLEMENTARY SENSITIVITY
%
```

```
figure(38)
kfcompsen_sv = sigma(kfcompsen,w);
semilogx(w,20*log10(kfcompsen_sv))
title('KF Complementary Sensitivity')
xlabel('\omega (rad/sec)')
ylabel('singular values (dB)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on

%
%************************************************************
%************************************************************
%
% KF COMPLEMENTARY SENSITIVITY
%
figure(40)
semilogx(w,20*log10(kfsen_sv), w,20*log10(kfcompsen_sv))
title('KF Sensitivity and Complementary Sensitivity')
xlabel('$\omega$ (rad/sec)')
ylabel('singular values (dB)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on


%
%************************************************************
%************************************************************
%
% SINGULAR VALUES - xi to ytilde (SHOULD BE VERY VERY SMALL!!!)
%
figure(42)
sysxi2ytilde    = ss(acl, p, r, 0*ones(2,4) ); % xi to ytilde
sysxi2ytilde_sv = sigma(sysxi2ytilde,w);
semilogx(w,20*log10(sysxi2ytilde_sv))
title('$\xi \;\;\;\; to  \;\;\;\; \tilde{y}$','Interpreter','latex')
xlabel('\omega (rad/sec)')
ylabel('singular values (dB)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on


%
figure(44)
sysxi2ytilde1    = ss(a-h*c, l, c, 0*ones(2,4) ); % xi to ytilde
```

```
sysxi2ytilde1_sv = sigma(sysxi2ytilde1,w);
semilogx(w,20*log10(sysxi2ytilde1_sv))
title('$\xi \;\;\;\; to  \;\;\;\; \tilde{y}$','Interpreter','latex')
xlabel('\omega (rad/sec)')
ylabel('singular values (dB)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on

%
%************************************************************
%************************************************************
%
% SINGULAR VALUES - theta to ytilde (SHOULD BE VERY VERY SMALL!!!)
%
figure(50)
systheta2ytilde    = ss(acl, q*s, r, s );          % theta to ytilde
systheta2ytilde_sv = sigma(systheta2ytilde,w);
semilogx(w,20*log10(systheta2ytilde_sv))
title('$\theta \;\;\;\; to  \;\;\;\; \tilde{y}$','Interpreter','latex')
xlabel('\omega (rad/sec)')
ylabel('singular values (dB)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on


figure(52)
systheta2ytilde1    = ss(a-h*c, -h, c, eye(2,2) ); % theta to ytilde
systheta2ytilde1_sv = sigma(systheta2ytilde1,w);
semilogx(w,20*log10(systheta2ytilde1_sv))
title('$\theta \;\;\;\; to  \;\;\;\; \tilde{y}$','Interpreter','latex')
xlabel('\omega (rad/sec)')
ylabel('singular values (dB)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on


% xi to y_hat (singular values)


a_sv=[a 0*eye(4,4);h*c a-h*c];
b_sv=[l;0*l];
c_sv=[0*c c];
d_sv=0*eye(2,4);
```

```
figure(60)
sysxi2y_hat=ss(a_sv,b_sv,c_sv,d_sv);
sysxi2y_hat_sv = sigma(sysxi2y_hat,w);
semilogx(w,20*log10(sysxi2y_hat_sv))
title('$\xi \;\;\;\; to  \;\;\;\; \hat{y}$','Interpre
xlabel('\omega (rad/sec)')
ylabel('singular values (dB)')
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',2);
grid on

%
%************************************************************
%************************************************************
%
% CRITICAL DATA
%
mu

zpk(plant)   % Plant Transfer Function Matrix

Desired_KFBW % Choose mu to achieve this open loop KF BW

zpk(kfloop)  % Open Loop TFM for KFLOOP

allmargin(ss(a,h*[1 0]',[1 0]*c,0*ones(1,1)))

ans =

-6.9985
-6.9985
-1.0104
-1.0104
0
-0.0085
0
-0.0085


Pole        Damping       Frequency       Time Constant
(rad/TimeUnit)     (TimeUnit)

-7.00e+00     1.00e+00       7.00e+00          1.43e-01
-7.00e+00     1.00e+00       7.00e+00          1.43e-01
-1.01e+00     1.00e+00       1.01e+00          9.90e-01
-1.01e+00     1.00e+00       1.01e+00          9.90e-01
0.00e+00    -1.00e+00       0.00e+00              Inf
```

```
-8.47e-03     1.00e+00       8.47e-03          1.18e+02
0.00e+00    -1.00e+00       0.00e+00              Inf
-8.47e-03     1.00e+00       8.47e-03          1.18e+02

ans =

From input to output...
1:  0

-9.81
2:  --------------
s (s+0.008475)

Continuous-time zero/pole/gain model.


ans =

From input 1 to output...
8.0004 (s+0.8839)
1:  -----------------
s (s+0.008475)

-2.3553e-15 (s+0.325)
2:  ---------------------
s (s+0.008475)

From input 2 to output...
-2.3553e-15 (s+7.474)
1:  ---------------------
s (s+0.008475)

8.0004 (s+0.8839)
2:  -----------------
s (s+0.008475)

Continuous-time zero/pole/gain model.


ans =

struct with fields:

GainMargin: [10 double]
GMFrequency: [10 double]
PhaseMargin: 83.7942
PMFrequency: 8.0496
```

```
DelayMargin: 0.1817
DMFrequency: 8.0496
Stable: 0


mu =

0.0200


ans =

From input to output...
1:  0

-9.81
2:  --------------
s (s+0.008475)

Continuous-time zero/pole/gain model.


Desired_KFBW =

5


ans =

From input 1 to output...
8.0004 (s+0.8839)
1:  -----------------
s (s+0.008475)

-2.3553e-15 (s+0.325)
2:  --------------------
s (s+0.008475)

From input 2 to output...
-2.3553e-15 (s+7.474)
1:  --------------------
s (s+0.008475)

8.0004 (s+0.8839)
2:  -----------------
s (s+0.008475)
```

```
Continuous-time zero/pole/gain model.


ans =

struct with fields:

GainMargin: [10 double]
GMFrequency: [10 double]
PhaseMargin: 83.7942
PMFrequency: 8.0496
DelayMargin: 0.1817
DMFrequency: 8.0496
Stable: 0
```