Dynamical System Design for Control of Single and Multiple Non-holonomic Differential Drive Robots Based on Critical Design Trade Studies

by

Sai Sravan Manne

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

Approved August 2020 by the
Graduate Supervisory Committee:

Armando A. Rodriguez, Chair
Jennie Si
Spring Berman

ARIZONA STATE UNIVERSITY

May 2021

ABSTRACT

Over the past few decades, there is an increase in demand for various ground robot applications such as warehouse management, surveillance, mapping, infrastructure inspection, etc. This steady increase in demand has led to a significant rise in the non-holonomic differential drive vehicles (DDV) research. Albeit extensive work has been done in developing various control laws for trajectory tracking, point stabilization, formation control, etc., there are still problems and critical questions in regards to design, modeling, and control of DDV's - that need to be adequately addressed.

In this thesis, three different dynamical models are considered that are formed by varying the input/output parameters of the DDV model. These models are analyzed to understand their stability, bandwidth, input-output coupling, and control design properties. Furthermore, a systematic approach has been presented to show the impact of design parameters such as mass, inertia, radius of the wheels, and center of gravity location on the dynamic and inner-loop (speed) control design properties. Subsequently, extensive simulation and hardware trade studies have been conducted to quantify the impact of design parameters and modeling variations on the performance of outer-loop cruise and position control (along a curve). In addition to this, detailed guidelines are provided for when a multi-input multi-output (MIMO) control strategy is advisable over a single-input single-output (SISO) control strategy; when a less stable plant is preferable over a more stable one in order to accommodate performance specifications.

Additionally, a multi-robot trajectory tracking implementation based on receding horizon optimization approach is also presented. In most of the optimization-based trajectory tracking approaches found in the literature, only the constraints imposed by the kinematic model are incorporated into the problem. This thesis elaborates the fundamental problem associated with these methods and presents a systematic approach to understand and quantify when kinematic model-based constraints are sufficient and when dynamic model-based constraints are necessary to obtain good tracking properties.

Detailed instructions are given for designing and building the DDV based on performance specifications, and also, an open-source platform capable of handling high-speed multi-robot research is developed in C++.

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my thesis advisor, Professor Armando Antonio Rodriguez, for his continuous support of my studies; for sharing his immense knowledge and experience with me; and for providing the necessary facilities and resources to conduct my research. The persistent encouragement and guidance he has given me at all times kept me motivated in spite of various difficulties I faced throughout my journey at ASU, and made this thesis possible.

I would also like to thank Kaustav Mondal for the numerous discussions we had that helped me build a strong foundation for my thesis. I am also grateful to my friends and lab colleagues at ASU - Shi Lu, Abdulla, Brent, Karan, Rohit Vaddi, Vignesh and Pavitra, for their help and support throughout this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION AND OVERVIEW OF WORK

1.1 Introduction and Motivation

Over the past three decades, the promise of driverless and robotic vehicles has greatly accelerated research in the area [86]-[59]. This promise includes a wide range of application areas; e.g. search and rescue, surveillance, mapping, assisting first responders, assisting law enforcement, infrastructure inspection, warehouse logistics, and much more. The steady increase in research efforts in this area can also be partially attributed to the advances in networking, sensing, and computing technologies, which resulted in the development of powerful and cost-efficient hardware. Particularly, these new technologies (e.g., NVIDIA Jetson, Teensy, Raspberry Pi, Intel RealSense, RP LIDAR) have enabled researchers to incorporate principled methods from areas such as optimization, data science, computer vision, control theory, and machine learning, which were once considered computationally intensive. Hence, a vast amount of literature is currently available addressing various driverless/robotic vehicle outer-loop control objectives such as trajectory tracking, static and dynamic obstacle avoidance, multi-robot formation control, etc. Given this, there are still fundamental problems and critical questions that have to be adequately addressed in order to unleash the true potential of these forward-looking vehicles. This forms the primary focus of this thesis and will be presented in detail in the forthcoming paragraphs.

The work presented in this thesis is an extension of the master's thesis research conducted by Zhenyu Lin [46], and Zhicho Li [45] [1] . The central objective of their work involves utilizing off-the-shelf technologies (e.g. Arduino, Raspberry Pi, commercial RC cars) to develop cost-efficient ground robots that are capable of facilitating multi-vehicle robotic research. This is a major step intended to achieve the long-term goal of developing a fleet of Flexible Autonomous Machines Operating in an Uncertain Environment (FAME). This fleet can involve multiple ground and air robots that can work collectively in order to perform a common task. They have also thoroughly examined the kinematic and dynamic models of non-holonomic differential drive ground vehicle (DDV) and rear-wheel drive vehicle followed by a system identification procedure to estimate the nominal plant parameters. Further to this, the following outer-loop control objectives have been implemented on hardware: (1) cruise-control along a curve, (2) planar $(x - y)$ Cartesian stabilization, (3) vehicle-target spacing-control, (4) multi-robot spacing-control along line/curve, (5) tracking slowly-moving remote-controlled quadrotor, (6) avoiding obstacle while moving towards a target.

This thesis attempts to answer the following critical questions involved in the modeling, design, and control of DDV's [2] : 1) What critical parameters impact key vehicle characteristics (i.e. static, dynamic and control properties)? 2) When is a single-input single-output (SISO) controller sufficient? When is a multiple-input multiple-output (MIMO) controller necessary? 5) When is a kinematic model sufficient for design and evaluation? When is a dynamic model essential for design and evaluation? 6) How do the above impact speed and position-direction control design (along a curved path)? Further to this, a detailed literature survey has been presented in the next section, which will form the basis for outlining the central contributions of this thesis in the

[1]Zhenyu Lin and Zhico Li are former graduate students who have completed their MS Thesis work under Dr. Armando A. Rodriguez

[2]DDV - throughout this thesis, DDV will refer to non-holonomic differential drive vehicle

upcoming sections.

1.2 Literature Survey: Ground Robotics - State of the Field

As mentioned earlier, a great deal of work has been done in the areas of hardware design, modelling, and control of ground robots/vehicles (includes both holonomic and non-holonomic). An effort is made to shed light on some of the works which are most relevant to developments within this thesis. The wide range of research works are topically organized as follows:

- nonlinear system control work within [14] (asymptotic stabilization);
- DDV modelling and control work within [5] (local stabilizability of non-holonomic systems), [85] (the classic parking problem involved with under-actuated systems and non-smooth stabilization issues), [31] (Lie bracket based controllability for DDV's), [22] (dynamic modelling of a DDV using Newton-Euler and Lagrangian Methodologies), [4], [69] (input-output coupling effects in dynamic modelling of DDV and SISO controller design);
- modelling and control of longitudinal platoon of non-identical vehicles [71], [72];
- trajectory tracking of single and multiple DDV's [36], [37], [38], [68] (nonlinear outer-loop control design and stability robustness issues);
- formation control strategies for DDV's [19], [30] (leader follower approach, lyapunov based nonlinear controller design);
- formation control of DDV's using receding horizon optimization approach [19], [35], [16], [53];

The following paragraphs are intended to provide a brief overview of the various technical details that would be considered throughout this thesis.

- **DDV Modelling.** A differential drive/deferentially steered/deferentially wheeled robot is a mobile robot that has two rear wheels that are capable of rotating independent of each other i.e., each wheel is attached to a separate actuator. Since the wheels can rotate independently of one another, there is no requirement for any sort of additional steering mechanism. This makes it a widely used platform in both academic and commercial robotic applications. Depending on the actuators used to control the wheel speed, the inputs to the DDV vary, for example, if the actuators consist of armature controlled Direct Current (DC) motors the input signal would be the voltage supplied to these motors. The sum of voltages contributes to the linear velocity v and the difference of voltages contributes to the angular velocity ω of the vehicle. Other commonly used actuators include stepper motors and brushless DC motors. In Chapter 2, detailed step by step instructions for the construction of a DDV is provided.

  - *Kinematic Model.* [27], [36] present the kinematic model (ignoring the effect of forces/torques acting on the system) of a DDV. The kinematic model defines the relation between the inputs v, ω [3] and the pose of DDV (x, y, θ). This model assumes that any form of linear and angular velocities (v, ω) can be attained instantaneously by the DDV. This, of course, is not a realistic assumption because from Newton's second law of motion we know that achieving instantaneous velocity would require infinite acceleration. Admittedly, this model is the most simple representation of a differential drive robot and is widely used in several simulators, e.g. in MATLAB, Gazebo, etc. Nevertheless, it should be noted that in real-world conditions it is impossible to generate the (v, ω) instantaneously due to the actuator

[3] v refers to linear velocity and ω refers to the angular velocity, this is notation will be followed throughout this thesis

limitations and mass-inertia effects. Therefore, it is important to model the dynamics of a DDV including that of actuators in order to truly understand the behavior of the system.

– *Dynamical Model.* As mentioned earlier, the dynamic model considers the impact of various forces acting on a system and the actuators responsible for generating the forces. The dynamic model of the DDV ($(e_{a_r}, e_{a_l}) \rightarrow (v, \omega)$) can be divided into two parts: 1) dynamics of the vehicle excluding the actuators [60], [12] - from input actuator torque to output linear and angular velocities $(\tau_r, \tau_l) \rightarrow (v, \omega)$, 2) dynamics of the actuator - from input voltages to output actuator torques $(e_{a_r}, e_{a_l}) \rightarrow (\tau_r, \tau_l)$. The input to the actuator dynamics vary depending on the actuator considered, in this thesis an armature controlled DC motor is being used. In [22], the authors have presented a two-input two-output (TITO) nonlinear time invariant model of the DDV - including the DC motor dynamics as well as the mass-inertia effects of the vehicle. This nonlinear model can be linearized to obtain a fourth-order TITO linear time invariant (LTI) model. This TITO LTI model has been exploited within [46], [4], [45] for control design, and also as the basis for all the studies presented within this thesis. Additionally, in this thesis, we consider three different input/output variations of the TITO LTI model: Model 1 :- $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$, Model 2 :- $(e_{a_r}, e_{a_l}) \rightarrow (v, \omega)$, Model 3 :- $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$. The first dynamical model representation is widely used in most of the literature since its easy to obtain a reliable and accurate measurement of wheel angular velocities using less sophisticated sensors such as encoders, but nowadays, with the development of powerful and cost-effective microcontrollers and sensors such as IMUs, LIDARs, stereo cameras it has become possible to measure the linear and angular velocities in a reliable and accurate manner. Moreover, model 1 is decoupled at low frequencies (this is not true for model 2) i.e. frequencies below $\frac{\beta}{I_w}$, where β denotes the motor shaft angular velocity damping constant and I_w denotes the rotational moment of inertial, thereby facilitating the use of a simple PI-based controller. In regard to model 2 - $(e_{a_r}, e_{a_l}) \rightarrow (v, \omega)$, the map from input voltages to linear and angular velocities remains coupled at all frequencies, which would require the use of MIMO control design ideas. In [46], the authors presented the idea of employing a decentralized PI controller which is originally designed for a $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ system in order to control the $P_{[e_{a_r}, e_{a_l}] \rightarrow [v, \omega]}$ system. They have provided mathematical proof stating that such a controller design would indeed be feasible however, it comes at a cost of increased uncertainty in controller effort that can lead to controller saturation. In order to overcome the limitations of model 2 we have come up with model 3 - $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ in which the map from sum and difference of input voltages to the linear and angular velocity of the DDV remains decoupled [4] at all frequencies, thereby facilitating the use of a simple PI-based controller. Furthermore, an in-depth analysis is presented in Chapter 4 to understand and quantify the impact of critical design parameters on the static and dynamic properties of model 1 and model 3 (e.g., mass, moment of inertia, center of gravity, radius of the wheel, and operation point). This analysis becomes crucial in corroborating the results presented in Chapters 5, 6.

– *Non-Holonomic DDV Controllability.* The non-holonomic restrictions/-

[4] absolute decoupling in case of both model 1 - $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and model 3 - $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ is achieved only when $d = 0$ i.e. the center of gravity coincides with the midpoint of the axis joining the two wheels

constraints are introduced by the underactuated nature of the DDV i.e. the system has two independent control inputs (left/right motor voltages (e_{a_r}, e_{a_l})) which are less than the three degrees of freedom (x, y, θ) that are meant to be controlled. A system is said to be controllable if there exists an input function $u(t)$ that can transfer the state of the system from any initial state $x_i(t)$ to any final state $x_f(t)$ within a finite amount of time. And according to Brockett's Theorem [14], it is impossible to stabilize a non-holonomic system using a continuous time-invariant feedback law. Furthermore, [5] exploits the work of Brockett to show that a classic position stabilization objective $(x_{ref}, y_{ref}, \theta_{ref})$ cannot be attained with a single continuous control law i.e., in order to park a vehicle at the desired position, one has to switch control laws or the use of a discontinuous time-invariant/time-varying/non-smooth control law is essential.

An underlying consequence of the above is that the linearized kinematic model of the DDV is uncontrollable [14] - this might seem obvious since the DDV cannot move sideways or in the lateral direction. In spite of this, from a nonlinear geometric (Lie-bracket) point of view [31]; i.e. the nonlinear kinematic model of the DDV is controllable. This confirms our common real-world experience that a non-holonomic DDV can be moved from any initial state (x_i, y_i, θ_i) to the final state (x_f, y_f, θ_f) i.e the vehicle can be parked in any location. Thus, it can be said that a DDV is locally (linearly) uncontrollable while it is globally (nonlinearly) controllable. In Section 4.5.1 of [46] a more thorough mathematical review of the above ideas has been presented.

- **Classical Controls.** The text [66] addresses the classical control design fundamentals. Internal model principal concepts that are critical for reference command following, and input output disturbance attenuation are addressed within [28], [66]. The general proportional plus integral plus derivative (PID) control design, analysis and tuning concepts are presented in the text [7]. Fundamental performance limitations are addressed in [66], [79].

- **Multivariable Control.** Detailed discussion on multivariable system analysis and control system design are presented in [67], [64], [63], [52], [78], [29].

- **Relevant Nonlinear Control.** Fundamental theory addressing the existence of continuous stabilizing control laws for nonlinear systems was first presented within this novel work [14]. This work has been exploited within [5] and [85] to address the classical parking problem (position control) for DDV's. A nonlinear control law for position control while utilizing the Lyapunov ideas to guarantee asymptotic stabilization of the system is presented in [36].

- **DDV Inner-Loop Control.** In [9] and [75], a PID based inner-loop control design has been presented; within [26], [4] a PI-based inner-loop control design has been discussed. In this thesis, within Chapter 4, we have presented PI-based inner-loop control laws for $P_{[e_{a_r},e_{a_l}]\to[\omega_r,\omega_l]}$ and $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\to[v,\omega]}$ plants respectively. Specifically, the inner-loop control law design methodology and parameter trade studies presented in [46] are utilized to design the inner-loop control law presented within this thesis. Though we have presented a decentralized inner-loop control law, a centralized design becomes necessary under specific DDV design configurations. A detailed review of the conditions required in order to switch from a decentralized to centralized inner-loop control law is presented within Chapter 5.

- **DDV Outer-Loop Control.** One of the primary objectives of this thesis is to understand and quantify the impact of design parameters on the performance of outer-loop control laws. To do this, we have exploited the existing literature on speed and position-direction control and have conducted extensive simulation and hardware trails to collect the required data. Before we proceed further, it is important to highlight the difference between trajectory tracking and path following. In some of the works that are currently available we often find these terms being used interchangeably. Trajectory tracking refers to following $x(t), y(t)$ commands i.e., (x, y) commands with very specific temporal constraints, whereas path following refers to following a curve/path without strict temporal constraints [2]. Standard linear control laws are designed for trajectory tracking and path following within [21]; nonlinear techniques such as feedback linearization and Lyapunov-based controller design are presented within [36], [61], [23] and [25].

  - *Cruise Control.* Cruise control is one of the most simple and widely used feature in robotic research platforms and commercial on-road vehicles. The cruise control law is designed based on the plant and inner-loop PI control laws presented in Chapters 3 and 4. As mentioned earlier, we have designed decentralized inner-loop speed control laws for two different input/output variations of the plant i.e. $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ and $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \rightarrow [v, \omega]}$. The inputs to each of these inner-loop systems is (v_{ref}, ω_{ref}) and output is (v, ω). The map from reference commands (v_{ref}, ω_{ref}) to actual velocities (v, ω) can be approximated as a simple first-order decoupled system (e.g. $diag(\frac{a_1}{s+a_1}, \frac{a_2}{s+a_2})$). This is a consequence of the well designed inner-loop control system. Therefore the output θ controller just sees a simple first order system $\frac{a_2}{s+a_2}$ from $\omega_{ref} \rightarrow \omega$, and from classical root locus ideas [66], a simple proportional (P) controller will be sufficient to track the reference commands θ_{ref}. However, if the gain is too large, oscillations (or even limit cycle behaviour) are expected in θ and in that case, a simple PD controller with roll off can be designed to resolve the issue [46].

  - *Planar Cartesian Stabilization.* In [85], the authors have presented the design of linear control laws to stabilize the posture of the vehicle at a desired position (x_{ref}, y_{ref}) and orientation θ_{ref} in the two dimensional space. This can also be referred to as the classic parking problem. A subcategory of this problem is the planar Cartesian stabilization problem which refers to moving the vehicle from an initial (x, y) coordinate to the target (x_{ref}, y_{ref}) coordinate. Within [85], the authors have presented a linear control law that utilizes the error between the vehicle current pose (x, y, θ_{ref}) and the target pose $(x_{ref}, y_{ref}, \theta_{ref})$ in order to get arbitrarily ϵ-close to the desired position (x_{ref}, y_{ref}). This work forms the foundation for the linear control laws for planar Cartesian stabilization presented in Chapter 5.

- **Optimal Control.** According to Brockett's theorem [14], it is impossible to stabilize a non-holonomic DDV at a given posture $(x_{ref}, y_{ref}, \theta_{ref})$ using a smooth, time-invariant and static state feedback control law. In order to solve this trajectory tracking problem, several control algorithms have been developed [34]. Out of the several approaches that were produced, feedback linearization and receding horizon optimal control have gained a lot of prominences. In feedback linearization, an algebraic transformation is applied to the nonlinear dynamics of the system in order to obtain the equivalent linear dynamics - so that linear control laws can be applied directly to the nonlinear system. Within [56], [17] and [74], the authors have presented different methods for solving the trajectory tracking problem using feedback linearization. Although feedback linearization is a promising approach to solve the trajectory tracking problem it is not possible to impose state constraints that fundamentally exist in real-world scenarios. These constraints can be the result of nonlinearities imposed due to actuators (such as bandwidth limitation, actuator saturation, high-frequency noise, etc.) or due to dynamic obstacles present in the path. Whereas, optimal control based approaches facilitate incorporating these constraints into the optimization problem in a systematic manner. Therefore, optimal control based approaches have been very successful and are widely being employed for real-world robotics applications such trajectory tracking [40], lane changing [89], obstacle avoidance [3], multi-robot formation control [42], etc.

In optimization-based approaches, the error dynamics of the trajectory tracking problem are first computed. Using the error dynamics and model of the system as constraints, an optimization algorithm is employed to generate a sequence of control inputs that can minimize the trajectory tracking error (cost function) over a finite time horizon while subject to various system and control constraints. Depending on the nature of the constraints, the optimization problem can be categorized as linear and nonlinear optimization. One of the major disadvantages of using nonlinear optimization methods over linear methods is the extensive computational burden involved in solving the non-convex optimization problem on-line. A linear optimization involves finding a global solution by solving a convex optimization problem. Therefore, linear optimization approaches are preferred over nonlinear approaches especially if the system at hand involves faster dynamics and has limited computational power. However, the majority of the linear optimization approaches found in the literature assume high inner-loop bandwidth and therefore just consider only the kinematic model while formulating the optimization dynamics. This is fundamentally incorrect because assuming a high inner-loop bandwidth means that actuators can instantaneously generate the angular speed corresponding to the reference commands/input voltages - which is not practically possible in the real-world. Every actuator has a bandwidth limitation and output saturation that are not represented by the kinematic model. Therefore, considering the dynamic model of the system along with the kinematic model of the system during the optimization process is critical for implementing trajectory tracking in the real-world.

- **Multi-Robot Formation Control.** The area of multi-robot coordination has received a great deal of attention over the past decade. As stated within [47], a group of mobile robots can exhibit a high level of robustness and fault-tolerant properties under highly efficient principles. A group of robots can perform several tasks that may be impossible for a single robot; some examples include large area exploration [15], surveillance [82], object transportation [88], [11], construction [80], etc. Multi-robot formation control is one of the key aspects of multi-robot coordination and has received much attention in recent years. There are several formation control strategies in literature and some of the widely used ones are behavior methods, leader-follower methods, and virtual structure methods. In behaviour based methods, the main objective is divided into several low-level tasks that are to be performed by individual robots in order to achieve the group behaviour [8], [58], [41], [48], and [11]. In the virtual structure approach, the entire formation is treated as a single rigid entity, and the desired trajectory is assigned to the virtual structure which traces it down to the trajectories that each individual member in the virtual structure should follow [43], [24] and [65]. In the leader-follower approach, one of the robots is designated as a leader and the rest are considered as followers. The objective of the followers is to track the leader robot while maintaining a fixed distance and orientation with respect to the leader robot. Once the motion of the leader is given, each follower employs a local control law to track the leader, and thus the desired formation of the system is achieved [20], [18], [83], [81] and [33].

A significant advantage of the leader-follower approach when compared to others is that conventional single robot trajectory tracking algorithms can be directly applied to design the local control laws for the follower [44]. In this thesis (Chapter 6), we consider the leader-follower approach for multi-robot trajectory tracking and each follower employs the receding horizon optimization scheme. The pose (x, y, θ) information of the leader along with the required relative distance and orientation information $(\Delta d, \Delta\theta)$ with respect to leader robot are utilized to generate the reference commands $(x_{ref}, y_{ref}, \theta_{ref})$ for the follower robot. These reference commands will serve as the input to the optimization-based trajectory tracking approach mentioned earlier. One of the key objectives of this thesis is to quantify the performance variations that occur due to the incorporation of dynamic and kinematic constraints into the optimization problem when compared to just using the kinematic constraints. We have conducted extensive simulation and hardware trails to collect the required data and the results have been summarized in Chapter 6.

1.3 Contributions - Fundamental Questions Addressed

The following questions have been addressed within this thesis

- **How to build a high-speed DDV for research? How to design an open-source platform capable of handling multi-robot research?** In order to build a DDV it is important to have a clear understanding of the design requirements. These design requirements are dictated by the trade studies that are intended to be performed and also by the vehicle dynamics - especially the limitations of the dynamic model. A brief description of the design requirements followed by detailed instructions for building the DDV has been presented in Chapter 2. Another crucial part of building a DDV is actuator selection. In Chapter 2, we have presented guidelines for selecting an actuator based on the maximum velocity and minimum bandwidth requirements along with a thorough market analysis of existing actuators and their characteristics.

  Furthermore, to carry out high-speed multi-robot research an open-source platform has been developed using open-source software and hardware tools. The major hardware and software components of this platform are: 1) Ubuntu 16.04 Operating System 2) Robotics Operating System 3) FKIE Multimaster Package 3) HTC Vive Virtual Reality System 4) NVIDIA Jetson TX2 Module. A detailed overview of the system-level architecture involved in addressing the global/local command, control, computing, communications (C^4), and sensing (S) requirements of the fleet of DDV's, followed by a brief description of individual software nodes/components is presented in Chapter 2.

- **What critical design parameters (e.g., mass, moment of inertia, center of gravity, radius of the wheel, operation points, etc.) impact the dynamic characteristics of the DDV?** In this thesis, a systematic approach is taken to understand and quantify the impact of these design parameters on the performance of DDV. In Chapter 4, we have presented the kinematic model of the vehicle along with the non-holonomic constraints and its limitations. Furthermore, a nonlinear dynamical model of the DDV including the actuator dynamics is presented. Using this dynamical model, detailed frequency domain trade studies are presented to understand the impact of critical design parameters on the input-output coupling, stability, and bandwidth properties of the DDV. These critical design parameters include mass (m), radius of the wheel (r), moment of inertia (I), center of gravity (d), equilibrium linear and angular velocity (v_{eq}, ω_{eq}).

In addition to the trade studies, we have presented the dynamic decoupling moment of inertia condition ($I_{decoupling} = m(\frac{d_w}{2})^2$) and aspect ratio condition ($AR_{decoupling} = \frac{l}{d_w} \approx \sqrt{2}$). Here, l represents the length and d_w represents the width of the DDV. The implication of these $I_{decoupling}$ and $AR_{decoupling}$ conditions is that if a DDV is designed adhering to either of these conditions, then the inputs and outputs of the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ model will become fundamentally decoupled allowing for a simple decentralized (SISO) control design. To our knowledge, these dynamic decoupling conditions are not examined in the literature and the $AR_{decoupling}$ condition was first introduced in the thesis work of Anvari [4], who was a former member of this lab.

- **What is the optimal way to model a DDV and how does it impact the controller design?** In this thesis, we consider three different input/output variations of the TITO LTI model: 1) $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$. 2) $(e_{a_r}, e_{a_l}) \rightarrow (v, \omega)$. 3) $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$. The $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ dynamical model representation is widely used in most of the literature since its easy to obtain a reliable and accurate measurement of wheel angular velocities using less sophisticated sensors such as encoders, but nowadays, with the development of powerful and cost-effective microcontrollers and sensors such as IMUs, LIDARs, stereo cameras it has become possible to measure the linear and angular velocities in a reliable and accurate manner. Moreover, $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ model is decoupled at low frequencies (this is not true for $(e_{a_r}, e_{a_l}) \rightarrow (v, \omega)$ system) i.e. frequencies below $\frac{\beta}{I_w}$, where β denotes the motor shaft angular velocity damping constant and I_w denotes the rotational moment of inertial, thereby facilitating the use of a simple PI-based controller. In regard to the $(e_{a_r}, e_{a_l}) \rightarrow (v, \omega)$ model, the map from input voltages to linear and angular velocities remains coupled at all frequencies, which would require the use of MIMO control design ideas. In order to overcome this, we have come up with $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ model representation in which the map from sum and difference of input voltages to the linear and angular velocity of the DDV remains completely decoupled [5] at all frequencies, thereby facilitating the use of a decentralized PI-based controller. Furthermore, an in-depth analysis is presented in Chapter 4 to understand the impact of critical design parameters on the dynamic properties of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ and $P_{[e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}] \rightarrow [v, \omega]}$ systems.

- **How do these critical design parameters impact the speed and position-direction control design?** In Chapter 4, detailed trade studies have been presented that show the impact of variation in design parameters on the stability, bandwidth, and input-output coupling of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ and $P_{[e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}] \rightarrow [v, \omega]}$ models. To further understand and quantify the impact of these design parameters on the speed and position-direction control, eight different DDV designs have been considered that are formed by varying the moment of inertia, center of gravity location, and input-output modeling. By designing and implementing

[5] absolute decoupling in case of both $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ models is achieved only when $d = 0$ i.e. the center of gravity coincides with the midpoint of the axis joining the two wheels

outer-loop cruise control and planar Cartesian stabilization algorithms, we have compared and quantified the performance of each of these eight DDV designs.

In Chapter 5, a systematic approach has been presented to perform these outer-loop speed (cruise control) and position-direction (planar Cartesian stabilization) control performance trade studies, and the corresponding simulation and hardware results are discussed in detail. An overview of some of the key hardware results [6] is as follows:

- *Cruise Control.* From varying radius of curvature of the trajectory (R) for fixed tracking velocity ($v_{ref} = 1m/s$), it was observed that systems with higher moment of inertia exhibit a steep increase in errors ($||v_e||_\infty, ||\theta_e||_\infty$) and control effort for a decrease in $R \leq 0.75$ m. For $R > 0.75$ m it was observed that designs with higher coupling between the inputs and outputs at lower frequencies tend to exhibit higher errors and control effort. Similarly, by varying the trajectory tracking velocity while maintaining a fixed radius of curvature ($R = 1.5$ m), it was observed that for $v_{ref} \leq 1.7$ m/s, designs with higher coupling between the inputs and outputs at lower frequencies tend to exhibit higher errors and control effort when compared to other systems.
- *Planar Cartesian Stabilization.* From varying trajectory tracking velocity (v_{ref}) for fixed radius of curvature of trajectory ($R = 1.5$ m), it was observed that systems with higher moment of inertia exhibit a steep increase in the tracking errors ($||x_e||_\infty, ||y_e||_\infty$) and control effort with an increase in $v_{ref} \geq 1.8$ m/s. For $v_{ref} < 1.8$ m/s, it was observed that designs with higher coupling between the inputs and outputs at lower frequencies tend

[6] for detailed information regarding the performance metrics and DDV design configurations please look into Chapter 5

to exhibit higher tracking errors and control effort. Similarly, by varying the radius of curvature of the trajectory while maintaining a fixed tracking velocity ($v_{ref} = 1$ m/s), it was observed that for $R \geq 1.25$ m, systems with higher coupling between the inputs and outputs at lower frequencies tend to exhibit higher errors and control effort when compared to other systems.

- **Does a less stable system possess any advantage when compared to a more stable system?** For high-speed trajectory tracking that requires sharp/aggressive maneuvers it is advantageous to have dynamically coupled or less stable systems. This is due to the fact that they are inherently unbalanced and therefore do not require additional control effort to perform these aggressive maneuvers [54]. This intuitive understanding has been bolstered by the speed and position-direction control performance trade studies presented in Chapter 4. In case of cruise control, it was observed that the systems with more stable plants tend to exhibit higher tracking errors ($||v_e||_\infty, ||\theta_e||_\infty$) and control effort when compared to the less stable systems, with an increase in the tracking velocity $v_{ref} \geq 1.7$ m/s at a constant radius of curvature of the trajectory $R = 1.5$ m. Similarly, in the case of planar Cartesian stabilization, it was observed that the systems with more stable plants tend to exhibit higher tracking errors ($||x_e||_\infty, ||y_e||_\infty$) and control effort when compared to the less stable systems, with a decrease in radius of curvature of the trajectory $R \leq 1.25$ m at a constant tracking velocity $v_{ref} = 1$ m/s.

- **When is a SISO controller sufficient? and when is MIMO controller necessary?** This question can be answered at different levels based on the design of the DDV. In Chapter 3, it is shown that $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$

model of a DDV is fully decoupled at all frequencies when $d = 0$. So in this case, a SISO controller design will suffice. In the case of a $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ system, it has been shown that when either of the dynamic decoupling conditions are met i.e., $I_{decoupling} = m(\frac{d_w}{2})^2$ or $AR_{decoupling} = \frac{l}{d_w} \approx \sqrt{2}$, the model becomes fully decoupled and therefore a SISO control design is sufficient. However, for $d = 0$, $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ system is shown to exhibit very little coupling between the inputs and outputs at dc, which gradually increases with the frequency of operation. In this particular case, from the results presented in [4], a SISO controller would be sufficient as long as the operating bandwidth is significantly lower compared to the peak coupling frequency. A MIMO controller becomes necessary only in case of high performance/aggressive control objectives in which the bandwidth of the control loop is sufficiently close to the peak coupling frequency. A more concrete answer to this question is provided via the outer-loop cruise control and planar Cartesian stabilization performance trade studies presented in Chapter 4. An overview of those results is as follows:

– *Cruise Control.* From the trade studies performed by varying tracking velocity (v_{ref}) for a fixed radius of curvature ($R = 1.5$ m), it was observed that for $v_{ref} \leq 1.8$ m/s and $R \geq 1.5$ m, a SISO controller is sufficient to provide good [7] trajectory tracking performance for a system with input-output coupling. However, for $v_{ref} < 1.85$ m/s, a MIMO controller is necessary to achieve a similar performance. Similarly, from the trade studies performed by varying radius of curvature of the trajectory for a fixed tracking velocity ($v_{ref} = 1$ m/s), it was observed that for $R \geq 1$ m and $v_{ref} \geq 1$

[7]good trajectory tracking performance is a relative measure that depends on the control objectives/application requirements, please refer to the trade studies presented in Chapter 5 to get a detailed view of the performance metrics that are considered

m/s, a SISO controller is sufficient to provide good trajectory tracking performance for a system with input-output coupling. However, for $R < 1$ m, a MIMO controller is necessary to achieve similar performance.

  - *Planar Cartesian Stabilization.* From the trade studies performed by varying tracking velocity (v_{ref}) for a fixed radius of curvature ($R = 1.5$ m), it was observed that for $v_{ref} \leq 1.35$ m/s and $R \geq 1.5$ m, a SISO controller is sufficient to provide good trajectory tracking performance for a system with input-output coupling. However, for $v_{ref} > 1.35$ m/s, a MIMO controller is necessary to achieve a similar performance. Similarly, from the trade studies performed by varying radius of curvature of the trajectory for a fixed tracking velocity ($v_{ref} = 1$ m/s), it was observed that for $R \geq 1.8$ m and $v_{ref} \geq 1$ m/s, a SISO controller is sufficient to provide good trajectory tracking performance for a system with input-output coupling. However, for $R < 1.8$ m, a MIMO controller is necessary to achieve a similar performance.

- **When a kinematic model sufficient for design and evaluation? When is a dynamic model essential for design/evaluation?** To answer this question, we consider a hierarchical inner-outer loop architecture with a PI-based wheel angular velocity (ω_r, ω_l) controller in the inner-loop and a receding horizon optimization-based outer-loop controller. Detailed discussion on why an optimization-based approach is chosen for trajectory tracking (x, y, θ) is presented in Section 6.1. In most of the literature available, the trajectory tracking optimization problem is formulated based on only the kinematic model of the DDV, and the dynamics of the system are completely ignored. In these approaches, the inner-loop speed control system is assumed to offer perfect tracking i.e. infinite bandwidth. This is clearly not the case with real-world systems because every actuator or a real-world system will have limitations, and therefore it's incorrect to consider perfect inner-loop tracking because an actuator will never produce instantaneous speeds for a given input voltage. Hence, it is necessary to include the constraints imposed by the dynamical model in addition to those of the kinematic model in order to improve the performance of the trajectory tracking controller. To further provide a quantitative answer to the above question, we have taken a systematic approach in performing the simulation and hardware trade studies and the results obtained are discussed extensively in Chapter 6. A brief summary of those results is as follows:

- Based on the trade studies performed at constant trajectory tracking velocity (v_{ref}) and radius of curvature of trajectory (R) while varying the inner-loop bandwidth (B_i), it is observed that for $B_i \geq 7.5$ rad/sec, a kinematic model-based optimization approach is sufficient to provide good [8] trajectory tracking properties, given that $v_{ref} \leq 1$ m/s, and $R \geq 1.5$ m. However, for $B_i < 7.5$ rad/sec, a combined kinematic and dynamic model-based optimization approach is necessary to achieve similar performance.
- Based on the trade studies performed at constant radius of curvature (R) and inner-loop bandwidth (B_i) while varying the trajectory tracking velocity (v_{ref}), it is observed that for $v_{ref} \leq 1.6$ m/s, a kinematic model based optimization approach is sufficient to provide good trajectory tracking properties, given that $B_i \geq 10$ rad/sec, and $R \geq 1.5$ m. However, for $v_{ref} > 1.6$ m/s, a combined kinematic and dynamic model based optimiza-

[8] good trajectory tracking performance is a relative measure that depends on the control objectives/application requirements, please refer to the trade studies presented in Chapter 6 to get a detailed view of the performance metrics that are considered

tion approach is necessary to achieve a similar performance.

- Based on the trade studies performed at constant trajectory tracking velocity (v_{ref}) and inner-loop bandwidth (B_i) while varying the radius of curvature of trajectory (R), it is observed that for $R \geq 1.6$ m, a kinematic model-based optimization approach is sufficient to provide good trajectory tracking properties, given that $v_{ref} \leq 1$ m/s, and $B_i \geq 10$ rad/sec. However, for $R < 1.6$ m, a combined kinematic and dynamic model based optimization approach is necessary to achieve a similar performance.

1.4 Organization of Thesis

The remainder of the thesis is organized as follows:

- Chapter 2 (page 24) presents the detailed design process involved in the development of the DDV followed by a brief discussion on the command, control, communications and sensing (C^4S) requirements and software architecture of the open-source platform that is developed in order to conduct multi-robot research.

- Chapter 3 (page 55) describes the mathematical concepts that are frequently used throughout this thesis.

- Chapter 4 (page 65) presents the dynamic model of a DDV and a thorough discussion on the three different input-output representations of the model followed by extensive design trade studies.

- Chapter 5 (page 113) presents the eight DDV designs and the corresponding inner and outer-loop control laws, followed by a detailed discussion on the trade studies conducted.

- Chapter 6 (page 164) describes a multi-robot formation control strategy based on receding horizon optimization and the corresponding simulation and hardware trade studies.
- Chapter 7 (page 191) presents the summary and future research directions.

Chapter 2

OVERVIEW OF THE DIFFERENTIAL DRIVE VEHICLE PLATFORM

2.1 Introduction and Overview

This chapter consists of two parts: First, we present a brief overview of the design requirements that were considered followed by a step by step procedure involved in the development of the DDV; Second, a detailed description of the global and local command, control, communications, computing (C^4), and sensing (S) capabilities of the DDV platform are mentioned followed by a brief overview of the software framework.

2.2 Hardware Design Requirements

The typical design parameters of a DDV are its length, width, height, total mass, center of gravity location, moment of inertia [1] and actuator characteristics. It is required that the DDV design should be capable of having adjustable total mass, moment of inertia, and center of gravity in order to perform the various trade studies. Chapter 5 provides a detailed description of the various trade studies that were considered in this thesis. In addition, the dynamic model of the DDV presented in this thesis is based on the two-dimensional approximation of the actual vehicle i.e. the height of the center of gravity is not considered while modeling the dynamics of the vehicle [2] . This means that the current DDV dynamical model does not provide

[1] moment of inertia refers to the total moment of inertia (I) of DDV

[2] Albeit this two-dimensional vehicle dynamical model is adapted in most of the existing literature, this does not mean that the height of the vehicle has negligible impact on dynamics. Incorporating the height of the vehicle into the dynamical model is an active topic and will be considered for future research

us with any insight into choosing the height of the vehicle, so what should be the ideal height of the DDV? A simple thought experiment would help up understand that the taller the vehicle is, the easier it is to topple during aggressive maneuvers. Therefore, in order for the two-dimensional model to be valid, especially during high-speed maneuvers, the center of gravity should be maintained as close to the ground as possible.

Generally, variation in the moment of inertia can be achieved by changing the placement of various components on the vehicle while holding the length, width, and center of gravity of the vehicle to be constant. Furthermore, variation in mass can be achieved by adjusting(add/remove) the mass directly at the center of gravity, this would ensure no change in the center of gravity. Besides, adjusting the mass will impact the moment of inertia of the vehicle unless the center of gravity (the point where the mass is added) coincides with the wheel axis ($d = 0$). Similarly, variations in the center of gravity will cause a variation in the moment of inertia of the vehicle, and this variation can be achieved either by changing the placement of various components on the vehicle or by adding additional mass at different locations. In this thesis, the first option is considered because altering the mass will impact the actuator performance and will therefore bias the variations due to the center of gravity and moment of inertia.

In Appendix A, a MATLAB program for calculating the moment of inertia as a function of the center of gravity, length, width, and mass of the vehicle is given. Additionally, this function also gives the minimum and maximum values of the moment of inertia and center of gravity that can be obtained by varying the positions of the camera, Lithium Polymer battery, and the Li-on battery.

2.3 Actuator Selection

A DDV can be equipped with different kinds of motors, e.g. DC motors, BLDC motors, stepper motors[73], etc. In general, DC motors are mostly preferred for low speed and high torque applications and have a very simple speed/torque control setup. Whereas, BLDC motors are widely used for high-speed high torque applications such as quadcopters, electronic skateboards, etc. And also, BLDC motors have become more prominent nowadays due to the development of Li-ion battery technologies. Thereby it is advisable to use BLDC motors when the overall size and weight of the vehicle is a concern. In this thesis, we would be using an armature controlled DC motor to perform various trade studies. In the future, these trade studies can be replicated for BLDC motors as well.

The following steps demonstrate the process involved in selecting an actuator: First, let us introduce the parameters "m" (mass of the fully-loaded vehicle),"m_c" (mass of the vehicle without wheels and motors), "m_w" (mass of the wheels and motors),"R" (radius of the wheels). Given that m_c = 4.17 Kg, $m = m$, $R = 0.039$ m, we have to select the actuators that can produce a maximum velocity = 3.0 m/s; minimum settling time = 0.3 s.

Second, the required speed of the motor i.e rated speed = 735 RPM (calculated based on the maximum required velocity). Using Newtons equations of motion, we can calculate the minimum required acceleration(a_{min}) and torque(τ_{min}) to ensure the velocity and rise time requirements are satisfied.

$$a_{min} = 10.0 m/s^2, \ \tau_{min} \geq (m_c + m_w)\frac{R\, a_{min}}{2} \tag{2.1}$$

From equation (2.1), it can be observed that τ_{min} is a function of mass of the motors (m_w). Usually, the total mass of the motor drivers and batteries remains constant irrespective of the motor selected and thereby can be included in m_c, but, if the selected motors require specific motor drivers or batteries, equation (2.1) should be modified to reflect this.

To sum it up, the motors to be selected should meet the following specifications

$$\text{Rated Speed} > \ 735\ RPM, \ \ \text{Rated Torque} > \ \tau_{min}$$

The motor drivers and batteries should be chosen according to the stall, continuous, and peak current requirements of the selected motors.

<u>Note:</u>

1. Product websites generally mention the stall torque and no-load speed, and this is often confused with rated speed and rated torque.

2. The calculations mentioned above are valid under the assumption that there is no loss due to friction (static/dynamic). Therefore, it is always advisable to select the motors that are beyond the above specifications by 10 to 15 percent.

A detailed market analysis has been performed in order to understand the characteristics of commercial motors available. We have considered 22 various motor models from different manufactures - complete information including the web-page information of these 22 motor models is presented in Appendix A. Figure 2.1 shows the characteristics of these 22 motor models. Figures 2.2,2.3 compare the speed, torque, input and output power characteristics of these motors. In Figure 2.2, models 3,4 marked in red have the required rated speed and rated torque characteristics, and the speed vs torque plot in Figure 2.3 gives a better understanding of their performance. In Figure 2.3, only motor models 3,4,11,22 are highlighted since they have significantly better performance curves compared to other models and also in order to reduce the clumsiness of the plots.

Finally, based on the rated speed and rated torque requirements, motor model 3 has been chosen for this thesis. Figure 2.4 shows the characteristics of this motor. Please note that models 3,4 both have the same performance characteristics.

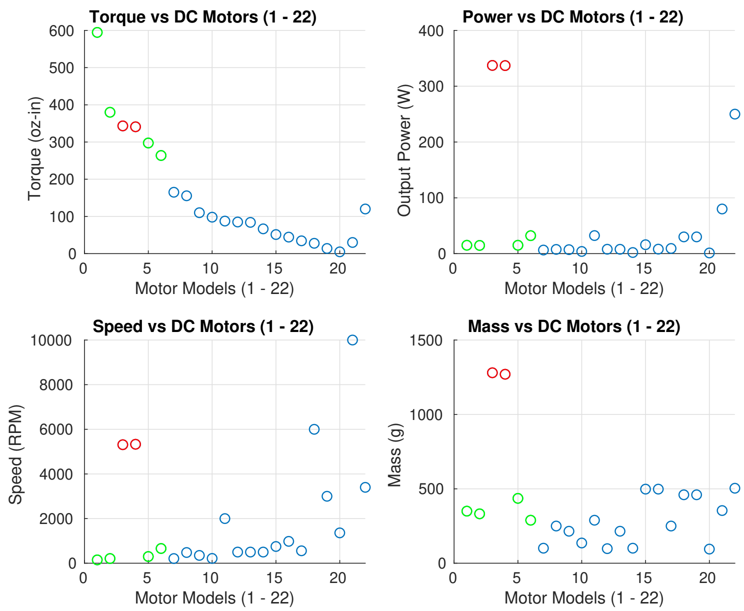


Figure 2.1: DC Motor(Models 1 - 22) Characteristics

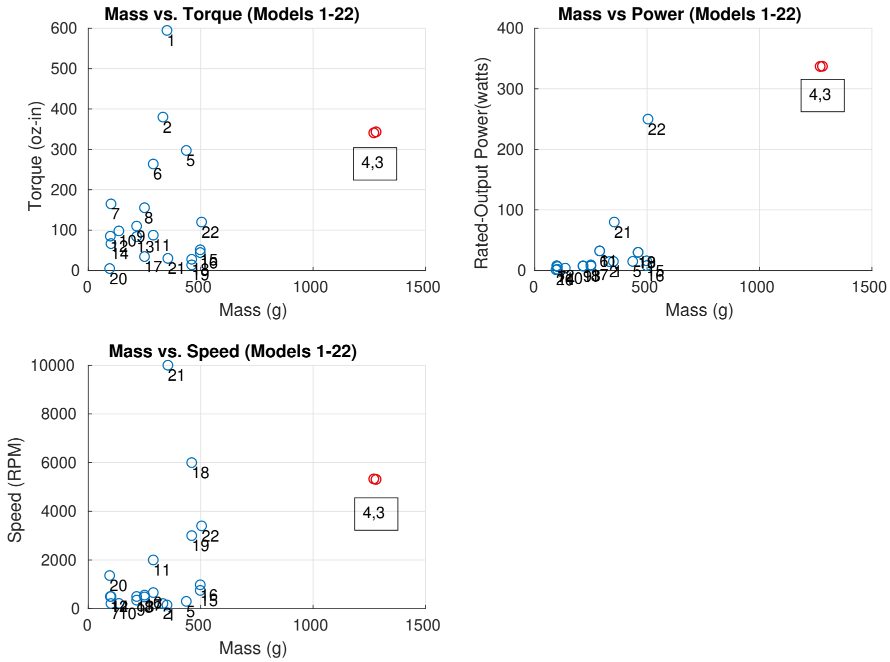


Figure 2.2: DC Motor(Models 1 - 22) Comparison

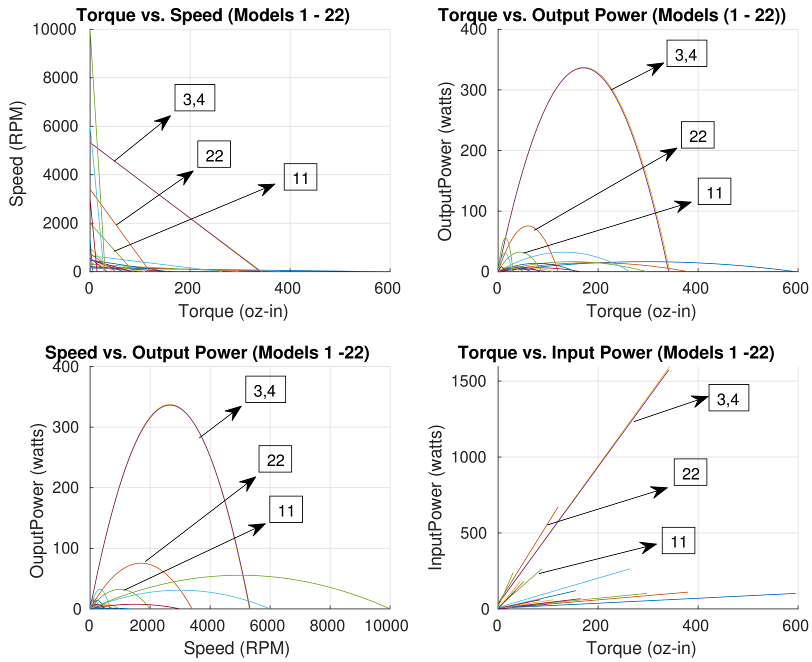


Figure 2.3: DC Motor(Models 1 - 22) Comparison

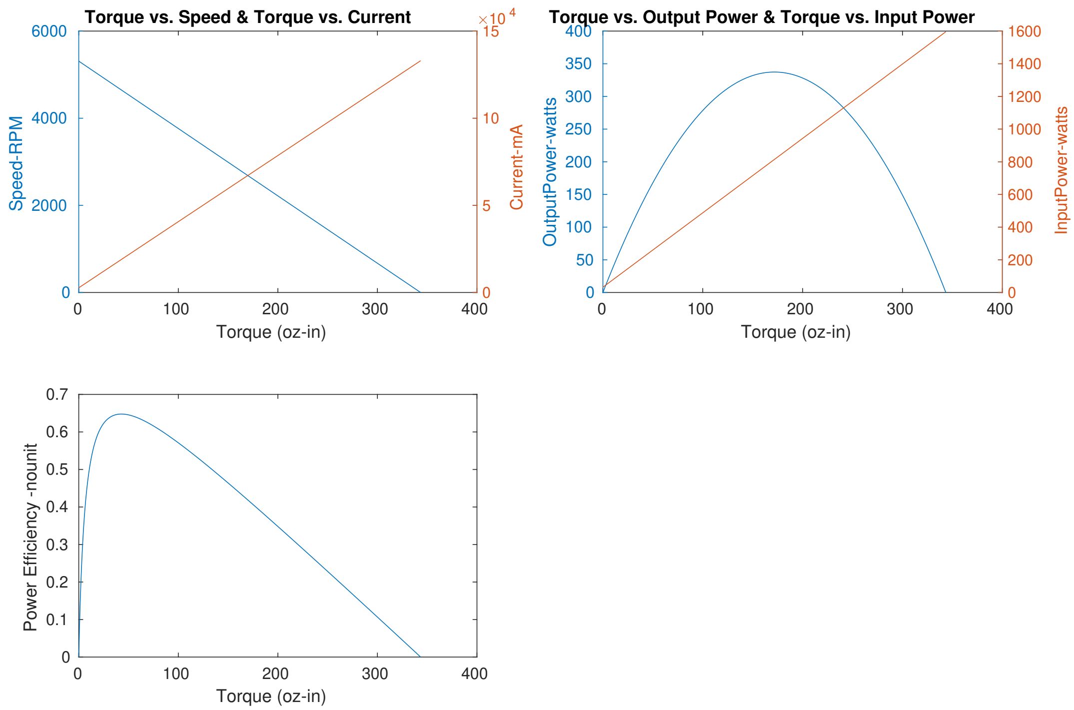


Figure 2.4: Selected DC Motor(Model 3) Characteristics

2.4 Detailed DDV Build Guide

In this section, details of all the hardware components necessary for building the DDV are presented followed by brief instructions to assemble these components. The following is the list of all the required components:

1. **HTC Vive VR System**: We would be using the Vive to track the motion of the DDV to a fraction of a millimeter. The Vive consists of one headset, two base stations, two controllers, and trackers. Each base station can enable tracking in an area of 5m × 5m, and the controllers are required to calibrate the system. We use two base stations to track an area of approximately $8m \times 4m$. The tracker should be placed vertically above the center of the wheel axis of the DDV, and the tracking information consists of position and velocity data along

the x, y and z axis i.e. $(x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})$, as well as the orientation and angular velocity information along these axis i.e. $(\theta_x, \theta_y, \theta_z, \dot{\theta}_x, \dot{\theta}_y, \dot{\theta}_z)$. The update rate of tracking can be varied up to 1kHz [87], [1].

Figure 2.5: From top, Base Stations, VR Headset, Controllers

Figure 2.6: Vive Tracker

2. **NVIDIA Jetson TX2**: Jetson TX2 module is a highly power-efficient(15W) embedded artificial intelligence(AI) computing platform manufactured by

NVIDIA. This module consists of a 256-core NVIDIA Pascal GPU architecture with 256 CUDA cores and two Denver 64-bit CPUs along with Quad-Core A57 Complex. Also, it has 8GB of 128-bit LPDDR4 Memory, 32GB of internal storage, and 59.7GB/s of memory bandwidth. This device is capable of running a standard Ubuntu OS (14.04, 16.04 or 18.04). Additionally, it has a built-in WiFi module with a frequency range of 2.4GHz to 2.5GHz and can be powered by using the 19V Energizer battery pack.

Figure 2.7: NVIDIA Jetson TX2 Module

3. **Arduino Mega 2560**: This is a microcontroller module based on the ATmega2560. It has 16 analog pins and 54 input/output pins out of which 16 are PWM capable. Also, it has 256KB of flash memory, 8KB of SRAM and 4KB of EEPROM, and a 16MHz crystal oscillator. This device can be powered by connecting it to the NVIDIA TX2's USB port.

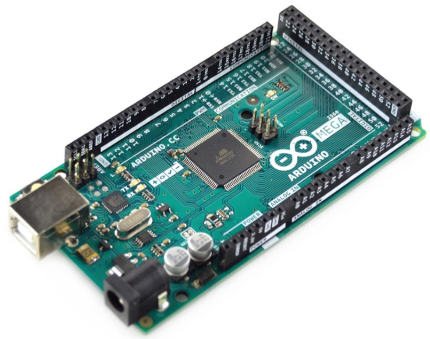


Figure 2.8: Arduino Mega 2560 Module

4. **Vyper DC Motor Driver**: Vyper is a high current(120A continuous current) single-channel DC motor driver which can be operated at 7V to 36V battery voltage range. It can withstand a peak current of 250A and also has a fail-safe shut off functionality in case of control signal disconnection or over-heating. This functionality is highly essential because at full load the actuator can draw up to 133A. Since it is a single-channel controller, we have to use two of these in order to control the robot. Each of these motor drivers is connected to the 12V Li-ion battery supply, and the DC motor.

Figure 2.9: Vyper DC Motor Driver

5. **Wheel Encoder**: This is a Hall-effect sensor-based magnetic encoder. This encoder has a two-channel quadrature output with 256 pulses per channel per revolution (i.e. 1024 counts per revolution) for sensing the speed and the direction of the motor. This encoder should be mounted on the motor drive side - to the shaft, and would require both 3.3V and 5.0V power supply as input (can be supplied by connecting to Arduino Mega Module).

Figure 2.10: Wheel Encoder

6. **Energizer Battery Pack**: This lithium-polymer battery pack has a capacity of 18000mAh and can output DC voltages 5V, 12V, 19V at rated current of 2100mA, 2000mA, and 3500mA respectively. It requires an input of 19V at a

rated current of 3500mA in order to recharge. This battery pack(19V output) is used to power the NVIDIA TX2 module and the 5V output can be utilized to power the USB hub or Arduino Mega module. The USB bub is utilized to connect external devices/sensors such as Intel RealSense Depth Camera or RPLIDAR to the NVIDIA TX2 since the TX2 module has only one USB3.0 port. These devices draw power from the 5V supply connected to the USB hub.

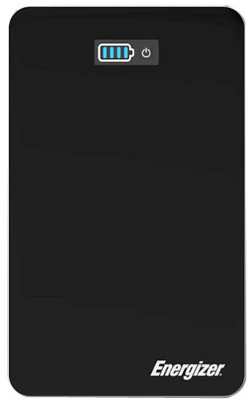


Figure 2.11: Energizer Battery Pack

7. **Hyperion LiPo Battery**: This a 5000mAh lithium-polymer battery pack that outputs a nominal voltage of 11.1V at a continuous discharge rate of 125A and burst discharge rate of 250 A. Since each of the motors can draw a maximum of 133A each, we need two of these lithium polymer batteries to power each motor individually.

Figure 2.12: Hyperion LiPo Battery

8. **Single Pole Single Throw Switch (SPST)**: This SPST switch is required to turn on/off the power supply to the motors. Two SPST switches are required since each motor has a separate power supply. These switches are capable of operating at 250V at a rated current of 180A.

Figure 2.13: SPST Switch

9. **Wheels and Castor Wheels**: The radius of the wheel is directly proportional to the rated torque and inversely proportional to the rated speed of the motor. Hence wheels of any size can be chosen as long as it adheres to the rated speed and rated torque calculations specified in Section 2.3. Apart from the size of the wheel, it is important to choose wheels that can provide sufficient grip between the contact surface and the wheels in order to reduce the chance of slipping. This is important because the DDV dynamical model presented in Chapter 3 assumes there is no slip between the wheels and the contact surface.

When it comes to the castor wheels, we have chosen 0.5-inch metal ball castors over wheel castors since they have very minimal to almost no impact on the dynamical model.

Figure 2.14: Metal Castor Wheel

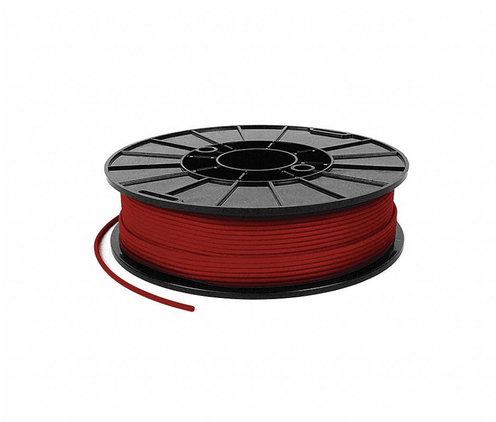
10. **Motor Bracket**: This is a C-Channel Aluminium motor bracket designed to attach the model 3 motor to the chassis of the DDV. Please note that this is not a universal bracket and has to be replaced according to the motor model chosen. We require two motor brackets(one for each motor) and additional information such as CAD models or technical drawings are available in Appendix A.

Figure 2.15: Motor Bracket

11. **Polylactic Acid(PLA) 3D Printer Filament**: PLA is a low cost - biodegradable material that is widely used as a 3D printing filament. A single 1Kg spool

will be sufficient to print all the parts required for 6 DDV's. Additional details about the 3D printed parts will be provided in the next section.

12. **Acrylic Sheets**: This is a transparent thermoplastic homopolymer that is well known for its lightweight and high impact resistance properties. It is used to build the chassis of the DDV; additional details about building the chassis will be provided in the next section. We would require two acrylic sheets of dimensions 62cm $\times$ 42cm $\times$ 0.5cm in order to build one DDV.



(a) PLA 3D Printer Filament

(b) Acrylic Sheets

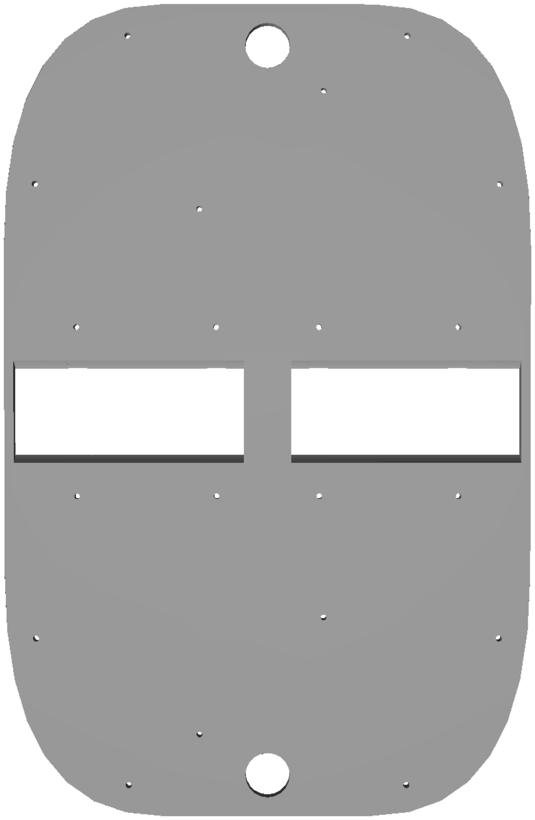
Figure 2.16: DDV Chassis Build Materials

13. **Miscellaneous Parts**: The following table shows the list of various nuts, bolts, spacers, and connecting wires that are required for building the DDV

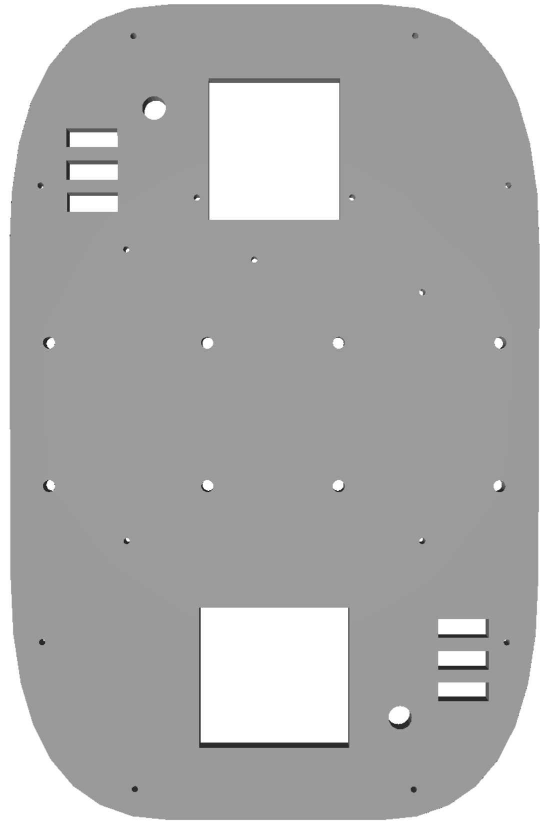
Name	Dimension	Quantity
M3 Bolt,Nut,Flat & Split Washer	30 mm	12
M3 Bolt,Nut,Flat & Split Washer	20 mm	20
M3 Bolt,Nut,Flat & Split Washer	6 mm	3
3/16″ - 32 Bolt	35 mm	4
1/4″ - 28 Bolt,Nut & Washer	30 mm	4
M3 Standoffs	40 mm	24
M3 Standoffs	20 mm	16
XT60 Connectors	-	2

Table 2.1: Miscellaneous Parts

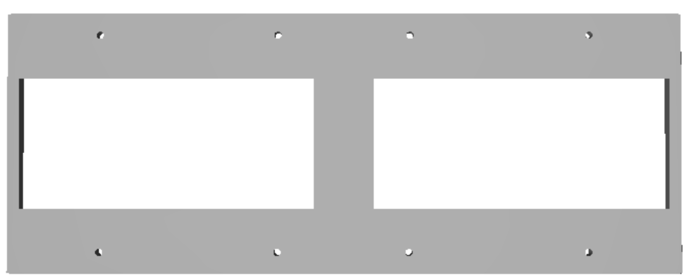
Components Assembly: Let us begin with the list of components that have to be 3D printed or laser cut out of acrylic sheets. The components shown in Figure 2.17 are made out of acrylic sheets, and the components shown in Figure 2.18 are 3D printed. The link to the .stl files of all these models is available in Appendix A.



(a) Base Plate(Top View)

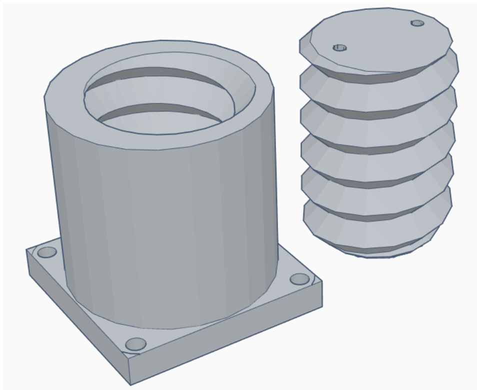


(b) Top Plate (Top View)

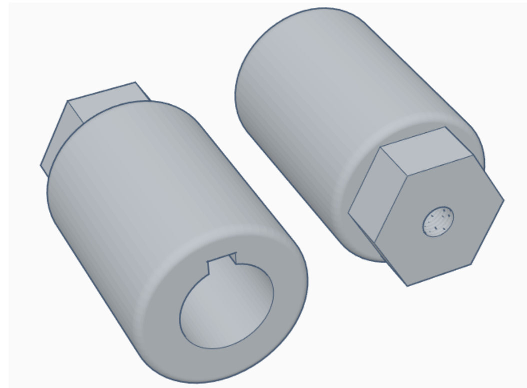


(c) Support Plate (Top View)

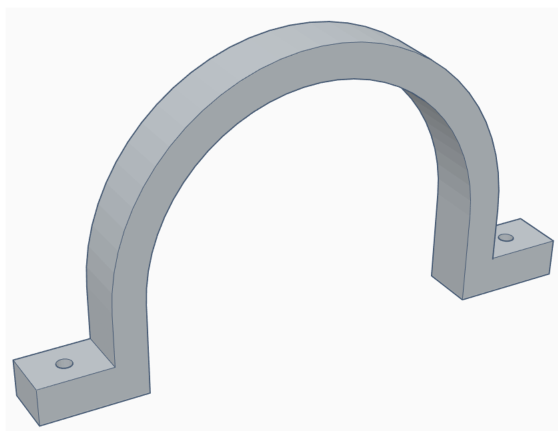
Figure 2.17: Laser Cut Components



(a) Adjustable Castor Wheel Mount (Isometric View)



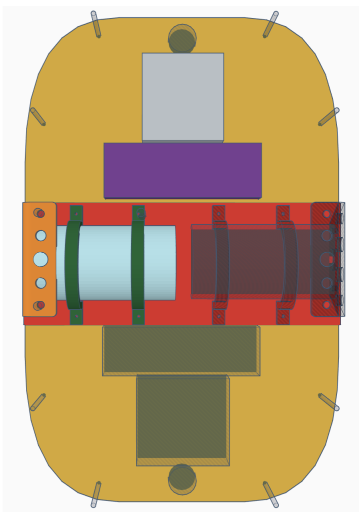
(b) Wheel Adapter (Isometric View)



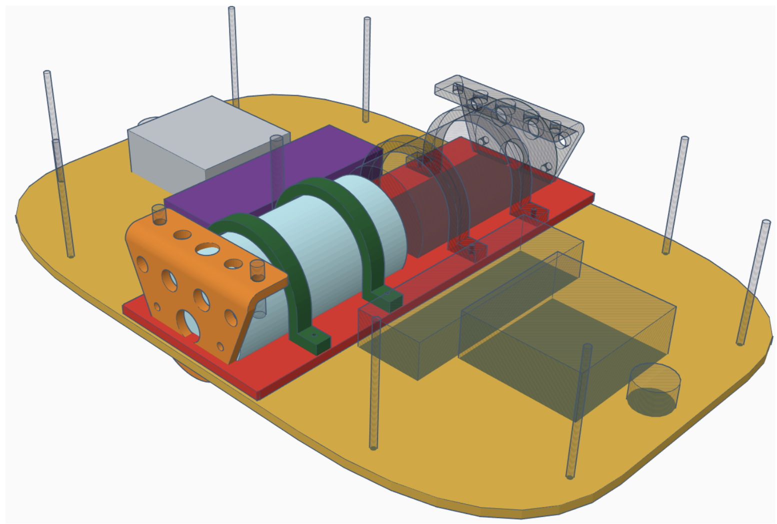
(c) Motor Clamp (Isometric View)

Figure 2.18: 3D Printed Components

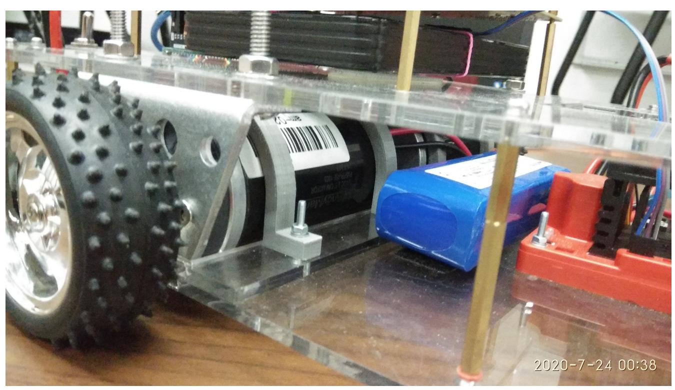
The components shown in Figure 2.17 and the motor clamp shown in Figure 2.18c are required to build the chassis of the vehicle. The following figures will illustrate the assembly procedure. First, attach the encoder (Figure 2.10) and the motor bracket (Figure 2.15) to the motor using the 3/16″- 32 size bolts. Second, arrange the base plate, support plate, and the motors as shown in Figure 2.19 and fix them together with the motor clamp (Figure 2.18c) using the 30mm M3 bolts. In Figure 2.19b, the colors highlight the following components: Yellow - Base Plate; Red - Support Plate; Blue - Motor; Orange - Motor Bracket; Green - Motor Clamp; Violet - Li-Po Battery; Grey - Motor Driver. The motor driver and the Li-Po battery can either be attached using the M3 30mm bolts or Velcro strips.



(a) Top View



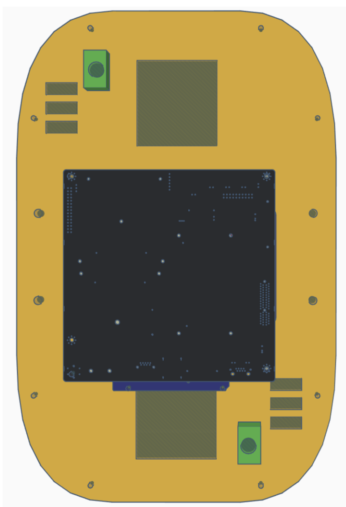
(b) Isometric View



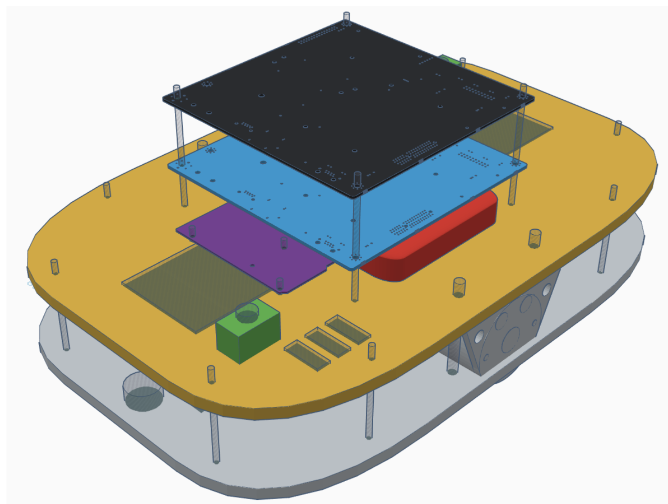
(c) Actual Assembly

Figure 2.19: Chassis Assembly 1

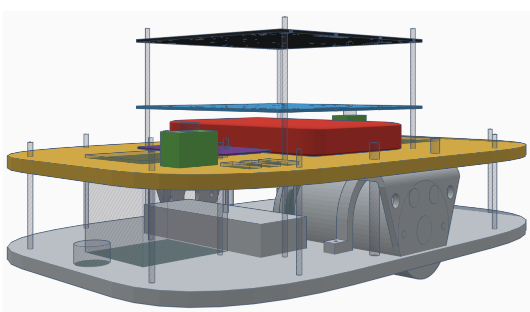
Second, we have to fix the components (i.e. Arduino, SPST switch, NVIDIA TX2, Vive tracker, Li-Po Battery) on the top plate, before attaching the top plate to the motor clamps using the 1/4″ - 28 size bolts (Figure 2.20g) and M3 standoffs (Figure 2.20f). The SPST switch can be screwed into the base plate (Figure 2.20h), while the Energizer Li-Po battery can be fixed using Velcro strips. The Arduino Mega and the TX2 modules have to be attached using the M3 standoffs. This assembly procedure can be seen in Figure 2.20. In Figure 2.20b, the colors highlight the following components: Yellow - Top Plate; Green - SPST Switch; Red - Energizer Li-Po Battery; Violet - Arduino Mega Module; Blue - NVIDIA TX2 Module; Black - Cardboard/Plastic Base for Tracker



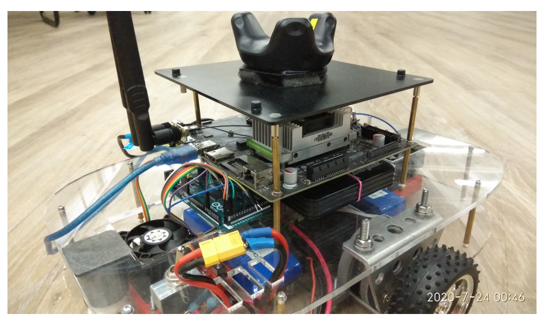
(a) Top View



(b) Isometric View

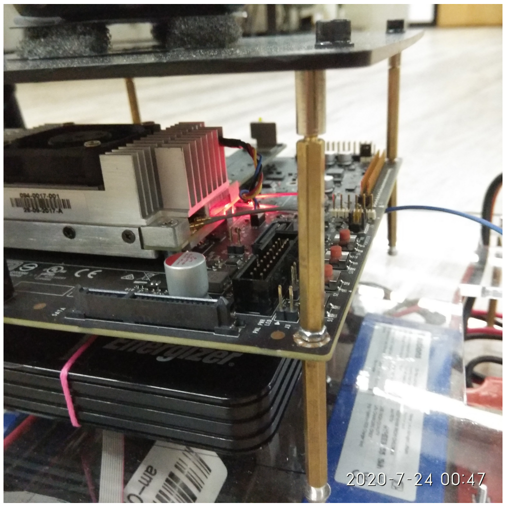


(c) Side View

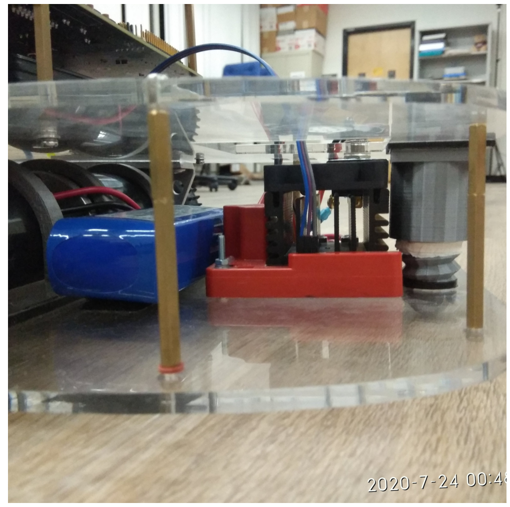


(d) Actual Assembly

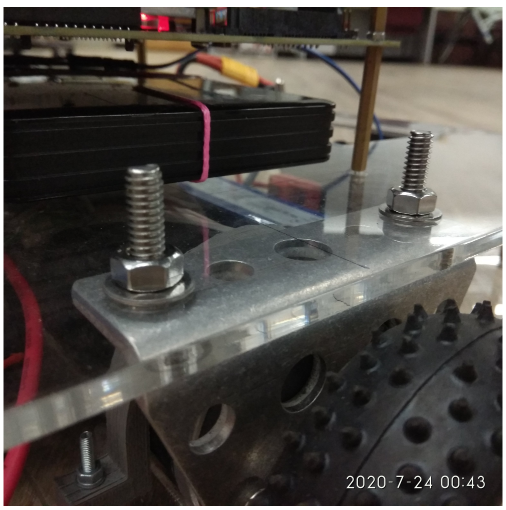
Figure 2.20: Chassis Assembly 2



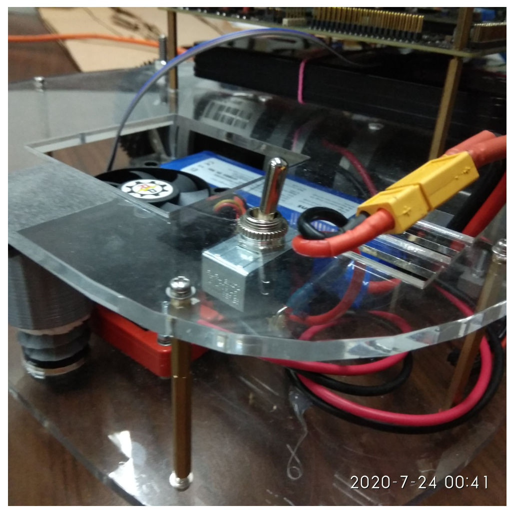
(e) M3 Standoffs(20mm, 60mm)



(f) M3 Standoffs(20mm, 60mm)



(g) Top Plate - Motor Bracket Assembly



(h) SPST Switch

Figure 2.20: Chassis Assembly 2

Third, the wheels should be attached to the motor shaft using the wheel adapter shown in Figure 2.18b. Next, the castor wheels (Figure 2.14) should be fixed to the adjustable castor wheel mounts (Figure 2.18a). As shown in Figure 2.21 this castor wheel setup should be attached to the front and rear of the DDV by either using hot glue or 20mm M3 bolts. The adjustable mount allows the castor wheels to be raised or lowered based on the requirement. This feature is required because over prolonged use of the DDV, the elasticity of the wheels changes and this can cause a change in the height of the DDV which requires a realignment of the castor wheels. Furthermore, depending on the rigidity of the surface/ground material, the drop in the DDV height varies, this again requires a realignment of the castor wheels height in order to maintain proper contact.

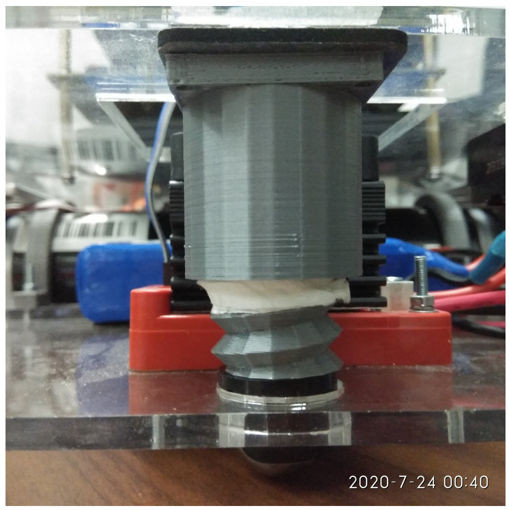


Figure 2.21: Adjustable Castor Wheel

Finally, the Vive tracker should be attached vertically above the center of the wheel. Since the Vive tracker is prone to low-frequency noise caused by chassis vibrations or uneven ground surfaces, a damper such as foam or sponge (Figure 2.22) can be sandwiched between the tracker and the DDV. This damper will absorb the vibrations and thereby improves the quality of measurement. Figure 2.23 shows the final form of the DDV.

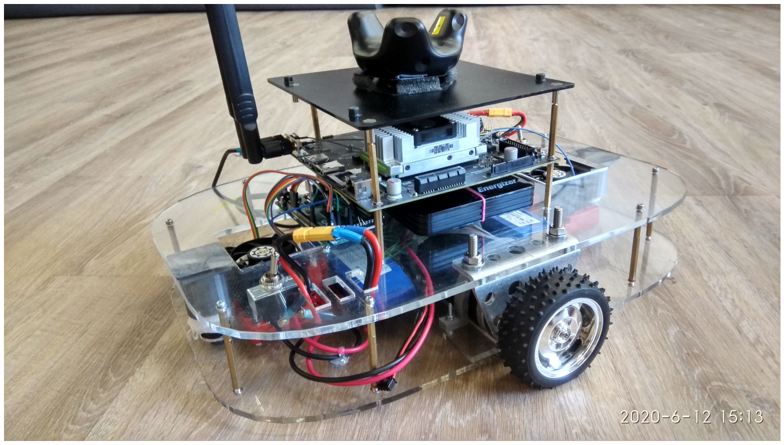
Figure 2.22: Tracker with Damper

2.5 C^4S Requirements and Software Architecture

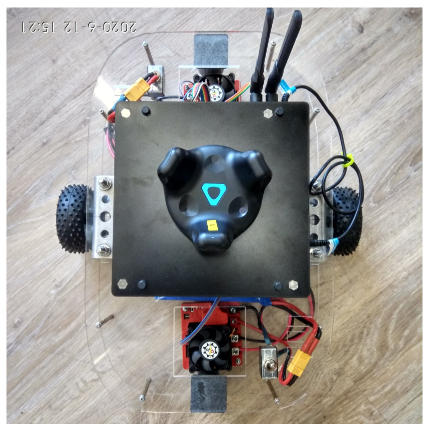
In this section, a detailed overview of the system-level architecture involved in addressing the global/local command, control, computing, communications (C^4), and sensing (S) requirements of the fleet of DDV's. Figure 2.24 describes the software architecture of the DDV, and the following discussion on the global/local C^4S requirements will be centered around this architecture.

- **Global and Local Computing**: The purpose of the global/central computer is to gather information from various sensors and perform all the heavy computing that would facilitate an analytical understanding of the performance of all the robotic vehicles [3] in the fleet and also for several other purposes. These purposes include online optimization, decision making, data transmission/broadcasting, objective adaptation, etc. The local computing involves onboard computers or embedded devices that handle the low-level control and sensing requirements of a single robotic vehicle. In this thesis, the global/central computing is performed by an extremely powerful DELL Precision 5820

[3]robotic vehicles include ground, air, space, sea or underwater vehicles



(a) Isometric View



(b) Top View

Figure 2.23: DDV with Vive Tracker

Tower Workstation. This workstation runs the Ubuntu 16.04 Operating System (OS) and is directly linked to HTC Vive VR System, and also with all the DDV's via WiFi. This workstation computes the position and velocity information $(x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})$ of all the DDV's, accepts the control commands from the user, and transmits this information to all the DDV's via WiFi. Apart from the transmission of data, it also records the crucial information sent by different DDV's. We utilize the Robotics Operating System (ROS) framework to write

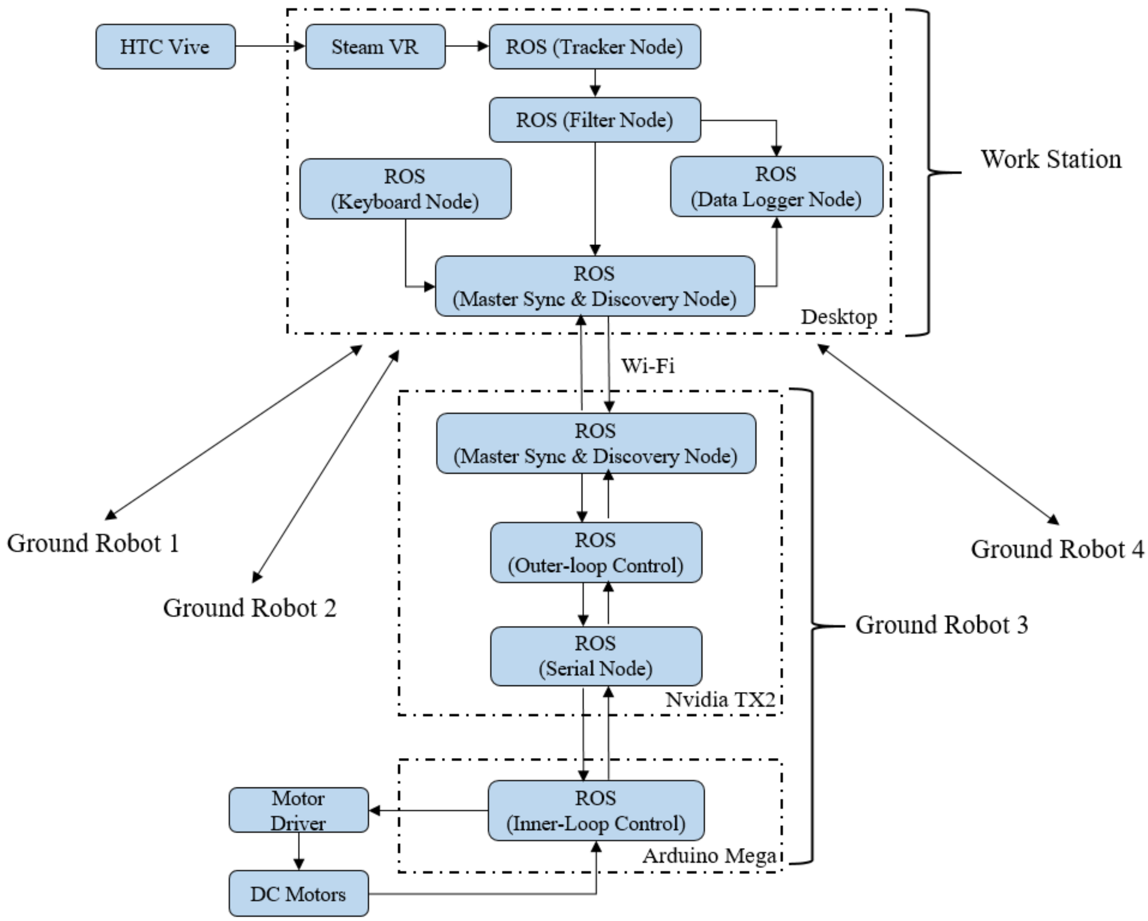


Figure 2.24: DDV Software Architecture

all the software nodes that would facilitate the process mentioned above. An overview of various nodes run in the workstation are shown in Figure 2.24 and a brief description of each of these nodes is mentioned further below.

When it comes to local computing, each of the DDV's is equipped with NVIDIA TX2 Module and an Arduino Mega Module. These on-board embedded devices handle all the computations required for the independent functioning of the DDV's. Generally, the NVIDIA TX2 board runs the outer loop controller and handles data from sensors such as Intel RealSense Camera, RPLIDAR, IMU, etc, while the Arduino Mega runs the inner-loop controller, sends commands to the motor drivers, and handles data from wheel encoders. In this thesis, we utilize the NVIDIA TX2 to run the outer loop controller, communicate with the Arduino module, and to transmit/receive data from the central workstation. The TX2 module runs the Ubuntu 16.04 OS and utilizes the ROS framework to implement the above functionalities. The Arduino Mega runs the inner-loop controller, obtains the rotation information from the wheel encoders, receives/-transmits information with the NVIDIA TX2 module, and sends the commands to the motor driver. A detailed overview of the various ROS nodes run in the TX2 and Arduino modules are shown in Figure 2.24.

- **Global and Local Sensing**: The purpose of global sensing is to understand the state of the robotic vehicles in the fleet and also to detect the changes in the operating environment. The global sensing suite can involve sensors such as stereo cameras, motion capture systems, QR code scanners, etc. In this thesis, we are using an HTC Vive Motion Capture System and the data received from the individual vehicles in the fleet as the global sensing unit. This system enables us to precisely determine the position and velocity information of individual vehicles as well as the status of inner and outer control loops.

  The local sensing consists of Hall effect sensor based magnetic encoders that provide the linear and angular velocity of the vehicle. Though the HTC Vive Motion Capture system can provide us with data up to fraction of a millimeter accuracy, the only disadvantage is that it is not portable. Nevertheless, we still use it because the focus of this thesis is to conduct and analyze the trade-studies. In the future, a vision-lab-based localization system can be considered instead of the current system since they offer greater portability and lesser computing power requirement. [49], [55], [76] and [13] highlight the on-going works in this area.

- **Command and Control**: As mentioned earlier, the central workstation does all the heavy computing such as obtaining data from the global sensing suite, data logging, filtering, estimation, trajectory planning, switching between control schemes/objectives, safety-critical maneuvers, etc. Basically, the central workstation updates the objectives for individual vehicles while continuously monitoring their performance. During safety-critical maneuvers that involve a possible collision amongst the vehicles or with an external object the central workstation overrides the local control of individual vehicles or issues commands to bring them to an immediate halt. In this thesis, during a safety-critical scenario, a human operator can assume control of the local vehicle - manual control mode. The user can issue velocity and angular velocity commands using the keyboard or joystick connected to the central computer. When multiple vehicles are involved, the user can choose to control the vehicles individually or stop the motion of the entire fleet. In the future, this process can be completely automated.

  The local command and control are constrained to individual vehicles. Each vehicle would receive commands from the central workstation. These commands include start/stop motion, trajectory coordinates, switching between control schemes such as manual control (receives direct commands from the user), cruise control, planar Cartesian stabilization, or position control. More specifically, the local control can be divided into two parts: Outer-Loop Control; Inner-Loop Control. The outer-loop control/controller issues command to the inner-loop controller based on the commands received from the central workstation. The inner-loop controller is associated with the linear and angular velocity (v, ω) control of the DDV. It receives the desired linear and angular velocity commands (v_{ref}, ω_{ref}) from the outer loop controller and generates the actuator control - PWM signals (u_1, u_2) to vary the speed of the motors.

- **Global and Local Communications**: Here, the global communication refers to data transfer between the individual vehicles and the central workstation; and the local communication refers to the communication between various vehicles within the fleet and also the communication between various devices within a vehicle. Global communication and inter-vehicle communication is achieved over WiFi (IEEE 802.11 (2.4GHz)). In the case of intra-vehicle communication, the data exchange between Arduino Mega, NVIDIA TX2, and various sensor modules is achieved via serial communication.

- **Software Framework**: As mentioned before, Ubuntu 16.04 OS is installed on the workstation and in each of the NVIDIA TX2 boards. All the NVIDIA TX2's and the workstation are connected to a common WiFi network. Further, we utilize the ROS framework to implement the various function discussed earlier. The following is a brief description of the various nodes mentioned in Figure 2.24.

  - *Steam VR.* Steam VR is a part of the Steam application suite. It provides the drivers for the HTC Vive VR hardware and also the software support that is required to convert the raw sensor data recorded by the Vive into a meaningful pose and velocity data.

  - *Tracker Node.* This node acts as a bridge between the ROS framework and the Steam VR application. This node obtains the pose and velocity data generated by the Steam VR app and converts them into ROS compatible data (topics/messages). Further to this, this node converts the data from quaternions to Euler angles and performs basic axis rotation and translation operations in order to map the data with the real-world work-space. Apart from this, this node can also adjust the frequency of the pose and velocity data before sending it to the filter node.

– *Filter Node.* This node obtains the pose and velocity information from the tracker node and passes it through a moving average filter (a simplified form of low pass filter) in order to remove the high-frequency noise. The window size of this moving average filter has to be determined by trial and error.

– *Master Sync* & *Discovery Nodes.* The master sync and discovery nodes are part of the FKIE Multimaster package [84]. The main idea of these nodes is to set up and manage a multi-master network. In simpler terms, this package lets every device present on a network run their own ROS Master (roscore) while facilitating the exchange of topics and services across these devices i.e. multi-master network. In our case, each of the devices connected to the WiFi network i.e. the workstation and ground robots (Nvidia TX2's), have their own ROS master running and each of them run the master sync and discovery nodes in order to exchange information among themselves. The number of devices that can communicate using this package is limited by the WiFi router capacity i.e. the maximum number of clients allowed by the router. Additional details about the functionalities of master sync and discovery nodes can be obtained from [32].

– *Keyboard Node.* As the name suggests, this node obtains the user input via the keyboard or the joystick and transmits it to the ground robots. Proper functioning of this node is crucial for all the manual control operations.

- *Data Logger Node.* This node obtains the data from the filter node and individual ground robots and records them into .csv files. The data recorded by this node consists of the global pose and velocity information of each of the ground robots and also data such as control signal, error signal, encoder readings, and other miscellaneous data that is required to analyze the performance of the inner and outer-loop controllers in each of the ground robots.

- *Outer-loop Control Node.* This node runs on the NVIDIA TX2 in each of the ground robots and it implements the outer loop control laws and also establishes communication with the ground station and the other robots in the network.

- *Serial Node.* The serial node is responsible for establishing serial communication between the Arduino Mega and the NVIDIA TX2.

- *Inner-Loop Control Node.* This is the most preliminary node in this software framework and it runs on the Arduino Mega in each of the ground robots. This node implements the inner-loop control laws, obtains the rotation data from the encoder, provides the control signals for the motor drivers, and communicates with the outer-loop control node running on the NVIDIA TX2.

Chapter 3

MATHEMATICAL PRELIMINARIES

3.1 Overview

The work presented in this thesis mainly utilizes concepts from the following areas: classical control theory, optimization, non-linear systems, linear systems, dynamic modeling. Most of the content presented in the upcoming chapters should be easy to comprehend provided the reader has a basic background in the topics mentioned above. This chapter reviews some of the mathematical concepts that are frequently used throughout this work. These mathematical concepts include discretization of continuous-time linear state-space models, conversion between Euler angles and quaternions, and linearization of nonlinear systems.

3.2 Discretization of Linear State Space Models

Discretization is the process of converting a continuous-time system into a discrete-time system. This process is highly crucial: in order for any continuous time variable, mathematical functions to be analyzed using a digital computer, it has to be discretized first; in order to implement a controller on an embedded platform such as Arduino or NVIDIA TX2, it has to be discretized first. Therefore, it is highly crucial to understand the properties or behavior of a system post the discretization process. The behavior of a discrete-time system converges with that of its continuous-time equivalent only when the sample time $T_s \rightarrow 0$. However, this assumption is not possible in the real-world since the computations become intractable as the sampling time is reduced. So in order to reduce the discretization error (the deviation of the discrete model behavior with respect to its continuous-time equivalent) various numerical approximation techniques are developed, such as Forward Euler, Backward Euler, Tustin, and Zero-order Hold (Exact discretization) (Chapter 3 in [62]).

Let us consider a continuous-time representation of plant $P(s)$ in state-space representation as:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{3.1}$$

$$y = Cx(t) + Du(t) \tag{3.2}$$

The discrete time equivalent of this system can be written as:

$$x_{k+1} = A_d x_k + B_d u_k \tag{3.3}$$

$$y_k = C_d x_k + D_d u_k \tag{3.4}$$

The following table provides the relation between the discrete-time matrices (A_d, B_d, C_d, D_d) and the continuous-time matrices (A, B, C, D) for each of the approximation method. The last row shows the relation between the s domain and z domain for each of the approximation methods [50].

Continuous	**Forward Euler**	**Backward Euler**	**Tustin**	**ZOH**
A	$A_d = I + TA$	$(I - TA)^{-1}$	$(I + \frac{AT}{2})(I - \frac{AT}{2})^{-1}$	e^{AT}
B	$B_d = TB$	$T(I - TA)^{-1}B$	$(I - \frac{AT}{2})^{-1}B\sqrt{T}$	$\int_0^T e^{AT} B d\tau$
C	$C_d = C$	$C(I - TA)^{-1}$	$\sqrt{T}C(I - \frac{AT}{2})^{-1}$	C
D	$D_d = D$	$D + C(I - TA)^{-1}BT$	$D + C(I - \frac{AT}{2})^{-1}B\frac{T}{2}$	D
$P(s)$	$s = \frac{1}{T}(z-1)$	$s = \frac{1}{T}\frac{(z-1)}{z}$	$s = \frac{2}{T}\frac{(z-1)}{(z+1)}$	$(1 - z^{-1})Z[\frac{P(s)}{s}]$

Table 3.1: Discretization Methods

When it comes to designing and implementing a controller for a plant, there are two methods: 1) Approximate method 2) Exact method. In the approximate method, a continuous-time controller is first designed based on the continuous-time plant and it is later discretized using one of the numerical approximation methods. In the exact method, the continuous-time plant is first discretized and a discrete-time controller is designed based on the discrete-time plant. In this thesis, we will be using the first method i.e. the approximate method for designing the controllers. And also, out of the four different approximation methods that have been mentioned above, Tustin approximation best preserves the frequency domain characteristics of the continuous-time plant post discretization, while the ZOH best preserves the time domain characteristics of the continuous-time plant post discretization. In this thesis, we would be using the ZOH in order to discretize the continuous-time controllers.

3.3 Reference Frames, Euler Angles and Quaternions

The position of an object in a three-dimensional Euclidean space can be represented using a three-dimensional vector. Several coordinate systems have been developed to describe a point in three-dimensional space such as Cartesian coordinates, spherical coordinates, cylindrical coordinates, etc. In this thesis, we would be primarily dealing with the Cartesian coordinates system to model the dynamics of the vehicle. When it comes to representing the orientation of the object, we will be using the Euler angles since they provide us with an intuitive understanding of the real-world object orientation. Before we proceed further into understanding Euler angles and rotation matrices, it is very important to discuss the frame of reference.

In order to define the position and orientation of an object, it is important to define the frame of reference from which the position and orientation of the system are being measured/observed. In most of the ground-robot literature, we come across two different frames of reference: Ground/Earth Frame of Reference, Robot Frame of Reference. The ground frame is a global frame that is fixed to the environment or the plane in which the DDV moves. This frame is denoted by (X_I, Y_I, Z_I); for a DDV the Z_I axis can be ignored since its motion is constrained within the $X - Y$ plane. The robot frame of reference is a local reference frame attached to the DDV, and thus, keeps moving with it. This frame is denoted as (X_r, Y_r). These frames are visualized in Figure 3.1. Please note that the coordinate systems defined with respect to the ground frame and robot frame follow the Right-Hand rule.

Since the movement of the DDV is constrained within a plane, we would only require a single variable θ (yaw angle) to define the orientation of the DDV at any given point in space. If the Z_I and Z_r axis are parallel, then the yaw angle can also be defined as the angle by which the ground frame should be rotated about the Z_I axis in order to align with the robot frame. Also, the primary reason for defining a robot coordinate frame is because, in multi-robot problem formulations such as trajectory tracking, formation control, obstacle avoidance, etc., it is observed that defining the error dynamics of the system with respect to the robot frame simplifies the complexity of the error dynamics and also provides a more intuitive understanding of the system behavior [16], [36].

Consider the scenarios presented in Figure 3.1. In this case, the coordinates of the point P with respect to the robot frame can be obtained using the following equations.

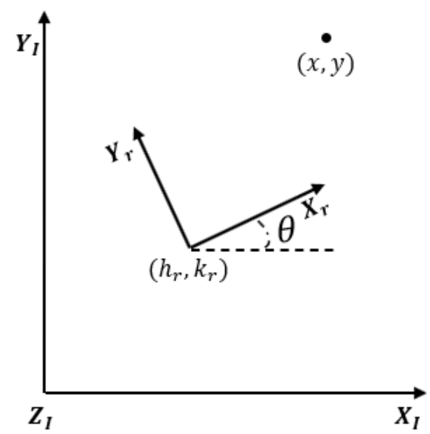


Figure 3.1: Rotation of Translation of Coordinate Axis in Two Dimensions

$$x' = (x - h_r)\cos\theta - (y - k_r)\sin\theta \tag{3.5}$$

$$y' = (x - h_r)\sin\theta + (y - k_r)\cos\theta \tag{3.6}$$

This operation can be generalized for the three-dimensional case as well. Now, one might wonder why do we need to consider three-dimensional rotation or transnational operations since a two-dimensional space is sufficient to define the motion and the associated dynamics of the DDV. This is because the existing localization and mapping libraries currently available (including the HTC Vive system) provide the position and orientation of the object in three-dimensional space. In order to work with these libraries and manipulate the data as per our requirement, we need to be aware of rotation operations in three-dimensional space. It is also important to note that in most of these libraries/software packages the rotation operations are represented in Quaternions - more information regarding Quaternions will be discussed in the upcoming paragraphs.

Consider the following Figure 3.2. Here, (X_I, Y_I, Z_I) represent the ground coordinate axis and (X_r, Y_r, Z_r) represent the robot coordinate axis. The angles $\phi, \theta, and \psi$ represent rotation around X_I, Y_I and Z_I axis respectively. We use the standard right-hand rule to assign the direction of rotation i.e. rotation in the counterclockwise direction is considered to be positive.

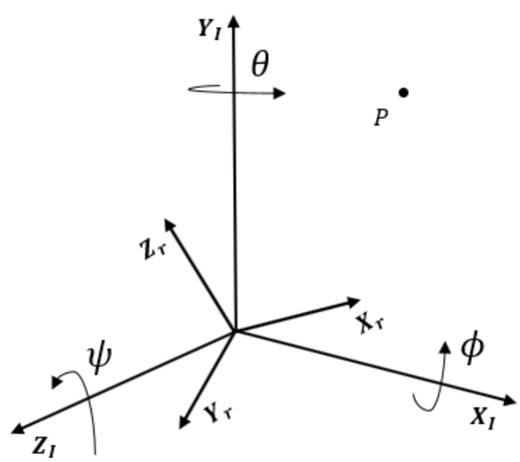


Figure 3.2: Rotation in Three Dimensions

Given the angles ϕ, θ and ψ - the angles by which the ground coordinate axis has to be rotated in order to coincide with the robot coordinate axis [1] , the new

[1]Please note that rotation operation is non-commutative

coordinates of point P with respect to robot coordinate axis (P') are given by the following rotation matrices:

$$P' = R_{z(\psi)y(\theta)x(\phi)}P = R_{z(\psi)}R_{y(\theta)}R_{x(\phi)}P \tag{3.7}$$

where

$$R_{y(\theta)} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.8}$$

$$R_{x(\phi)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{3.9}$$

$$R_{z(\psi)} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

$$\begin{aligned} R_{z(\psi)y(\theta)x(\phi)} &= R_{z(\psi)}R_{y(\theta)}R_{x(\phi)} \\ &= \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\cos\phi & \cos\theta\cos\phi \end{bmatrix} \end{aligned} \tag{3.11}$$

Although the rotation matrix has nine entries, it really requires only three numbers ϕ, θ, and ψ to construct it. Using rotation matrix can be considered computationally inefficient since each of the nine elements needs to be calculated and when applied within a loop, perhaps after thousands of iterations, the numerical inaccuracies due to sin and cosine computations can degrade the orthogonality i.e. the rows will lose their orthogonality [39]. This begs the question: is there a better way to represent rotations in three-dimensional space and perhaps reducing the numerical complexity in the process? This problem is addressed by Quaternions. A quaternion is a 4-tuple that can represented as $q_0 + q_1 i + q_2 j + q_3 k$, where $(q_i \in \mathbb{R})$ and the symbols i, j, k satisfy the following identities: $i^2 = j^2 = k^2 = -1; ij = k; ji = -k; jk = i, kj = -i; ki = j, ik = -j$. In [10], the authors have presented a more detailed and intuitive explanation for quaternions.

A vector in three dimensional space can be expressed using quaternion as $q = 0 + xi + yj + zk$. A rotation operation can be expressed using the quaternion q_R such that norm $|q_R| = 1$. The coordinates of point P defined in the ground frame can now be expressed with respect to the robot frame as:

$$q_{P'} = q_R \; q_P \; q_R^* \tag{3.12}$$

and here

$$q_R = q_0 + q_1 i + q_2 j + q_3 k \tag{3.13}$$

$$q_R^* = q_0 - q_1 i - q_2 j - q_3 k \tag{3.14}$$

Given the Euler angles - ϕ, θ, ψ one can calculate the q_R as follows:

$$|q_0| = \sqrt{\frac{Trace(R) + 1}{4}} \tag{3.15}$$

$$|q_1| = \sqrt{\frac{R_{11}}{2} + \frac{1 - Trace(R)}{4}} \tag{3.16}$$

$$|q_2| = \sqrt{\frac{R_{22}}{2} + \frac{1 - Trace(R)}{4}} \tag{3.17}$$

$$|q_3| = \sqrt{\frac{R_{33}}{2} + \frac{1 - Trace(R)}{4}} \tag{3.18}$$

here R is the rotation matrix from equation (3.11). Similarly, given q_R the rotation matrix can be calculated as follows:

$$R = 2\begin{bmatrix} q_0^2 + q_1^2 - 0.5 & q_1q_2 - q_0q_3 & q_0q_2 + q_1q_3 \\ q_0q_3 + q_1q_2 & q_0^2 + q_2^2 - 0.5 & q_2q_3 - q_0q_1 \\ q_1q_3 - q_0q_2 & q_0q_1 + q_2q_3 & q_0^2 + q_3^2 - 0.5 \end{bmatrix} \tag{3.19}$$

One common misconception about quaternions is that they can prevent the gimbal lock problem. In fact, there have been several articles on the internet that address this issue incorrectly and often attribute this to the use of Euler angles. In [39], the authors have given a clear explanation (including experimental results) regarding the significance of using quaternions over rotation matrices; the gimbal lock problem, and how it has been misinterpreted by the so-called internet community.

3.4 Linearization of Nonlinear Systems

Consider a non-linear state space representation of form:

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \tag{3.20}$$

$$y(t) = g(x(t), u(t)) \tag{3.21}$$

where, f is function mapping $\mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, and g is a function mapping $\mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$. A point $x_e \in \mathbb{R}^n$ is called an equilibrium point if there a specific $u_e \in \mathbb{R}^m$ such that

$$f(x_e, y_e) = 0 \quad \text{for all } t \geq 0 \tag{3.22}$$

A linear state space representation that approximates the non-linear system about the equilibrium point (x_e, y_e) can be obtained as follows [66]:

$$\delta\dot{x}(t) = A\delta x(t) + B\delta u(t), \quad \delta x(0) = \delta x_0 \tag{3.23}$$

$$\delta y(t) = C\delta x(t) + D\delta u(t) \tag{3.24}$$

where

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}_{(x_e,u_e)} \qquad B = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \cdots & \frac{\partial f_1}{\partial u_m} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial u_1} & \cdots & \frac{\partial f_n}{\partial u_m} \end{bmatrix}_{(x_e,u_e)} \tag{3.25}$$

$$C = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial g_p}{\partial x_1} & \cdots & \frac{\partial g_p}{\partial x_n} \end{bmatrix}_{(x_e,u_e)} \qquad D = \begin{bmatrix} \frac{\partial g_1}{\partial u_1} & \cdots & \frac{\partial g_1}{\partial u_m} \\ \vdots & \vdots & \vdots \\ \frac{\partial g_p}{\partial u_1} & \cdots & \frac{\partial g_p}{\partial u_m} \end{bmatrix}_{(x_e,u_e)} \tag{3.26}$$

$$\delta u(t) \stackrel{\text{def}}{=} u(t) - u_e; \quad \delta x(t) \stackrel{\text{def}}{=} x(t) - x_e; \tag{3.27}$$

$$\delta y(t) \stackrel{\text{def}}{=} y(t) - y_e; \quad \delta x_0 \stackrel{\text{def}}{=} x_0 - x_e; \quad y_e \stackrel{\text{def}}{=} g(x_e, u_e) \tag{3.28}$$

Chapter 4

MODELLING AND DESIGN TRADE STUDIES FOR A DIFFERENTIAL-DRIVE MOBILE ROBOT

4.1 Introduction and Overview

In this chapter, we try to shed some light on the dynamic properties of three different input/output representations of DDV models [1] . Further to this, we also try to answer some of the fundamental questions associated with the modeling of DDV's: 1) How the different design parameters (such as mass, moment of inertia, center of gravity, radius of wheels) will impact the behavior of these models? 2) When is a decentralized controller sufficient? When is a centralized controller necessary. In order to answer these questions, we first present a brief overview of the kinematic (Section 4.2.1) and dynamic model (Section 4.2.2) of the DDV followed by actuator dynamics and parameter estimation (Section 4.2.3). The two-input two-output (TITO) nonlinear time-invariant model, taken from [22], will form the basis of this discussion. Next, we present the three different models of the DDV and analyze each of these designs in the frequency domain while varying the different design parameters mentioned above. Further, the frequency domain analysis presented in this chapter will pave the way for illustrating the impact of these trade studies on the performance of speed and position-direction control laws presented in Chapter 4.

[1] $P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$; $P_{[e_{a_r},e_{a_l}]\rightarrow[v,\omega]}$; $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[\omega_r,\omega]}$. Here, (e_{a_r}, e_{a_l}) represent the voltage inputs to the right and left actuators, (v, ω) represent the linear and angular velocity of the vehicle, and (ω_r, ω_l) represent the right and left wheel angular velocities

4.2 Modeling of a Differential-Drive Ground Robot

Mobile robots are categorized as differential drive vehicles because of their use of the so-called differential drive mechanism. This mechanism involves two motors, aligned along the same axis, that can rotate independently. Rotation in the same direction allows the vehicle to go forward/backward, while rotation in opposite directions allows the vehicle to turn left/right. This differential drive mechanism eliminates the necessity for a steering mechanism, which in turn simplifies the dynamics of the vehicle and thereby used by the majority of researchers. Withing in this thesis, we assume that both of the actuators are identical in order to simplify the dynamical model and gain useful insights, however, in practice the differences in the actuators should be considered.

We first discuss the kinematic model of the DDV, followed by the dynamic model without including the DC motor dynamics. Next, we discuss the actuator dynamics and present a brief overview of the procedure for estimating the motor dynamics. Finally, we present the linear state-space representation of the dynamic model of the DDV including the actuator dynamics. In order to comprehend the discussions presented in forthcoming sections, it is necessary to take a look at the various parameter definitions presented in Table 4.1. Further to this, in Table 4.2, the nominal values for these vehicle parameters have been specified. Please note that column two (DDV - 150 RPM) in Table 4.2 corresponds to an older DDV that is equipped with low-speed high torque - 150 RPM motors, while column three corresponds to the high-speed high torque - 5,300 RPM motors described in Chapter 2. The parameters mentioned in Table 4.2 are directly measurable except for I, the value of I is estimated by approximating the moment of inertia values of individual components and combining them together using the parallel and perpendicular axis theorems. The MATLAB code for estimating the value of I has been presented in Appendix B.

Parameter	**Definition**
m	Mass of fully loaded vehicle, $m = m_c + 2m_w$
m_c	Mass of vehicle without wheels and motors
m_w	Mass of single wheel-motor combination
I_w	Wheel+motor moment of inertia about axle
I	Total inertia: $I = I_c + m_c d^2 + 2m_w L_w^2 + I_w$
r	Radius of wheels
l	Length of the robot chassis
d_w	Distance between two wheels (at midpoint)
d	Distance c.g. lies forward of wheel axles
L_a	Armature inductance
R_a	Armature resistance
K_b	Back EMF constant
K_t	Torque constant
K_g	Motor-wheel gear (down) ratio
β	Speed damping constant

Table 4.1: DDV Nominal Parameter Definitions

Parameter	Value (DDV - 150 RPM)	Value (DDV - 5,300 RPM)
m	3.4 kg	6.80 kg
m_c	2.76 kg	4.17 kg
m_w	0.32 kg	1.315 kg
I	0.042 kg m^2	0.332 kg m^2
r	0.042 m	0.039 m
l	0.28 m	0.434 m
d_w	25 m	0.324 m
d	0.025 m	0 m

Table 4.2: DDV Nominal Parameter Values

4.2.1 Differential-Drive Robot Kinematics

The purpose of the kinematic model is to represent the motion of the system without considering the mass, inertia, or the forces affecting the motion. For a DDV, the purpose of a kinematic model is to relate the linear and angular velocities of the vehicle with the geometric parameters of the vehicle i.e. wheel radius, and width of the vehicle.

The various parameters associated with the DDV are presented in Figure 4.1. Let (X_I, Y_I) represents that coordinate axis of the global frame, and it can be seen that (x, y) represent the position coordinates of the DDV with respect to the global frame and θ represents that angle made by the DDV's longitudinal axis with the X_I axis. More precisely, the (x, y) coordinates correspond to the center of the wheel axis.

Before we present the kinematic model, it is important to understand the constraints/assumptions that characterize the motion of a differential drive vehicle.

- **Zero Lateral Motion**: This means that the DDV cannot move along its lateral axis. This constraint is quite intuitive since the vehicle cannot move sideways without turning unless the vehicle is fitted with mecanum wheels. It is this constraint that results in the non-holonomic nature of the DDV. Therefore, by equating the velocity of the vehicle in the lateral direction to zero, we obtain the following condition

$$-\dot{x}\sin\theta + \dot{y}\cos\theta = 0 \tag{4.1}$$

- **Zero Wheel Slip/Pure Rolling Motion**: This constraint assumes that there is always sufficient friction between the wheels and the ground surface to ensure that there is no slip along the longitudinal axis and no skidding along the lateral axis of the DDV.

Based on the above constraints, the kinematic model of the robot can be derived as shown in [22]. Equation (4.2) shows the kinematic model of the DDV.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega \tag{4.2}$$

where $v = \sqrt{\dot{x}^2 + \dot{y}^2}$ represents the linear velocity of the vehicle, and $\omega = \dot{\theta}$ represents the angular velocity of the vehicle. From the above model, (v, w) can be seen as control inputs to vary the position (x, y, θ) of the vehicle. Well, this is not intuitive - especially to a controls person because in real-world scenarios the mass and inertia effects prevent the system from generating instantaneous (v, ω). This is the reason why a dynamic model is essential to design a practical control law, and a direct consequence of this phenomenon will be discussed in Chapter 6.

The linear and angular velocities of the DDV can be related to the wheel angular

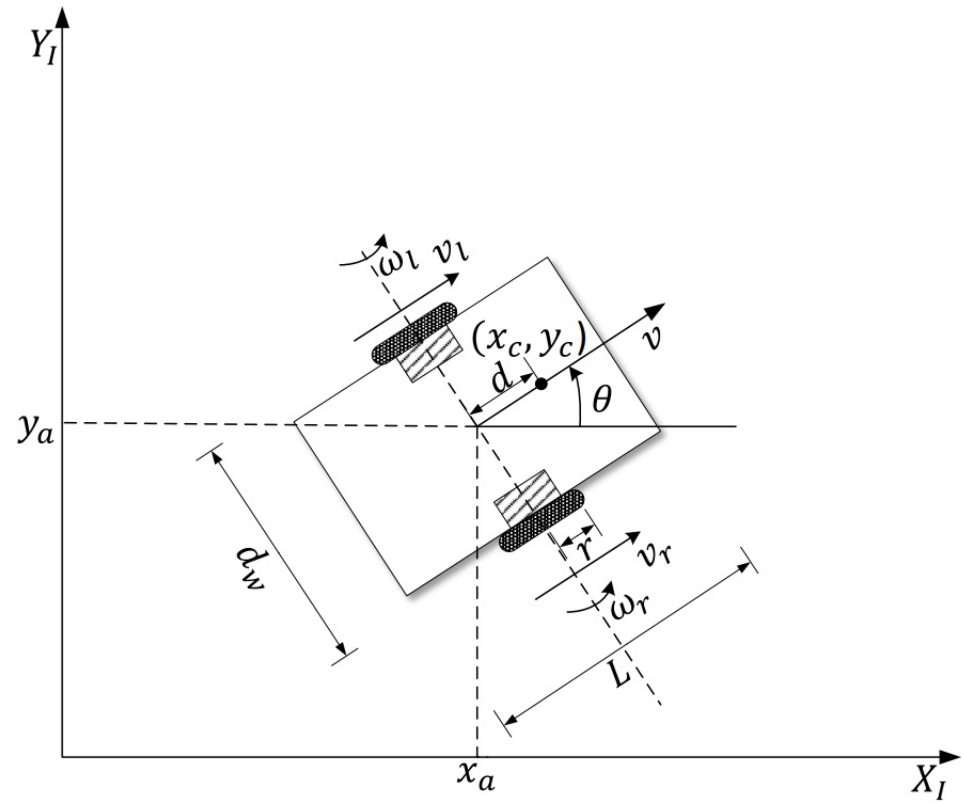


Figure 4.1: DDV Visualization

velocities as shown in equations (4.3),(4.4). A detailed derivation of this relation is presented in [46].

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = M \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \tag{4.3}$$

$$M = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{d_w} & -\frac{r}{d_w} \end{bmatrix}, \quad M^{-1} = \begin{bmatrix} \frac{1}{r} & \frac{d_w}{2r} \\ \frac{1}{r} & -\frac{d_w}{2r} \end{bmatrix} \tag{4.4}$$

4.2.2 Differential-Drive Robot Dynamics

The purpose of the dynamical model is to capture the effect of various forces acting on a system that can impact its motion. There are two widely used approaches for deriving the dynamic model of a plant or system in general - the Lagrange Dynamic Approach and the Newton-Euler Approach. In [22], the author has presented the detailed steps involved in deriving the dynamics of a DDV using both approaches. The results obtained are presented in equations (4.5) - (4.7).

$$(m + \frac{2I_w}{r^2})\dot{v} - m_c d\omega^2 = \frac{1}{r}(\tau_r + \tau_l) \tag{4.5}$$

$$(I + \frac{d_w^2 I_w}{2r^2})\dot{\omega} + m_c d\omega v = \frac{d_w}{2r}(\tau_r - \tau_l) \tag{4.6}$$

$$\begin{aligned} &\begin{bmatrix} I_w + \frac{r^2}{d_w^2}(\frac{1}{4}md_w^2 + I) & \frac{r^2}{d_w^2}(\frac{1}{4}md_w^2 - I) \\ \frac{r^2}{d_w^2}(\frac{1}{4}md_w^2 - I) & I_w + \frac{r^2}{d_w^2}(\frac{1}{4}md_w^2 + I) \end{bmatrix} \begin{bmatrix} \dot{\omega}_r \\ \dot{\omega}_l \end{bmatrix} \\ &\qquad = \begin{bmatrix} 0 & \frac{r_2}{d_w^2}m_c d\omega \\ -\frac{r^2}{d_w}m_c d\omega & 0 \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} + \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \end{aligned} \tag{4.7}$$

In the above nonlinear time-invariant equations, (τ_R, τ_L) represent the input torques acting on the right and left wheels, (v, ω) represent the linear and angular velocities of the vehicle, and (ω_r, ω_l) represent the angular velocities of the right and left wheels. Equations (4.5), (4.6) and equation (4.7) essential represent the same dynamics - substituting equation (4.3) in equations (4.5), (4.6) yields equation (4.7). The other parameters that appear in the above equations are as follows: I_w represents the moment of inertia of the wheel-motor system about the vehicle axis, r represents the radius of the wheel, d_w represents the distance between the midpoints of two wheels, m represents the mass of the entire vehicle, I represents the total moment of inertia of the vehicle including the motors and wheels, m_c represents the mass of the vehicle excluding the actuators and wheels, and d represents the distance between the center of gravity and the midpoint on the vehicle axis.

4.2.3 Actuator (DC Motor) Dynamics and Parameter Estimation

DC motors are one of the most commonly used actuators in mobile robotic applications. There are two classes of DC motors: 1) Field-current controlled; 2) Armature-current controlled [66]. In this thesis, we would be dealing with the armature-current controlled DC motor. In this particular motor class, the armature voltage (e_a) is used as a control input while maintaining the field circuit conditions to be constant. Specifically, for a permanent magnet DC motor, the dynamics can be modeled by the linear time-invariant (LTI) equations presented below. Figure 4.2 shows the block diagram representation of DC motor dynamics.

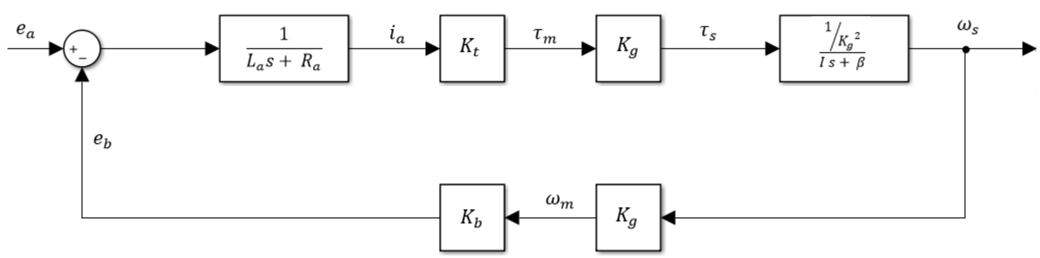


Figure 4.2: DC Motor Speed - Voltage Dynamical Model

Armature Equation:

$$e_{a_{r,l}} = L_a \frac{di_{a_{r,l}}}{dt} + R_a i_{a_{r,l}} + e_{b_{r,l}} \tag{4.8}$$

Back EMF Equation:

$$e_{b_{r,l}} = K_b K_g \omega_{r,l} \tag{4.9}$$

Torque Equation:

$$\tau_{b_{r,l}} = K_t i_{a_{r,l}} \tag{4.10}$$

Load Equation:

$$\tau_{r,l} = K_g \tau_{b_{r,l}} - K_g^2 \beta \omega_{r,l} \tag{4.11}$$

here, $e_{a_{r,l}}$ represent the input voltages applied to the right and left motors, and τ_r, τ_l represent the output torque generated by the corresponding DC motors. The following are the other parameters presented in the above equation: $\omega_{r,l}$ and $e_{b_{r,l}}$ represent the wheel angular velocities and back emf of right and left motors respectively, K_b represents the back emf constant, K_t represents the motor torque constant, β represents the damping constant, R_a represents the armature resistance, L_a represents the armature inductance, $i_{a_{r,l}}$ represents the armature current.

Based on the above equations, the motor transfer function from input voltage e_a to the angular speed ω can be represented as follows:

$$\frac{\omega_{r,l}}{e_{a_{r,l}}} = \left[\frac{\frac{K_t}{K_g}}{(I_w s + \beta)(L_a s + R_a) + K_t K_b} \right] \tag{4.12}$$

Equation (4.12) is a second-order transfer function. Furthermore, we can assume that the armature inductance L_a is negligibly small (since $L_a/R_a << 1$), in which case, the motor transfer function presented in equation (4.12) can be approximated as a first order system as shown below

$$\frac{\omega_{r,l}}{e_{a_{r,l}}} \approx \left[\frac{\frac{K_t}{K_g R_a I_w}}{s + \frac{R_a \beta + K_t K_b}{R_a I_w}} \right] \tag{4.13}$$

Given the above, we arrive at the following equations for the motor dominant pole and DC gain

$$\text{Motor DC Gain} = \frac{\frac{K_t}{K_g}}{R_a \beta + K_t K_b} \tag{4.14}$$

$$\text{Motor Dominant Pole} = \frac{R_a \beta + K_t K_b}{R_a I_w} \tag{4.15}$$

From equation (4.15) it can be inferred that the bandwidth of the motor is dependent on the parameters $\beta, K_t, K_b, I_w,$ and R_a i.e. the motor response is faster for larger β, K_t, and K_b values, and for smaller I_w, and R_a values. Although this inference might seem interesting, the parameters cannot be varied in an actual motor since they are fixed during manufacturing. Therefore, the above understanding can be used as a guide while selecting/purchasing the motors. The motor parameters mentioned in Table 4.3 are estimated by iterating between the model-based simulations and experimental step response data collected at different input voltages and operation speeds. Column two (DDV - 150 RPM) corresponds to an older DDV that is equipped with 150 RPM motors, while columns three and four correspond to the new motors presented in Chapter 2. A detailed description of the procedure involved in estimating these values has been presented in [45] - section 3.5.2.

One major drawback of the first-order model in equation (4.13) is that it does not include the effect of static friction and battery internal resistance. Under no-load/off-ground case, the DC motor has a linear behavior throughout the operating point, however, under load/on-ground case, the nonlinearities due to static friction (causes an increase in armature current at low voltage) and battery internal resistance (responsible for torque saturation at high voltage) dominate. A simple method to model the behavior of the motor under load is to estimate the parameters locally using piece-wise linear first-order models that fit locally. Please refer to section 3.7 in [45] to get a thorough insight on dealing with DC motor nonlinearities under load. A small scale chassis dynamo-meter setup has been built (Figure 4.3) to estimate the parameters of the motor under load that are accurate within the local operation points (Input: 0 V - 6 V; Output: 0 rad/s - 51 rad/sec).

Par	**150 RPM Motor (With Load)**	**5,300 RPM Motor (With Load)**	**5,300 RPM Motor (No Load)**
I_w	1.67×10^{-6} kg m^2	573×10^{-6} kg m^2	29×10^{-6} kg m^2
L_a	1.729×10^{-6} H	13.2×10^{-6} H	13.2×10^{-6} H
R_a	3.01 Ω	0.8 Ω	0.8 Ω
K_b	9.5×10^{-3} V/rad/s	0.201 V/rad/s	0.0183 V/rad/s
K_t	9.5×10^{-3} Nm/A	0.201 Nm/A	0.0183 Nm/A
K_g	50	1	1
β	3.29×10^{-6} Nms	7.4×10^{-6} Nms	110×10^{-6} Nms

Table 4.3: DC Motor Parameter Values

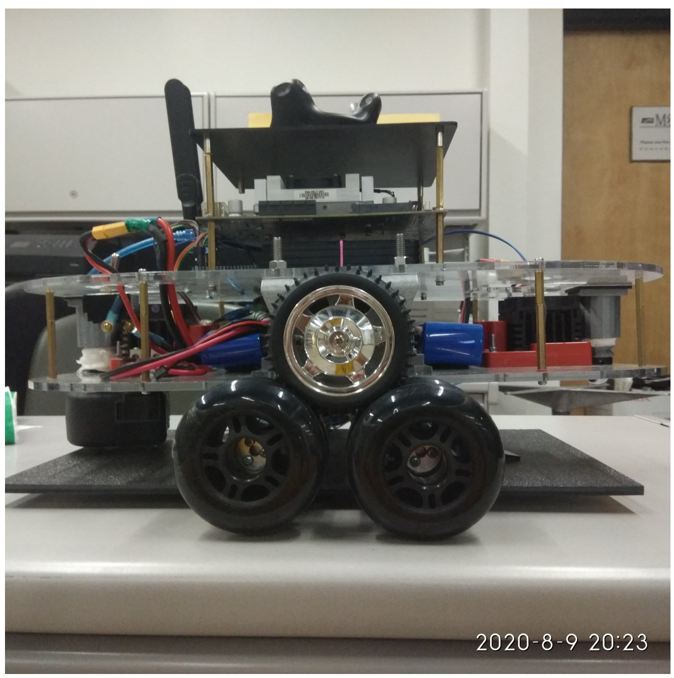


Figure 4.3: Small Scale Chassis Dynamo-meter Setup

4.2.4 TITO LTI Model with Actuator Dynamics

In this section, we combine the dynamic model of the DDV discussed in Section 4.2.2 with the actuator dynamics presented in the previous section to form the complete nonlinear dynamics representation of the DDV. Further to this, we derive the state space representation of the TITO LTI model of the DDV including the actuator dynamics. Though this state representation is first presented in [46], [45], we have made minor corrections to the A and B matrices. This TITO LTI model will be used as the basis for controller design and parameter trade studies in the frequency domain presented in the forthcoming sections. Figure 4.5 represents the block diagram of the nonlinear dynamical model from motor input voltages to wheel angular velocities, and Figure 4.4 represents the DDV dynamics excluding the actuator dynamics.

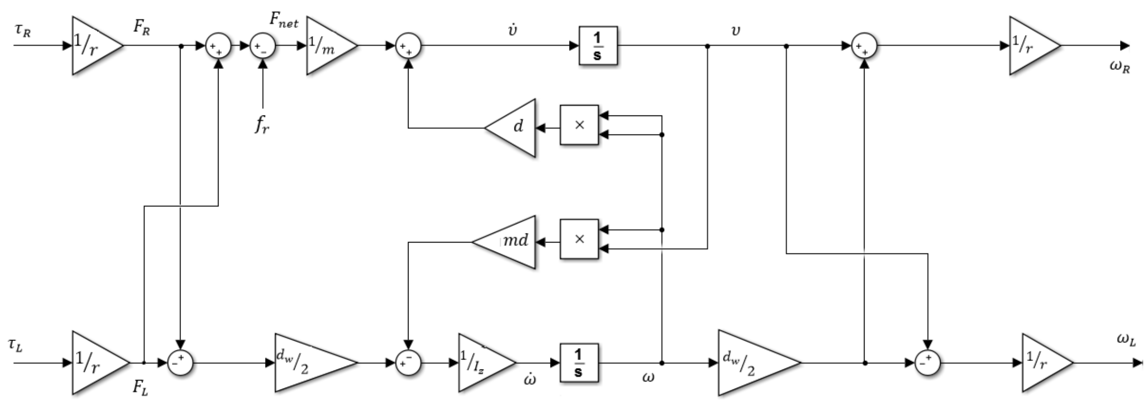


Figure 4.4: DDV Dynamics $(\tau_r, \tau_l) \rightarrow (\omega_r, \omega_l)$

From Figure 4.4 it can be clearly seen that there is coupling introduced due to the motor torques (τ_r, τ_l) at the input and also due to the intermediate linear and angular velocities (v, ω) outputs. The associated fourth-order state-space representation of the TITO LTI model is given by

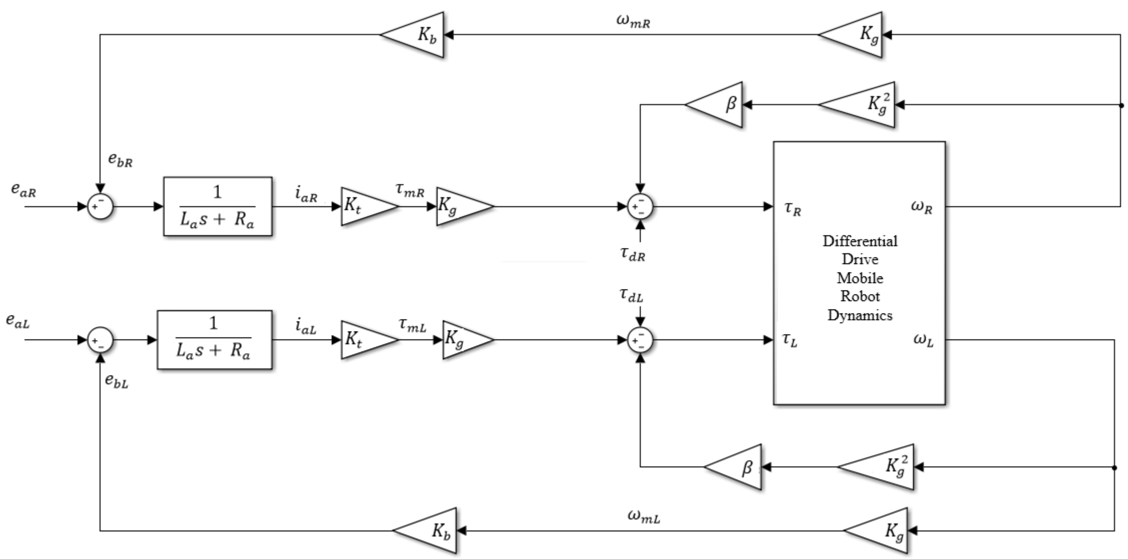


Figure 4.5: DDV Dynamics $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$

$$\dot{x} = Ax + Bu \qquad y = Cx + Du \tag{4.16}$$

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

$$A = \begin{bmatrix} \frac{-2\beta K_g^2}{r^2 A} & \frac{2m_c d\omega_{eq}}{A} & \frac{K_t K_g}{rA} & \frac{K_t K_g}{rA} \\ \frac{-m_c d\omega_{eq}}{B} & -(\frac{m_c d v_{eq}}{B} + \frac{\beta d_w^2 K_g^2}{2r^2 B}) & \frac{K_t d_w}{2rB} & \frac{-K_t d_w}{2rB} \\ \frac{-K_g K_b}{L_a r} & \frac{-K_g K_b d_w}{2L_a r} & \frac{-R_a}{L_a} & 0 \\ \frac{-K_g K_b}{L_a r} & \frac{K_g K_b d_w}{2L_a r} & 0 & \frac{R_a}{L_a} \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{L_a} & 0 \\ 0 & \frac{1}{L_a} \end{bmatrix} \tag{4.17}$$

$$C = \begin{bmatrix} \frac{1}{r} & \frac{d_w}{2r} & 0 & 0 \\ \frac{1}{r} & \frac{-d_w}{2r} & 0 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.18}$$

where $A = (m + \frac{2I_w}{r^2})$ and $B = (I + \frac{d_w^2 I_w}{2r^2})$, v represents the linear velocity of the vehicle, ω represents the angular velocity of the vehicle, (i_{a_r}, i_{a_l}) represent the armature current of the right and left motors respectively and (e_{a_r}, e_{a_l}) represent the input voltages to the right and left motors respectively. The latter being the control inputs to the DDV. Additional system parameters shown in equations (4.17), (4.18) are as follows: β represents the damping constant of the motor, K_g represents the motor gear ratio, K_b represents the back emf constant, K_t represents the torque constant of the vehicle, R represents the radius of the wheel, m_c represents the mass of the vehicle excluding the motors and wheels, d represents the distance between the center of gravity and the midpoint of the DDV wheel axis, d_w represents the distance between the midpoints of two wheels, (v_{eq}, ω_{eq}) represent the equilibrium linear and angular velocities at which the TITIO model has been linearized, R_a represents the resistance of the motor armature winding, and L_a represents the inductance of the winding (often negligibly small). Please note that during the derivation of this state-space representation, both the motors are assumed to have the same internal parameters: β, K_b, K_t, R_a, L_a. Though this is a widely used assumption throughout the literature, its implications will be considered in future work.

4.3 Design Trade Studies

In this section, we will introduce the three different variations of the TITO LTI system - presented in section 4.2.4 - based on the input and output parameters: 1) $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$; 2) $(e_{a_r}, e_{a_l}) \rightarrow (v, \omega)$; 3) $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$. Further to this, we will address the reasons for considering these different models and provide insights into the input-output coupling and controller design aspects. In addition to this, we will present the frequency domain analysis of these models for different design parameter variations.

4.3.1 TITO LTI Model - Input/Output Variations and Analysis

Lets consider the notations $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$, $P_{[e_{a_r}, e_{a_l}] \rightarrow [v, \omega]}$ and $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \rightarrow [v, \omega]}$ to represent the input to output transfer functions of the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$, (e_{a_r}, e_{a_l})

$\rightarrow(v,\omega)$ and $(e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}) \rightarrow (v,\omega)$ LTI systems. With these notations in mind, we can establish the relationship between them as follows:

$$
\begin{aligned}
P_{[e_{a_r},e_{a_l}]\rightarrow[v,\omega]} &= MP_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]} && (4.19)\\
P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[v,\omega]} &= MP_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}E && (4.20)
\end{aligned}
$$

$$
M = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{d_w} & -\frac{r}{d_w} \end{bmatrix}, \qquad E = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{2}{2} \end{bmatrix} \tag{4.21}
$$

The above transfer functions relationship can be extended to the state-space representations as well. The fourth-order state-space representation of the $(e_{a_r}, e_{a_l}) \rightarrow (v,\omega)$ system is given by

$$
\dot{x} = Ax + Bu \qquad y = MCx + Du \tag{4.22}
$$

where $x = [v\ \omega\ i_{a_r}\ i_{a_l}]^T$, $u = [e_{a_r}\ e_{a_l}]^T$ and $y = [v\ \omega]^T$. The corresponding representation for the $(e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}) \rightarrow (v,\omega)$ system is given by

$$
\dot{x} = Ax + BEu \qquad y = MCx + Du \tag{4.23}
$$

where $x = [v\ \omega\ i_{a_r}\ i_{a_l}]^T$, $u = [e_{a_r}+e_{a_l}\ e_{a_r}-e_{a_l}]^T$ and $y = [v\ \omega]^T$. Please note that the A, B, C and D matrices correspond to those in equations (4.17), (4.18).

Before we proceed with discussing the coupling and control issues associated with these three systems, it is important that we make the following observation in order to guide the discussion.

1. From equation (4.7) and Figure 4.5, it can be inferred that the dynamics of a DDV are nonlinear in nature and naturally there exists coupling between the input wheel torques and the output of the system (irrespective of what we consider as output i.e. (v,ω) or (ω_r,ω_l).

2. However, under specific conditions, the dynamical models presented in equations (4.5) - (4.7) exhibits the special properties as shown here:

   - At d = 0 i.e. when the center of gravity of the DDV coincides with the midpoint of its wheel axis, the dynamical model becomes linear in nature. This also means that the state-space representation of the TITO LTI model becomes independent of the (v_{eq}, ω_{eq}) terms. At $d = 0$ we have

$$\begin{bmatrix} \dot{\omega}_r \\ \dot{\omega}_l \end{bmatrix} = \begin{bmatrix} \frac{1}{c_1 r^2} + \frac{d_w^2}{4c_2 r^2} & \frac{1}{c_1 r^2} - \frac{d_w^2}{4c_2 r^2} \\ \frac{1}{c_1 r^2} - \frac{d_w^2}{4c_2 r^2} & \frac{1}{c_1 r^2} + \frac{d_w^2}{4c_2 r^2} \end{bmatrix} \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \tag{4.24}$$

$$\begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{1}{c_1 r} & \frac{1}{c_1 r} \\ \frac{d_w}{c_2 r} & -\frac{d_w}{c_2 r} \end{bmatrix} \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \tag{4.25}$$

   where $c_1 = (m + \frac{2I_w}{r^2})$ and $c_2 = (I + \frac{d_w^2 I_w}{2r^2})$. We consider the motor output torques (τ_r, τ_l) as the input to the system instead of motor voltages (e_{a_r}, e_{a_l}) because this would simplify the calculations involved. Moreover, the results obtained based on the input torque system will be valid for the input voltage system as well.

   - In order to further examine the models is equations (4.24), (4.25), lets adopt the following notation

$$\begin{bmatrix} \dot{\omega}_r \\ \dot{\omega}_l \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \tag{4.26}$$

$$\begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \tag{4.27}$$

   where $P_{11} = P_{22} = \frac{1}{c_1 r^2} + \frac{d_w^2}{4c_2 r^2}$, $P_{12} = P_{21} = \frac{1}{c_1 r^2} - \frac{d_w^2}{4c_2 r^2}$, $Q_{11} = Q_{12} = \frac{1}{c_1 r}$, $Q_{21} = -Q_{22} = \frac{d_w}{c_2 r}$. From equation (4.26) it can be seen that the matrix is symmetric i.e. same diagonal and off-diagonal elements, and there exists coupling between the inputs (τ_r, τ_l) and the outputs (ω_r, ω_l). In order to decouple the system, the off-diagonal element (P_{12}, P_{21}) should become zero. Hence, equating the terms (P_{12}, P_{21}) to zero yields the following condition.

$$I_{decoupling} = m(\frac{d_w}{2})^2 \tag{4.28}$$

The consequence of this result is that

<u>if we can design a DDV with $d = 0$ and $I = I_{decoupling}$, the resulting $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ system is linear and decoupled across all input-output combinations</u>.

Further, if we assume that the mass of the vehicle is uniformly distributed and the moment of inertia of the wheel motor combination is negligible ($I_w \approx 0$), we can further simplify the $I_{decoupling}$ condition to obtain the following

$$AR_{decoupling} = \frac{l}{d_w} = \sqrt{2}\sqrt{1 - 6(\frac{I_w}{m_c d_w^2})} \approx \sqrt{2} \tag{4.29}$$

here l represents the length of the vehicle and d_w represents the width of the vehicle (width of the vehicle is also assumed to be equal to the distance between the wheel midpoints). This condition gives us the ratio of length to the width that has to considered while designing a DDV in order to obtain a linear and decoupled dynamical system. An aspect ration of $\sqrt{2}$ is quite intuitive and can be observed in most of the current on-road vehicles. Please note that the AR condition has been first addressed in [4], and minor corrections have been made to that result before reintroducing it in this thesis.

- Now that we have derived the $I_{decoupling}$ condition that would decouple the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ system in equation (4.24), can a similar result be obtained for the $(e_{a_r}, e_{a_l}) \rightarrow (v, \omega)$ system shown in equation (4.25)? This will not be possible because from equation (4.27), we know that $Q_{11} = Q_{12}$ and $Q_{21} = -Q_{22}$. So, any attempt to constrain the design parameters such as I,d_w,m or R in order to make the off-diagonal elements to zero will effect the diagonal elements as well.

From the aforementioned paragraph, it is clear that $(e_{a_r}, e_{a_l}) \rightarrow (v, \omega)$ cannot be decoupled by constraining the design parameters, nevertheless, let us consider the following linear transformation on the plant input

$$\begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} \tau_r + \tau_l \\ \tau_r - \tau_l \end{bmatrix} = E \begin{bmatrix} \tau_r + \tau_l \\ \tau_r - \tau_l \end{bmatrix} \tag{4.30}$$

here E is a non-singular matrix. Substituting this in equation (4.25) yields the following relation

$$\begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{1}{c_1 r} & 0 \\ 0 & -\frac{d_w}{c_2 r} \end{bmatrix} \begin{bmatrix} \tau_r + \tau_l \\ \tau_r - \tau_l \end{bmatrix} \tag{4.31}$$

Equation (4.31) suggests that if we consider the sum and difference of torques as inputs to the (v, ω) system, the resultant system remains decoupled for all operating points (v_{eq}, ω_{eq}), irrespective of the variations in the design parameters. Since the output motor torque is a linear function of the applied input voltage [2] , this result can be extended to the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ system as well. The consequence of this result that

[2] more specifically, output motor torque is a function of input voltage and angular velocity of the wheel as shown in Figure 4.5, but in most practical cases, the damping constant is very small making the impact of wheel angular velocity negligible

for $d = 0$, if we consider the sum and difference of voltages as inputs to the (v, ω) system, the resulting $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ system is decoupled at all operating points (v_{eq}, ω_{eq}) - irrespective of any variations in design parameters such as I, d_w, m or r.

As a result of these decoupling properties, it is desirable to consider the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ system over the $(e_{a_r}, e_{a_l}) \rightarrow (v, \omega)$ system.

Based on the above discussion, it can be justified to classify the DDV models into two categories - $d = 0$ and $d \neq 0$ - and perform design trade studies to understand their impact on input-output coupling and bandwidth of the system.

4.3.2 Trade Studies for $d = 0$

As mentioned earlier, at $d = 0$ the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ systems become independent of the operating points (v_{eq}, ω_{eq}). Assuming that the motor parameters are fixed and since $d = 0$, we can consider the moment of inertia I, the mass of the vehicle m, and the radius of the wheel r as the critical design parameters.

Variation in Moment of Inertia I. The following bode magnitude and singular value plots will enable us to understand the variation in dynamic properties of the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ systems as the parameter I is varied.

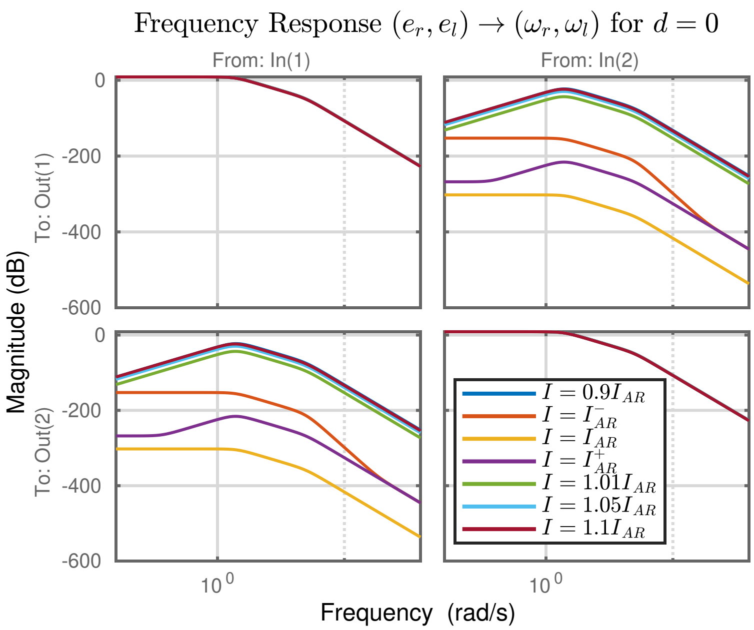


Figure 4.6: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ System: Varying I

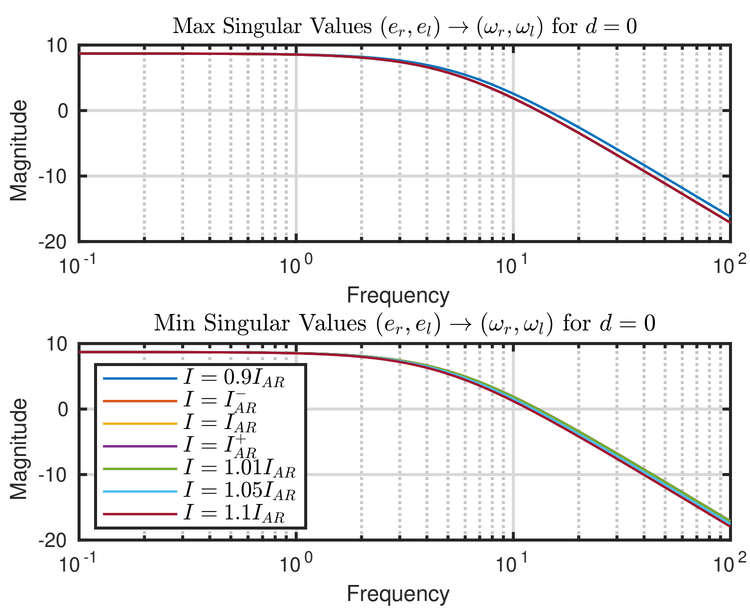


Figure 4.7: Singular Value Response of $(e_{a_r}, e_{a_l}) \to (\omega_r, \omega_l)$ System: Varying I

From the frequency response plots presented above, the following observations can

be made for the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ plant:

- The system remains decoupled when the moment of inertia I is equal to I_{AR} [3]
- At dc there is a little coupling between the inputs and outputs, but as the value of I is varied beyond the I_{AR}, there is a peak in the off-diagonal elements at 11 rad/sec
- It is also observed that the diagonal elements show a negligible amount of variation with the changes in the moment of inertia

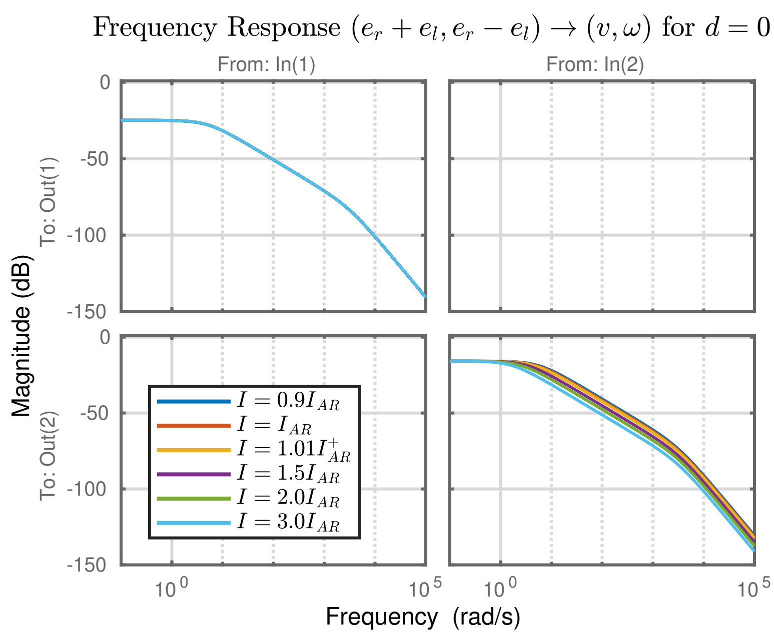


Figure 4.8: Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \rightarrow [v,\omega]}$ System: Varying I

[3] $I_{AR} = I_{decoupling}$

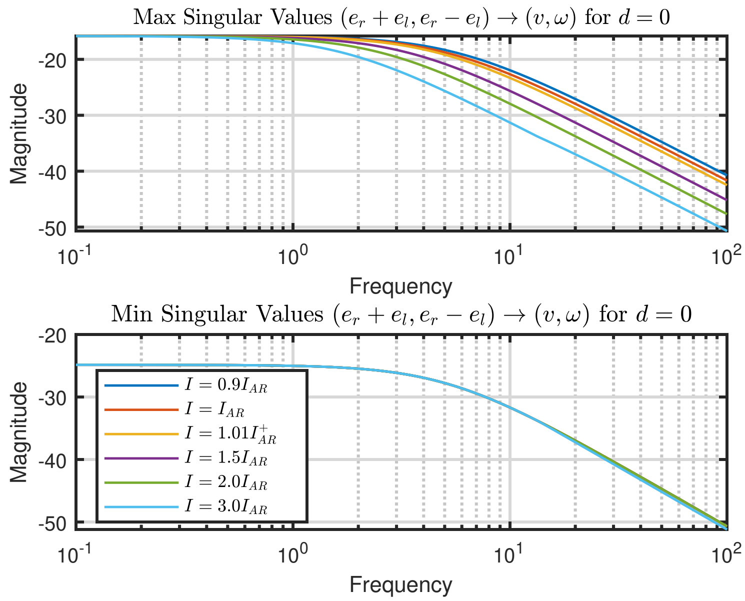


Figure 4.9: Singular Value Response of $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \to (v, \omega)$ System: Varying I

From the frequency response plots mentioned above, the following observations can be made for the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \to (v, \omega)$ plant:

- The system is decoupled for all possible variations of I
- It can be see that the input-output frequency response is unsymmetric, and the minimum singular value at low frequencies is associated with the output velocity v channel
- The response from the $(e_{a_r} + e_{a_l} \to v)$ channel is unperturbed by the variations in I, however, the response from $(e_{a_r} - e_{a_l} \to \omega)$ channel becomes slightly faster as the value of I is reduced.

Variation in Mass of the Vehicle m. The mass m of the DDV is increased while keeping the mass of the motor wheel combination m_w to be the same. This variation in m is considered under the assumption that actuator characteristics remain the same in spite of an increase in the mass.

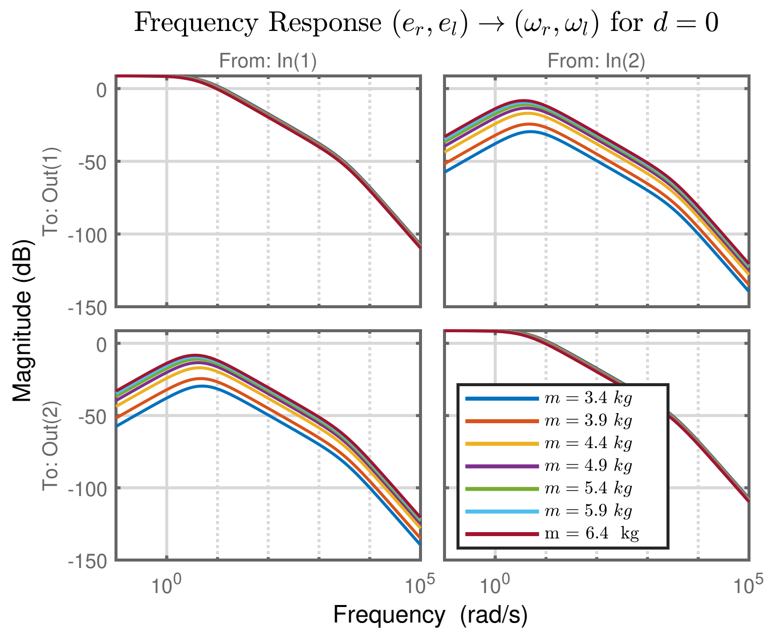


Figure 4.10: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ System: Varying m

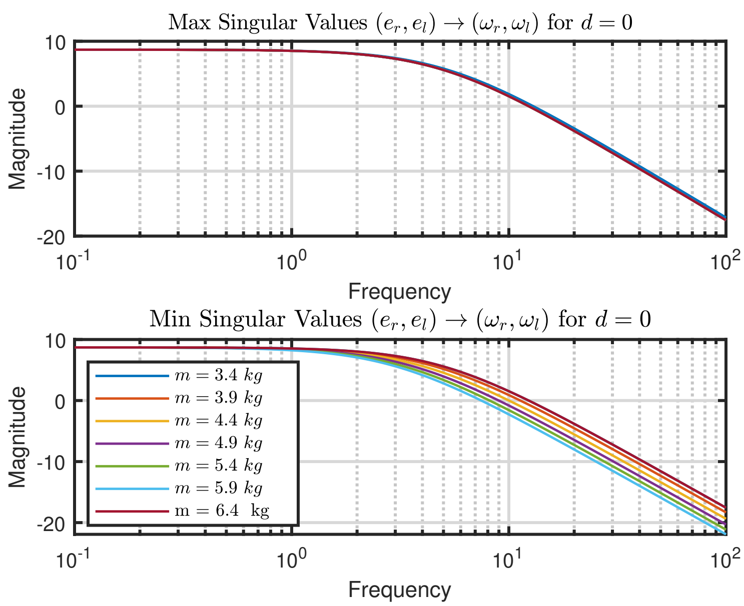


Figure 4.11: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ System: Varying m

From the frequency response plots presented above, the following observations can be made for the $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ plant:

- At dc there is a little coupling between the inputs and outputs, but with the increase in the value of m, there is an increase in the input-output coupling with the peak occurring at 3.56 rad/sec
- It can also be noticed that varying the mass of the system has a negligible impact on the response of diagonal elements

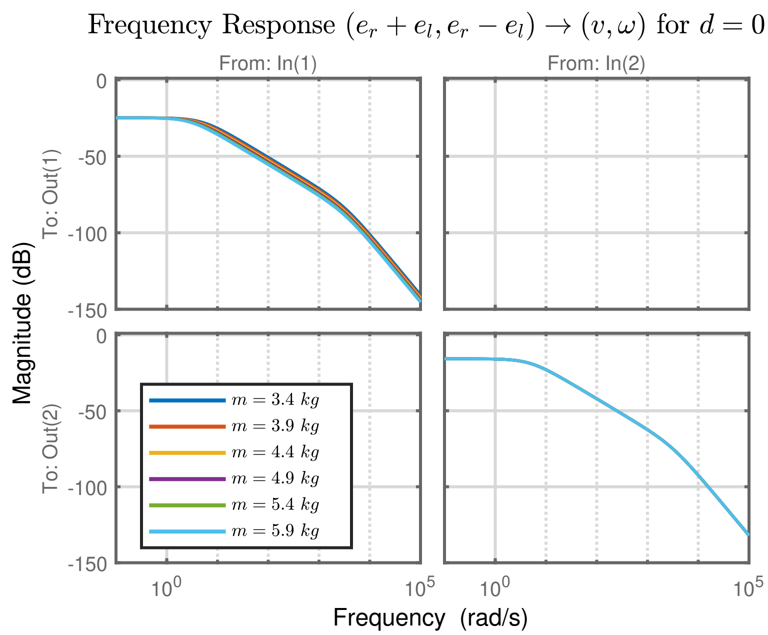


Figure 4.12: Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \to [v,\omega]}$ System: Varying m

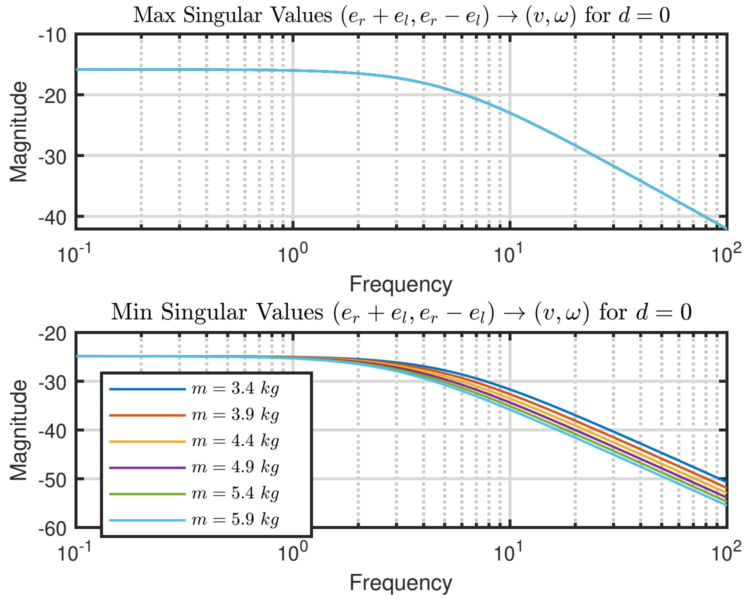


Figure 4.13: Singular Value Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}]\to[v,\omega]}$ System: Varying m

From the frequency response plots presented above, the following observations can be made for the $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}]\to[v,\omega]}$ plant:

- The major point over here is that the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \to (v, \omega)$ system remains decoupled irrespective of the variation in the total mass of the system
- The response from the $(e_{a_r} - e_{a_l} \to \omega)$ channel is unperturbed by the variations in m, however, the response from $(e_{a_r} + e_{a_l} \to \check{})$ channel becomes slightly faster as the value of m is increased

Variation in Radius of the Wheel r. Here the radius of the wheel is varied under the assumption that the actuator characteristics remain constant.

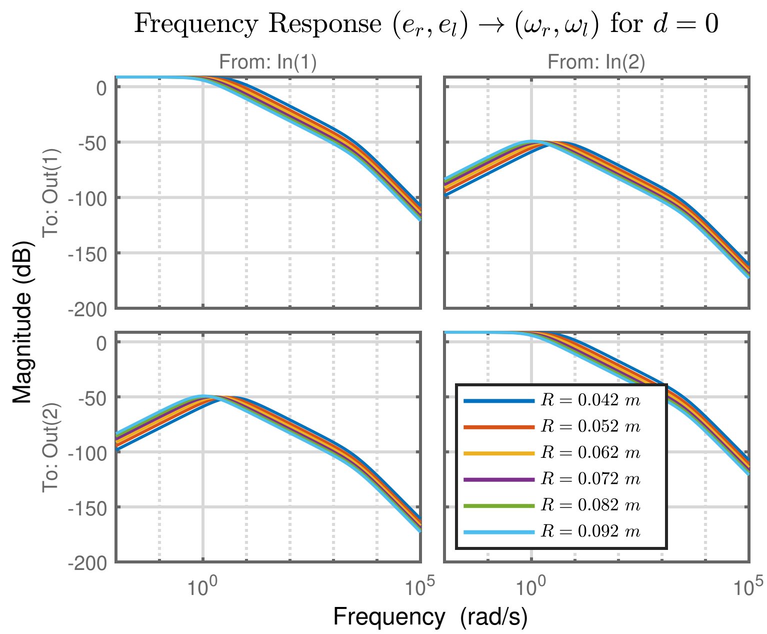


Figure 4.14: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ System: Varying r

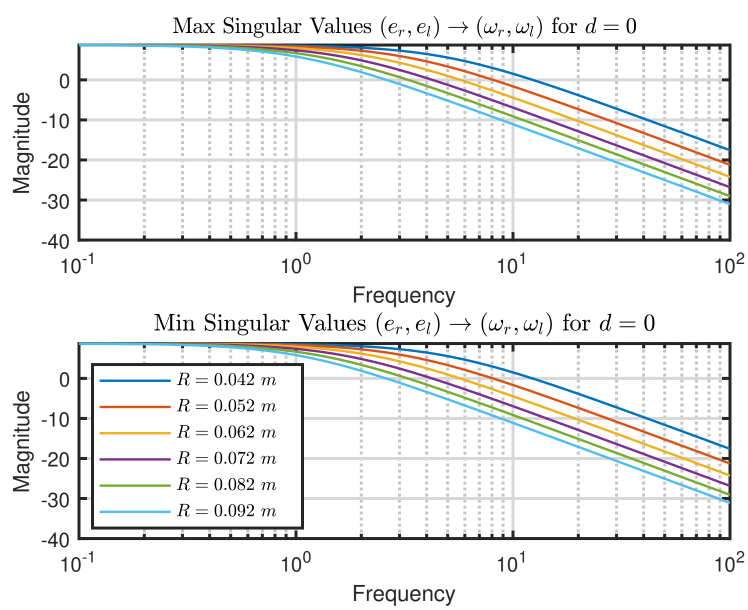


Figure 4.15: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ System: Varying r

From the frequency response plots presented above, the following observations can

be made for the $P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$ plant:

- From the response of the diagonal elements, it can be noticed that the increase in the radius of the wheel makes the response slower
- At dc it can be noticed that there is a little coupling between the inputs and outputs, but it gradually increases with frequency until reaching a peak
- Further, it can be observed that the increase in the radius of the wheel has shifted the peak from 5.27 rad/sec to 1.04 rad/sec

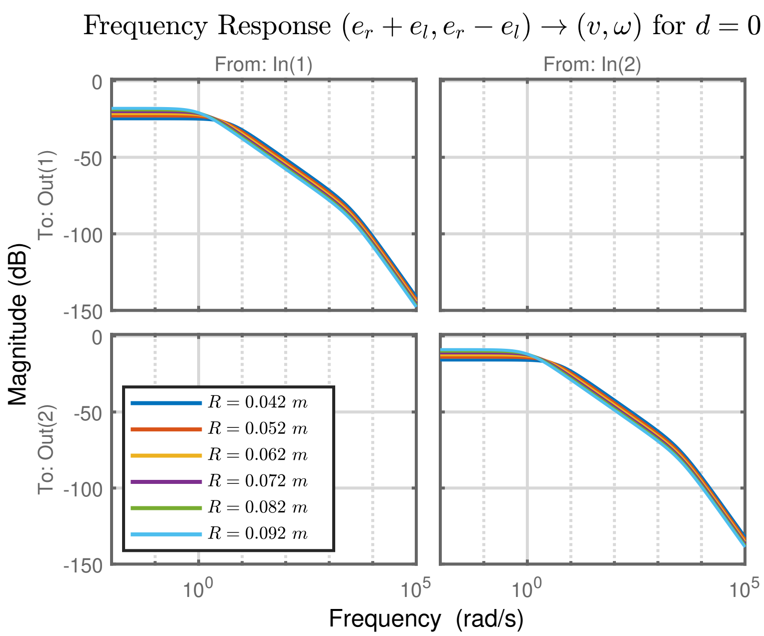


Figure 4.16: Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[v,\omega]}$ System: Varying r

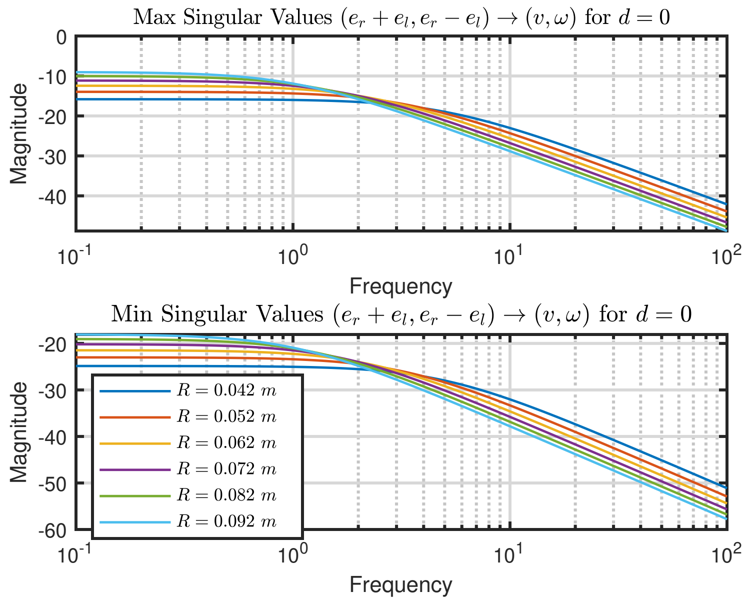


Figure 4.17: Singular Value Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}]\to[v,\omega]}$ System: Varying r

From the frequency response plots presented above, the following observations can be made for the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \to (v, \omega)$ plant:

- The main point to be noticed is that $(e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}) \to (v, \omega)$ system remains decoupled irrespective of the variations in the radius of the wheel r

- In the case of off-diagonal elements it can be noticed that an increase in the r values causes an increase in the gain while reducing the bandwidth of the system

4.3.3 Trade Studies for $d \neq 0$

For $d \neq 0$, the dynamics of $(e_{a_r}, e_{a_l}) \to (\omega_r, \omega_l)$ and $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \to (v, \omega)$ systems become nonlinear and hence the TITO LTI models are dependent on the operation points (v_{eq}, ω_{eq}). Not to mention, the models also lose their unique decoupling properties that existed when $d = 0$. Given this, we can consider v_{eq}, ω_{eq},

d, m, r and I as the critical design parameters. Please note that the effect of other parameters such as length, width, or mass of the wheel-motor combination will be captured by one or the other parameters that are mentioned earlier. The following bode magnitude and singular value plots will enable us to understand the variation in dynamic properties of the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ systems as each of these parameters are varied.

Dominant Pole Variation with d. The figure shown below represents the variation in the position of the dominant pole with respect to the variation in the value of the center of gravity d as the equilibrium velocity is increased v_{eq}. The position of m_c is shifted in order to vary the value of d. The value of d is limited to $(-2.8, +2.8)$ m since it is the maximum possible variation that is allowed within the DDV dimensions without adding additional mass. A positive value of the d indicates that the center of gravity of the vehicle is located towards the front of the vehicle from the wheel axis, and a positive v_{eq} denotes that the vehicle is moving in the forward direction.

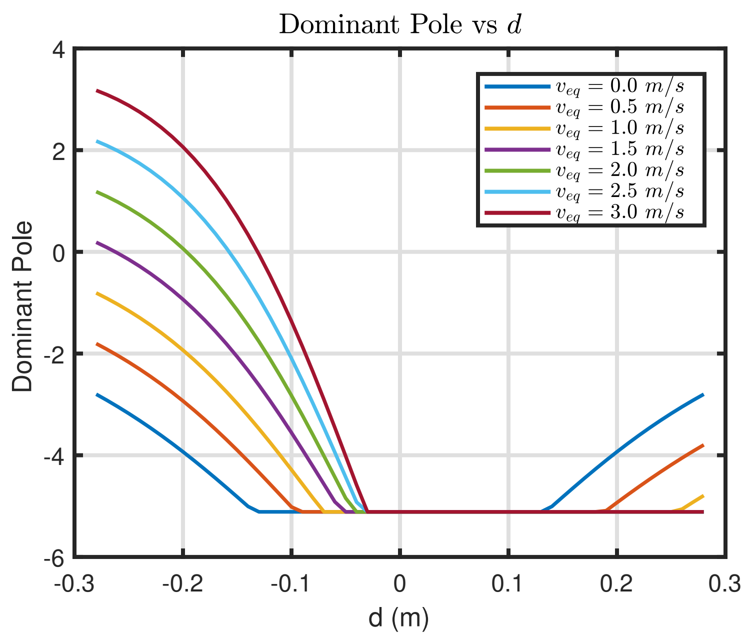


Figure 4.18: Dominant Pole Variation with d

From the above above figure, the following observations can be made:

- Irrespective of the variation in d the system remains stable as long as the v_{eq} is less than 1.5 m/s
- For $v_{eq} \geq 1.5$ m/s and $d < 0$, the system becomes unstable when the d is reduced beyond a certain value within the existing limits i.e. $-2.8 \leq d < 0$
- For $d > 0$, the system remains stable irrespective of the variations in the value of v_{eq}, and for $v_{eq} > 1$ m/s the variations in the dominant pole location is almost negligible

Variation in Equilibrium Velocity v_{eq}. The following figures show the frequency response of the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ systems for the variation in v_{eq} values at $d = 0.1$ m and $\omega_{eq} = 0.8$ rad/sec.

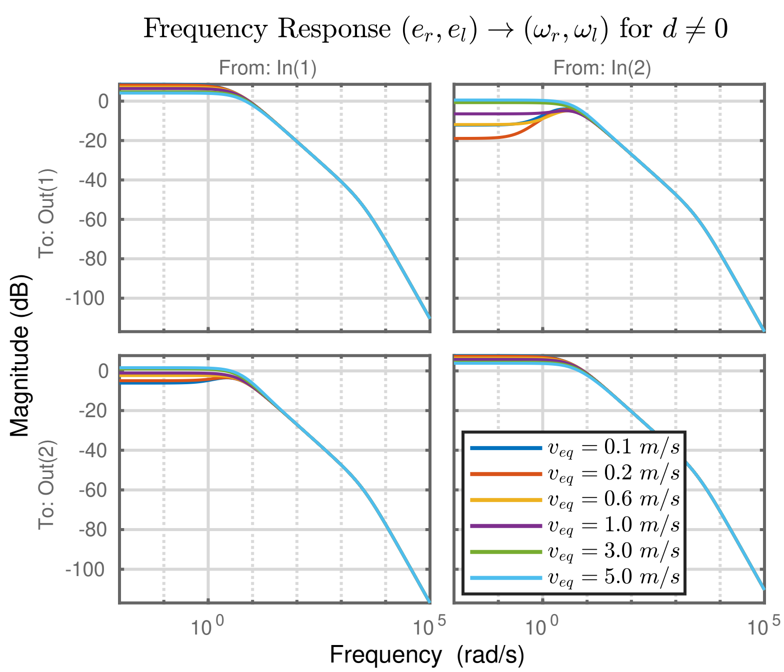


Figure 4.19: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying v_{eq}

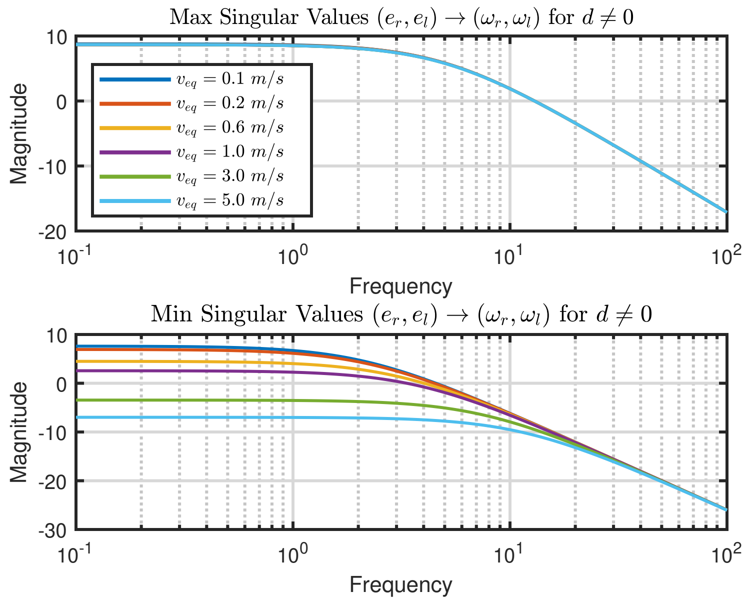


Figure 4.20: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ System: Varying v_{eq}

From the frequency response plots presented above, the following observations can be made for the $(e_{a_r}, e_{a_l}) \to (\omega_r, \omega_l)$ plant:

- There is a peak in off-diagonal element response at a frequency of 3.8 rad/sec
- There is an increase in coupling at low frequencies with an increase in the equilibrium velocity v_{eq}
- The diagonal elements are less susceptible to the variation in the equilibrium velocity v_{eq}

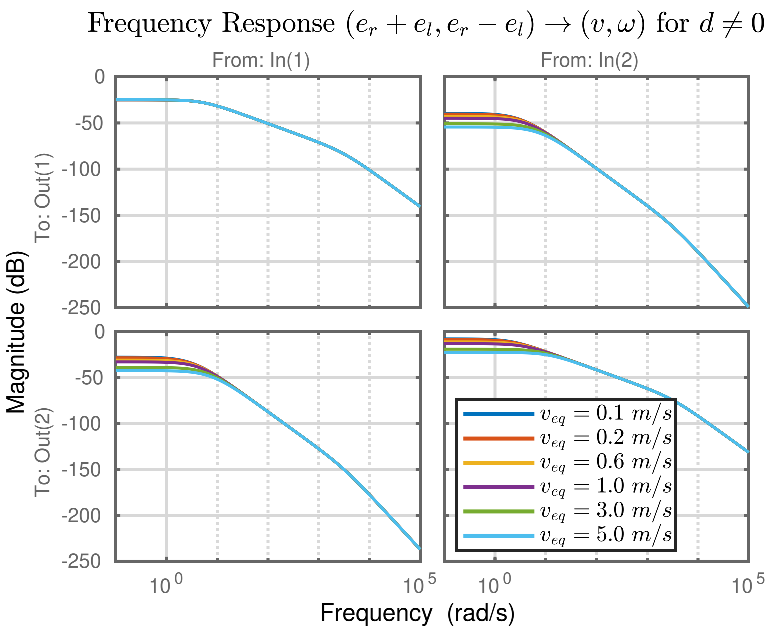


Figure 4.21: Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}]\to[v,\omega]}$ System: Varying v_{eq}

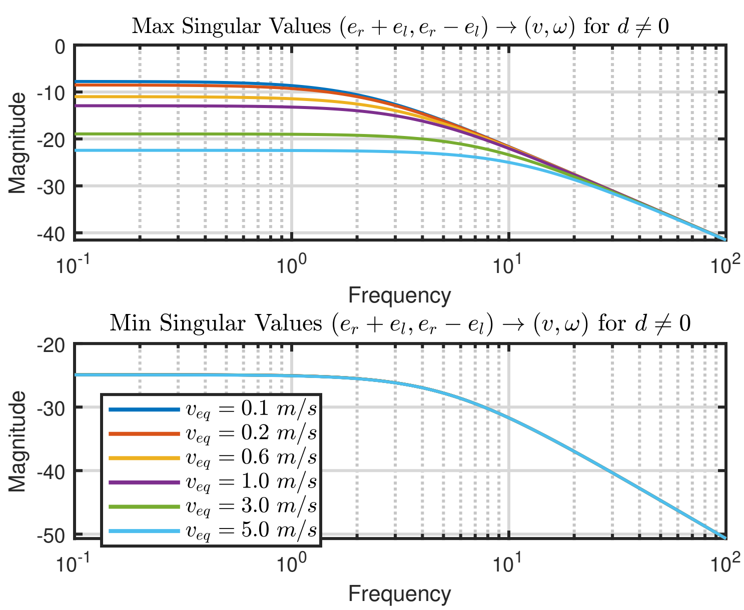


Figure 4.22: Singular Value Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}]\to[v,\omega]}$ System: Varying v_{eq}

From the frequency response plots presented above, the following observations can

be made for the $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \rightarrow [v,\omega]}$ plant:

- It can be observed that there is a decrease in coupling as the vehicle moves at higher speeds
- Unlike the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ plant, there is no peak in off-diagonal element response with an increase in the equilibrium velocity
- It can also be noticed that the response from $(e_{a_r} - e_{a_l} \rightarrow \omega)$ tends to become slower with the increase in v_{eq} while the $(e_{a_r} + e_{a_l} \rightarrow \check{\ })$ remains unperturbed

Variation in Equilibrium Angular Velocity ω_{eq}. The following figures show the frequency response of the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ systems for the variation in ω_{eq} values at $d = 0.1$ m and $v_{eq} = 0.8$ rad/sec.

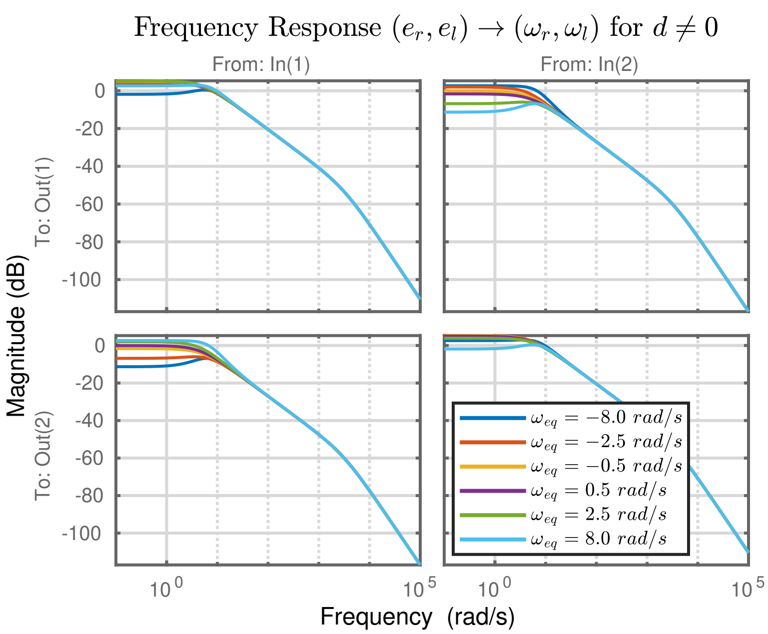


Figure 4.23: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying ω_{eq}

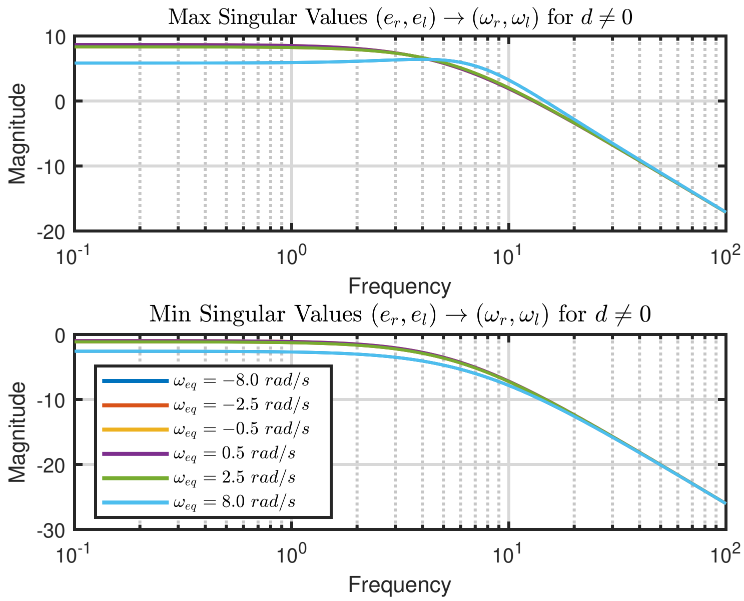


Figure 4.24: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ System: Varying ω_{eq}

From the frequency response plots presented above, the following observations can be made for the $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ plant:

- There is a peak in off-diagonal elements response at a frequency of 6 rad/sec
- There is an increase in coupling at lower frequencies with an increase in the absolute value of the equilibrium angular velocity $|\omega_{eq}|$

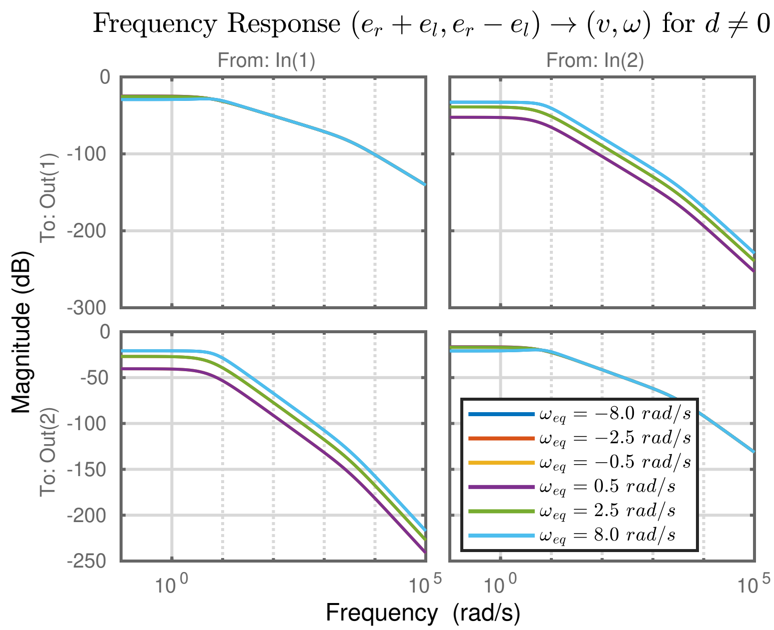


Figure 4.25: Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \rightarrow [v,\omega]}$ System: Varying ω_{eq}

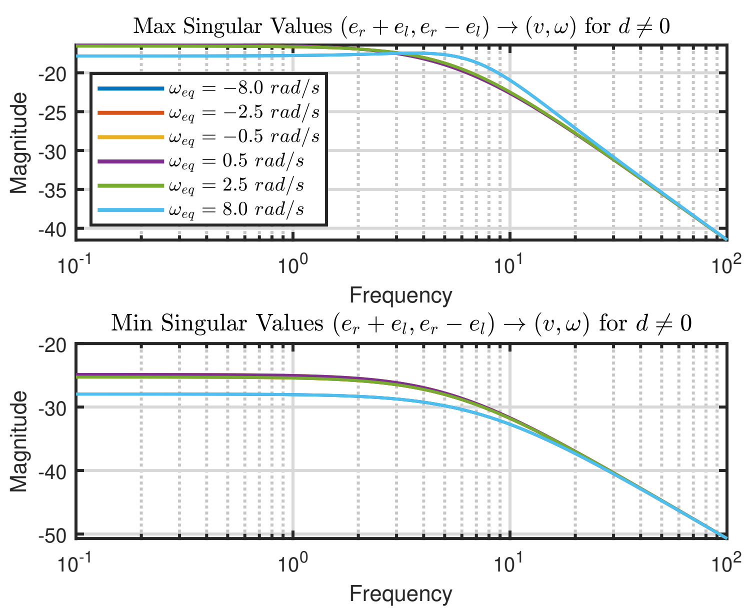


Figure 4.26: Singular Value Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \rightarrow [v,\omega]}$ System: Varying ω_{eq}

From the frequency response plots presented above, the following observations can

be made for the $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[v,\omega]}$ plant:

- There is an increase in coupling with an increase in the $|\omega_{eq}|$
- No significant change is observed in the response of the diagonal elements with the variation in ω_{eq}

Variation in Center of Gravity Location d. The following figures show the frequency response of the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ systems for the variation in d values at $v_{eq} = 2.0$ m/sec and $w_e q = 0.8$ rad/sec.

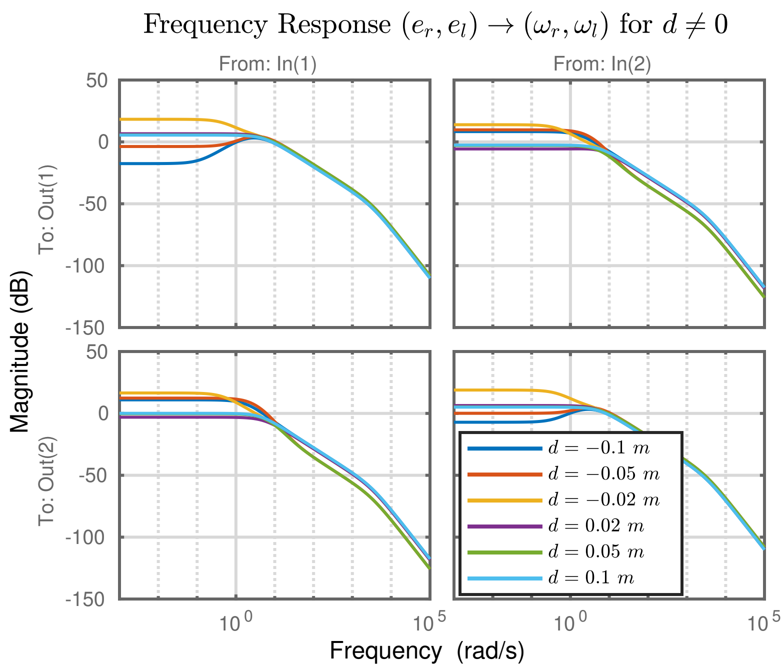


Figure 4.27: Bode Magnitude Response of $P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$ System: Varying d

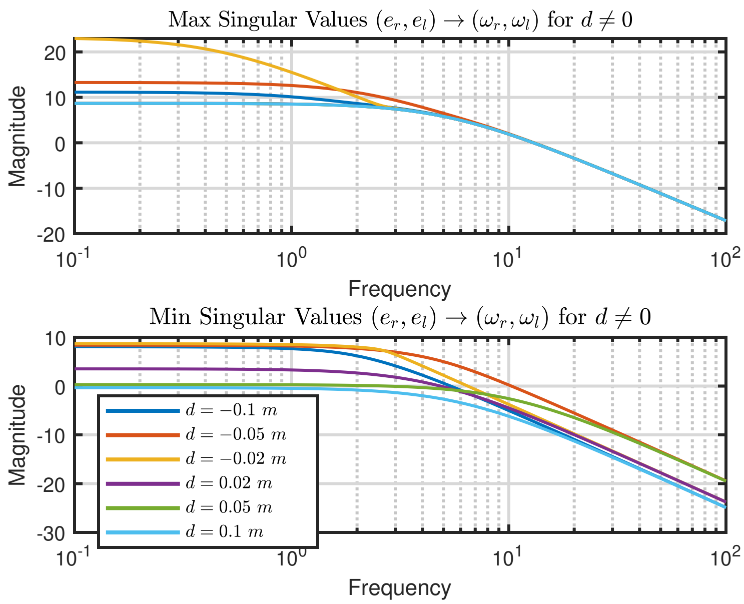


Figure 4.28: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ System: Varying d

From the frequency response plots presented above, the following observations can be made for the $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ plant:

- There is a significant increase in coupling if $dv < 0$ i.e. the velocity of the vehicle and the direction of the center of gravity from the wheel axis are in opposite directions

- There is a significant variation in case of diagonal elements if the $dv < 0$

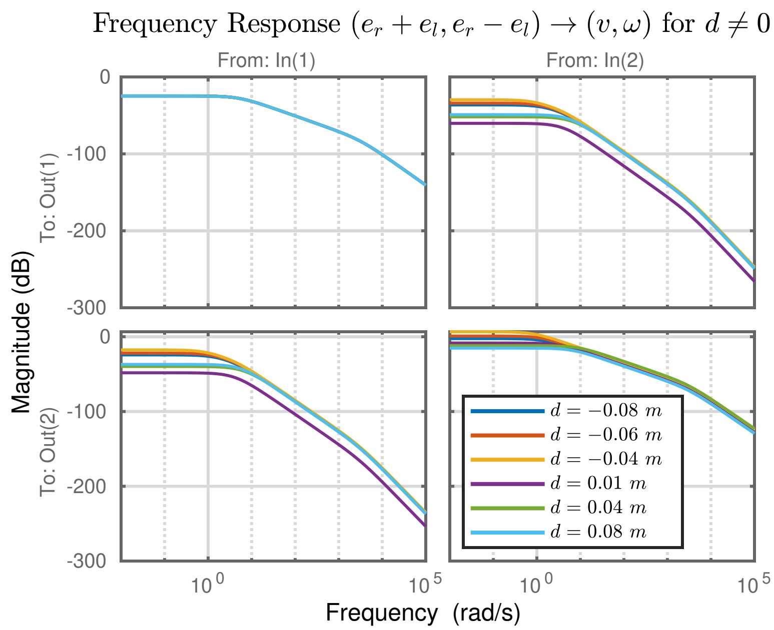


Figure 4.29: Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \to [v,\omega]}$ System: Varying d

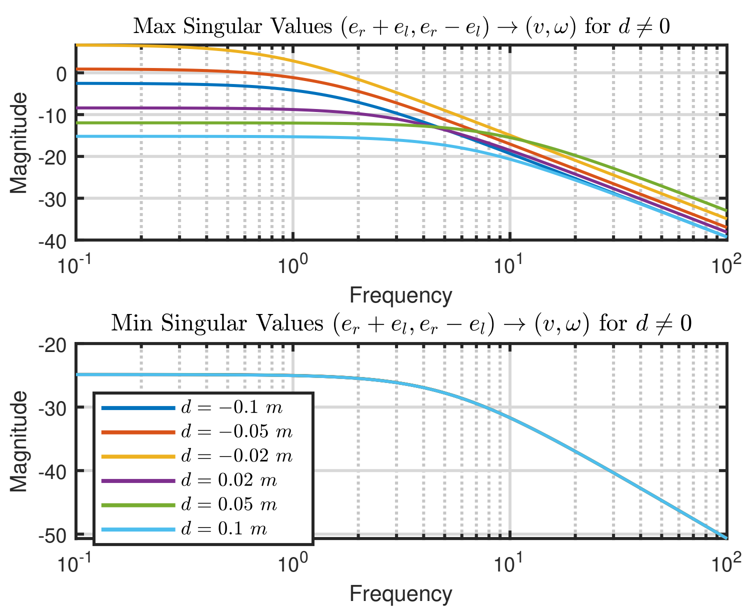


Figure 4.30: Singular Value Response of *psdv* System: Varying d

From the frequency response plots presented above, the following observations can

be made for the $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}]\to[v,\omega]}$ plant:

- There is a significant increase in coupling if $dv < 0$ i.e. the velocity of the vehicle and the direction of the center of gravity from the wheel axis are in opposite directions

- Unlike the diagonal elements in $(e_{a_r}, e_{a_l}) \to (\omega_r, \omega_l)$ plant, the diagonal elements in $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \to (v, \omega)$ are less susceptible to variations in d

- Almost negligible variation in the response from $(e_{a_r} + e_{a_l} \to \check{)}$ channel is observed with the variation in center of gravity location d

Variation in Moment of Inertia I. The following figures show the frequency response of the $(e_{a_r}, e_{a_l}) \to (\omega_r, \omega_l)$ and the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \to (v, \omega)$ systems for the variation in I values at $v_{eq} = 2.0$ m/sec, $w_eq = 0.8$ rad/sec and $d = 0.1$ m.

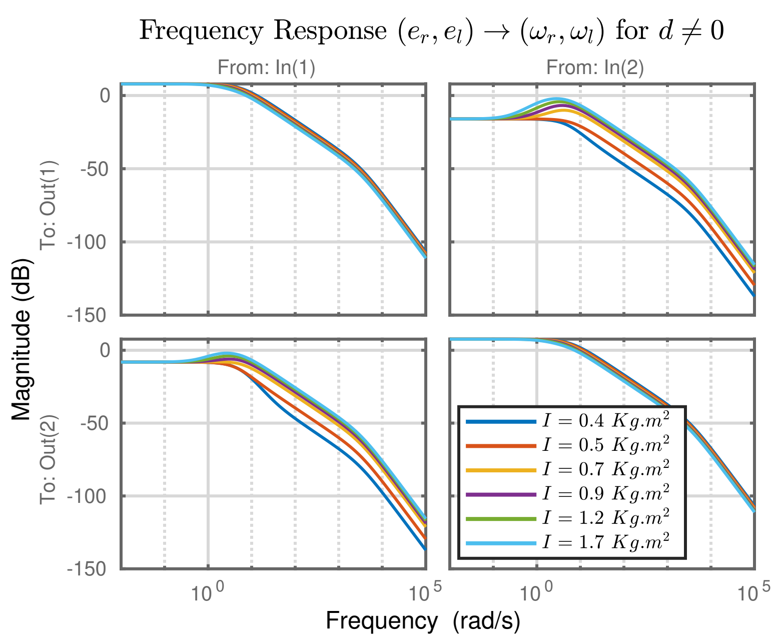


Figure 4.31: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}]\to[\omega_r, \omega_l]}$ System: Varying I

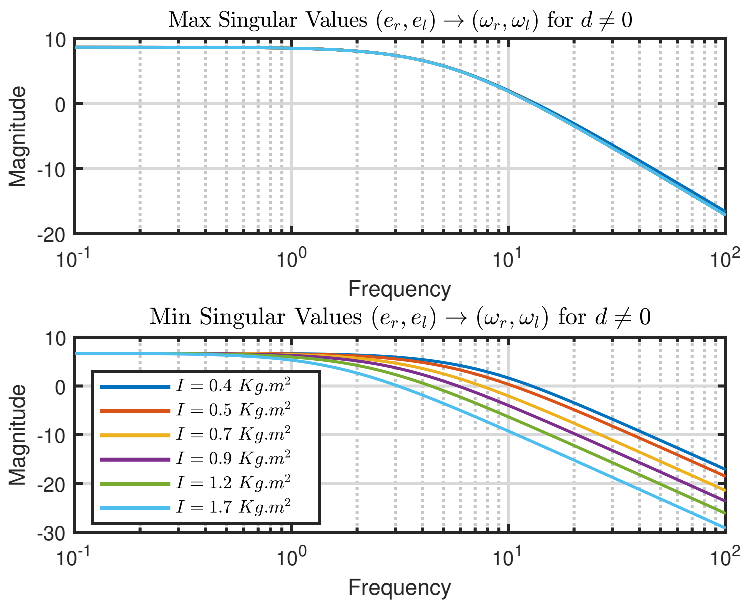


Figure 4.32: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ System: Varying I

From the frequency response plots presented above, the following observations can be made for the $(e_{a_r}, e_{a_l}) \to (\omega_r, \omega_l)$ plant:

- With the increase in the value of the moment of inertia I, there is an increase in coupling at high frequencies with a peak occurring at 4.2 rad/sec
- However, it can be observed that the diagonal elements are less susceptible to the variations in I

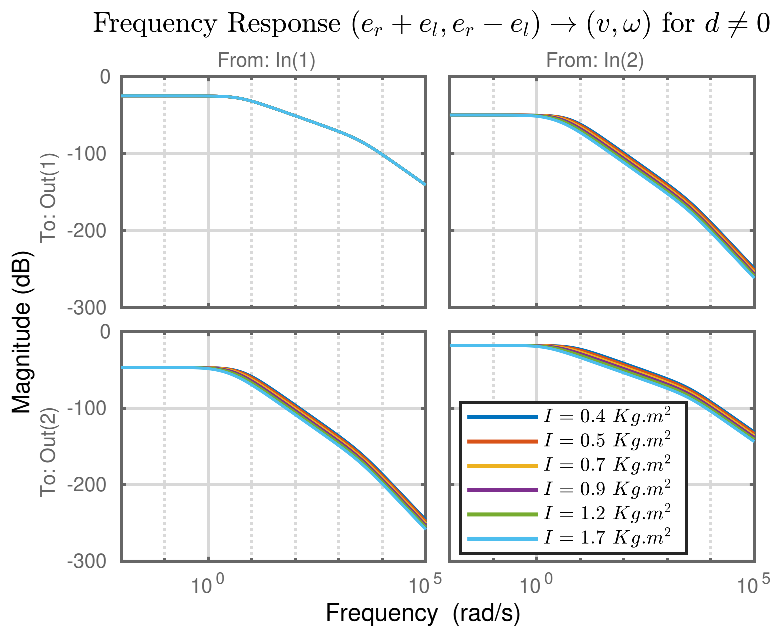


Figure 4.33: Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\to[v,\omega]}$ System: Varying I

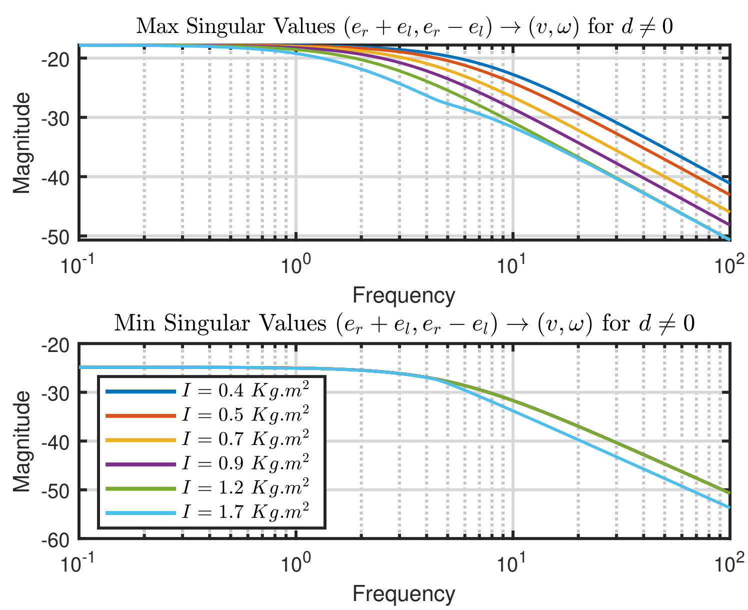


Figure 4.34: Singular Value Response of $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\to[v,\omega]}$ System: Varying I

From the frequency response plots presented above, the following observations can be made for the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ plant:

- There is a slight increase in coupling at high frequencies but this is negligible when compared to that of the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ plant.

- Almost negligible variation in the response from $(e_{a_r} + e_{a_l} \rightarrow \check{)}$ channel is observed with the variation in I, though a slight increase in the response of $(e_{a_r} - e_{a_l} \rightarrow \omega)$ channel is observed at high frequencies

Variation in Mass of the Vehicle m. The following figures show the frequency response of the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ systems for the variation in m values at $v_{eq} = 2.0$ m/sec, $w_eq = 0.8$ rad/sec and $d = 0.1$ m. Please note that m is varied by adding mass to the center of gravity location without changing the mass of motor wheel combination m_w. And also, it is assumed that varying the mass m of the system does not affect the motor characteristics.

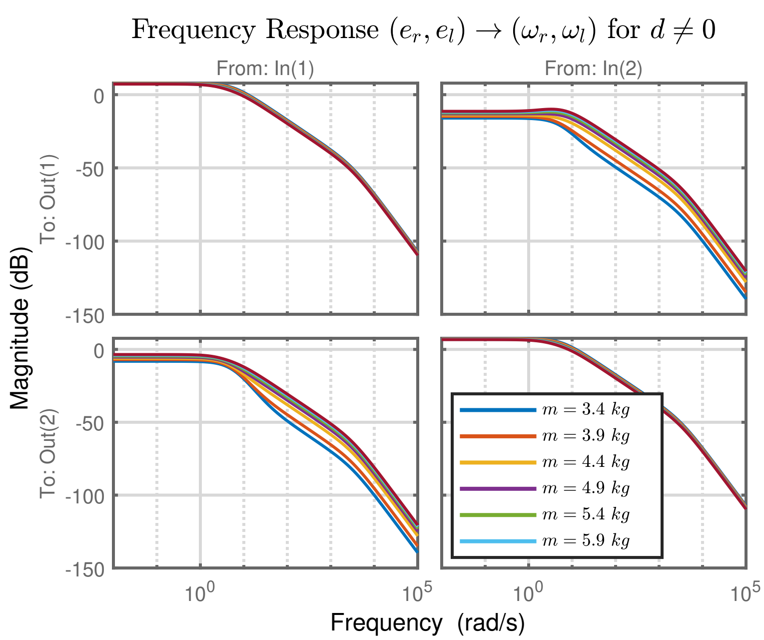


Figure 4.35: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying m

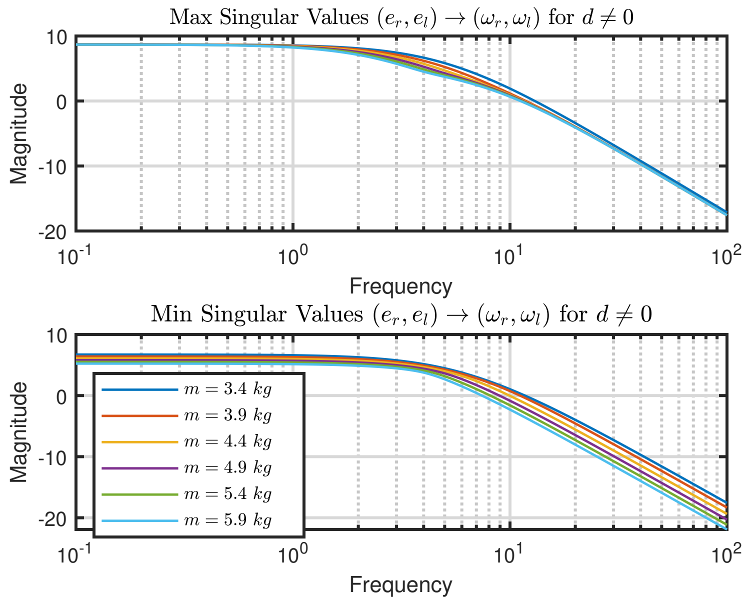


Figure 4.36: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ System: Varying m

From the frequency response plots presented above, the following observations can be made for the $(e_{a_r}, e_{a_l}) \to (\omega_r, \omega_l)$ plant:

- From the response of the off-diagonal elements, it can be noticed that increasing the mass m causes an increase in the input-output coupling with a peak occurring at 4.65 rad/sec

- When compared to the off-diagonal elements, the response of the diagonal elements shows a negligible variation with an increase in the value of m

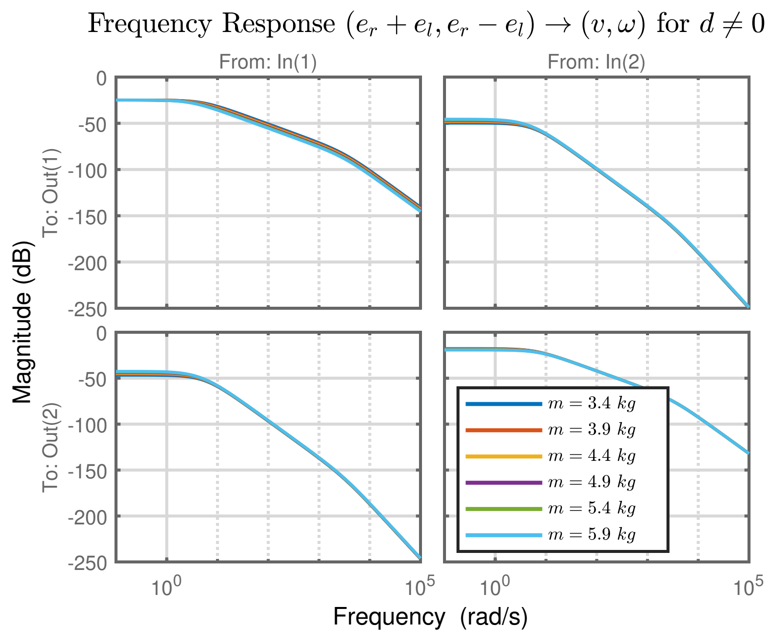


Figure 4.37: Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \to [v,\omega]}$ System: Varying m

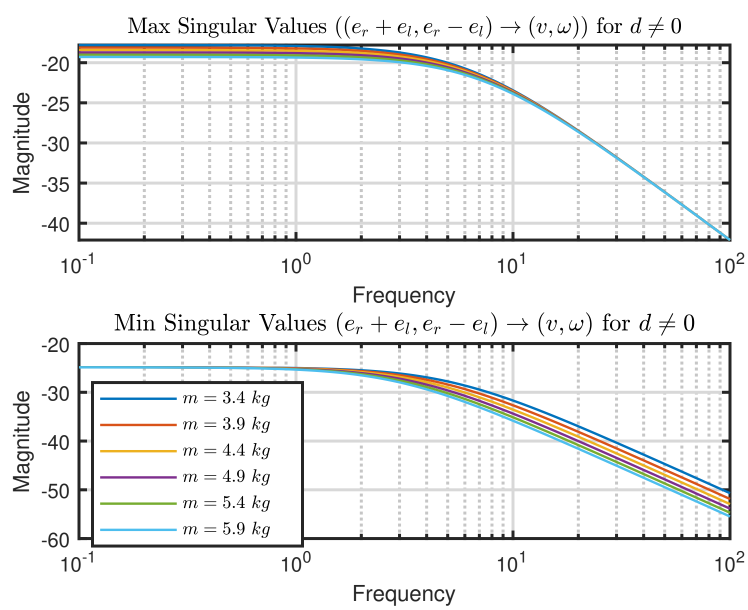


Figure 4.38: Singular Value Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \to [v,\omega]}$ System: Varying m

From the frequency response plots presented above, the following observations can

be made for the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ plant:

- From the off-diagonal elements response, it can be noticed that an increase in the value of m causes a very slight increase in the input-output coupling that is almost negligible when compared to that of $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ system

- In case of diagonal elements response, there is a slight reduction in the gain at higher frequencies for the $(e_{a_r} + e_{a_l} \rightarrow \check{)}$ channel, but the $(e_{a_r} e_{a_l} \rightarrow \omega)$ channel shows negligible variations with an increase in m

Variation in Radius of the Wheel r. he following figures show the frequency response of the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ systems for the variation in r values at $v_{eq} = 2.0$ m/sec, $w_e q = 0.8$ rad/sec and $d = 0.1$ m.

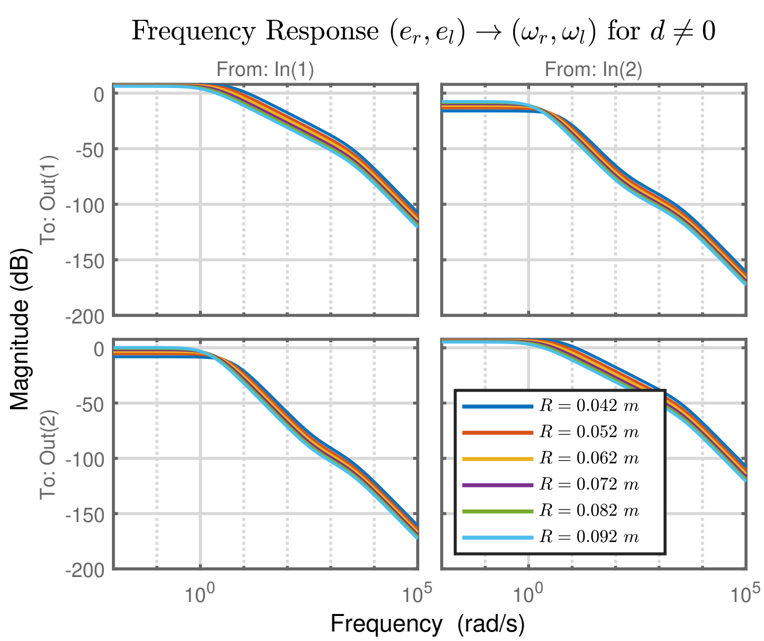


Figure 4.39: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying r

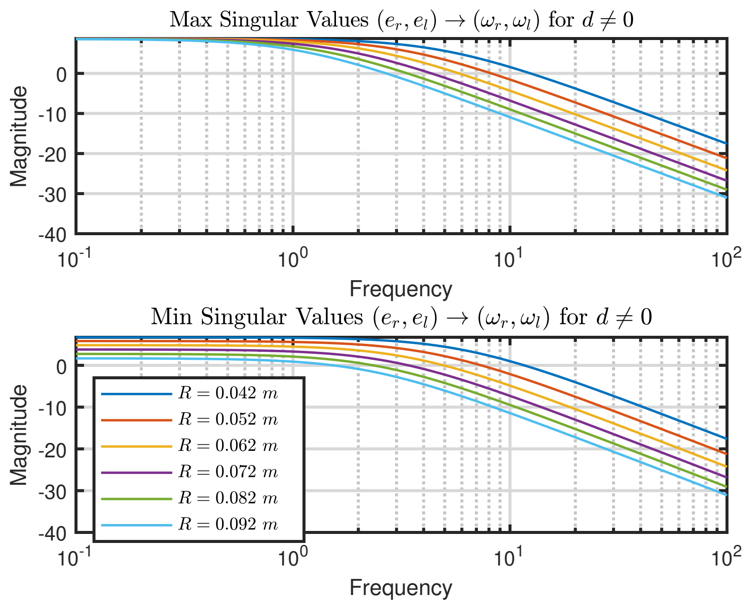


Figure 4.40: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ System: Varying r

From the frequency response plots presented above, the following observations can be made for the $(e_{a_r}, e_{a_l}) \to (\omega_r, \omega_l)$ plant:

- From the response of the diagonal elements, it can be noticed that increasing the radius of the wheel r tends to make the diagonal elements slower
- From the response of the off-diagonal elements, it can be noticed that increasing the value of r causes an increase in coupling at lower frequencies

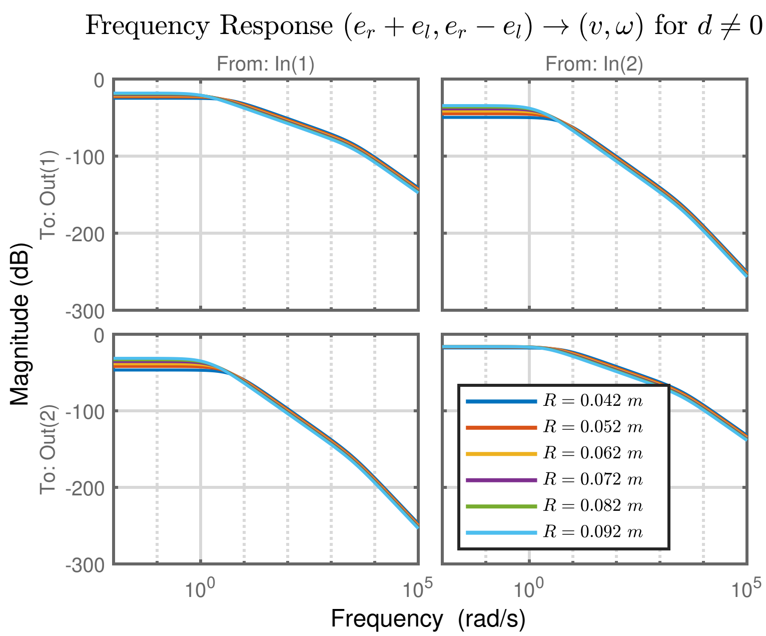


Figure 4.41: Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \to [v,\omega]}$ System: Varying r

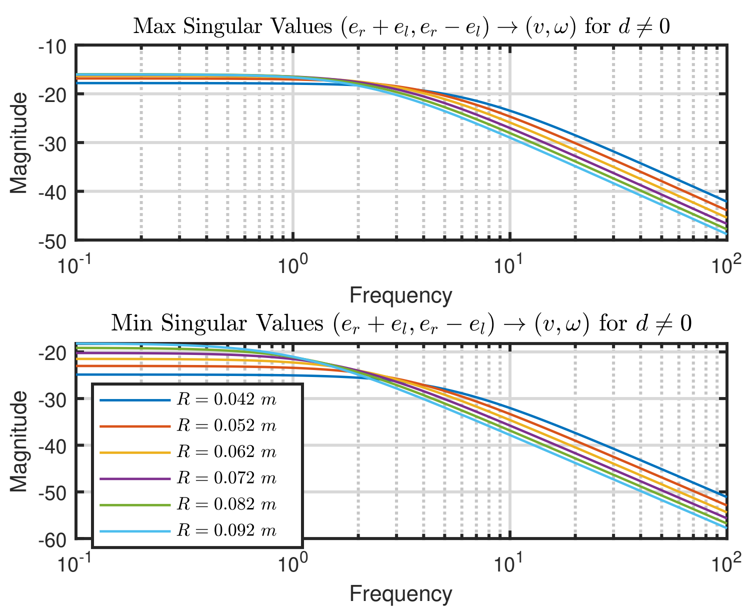


Figure 4.42: Singular Value Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \to [v,\omega]}$ System: Varying r

From the frequency response plots presented above, the following observations can

be made for the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ plant:

- From the off-diagonal elements response, it can be seen that an increase in the radius of the wheel r causes an increase in the input-output coupling at lower frequencies

- For the off diagonal elements response, it can be noticed that $(e_{a_r} + e_{a_l} \rightarrow \check{\ })$ channel response becomes slightly slower with an increase in r, however, the $(e_{a_r} - e_{a_l} \rightarrow \omega)$ channel response is almost unchanged especially at lower frequencies

Chapter 5

IMPACT OF DESIGN PARAMETERS ON OUTER-LOOP: SPEED AND POSITION CONTROL PERFORMANCE

5.1 Introduction and Overview

In the previous chapter, we have seen how the design parameters can impact the coupling and bandwidth properties of the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ systems. The goal of this chapter is to understand how these design parameters impact the performance of trajectory tracking along a curve for different outer loop algorithms. In order to conduct this study, we first consider eight variations of the DDV vehicle based on our study in the previous chapter and theorize about different aspects that we intend to observe (Section 5.2). Secondly, in Section 5.3 we consider the low-frequency approximated models of these eight variations and design decentralized PI controllers for inner-loop (v, ω) speed control, that would enable these eight different models to have an identical closed-loop i.e. same bandwidth and phase margin. In the last section, Section 5.4, we briefly discuss the design and implementation of outer-loop cruise control and planar Cartesian stabilization along a curve followed by detailed time-domain analysis of the performance trade studies.

5.2 DDV Design Notations and Analysis

Figure 5.1 shows the different design variations that we would be considering in this chapter and also the corresponding notations. Primarily, the designs are separated based on the location of the center of gravity d i.e. $d = 0$ and $d > 0$, and subsequently, the moment of inertia I is varied to classify the designs further.

Finally, these designs are further classified based on the input-output models. Albeit in the previous chapter we have presented the impact of variations in m, r, I and d on the performance of the plant, we have considered only the variations due to d, I and input-output modeling for experimental trade studies in this chapter. This is because variations in r and m will affect the torque-speed characteristics of the motor which in turn changes several other motor parameters in the DDV model.

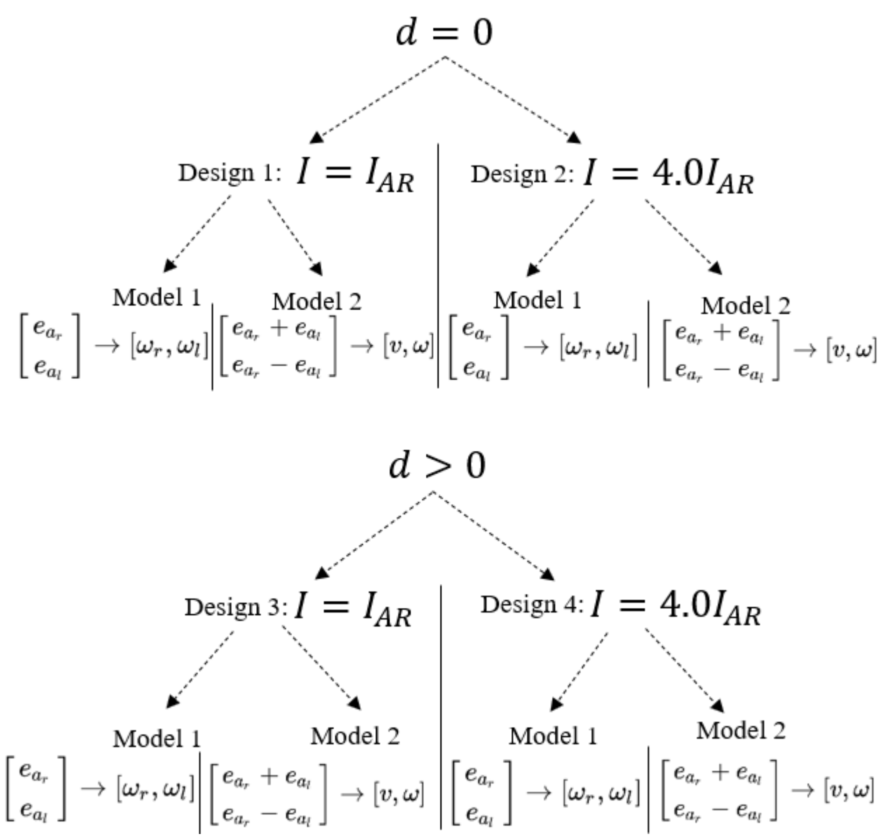


Figure 5.1: DDV Design Notations

The eight design variations represented in Figure 5.1 will be abbreviated as follows:

- **D1M1**: $d = 0$, $I = I_{AR}$, $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$
- **D1M2**: $d = 0$, $I = I_{AR}$, $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$

- **D2M1**: $d = 0$, $I = 4I_{AR}$, $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$
- **D2M2**: $d = 0$, $I = 4I_{AR}$, $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$
- **D3M1**: $d > 0$, $I = I_{AR}$, $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$
- **D3M2**: $d > 0$, $I = I_{AR}$, $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$
- **D4M1**: $d > 0$, $I = 4I_{AR}$, $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$
- **D4M2**: $d > 0$, $I = 4I_{AR}$, $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$

The following equations represent the TITO transfer function representation of the eight design variations. For designs D3Mi and D4Mi (i = 1,2), since the $d \neq 0$, the transfer function representations shown in equations (5.3) - (5.4) and (5.7) - (5.8) are obtained at operation point $v_{eq} = 1$ m/s,$\omega_{eq} = 0.8$ rad/sec [1] . Additionally, Figures 5.2 - 5.5 represent the bode magnitude plots and singular value plots corresponding to these eight design variations. Based on these plots and the transfer function pole locations, the following inference can be made

- *Stability*: D1M1 ≈ D1M2 > D2M1 ≈ D2M2; D3M1 ≈ D3M2 > D4M1 ≈ D4M2
- *Moment of Inertia*: D2M1 ≈ D2M2 > D1M1 ≈ D1M2; D4M1 ≈ D4M2 > D3M1 ≈ D3M2
- *Input-Output Coupling*: D2M1 > D1M1 ≈ D2M2 ≈ D1M1; D4M1 > D3M1 > D3M2 ≈ D4M2

[1] For $d \neq 0$, the TITO DDV model is non-linear and therefore it has to be linearized at specific operating points to represent in transfer function from, please refer to Section 4.3.1

PD1M1:

$$P_{e_{a_{r,l}} \to [\omega_r, \omega_l]} = \frac{\begin{bmatrix} 44335(s+3184)(s+5.113) & 1.4552\text{e-}11(s+3067)(s+5.343) \\ 1.4552\text{e-}11(s+3067)(s+5.343) & 44335(s+3184)(s+5.113) \end{bmatrix}}{(s+3184)^2(s+5.113)^2} \tag{5.1}$$

PD2M1:

$$P_{e_{a_{r,l}} \to [\omega_r, \omega_l]} = \frac{\begin{bmatrix} 27711(s+3186)(s+2.045) & 16623s(s+3187) \\ 16623s(s+3187) & 27711(s+3186)(s+2.045) \end{bmatrix}}{(s+3186)(s+3184)(s+5.113)(s+1.278)} \tag{5.2}$$

PD3M1:

$$P_{e_{a_{r,l}} \to [\omega_r, \omega_l]} = \frac{\begin{bmatrix} 44335(s+3184)(s+5.706) & 32787(s+3187) \\ 32787(s+3187) & 44335(s+3184)(s+5.706) \end{bmatrix}}{(s+3184)^2(s+6.109)(s+5.156)} \tag{5.3}$$

PD4M1:

$$P_{e_{a_{r,l}} \to [\omega_r, \omega_l]} = \frac{\begin{bmatrix} 27711(s+3186)(s+2.282) & 16623(s+3187)(s+0.4933) \\ 16623(s+3187)(s+0.4933) & 27711(s+3186)(s+2.282) \end{bmatrix}}{(s+3184)^2(s+5.11)(s+1.541)} \tag{5.4}$$

PD1M2:

$$P_{\begin{bmatrix} e_{a_r} + e_{a_l} \\ e_{a_r} - e_{a_l} \end{bmatrix} \to [v, \omega]} = \frac{\begin{bmatrix} 931.03(s+5.106) & 0 \\ 0 & 2629.6(s+5.113) \end{bmatrix}}{(s+3184)(s+5.113)(s+5.106)} \tag{5.5}$$

PD2M2:

$$P_{\begin{bmatrix} e_{a_r} + e_{a_l} \\ e_{a_r} - e_{a_l} \end{bmatrix} \to [v, \omega]} = \frac{\begin{bmatrix} 931.03(s+1.278) & 0 \\ 0 & 658.61(s+5.113) \end{bmatrix}}{(s+3186)(s+5.113)(s+1.278)} \tag{5.6}$$

PD3M2:

$$P_{\begin{bmatrix} e_{a_r}+e_{a_l} \\ e_{a_r}-e_{a_l} \end{bmatrix} \to [v,\omega]} = \frac{\begin{bmatrix} 931.03(s+3184)(s+6.152) & -386.74(s+3187) \\ 273.47(s+3187) & 2633.3(s+3184)(s+5.113) \end{bmatrix}}{(s+3184)^2(s+6.109)(s+5.156)} \tag{5.7}$$

PD4M2:

$$P_{\begin{bmatrix} e_{a_r}+e_{a_l} \\ e_{a_r}-e_{a_l} \end{bmatrix} \to [v,\omega]} = \frac{\begin{bmatrix} 931.03(s+3186)(s+1.538) & -96.726(s+3187) \\ 68.413(s+3186) & 658.61(s+3184)(s+5.113) \end{bmatrix}}{(s+3186)(s+3184)(s+5.11)(s+1.541)} \tag{5.8}$$

Plant Frequency Response. The bode magnitude response of each of the eight designs is presented in the figures below. From Figure 5.2, it can be observed that D1M2, D2M2, and D1M1 plants remain completely decoupled at all frequencies, whereas for D2M1, there is a little coupling at dc between the input and output which further increase with frequency and reaches its peak at 3.41 rad/sec (a SISO control strategy is sufficient at lower operation bandwidth i.e. close to DC, but would require a MIMO control strategy for operation bandwidth close to or greater than 3.41 rad/sec). From Figure 5.3, it can be noticed that all the four plants i.e. D3M1, D3M2, D4M1, and D4M2, exhibit a significant amount of coupling between the input and output at all frequencies, and particularly D4M1 exhibits an increase in coupling at higher frequencies with a peak occurring at 3.41 rad/sec.

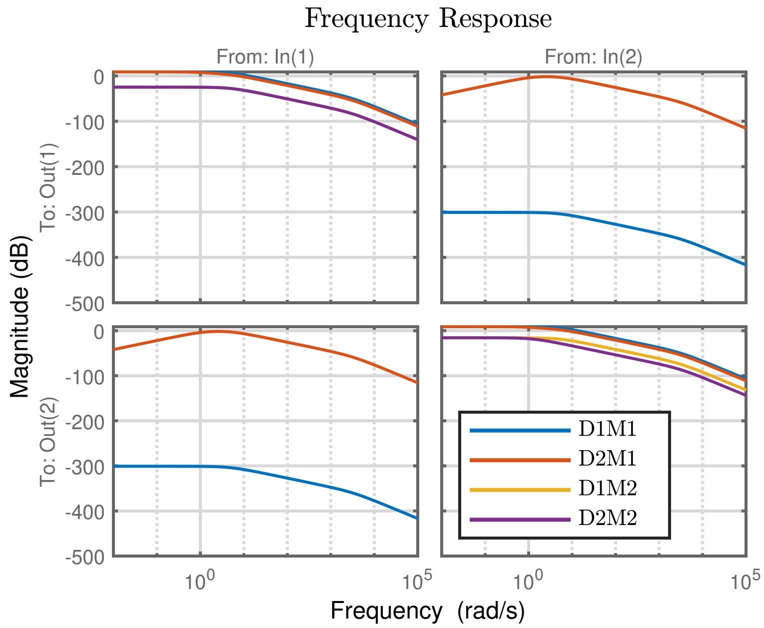


Figure 5.2: Plant Frequency Response D1 & D2

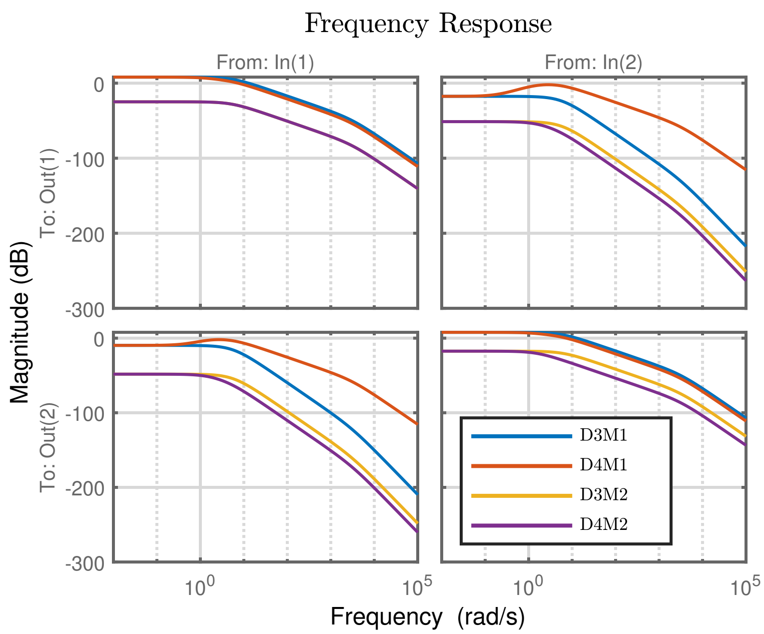


Figure 5.3: Plant Frequency Response D3 & D4

Plant Singular Values. The singular values of each of the eight designs are presented in the figures below. From Figure 5.4, it can be noticed that the minimum and maximum singular values of the D1M1 plant coincide at all frequencies due to the symmetric nature of the plant and the absence of coupling between the input and output, whereas in the case of D2M1, we can see that the minimum and maximum singular values coincide at lower frequencies but diverge at a frequency of 3.4 rad/sec due to an increase in the coupling between the input and output. In case of D1M2 and D2M2, performing a svd analysis at dc showed that the minimum singular value is associated with the $(e_{a_r} + e_{a_l} \rightarrow v)$ channel while the maximum singular value is associated with the $(e_{a_r} - e_{a_l} \rightarrow \omega)$ channel. From Figure 5.5, it can be observed that there is a slight deviation between the minimum and maximum singular values of the D3M1 and D4M1 plants due to coupling between the input and output at lower frequencies and particularly in the case of D4M1, we can see that the minimum and maximum singular values diverge even more at a frequency greater than 3.4 rad/sec due to an increase in the input-output coupling as seen in the bode magnitude plot. In case of D3M2 and D4M2, performing a svd analysis at dc showed that the minimum singular value is associated predominantly with the $(e_{a_r} + e_{a_l} \rightarrow v)$ channel while the maximum singular value is associated predominantly with the $(e_{a_r} - e_{a_l} \rightarrow \omega)$ channel.

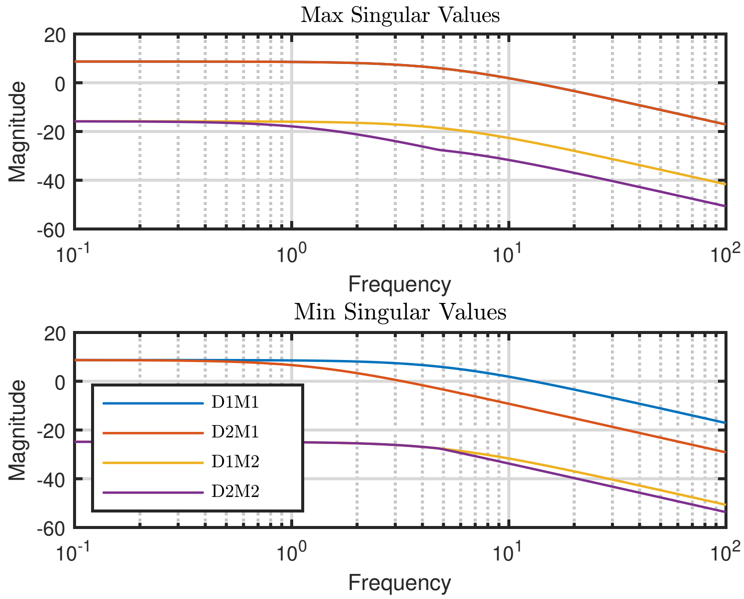


Figure 5.4: Plant Singular Values D1 & D2

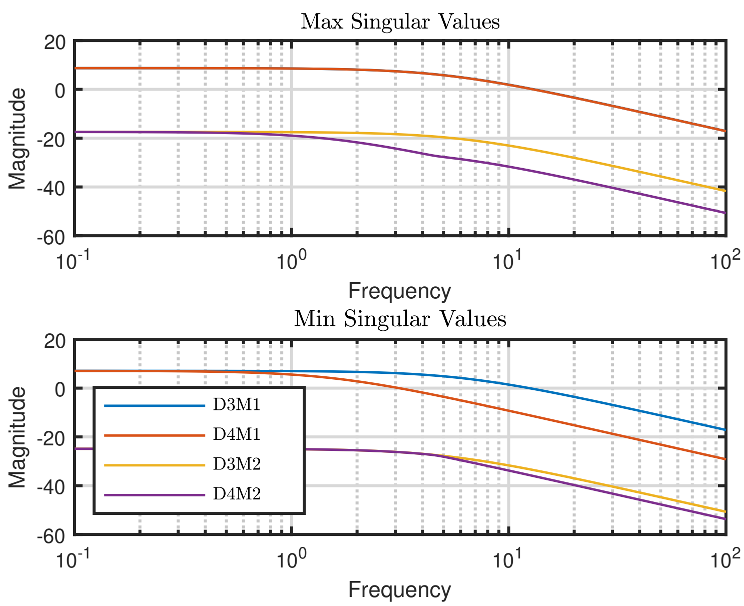


Figure 5.5: Plant Singular Values D3 & D4

5.3 Inner-Loop Decentralized Control Design and Implementation

From Figures 5.2 - 5.3 it can be observed that D1M1, D1M2, and D2M2 are completely decoupled while the remaining designs have a significant amount of coupling. As we already know, in the case of a decoupled system we can go for a decentralized PID based controller design, while we require a multi-variable controller in case of a D3M1, D3M2, D4M1, and D4M2 in order to overcome the input-output coupling. However, in this performance study, we would design decentralized PI-based controllers for all the eight model variations so that we can have a common base for comparing their performance. Figures 5.6, 5.7 show the block diagram representation of the closed-loop control for $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and $(e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}) \rightarrow (v, \omega)$ systems. Form these figures it can be seen that both the inner-loop systems are designed to accept (v_{ref}, ω_{ref}) as input and produce (v, ω) as output.

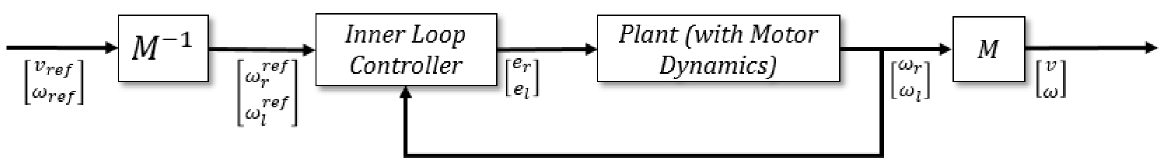


Figure 5.6: $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ System Inner-Loop Control Block Diagram

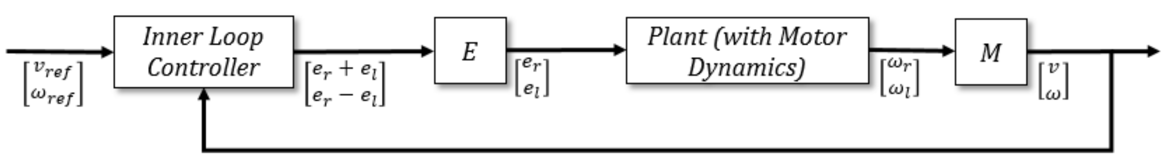


Figure 5.7: $(e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}) \rightarrow (v, \omega)$ System Inner-Loop Control Block Diagram

As mentioned earlier in Section 5.1, we would like to design the inner-loop controller such that all the eight design variations have similar closed-loop characteristics i.e. bandwidth and phase margin. From [46], it can be seen that a PI-based inner-loop speed controller with one pole roll-off and a command prefilter is sufficient to control the low-frequency dynamics of a DDV. Given this, we choose the inner-loop design criteria as follows: 1) Stable closed-loop system 2) Exhibits zeros steady-state error to step reference input, step output disturbances, and step input disturbances 3) Closed-loop phase margin of approximately 60° and a bandwidth of 10rad/sec 4) High-frequency sensor noise and output overshoot attenuation. Based on these criteria the controller chosen has the following structure:

$$K_i = \frac{g_i(s + z_i)}{s}\left(\frac{100}{s + 100}\right), \quad W = \frac{z_i}{s + z_i} \tag{5.9}$$

The pole at the origin (the integrator) is required, based on the internal model principle, in order to ensure zero steady-state error to step reference commands, step input disturbances, and step output disturbances. The prefilter is required in order to ensure there is no overshoot in the output signal which is caused due to the derivative action of the controller zero. Furthermore, a one-pole roll-off almost a decade above the open-loop unit gain crossover frequency (10 rad/sec) is added in order to attenuate high-frequency controller inputs i.e. $K(\infty) \rightarrow 0$. Theoretically, we can increase the bandwidth of the inner-loop indefinitely by using a controller (since none of the plants have transmission zeros), however, in a real-world, every practical system has limitations introduced due to peripheral components such as sensors, actuators, analog to digital converters, sampling rates, etc. and in our case, the actuators have the least bandwidth limitation close to 11 rad/sec for speeds beyond 3 m/s. Ideally, based on the factor of ten rule, it is advised to set the inner-loop bandwidth close to one-tenth of the minimum bandwidth limit enforced by the peripheral components. However, in our design we choose it to be approximately 10 rad/sec in order to understand the effect of input-output coupling and external disturbances that are prevalent at higher frequencies.

The decentralized PI controllers designed based on the above-mentioned criteria are presented below. Please note that the prefilter and the roll-off at high frequency are omitted for brevity.

$$
\begin{aligned}
KD1M1 = K_{e_{a_{r,l}} \rightarrow [\omega_r, \omega_l]} &= \begin{bmatrix} \frac{(0.64s+4.86)}{s} & 0 \\ 0 & \frac{(0.64s+4.86)}{s} \end{bmatrix} \\
KD2M1 = K_{e_{a_{r,l}} \rightarrow [\omega_r, \omega_l]} &= \begin{bmatrix} \frac{(1.07s+6.90)}{s} & 0 \\ 0 & \frac{(1.07s+6.90)}{s} \end{bmatrix} \\
KD3M1 = K_{e_{a_{r,l}} \rightarrow [\omega_r, \omega_l]} &= \begin{bmatrix} \frac{(0.63s+5.22)}{s} & 0 \\ 0 & \frac{(0.63s+5.22)}{s} \end{bmatrix} \\
KD4M1 = K_{e_{a_{r,l}} \rightarrow [\omega_r, \omega_l]} &= \begin{bmatrix} \frac{(1.06s+6.96)}{s} & 0 \\ 0 & \frac{(1.06s+6.96)}{s} \end{bmatrix}
\end{aligned} \tag{5.10}
$$

$$
\begin{aligned}
KD1M2 = K_{\begin{bmatrix} e_{a_r} + e_{a_l} \\ e_{a_r} - e_{a_l} \end{bmatrix} \rightarrow [v, \omega]} &= \begin{bmatrix} \frac{(30.64s+231.5)}{s} & 0 \\ 0 & \frac{10.83s+81.87}{s} \end{bmatrix} \\
KD2M2 = K_{\begin{bmatrix} e_{a_r} + e_{a_l} \\ e_{a_r} - e_{a_l} \end{bmatrix} \rightarrow [v, \omega]} &= \begin{bmatrix} \frac{(30.64s+231.5)}{s} & 0 \\ 0 & \frac{(46.56s+144.8)}{s} \end{bmatrix} \\
KD3M2 = K_{\begin{bmatrix} e_{a_r} + e_{a_l} \\ e_{a_r} - e_{a_l} \end{bmatrix} \rightarrow [v, \omega]} &= \begin{bmatrix} \frac{(30.64s+231.5)}{s} & 0 \\ 0 & \frac{(10.61s+94.2)}{s} \end{bmatrix} \\
KD4M2 = K_{\begin{bmatrix} e_{a_r} + e_{a_l} \\ e_{a_r} - e_{a_l} \end{bmatrix} \rightarrow [v, \omega]} &= \begin{bmatrix} \frac{(30.64s+231.5)}{s} & 0 \\ 0 & \frac{(46.34s+157.2)}{s} \end{bmatrix}
\end{aligned} \tag{5.11}
$$

Reference Signal to Output (T_{ry}) Frequency Response. The bode magnitude response of the inner-loop system with pre-filter for each of eight designs is presented in the figures below. From Figure 5.8, by observing the response of the diagonal elements it can be seen that all the systems exhibit a close loop bandwidth of approximately 10 rad/sec with the D2M1 system exhibiting a slight peak at 4 rad/sec. More importantly, it can be noticed that D1M2 and D2M2 have no coupling between the inputs and outputs. In the case of D1M1 and D2M1, it can be noticed that they exhibit a slight but almost negligible amount of coupling at low frequencies with a peak occurring at 4 rad/sec for the D2M1 system. From Figure 5.9, by observing the response of the diagonal elements it can be seen that all the systems exhibit a close loop bandwidth of approximately 10 rad/sec with the D4M1 system exhibiting a slight peak at 4 rad/sec. From observing the response of the off-diagonal elements, it can be seen that all the four designs i.e. D3M1, D3M2, D4M1, D4M2, exhibit little coupling between the inputs and outputs at dc that gradually increases with frequency and with a peak occurring in between 4 rad/sec - 6 rad/sec.

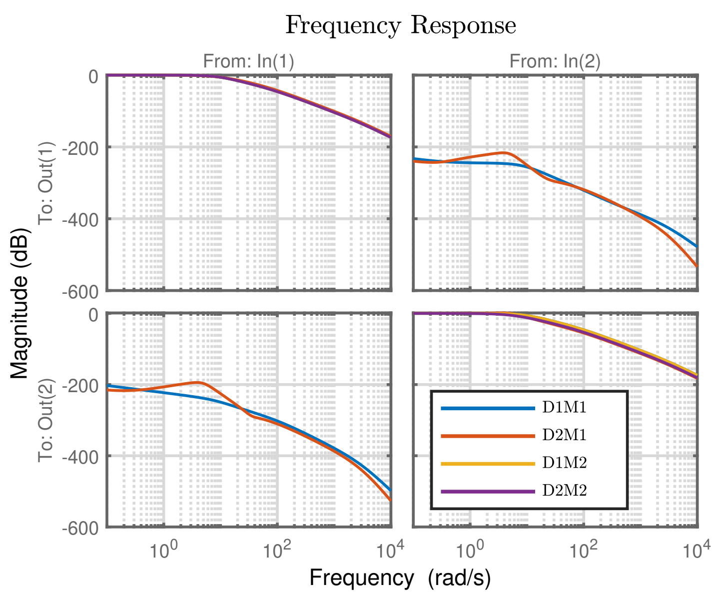


Figure 5.8: Inner-Loop Frequency Response T_{ry}: D1 & D2

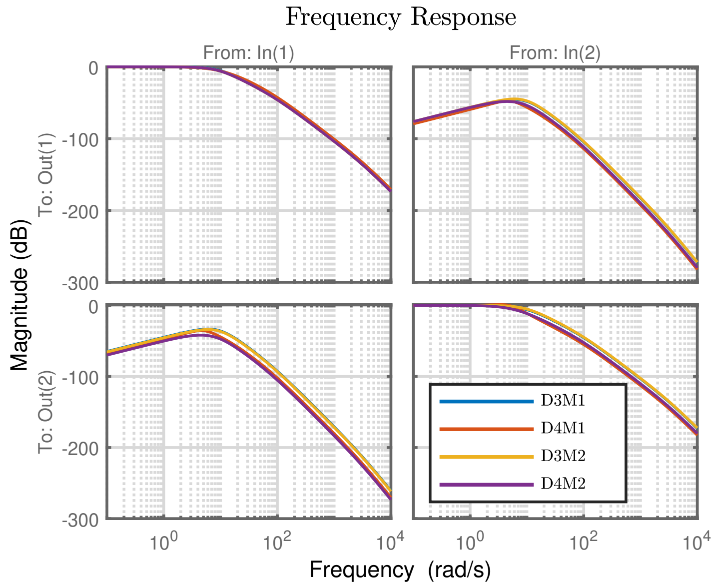


Figure 5.9: Inner-Loop Frequency Response T_{ry}: D3 & D4

Open Loop Singular Values. The open-loop singular values at error for the inner-loop system for each of the eight designs are presented in the figures below. Since we are using a decentralized controller, the open-loop singular values at error will be the same as those at the input for the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ models, whereas they would differ for the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ models. From Figure 5.10, it can be seen that the minimum and maximum singular values match at lower frequencies for all the systems except for the D2M2 for which they exhibit a very slight deviation. At low frequencies, the singular value plots exhibit a slope of -20 dB/dec due to the integral action in each control channel, and this suggests that low-frequency reference commands will be followed, and low-frequency output disturbances and high-frequency sensor noise will be attenuated. More specifically, reference commands with frequency content below 1.2 rad/sec will be followed to within about 20 dB i.e. with a steady-state error of about 10% and output disturbances with frequency content below 1.2 rad/sec should be attenuated by approximately 20 dB for all the designs.

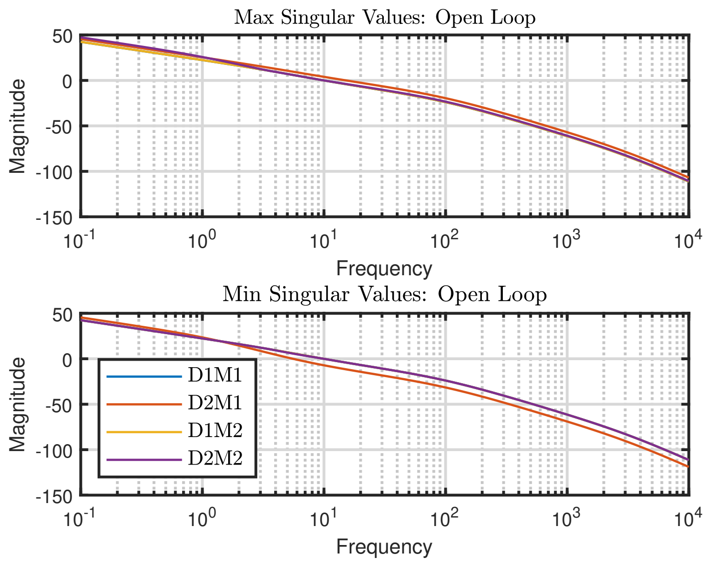


Figure 5.10: Open Loop Singular Values: D1 & D2

From Figure 5.11, it can be seen that the minimum and maximum singular values show a very slight deviation at low frequencies for all the designs, particularly in the case of D3M1 and D3M2. At low frequencies, the singular value plots exhibit a slope of -20 dB/dec due to the integral action in each control channel. This suggests that low-frequency reference commands will be followed, and low-frequency output disturbances and high-frequency sensor noise will be attenuated. More specifically, reference commands with frequency content below 1.2 rad/sec will be followed to within about 20 dB i.e. with a steady-state error of about 10% and output disturbances with frequency content below 1.2 rad/sec should be attenuated by approximately 20 dB for all the designs.

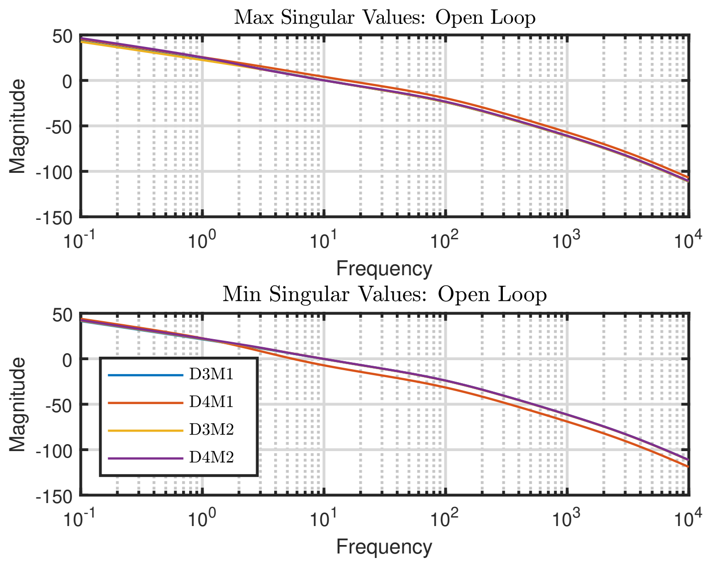


Figure 5.11: Open Loop Singular Values: D3 & D4

Sensitivity Singular Values. The sensitivity singular values at error for the inner-loop system for each of the eight designs are presented in the figures below. Since we are using a decentralized controller, the sensitivity singular values at error will be the same as that at the input for the $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ models, whereas they would differ for the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ models. From Figure 5.12, it can be seen that low-frequency reference commands will be followed and low-frequency output disturbances will be attenuated. More specifically, it can be seen that reference commands with frequency content below 1.27 rad/sec should be followed to within about 20 dB i.e. with a steady-state error of about 10%, and output disturbances with frequency content below 1.27 rad/sec should be attenuated by approximately 20 dB for all the designs. Further, in the case of D2M1, there is a peak of approximately 1.8 dB in the maximum singular values at 6.8 rad/sec, while in the case of remaining plants i.e. D1M1, D1M2, D2M2 a peak of approximately 0.65 dB, which is not significant enough, is observed at 48.8 rad/sec.

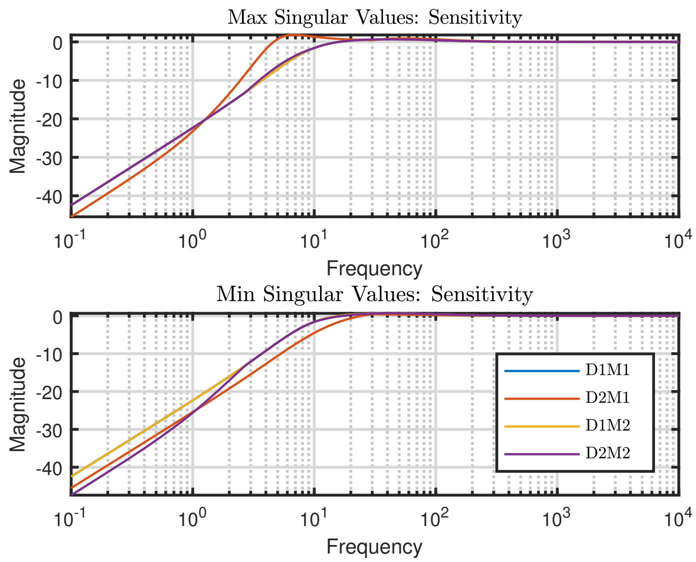


Figure 5.12: Sensitivity Singular Values: D1 & D2

From Figure 5.13, it can be seen that low-frequency reference commands will be followed and low-frequency output disturbances will be attenuated. More specifically, it can be seen that reference commands with frequency content below 1.20 rad/sec should be followed to within about 20 dB i.e. with a steady-state error of about 10%, and output disturbances with frequency content below 1.27 rad/sec should be attenuated by approximately 20 dB for all the designs. Further, in the case of D4M1, there is a peak of approximately 1.58 dB in maximum singular values at 6.8 rad/sec, while in the case of remaining plants i.e. D3M1, D3M2, D4M2 a peak of approximately 0.65 dB, which is not significant enough, is observed at 48.8 rad/sec.

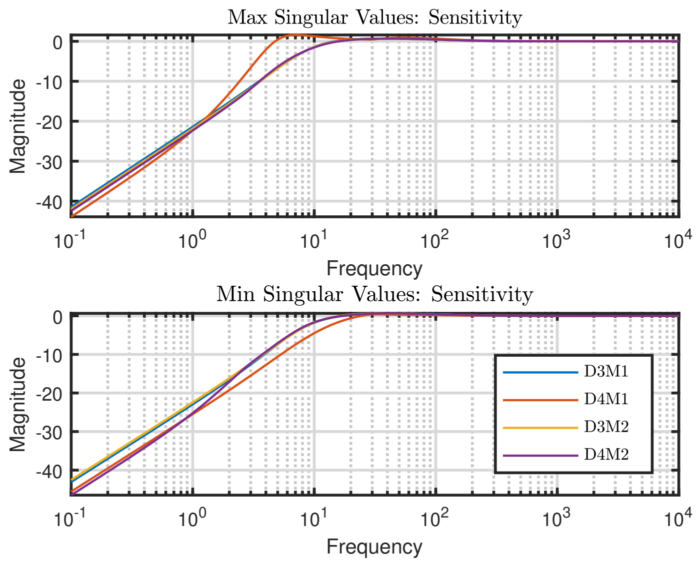


Figure 5.13: Sensitivity Singular Values: D3 & D4

Complementary Sensitivity Singular Values. The complementary sensitivity singular values at error for the inner-loop system for each of the eight designs are presented in the figures below. Since we are using a decentralized controller, the complementary sensitivity singular values at error will be the same as that at the input for the $(e_{a_r}, e_{a_l}) \to (\omega_r, \omega_l)$ models, whereas they would differ for the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \to (v, \omega)$ models. From Figure 5.14, it can be seen that low-frequency reference commands will be followed for all the designs, although a better inference regarding the same can be made from the sensitivity singular values. Further, in the case of D2M1, there is a peak of approximately 2.64 dB in maximum singular values at 4.0 rad/sec, while in the case of D2M2 a peak of approximately 0.68 dB, which is not significant enough, is observed at 4.0 rad/sec.

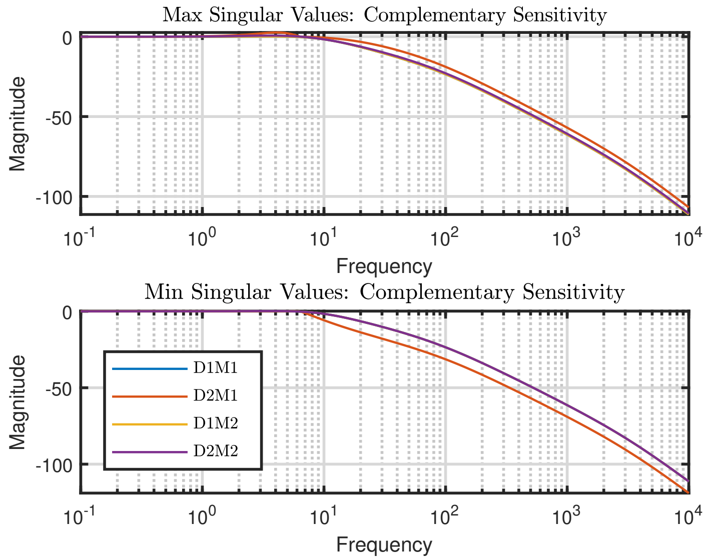


Figure 5.14: Complementary Sensitivity Singular Values: D1 & D2

From Figure 5.15, it can be seen that low-frequency reference commands will be followed for all designs, though a better inference regarding the same can be made from the sensitivity singular values. Further, in the case of D4M1, there is a peak of approximately 2.24 dB in the maximum singular values at 4.0 rad/sec, while in the case of D4M2 a peak of approximately 0.68 dB, which is not significant enough, is observed at 4.0 rad/sec.

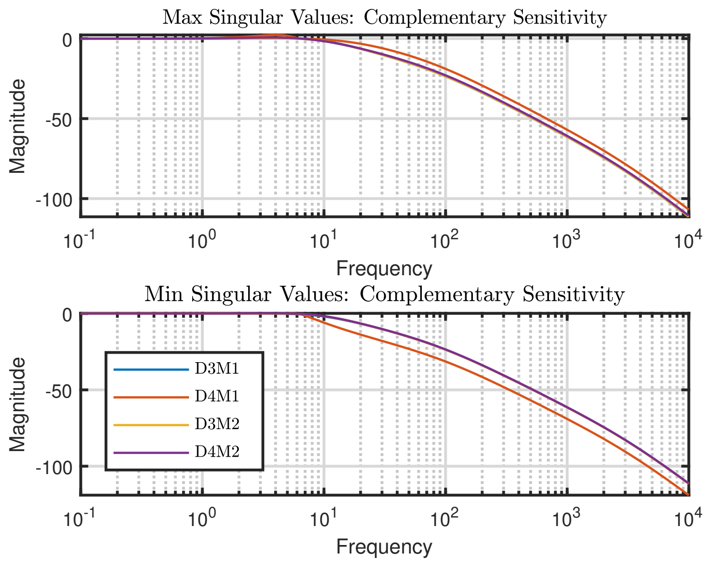


Figure 5.15: Complementary Sensitivity Singular Values: D3 & D4

5.4 Outer-Loop Control Design and Impact of Design Parameters: Simulation and Hardware Trade Studies

Based on the results presented in Sections 5.2, 5.3, we were able to rank the eight designs based on their natural - stability, input-output coupling, and closed-loop control effort. These variations are a consequence of the changes in design parameters. Now, in order to understand how the eight designs impact the performance of outer loop trajectory tracking algorithms, we shall begin by designing and implementing PID based (v, θ) Cruise Control and Planar (x, y) Cartesian Stabilization along a curve. Figure 5.16 shows the simulation trajectory for testing and recording the performance of these algorithms. Column 1 in Table 5.1 shows various parameters that we would be comparing for each of the eight designs. The following sections will provide more details into the design, implementation, and time-domain analysis of the performance of these algorithms.

Name	**Cruise Control**	**Planar Cartesian Stabilization**
$\|\|\theta_e\|\|_\infty$ vs v_{ref} [2]	*	
$\|\|\theta_e\|\|_\infty$ vs R [3]	*	
$\|\|v_e\|\|_\infty$ vs v_{ref}	*	
$\|\|v_e\|\|_\infty$ vs R	*	
$\|\|x_e\|\|_\infty$ vs v_{ref}		*
$\|\|x_e\|\|_\infty$ vs R		*
$\|\|y_e\|\|_\infty$ vs v_{ref}		*
$\|\|y_e\|\|_\infty$ vs R		*
U [4] s v_{ref}	*	*
U vs R	*	*

[2] v_{ref} - tracking velocity

[3] R - radius of the track

[4] U - control effort

Table 5.1: Summary of Trade Studies Conducted

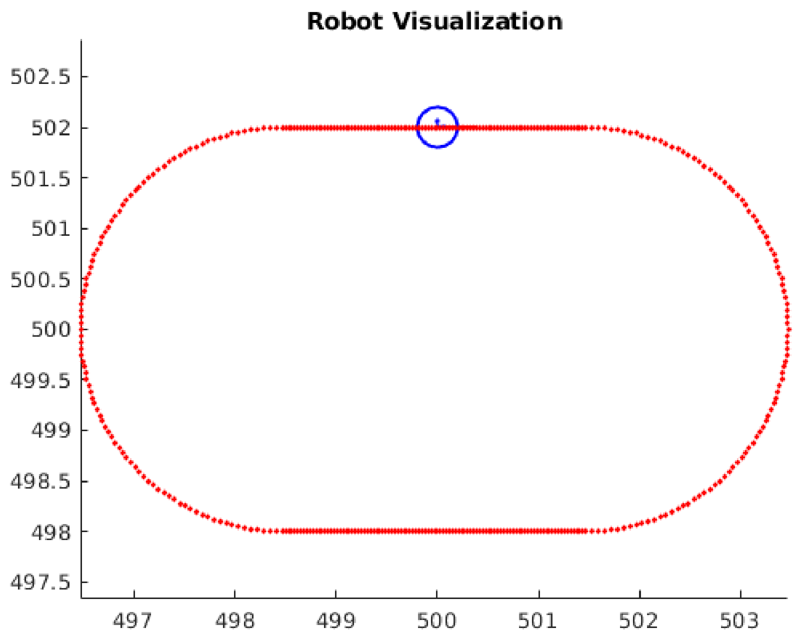


Figure 5.16: Reference Trajectory Visualization

5.4.1 Outer-Loop 1: (v, θ) Cruise Control

In this section, we will show the design and implementation of the (v, θ) cruise control along a curve. Figures 5.17, 5.18 show the block diagrams of the closed-loop system implementation for both $P_{[e_{a_r}, e_{a_l}] \to [\omega_r, \omega_l]}$ and $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \to [v, \omega]}$ plants in the inner-loop. Here, the (v, θ) are obtained from the HTC Vive Motion capture system, the data is passed through a moving average filter before passing into the feedback loop. The (v_{ref}, θ_{ref}) commands are predetermined based on reference velocity, sampling rate, length, and radius of curvature of the trajectory.

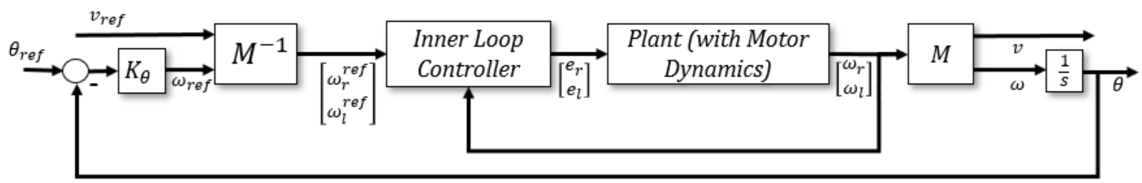


Figure 5.17: $(e_{a_r}, e_{a_l}) \to (\omega_r, \omega_l)$ System Outer-Loop Cruise Control Block Diagram

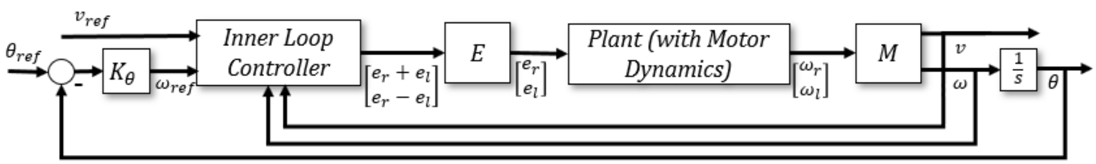


Figure 5.18: $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \to (v, \omega)$ System Outer-Loop Cruise Control Block Diagram

The error dynamics of the cruise control system are quite simple and can be expressed as

$$\dot{e}_\theta = -\omega \tag{5.12}$$

where e_θ is the error between the desired orientation and actual orientation, and ω is the angular velocity of the DDV. These error dynamics can be stabilized using a proportional controller as follows

$$\omega = -k_\theta\, e_\theta \tag{5.13}$$

Using a proportional controller is completely justified because the system from inputs (v_{ref}, ω_{ref}) to outputs (v, ω) can be expressed as $diag(\frac{a}{s+a}, \frac{b}{s+b})$ - as long as the inputs are within the bandwidth limit of the inner-loop. This is a consequence of the well designed inner loop. Therefore, for the outer-loop θ control we can just use a proportional controller to stabilize the system provided that the gain is not too large - remember, the bandwidth of the outer-loop control should always be less than that of the inner-loop by a factor of five at least ($BW_{outerloop} \leq 0.2\, BW_{innerloop}$). If the gain is too large, the system will begin to oscillate, and in that case, a PD controller with a proper roll-off, and prefilter would be better.

Simulation and Hardware Trade Studies

The following figures show the variation in the v_e, θ_e and RMS Voltage (control effort) with respect to changes in the radius of the track, and reference velocity - for each of the eight design variations.

Increasing Tracking Velocity (v_{ref}) for Fixed Radius of Curvature of Trajectory (R). The simulation and hardware data presented in Figure 5.19 - 5.28 are obtained at inner-loop bandwidth $B_i = 10$ rad/sec and radius of track $R = 1.5$ m while varying the trajectory tracking velocity. The hardware results had to be limited to trajectory tracking velocity $v_{ref} \leq 2$ m/s due to the physical restrictions of the experimental setup.

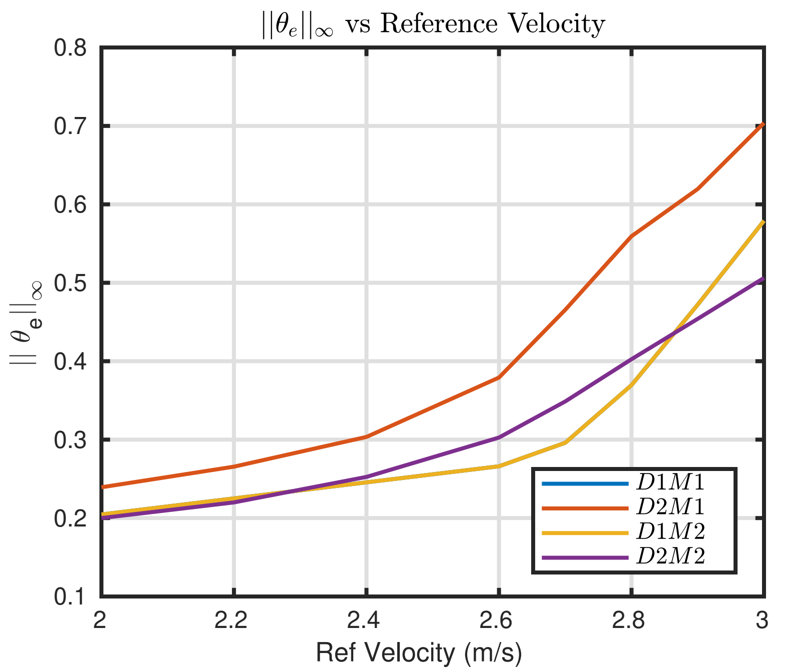


Figure 5.19: $||\theta_e||_\infty$ vs Reference Velocity: Simulation Results

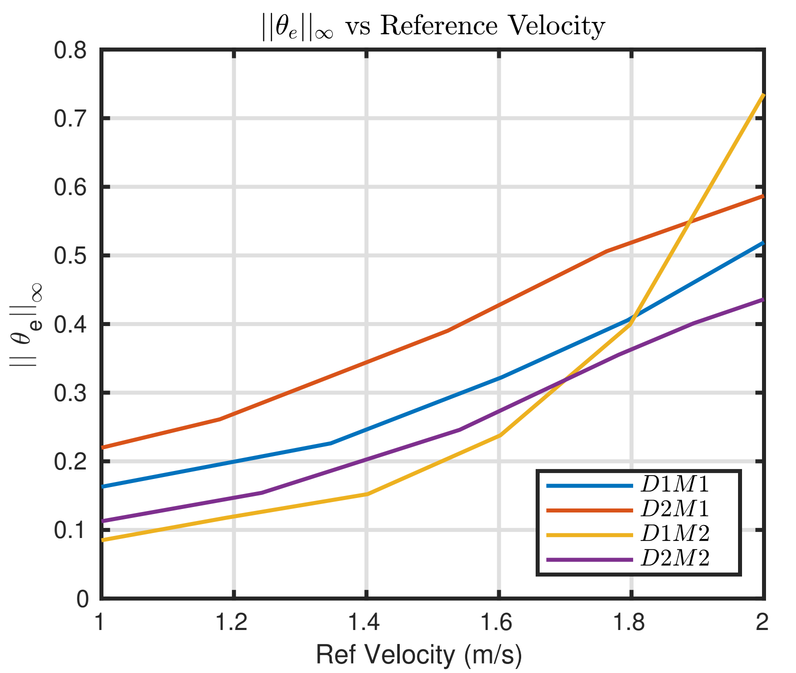


Figure 5.20: $||\theta_e||_\infty$ vs Reference Velocity: Hardware Results

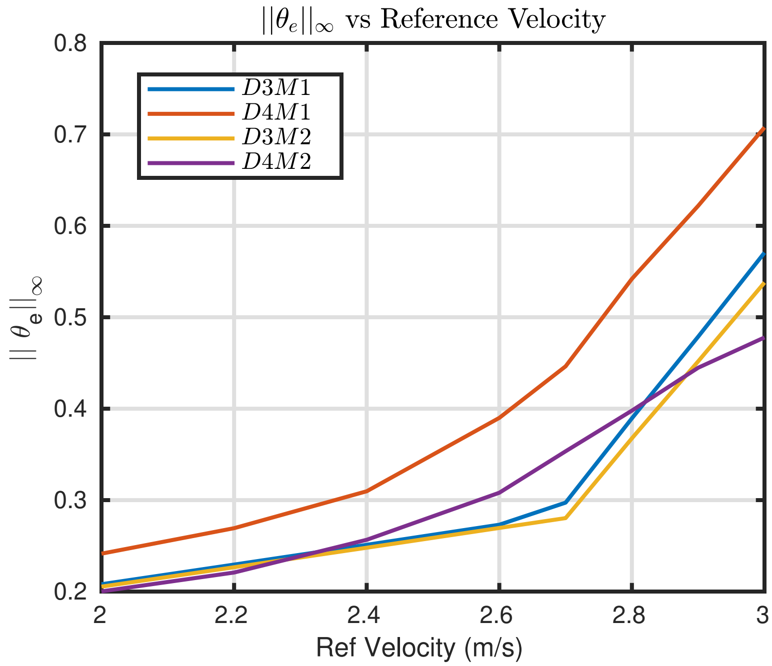


Figure 5.21: $||\theta_e||_\infty$ vs Reference Velocity: Simulation Results

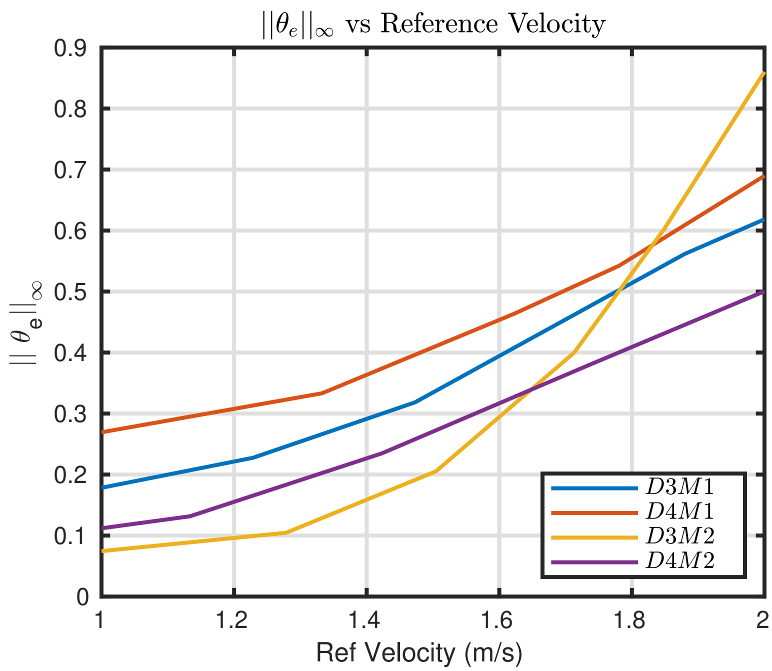


Figure 5.22: $||\theta_e||_\infty$ vs Reference Velocity: Hardware Results

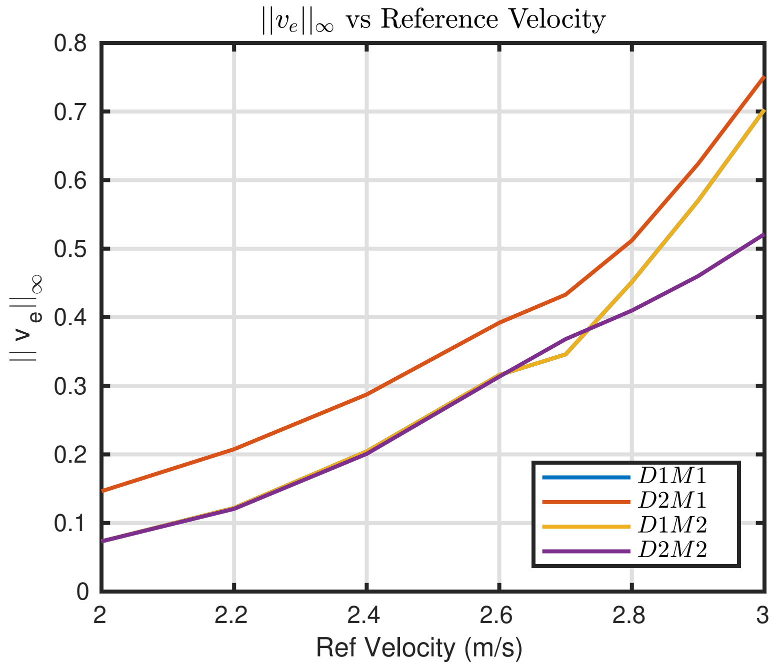


Figure 5.23: $||v_e||_\infty$ vs Reference Velocity: Simulation Results

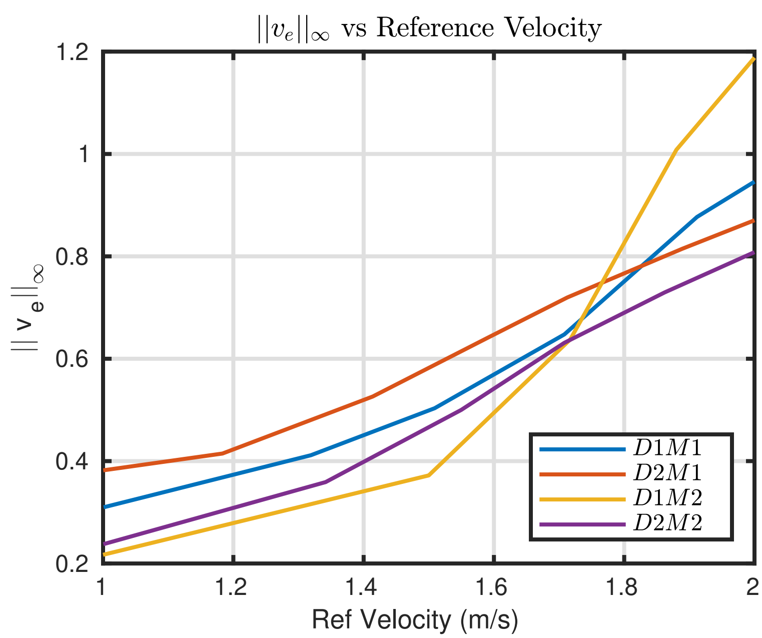


Figure 5.24: $||v_e||_\infty$ vs Reference Velocity: Hardware Results

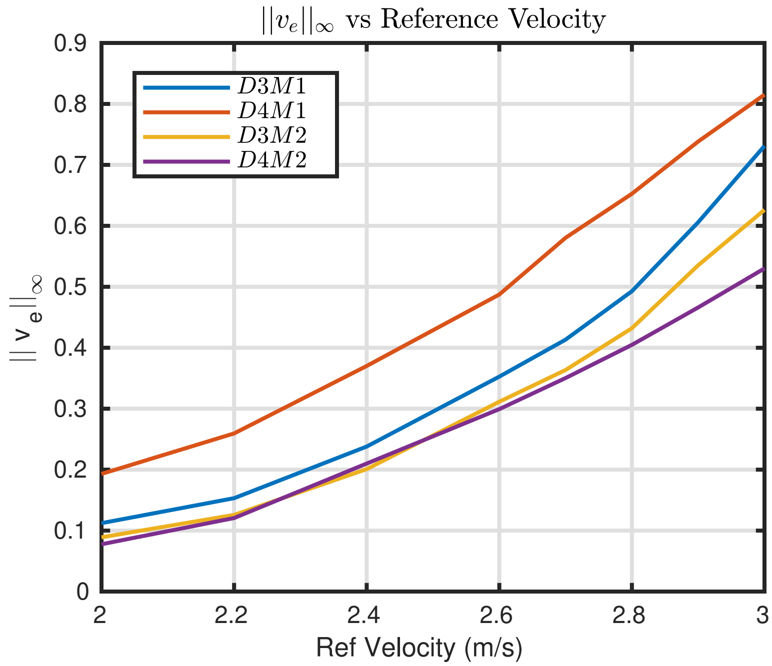


Figure 5.25: $||v_e||_\infty$ vs Reference Velocity: Simulation Results

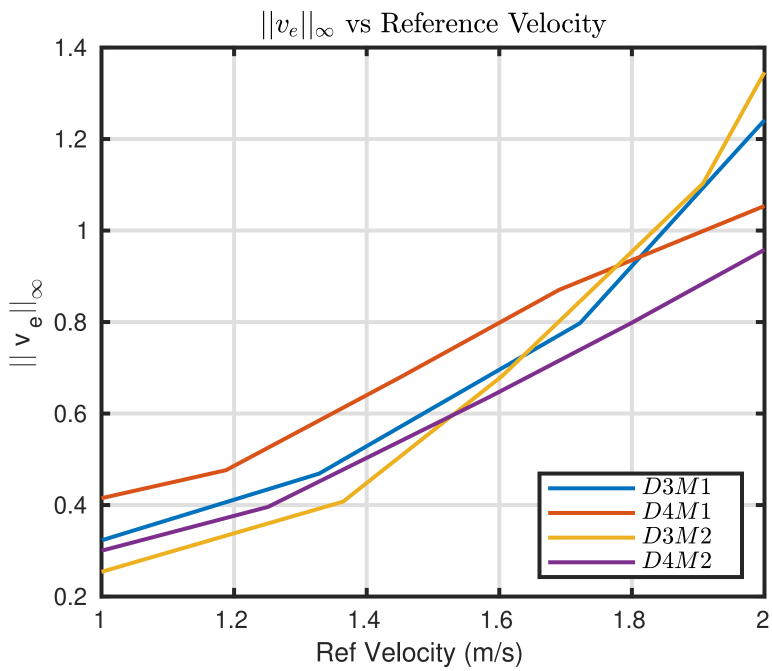


Figure 5.26: $||v_e||_\infty$ vs Reference Velocity: Hardware Results

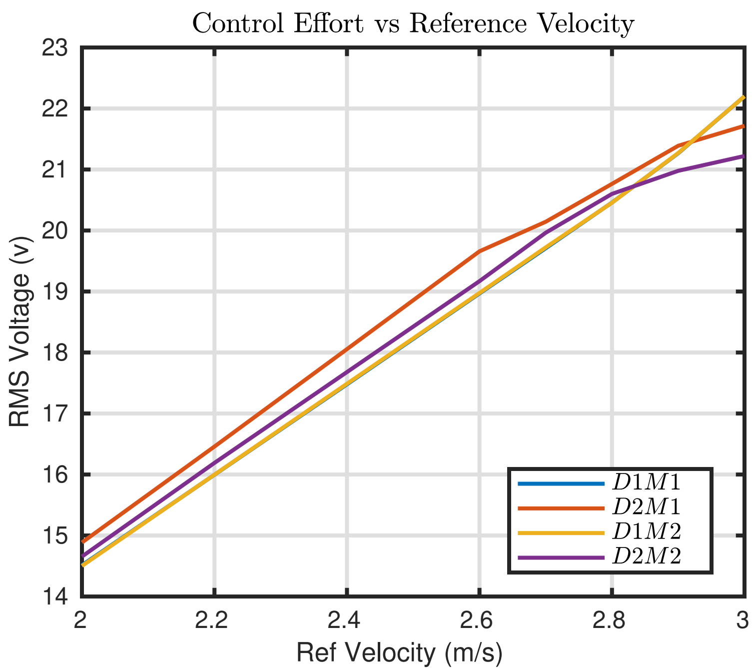


Figure 5.27: Control Effort vs Reference Velocity: Simulation Results

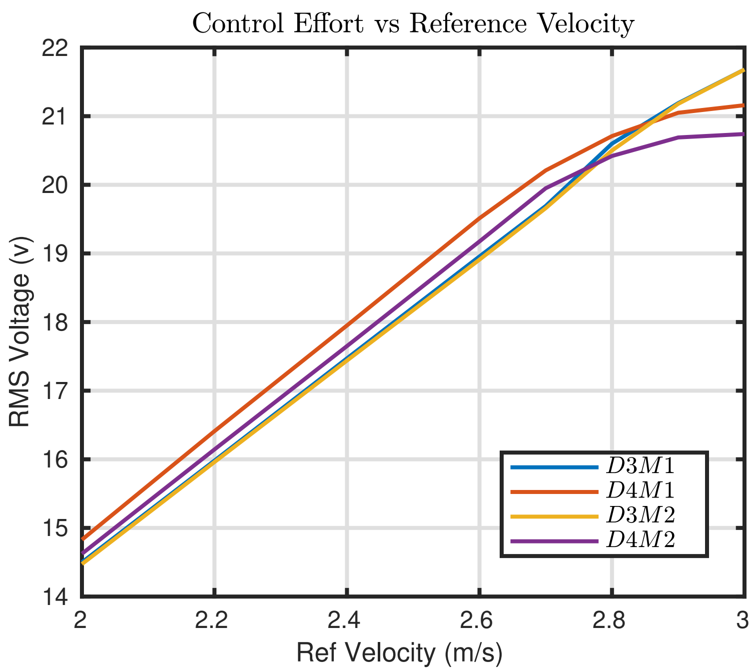


Figure 5.28: Control Effort vs Reference Velocity: Simulation Results

From Figures 5.19 - 5.28, the following observations can be made:

- Comparing simulation and hardware results it can be noticed that the eight design variations follow the same trend, however, the deviations in errors corresponding to different designs are more significant in case of hardware results.

- In both simulation and hardware plots, increasing the trajectory tracking velocity v_{ref} causes an increase in $||v_e||_\infty, ||\theta_e||_\infty$, RMS Voltage irrespective of the properties of each of the eight systems.

- From $||v_e||_\infty, ||\theta_e||_\infty$ vs trajectory tracking velocity plots, it can be seen that the systems with more stable plants i.e. D1M1, D1M2, D3M2, and D3M1, exhibit higher errors and control effort, when compared to the other systems, with an increase in trajectory tracking velocity $v_{ref} \geq 1.7$ m/s, at a constant $R = 1.5$ m.

- For $v_{ref} \leq 1.7$ m/s, it can be noticed that systems with higher input-output coupling at lower frequencies i.e. D2M1 and D4M1, exhibit higher errors $||v_e||_\infty, ||\theta_e||_\infty$ and control effort when when compared to other systems.

- For D2M2, we notice that $||v_e||_\infty \leq 0.8, ||\theta_e||_\infty \leq 0.44$ at $R = 1.5$ m and $B_i = 10$ rad/sec. This means that as long as radius of curvature $R \geq 1.5$ m and inner-loop bandwidth $B_i = 10$ rad/sec, the trajectory tracking performance will not be affected significantly for variations in reference velocity $v_{ref} \leq 2$ m/s.

- For D4M2, we can see that $||v_e||_\infty \leq 0.8, ||\theta_e||_\infty \leq 0.44$ for $1 \leq v_{ref} \leq 1.8$, at $R \geq 1.5$ m and $B_i \geq 10$ rad/sec. Within this reference tracking velocity range $1 \leq v_{ref} \leq 1.8$ it can be seen that D4M2 performance is similar to that of D2M2, but steeply increases for $v_{ref} > 1.8$ m/s.

- For $v_{ref} < 1.8$ m/s, a SISO controller is sufficient to provide good trajectory tracking properties for a system with input-output coupling, at $R \geq 1.5$ m.

- However, for $v_{ref} < 1.85$, a MIMO controller is necessary to achieve a similar performance.

Varying Radius of Curvature of Trajectory (R) for Fixed Tracking Velocity (v_{ref}). The simulation and hardware data presented in Figures 5.29 - 5.38 is obtained at inner-loop bandwidth $B_i = 10$ rad/sec and trajectory tracking velocity $v_{ref} = 1$ m/s while varying the radius of curvature. The hardware results had to be limited to radius of curvature $R \leq 2$ m due to the physical restrictions of the experimental setup.

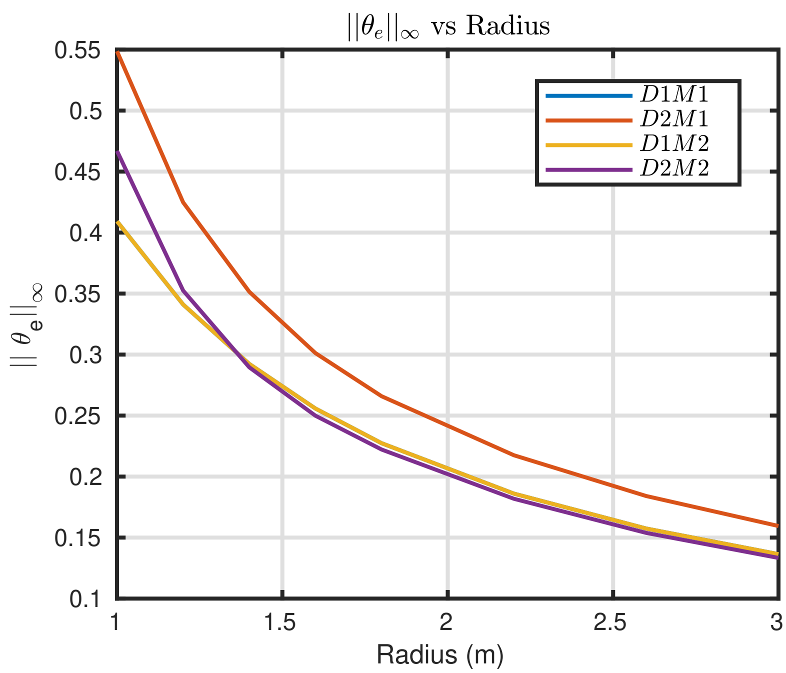


Figure 5.29: $||\theta_e||_\infty$ vs Radius of Track: Simulation Results

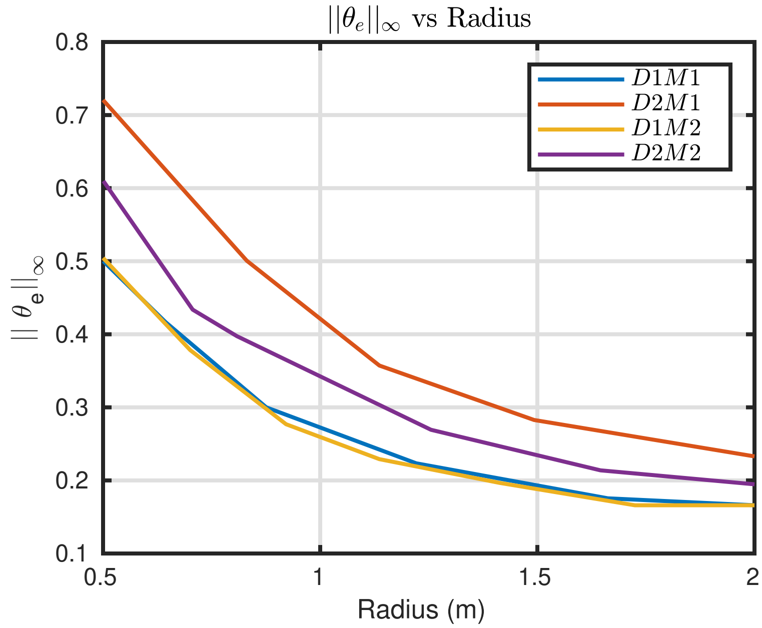


Figure 5.30: $||\theta_e||_\infty$ vs Radius of Track: Hardware Results

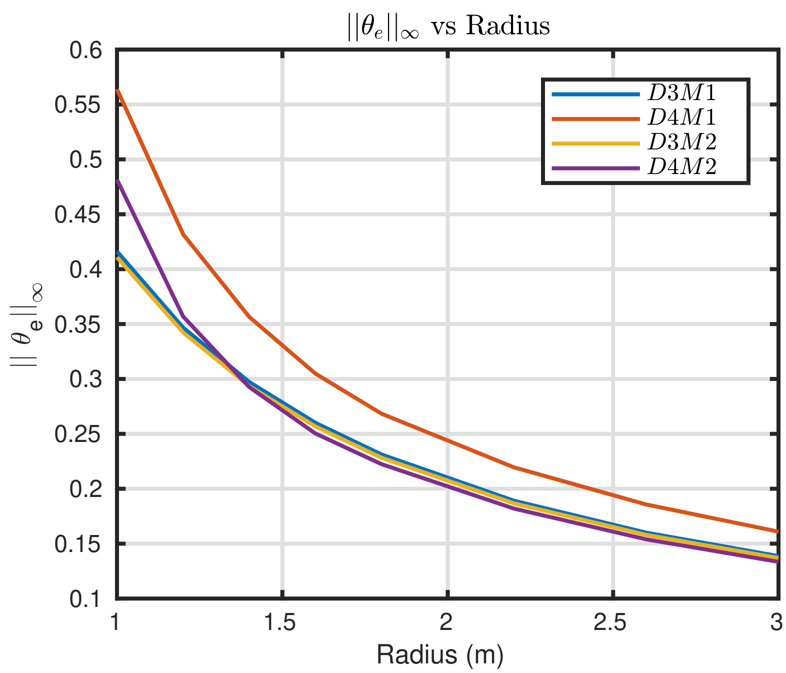


Figure 5.31: $||\theta_e||_\infty$ vs Radius of Track: Simulation Results

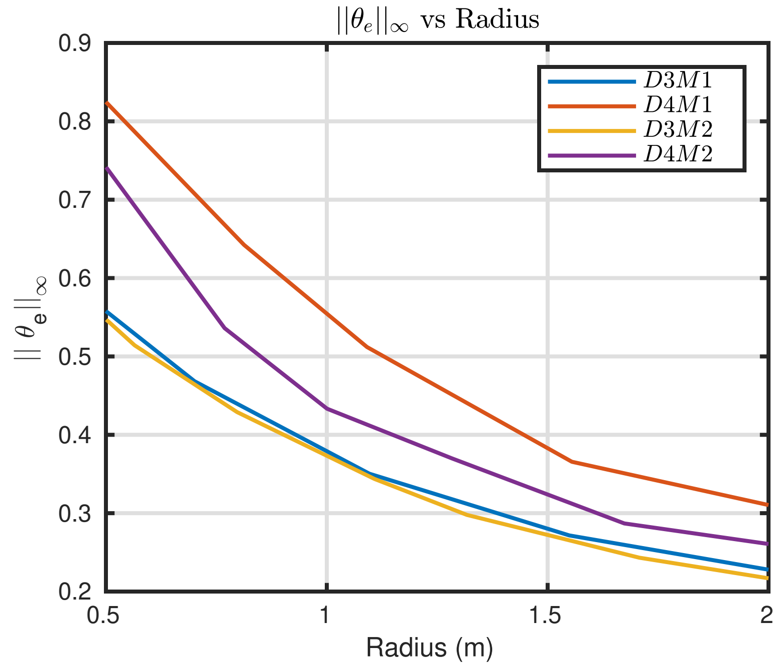


Figure 5.32: $||\theta_e||_\infty$ vs Radius of Track: Hardware Results

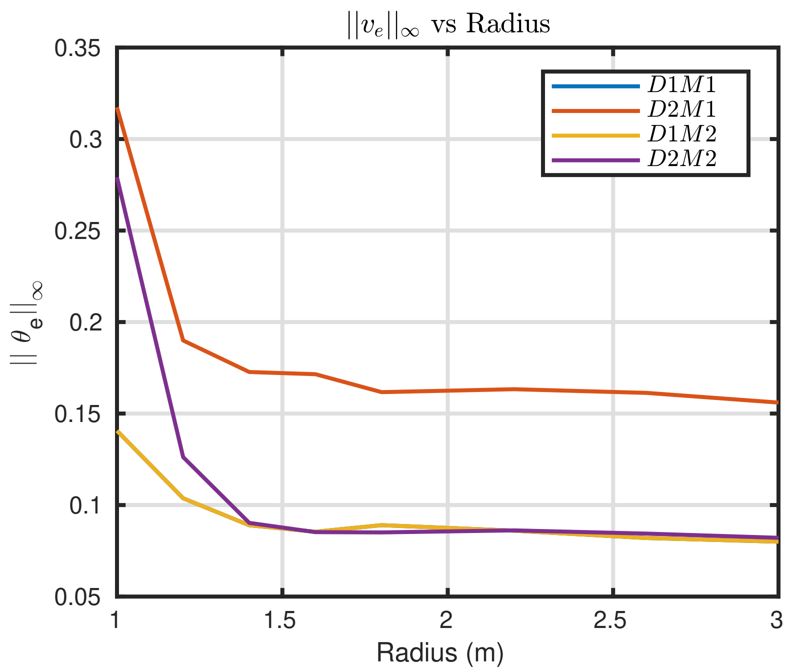


Figure 5.33: $||v_e||_\infty$ vs Radius of Track: Simulation Results

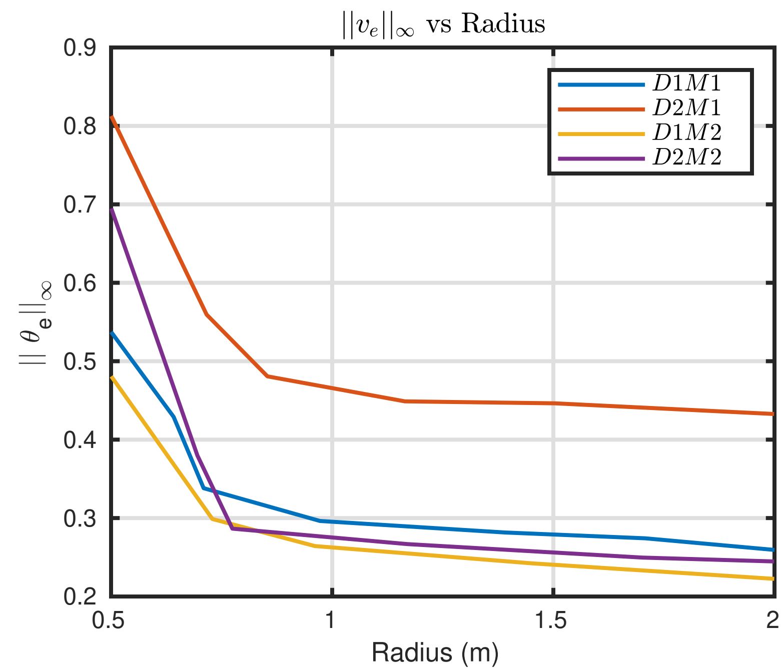


Figure 5.34: $||v_e||_\infty$ vs Radius of Track: Hardware Results

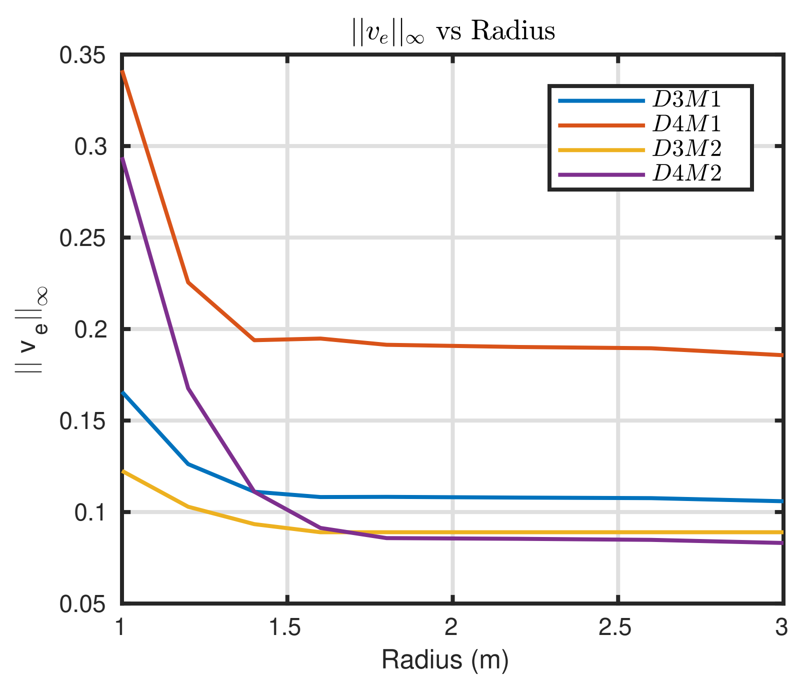


Figure 5.35: $||v_e||_\infty$ vs Radius of Track: Simulation Results

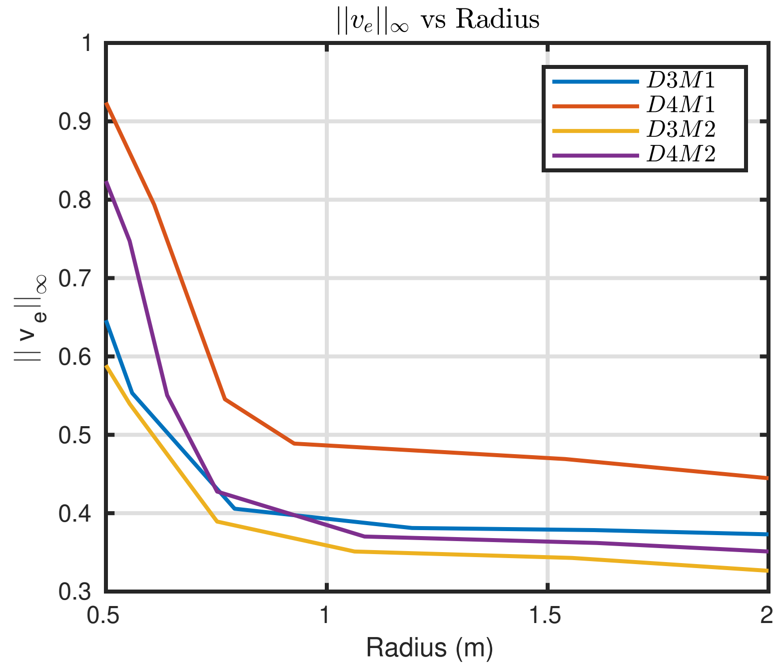


Figure 5.36: $||v_e||_\infty$ vs Radius of Track: Hardware Results

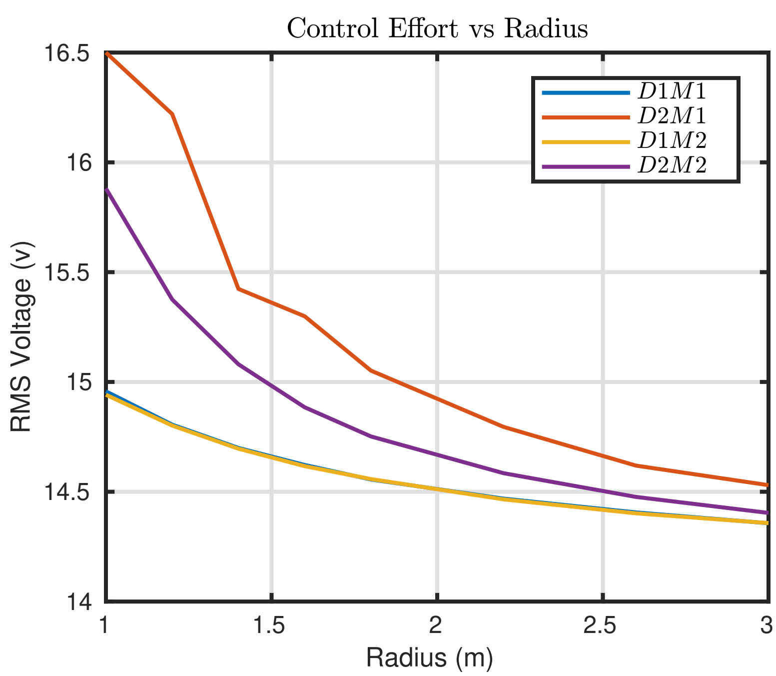


Figure 5.37: Control Effort vs Radius of Track: Simulation Results

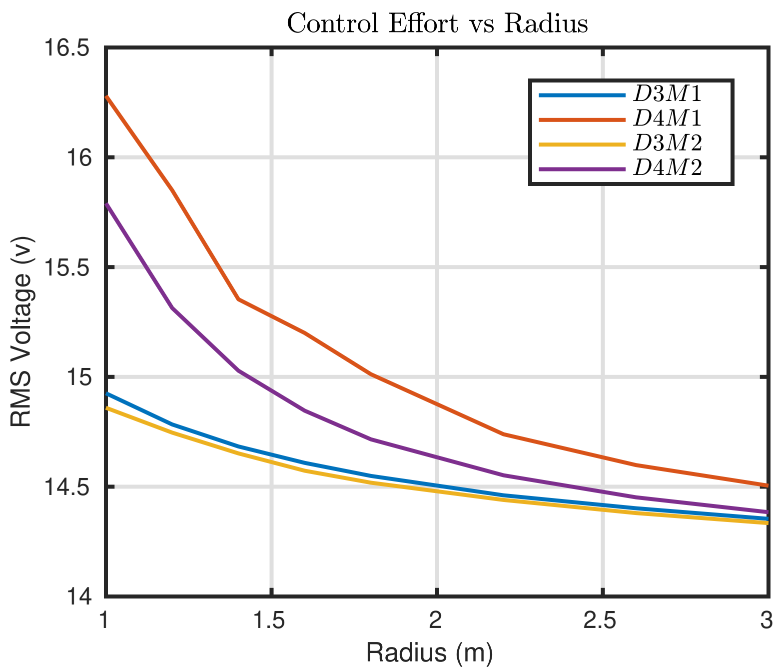


Figure 5.38: Control Effort vs Radius of Track: Simulation Results

From Figures 5.29 - 5.38, the following observations can be made:

- Comparing simulation and hardware results it can be noticed that the eight design variations follow the same trend, however, the deviations in errors corresponding to different designs are more significant in case of hardware results.
- In both simulation and hardware plots, reducing the radius of curvature R causes an increase in $||v_e||_\infty, ||\theta_e||_\infty$, and RMS Voltage irrespective of the properties of each of the eight systems.
- From $||v_e||_\infty, ||\theta_e||_\infty$ vs radius of curvature plots, it can be seen that the systems with higher moment of inertia i.e. D2M1, D2M2, D4M1, and D4M2, exhibit a steep increase in errors and control effort, when compared to the other systems, with a decrease in radius of curvature $R \leq 0.75$ m, at a constant $v_{ref} = 1$ m/s.

- For $R \geq 0.75$ m, it can be noticed that systems with higher input-output coupling at lower frequencies i.e. D2M1 and D4M1, exhibit higher errors $||v_e||_\infty, ||\theta_e||_\infty$ and control effort when when compared to other other systems.

- For D2M2, we notice that $||v_e||_\infty \leq 0.4, ||\theta_e||_\infty \leq 0.44$ at $v_{ref} = 1$ m/s and $B_i = 10$ rad/sec. This means that as long as tracking velocity $v_{ref} \leq 1$ m/s and inner-loop bandwidth $B_i = 10$ rad/sec, the trajectory tracking performance will not be affected significantly for variations in radius of curvature $0.75 \leq R$ m.

- For D4M2, we can see that $||v_e||_\infty \leq 0.4, ||\theta_e||_\infty \leq 0.44$ for $2 \geq R \geq 1$, at $v_{ref} \leq 1$ m/s and $B_i = 10$ rad/sec. Within this radius of curvature range $2 \geq R \geq 1$ it can be seen that D4M2 performance is similar to that of D2M2, but steeply increases for $R < 1$ m.

- For $R > 1$ m, a SISO controller is sufficient to provide good trajectory tracking properties for a system with input-output coupling, at $v_{ref} \geq 1$ m/s.

- However, for $R < 1$ m, a MIMO controller is necessary for systems to achieve similar performance.

5.4.2 *Outer-Loop 2: Planar (x, y) Cartesian Stabilization*

In this section, we will show the design and implementation of the planar (x, y, θ) outer-loop control law [85]. Figures 5.39 and 5.40 show the block diagram representations of the closed loop system implementation for both $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ and $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \rightarrow [v, \omega]}$ plants in the inner-loop. Here, (x, y, θ) are obtained from the HTC Vive Motion capture system, and the data is passed through a moving average filter before passing it to the feedback loop. The (x_{ref}, y_{ref}) commands are predetermined based on the reference velocity, sampling rate, length, and radius of the trajectory.

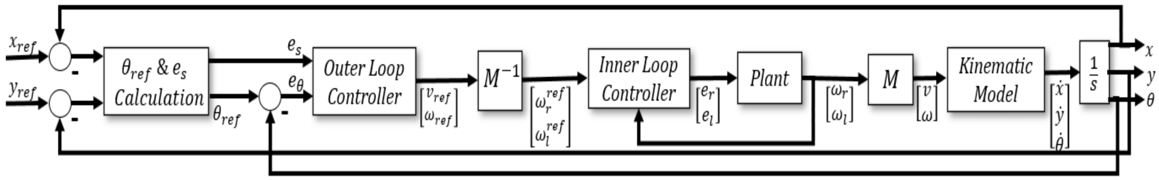


Figure 5.39: $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ System Outer-Loop Control Block Diagram

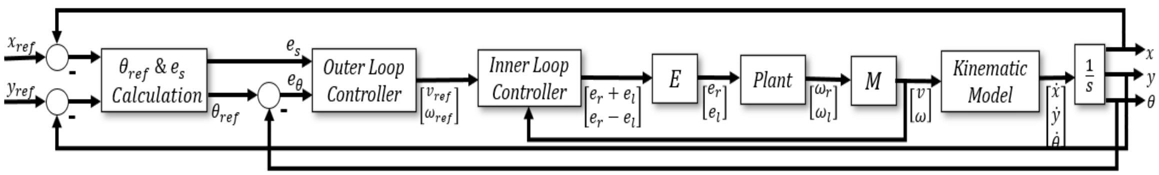


Figure 5.40: $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ System Outer-Loop Control Block Diagram

Figure 5.39 shows the notations used to define the error dynamics of the system. Here e_s represents the distance between the desired position and actual position of the vehicle, e_θ represents the angle between the desired longitudinal axis orientation and actual longitudinal axis orientation of the vehicle. The non-linear error dynamics of this system can be expressed using the following equations

$$\begin{bmatrix} \dot{e}_s \\ \dot{e}_\theta \end{bmatrix} = \begin{bmatrix} -1 & \tan(e_\theta)e_s \\ \frac{\sin(e_\theta)\cos(e_\theta)}{e_s} & -1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{5.14}$$

Let us consider the proportional control law as shown in [85], [46] - which is as follows

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} k_s \\ k_\theta \end{bmatrix} \begin{bmatrix} e_s \\ e_\theta \end{bmatrix} \tag{5.15}$$

Substituting the above equation in the non-linear error dynamics and linearizing them about the equilibrium $e_s = e_\theta = 0$ yields the following

$$\begin{bmatrix} \dot{e}_s \\ \dot{e}_\theta \end{bmatrix} = \begin{bmatrix} -k_s & 0 \\ 0 & k_s - k_\theta \end{bmatrix} \begin{bmatrix} e_s \\ e_\theta \end{bmatrix} \tag{5.16}$$

These linearized error dynamics will be exponentially stable (local stability at the equilibrium point) if $k_\theta > k_s > 0$. As mentioned in [45], the use of a proportional controller is justified as long as the bandwidth of the outer-loop is less than that of the inner-loop by a factor of 5 ($BW_{outerloop} \leq 0.2\, BW_{innerloop}$).

The following figures show the variation in x_e, y_e, θ_e and RMS Voltage (control effort) with respect to changes in the radius of the track, and reference velocity - for each of the eight design variations.

Increasing Tracking Velocity (v_{ref}) for Fixed Radius of Curvature of Trajectory (R). The simulation and hardware data presented in Figure 5.41 - 5.50 are obtained at inner-loop bandwidth B_i = 10 rad/sec and radius of track R = 1.5 m while varying the trajectory tracking velocity. The hardware results had to be limited to trajectory tracking velocity $v_{ref} \leq 2$ m/s due to the physical restrictions of the experimental setup.

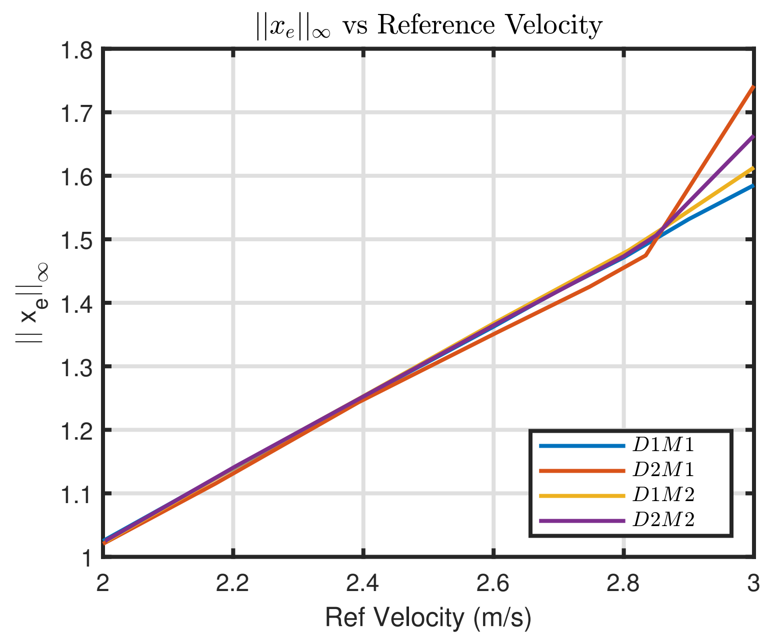


Figure 5.41: $||x_e||_\infty$ vs Reference Velocity: Simulation Results

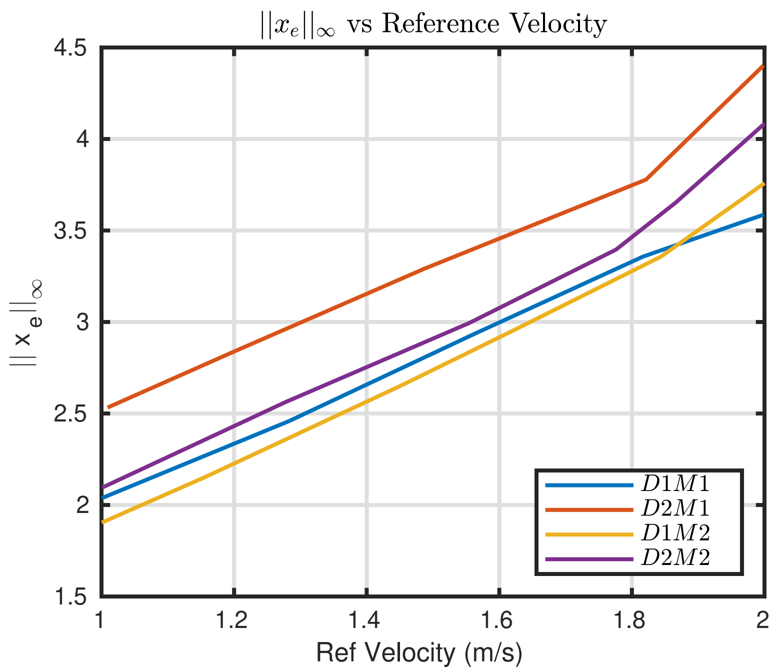


Figure 5.42: $||x_e||_\infty$ vs Reference Velocity: Hardware Results

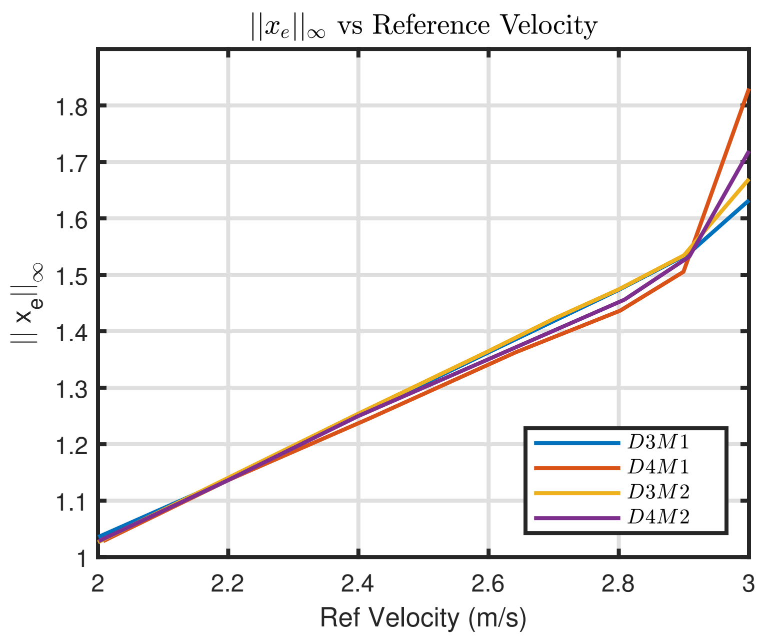


Figure 5.43: $||x_e||_\infty$ vs Reference Velocity: Simulation Results

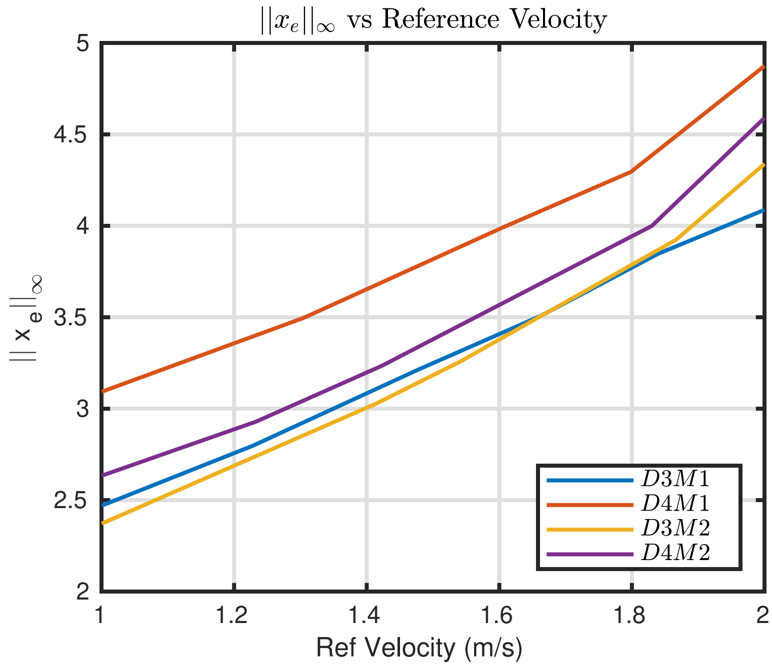


Figure 5.44: $||x_e||_\infty$ vs Reference Velocity: Hardware Results

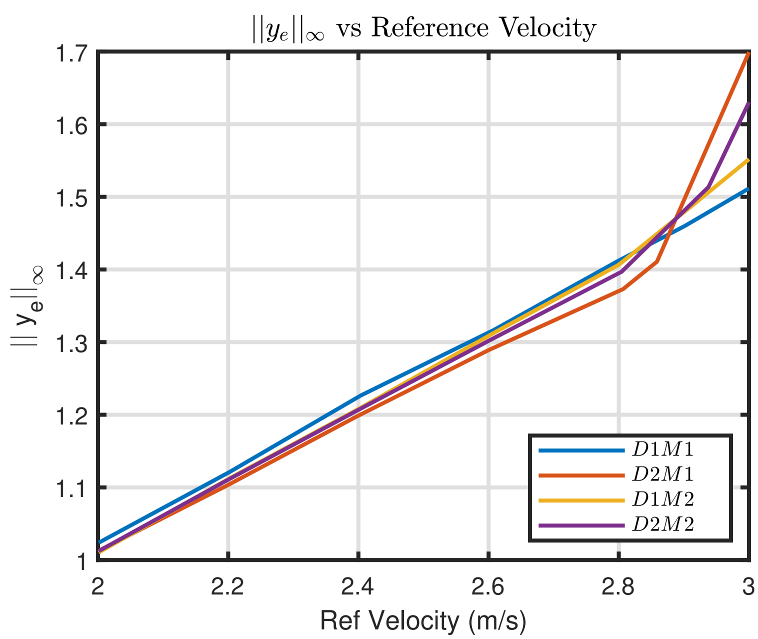


Figure 5.45: $||y_e||_\infty$ vs Reference Velocity: Simulation Results

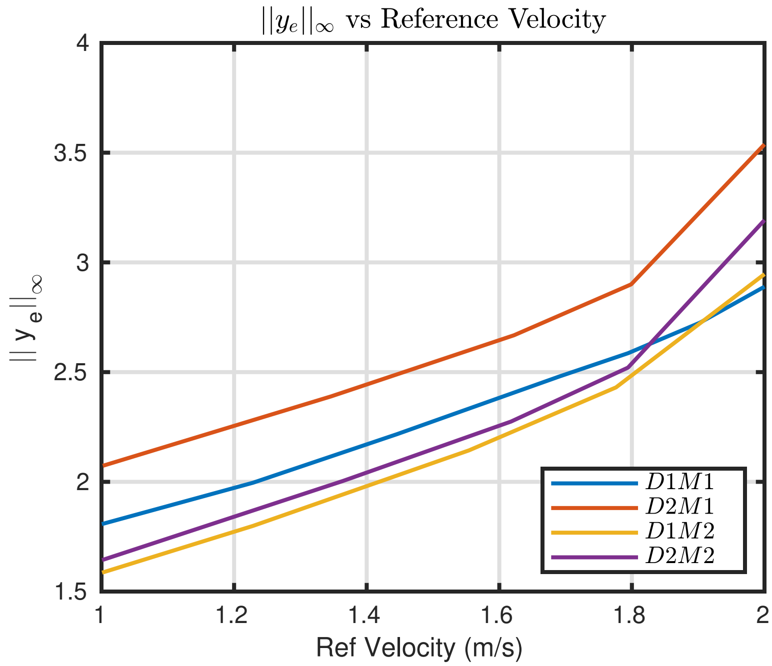


Figure 5.46: $||y_e||_\infty$ vs Reference Velocity: Hardware Results

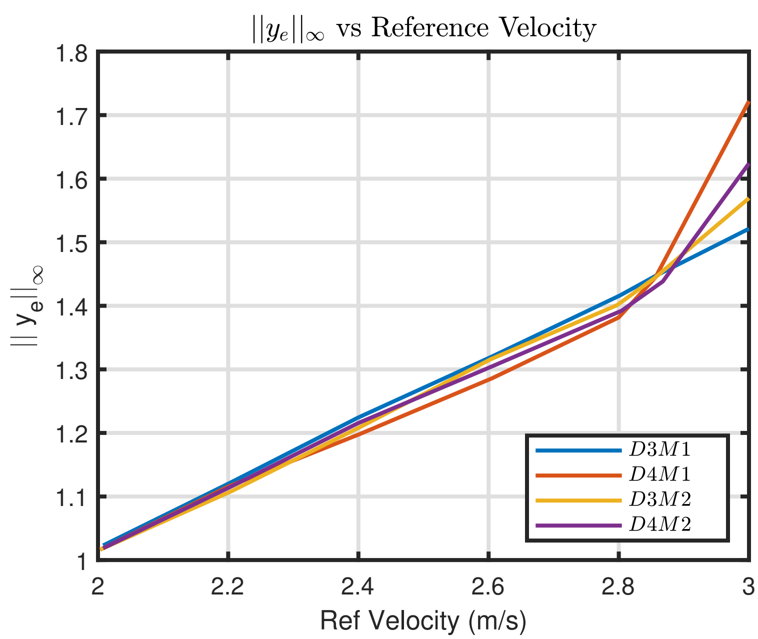


Figure 5.47: $||y_e||_\infty$ vs Reference Velocity: Simulation Results

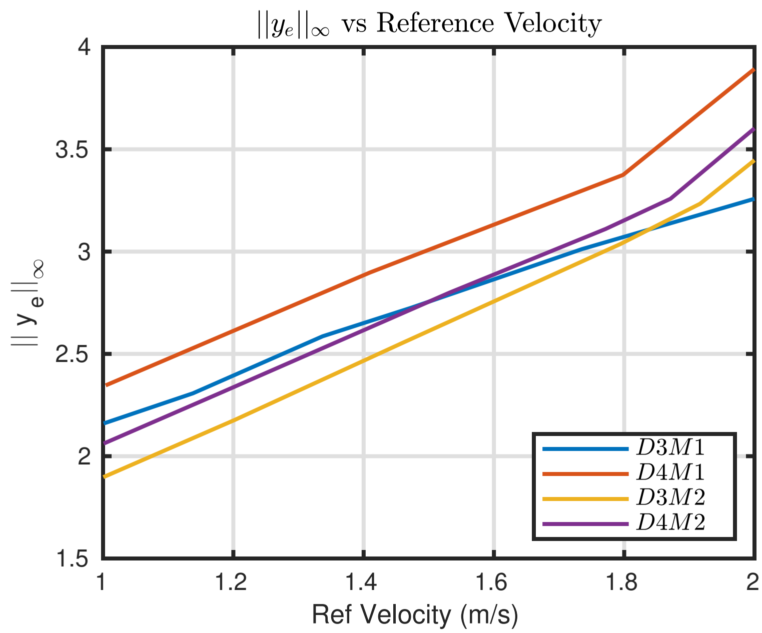


Figure 5.48: $||y_e||_\infty$ vs Reference Velocity: Hardware Results

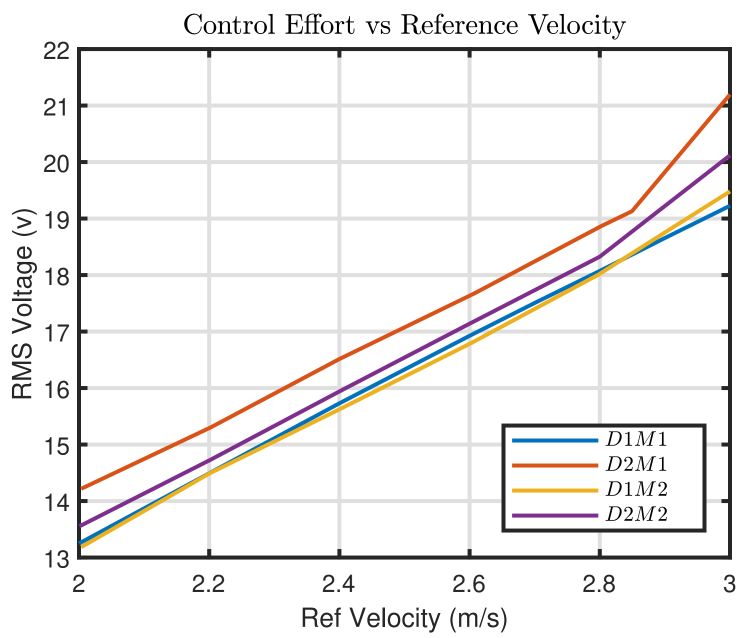


Figure 5.49: Control Effort vs Reference Velocity: Simulation Results

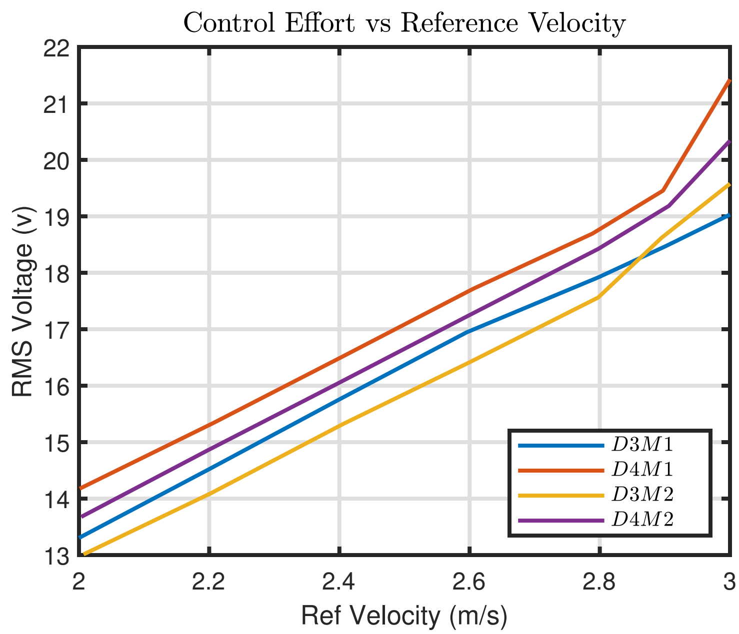


Figure 5.50: Control Effort vs Reference Velocity: Simulation Results

From Figures 5.41 - 5.50, the following observations can be made:

- Comparing simulation and hardware results it can be noticed that the eight design variations follow the same trend, however, the deviations in errors corresponding to different designs are more significant in case of hardware results.

- In both simulation and hardware plots, increasing the trajectory tracking velocity v_{ref} causes an increase in $||x_e||_\infty, ||y_e||_\infty$, RMS Voltage irrespective of the properties of each of the eight systems.

- From $||x_e||_\infty, ||y_e||_\infty$ vs trajectory tracking velocity plots, it can be seen that the systems with higher moment of inertia i.e.D2M1, D2M2, D4M1, and D4M2, exhibit a steep increase in errors and control effort, when compared to the other systems, with an increase in trajectory tracking velocity $v_{ref} \geq 1.8$ m/s, at a constant $R = 1.5$ m.

- For $v_{ref} \leq 1.8$ m/s, it can be noticed that systems with higher input-output coupling at lower frequencies i.e. D2M1 and D4M1, exhibit higher errors $||x_e||_\infty, ||y_e||_\infty$ and control effort when when compared to other other systems.

- For D2M2, we notice that $||x_e||_\infty \leq 3.5, ||y_e||_\infty \leq 2.5$ at $R = 1.5$ m and $B_i = 10$ rad/sec. This means that as long as radius of curvature $R \geq 1.5$ m and inner-loop bandwidth $B_i = 10$ rad/sec, the trajectory tracking performance will not be affected significantly for variations in reference velocity $v_{ref} \leq 1.8$ m/s.

- For D4M2, we can see that $||x_e||_\infty \leq 3.5, ||y_e||_\infty \leq 2.5$ for $1 \geq v_{ref} \geq 1.35$, at $R \geq 1.5$ m and $B_i = 10$ rad/sec. Within this reference tracking velocity range $1 \leq v_{ref} \leq 1.35$ it can be seen that D4M2 performance is similar to that of D2M2, but steeply increases for $v_{ref} > 1.35$ m/s.

- For $v_{ref} < 1.35$ m/s, a SISO controller is sufficient to provide good trajectory tracking properties for a system with input-output coupling, at $R \geq 1.5$ m.

- However, for $v_{ref} > 1.35$ m/s, a MIMO controller is necessary to achieve a similar performance.

Varying Radius of Curvature of Trajectory (R) for Fixed Tracking Velocity (v_{ref}). The simulation and hardware data presented in Figure 5.51 - 5.59 are obtained at inner-loop bandwidth $B_i = 10$ rad/sec and trajectory tracking velocity $v_{ref} = 1$ m/s while varying the radius of curvature. The hardware results had to be limited to radius of curvature $R \leq 2$ m due to the physical restrictions of the experimental setup.

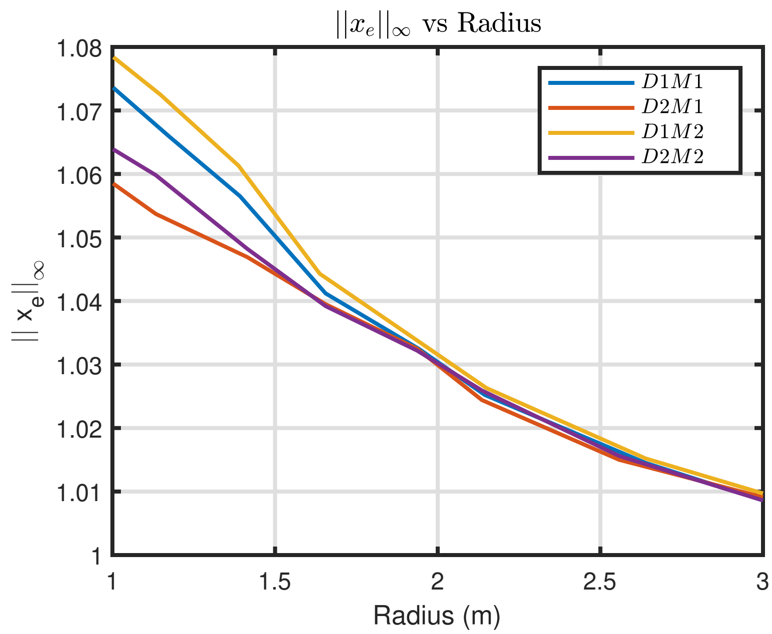


Figure 5.51: $||x_e||_\infty$ vs Radius of Track: Simulation Results

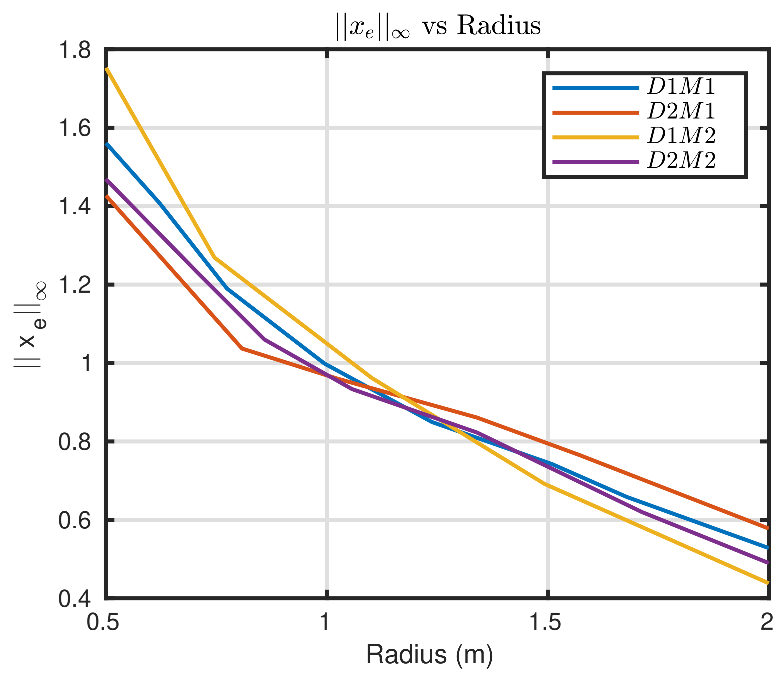


Figure 5.52: $||x_e||_\infty$ vs Radius of Track: Hardware Results

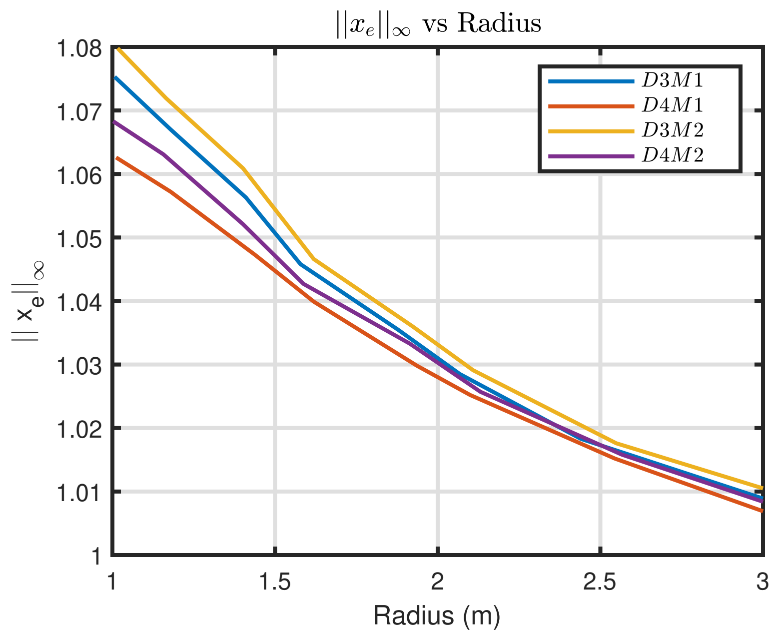


Figure 5.53: $||x_e||_\infty$ vs Radius of Track: Simulation Results

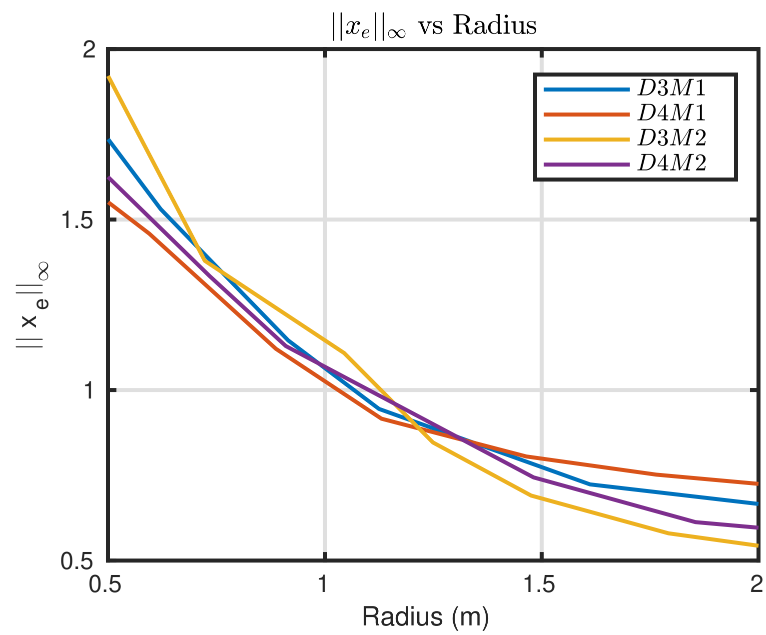


Figure 5.54: $||x_e||_\infty$ vs Radius of Track: Hardware Results

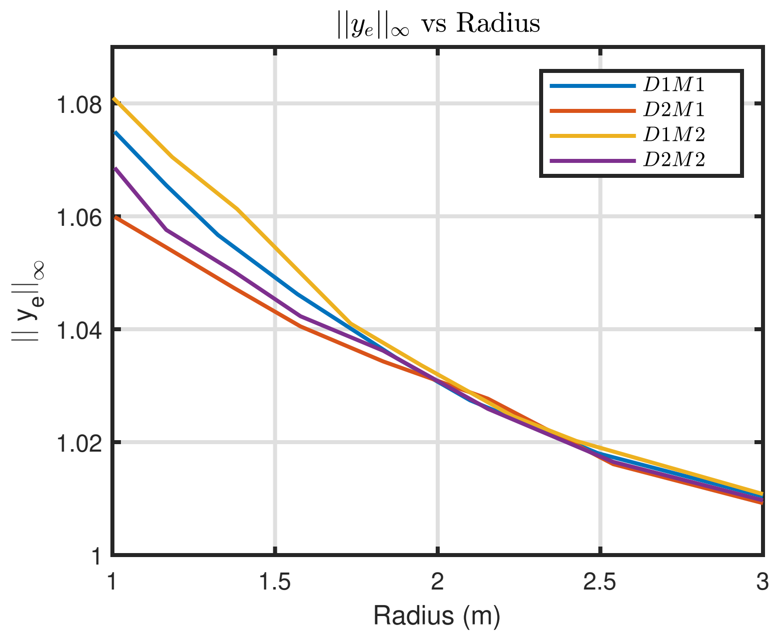


Figure 5.55: $||y_e||_\infty$ vs Radius of Track: Simulation Results

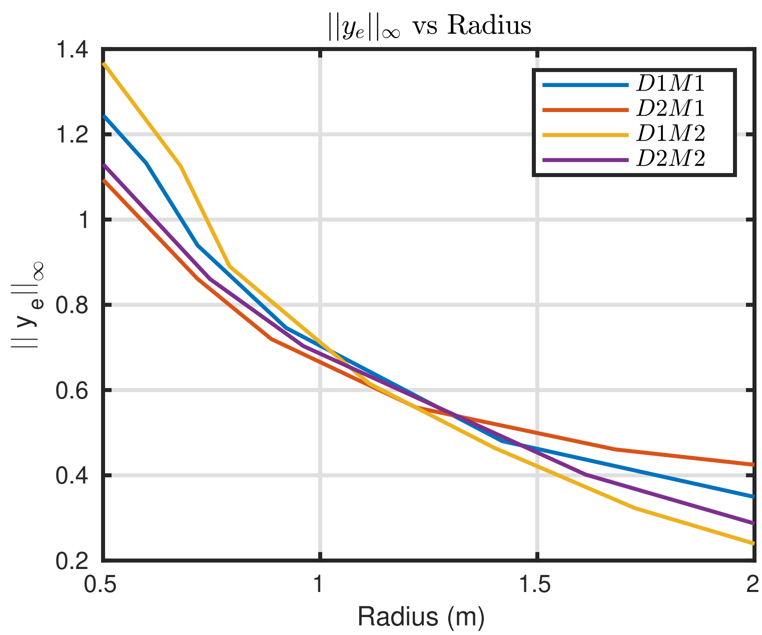


Figure 5.56: $||y_e||_\infty$ vs Radius of Track: Hardware Results

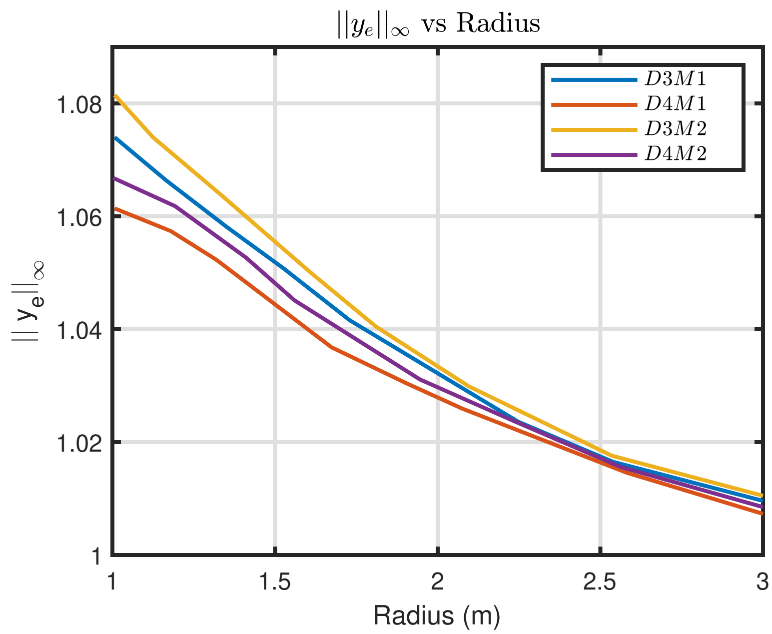


Figure 5.57: $||y_e||_\infty$ vs Radius of Track: Simulation Results

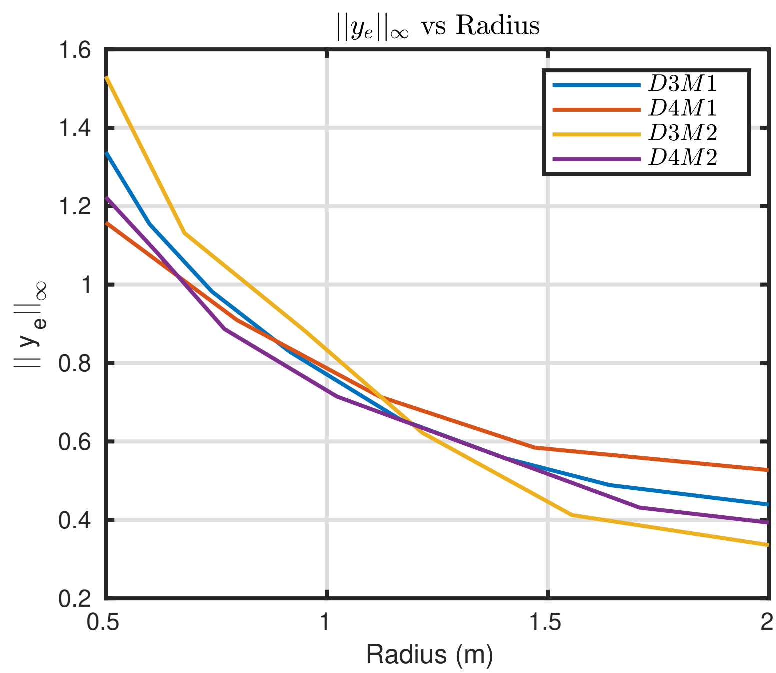


Figure 5.58: $||y_e||_\infty$ vs Radius of Track: Hardware Results

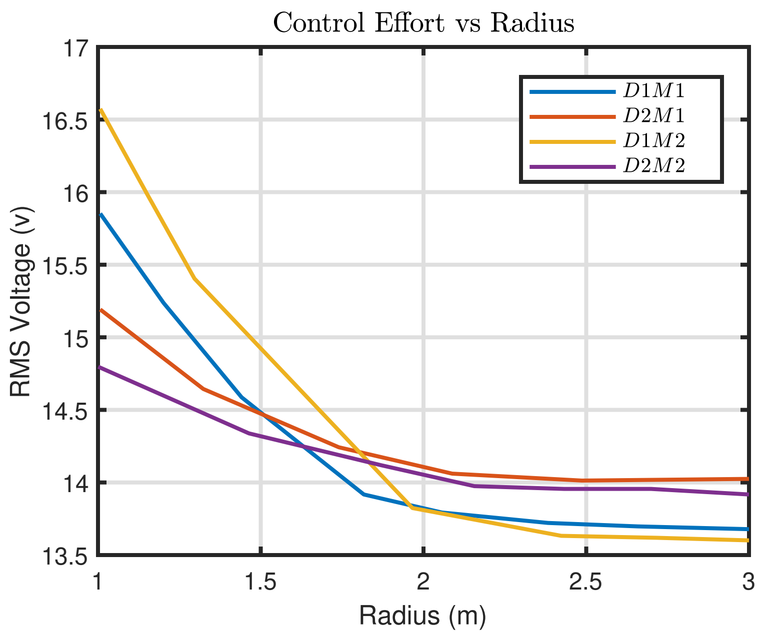


Figure 5.59: Control Effort vs Radius of Track: Simulation Results

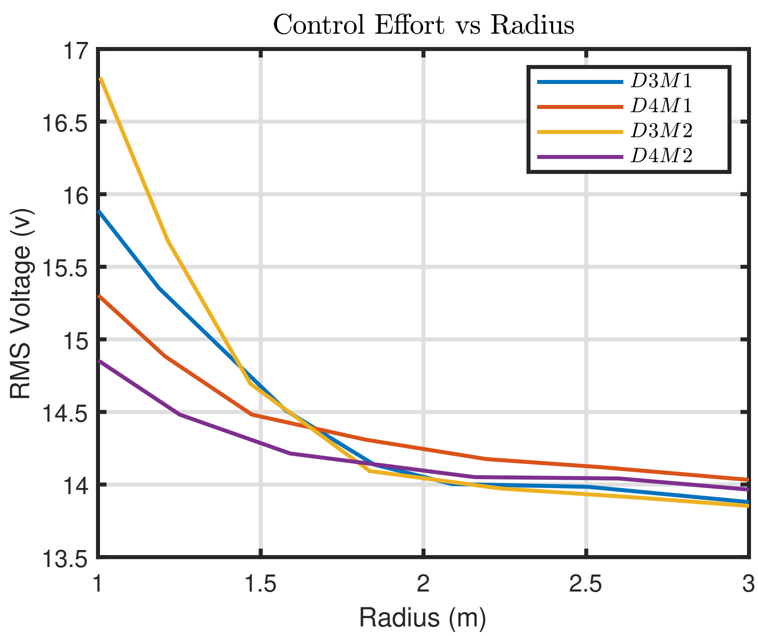


Figure 5.60: Control Effort vs Radius of Track: Simulation Results

From Figures 5.51 - 5.59, the following observations can be made:

- Comparing simulation and hardware results it can be noticed that the eight design variations follow the same trend, however, the deviations in errors corresponding to different designs are more significant in case of hardware results.

- In both simulation and hardware plots, reducing the radius of curvature R causes an increase in $||x_e||_\infty, ||_e||_\infty$, and RMS Voltage irrespective of the properties of each of the eight systems.

- From $||x_e||_\infty, ||y_e||_\infty$ vs radius of curvature plots, it can be seen that the systems with more stable plants i.e. D1M1, D1M2, D3M2, and D3M1, exhibit higher errors and control effort, when compared to the other systems, with a decrease in radius of curvature $R \leq 1.25$ m, at a constant $v_{ref} = 1$ m/s.

- For $R \geq 1.25$ m, it can be noticed that systems with higher input-output coupling at lower frequencies i.e. D2M1 and D4M1, exhibit higher errors $||x_e||_\infty, ||y_e||_\infty$ and control effort when when compared to other other systems.

- For D2M2, we notice that $||x_e||_\infty \leq 1.0, ||y_e||_\infty \leq 0.7$ at $v_{ref} = 1$ m/s and $B_i = 10$ rad/sec. This means that as long as tracking velocity $v_{ref} \leq 1$ m/s and inner-loop bandwidth $B_i = 10$ rad/sec, the trajectory tracking performance will not be affected significantly for variations in radius of curvature $1 \leq R$ m.

- For D4M2, we can see that $||x_e||_\infty \leq 1.0, ||y_e||_\infty \leq 0.7$ for $2 \geq R \geq 1.8$, at $v_{ref} \leq 1$ m/s and $B_i = 10$ rad/sec. Within this radius of curvature range $2 \geq R \geq 1$ it can be seen that D4M2 performance is similar to that of D2M2, but steeply increases for $R < 1.8$ m.

- For $R > 1.8$ m, a SISO controller is sufficient to provide good trajectory tracking properties for a system with input-output coupling, at $v_{ref} \leq 1$ m/s.

- However, for $R < 1.8$ m, a MIMO controller is necessary to achieve similar performance.

Chapter 6

MULTI-ROBOT FORMATION CONTROL USING RECEDING HORIZON OPTIMIZATION

6.1 Introduction and Overview

Over the past two decades, the area of multi-robot control has received a significant amount of attention. This surge in research efforts is because of the fact that a group of multi-robot systems, under well-defined control and coordination principles, can behave like a single entity and exhibit a high level of fault tolerance and robustness when compared to single robot systems [35]. A multi-robot fleet can accomplish tasks that would be highly impossible for a single robot system. These tasks include large area exploration [15], surveillance and mapping [82], object transportation [88], construction and manufacturing [80].

The basis for achieving all these high-level objectives mentioned above includes some of the fundamental tasks such as multi-robot trajectory tracking, longitudinal platooning, formation control and, static and dynamic obstacle avoidance. There are various approaches to implement these tasks and some of the most common ones include leader-follower-based, virtual structure-based, and behavior-based. In the virtual structure approach, the whole formation is treated as a single structure, and the desired motion of each element in this virtual structure is converted to the desired trajectories that have to be followed by each robot in the formation. Whereas in the behavior-based approach, each robot is pre-assigned with several desired behaviors, and the final control is derived by weighing each of the individual robot behaviors. In the leader-follower approach, one of the robots acts as a leader and all other robots have to maintain a fixed distance and orientation with respect to the leader. Here, only the leader pose information, and the desired relative distance and orientation information have to be passed to the individual followers, and each of the followers has a local control law that would enable them to maintain the desired relative position and orientation. Consequently, the formation control problem can be viewed as a natural extension of the single robot trajectory tracking problem [44]. Therefore, the approaches used for a single-robot trajectory tracking problem can be extended to design the control laws for the leader-follower approach. For this reason, we would be considering the leader-follower approach within this thesis.

A single DDV trajectory tracking problem has always caught the eye of the researchers because of the challenges it imposed due to the under-actuated, non-linear, and multivariable nature of the dynamical model [57]. According to Brockett's theorem [14], it would be impossible to design a smooth, time-invariant, and continuous feedback control law in order to asymptotically stabilize the non-holonomic system in a given configuration. Given this, several methods are proposed overtime to control this system, they include, receding horizon optimization approaches [16], feedback linearization approaches [56], time-varying control approaches [70], discontinuous time-varying feedback [6], etc. One of the common problems with traditional trajectory tracking controllers [36] is that it would not be possible to include additional control objectives such as obstacle avoidance; optimizing performance parameters such as total control effort, tracking time, etc; or to incorporate constraints on states or output variables that are fundamental to trajectory tracking in real-world scenarios. For this reason, optimization-based approaches are gaining prominence because they systematically address these limitations.

In the receding horizon approach, an optimization problem is solved at every time step in order to generate a finite control sequence that would minimize the tracking error over a finite horizon while subject to the constraints imposed by the prediction model, input/output parameters, and control parameters. Depending on the nature of the objective function and the constraints imposed, the optimization problem can be further classified into linear or non-linear optimization. While non-linear optimization is computationally intensive due to the NP-hard nature of the optimization, linear optimization approaches are widely preferred because of their comparatively less expensive computational demands and also due to the existence of a global solution to the quadratic/linear objective functions. Albeit the linear optimization approaches are highly successful and widely used in several applications, in most of the literature available, the optimization problem is formulated based on only the kinematic model, and the dynamics of the system are completely ignored. In these approaches, the inner-loop speed control system is assumed to offer perfect tracking i.e. infinite bandwidth. This is clearly not the case with real-world systems because every actuator or a real-world system will have limitations, and therefore it's incorrect to consider perfect inner-loop tracking because an actuator will never produce the instantaneous speeds for a given input voltage. Moreover, several practical effects such as input voltage dead-zone, minimum actuator reaction time (or bandwidth), actuator saturation, or high-frequency noise are not modeled within a kinematic model. Hence, it is necessary to include the constraints imposed by the dynamical model in addition to those of the kinematic model in order to improve the performance of the trajectory tracking controller. A drawback of including the constraints imposed by the dynamical model along with that of the kinematic model is that increased computation load.

In this chapter, we try to answer when a kinematic model would be sufficient? and when a kinematic plus dynamical model is necessary for trajectory tracking?. In order to answer this, we examine the two different optimization problem formulations to understand their impact on the performance of trajectory tracking i.e. 1) with only kinematic model-based constraints, 2) with both kinematic and dynamical model-based constraints. In Section 6.2, we provide the multi-robot problem formulation based on the leader-follower approach, and in Section 6.3 we present the simulation and hardware performance results for the two different optimization formulations. Table 6.1 shows the summary of various performance trade studies that were conducted and Figure 6.1 shows the reference trajectory that was considered for these trade studies.

	Kinematic Constraints	**Kin + Dyn Constraints**
$\|\|x_e\|\|_\infty$ vs v_{ref}	*	*
$\|\|y_e\|\|_\infty$ vs v_{ref}	*	*
$\|\|\theta_e\|\|_\infty$ vs v_{ref} [1]	*	*
$\|\|x_e\|\|_\infty$ vs R	*	*
$\|\|y_e\|\|_\infty$ vs R	*	*
$\|\|\theta_e\|\|_\infty$ vs R [2]	*	*
$\|\|x_e\|\|_\infty$ vs BW	*	*
$\|\|y_e\|\|_\infty$ vs BW	*	*
$\|\|\theta_e\|\|_\infty$ vs BW [3]	*	*

[1] v_{ref} - tracking velocity

[2] R - radius of the track

[3] BW - bandwidth of inner-loop

Table 6.1: Summary of Trade Studies Conducted

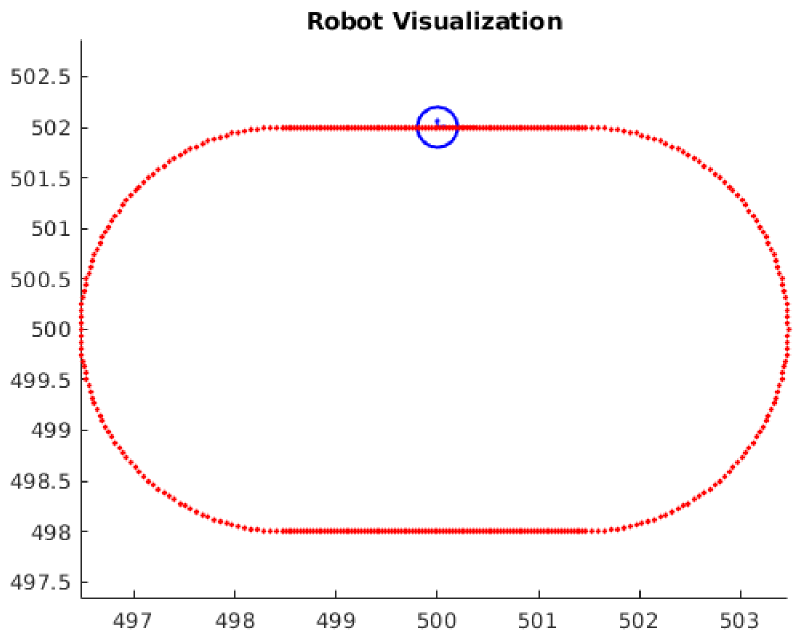


Figure 6.1: Reference Trajectory Visualization

6.2 Problem Formulation: Leader-Follower Approach

Consider a group of N non-holonomic differential drive robots denoted by the subscripts i and j, where $i \in \{2, 3, ..., N\}$ represents the follower robots and $j \in \{1\}$ represents the leader robot. The goal of the follower i is to always maintain a fixed distance $l_{i,j}^d$ and orientation $\theta_{i,j}^d$ with respect to the leader j and Figure 6.2 represents this leader-follower formulation of the robots. The pose information of the leader is always available to the follower along with the relative distance $l_{i,j}^d$ and orientation $\theta_{i,j}^d$. Using this information, the desired pose of the follower $(x_i^d, y_i^d, \theta_i^d)$ can be calculated as

$$\begin{bmatrix} x_i^d \\ y_i^d \\ \theta_i^d \end{bmatrix} = \begin{bmatrix} x_j + l_{i,j}^d \cos\theta_{i,j}^d \\ y_j + l_{i,j}^d \sin\theta_{i,j}^d \\ \theta_j \end{bmatrix} \tag{6.1}$$

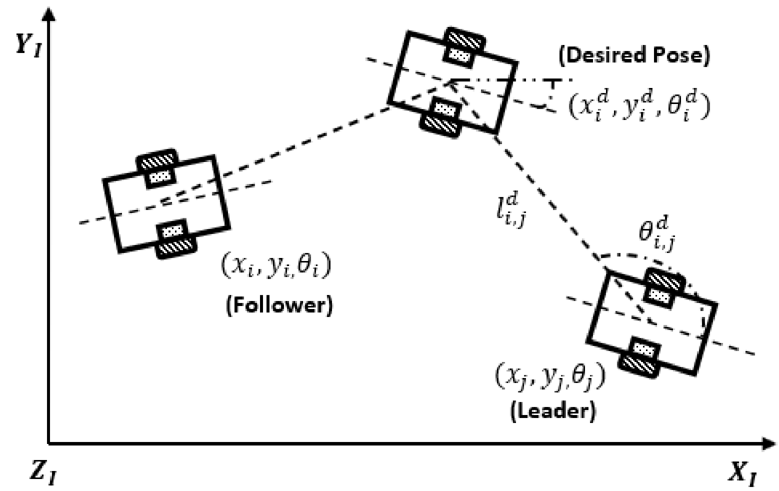


Figure 6.2: Multi-Robot Leader-Follower Formulation

where (x_i, y_i, θ_i) represent the current pose of the follower. The desired pose $(x_i^d, y_i^d, \theta_i^d)$ is given as the input reference command to the optimization-based outer-loop controller that is being implemented in each follower robots. Figure 6.3 represents the implementation of the outer-loop control law in each of the follower robots.

6.2.1 Prediction Model

The kinematic model along with inner-loop control system i.e. $(\omega_{r,i}^{ref}, \omega_{l,i}^{ref}) \rightarrow (\omega_{r,i}, \omega_{l,i})$, forms the prediction model for the optimization problem. The kinematic model of the follower robot can be represented as follows:

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \\ 0 \end{bmatrix} v_i + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega_i \tag{6.2}$$

$$\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{d_w} & -\frac{r}{d_w} \end{bmatrix} \begin{bmatrix} \omega_{r,i} \\ \omega_{l,i} \end{bmatrix} \tag{6.3}$$

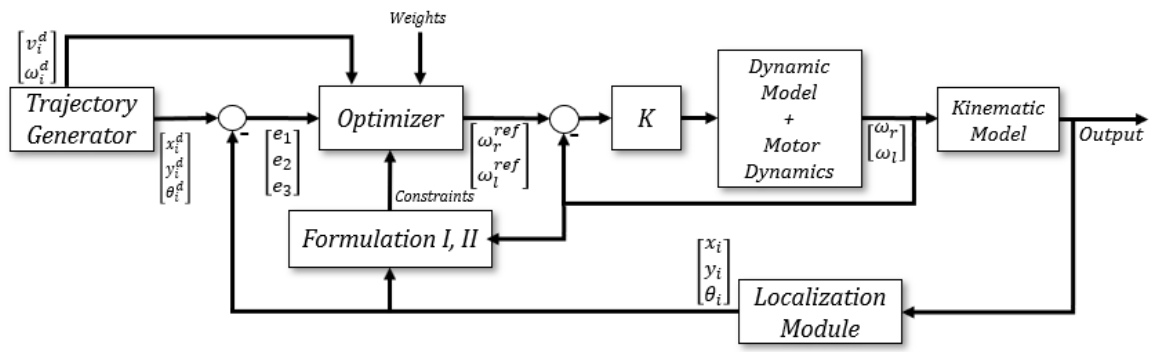


Figure 6.3: Closed-Loop Block Diagram Representation

here, (v_i, ω_i) represent the linear and angular velocity of the follower, $(\omega_{r,i}, \omega_{l,i})$ represent the left and right wheel angular velocities respectively, r represents the radius of the wheel and d_w represents the width [4] of the DDV. In order to incorporate the kinematic model into the optimization problem, it has to be linearized about the time varying local operation points $(x_i^{op}, y_i^{op}, \theta_i^{op})$. The linearized kinematic model is given by

$$\dot{x}_i = [-\frac{r}{2}\omega_{r,i}^{op}\sin\theta_i^{op} - \frac{r}{2}\omega_{l,i}^{op}\sin\theta_i^{op}]\theta_i + [\frac{r}{2}\cos\theta_i^{op}]\omega_{r,i} + [\frac{r}{2}\cos\theta_i^{op}]\omega_{l,i} \tag{6.4}$$

$$\dot{y}_i = [\frac{r}{2}\omega_{r,i}^{op}\cos\theta_i^{op} + \frac{r}{2}\omega_{l,i}^{op}\cos\theta_i^{op}]\theta_i + [\frac{r}{2}\sin\theta_i^{op}]\omega_{r,i} + [\frac{r}{2}\sin\theta_i^{op}]\omega_{l,i} \tag{6.5}$$

$$\dot{\theta}_i = \frac{r}{d_w}\omega_{r,i} - \frac{r}{d_w}\omega_{l,i} \tag{6.6}$$

For nominal plant parameters i.e. $d = 0$, the inner-loop control system can be approximated as a first order transfer function $[\omega_{r,i}, \omega_{l,i}] = diag(\frac{B_i}{s+B_i}, \frac{B_i}{s+B_i})[\omega_{r,i}^{ref}, \omega_{l,i}^{ref}]$, where B_i represents the bandwidth of the inner-loop. This simple first order approximation is a direct consequence of the well designed inner loop PI controller. In time domain, this inner-loop system can be expressed as first order ODEs as

$$\dot{\omega}_{r,i} = -B_i\omega_{r,i} + B_i\omega_{r,i}^{ref} \tag{6.7}$$

$$\dot{\omega}_{l,i} = -B_i\omega_{l,i} + B_i\omega_{l,i}^{ref} \tag{6.8}$$

[4] the distance between the two wheels (at midpoint)

6.2.2 Trajectory Tracking

The fundamental thought behind trajectory tracking is to ensure that the error between the current follower pose (x_i, y_i, θ_i) and the desired pose $(x_i^d, y_i^d, \theta_i^d)$ converges to zero. The following equations represent the error between the desired and the actual pose of the follower (in robot frame of reference), and Figure 6.4 shows the physical interpretation of these errors (e_1, e_2, e_3).

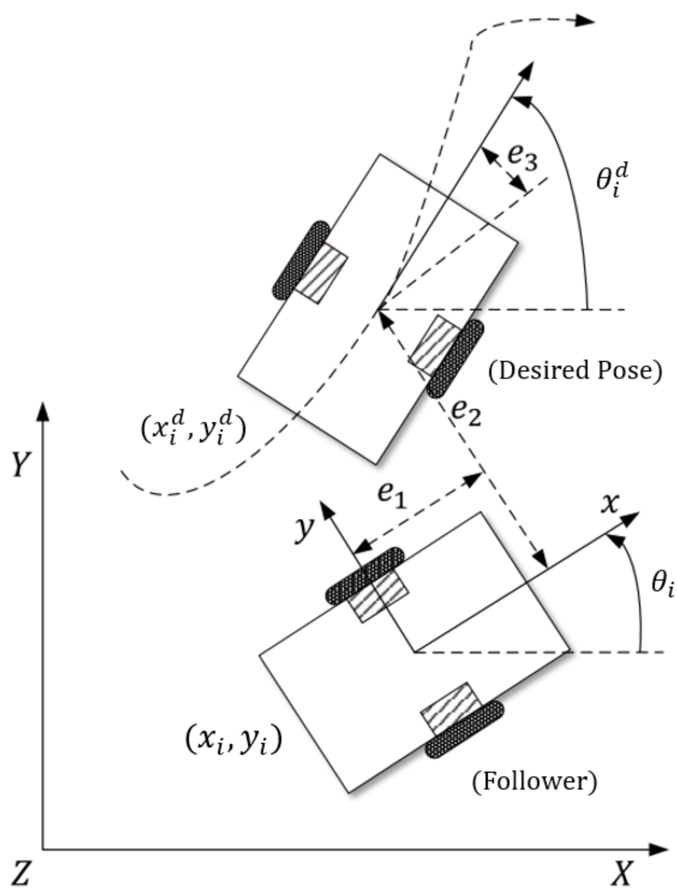


Figure 6.4: Desired and Actual Pose - Error Representation

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\theta_i & \sin\,\theta_i & 0 \\ -\sin\,\theta_i & \cos\,\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i^d - x_i \\ y_i^d - y_i \\ \theta_i^d - \theta_i \end{bmatrix} \tag{6.9}$$

Using the above equations, the non-linear error dynamics can be derived as follows:

$$\dot{e}_1 = v_i^d \cos(\theta_i^d - \theta_i) + [e_2 \frac{r}{d_w} - \frac{r}{2}]\,\omega_{r,i} - [e_2 \frac{r}{d_w} + \frac{r}{2}]\,\omega_{l,i} \tag{6.10}$$

$$\dot{e}_2 = v_i^d \sin(\theta_i^d - \theta_i) - e_1 \frac{d_w}{r}\omega_{r,i} + e_1 \frac{d_w}{r}\omega_{l,i} \tag{6.11}$$

$$\dot{e}_3 = \omega_i^d - \frac{r}{d_w}\omega_{r,i} + \frac{r}{d_w}\omega_{l,i} \tag{6.12}$$

where $v_i^d = \sqrt{(\dot{x}_i^d)^2 + (\dot{y}_i^d)^2}$, $\omega_i^d = \dot{\theta}_i^d$ and $\omega_{r,i}, \omega_{l,i}$ are the right and left wheel angular velocities. These non-linear error dynamics have to be linearized about the time varying local operating points $(e_1, e_2, e_3, \omega_{r,i}, \omega_{l,i})$ before they can be incorporated into the optimization problem. The linearized error dynamics are as follows:

$$\dot{e}_1 = [\frac{r}{d_w}\omega_{r,i}^{op} - \frac{r}{d_w}\omega_{l,i}^{op}]e_2 - [v_i^d \sin(e_3^{op})]e_3 + [e_2^{op}\frac{r}{d_w} - \frac{r}{2}]\omega_{r,i} - [e_2^{op}\frac{r}{d_w} + \frac{r}{2}]\omega_{l,i} \tag{6.13}$$

$$\dot{e}_2 = [-\frac{r}{d_w}\omega_{r,i}^{op} + \frac{r}{d_w}\omega_{l,i}^{op}]e_1 + [v_i^d \cos(e_3^{op})]e_3 - [e_1^{op}\frac{r}{d_w}]\omega_{r,i} + [e_1^{op}\frac{r}{d_w}]\omega_{l,i} \tag{6.14}$$

$$\dot{e}_3 = -\frac{r}{d_w}\omega_{r,i} + \frac{r}{d_w}\omega_{l.i} \tag{6.15}$$

6.2.3 Control Strategy

As shown in Figure 6.3, we employ a hierarchical inner-outer loop control for trajectory tracking. The outer-loop employs a receding horizon optimization scheme which minimizes the quadratic objective function to generate the reference angular velocity commands $(\omega_{r,i}^{ref}, \omega_{l,i}^{ref})$ for the inner-loop controller. The inner-loop system employs a decentralized PI controller (Chapter 5, page 121) to track these reference commands generated by the outer-loop. The receding horizon optimization scheme utilizes a model of the system and the error dynamics of the task to make predictions about the system's future behavior. A quadratic programming library/solver (such as MATLAB's Interior Point Solvers) is utilized to solve for the control actions ($\omega_{r,i}^{ref}, \omega_{l,i}^{ref}$) that can minimize the objective function J while subject to constraints imposed by the prediction model and error dynamics. At every time step/sampling instant, the optimizer produces a control sequence for a given prediction horizon. The first input from the control sequence is applied as input to the inner-loop system; the prediction horizon is shifted by one-time step and the optimization problem is resolved with the updated prediction model and error dynamics.

In order to track aggressive maneuvers, a non-linear optimization method is considered to provide better performance when compared to a linear optimization method. This is because a non-linear optimization method utilizes the non-linear model of the system to predict the system's behavior while minimizing the objective function within the prediction horizon and thereby, it can closely track the non-linear behavior of the trajectory. However, the non-linear optimization methods are computationally intensive due to the NP-hard nature of the problem, and thereby less preferred when it comes to real-time implementation in systems with high-speed dynamics. Therefore, we would be employing the linear optimization method (linear receding horizon optimization) within this thesis. In the linear optimization method, the non-linear prediction model of the system is linearized at local operating points and updated into the optimization problem at every time step/sampling instant in order to capture the non-linear dynamics of the system as closely as possible, this is called as successive online linearization. Furthermore, in linear optimization, the objective function is formulated as a quadratic function which is subject to linear equality/inequality constraints. This ensures that the optimization problem is a quadratic programming problem (QP) that is convex in nature and hence, a globally optimal solution is attainable.

The trajectory tracking optimization problem at time instant t can be formulated as shown in equation (6.16), here t is omitted for brevity. As mentioned in section 6.1, in order to emphasize the importance of dynamic model-based constraints in the design of the linear optimization-based controller for trajectory tracking, we compare the performance differences of the two optimization problem formulations: Formulation I (kinematic + dynamic model), Formulation II (kinematic model only).

$$\min_{u(.)} \quad J = \sum_{i=1}^{p-1} x_e^T(n+i)Qx_e(n+i) + \sum_{i=0}^{c} u(n+i)^T Ru(n+i) \tag{6.16}$$

$$\textit{subject to} \tag{6.17}$$

Formulation I or *Formulation II,*

$u(n+i) \in \mathcal{U}$

where n represents the sampling instant, $x_e = [e_1,\, e_2,\, e_3]$ is the trajectory tracking error that has to be minimized, $u = [\omega_{r,i}^{ref},\, \omega_{l,i}^{ref}]$ is the input variable to the inner-loop controller, p is the prediction horizon, c is the control horizon, $Q \in \mathbb{R}^3 \times \mathbb{R}^3$ and $R \in \mathbb{R}^2 \times \mathbb{R}^2$ are weighing matrices to tune the performance of the trajectory tracking $(Q > 0, R > 0)$, $\mathcal{U} \subset \mathbb{R}^2$ are the constraints on the input variables which are represented as inequality constraints $\mathcal{U} = \{u \in \mathbb{R}^2 : 0 \text{ rad/sec} \leq u \leq 51.3 \text{ rad/s}\}$.

Formulation I. This includes the equality constraints introduced by the trajectory tracking error dynamics and the prediction model of the plant; the prediction model consists of the kinematic model and the dynamic model of the inner-loop system i.e. the inner-loop $(\omega_{r,i}^{ref}, \omega_{l,i}^{ref}) \rightarrow (\omega_{r,i}, \omega_{l,i})$ is assumed to have finite bandwidth. The constraints included in Formulation I are

- Error Dynamics: Equations (6.13) - (6.15)
- Kinematic Model: Equations (6.4) - (6.6)

- Dynamic Model: Equations (6.7) - (6.8)

Formulation II. This also includes the equality constraints introduced by the trajectory tracking error dynamics and the prediction model, however, the prediction model only considers the kinematic model i.e. the inner-loop $(\omega_{r,i}^{ref}, \omega_{l,i}^{ref}) \rightarrow (\omega_{r,i}, \omega_{l,i})$ system is assumed to have infinite bandwidth. The constraints included in Formulation II are

- Error Dynamics: Equations (6.13) - (6.15)
- Kinematic Model: Equations (6.4) - (6.6)
- Dynamic Model: $\omega_{r,i} = \omega_{r,i}^{ref};\ \omega_{l,i} = \omega_{l,i}^{ref}$

Figure 6.5 shows the visual representation of the two formulations.

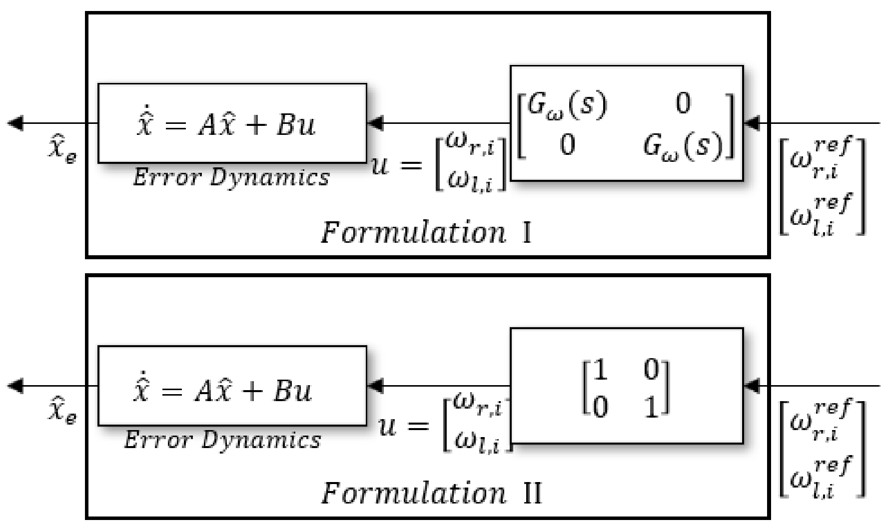


Figure 6.5: Formulation I and II - Block Diagram Representation

The performance of an optimization-based controller depends on the process parameters such as prediction horizon (p), control horizon (c), weighing matrices Q & R. The matrix Q penalizes the trajectory tracking errors, which means larger the value of Q, a faster convergence rate for the error terms (e_1, e_2, e_3). However, a faster convergence rate would mean larger control inputs $(\omega_{r,i}^{ref}, \omega_{l,i}^{ref})$, and more chance for overshoot and control saturation. Therefore, there is always a trade-off between the error convergence rate (bandwidth) and the control signal overshoot. Increasing the value of the R matrix decreases the overshoot in the control inputs but the settling time increases. In other words, R is inversely proportional to the bandwidth of the controller. Having a high bandwidth is essential to achieve good maneuverability during aggressive maneuvers. The prediction horizon (p) affects the convergence rate of the tracking error as well. As the prediction horizon increases, there is a decrease in the settling time of tracking errors, which in turn increases the overshoot in control inputs that can cause controller saturation. Also, increasing the prediction horizon p would mean more number of prediction steps, which in turn lead to an increase in the computation time per sampling instant. A simple rule of thumb while choosing p is to start with a value of less than $15 \sim 20$ samples and keep increasing it until further increase has only minor impacts on performance. The control horizon (c) should be within $20 - 30\%$ of prediction horizon [51]. After carefully turning the controller for the desired tracking performance, the final parameters are as follow:

$$Q = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, \quad p = 22, \quad c = 4 \tag{6.18}$$

6.3 Impact of Kinematic and Dynamic Model Constraints: Simulation and Hardware Trade Studies

The following figures show the variation in x_e, y_e, and θ_e with respect to changes in radius of the track, tracking velocity and bandwidth of the inner-loop - for both

Formulation I, Formulation II.

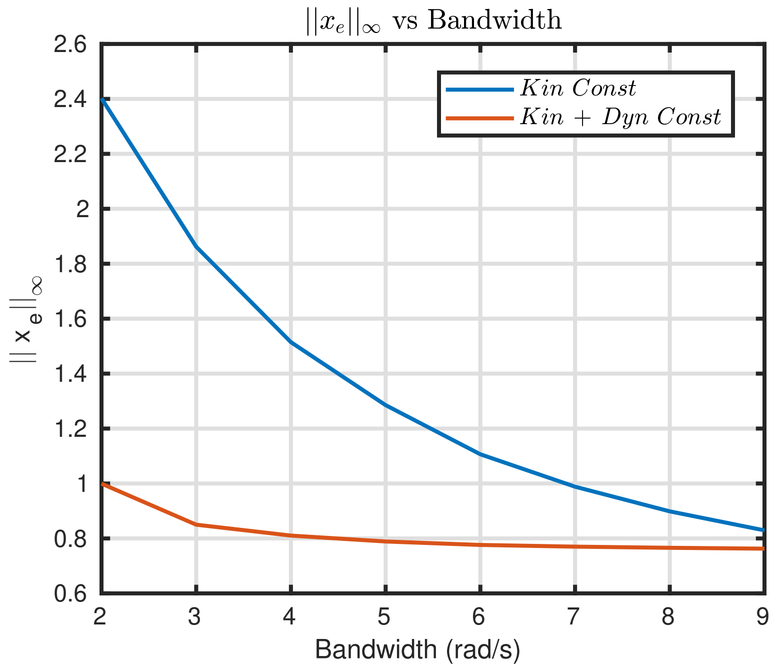


Figure 6.6: $||x_e||_\infty$ vs Inner-Loop Bandwidth: Simulation Results

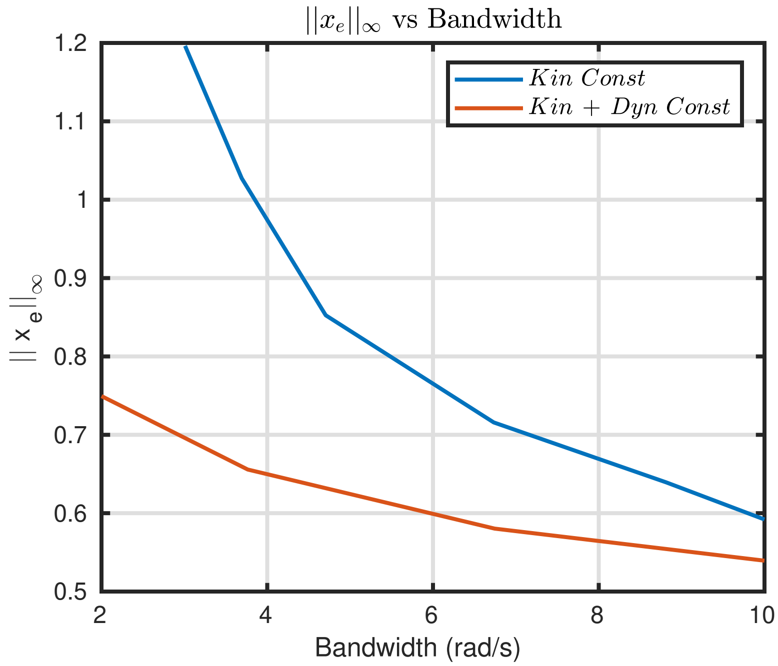


Figure 6.7: $||x_e||_\infty$ vs Inner-Loop Bandwidth: Hardware Results

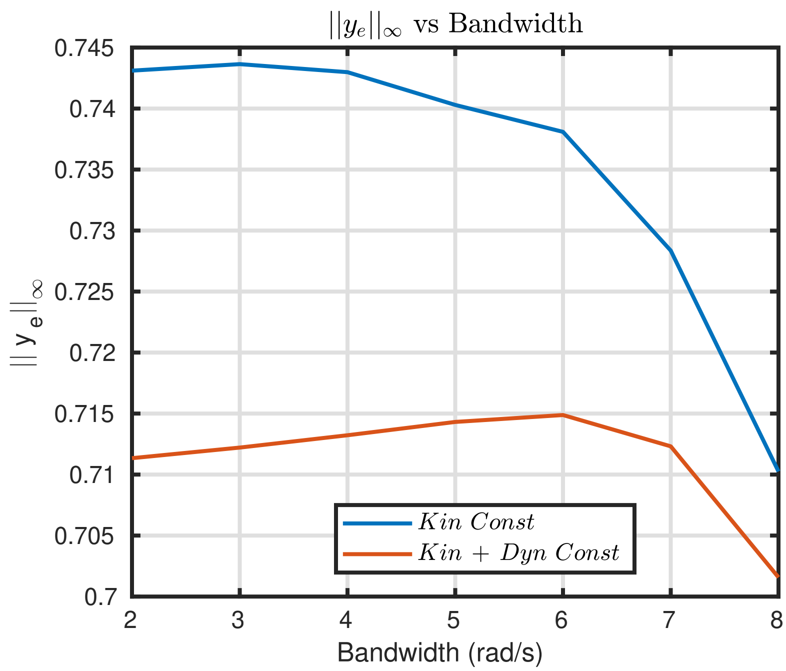


Figure 6.8: $||y_e||_\infty$ vs Inner-Loop Bandwidth: Simulation Results

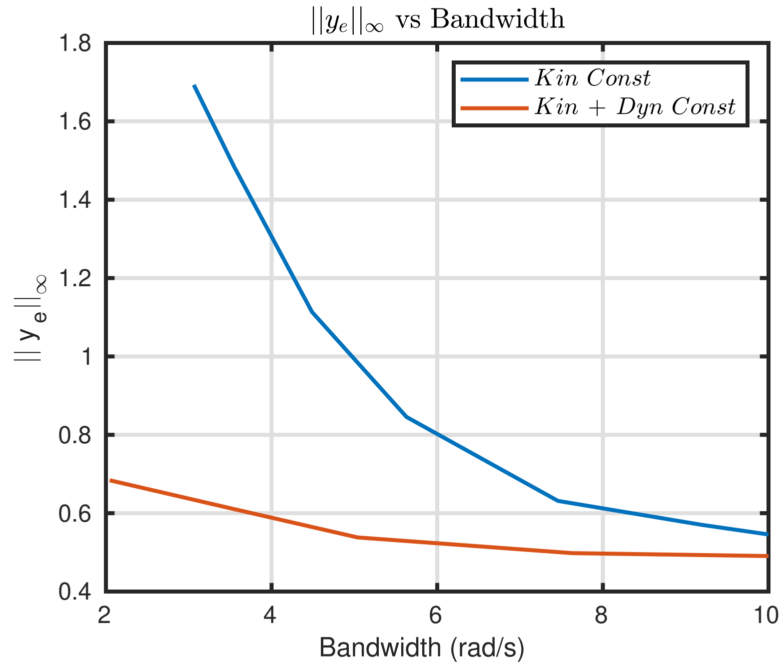


Figure 6.9: $||y_e||_\infty$ vs Inner-Loop Bandwidth: Hardware Results

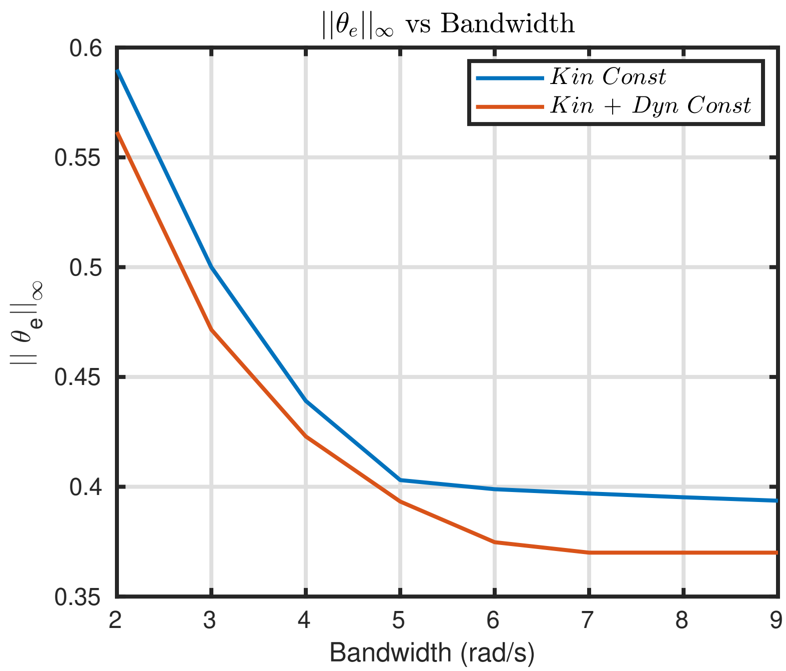


Figure 6.10: $||\theta_e||_\infty$ vs Inner-Loop Bandwidth: Simulation Results

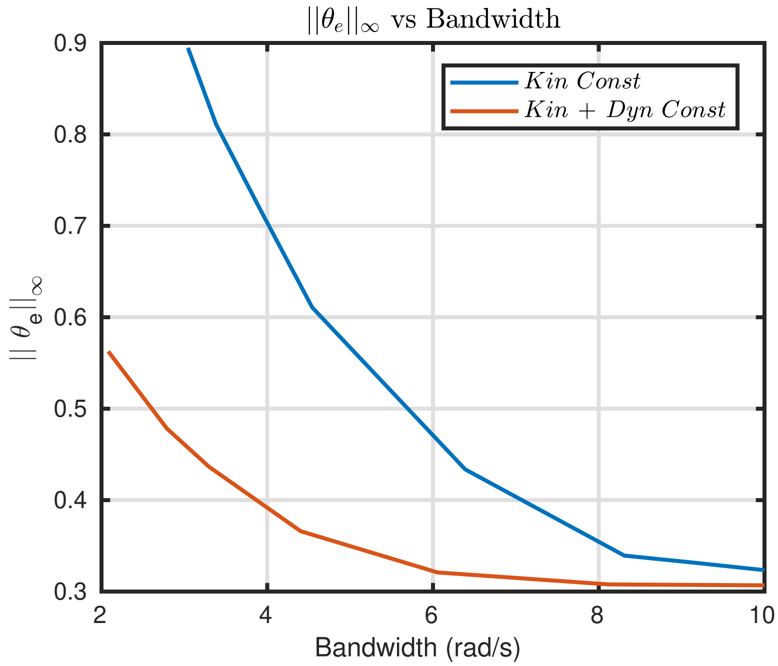


Figure 6.11: $||\theta_e||_\infty$ vs Inner-Loop Bandwidth: Hardware Results

Varying Inner-Loop Bandwidth Bi. The simulation and hardware results presented in Figures 6.6 - 6.11 are obtained at reference velocity $v_{ref} = 1$ m/s and radius of track $R = 1.5$ m while varying the inner-loop bandwidth. The hardware results had to be limited to inner-loop bandwidth $B_i \leq 10$ rad/s due to the physical restrictions of the experimental setup.

From Figures 6.6 - 6.11, the following observations can be made:

- In both simulation and hardware plots, reducing the inner-loop bandwidth B_i causes an increase in $||x_e||_\infty, ||y_e||_\infty, ||\theta_e||_\infty$ irrespective of Formulation I (Kin + Dyn Const) or Formulation II (Kin Const).

- For Formulation I, we notice that $||x_e||_\infty \leq 0.75, ||y_e||_\infty \leq 0.7, ||\theta_e||_\infty \leq 0.57$ at $v_{ref} = 1$ m/s and $R = 1.5$ m. This means that as long as tracking velocity $v_{ref} \leq 1$ m/s and radius of track $R \geq 1.5$ m, the trajectory tracking performance will not be affected significantly for lower inner-loop bandwidth $2 \leq B_i$ rad/sec.

- For Formulation II, we can see that $||x_e||_\infty \leq 0.75, ||y_e||_\infty \leq 0.7, ||\theta_e||_\infty \leq 0.57$ for $10 \geq B_i \geq 7.5$, at $v_{ref} \leq 1$ m/s and $R \geq 1.5$ m. Within this inner-loop bandwidth range $7.5 \geq B_i \geq 10$ it can be seen that Formulation II performance is similar to that of Formulation I, but steeply increases for $B_i < 7.5$ rad/sec.

- For $B_i \leq 5$ rad/sec, we can notice that $||x_e^{Kin+Dyn} - x_e^{kin}||_\infty \geq 0.1, ||y_e^{Kin+Dyn} - y_e^{kin}||_\infty \geq 0.1, ||\theta_e^{Kin+Dyn} - \theta_e^{kin}||_\infty \geq 0.1$. This means that for $B_i \leq 5$ rad/sec, a minimum deviation of 0.1 m can be expected in the position tracking performance between Formulation I and Formulation II.

- For $B_i > 7.5$ rad/sec, a kinematic model based optimization scheme is sufficient to provide good trajectory tracking properties, given that $v_{ref} \leq 1$ m/s, and $R \geq 1.5$ m.

- However, for $B_i < 7.5$ rad/sec, a dynamic model-based optimization scheme is necessary to achieve similar performance.

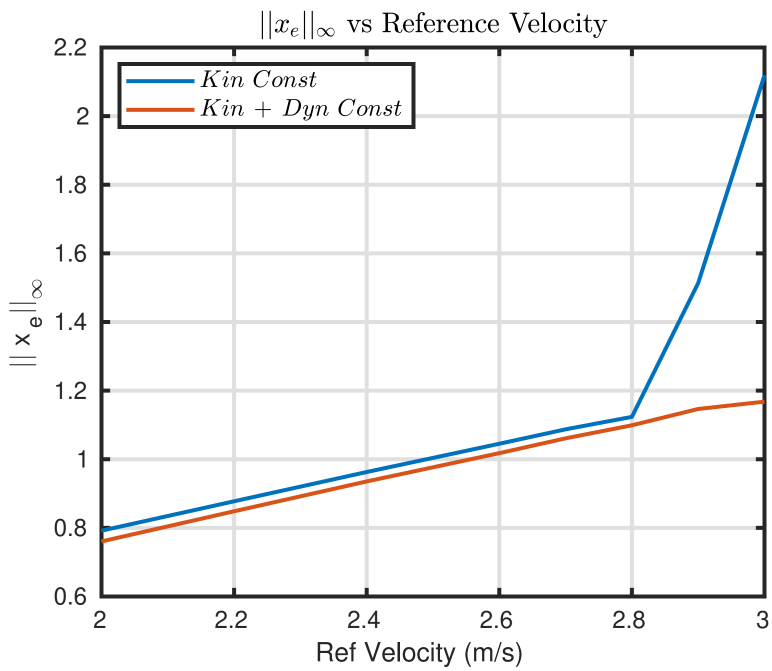


Figure 6.12: $||x_e||_\infty$ vs Reference Velocity: Simulation Results

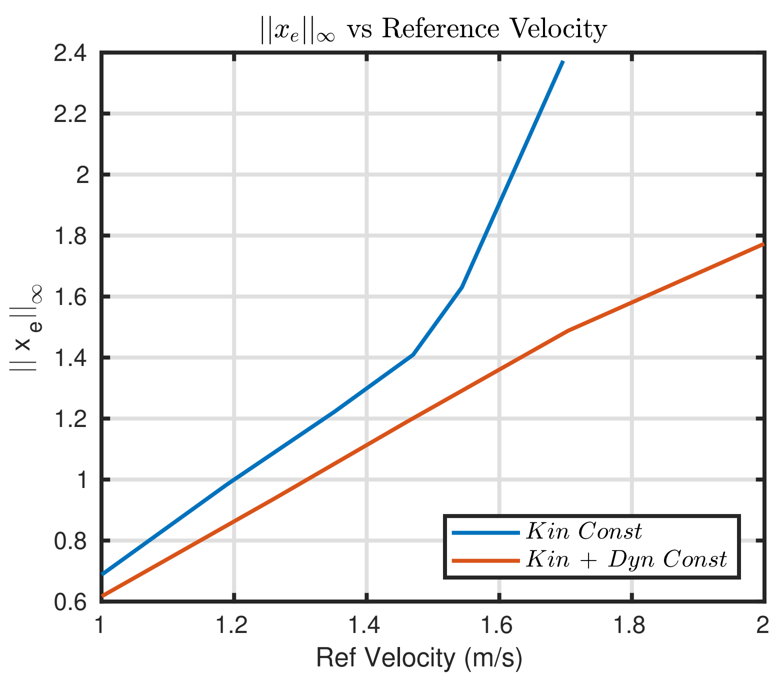


Figure 6.13: $||x_e||_\infty$ vs Reference Velocity: Hardware Results

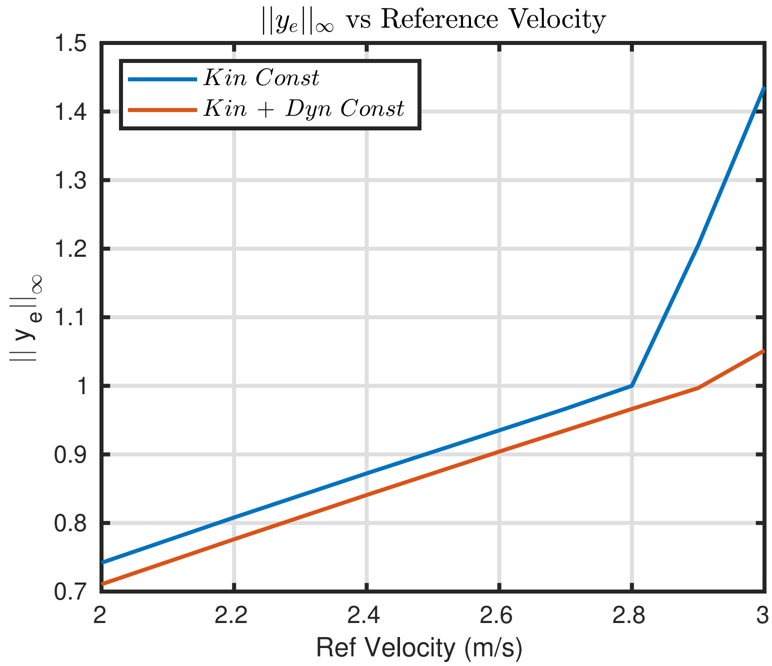


Figure 6.14: $||y_e||_\infty$ vs Reference Velocity: Simulation Results

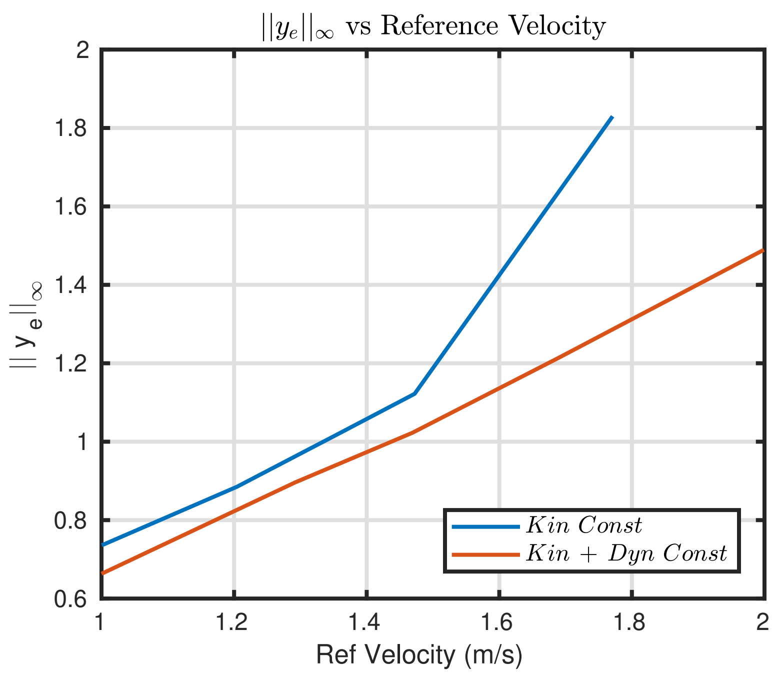


Figure 6.15: $||y_e||_\infty$ vs Reference Velocity: Hardware Results

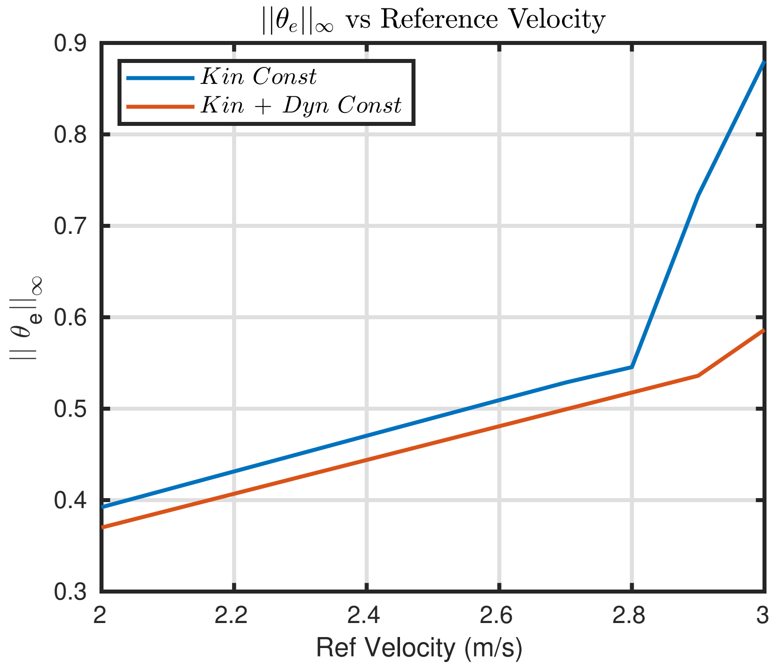


Figure 6.16: $||\theta_e||_\infty$ vs Reference Velocity: Simulation Results

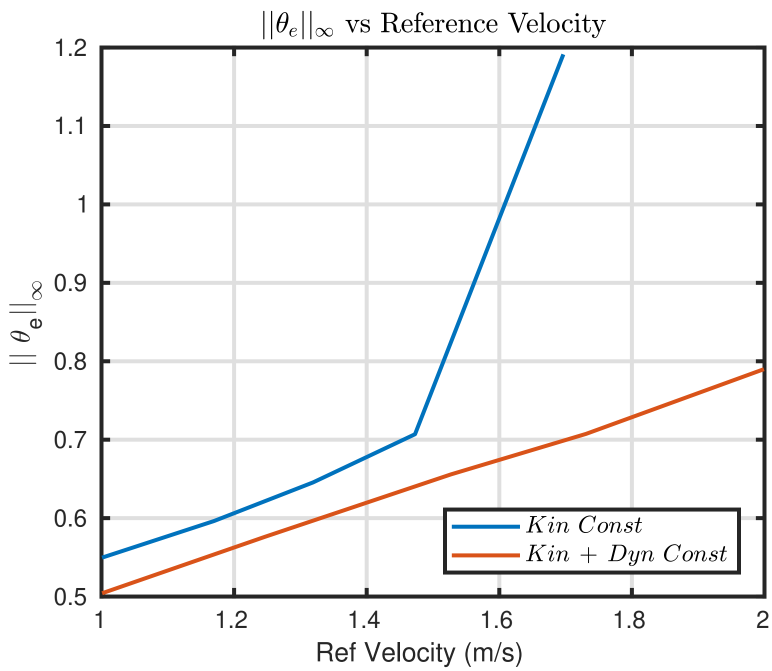


Figure 6.17: $||\theta_e||_\infty$ vs Reference Velocity: Hardware Results

Varying Trajectory Tracking Velocity v_{ref}. The simulation and hardware results presented in Figures 6.12 - 6.17 are obtained at inner-loop bandwidth $B_i = 10$ rad/sec and radius of track $R = 1.5$ m while varying the trajectory tracking velocity. The hardware results had to be limited to trajectory tracking velocity $v_{ref} \leq 2$ m/s due to the physical restrictions of the experimental setup.

From Figures 6.12 - 6.17, the following observations can be made:

- In both simulation and hardware plots, increasing the reference tracking velocity v_{ref} causes an increase in $||x_e||_\infty, ||y_e||_\infty, ||\theta_e||_\infty$ irrespective of Formulation I (Kin + Dyn Const) or Formulation II (Kin Const).

- For Formulation I, we notice that $||x_e||_\infty \leq 1.8, ||y_e||_\infty \leq 1.5, ||\theta_e||_\infty \leq 0.8$ at $B_i = 10$ rad/sec and $R = 1.5$ m. This means that as long as the inner-loop bandwidth $B_i \geq 10$ rad/sec and radius of track $R \geq 1.5$ m, the trajectory tracking performance will not be affected significantly for variations in trajectory tracking velocity $v_{ref} \leq 2$ m/s.

- For Formulation II, we can see that $||x_e||_\infty \leq 1.8, ||y_e||_\infty \leq 1.5, ||\theta_e||_\infty \leq 0.8$ for $1.6 \geq v_{ref} \geq 1$, at $B_i \geq 10$ rad/sec and $R \geq 1.5$ m. Within this trajectory tracking velocity range $1.6 \geq v_{ref} \geq 1$ it can be seen that Formulation II performance is similar to that of Formulation I, but steeply increases for $v_{ref} >$ 1.6 m/s.

- For $v_{ref} \geq 1.65$ m/s, we can notice that $||x_e^{Kin+Dyn} - x_e^{kin}||_\infty \geq 0.1, ||y_e^{Kin+Dyn} - y_e^{kin}||_\infty \geq 0.1, ||\theta_e^{Kin+Dyn} - \theta_e^{kin}||_\infty \geq 0.2$. This means that for $v_{ref} \geq 1.65$ m/s, a minimum deviation of 0.1 m can be expected in the position tracking performance between Formulation I and Formulation II.

- For $v_{ref} < 1.6$ m/s, a kinematic model-based optimization scheme is sufficient to provide good trajectory tracking properties, given that $B_i \geq 10$ rad/sec, and $R \geq 1.5$ m.

- However, for $v_{ref} > 1.6$ m/s, a dynamic model-based optimization scheme is necessary to achieve a similar performance.

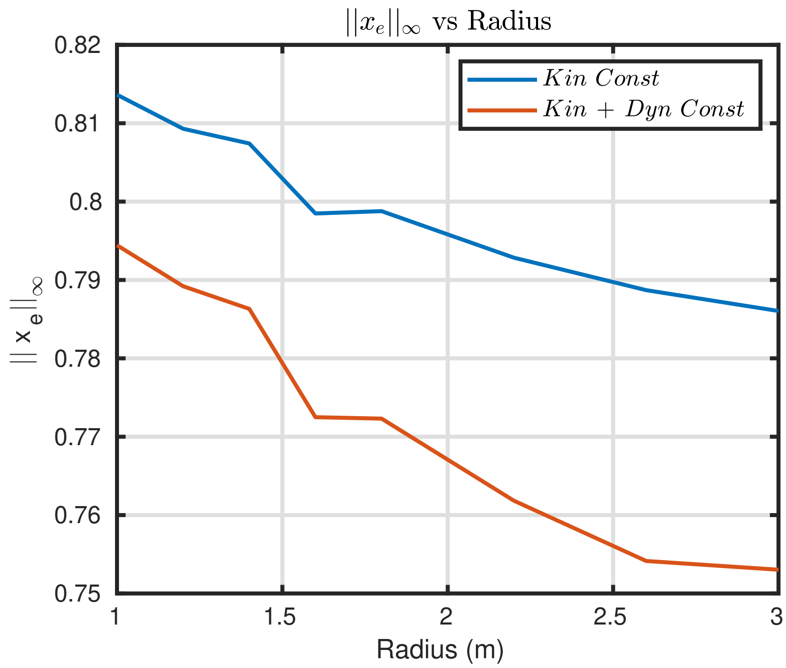


Figure 6.18: $||x_e||_\infty$ vs Radius of Track: Simulation Results

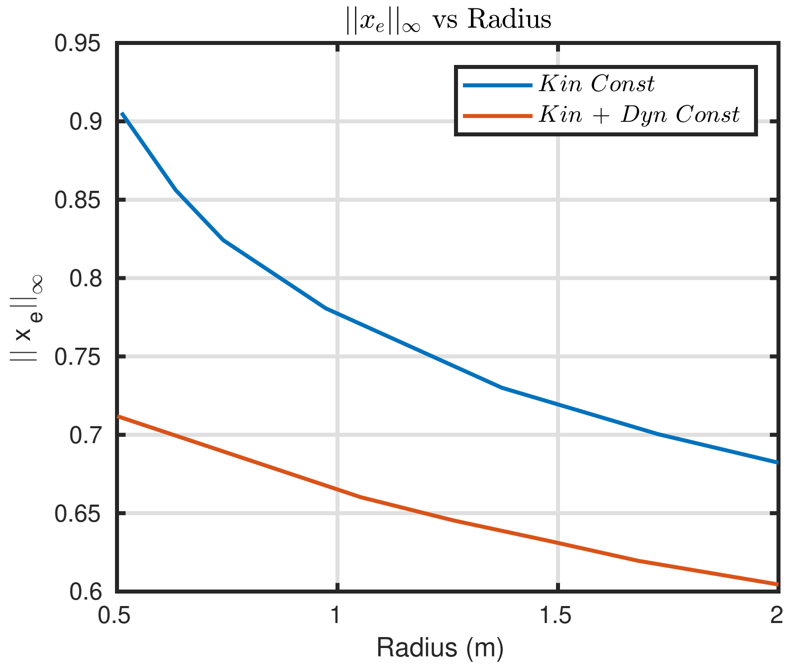


Figure 6.19: $||x_e||_\infty$ vs Radius of Track: Hardware Results

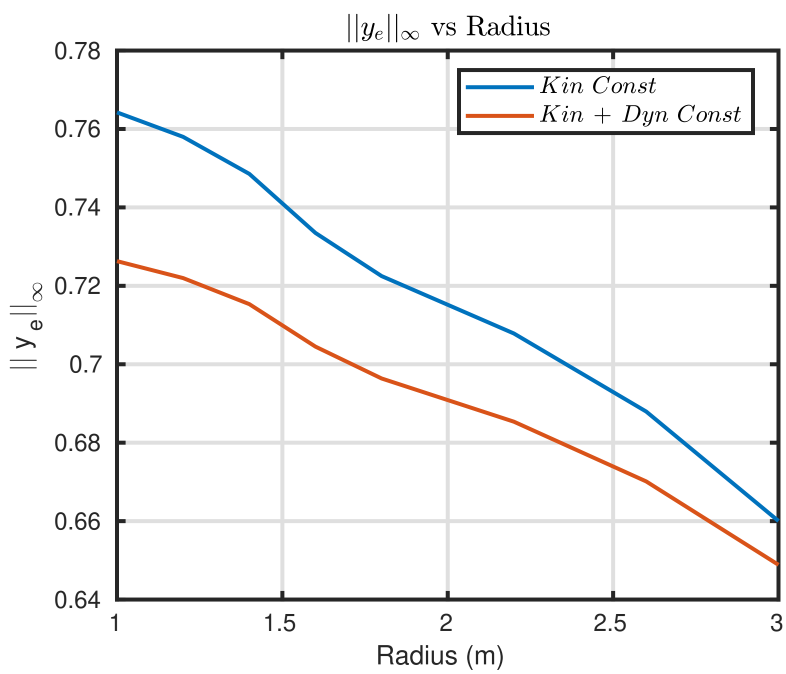


Figure 6.20: $||y_e||_\infty$ vs Radius of Track: Simulation Results

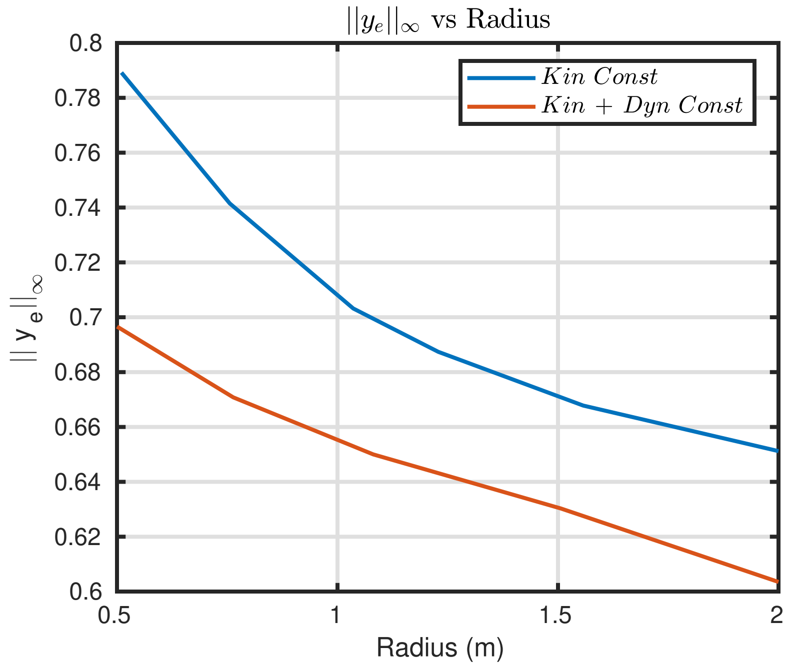


Figure 6.21: $||y_e||_\infty$ vs Radius of Track: Hardware Results

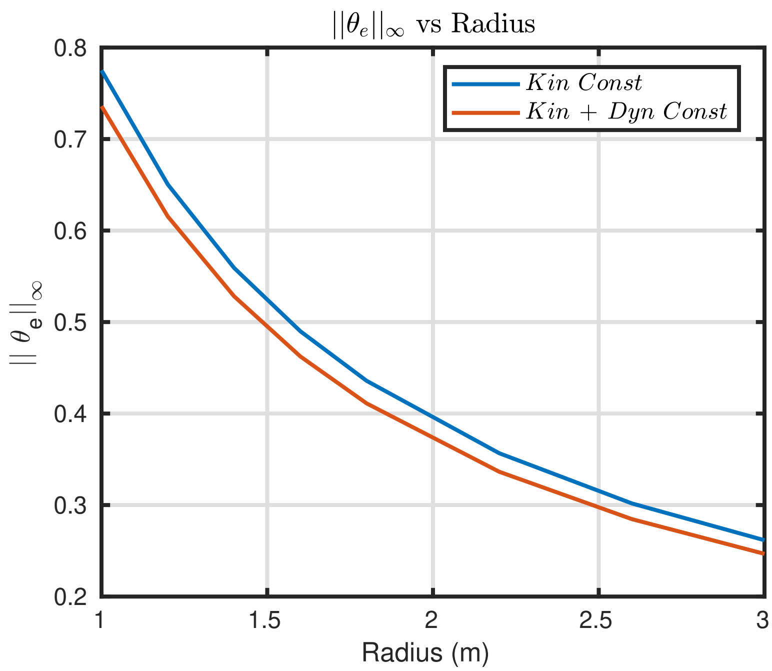


Figure 6.22: $||\theta_e||_\infty$ vs Radius of Track: Simulation Results

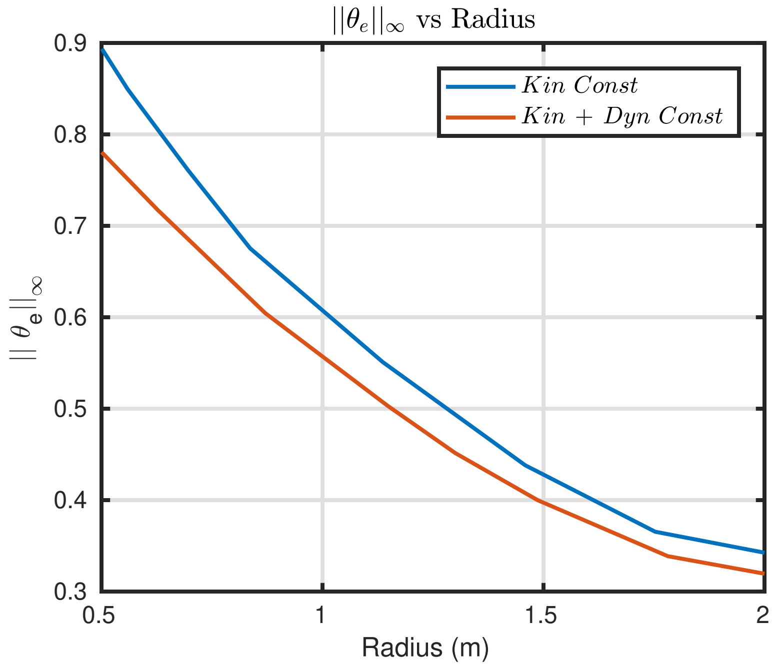


Figure 6.23: $||\theta_e||_\infty$ vs Radius of Track: Hardware Results

Varying Radius of Curvature of Trajectory R. The simulation and hardware results presented in Figures 6.18 - 6.23 are obtained at reference velocity $v_{ref} = 1$ m/s and inner-loop bandwidth $B_i = 10$ rad/s while varying the radius of curvature. The hardware results had to be limited to radius of curvature $R \leq 2$ m due to the physical restrictions of the experimental setup.

From Figures 6.18 - 6.23, the following observations can be made:

- In both simulation and hardware plots, reducing the radius of curvature of trajectory R causes an increase in $||x_e||_\infty, ||y_e||_\infty, ||\theta_e||_\infty$ irrespective of Formulation I (Kin + Dyn Const) or Formulation II (Kin Const).

- For Formulation I, we notice that $||x_e||_\infty \leq 0.7, ||y_e||_\infty \leq 0.7, ||\theta_e||_\infty \leq 0.78$ at $v_{ref} = 1$ m/s and $B_i = 10$ rad/sec. This means that as long as tracking velocity $v_{ref} \leq 1$ m/s and inner-loop bandwidth $B_i \geq 10$ rad/sec, the trajectory tracking performance will not be affected significantly for variations in radius of curvature $0.5 \leq R$ m.

- For Formulation II, we can see that $||x_e||_\infty \leq 0.7, ||y_e||_\infty \leq 0.7, ||\theta_e||_\infty \leq 0.78$ for $2 \geq R \geq 1.6$, at $v_{ref} \leq 1$ m/s and $B_i \geq 10$ rad/sec. Within this radius of curvature range $2 \geq R \geq 1.6$ it can be seen that Formulation II performance is similar to that of Formulation I, but steeply increases for $R < 1.6$ m.

- For $R \leq 0.8$ m, we can notice that $||x_e^{Kin+Dyn} - x_e^{kin}||_\infty \geq 0.1, ||y_e^{Kin+Dyn} - y_e^{kin}||_\infty \geq 0.1, ||\theta_e^{Kin+Dyn} - \theta_e^{kin}||_\infty \geq 0.05$. This means that for $R \leq 0.8$ m, a minimum deviation of 0.1 m can be expected in the position tracking performance between Formulation I and Formulation II.

- For $R > 1.6$ m, a kinematic model-based optimization scheme is sufficient to provide good trajectory tracking properties, given that $v_{ref} \leq 1$ m/s, and $B_i \geq 10$ rad/sec.

- However, for $R < 1.6$ m, a dynamic model-based optimization scheme is necessary to achieve similar performance.

Chapter 7

SUMMARY AND FUTURE DIRECTIONS

7.1 Summary of Work

In this thesis, we have presented detailed instructions on how to choose the actuators based on the DDV performance requirements, and also a step by step guide to building the DDV. An open-source software framework is developed using C++ that is capable of handling multi-robot research. We have also compared and analyzed the dynamic and control design properties of three different DDV models ($P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$, $P_{[e_{a_r},e_{a_l}]\rightarrow[v,\omega]}$, and $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[v,\omega]}$). Additionally, we have also shown how the critical design parameters such as mass, moment of inertia, radius of wheels and center of gravity location can impact the bandwidth, stability, and decoupling properties of the DDV. Subsequently, the impact of critical design parameters on the performance of the outer-loop cruise (v, ω) and position control (x, y) algorithms is also presented. Classical control methodologies have been used to design the inner-loop and outer-loop control laws. A multi-robot trajectory tracking strategy based on receding horizon optimization is presented. In addition to this, the impact of kinematic model-based constraints and the dynamic model based constraints on the performance of trajectory tracking is also studied. Finally, all the simulation results have been compared and verified with hardware data.

7.2 Directions for Future Research

Future work will involve each of the following:

- **Localization.** Development of a lab-based localization system using a variety of on-board technologies (e.g. cameras, lidar, ultrasonic, etc.). Localization is essential for multi-robot systems that operate in both static and dynamic environments - in these scenarios, having an on-board system capable of performing localization at high navigation speeds will be very crucial.

- **Onboard Sensing.** Addition of multiple onboard sensors; e.g. additional ultrasonics, cameras, lidar, GPS, etc. that can duplicate the potential of a motion capture system in open environments will be extremely beneficial.

- **Advanced Image Processing.** Use of advanced image processing and optimization algorithms [49], [77].

- **Multi-Vehicle Cooperation.** Cooperation between ground, air, and sea vehicles - including quad-rotors, micro-air vehicles, and eventually nano-air vehicles.

- **Impact of DDV Dynamics on Multi-Robot Control Objectives.** The trade studies presented in Chapter 6 for multi-robot formation control can be revisited with additional constraints on the optimization problem such as inter-robot collision avoidance, minimum time/energy trajectory tracking or adaptive formation control in response to external obstacles, etc.

- **Environment Mapping.** Rapid and efficient mapping of unknown and partially known areas via multiple robotic agents using pose graph optimization.

- **Modeling and Control.** More accurate dynamic models and control laws. This can include the development of multi-rate control laws that can significantly lower sampling requirements and dynamic models that incorporate the 3-dimensional model of the DDV.

- **Control-Centric Vehicle Design.** Understanding when simple control laws are possible and when complex control laws are essential. This includes understanding how control-relevant specifications impact (or can drive) the design of a vehicle.

[1] "Steam vr tracking", URL https://partner.steamgames.com/vrlicensing, [Online; accessed 13-November-2020] (????).

[2] Aguiar, A. P., D. B. Dačić, J. P. Hespanha and P. Kokotović, "Path-following or reference tracking?: An answer relaxing the limits to performance", IFAC Proceedings Volumes **37**, 8, 167–172 (2004).

[3] Alrifaee, B., J. Maczijewski and D. Abel, "Sequential convex programming mpc for dynamic vehicle collision avoidance", in "2017 IEEE Conference on Control Technology and Applications (CCTA)", pp. 2202–2207 (IEEE, 2017).

[4] Anvari, I., Non-holonomic differential drive mobile robot control & design: Critical dynamics and coupling constraints, Ph.D. thesis, Arizona State University (2013).

[5] Astolfi, A., "On the stabilization of nonholonomic systems", in "Proceedings of 1994 33rd IEEE Conference on Decision and Control", vol. 4, pp. 3481–3486 (IEEE, 1994).

[6] Astolfi, A., "Discontinuous control of nonholonomic systems", Systems & control letters **27**, 1, 37–45 (1996).

[7] Åström, K. J. and T. Hägglund, PID controllers: theory, design, and tuning, vol. 2 (Instrument society of America Research Triangle Park, NC, 1995).

[8] Balch, T. and R. C. Arkin, "Behavior-based formation control for multirobot teams", IEEE transactions on robotics and automation **14**, 6, 926–939 (1998).

[9] Batten, C., "Control for mobile robots", Maslab IAP Robotics Course (2005).

[10] Ben-Ari, M., "A tutorial on euler angles and quaternions", Weizmann Institute of Science, Israel (2014).

[11] Berman, S., Y. Edan and M. Jamshidi, "Navigation of decentralized autonomous automatic guided vehicles in material handling", IEEE Transactions on Robotics and Automation **19**, 4, 743–749 (2003).

[12] Bloch, A. M., M. Reyhanoglu and N. H. McClamroch, "Control and stabilization of nonholonomic dynamic systems", IEEE Transactions on Automatic control **37**, 11, 1746–1757 (1992).

[13] Bradski, G. and A. Kaehler, Learning OpenCV: Computer vision with the OpenCV library (" O'Reilly Media, Inc.", 2008).

[14] Brockett, R. W. et al., "Asymptotic stability and feedback stabilization", Differential geometric control theory **27**, 1, 181–191 (1983).

[15] Burgard, W., M. Moors, C. Stachniss and F. E. Schneider, "Coordinated multi-robot exploration", IEEE Transactions on robotics **21**, 3, 376–386 (2005).

[16] Chen, J., D. Sun, J. Yang and H. Chen, "Leader-follower formation control of multiple non-holonomic mobile robots incorporating a receding-horizon scheme", The International Journal of Robotics Research **29**, 6, 727–747 (2010).

[17] Chwa, D., "Tracking control of differential-drive wheeled mobile robots using a backstepping-like feedback linearization", IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans **40**, 6, 1285–1295 (2010).

[18] Das, A. K., R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer and C. J. Taylor, "A vision-based formation control framework", IEEE transactions on robotics and automation **18**, 5, 813–825 (2002).

[19] Desai, J. P., "Motion planning and control of cooperative robotic systems", (1998).

[20] Desai, J. P., J. P. Ostrowski and V. Kumar, "Modeling and control of formations of nonholonomic mobile robots", IEEE transactions on Robotics and Automation **17**, 6, 905–908 (2001).

[21] DeSantis, R. M., "Path-tracking for a tractor-trailer-like robot: communication", The International Journal of Robotics Research **13**, 6, 533–544 (1994).

[22] Dhaouadi, R. and A. A. Hatab, "Dynamic modelling of differential-drive mobile robots using lagrange and newton-euler methodologies: A unified framework", Advances in Robotics & Automation **2**, 2, 1–7 (2013).

[23] Díaz del Río, F., G. Jiménez, J. L. Sevillano, S. Vicente and A. Civit Balcells, "A path following control for unicycle robots", Journal of Robotic Systems **18**, 7, 325–342 (2001).

[24] Egerstedt, M. and X. Hu, "Formation constrained multi-agent control", IEEE transactions on robotics and automation **17**, 6, 947–951 (2001).

[25] El-Hawwary, M. I. and M. Maggiore, "Global path following for the unicycle and other results", in "2008 American Control Conference", pp. 3500–3505 (IEEE, 2008).

[26] Feng, L., Y. Koren and J. Borenstein, "Cross-coupling motion controller for mobile robots", IEEE Control Systems Magazine **13**, 6, 35–43 (1993).

[27] Fierro, R. and F. L. Lewis, "Control of a nonholomic mobile robot: Backstepping kinematics into dynamics", Journal of robotic systems **14**, 3, 149–163 (1997).

[28] Francis, B. A. and W. M. Wonham, "The internal model principle of control theory", Automatica **12**, 5, 457–465 (1976).

[29] FUJII, T. and N. MIZUSHIMA, "A new approach to lq design", Transactions of the Society of Instrument and Control Engineers **23**, 2, 129–135 (1987).

[30] Gamage, G. W., G. K. Mann and R. G. Gosine, "Leader follower based formation control strategies for nonholonomic mobile robots: Design, implementation and experimental validation", in "Proceedings of the 2010 American control conference", pp. 224–229 (IEEE, 2010).

[31] Hedrick, J. K. and A. Girard, "Control of nonlinear dynamic systems: Theory and applications", Controllability and observability of Nonlinear Systems **48** (2005).

[32] Hernández Juan, S. and F. Herrero Cotarelo, "Multi-master ros systems", (2015).

[33] Huang, J., S. M. Farritor, A. Qadi and S. Goddard, "Localization and follow-the-leader control of a heterogeneous group of mobile robots", IEEE/ASME transactions on Mechatronics **11**, 2, 205–215 (2006).

[34] Kamel, M. A. and Y. Zhang, "Developments and challenges in wheeled mobile robot control", in "International Conference on Intelligent Unmanned Systems (ICIUS)", (2014).

[35] Kamel, M. A. and Y. Zhang, "Linear model predictive control via feedback linearization for formation control of multiple wheeled mobile robots", in "2015 IEEE International Conference on Information and Automation", pp. 1283–1288 (IEEE, 2015).

[36] Kanayama, Y., Y. Kimura, F. Miyazaki and T. Noguchi, "A stable tracking control method for an autonomous mobile robot", in "Proceedings., IEEE International Conference on Robotics and Automation", pp. 384–389 (IEEE, 1990).

[37] Klančar, G., D. Matko and S. Blažič, "Wheeled mobile robots control in a linear platoon", Journal of Intelligent and Robotic Systems **54**, 5, 709–731 (2009).

[38] Klančar, G., D. Matko and S. Blažič, "A control strategy for platoons of differential drive wheeled mobile robot", Robotics and Autonomous Systems **59**, 2, 57–64 (2011).

[39] Koks, D., "Using rotations to build aerospace coordinate systems", Tech. rep., DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION SALISBURY (AUSTRALIA) SYSTEMS ... (2008).

[40] Kuhne, F., W. F. Lages and J. G. da Silva Jr, "Model predictive control of a mobile robot using linearization", in "Proceedings of mechatronics and robotics", pp. 525–530 (Citeseer, 2004).

[41] Lawton, J. R., R. W. Beard and B. J. Young, "A decentralized approach to formation maneuvers", IEEE transactions on robotics and automation **19**, 6, 933–941 (2003).

[42] Lee, S.-M. and H. Myung, "Receding horizon particle swarm optimisation-based formation control with collision avoidance for non-holonomic mobile robots", IET Control Theory & Applications **9**, 14, 2075–2083 (2015).

[43] Lewis, M. A. and K.-H. Tan, "High precision formation control of mobile robots using virtual structures", Autonomous robots **4**, 4, 387–403 (1997).

[44] Li, X., J. Xiao and J. Tan, "Modeling and controller design for multiple mobile robots formation control", in "2004 IEEE International Conference on Robotics and Biomimetics", pp. 838–843 (IEEE, 2004).

[45] Li, Z., Modeling and control of a longitudinal platoon of ground robotic vehicles (Arizona State University, 2016).

[46] Lin, Z., Modeling, design and control of multiple low-cost robotic ground vehicles (Arizona State University, 2015).

[47] Liu, J. and J. Wu, Multiagent robotic systems (CRC press, 2018).

[48] Long, M., A. Gage, R. Murphy and K. Valavanis, "Application of the distributed field robot architecture to a simulated demining task", in "proceedings of the 2005 IEEE International Conference on Robotics and Automation", pp. 3193–3200 (IEEE, 2005).

[49] Lopez, J. A., Image processing based control of mobile robotics (Arizona State University, 2016).

[50] Luca Schenato, M. Baccega, A. Carraro, F. Marchetti, "Control laboratory lecture 19-21", URL `http://automatica.dei.unipd.it/tl_files/utenti/lucaschenato/Classes/Control_Lab_2016/CL_2016_Lecture_19.pdf`, [Online; accessed 13-November-2020] (2016).

[51] MathWorks, "Choosing prediction and control horizons", URL `https://www.mathworks.com/help/mpc/ug/choosing-sample-time-and-horizons.html`, [Online; accessed 13-November-2020] (2020).

[52] Mondal, K., Multivariable control of fixed wing aircrafts (Arizona State University, 2015).

[53] Mondal, K., A. A. Rodriguez, S. S. Manne, N. Das and B. Wallace, "Comparison of kinematic and dynamic model based linear model predictive control of non-holonomic robot for trajectory tracking: Critical trade-offs addressed", URL `https://asu.pure.elsevier.com/en/publications/comparison-of-kinematic-and-dynamic-model-based-linear-model-pred` (1970).

[54] Mondal, K., B. Wallace and A. A. Rodriguez, "Stability versus maneuverability of non-holonomic differential drive mobile robot: Focus on aggressive position control applications", in "2020 IEEE Conference on Control Technology and Applications (CCTA)", pp. 388–395 (IEEE, 2020).

[55] Ohya, I., A. Kosaka and A. Kak, "Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing", IEEE transactions on robotics and automation **14**, 6, 969–978 (1998).

[56] Oriolo, G., A. De Luca and M. Vendittelli, “Wmr control via dynamic feedback linearization: design, implementation, and experimental validation”, IEEE Transactions on control systems technology **10**, 6, 835–852 (2002).

[57] Pappalardo, C. M. and D. Guida, “Forward and inverse dynamics of a unicycle-like mobile robot”, Machines **7**, 1, 5 (2019).

[58] Parker, L. E., “Alliance: An architecture for fault tolerant multirobot cooperation”, IEEE transactions on robotics and automation **14**, 2, 220–240 (1998).

[59] Pendleton, S., Z. J. Chong, B. Qin, W. Liu, T. Uthaicharoenpong, X. Shen, G. M. J. Fu, M. Scarnecchia, S.-W. Kim, M. H. Ang et al., “Multi-class driverless vehicle cooperation for mobility-on-demand”, in “Intelligent Transportation Systems World Congress (ITSWC)”, (2014).

[60] Pepy, R., A. Lambert and H. Mounier, “Path planning using a dynamic vehicle model”, in “2006 2nd International Conference on Information & Communication Technologies”, vol. 1, pp. 781–786 (IEEE, 2006).

[61] Petrov, P., “Modeling and adaptive path control of a differential drive mobile robot”, in “Proceedings of the 12th WSEAS international conference on Automatic control, modelling & simulation”, vol. 6, pp. 403–408 (World Scientific and Engineering Academy and Society (WSEAS), 2010).

[62] Pletcher, R. H., J. C. Tannehill and D. Anderson, Computational fluid mechanics and heat transfer (CRC press, 2012).

[63] Puttannaiah, K., J. A. Echols, K. Mondal and A. A. Rodriguez, “Analysis and use of several generalized (mixed sensitivity frameworks for stable multivariable plants subject to simultaneous output and input loop breaking specifications”, in “2015 54th IEEE Conference on Decision and Control (CDC)”, pp. 6617–6622 (IEEE, 2015).

[64] Puttannaiah, K., 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 “2016 American Control Conference (ACC)”, pp. 5632–5637 (IEEE, 2016).

[65] Ren, W. and R. W. Beard, “Formation feedback control for multiple spacecraft via virtual structures”, IEE Proceedings-Control Theory and Applications **151**, 3, 357–368 (2004).

[66] Rodriguez, A., “Analysis and design of feedback control systems”, Tempe, AZ: Control3D (2003).

[67] Rodriguez, A. A., “Analysis and design of multivariable feedback control systems”, USA: Control3D Education Series (2004).

[68] Rodriguez, A. A., K. Puttannaiah, Z. Lin, J. Aldaco, Z. Li, X. Lu, K. Mondal, S. D. Sonawani, N. Ravishankar, N. Das et al., “Modeling, design and control of low-cost differential-drive robotic ground vehicles: Part ii—multiple vehicle study”, in “2017 IEEE Conference on Control Technology and Applications (CCTA)”, pp. 161–166 (IEEE, 2017).

[69] Rodriguez, A. A., K. Puttannaiah, Z. Lin, J. Aldaco, Z. Li, X. Lu, K. Mondal, S. D. Sonawani, N. Ravishankar, N. Das et al., “Modeling, design and control of low-cost differential-drive robotic ground vehicles: Part i—single vehicle study”, in “2017 IEEE Conference on Control Technology and Applications (CCTA)”, pp. 155–160 (IEEE, 2017).

[70] Samson, C., “Control of chained systems application to path following and time-varying point-stabilization of mobile robots”, IEEE transactions on Automatic Control **40**, 1, 64–77 (1995).

[71] Sheikholeslam, S. and C. A. Desoer, “Longitudinal control of a platoon of vehicles”, in “1990 American control conference”, pp. 291–296 (IEEE, 1990).

[72] Sheikholeslam, S. E., “Control of a class of interconnected nonlinear dynamical systems: The platoon problem.”, (1993).

[73] Shivakumar, T., M. S. Sravan and K. Selvajyothi, “Python based 3-axis cnc plotter”, in “2016 IEEE International Conference on Power and Energy (PECon)”, pp. 823–827 (IEEE, 2016).

[74] Shojaei, K., A. Tarakameh and A. M. Shahri, “Adaptive trajectory tracking of wmrs based on feedback linearization technique”, in “2009 International Conference on Mechatronics and Automation”, pp. 729–734 (IEEE, 2009).

[75] Singh, B., R. Payasi, K. Verma, V. Kumar and S. Gangwar, “Design of controllers pd, pi & pid for speed control of dc motor using igbt based chopper”, German Journal of Renewable and Sustainable Energy Research (GJRSER) **1**, 1 (2013).

[76] Sravan, M. S., S. Moolam, S. Sreepathi and P. Kokil, “Implementation of remote motion controller with visual feedback”, in “2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)”, pp. 1–7 (IEEE, 2017).

[77] Sravan, M. S., S. Natarajan, E. S. Krishna and B. J. Kailath, “Fast and accurate on-road vehicle detection based on color intensity segregation”, Procedia Comput. Sci **133**, 594–603 (2018).

[78] Sridharan, S., J. A. Echols, A. A. Rodriguez and K. Mondal, “Integrated design and control of hypersonic vehicles”, in “2014 American Control Conference”, pp. 1371–1376 (IEEE, 2014).

[79] Stein, G., “Respect the unstable”, IEEE Control Systems Magazine **23**, 4, 12–25 (2003).

[80] Stroupe, A., T. Huntsberger, A. Okon, H. Aghazarian and M. Robinson, "Behavior-based multi-robot collaboration for autonomous construction tasks", in "2005 IEEE/RSJ International Conference on Intelligent Robots and Systems", pp. 1495–1500 (IEEE, 2005).

[81] Takahashi, H., H. Nishi and K. Ohnishi, "Autonomous decentralized control for formation of multiple mobile robots considering ability of robot", IEEE Transactions on Industrial Electronics **51**, 6, 1272–1279 (2004).

[82] Tang, Z. and U. Ozguner, "Motion planning for multitarget surveillance with mobile sensor agents", IEEE Transactions on Robotics **21**, 5, 898–908 (2005).

[83] Tanner, H. G., G. J. Pappas and V. Kumar, "Leader-to-formation stability", IEEE Transactions on robotics and automation **20**, 3, 443–455 (2004).

[84] Tiderko, A., F. Hoeller and T. Röhling, "The ros multimaster extension for simplified deployment of multi-robot systems", in "Robot Operating System (ROS)", pp. 629–650 (Springer, 2016).

[85] Vieira, F. C., A. A. Medeiros, P. J. Alsina and A. P. Araújo Jr, "Position and orientation control of a two-wheeled differentially driven nonholonomic mobile robot.", in "ICINCO (2)", pp. 256–262 (2004).

[86] Vis, I. F., "Survey of research in the design and control of automated guided vehicle systems", European Journal of Operational Research **170**, 3, 677–709 (2006).

[87] Wikipedia contributors, "Htc vive", URL `https://en.wikipedia.org/wiki/HTC_Vive`, [Online; accessed 13-November-2020] (2020).

[88] Yamashita, A., T. Arai, J. Ota and H. Asama, "Motion planning of multiple mobile robots for cooperative manipulation and transportation", IEEE Transactions on Robotics and Automation **19**, 2, 223–237 (2003).

[89] Yoshida, H., S. Shinohara and M. Nagai, "Lane change steering manoeuvre using model predictive control theory", Vehicle System Dynamics **46**, S1, 669–681 (2008).

APPENDIX A

ADDITIONAL HARDWARE INFORMATION

Hardware Laser Cut Files - Website Link:
https://tinyurl.com/y4rxvpan

Hardware 3D Print Files - Website Link:
https://tinyurl.com/y4rxvpan

Motor Specifications and Product Purchase Website Links:
https://tinyurl.com/y4tazghq

APPENDIX B

MATLAB CODE

```matlab
% Trade Studies at d = 0
clc
close all
clear all
s = tf([1 0],[1]);
md = 0; m = 3.4; % if d = 0;
%% Different Plant Models with the respective parameters as input
% at d = 0
% Plant model from e_r, e_l to W_r, W_l decoupled
% Plant model from (e_r + e_l), (e_r-e_l) to V, W decoupled
%%
% Plant model from e_r, e_l to W_r, W_l
% Singular and Bode Plots for different values of I
%(including the I_AR conditions)
d = 0; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
%[max,min] = Imaxmin(d,Iw,L,md,dw);
Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)

I = [0.42499999999 0.42500 0.42500000001 0.4292 0.4462 0.4675 0.3825];

P1 = Plantww(d, Veq, Weq, dw, Iw, I(1), L, md,R);
P2 = Plantww(d, Veq, Weq, dw, Iw, I(2), L, md,R);
P3 = Plantww(d, Veq, Weq, dw, Iw, I(3), L, md,R);
P4 = Plantww(d, Veq, Weq, dw, Iw, I(4), L, md,R);
P5 = Plantww(d, Veq, Weq, dw, Iw, I(5), L, md,R);
P6 = Plantww(d, Veq, Weq, dw, Iw, I(6), L, md,R);
P7 = Plantvw(d, Veq, Weq, dw, Iw, I(7), L, md,R);

figure;
bodemag(P7);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Frequency Response for $ d = 0 $ ','FontWeight','bold',...
'FontSize',14,'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots


%% Singular Values Plot
winit   = -1;
wfin    = 2;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w); P4 = sigma(P4,w);
P5 = sigma(P5,w);
P6 = sigma(P6,w); P7 = sigma(P7,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
```

```
P4 = 20*log10(P4);
P5 = 20*log10(P5);
P6 = 20*log10(P6); P7 = 20*log10(P7);
figure;
subplot(2,1,1);
semilogx(w, P7(1,:), w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:),...
 w, P5(1,:), w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $ (e_r,e_l)\rightarrow(\omega_r,\omega_l) $ for $
d = 0 $', 'FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P7(2,:), w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:),...
 w, P5(2,:), w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values $ (e_r,e_l)\rightarrow(\omega_r,\omega_l) $ for $
d = 0 $','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots




% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$I = 0.9I_{AR}$','$I = I_{AR}^-$','$I = I_{AR}$',...
'$I = I_{AR}^+$','$I = 1.01I_{AR}$','$I = 1.05I_{AR}$',...
'$I = 1.1I_{AR}$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',1.2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
 % distance between lines in legend [x,y,width, height]


%%
% Plant model from (e_r + e_l), (e_r-e_l) to V, W
% Singular and Bode Plots for different values of I
```

```
%(including the I_AR conditions)
md = 0; m = 3.4; % if d = 0;
Plant1 = Plantsdv(d, Veq, Weq, dw, Iw, I_AR, L, md,R)


I = [0.3825 0.42560 0.4675 0.6375 0.8500 1.2750];

P1 = Plantsdv(d, Veq, Weq, dw, Iw, I(1), L, md,R);
P2 = Plantsdv(d, Veq, Weq, dw, Iw, I(2), L, md,R);
P3 = Plantsdv(d, Veq, Weq, dw, Iw, I(3), L, md,R);
P4 = Plantsdv(d, Veq, Weq, dw, Iw, I(4), L, md,R);
P5 = Plantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R);
P6 = Plantsdv(d, Veq, Weq, dw, Iw, I(6), L, md,R);

figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $(e_r + e_l, e_r - e_l)\rightarrow(v,\omega)$ for
$d = 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$I = 0.9I\_{AR}$','$I = I\_{AR}$',...
'$I = 1.01I\_{AR}^+$','$I = 1.5I\_{AR}$','$I = 2.0I\_{AR}$',...
'$I = 3.0I\_{AR}$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.24]);
% distance between lines in legend [x,y,width, height]

%% Singular Values Plot
winit   = -1;
wfin    = 2;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w); P4 = sigma(P4,w);
P5 = sigma(P5,w);
P6 = sigma(P6,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4);
 P5 = 20*log10(P5);
P6 = 20*log10(P6);
figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
 w, P6(1,:))
```

```
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $ (e_r + e_l, e_r - e_l)\rightarrow(v,\omega)$
for $d=0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);       % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
 w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values $ (e_r + e_l, e_r - e_l)\rightarrow(v,\omega) $ for
 $d=0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);       % Lines with thicker width for plots




% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$I = 0.9I_{AR}$','$I = I_{AR}$',...
'$I = 1.01I_{AR}^+$','$I = 1.5I_{AR}$','$I = 2.0I_{AR}$',...
'$I = 3.0I_{AR}$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',1.2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.24]);
% distance between lines in legend [x,y,width, height]





%%
% Plant model from e_r, e_l to W_r, W_l
% Singular and Bode Plots for different values of m
%(variations in total mass without changing I_w)

% Bode Plot

```

```
md = 0; m = 3.4; % if d = 0;
d = 0; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);
Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)

I = [0.42499999999 0.42500 0.42500000001 0.4292 0.4462 0.4675 0.3825];


P1 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m);
P2 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+0.5);
P3 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1);
P4 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1.5);
P5 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2);
P6 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2.5);
P7 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+3);

figure;
bodemag(P1,P2,P3,P4,P5,P6,P7);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $ (e_r,e_l)\rightarrow(\omega_r,\omega_l) $ for
$ d = 0 $ ','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$m = 3.4 \ kg$','$m = 3.9 \ kg$','$m = 4.4 \ kg$',...
'$m = 4.9 \ kg$','$m = 5.4 \ kg$','$m = 5.9 \ kg$','m = 6.4 \ kg'},...
'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
% distance between lines in legend [x,y,width, height]

%% Singular Values Plot
winit   = -1;
wfin    = 2;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w); P4 = sigma(P4,w);
 P5 = sigma(P5,w);
P6 = sigma(P6,w); P7 = sigma(P7,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
```

```
P4 = 20*log10(P4);
 P5 = 20*log10(P5);
P6 = 20*log10(P6); P7 = 20*log10(P7);
figure;
subplot(2,1,1);
semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), ...
w, P5(1,:), w, P6(1,:),w,P7(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for...
 $ d=0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:),...
 w, P5(2,:), w, P6(2,:),w,P7(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values $ (e_r,e_l)\rightarrow(\omega_r,\omega_l) $ for
$ d = 0 $','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots



% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$m = 3.4 \ kg$','$m = 3.9 \ kg$','$m = 4.4 \ kg$',...
'$m = 4.9 \ kg$','$m = 5.4 \ kg$','$m = 5.9 \ kg$','m = 6.4 \ kg'},...
'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',1.2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
% distance between lines in legend [x,y,width, height]

%%
% Plant model from e_r, e_l to W_r, W_l
% Singular and Bode Plots for different values of R
```

```

% Bode Plot

% change in R results in change in IW, however, no significant
% difference is observed

md = 0; m = 3.4; % if d = 0;
d = 0; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);
Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)

I = [0.42499999999 0.42500 0.42500000001 0.4292 0.4462 0.4675 0.3825];

R =  0.042; m_wheel = 0.096;
rm = 0.0248  ; m_motor = 0.224;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);

P1 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P2 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P3 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P4 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P5 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P6 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);


R = 0.042; Iw = 1.67e-06;
figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $ (e_r,e_l)\rightarrow(\omega_r,\omega_l) $ for
```

```
$ d = 0 $ ','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);       % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$R = 0.042 \ m$','$R = 0.052 \ m$',...
'$R = 0.062 \ m$','$R = 0.072 \ m$','$R = 0.082 \ m$',...
'$R = 0.092 \ m$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
% distance between lines in legend [x,y,width, height]

%% Singular Values Plot
winit   = -1;
wfin    = 2;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6);
figure;
subplot(2,1,1);
semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $ (e_r,e_l)\rightarrow(\omega_r,\omega_l) $ for
$ d = 0 $','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);       % Lines with thicker width for plots

subplot(2,1,2);
semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
 w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
```

```
'Min Singular Values $ (e_r,e_l)\rightarrow(\omega_r,\omega_l) $ for
$ d = 0 $','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);       % Lines with thicker width for plots




% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$R = 0.042 \ m$','$R = 0.052 \ m$',...
'$R = 0.062 \ m$','$R = 0.072 \ m$','$R = 0.082 \ m$',...
'$R = 0.092 \ m$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',1.2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
% distance between lines in legend [x,y,width, height]
%%
% Plant model from e_r + e_l, e_r - e_l to V,W
% Singular and Bode Plots for different values of m
%(variations in total mass without changing I_w)

% Bode Plot

md = 0; m = 3.4; % if d = 0;
d = 0; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
% has to be chosen based on the corresponding AR value
% (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
% has to be chosen based on the corresponding AR value
I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);
Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)

I = [0.42499999999 0.42500 0.42500000001 0.4292 0.4462 0.4675 0.3825];


P1 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m);
P2 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+0.5);
P3 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1);
P4 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1.5);
P5 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2);
P6 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2.5);
P7 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+3);

figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
```

```
title(...
'Frequency Response $(e_r + e_l, e_r - e_l)\rightarrow(v,\omega)$ for
$d = 0 $','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$m = 3.4 \ kg$','$m = 3.9 \ kg$','$m = 4.4 \ kg$',...
'$m = 4.9 \ kg$','$m = 5.4 \ kg$','$m = 5.9 \ kg$'},'Interpreter',...
'latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
% distance between lines in legend [x,y,width, height]

%% Singular Values Plot
winit    = -1;
wfin     = 2;
nwpts    = 200;
w        = logspace(winit,wfin,nwpts);
P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w); P7 = sigma(P7,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6); P7 = 20*log10(P7);
figure;
subplot(2,1,1);
semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), ...
w, P5(1,:), w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $(e_r + e_l, e_r - e_l)\rightarrow(v,\omega)$ for
 $d = 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
w, P6(2,:));
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
```

```
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values $(e_r + e_l, e_r - e_l)\rightarrow(v,\omega)$ for
$d = 0$', 'FontWeight','bold','FontSize',12, 'Interpreter','latex');

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);        % Lines with thicker width for plots




% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$m = 3.4 \ kg$','$m = 3.9 \ kg$','$m = 4.4 \ kg$',...
'$m = 4.9 \ kg$','$m = 5.4 \ kg$','$m = 5.9 \ kg$'},'Interpreter',...
'latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',1.2);               % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
% distance between lines in legend [x,y,width, height]

%%
% Plant model from e_r+ e_l, e_r+ e_l to V, W
% Singular and Bode Plots for different values of R

% Bode Plot

% change in R results in change in IW, however, no significant
% difference is observed

md = 0; m = 3.4; % if d = 0;
d = 0; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
% has to be chosen based on the corresponding AR value
% (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
% has to be chosen based on the corresponding AR value
I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);
Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)

I = [0.42499999999 0.42500 0.42500000001 0.4292 0.4462 0.4675 0.3825];

R =  0.042; m_wheel = 0.096;
rm = 0.0248  ; m_motor = 0.224;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);

P1 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P2 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
```

```
I = I_Newcalculation(0,Iw,L,md,dw);
P3 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P4 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P5 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P6 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);


R = 0.042; Iw = 1.67e-06;
figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $(e_r + e_l, e_r - e_l)\rightarrow(v,\omega)$ for
$d = 0 $','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$R = 0.042 \ m$','$R = 0.052 \ m$',...
'$R = 0.062 \ m$','$R = 0.072 \ m$','$R = 0.082 \ m$',...
'$R = 0.092 \ m$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
% distance between lines in legend [x,y,width, height]

%% Singular Values Plot
winit   = -1;
wfin    = 2;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6);
figure;
subplot(2,1,1);
```

```matlab
semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
 w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $ (e_r + e_l, e_r - e_l)\rightarrow(v,\omega)$for
$d = 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
 w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values $(e_r + e_l, e_r - e_l)\rightarrow(v,\omega)$ for
$d = 0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots




% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$R = 0.042 \ m$','$R = 0.052 \ m$',...
'$R = 0.062 \ m$','$R = 0.072 \ m$','$R = 0.082 \ m$',...
'$R = 0.092 \ m$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',1.2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
% distance between lines in legend [x,y,width, height]
```

```matlab
% Trade Studies at d ~= 0
clc
close all
clear all
s = tf([1 0],[1]);
md = 0; m = 3.4; % if d = 0;
%% Different Plant Models with the respective parameters as input
% at d = 0
```

```matlab
% Plant model from e_r, e_l to W_r, W_l decoupled
% Plant model from (e_r + e_l), (e_r-e_l) to V, W decoupled
%%
% Plant model from e_r, e_l to W_r, W_l
% Singular and Bode Plots for different values of Veq
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);
Plant1 = Plantww(d, Veq, Weq, dw, Iw, I, L, md,R);

Veq = [0.1 0.2 0.6 1 3 5];

P1 = Plantww(d, Veq(1), Weq, dw, Iw, I, L, md,R);
P2 = Plantww(d, Veq(2), Weq, dw, Iw, I, L, md,R);
P3 = Plantww(d, Veq(3), Weq, dw, Iw, I, L, md,R);
P4 = Plantww(d, Veq(4), Weq, dw, Iw, I, L, md,R);
P5 = Plantww(d, Veq(5), Weq, dw, Iw, I, L, md,R);
P6 = Plantww(d, Veq(6), Weq, dw, Iw, I, L, md,R);

figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $ (e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
 $d \neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$v\_{eq} = 0.1 \ m/s$','$v\_{eq} = 0.2 \ m/s$',...
'$v\_{eq} = 0.6 \ m/s$','$v\_{eq} = 1.0 \ m/s$',...
'$v\_{eq} = 3.0 \ m/s$','$v\_{eq} = 5.0 \ m/s$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.26 0.24]);
% distance between lines in legend [x,y,width, height]

%% Singular and Bode Plots for different values of Weq
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
```

```
[max,min] = Imaxmin(d,Iw,L,md,dw);

Weq = [-8.0 -2.5 -0.5 0.5 2.5 8.0];

P1 = Plantww(d, Veq, Weq(1), dw, Iw, I, L, md,R);
P2 = Plantww(d, Veq, Weq(2), dw, Iw, I, L, md,R);
P3 = Plantww(d, Veq, Weq(3), dw, Iw, I, L, md,R);
P4 = Plantww(d, Veq, Weq(4), dw, Iw, I, L, md,R);
P5 = Plantww(d, Veq, Weq(5), dw, Iw, I, L, md,R);
P6 = Plantww(d, Veq, Weq(6), dw, Iw, I, L, md,R);

figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
$d\neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$\omega\_{eq} = -8.0 \ rad/s$',...
'$\omega\_{eq} = -2.5 \ rad/s$','$\omega\_{eq} = -0.5 \ rad/s$',...
'$\omega\_{eq} = 0.5 \ rad/s$','$\omega\_{eq} = 2.5 \ rad/s$',...
'$\omega\_{eq} = 8.0 \ rad/s$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',10);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.28 0.24]);
% distance between lines in legend [x,y,width, height]

%% %% Singular and Bode Plots for different values of d
% the behaviour in the bode plots can be associated with the dominat
% pole variation wrt to d
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
% has to be chosen based on the corresponding AR value
% (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
% has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);

d = [-0.09 -0.08 -0.04 0.04 0.08 0.09];
I = [I_Newcalculation(d(1),Iw,L,md,dw) I_Newcalculation(d(2),Iw,L,md,dw)
     I_Newcalculation(d(3),Iw,L,md,dw) I_Newcalculation(d(4),Iw,L,md,dw)
     I_Newcalculation(d(5),Iw,L,md,dw)
     I_Newcalculation(d(6),Iw,L,md,dw)];

```

```
P1 = Plantww(d(1), Veq, Weq, dw, Iw, I(1), L, md,R);
P2 = Plantww(d(2), Veq, Weq, dw, Iw, I(2), L, md,R);
P3 = Plantww(d(3), Veq, Weq, dw, Iw, I(3), L, md,R);
P4 = Plantww(d(4), Veq, Weq, dw, Iw, I(4), L, md,R);
P5 = Plantww(d(5), Veq, Weq, dw, Iw, I(5), L, md,R);
P6 = Plantww(d(6), Veq, Weq, dw, Iw, I(6), L, md,R);

figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
 $d\neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex');

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);       % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$d = -0.1 \ m$','$d = -0.05 \ m$',...
'$d = -0.02 \ m$','$d = 0.02 \ m$','$d = 0.05 \ m$','$d = 0.1 \ m$'},...
'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',10);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.26 0.24]);
% distance between lines in legend [x,y,width, height]

%% plot of dominant pole vs d
figure;
load('dpole_2_1.mat');
d = -0.28:0.01:0.28
plot(d,h(1,:),d,h(2,:),d,h(3,:),d,h(4,:),d,h(5,:),d,h(6,:),d,h(7,:));
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('d (m)','FontSize',12);
ylabel('Dominant Pole ','FontSize',12);
set(h_axes,'LineWidth',2,'FontSize',12,'GridAlpha',0.15);
% size and brightness of grid and size of x & y axis numbers
title('Dominant Pole vs $d$','FontWeight','bold','FontSize',14, ...
'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.8);          % Lines with thicker width for
% plots
[hL,hObj]=legend({'$v_{eq} = 0.0 \ m/s$','$v_{eq} = 0.5 \ m/s$',...
'$v_{eq} = 1.0 \ m/s$','$v_{eq} = 1.5 \ m/s$','$v_{eq} = 2.0 \ m/s$',...
'$v_{eq} = 2.5 \ m/s$','$v_{eq} = 3.0 \ m/s$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',10);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
```

```
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.27 0.24]);
% distance between lines in legend [x,y,width, height]

%% %% Singular and Bode Plots for different values of I
% the behaviour in the bode plots can be associated with the dominat
% pole variation wrt to d
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);

I = [0.4 0.5 0.7 0.9 1.2 1.7];

P1 = Plantww(d, Veq, Weq, dw, Iw, I(1), L, md,R);
P2 = Plantww(d, Veq, Weq, dw, Iw, I(2), L, md,R);
P3 = Plantww(d, Veq, Weq, dw, Iw, I(3), L, md,R);
P4 = Plantww(d, Veq, Weq, dw, Iw, I(4), L, md,R);
P5 = Plantww(d, Veq, Weq, dw, Iw, I(5), L, md,R);
P6 = Plantww(d, Veq, Weq, dw, Iw, I(6), L, md,R);

figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);

% size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $ (e_r,e_l)\rightarrow(\omega_r,\omega_l) $ for
$d\neq0$','FontWeight','bold','FontSize',14, 'Interpreter','latex');

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);       % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$I = 0.4 \ Kg.m^2$','$I = 0.5 \ Kg.m^2$',...
'$I = 0.7 \ Kg.m^2$','$I = 0.9 \ Kg.m^2$','$I = 1.2 \ Kg.m^2$',...
'$I = 1.7 \ Kg.m^2$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',10);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.27 0.24]);
% distance between lines in legend [x,y,width, height]


%%
% Plant model from e_r, e_l to W_r, W_l
% Singular and Bode Plots for different values of m
%(variations in total mass without changing I_w)
```

```
% Bode Plot

md = 0; m = 3.4; % if d = 0;
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);
Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)

I = [0.42499999999 0.42500 0.42500000001 0.4292 0.4462 0.4675 0.3825];


P1 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m);
P2 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+0.5);
P3 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1);
P4 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1.5);
P5 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2);
P6 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2.5);
P7 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+3);

figure;
bodemag(P1,P2,P3,P4,P5,P6,P7);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
$d \neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex');

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$m = 3.4 \ kg$','$m = 3.9 \ kg$','$m = 4.4 \ kg$',...
'$m = 4.9 \ kg$','$m = 5.4 \ kg$','$m = 5.9 \ kg$'},'Interpreter',...
'latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);

%% Singular Values Plot
winit   = -1;
wfin    = 2;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
```

```
P6 = sigma(P6,w); P7 = sigma(P7,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6); P7 = 20*log10(P7);
figure;
subplot(2,1,1);
semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
$d \neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
$d \neq 0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots




% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$m = 3.4 \ kg$','$m = 3.9 \ kg$','$m = 4.4 \ kg$',...
'$m = 4.9 \ kg$','$m = 5.4 \ kg$','$m = 5.9 \ kg$'},'Interpreter',...
'latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',1.2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
% distance between lines in legend [x,y,width, height]


%%
% Plant model from e_r, e_l to W_r, W_l
```

```
% Singular and Bode Plots for different values of R

% Bode Plot

% change in R results in change in IW, however, no significant
% difference is observed

md = 0; m = 3.4; % if d = 0;
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);
Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)

I = [0.42499999999 0.42500 0.42500000001 0.4292 0.4462 0.4675 0.3825];

R =  0.042; m_wheel = 0.096;
rm = 0.0248  ; m_motor = 0.224;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);

P1 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P2 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P3 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P4 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P5 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P6 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);


R = 0.042; Iw = 1.67e-06;
figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
```

```
'Frequency Response $ (e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
$ d\neq0$','FontWeight','bold','FontSize',14, 'Interpreter','latex');

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);       % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$R = 0.042 \ m$','$R = 0.052 \ m$',...
'$R = 0.062 \ m$','$R = 0.072 \ m$','$R = 0.082 \ m$',...
'$R = 0.092 \ m$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
% distance between lines in legend [x,y,width, height]

%% Singular Values Plot
winit   = -1;
wfin    = 2;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6);
figure;
subplot(2,1,1);
semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
 w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
$d \neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);       % Lines with thicker width for plots

subplot(2,1,2);
semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
```

```
title(...
'Min Singular Values $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
$d \neq 0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);       % Lines with thicker width for plots




% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$R = 0.042 \ m$','$R = 0.052 \ m$',...
'$R = 0.062 \ m$','$R = 0.072 \ m$','$R = 0.082 \ m$',...
'$R = 0.092 \ m$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',1.2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
% distance between lines in legend [x,y,width, height]
```

```
% Trade Studies at d ~= 0
%continuation of the ppt_2_1_d_nonzero with the singular value plots
clc
close all
clear all
s = tf([1 0],[1]);
md = 0; m = 3.4;
winit   = -1;
wfin    = 2;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
%% Different Plant Models with the respective parameters as input
% at d = 0
% Plant model from e_r, e_l to W_r, W_l decoupled
% Plant model from (e_r + e_l), (e_r-e_l) to V, W decoupled
%%
% Plant model from e_r, e_l to W_r, W_l
% Singular and Bode Plots for different values of Veq
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);
Plant1 = Plantww(d, Veq, Weq, dw, Iw, I, L, md,R);

Veq = [0.1 0.2 0.6 1 3 5];

P1 = Plantww(d, Veq(1), Weq, dw, Iw, I, L, md,R);
P2 = Plantww(d, Veq(2), Weq, dw, Iw, I, L, md,R);
P3 = Plantww(d, Veq(3), Weq, dw, Iw, I, L, md,R);
P4 = Plantww(d, Veq(4), Weq, dw, Iw, I, L, md,R);
P5 = Plantww(d, Veq(5), Weq, dw, Iw, I, L, md,R);
```

```
P6 = Plantww(d, Veq(6), Weq, dw, Iw, I, L, md,R);

P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6);

figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
$d \neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
 w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
$d \neq 0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$v_{eq} = 0.1 \ m/s$','$v_{eq} = 0.2 \ m/s$',...
'$v_{eq} = 0.6 \ m/s$','$v_{eq} = 1.0 \ m/s$','$v_{eq} = 3.0 \ m/s$',...
'$v_{eq} = 5.0 \ m/s$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.26 0.24]);
% distance between lines in legend [x,y,width, height]

%% Singular and Bode Plots for different values of Weq
```

```
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);

Weq = [-8.0 -2.5 -0.5 0.5 2.5 8.0];

P1 = Plantww(d, Veq, Weq(1), dw, Iw, I, L, md,R);
P2 = Plantww(d, Veq, Weq(2), dw, Iw, I, L, md,R);
P3 = Plantww(d, Veq, Weq(3), dw, Iw, I, L, md,R);
P4 = Plantww(d, Veq, Weq(4), dw, Iw, I, L, md,R);
P5 = Plantww(d, Veq, Weq(5), dw, Iw, I, L, md,R);
P6 = Plantww(d, Veq, Weq(6), dw, Iw, I, L, md,R);

P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6);

figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
 w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $ e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
$d \neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
 w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
 $d \neq 0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
```

```
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$\omega_{eq} = -8.0 \ rad/s$',...
'$\omega_{eq} = -2.5 \ rad/s$','$\omega_{eq} = -0.5 \ rad/s$',...
'$\omega_{eq} = 0.5 \ rad/s$','$\omega_{eq} = 2.5 \ rad/s$',...
'$\omega_{eq} = 8.0 \ rad/s$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',10);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.28 0.24]);
% distance between lines in legend [x,y,width, height]

%% %% Singular and Bode Plots for different values of d
% the behaviour in the bode plots can be associated with the dominat...
pole
% variation wrt to d
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);

d = [-0.09 -0.08 -0.04 0.04 0.08 0.09];
I = [I_Newcalculation(d(1),Iw,L,md,dw) I_Newcalculation(d(2),Iw,L,md,dw)
    I_Newcalculation(d(3),Iw,L,md,dw) I_Newcalculation(d(4),Iw,L,md,dw)
    I_Newcalculation(d(5),Iw,L,md,dw)
    I_Newcalculation(d(6),Iw,L,md,dw)];

P1 = Plantww(d(1), Veq, Weq, dw, Iw, I(1), L, md,R);
P2 = Plantww(d(2), Veq, Weq, dw, Iw, I(2), L, md,R);
P3 = Plantww(d(3), Veq, Weq, dw, Iw, I(3), L, md,R);
P4 = Plantww(d(4), Veq, Weq, dw, Iw, I(4), L, md,R);
P5 = Plantww(d(5), Veq, Weq, dw, Iw, I(5), L, md,R);
P6 = Plantww(d(6), Veq, Weq, dw, Iw, I(6), L, md,R);

P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6);

figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:)...
w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
```

```
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
$d \neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);       % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
 w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values $(e_r,e_l)\rightarrow(\omega_r,\omega_l) $ for
$d\neq 0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);       % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$d = -0.1 \ m$','$d = -0.05 \ m$',...
'$d = -0.02 \ m$','$d = 0.02 \ m$','$d = 0.05 \ m$','$d = 0.1 \ m$'},...
'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',10);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.26 0.24]);
% distance between lines in legend [x,y,width, height]

%% %% Singular and Bode Plots for different values of I
% the behaviour in the bode plots can be associated with the dominat
% pole variation wrt to d
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);

I = [0.4 0.5 0.7 0.9 1.2 1.7];

P1 = Plantww(d, Veq, Weq, dw, Iw, I(1), L, md,R);
P2 = Plantww(d, Veq, Weq, dw, Iw, I(2), L, md,R);
P3 = Plantww(d, Veq, Weq, dw, Iw, I(3), L, md,R);
P4 = Plantww(d, Veq, Weq, dw, Iw, I(4), L, md,R);
P5 = Plantww(d, Veq, Weq, dw, Iw, I(5), L, md,R);
P6 = Plantww(d, Veq, Weq, dw, Iw, I(6), L, md,R);
```

```
P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6);

figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
$d \neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values $(e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
$d \neq 0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$I = 0.4 \ Kg.m^2$','$I = 0.5 \ Kg.m^2$',...
'$I = 0.7 \ Kg.m^2$','$I = 0.9 \ Kg.m^2$','$I = 1.2 \ Kg.m^2$',...
'$I = 1.7 \ Kg.m^2$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',10);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.27 0.24]);
% distance between lines in legend [x,y,width, height]

%%
% Plant model from e_r + e_l, e_r - e_l to V,W
% Singular and Bode Plots for different values of m
```

```
$(variations in total mass without changing I_w)

% Bode Plot

md = 0; m = 3.4; % if d = 0;
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);
Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)

I = [0.42499999999 0.42500 0.42500000001 0.4292 0.4462 0.4675 0.3825];


P1 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m);
P2 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+0.5);
P3 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1);
P4 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1.5);
P5 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2);
P6 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2.5);
P7 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+3);

figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
$d\neq0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$m = 3.4 \ kg$','$m = 3.9 \ kg$','$m = 4.4 \ kg$',...
'$m = 4.9 \ kg$','$m = 5.4 \ kg$','$m = 5.9 \ kg$'},'Interpreter',...
'latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
 % distance between lines in legend [x,y,width, height]

%% Singular Values Plot
winit   = -1;
wfin    = 2;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
```

```
P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w); P7 = sigma(P7,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6); P7 = 20*log10(P7);
figure;
subplot(2,1,1);
semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $((e_r + e_l,e_r - e_l)\rightarrow(v,\omega))$ for
$d\neq0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
$d\neq0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots




% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$m = 3.4 \ kg$','$m = 3.9 \ kg$','$m = 4.4 \ kg$',...
'$m = 4.9 \ kg$','$m = 5.4 \ kg$','$m = 5.9 \ kg$'},'Interpreter',...
'latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',1.2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
% distance between lines in legend [x,y,width, height]

%%
```

```
% Plant model from e_r+ e_l, e_r+ e_l to V, W
% Singular and Bode Plots for different values of R

% Bode Plot

% change in R results in change in IW, however, no significant
% difference is observed

md = 0; m = 3.4; % if d = 0;
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);
Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)

I = [0.42499999999 0.42500 0.42500000001 0.4292 0.4462 0.4675 0.3825];

R =  0.042; m_wheel = 0.096;
rm = 0.0248  ; m_motor = 0.224;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);

P1 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P2 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P3 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P4 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P5 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
R = R+0.01;
Iw =  0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
I = I_Newcalculation(0,Iw,L,md,dw);
P6 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);


R = 0.042; Iw = 1.67e-06;
figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
 % size and brightness of grid and size of x & y axis numbers
```

```
title(...
'Frequency Response $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
 $d\neq0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$R = 0.042 \ m$','$R = 0.052 \ m$',...
'$R = 0.062 \ m$','$R = 0.072 \ m$','$R = 0.082 \ m$',...
'$R = 0.092 \ m$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
 % distance between lines in legend [x,y,width, height]

%% Singular Values Plot
winit    = -1;
wfin     = 2;
nwpts    = 200;
w        = logspace(winit,wfin,nwpts);
P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
 P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6);
figure;
subplot(2,1,1);
semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
$d\neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
```

```
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
$d\neq0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots




% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$R = 0.042 \ m$','$R = 0.052 \ m$',...
'$R = 0.062 \ m$','$R = 0.072 \ m$','$R = 0.082 \ m$',...
'$R = 0.092 \ m$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',1.2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.26]);
% distance between lines in legend [x,y,width, height]
```

```
% Trade Studies at d ~= 0 Contd.
clc
close all
clear all
s = tf([1 0],[1]);
md = 0; m = 3.4; % if d = 0;
%% Different Plant Models with the respective parameters as input
% at d = 0
% Plant model from e_r, e_l to W_r, W_l decoupled
% Plant model from (e_r + e_l), (e_r-e_l) to V, W decoupled
%%
% Plant model from (e_r + e_l), (e_r-e_l) to V, W decoupled
% Singular and Bode Plots for different values of Veq
d = 0.1; Veq = 0; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);
Plant1 = Plantsdv(d, Veq, Weq, dw, Iw, I, L, md,R);

Veq = [0.1 0.2 0.6 1 3 5];

P1 = Plantsdv(d, Veq(1), Weq, dw, Iw, I, L, md,R);
P2 = Plantsdv(d, Veq(2), Weq, dw, Iw, I, L, md,R);
P3 = Plantsdv(d, Veq(3), Weq, dw, Iw, I, L, md,R);
P4 = Plantsdv(d, Veq(4), Weq, dw, Iw, I, L, md,R);
P5 = Plantsdv(d, Veq(5), Weq, dw, Iw, I, L, md,R);
P6 = Plantsdv(d, Veq(6), Weq, dw, Iw, I, L, md,R);

figure;
bodemag(P1,P2,P3,P4,P5,P6);
```

```
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $(e_r + e_l, e_r - e_l)\rightarrow(v,\omega)$ for
$d\neq0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$v\_{eq} = 0.1 \ m/s$','$v\_{eq} = 0.2 \ m/s$',...
'$v\_{eq} = 0.6 \ m/s$','$v\_{eq} = 1.0 \ m/s$',...
'$v\_{eq} = 3.0 \ m/s$','$v\_{eq} = 5.0 \ m/s$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.26 0.24]);
% distance between lines in legend [x,y,width, height]

%% Singular and Bode Plots for different values of Weq
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);

Weq = [-8.0 -2.5 -0.5 0.5 2.5 8.0];

P1 = Plantsdv(d, Veq, Weq(1), dw, Iw, I, L, md,R);
P2 = Plantsdv(d, Veq, Weq(2), dw, Iw, I, L, md,R);
P3 = Plantsdv(d, Veq, Weq(3), dw, Iw, I, L, md,R);
P4 = Plantsdv(d, Veq, Weq(4), dw, Iw, I, L, md,R);
P5 = Plantsdv(d, Veq, Weq(5), dw, Iw, I, L, md,R);
P6 = Plantsdv(d, Veq, Weq(6), dw, Iw, I, L, md,R);

figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
 $d\neq0$ ','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots
```

```

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$\omega\_{eq} = -8.0 \ rad/s$',...
'$\omega\_{eq} = -2.5 \ rad/s$','$\omega\_{eq} = -0.5 \ rad/s$',...
'$\omega\_{eq} = 0.5 \ rad/s$','$\omega\_{eq} = 2.5 \ rad/s$',...
'$\omega\_{eq} = 8.0 \ rad/s$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',10);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.28 0.24]);
% distance between lines in legend [x,y,width, height]

%% %% Singular and Bode Plots for different values of d
% the behaviour in the bode plots can be associated with the dominat
% pole variation wrt to d
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);

d = [-0.08 -0.07 -0.06 0.01 0.04 0.08];
I = [I_Newcalculation(d(1),Iw,L,md,dw) I_Newcalculation(d(2),Iw,L,md,dw)
    I_Newcalculation(d(3),Iw,L,md,dw) I_Newcalculation(d(4),Iw,L,md,dw)
    I_Newcalculation(d(5),Iw,L,md,dw)
    I_Newcalculation(d(6),Iw,L,md,dw)];

P1 = Plantsdv(d(1), Veq, Weq, dw, Iw, I(1), L, md,R);
P2 = Plantsdv(d(2), Veq, Weq, dw, Iw, I(2), L, md,R);
P3 = Plantsdv(d(3), Veq, Weq, dw, Iw, I(3), L, md,R);
P4 = Plantsdv(d(4), Veq, Weq, dw, Iw, I(4), L, md,R);
P5 = Plantsdv(d(5), Veq, Weq, dw, Iw, I(5), L, md,R);
P6 = Plantsdv(d(6), Veq, Weq, dw, Iw, I(6), L, md,R);

figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
 % size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
$d\neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);       % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$d = -0.08 \ m$','$d = -0.06 \ m$',...
'$d = -0.04 \ m$','$d = 0.01 \ m$','$d = 0.04 \ m$',...
```

```
'$d = 0.08 \ m$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',10);               % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);               % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.26 0.24]);
% distance between lines in legend [x,y,width, height]

%% plot of dominant pole vs d
figure;
load('dpole_3_1.mat');
plot(h(1,:),h(2,:));
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('d (m)','FontSize',12);
ylabel('Dominant Pole ','FontSize',12);
set(h_axes,'LineWidth',2,'FontSize',12,'GridAlpha',0.15);
% size and brightness of grid and size of x & y axis numbers
title('Dominant Pole vs $d$','FontWeight','bold','FontSize',14, ...
'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.8);          % Lines with thicker width for
% plots


%% %% Singular and Bode Plots for different values of I
% the behaviour in the bode plots can be associated with the dominat
% pole variation wrt to d
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);

I = [0.4 0.5 0.7 0.9 1.2 1.7];

P1 = Plantsdv(d, Veq, Weq, dw, Iw, I(1), L, md,R);
P2 = Plantsdv(d, Veq, Weq, dw, Iw, I(2), L, md,R);
P3 = Plantsdv(d, Veq, Weq, dw, Iw, I(3), L, md,R);
P4 = Plantsdv(d, Veq, Weq, dw, Iw, I(4), L, md,R);
P5 = Plantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R);
P6 = Plantsdv(d, Veq, Weq, dw, Iw, I(6), L, md,R);

figure;
bodemag(P1,P2,P3,P4,P5,P6);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Frequency Response $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
 $d\neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')
```

```
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);       % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$I = 0.4 \ Kg.m^2$','$I = 0.5 \ Kg.m^2$',...
'$I = 0.7 \ Kg.m^2$','$I = 0.9 \ Kg.m^2$','$I = 1.2 \ Kg.m^2$',...
'$I = 1.7 \ Kg.m^2$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',10);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.27 0.24]);
% distance between lines in legend [x,y,width, height]
```

```
% Trade Studies at d ~= 0 Contd.

clc
close all
clear all
s = tf([1 0],[1]);
md = 0; m = 3.4;
winit   = -1;
wfin    = 2;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
%% Different Plant Models with the respective parameters as input
% at d = 0
% Plant model from e_r, e_l to W_r, W_l decoupled
% Plant model from (e_r + e_l), (e_r-e_l) to V, W decoupled
%%
% Plant model from (e_r + e_l), (e_r-e_l) to V, W decoupled
% Singular and Bode Plots for different values of Veq
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);
Plant1 = Plantww(d, Veq, Weq, dw, Iw, I, L, md,R);

Veq = [0.1 0.2 0.6 1 3 5];

P1 = Plantsdv(d, Veq(1), Weq, dw, Iw, I, L, md,R);
P2 = Plantsdv(d, Veq(2), Weq, dw, Iw, I, L, md,R);
P3 = Plantsdv(d, Veq(3), Weq, dw, Iw, I, L, md,R);
P4 = Plantsdv(d, Veq(4), Weq, dw, Iw, I, L, md,R);
P5 = Plantsdv(d, Veq(5), Weq, dw, Iw, I, L, md,R);
P6 = Plantsdv(d, Veq(6), Weq, dw, Iw, I, L, md,R);

P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
```

```
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6);

figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:), ...
w, P6(1,:))
%clear sve_r, e_l to W_r, W_l
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values  $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
$d\neq0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:), ...
w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values  $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
$d\neq0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$v_{eq} = 0.1 \ m/s$','$v_{eq} = 0.2 \ m/s$',...
'$v_{eq} = 0.6 \ m/s$','$v_{eq} = 1.0 \ m/s$','$v_{eq} = 3.0 \ m/s$',...
'$v_{eq} = 5.0 \ m/s$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.26 0.24]);
% distance between lines in legend [x,y,width, height]

%% Singular and Bode Plots for different values of Weq
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
```

```
[max,min] = Imaxmin(d,Iw,L,md,dw);

Weq = [-8.0 -2.5 -0.5 0.5 2.5 8.0];

P1 = Plantsdv(d, Veq, Weq(1), dw, Iw, I, L, md,R);
P2 = Plantsdv(d, Veq, Weq(2), dw, Iw, I, L, md,R);
P3 = Plantsdv(d, Veq, Weq(3), dw, Iw, I, L, md,R);
P4 = Plantsdv(d, Veq, Weq(4), dw, Iw, I, L, md,R);
P5 = Plantsdv(d, Veq, Weq(5), dw, Iw, I, L, md,R);
P6 = Plantsdv(d, Veq, Weq(6), dw, Iw, I, L, md,R);

P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6);

figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:), ...
w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values  $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
$d\neq0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
 w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values  $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
 $d\neq0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$\omega_{eq} = -8.0 \ rad/s$',...
'$\omega_{eq} = -2.5 \ rad/s$','$\omega_{eq} = -0.5 \ rad/s$',...
```

```
'$\omega_{eq} = 0.5 \ rad/s$','$\omega_{eq} = 2.5 \ rad/s$',...
'$\omega_{eq} = 8.0 \ rad/s$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',10);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.28 0.24]);
% distance between lines in legend [x,y,width, height]

%% %% Singular and Bode Plots for different values of d
% the behaviour in the bode plots can be associated with the dominat
% pole variation wrt to d
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);

d = [-0.08 -0.07 -0.06 0.01 0.04 0.08];
I = [I_Newcalculation(d(1),Iw,L,md,dw) I_Newcalculation(d(2),Iw,L,md,dw)
    I_Newcalculation(d(3),Iw,L,md,dw) I_Newcalculation(d(4),Iw,L,md,dw)
    I_Newcalculation(d(5),Iw,L,md,dw)
    I_Newcalculation(d(6),Iw,L,md,dw)];

P1 = Plantsdv(d(1), Veq, Weq, dw, Iw, I(1), L, md,R);
P2 = Plantsdv(d(2), Veq, Weq, dw, Iw, I(2), L, md,R);
P3 = Plantsdv(d(3), Veq, Weq, dw, Iw, I(3), L, md,R);
P4 = Plantsdv(d(4), Veq, Weq, dw, Iw, I(4), L, md,R);
P5 = Plantsdv(d(5), Veq, Weq, dw, Iw, I(5), L, md,R);
P6 = Plantsdv(d(6), Veq, Weq, dw, Iw, I(6), L, md,R);

P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6);

figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
 w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values  $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
$d\neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);       % Lines with thicker width for plots
```

```
subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
 w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values  $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
$d\neq0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$d = -0.1 \ m$','$d = -0.05 \ m$',...
'$d = -0.02 \ m$','$d = 0.02 \ m$','$d = 0.05 \ m$',...
'$d = 0.1 \ m$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',10);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.26 0.24]);
% distance between lines in legend [x,y,width, height]

%% %% Singular and Bode Plots for different values of I
% the behaviour in the bode plots can be associated with the dominat
% pole variation wrt to d
d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I = I_Newcalculation(d,Iw,L,md,dw);
[max,min] = Imaxmin(d,Iw,L,md,dw);

I = [0.4 0.5 0.7 0.9 1.2 1.7];

P1 = Plantsdv(d, Veq, Weq, dw, Iw, I(1), L, md,R);
P2 = Plantsdv(d, Veq, Weq, dw, Iw, I(2), L, md,R);
P3 = Plantsdv(d, Veq, Weq, dw, Iw, I(3), L, md,R);
P4 = Plantsdv(d, Veq, Weq, dw, Iw, I(4), L, md,R);
P5 = Plantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R);
P6 = Plantsdv(d, Veq, Weq, dw, Iw, I(6), L, md,R);

P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
P4 = sigma(P4,w); P5 = sigma(P5,w);
P6 = sigma(P6,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4); P5 = 20*log10(P5);
P6 = 20*log10(P6);
```

```
figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
 w, P6(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Max Singular Values  $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
$d\neq 0$','FontWeight','bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
 w, P6(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title(...
'Min Singular Values  $(e_r + e_l,e_r - e_l)\rightarrow(v,\omega)$ for
$d\neq0$','FontWeight','bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'$I = 0.4 \ Kg.m^2$','$I = 0.5 \ Kg.m^2$',...
'$I = 0.7 \ Kg.m^2$','$I = 0.9 \ Kg.m^2$','$I = 1.2 \ Kg.m^2$',...
'$I = 1.7 \ Kg.m^2$'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',10);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.27 0.24]);
% distance between lines in legend [x,y,width, height]
```

```
% Inner-Loop frequency response plots
clc
close all
clear all
s = tf([1 0],[1]);
md = 0; m = 3.4; % if d = 0;
%% Different Plant Models with the respective parameters as input
% at d = 0
% Plant model from e_r, e_l to W_r, W_l decoupled
```

```
d1 = 0; d2 = 0.08; Veq = 2; Weq = 0.8; % independent of Veq and Weq
L = 1; dw = 1/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;  A = m + 2*Iw/(R*R); % default values //\\
%has to be chosen based on the corresponding AR value
I1 = 0.4250; % I_AR
I2 = 1.7;% 4.0I_AR
%[max,min] = Imaxmin(d2,Iw,L,md,dw)
PlantD1M1 = Plantww(d1, Veq, Weq, dw, Iw, I1, L, md,R); I1 = 0.42560;
PlantD1M2 = Plantsdv(d1, Veq, Weq, dw, Iw, I1, L, md,R); I1 = 0.4250;

PlantD2M1 = Plantww(d1, Veq, Weq, dw, Iw, I2, L, md,R);
PlantD2M2 = Plantsdv(d1, Veq, Weq, dw, Iw, I2, L, md,R);

PlantD3M1 = Plantww(d2, Veq, Weq, dw, Iw, I1, L, md,R);
PlantD3M2 = Plantsdv(d2, Veq, Weq, dw, Iw, I1, L, md,R);


PlantD4M1 = Plantww(d2, Veq, Weq, dw, Iw, I2, L, md,R);
PlantD4M2 = Plantsdv(d2, Veq, Weq, dw, Iw, I2, L, md,R);

%bodemag(PlantD1M2, PlantD2M2)
PD1M1 = minreal(zpk(tf(PlantD1M1)));
PD1M2 = minreal(zpk(tf(PlantD1M2)));

PD2M1 = minreal(zpk(tf(PlantD2M1)));
PD2M2 = minreal(zpk(tf(PlantD2M2)));

PD3M1 = minreal(zpk(tf(PlantD3M1)));
PD3M2 = minreal(zpk(tf(PlantD3M2)));

PD4M1 = minreal(zpk(tf(PlantD4M1)));
PD4M2 = minreal(zpk(tf(PlantD4M2)));

KD1M1 = [0.6435 + 4.8633/s 0; 0 0.6435 + 4.8633/s]*(100/(s+100));
KD1M2 = [30.6428 + 231.5868/s 0; 0 10.8341  + 81.8799/s]*(100/(s+100));

KD2M1 = [1.0713 + 6.9057/s 0; 0 1.0713 + 6.9057/s]*(100/(s+100));
KD2M2 = [30.6428 + 231.5868/s 0; 0 46.5661  + 144.8851/s]*(100/(s+100));

KD3M1 = [0.63861 + 5.2248/s 0; 0 0.63861 + 5.2248/s]*(100/(s+100));
KD3M2 = [30.6428 + 231.5868/s 0; 0 10.6159  + 94.2519/s]*(100/(s+100));

KD4M1 = [1.0666 + 6.9621/s 0; 0 1.0666 + 6.9621/s]*(100/(s+100));
KD4M2 = [30.6428 + 231.5868/s 0; 0 46.3477  + 157.2715/s]*(100/(s+100));


WD1M1 = [ (4.8633/0.6435)/(s+ 4.8633/0.6435) 0;...
 0  (4.8633/0.6435)/(s+(4.8633/0.6435))]
WD1M2 = [ (231.5868/30.6428)/(s + 231.5868/30.6428)  0;...
 0   (81.8799/10.8341)/(s +  81.8799/10.8341)]

WD2M1 = [ (6.9057/1.0713)/(s + 6.9057/1.0713) 0;...
 0 (6.9057/1.0713)/(s + 6.9057/1.0713)]
WD2M2 = [ (231.5868/30.6428)/(s+231.5868/30.6428) 0;...
 0  (144.8851/46.5661)/(s+144.8851/46.5661)]
```

```
WD3M1 = [ (5.2248/0.63861)/(s+5.2248/0.63861) 0;...
 0 (5.2248/0.63861)/(s+5.2248/0.63861)]
WD3M2 = [ (231.5868/30.6428)/(s+231.5868/30.6428) 0;...
 0  (94.2519/10.6159)/(s+94.2519/10.6159)]

WD4M1 = [(6.9621/1.0666)/(s+6.9621/1.0666) 0;...
 0 (6.9621/1.0666)/(s+6.9621/1.0666)]
WD4M2 = [(231.5868/30.6428)/(s+231.5868/30.6428) 0;...
 0  (157.2715/46.3477)/(s+157.2715/46.3477)]


LD1M1 = PD1M1*KD1M1;
LD1M2 = PD1M2*KD1M2;

LD2M1 = PD2M1*KD2M1;
LD2M2 = PD2M2*KD2M2;

LD3M1 = PD3M1*KD3M1;
LD3M2 = PD3M2*KD3M2;

LD4M1 = PD4M1*KD4M1;
LD4M2 = PD4M2*KD4M2;
%%
SD1M1 = (eye(2) + LD1M1)^-1;
SD1M2 = (eye(2) + LD1M2)^-1;

SD2M1 = (eye(2) + LD2M1)^-1;
SD2M2 = (eye(2) + LD2M2)^-1;

SD3M1 = (eye(2) + LD3M1)^-1;
SD3M2 = (eye(2) + LD3M2)^-1;

SD4M1 = (eye(2) + LD4M1)^-1;
SD4M2 = (eye(2) + LD4M2)^-1;

%% complimentary sensitivity
CD1M1 = LD1M1*(eye(2) + LD1M1)^-1;
CD1M2 = LD1M2*(eye(2) + LD1M2)^-1;

CD2M1 = LD2M1*(eye(2) + LD2M1)^-1;
CD2M2 = LD2M2*(eye(2) + LD2M2)^-1;

CD3M1 = LD3M1*(eye(2) + LD3M1)^-1;
CD3M2 = LD3M2*(eye(2) + LD3M2)^-1;

CD4M1 = LD4M1*(eye(2) + LD4M1)^-1;
CD4M2 = LD4M2*(eye(2) + LD4M2)^-1;

%% Try
inM =  ([1/R dw/(2*R); 1/R -dw/(2*R)]);
M = [R/2 R/2; R/dw -R/dw];

TD1M1 = M*(LD1M1*(eye(2) + LD1M1)^-1)*WD1M1*inM;
TD1M2 = CD1M2*WD1M2;

TD2M1 = M*(LD2M1*(eye(2) + LD2M1)^-1)*WD2M1*inM;
TD2M2 = CD2M2*WD2M2;
```

```
TD3M1 = M*(LD3M1*(eye(2) + LD3M1)^-1)*WD3M1*inM;
TD3M2 = CD3M2*WD3M2;

TD4M1 = M*(LD4M1*(eye(2) + LD4M1)^-1)*WD4M1*inM;
TD4M2 = CD4M2*WD4M2;

%%  Tru
TRD1M1 = (KD1M1*(eye(2) + LD1M1)^-1)*WD1M1*inM;
TRD1M2 = KD1M2*SD1M2*WD1M2;

TRD2M1 = (KD2M1*(eye(2) + LD2M1)^-1)*WD2M1*inM;
TRD2M2 = KD2M2*SD2M2*WD2M2;

TRD3M1 = (KD3M1*(eye(2) + LD3M1)^-1)*WD3M1*inM;
TRD3M2 = KD3M2*SD3M2*WD3M2;

TRD4M1 = (KD4M1*(eye(2) + LD4M1)^-1)*WD4M1*inM;
TRD4M2 = KD4M2*SD4M2*WD4M2;


%% Open Loop
winit   = -1;
wfin    = 4;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
P1 = sigma(LD1M1,w); P2 = sigma(LD2M1,w); P3 = sigma(LD1M2,w);
P4 = sigma(LD2M2,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4);
figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Max Singular Values: Open Loop','FontWeight','bold',...
'FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Min Singular Values: Open Loop','FontWeight','bold',...
'FontSize',12, 'Interpreter','latex')
```

```matlab

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots


% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'D1M1','D2M1','D1M2','D2M2'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',1.2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.2 0.2]);
% distance between lines in legend [x,y,width, height]

%%
winit   = -1;
wfin    = 4;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
P1 = sigma(LD3M1,w); P2 = sigma(LD4M1,w); P3 = sigma(LD3M2,w);
P4 = sigma(LD4M2,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4);
figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
 % size and brightness of grid and size of x & y axis numbers
title('Max Singular Values: Open Loop','FontWeight','bold',...
'FontSize',14,'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Min Singular Values: Open Loop','FontWeight','bold',...
'FontSize',12,'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots


% Put legend and enhance appearance
```

```
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'D3M1','D4M1','D3M2','D4M2'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',1.2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.2 0.2]);
% distance between lines in legend [x,y,width, height]
%% Sensitivity

winit   = -1;
wfin    = 4;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
P1 = sigma(SD1M1,w); P2 = sigma(SD2M1,w); P3 = sigma(SD1M2,w);
P4 = sigma(SD2M2,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4);
figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Max Singular Values: Sensitivity','FontWeight','bold',...
'FontSize',14,'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Min Singular Values: Sensitivity','FontWeight','bold',...
'FontSize',12,'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots


% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'D1M1','D2M1','D1M2','D2M2'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',1.2);              % thickness of lines in legend
```

```
set(hL,'FontSize',1,'Position',[0.5 0.5 0.2 0.2]);
% distance between lines in legend [x,y,width, height]

%%

winit   = -1;
wfin    = 4;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
P1 = sigma(SD3M1,w); P2 = sigma(SD4M1,w); P3 = sigma(SD3M2,w);
 P4 = sigma(SD4M2,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4);
figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Max Singular Values: Sensitivity','FontWeight','bold',...
'FontSize',14,'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Min Singular Values: Sensitivity','FontWeight','bold',...
'FontSize',12,'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots


% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'D3M1','D4M1','D3M2','D4M2'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',1.2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.2 0.2]);
% distance between lines in legend [x,y,width, height]

%% Complimentary Sensitivty

```

```
winit    = -1;
wfin     = 4;
nwpts    = 200;
w        = logspace(winit,wfin,nwpts);
P1 = sigma(CD1M1,w); P2 = sigma(CD2M1,w); P3 = sigma(CD1M2,w);
P4 = sigma(CD2M2,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4);
figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Max Singular Values: Complementary Sensitivity','FontWeight',...
'bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);       % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Min Singular Values: Complementary Sensitivity','FontWeight',...
'bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);       % Lines with thicker width for plots


% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'D1M1','D2M1','D1M2','D2M2'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',1.2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.2 0.2]);
% distance between lines in legend [x,y,width, height]

%%

winit    = -1;
wfin     = 4;
nwpts    = 200;
w        = logspace(winit,wfin,nwpts);
P1 = sigma(CD3M1,w); P2 = sigma(CD4M1,w); P3 = sigma(CD3M2,w);
```

```
P4 = sigma(CD4M2,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4);
figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Max Singular Values: Complementary Sensitivity','FontWeight',...
'bold','FontSize',14, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Min Singular Values: Complementary Sensitivity','FontWeight',...
'bold','FontSize',12, 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots


% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'D3M1','D4M1','D3M2','D4M2'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',1.2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.2 0.2]);
% distance between lines in legend [x,y,width, height]

%% Try
figure;
bodemag(TD1M1,TD2M1,TD1M2,TD2M2,w);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Frequency Response','FontWeight','bold','FontSize',14, ...
'Interpreter','latex')
```

```
h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'D1M1','D2M1','D1M2','D2M2'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.2]);
% distance between lines in legend [x,y,width, height]


figure;
bodemag(TD3M1,TD4M1,TD3M2,TD4M2,w);
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Frequency Response','FontWeight','bold','FontSize',14, ...
'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.5);      % Lines with thicker width for plots

% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'D3M1','D4M1','D3M2','D4M2'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.2]);
% distance between lines in legend [x,y,width, height]

%% Tru

winit   = -1;
wfin    = 4;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
P1 = sigma(TRD1M1,w); P2 = sigma(TRD2M1,w); P3 = sigma(TRD1M2,w);
 P4 = sigma(TRD2M2,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4);
figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18); ...
```

```
% size and brightness of grid and size of x & y axis numbers
title('Max Singular Values: Tru','FontWeight','bold','FontSize',14,...
 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);       % Lines with thicker width for plots

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Min Singular Values: Tru','FontWeight','bold','FontSize',12,...
 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);       % Lines with thicker width for plots


% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'D1M1','D2M1','D1M2','D2M2'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');    %
set(hTL,'FontSize',11);             % font size for letters in legend
hTL=findobj(hObj,'type','line');    %
set(hTL,'LineWidth',1.2);             % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.2 0.2]);
% distance between lines in legend [x,y,width, height]
winit   = -1;

%%
wfin    = 4;
nwpts   = 200;
w       = logspace(winit,wfin,nwpts);
P1 = sigma(TRD3M1,w); P2 = sigma(TRD4M1,w); P3 = sigma(TRD3M2,w);
P4 = sigma(TRD4M2,w);
P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
P4 = 20*log10(P4);
figure;
subplot(2,1,1);
semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Max Singular Values: Tru','FontWeight','bold','FontSize',14,...
 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);       % Lines with thicker width for plots
```

```

subplot(2,1,2);
semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
%clear sv
grid on;
h_axes = findobj(gcf, 'type', 'axes');
xlabel('Frequency','FontSize',12);
ylabel('Magnitude','FontSize',12);
set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
% size and brightness of grid and size of x & y axis numbers
title('Min Singular Values: Tru','FontWeight','bold','FontSize',12,...
 'Interpreter','latex')

h_line = findobj(gcf, 'type', 'line');
set(h_line, 'LineWidth',1.2);      % Lines with thicker width for plots


% Put legend and enhance appearance
% Legend  bug with subscript, use '\_' instead of '_'
[hL,hObj]=legend({'D3M1','D4M1','D3M2','D4M2'},'Interpreter','latex');
hTL=findobj(hObj,'type','Text');     %
set(hTL,'FontSize',11);              % font size for letters in legend
hTL=findobj(hObj,'type','line');     %
set(hTL,'LineWidth',1.2);              % thickness of lines in legend
set(hL,'FontSize',1,'Position',[0.5 0.5 0.2 0.2]);
% distance between lines in legend [x,y,width, height]
```

```
% dominant pole vs d for increasing v_eq
clc
close all
clear all

loop = 3; % 1 - PID design %2 Wr,Wl design %3 ICC Paper Plant
s = tf([1 0],[1]);
md = 0; % if d = 0;
%% Different Plant Models with the respective parameters as input

% Plant model from e_r, e_l to W_r, W_l with coupling
% Plant model from e_r, e_l to W_r, W_l decoupled
% Plant model from e_r, e_l to V, W with coupling
% Plant model from (e_r + e_l), (e_r-e_l) to V, W decoupled
%%
% Plant model from e_r, e_l to W_r, W_l  (with coupling)
Veq = 1.0; % in m/s
Weq = 0.0; % in m/s
d = 0.28; % in m/s max value is 0.14 for hardware
L = 0.3536; dw = L*sqrt(2); R = 0.042; % default values L = 0.3536//\\
%has to be chosen based on the corresponding AR value (AR_calculation.m)
Iw = 1.67e-06;% default values //\\
%has to be chosen based on the corresponding AR value
I = I_cal(d,Iw,L,md,dw);
[ma,mi] = Imaxmin(d,Iw,L,md,dw);
i = 0;  kk = 1; h = ones(7,57)
for j = 0: 0.5:3
   Veq = j;
for d = -0.28:0.01:0.28
```

```
    i= i+1;
    I = I_cal(d,Iw,L,md,dw);
    Plant1 = Plantww(d, Veq, Weq, dw, Iw, I, L, md,R);
    dpole(i) = max(real(pole(Plant1)));
end
h(kk,:)=dpole;
kk = kk +1;
i = 0;
end
d = -0.28:0.01:0.28;
plot(d,h(1,:),d,h(2,:),d,h(3,:),d,h(4,:),d,h(5,:),d,h(6,:),d,h(7,:))
```

```
% e_r, e_l to V,W Plant
function [Plant] = Plantvw(d,Veq,Weq,dw,Iw,I,L,md,R)

% Plant model from e_r, e_l to V, W

s = tf([1 0],[1]);
z = tf('z');

%% Plant Model
m = 3.4; mc = 2.76 + md; I = I ; Iw = Iw; L = L; d = d; dw = dw;
Weq = Weq; Veq = Veq;
Ao = m + 2*Iw/(R^2); Bo = I + (2*Iw*dw^2)/R^2;
% Linear Plant without actuator - Motor Torque to V,W

% Linear Plant without actuator - Motor Torque to Wr, Wl
 Tmain2 = [(1/(s*R*R))*((1/Ao)+(L*L/Bo)) ...
  (1/(s*R*R))*((1/Ao)-(L*L/Bo)); ...
  (1/(s*R*R))*((1/Ao)-(L*L/Bo)) (1/(s*R*R))*((1/Ao)+(L*L/Bo))];

% Right Motor Actuator Dynamics - voltage to motor torque
Kt = 0.0337; Kg = 50; Kb = Kt; B = 1.3023e-04; La = 22.8e-06; Ra = 2.9;

% Left Motor Actuator Dynamics - voltage to motor torque
Kt = 0.0046; Kg = 50; Kb = Kt; Beta = 2.29e-06; La =1.729e-03;Ra = 5.51;

A = m + 2*Iw/(R*R); B = I + dw*dw*Iw/(2*R*R);

% ULTIMATE - CORRECT representation of Plant using State Space
%form derived on Sept 21st, 2019
As =[-2*Beta*Kg*Kg/(A*R*R)  2*mc*d*Weq/A   (Kt*Kg/(R*A))  (Kt*Kg/(A*R));
     -mc*d*Weq/B  (-mc*d*Veq/B)-(Beta*dw*dw*Kg*Kg/(2*R*R*B)) ...
     dw*Kt*Kg/(2*R*B) -dw*Kt*Kg/(2*B*R) ;
     -Kb*Kg/(La*R)   -Kb*Kg*dw/(2*La*R)  -Ra/La      0 ;
     -Kb*Kg/(La*R)    Kb*Kg*dw/(2*La*R)    0       -Ra/La ];
Bs = [0 0; 0 0; 1/La 0; 0 1/La]; Cs = [1 0 0 0; 0 1 0 0];Ds = [0 0;0 0];
Mains = ss(As,Bs,Cs,Ds);
MainTfs = minreal(tf(Mains)); Mains = ss(MainTfs);
MainTfs = minreal(zpk(MainTfs));

% Alternate representation of Plant from Lin's thesis ea -r,l to Wr,Wl
% H1 = Kt/(La*m*R*R*s*s + (Ra*m*R*R + 2*La*B)*s + (2*Kb*Kt + 2*Ra*B));
% H2 = dw*dw*Kt/(I*La*R*R*s*s + (I*Ra*R*R + dw*dw*La*B)*s +
%(Kb*Kt*L*L + dw*dw*Ra*B));
% Plant3 = [H1+0.5*H2  H1-0.5*H2; H1-0.5*H2 H1+0.5*H2];
```

```
%Plant3 = minreal(zpk(Plant3));
Plant = Mains;

end
```

```
 % e_r, e_l to W_r, W_l Plant
 function [Plant] = Plantww(d, Veq, Weq, dw, Iw, I, L, md,R)

% Plant model from e_r, e_l to W_r, W_l

R = R; dw = dw;

Temp = Plantvw(d,Veq,Weq, dw, Iw, I,L,md,R);
%MainTfs = ([1/R dw/(2*R); 1/R -dw/(2*R)])*Temp;     % This code is for
%obtaining the transfer function represntation of the plant
%MainTfs = minreal(tf(MainTfs)); MainTfs = minreal(zpk(MainTfs));

Temp.C =  ([1/R dw/(2*R); 1/R -dw/(2*R)])*Temp.C;
Plant = Temp;

end
```

```
% (e_r + e_l), (e_r-e_l) to V, W Plant
function [Plant] = Plantsdv(d, Veq, Weq, dw, Iw, I, L, md,R)

% Plant model from (e_r + e_l), (e_r-e_l) to V, W

 dw = dw;

Temp = Plantvw(d,Veq,Weq,dw,Iw,I,L,md,R);
%MainTfs = Temp*[0.5 0.5; 0.5 -0.5];      % this code is for transfer
%function representation of the plant
%MainTfs = minreal(tf(MainTfs)); MainTfs = minreal(zpk(MainTfs));

Temp.B = Temp.B*[0.5 0.5; 0.5 -0.5];
Plant = Temp;

end
```

```
% I as a function of d; d is varied by shifting m_c to new location (db)
function [I] = I_cal(d,Iw,L,md,dw)
% I as a function of d, dw
% range of d is -0.03 to 0.03
 m = 3.4;
 mc = 2.76; % mass without motors
 mw = (m - mc)*0.5 ; % mass of individual motor and wheel
 %dw = 0.25*167;
 % the Li-ion battery, camera and the Lipo battery are shifted to match
 % the new d value, so no new mass is being added md = 0 always
%% Iw estimation  (max and min value estimation)

 %rw =  0.1; m_wheel = 0.181;
 %rm = 0.0248  ; m_motor = 0.096;
```

```
 %maxval =  0.5*m_motor*rm*rm + 0.5*m_wheel*rw*rw;
% minval =  maxval/8;

%% Ic calculation

m_plate = 0.411;


db = d*m/(mc);


Ic = mc*(db - d)*(db - d);


%% I approximation

I_approx = (1/12)*m*(L*L + dw*dw); % in order to verify
%if the calculated I is right or wrong

%% I original

% Iw = wheel+motor moment of inertia about wheel axel
I = Ic + (mc+md)*d*d + 2*mw*dw*dw + Iw;


end
```

```
%I as a function of d; d is varied by shifting camera and LiPo battery
%location
function [I] = I_Newcalculation(d,Iw,L,md,dw)
% I as a function of d, dw
% range of d is -0.03 to 0.03
 m = 3.4;
 mc = 2.76; % mass without motors
 mw = (m - mc)*0.5 ; % mass of individual motor and wheel
 %dw = 0.25*167;
 % the Li-ion battery, camera and the Lipo battery are shifted to match
 % the new d value, so no new mass is being added md = 0 always
%% Iw estimation  (max and min value estimation)

 %rw =  0.042; m_wheel = 0.181;
 %rm = 0.0248  ; m_motor = 0.096;
 %maxval =  0.5*m_motor*rm*rm + 0.5*m_wheel*rw*rw;
%minval =  maxval/8;

%% Ic calculation
%
% plate
Ic = md*(L/2)^2; % initial value, md is the
%additional mass that has to be added to manipulate d,
m_plate = 0.411;


Ic = Ic + (2/12)*m_plate*(L*L + dw*dw) + 2*m_plate*d*d;
% for the two acryllic sheets
% 3d print
```

```
% m3d = 0.055;
% l3d = 0.079;
% Ic = Ic + 2*m3d*l3d*l3d;
% Nvidia
m_nvidia = 0.4;
l_nvidia = 0.12*dw;
Ic = Ic + m_nvidia*((l_nvidia^2) + (d^2));
m_bat = 0.492;
m_lipo = 0.185;
m_cam = 0.077;
db = d*mc/(m_bat + m_cam + m_lipo);

% battery
m_bat = 0.492;
l_cam = 0.404*L;
l = 0.108;
w = 0.101;5
Ic = Ic + (1/12)*m_bat*(l*l + w*w) + m_bat*(db - d)*(db - d);
% camera
m_cam = 0.077;
l_cam = 0.404*L;
Ic = Ic + m_cam*(db - d)*(db - d);
% LiPo
m_lipo = 0.185;
l_lipo = 0.404*L;
l = 0.10;
w = 0.034;
Ic = Ic + m_lipo*(db - d)*(db - d) + (1/12)*m_lipo*(l*l + w*w);
% arduino + motor shield
m_ard_shield = 0.062;
L_ard_shield = 0.033;
Ic = Ic + m_ard_shield*(L_ard_shield)^2; % doesn't really matter
%% I approximation

I_approx = (1/12)*m*(L*L + dw*dw); % in order to verify if
%the calculated I is right or wrong
%% I original

% Iw = wheel+motor moment of inertia about wheel axel
I = Ic + (mc+md)*d*d + 2*mw*dw*dw + Iw;

end
```

APPENDIX C

ROS AND ARDUINO CODE

```
// Description: Arduino code for generic PI inner-loop
#include <Servo.h>
#include <math.h>
#include <ros.h>
#include <ros/time.h>
#include <tf/tf.h>
#include <tf/transform_broadcaster.h>
#include <nav_msgs/Odometry.h>
#include <geometry_msgs/Vector3.h>
#include <geometry_msgs/Vector3Stamped.h>
#include <geometry_msgs/Twist.h>
#include <std_msgs/Int8.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>

Servo left;  // create servo object to control right motor
Servo right; // create servo object to control left motor

unsigned long Time=0; // Starting time
unsigned long lastMilli = 0;
double td = 0.0095; // T = 0.01 sec (100 hz)
unsigned long sample_time= td*1000*0.1 ;

double wd ;    // Desired angular speed of COM about ICC(Instantaneous
//center of curvature)
double vd ;    // Desired longitudinal speed of center of mass

double wR;      // present angular speed of right motor
double wL;      // present angular speed of left motor
double wRp=0.0; // previous angular speed right motor
double wLp=0.0; // previous angular speed left motor
double wLn;     // average angular speed (wL + wLp)/2
double wRn;     // average angular speed (wR + wRp)/2

double CPR = 1024.0; // encoder counts per revolution
double LdVal = 0.0;
double RdVal = 0.0;
long Lcount; // Present Encoder value
long Rcount; // Present Encoder value
long Lcount_last=0; // Previous encoder value
long Rcount_last=0;   // Previous encoder value

double Radius = 0.04; // Change it (radius of wheel) 0.045
double Length = 0.32; // Change it (distance between wheels) 0.555 0.308

double wdr = 0;       // Desired angular speed of right wheel using wd &
// vd prefilter parameter x_{n+1}
double wdl = 0;       // Desired angular speed of left wheel using wd &
//vd prefilter parameter x_{n+1}
double wdr_p= 0.0;   // prefilter parameter x_{n} for right motor
double wdl_p= 0.0;   // prefilter parameter x_{n} for left motor
double wrf;       // prefilter parameter y_{n+1} for right motor
double wlf;       // prefilter parameter y_{n+1} for left motor
double wrf_p= 0.0;   // prefilter paramter y_{n} for right motor
double wlf_p= 0.0;   // prefilter parameter y_{n} for left motor'
```

```
double CR;       // Controller output y_{n+2} Right motor
double CR_p=0.0;   // Controller output y_{n+1} Right motor
double CR_pp=0.0;  // Controller output y_{n}   Right motor
double CR_ppp=0.0;
double CR_pppp=0.0;
double CL;       // Controller output y_{n+2} Left motor
double CL_p=0.0;   // Controller output y_{n+1} Left motor
double CL_pp=0.0;  // Controller output y_{n}   Left motor
double CL_ppp=0.0;
double CL_pppp=0.0;

double Lerror;   // Lerror = wlf(output of prefilter/ reference speed)
// - wLn.....or... Controller input x_{n+2}
double Lerror_p = 0.0; // Controller input x_{n+1}
double Lerror_pp = 0.0; // Controller input x_{n}
double Lerror_ppp = 0.0;
double Lk = 0.0;
double Rerror;   // Rerror = wrf(output of prefilter/ reference speed)
// - wRn....or.....Controller input x_{n+2}
double Rerror_p = 0.0; // Controller input x_{n+1}
double Rerror_pp = 0.0; // Controller input x_{n}
double Rerror_ppp = 0.0;
double Rk = 0.0;

double Lx = 0.0; // left - integrator anti-windup
double Rx = 0.0; // right - integrator anti-windup

double PWMR; // Controller output for right motor
double PWML; // Controller output for left motor
int val; // input to the motors

double A ;          // Controller gain kp of
// K = (kp + ki/s) * (100/(s+100))
double B ;          // Controller gain ki
double C ;          // Controller gain kp of
// K = (kp + ki/s) * (100/(s+100))
double Kp = 0.5; double Ki; double Kd; long long EN; long long DE;
long long DE1;
double ta = 1/1260;
double Po = 0.0; double scale;
double g = 1.0; double z;          // Controller gain ki
double alpha = 200.0;  // Roll off parameter alpha
double h ;     //  prefilter parameter z = ki/kp
//obtained from K = (g(s+z)/s)*(100/(s+100))
// for PD controller double b1; double b0; double c1; double c0;
// double A;


// Subscriber call back to /cmd_vel
void twist_message_cmd(const geometry_msgs::Twist& msg)
{
  wdr = msg.linear.x  ;
  wdl = msg.angular.x ;
  //wdl = wdr;
  //if (wdr == 0) g = 0.0;
  //else g = 1.0;
```

```
  g = msg.linear.z;
  //z = msg.angular.z;
}


// Node handle
ros::NodeHandle arduino_nh ;

//geometry_msgs::Twist msg ;
//geometry_msgs::Vector3Stamped rpm_msg;
geometry_msgs::Twist rpm_msg ;


// Publisher of the right and left wheel angular velocities
//ros::Publisher pub("robot_2/arduino_vel", &rpm_msg); // Add robot_*
ros::Publisher pub("arduino_vel", &rpm_msg);

// Subscriber of the reference velocities coming from the outerloop
ros::Subscriber<geometry_msgs::Twist> sub("cmd_vel",...
&twist_message_cmd);
//ros::Subscriber<std_msgs::Int8> sub2("emergency_stop", &callBack );


// Left Encoder
#define LH_ENCODER_A PK0 //  pin A8 (PCINT16)
#define LH_ENCODER_B PK1 //  pin A9 (PCINT17)
static long left_ticks = 0L;
volatile bool LeftEncoderBSet ;

// Right Encoder
#define RH_ENCODER_A PB0  // Digital pin 53 (PCINT 0)
#define RH_ENCODER_B PB1  // Digital pin 52 (PCINT 1)
static long right_ticks = 0L;
volatile bool RightEncoderBSet ;

#define LEFT  0
#define RIGHT 1

static const int8_t ENC_STATES [] = {0,1,-1,0,-1,0,0,1,1,0,0,-1,0,-1,...
1,0};
//encoder lookup table

/* Interrupt routine for LEFT encoder, taking care of actual counting */
ISR (PCINT2_vect) // pin change interrupts for port K (A8,A9)
{
  static uint8_t enc_last=0;
  enc_last <<=2; //shift previous state two places
  enc_last |= (PINK & (3 << 0)) >> 0 ;
  left_ticks -= ENC_STATES[(enc_last & 0x0f)]; // changed from -ve to
  // +ve after interchanging the M1A and M1B wires
}

/* Interrupt routine for RIGHT encoder, taking care of actual
// counting */
ISR (PCINT0_vect) // pin change interrupts for port J
// (Digital pin 14,15)
{
```

```
  static uint8_t enc_last=0;
  enc_last <<=2; //shift previous state two places
  enc_last |= (PINB & (3 << 0)) >> 0 ;
  right_ticks -= ENC_STATES[(enc_last & 0x0f)]; // changed from -ve to
  // +ve after interchanging the M1A and M1B wires
}

void SetupEncoders()
{
  // Initializing the encoder pins as input pins

  // set as inputs DDRD(pins 0-7) , DDRC(A0-A5)
  // (The Port D Data Direction Register - read/write)
  DDRK &= ~(1<<LH_ENCODER_A); // PK0 pin A8
  DDRK &= ~(1<<LH_ENCODER_B); // PK1 pin A9
  DDRB &= ~(1<<RH_ENCODER_A); // Digital pin 53 (PB0)
  DDRB &= ~(1<<RH_ENCODER_B); // Digital pin 52 (PB1)

  /* Pin to interrupt map:
   * D0-D7 = PCINT 16-23 = PCIR2 = PD = PCIE2 = pcmsk2
   * D8-D13 = PCINT 0-5 = PCIR0 = PB = PCIE0 = pcmsk0
   * A0-A5 (D14-D19) = PCINT 8-13 = PCIR1 = PC = PCIE1 = pcmsk1
  */

  /*
     For Atmega 2560 pin change interrupt enable flags
     PCIE2 : PCINT23-16
     PCIE1 : PCINT15-8
     PCIE0 : PCINT7-0
  */

  // tell pin change mask to listen to left encoder pins and right pins
  PCMSK2 |= (1 << LH_ENCODER_A)|(1 << LH_ENCODER_B);
  PCMSK0 |= (1 << RH_ENCODER_A)|(1 << RH_ENCODER_B);

  // enable PCINT1 and PCINT2 interrupt in the general interrupt mask
  // the Pin Change Interrupt Enable flags have to be set in the PCICR
  //  register. These are bits PCIE0, PCIE1 and PCIE2 for the groups of
  // pins PCINT7..0, PCINT14..8 and PCINT23..16 respectively
  PCICR |= (1 << PCIE0) | (1 << PCIE2);
  //PCICR |= (1 << PCIE2) ;
}


void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);

  // initialize the encoders
  SetupEncoders();

  // attach servo to pin 51,11
  left.attach(51);
  right.attach(43);

  // Arduino node
  arduino_nh.initNode() ;
```

```
  //broadcaster.init(arduino_nh) ; //added

  arduino_nh.getHardware()->setBaud(115200);
  arduino_nh.advertise(pub); // setting up subscriptions
  arduino_nh.subscribe(sub); // setting up publications

}

void loop() {

  if (millis() - Time > sample_time)
    {
      Time = millis() ;

      // Update Motors with corresponding speed and send speed values
      //through serial port

     Update_Motors(vd, wd);
     publish_data();
     arduino_nh.spinOnce();

    }

}

// UPDATE MOTORS
void Update_Motors(double vd, double wd)
{

  //Prefilter
  wrf = ( (td*h)*wdr + (td*h)*wdr_p - (td*h - 2)*wrf_p )/(2 + td*h);
  wlf = ( (td*h)*wdl + (td*h)*wdl_p - (td*h - 2)*wlf_p )/(2 + td*h);
  wrf_p = wrf;
  wlf_p = wlf;
  wdr_p = wdr;
  wdl_p = wdl;


  // Encoder counts
  Lcount = left_ticks ;
  Rcount = right_ticks ;
  LdVal = (double) -(Rcount - Rcount_last)/(td) ; // Counts per second
  // simple interchagne to match notation
  RdVal = (double) (Lcount - Lcount_last)/(td) ; // Counts per second
  // simple interchagne to match notation
  Lcount_last = Lcount;
  Rcount_last = Rcount;

  // Present angular velocities
  wL = (LdVal/CPR)*(2*3.14159) ; // rads/sec
  wR = (RdVal/CPR)*(2*3.14159) ; // rads/sec

  wLn = (wL + wLp)/2.0;  // avg with previous values to make it even
  // smoother
  wRn = (wR + wRp)/2.0;

```

```
  wLp = wL; // saving present angular velocities to be used in the next
  // loop
  wRp  = wR; // saving present angular velocities to be used in the next
  //  loop

  Rerror = wdr - wRn ; // error (ref - present) pre-fileter - should be
  //  added to prevent overshoot, reduces the jerk
  Lerror = wdl - wLn ; // error (ref - present)

  // Inner loop controller PID

CL = CL_p + 0.259*Lerror - 0.2166*Lerror_p;

CR = CR_p + 0.259*Rerror - 0.2166*Rerror_p;


  CL_pppp = CL_ppp;
  CL_ppp = CL_pp;
  CL_pp = CL_p;

  CR_pppp = CR_ppp;
  CR_ppp = CR_pp;
  CR_pp = CR_p;

  Lerror_ppp = Lerror_pp;
  Lerror_pp = Lerror_p;
  Lerror_p = Lerror;
  Rerror_ppp = Rerror_pp;
  Rerror_pp = Rerror_p;
  Rerror_p = Rerror;

  if (CL < 0) CL = 0;
  if (CL > 100) CL = 100;
  if (CR < 0) CR = 0;
  if (CR > 100) CR = 100;

  CL_p = CL;
  CR_p = CR;

  CL = CL + 1570;
  CR = CR + 1570;
  left.writeMicroseconds(CL*g);
  right.writeMicroseconds(CR*g);


}

void publish_data(){

  rpm_msg.linear.x = wRn;//rigt_angularVelocity;
  rpm_msg.linear.y = wLn;//right_angularVelocity;
  rpm_msg.linear.z =  Time;
  rpm_msg.angular.x = CL;
  rpm_msg.angular.y = CR;
  rpm_msg.angular.z = g;
  pub.publish(&rpm_msg);
  //Serial.println(Time);
```

```

}
```

```

// Description: ROS Node to send Pose data from ground station
// (HTC Vive) to NVIDIA TX2s
#include "ros/ros.h"
#include "std_msgs/String.h"
#include "std_msgs/Int8.h"
#include <tf/tf.h>
#include<geometry_msgs/Vector3.h>
#include<geometry_msgs/Vector3Stamped.h>
#include<geometry_msgs/Twist.h>
#include<geometry_msgs/Point.h>
#include<geometry_msgs/PoseWithCovarianceStamped.h>
#include<sensor_msgs/Joy.h>
#include <sstream>
#include <iostream>
#include <fstream>


class readData{
    public:
     readData();
    private:
     ros::NodeHandle n;
     ros::Publisher pub;
     ros::Subscriber sub;
     ros::Publisher pub2;
     ros::Subscriber sub2;
     void callBack(const geometry_msgs::PoseWithCovarianceStamped::
     ConstPtr& msg);
     void callBack2(const geometry_msgs::PoseWithCovarianceStamped::
     ConstPtr& msg);
     geometry_msgs::Point tracker1;
         geometry_msgs::Point tracker2;
};

    readData::readData(){
    sub = n.subscribe<geometry_msgs::PoseWithCovarianceStamped>
    ("/vive/LHR_D254C151_pose", 1000, &readData::callBack,this);
    sub2 = n.subscribe<geometry_msgs::PoseWithCovarianceStamped>
    ("/vive/LHR_90C2F95A_pose", 1000, &readData::callBack2,this);
    pub = n.advertise<geometry_msgs::Point>("tracker_1",1000);
    pub2 = n.advertise<geometry_msgs::Point>("tracker_2",1000);
    }

    void readData::callBack(const geometry_msgs::
    PoseWithCovarianceStamped::ConstPtr& msg){tf::Quaternion q1(
            msg->pose.pose.orientation.x,
            msg->pose.pose.orientation.y,
            msg->pose.pose.orientation.z,
            msg->pose.pose.orientation.w);
     tf::Quaternion q2(0.707, 0.000, 0.000, 0.707);
     tf::Matrix3x3 m(q2*q1);
     double roll, pitch, yaw;
```

```
      static double yaw_prev = 0, relYaw = 0;
          m.getRPY(roll, pitch, yaw);
          // converts the Quaternion to Euler angles
      if ((std::abs(yaw - yaw_prev)) > 3.141) {
         if (yaw > yaw_prev) relYaw = relYaw - (2*3.141 -
         std::abs(yaw - yaw_prev));
         else relYaw = relYaw + (2*3.141 - std::abs(yaw - yaw_prev));
         //if (yaw < yaw_prev) relYaw = relYaw +
         //(2*3.141 - std::abs(yaw - yaw_prev));
      }
      else {
         relYaw = relYaw + (yaw - yaw_prev);
      }

      yaw_prev = yaw;
      tracker1.x = -(msg->pose.pose.position.x);
      tracker1.y = msg->pose.pose.position.z;
      tracker1.z = relYaw;
      pub.publish(tracker1);
}
     void readData::callBack2(const geometry_msgs::
     PoseWithCovarianceStamped::ConstPtr& msg){tf::Quaternion q1(
             msg->pose.pose.orientation.x,
             msg->pose.pose.orientation.y,
             msg->pose.pose.orientation.z,
             msg->pose.pose.orientation.w);
      tf::Quaternion q2(0.707, 0.000, 0.000, 0.707);
      tf::Matrix3x3 m(q2*q1);
      double roll, pitch, yaw;
          m.getRPY(roll, pitch, yaw);
      tracker2.x = -(msg->pose.pose.position.x);
      tracker2.y = msg->pose.pose.position.z;
      tracker2.z = yaw;
      pub2.publish(tracker2);
}
      //return 0;


int main(int argc, char **argv)
{
 ros::init(argc, argv, "vive_data_send");

 //TeleopJoy teleop_turtle;
 readData dude;

 ros::spin();

 return 0;
}
```

```

//Description: ROS Node for Cruise Control
//Fri 22 May 2020 12:17:46 AM MST
#include "ros/ros.h"
#include "ros/time.h"
#include "std_msgs/Int8.h"
```

```cpp
#include "std_msgs/String.h"
#include "std_msgs/Float64MultiArray.h"
#include <cmath>
#include <tf/tf.h>
#include<geometry_msgs/Vector3.h>
#include<geometry_msgs/Vector3Stamped.h>
#include<geometry_msgs/Twist.h>
#include<geometry_msgs/Point.h>
#include<geometry_msgs/PoseWithCovarianceStamped.h>
#include<sensor_msgs/Joy.h>
#include <sstream>
#include <iostream>
#include <fstream>

double Radius = 0.039;
double Length = 0.324;

class readData{
    public:
    readData();
    private:
    ros::NodeHandle n;
        ros::Publisher pub;  // publish to cmd vel: wr, wl
    ros::Publisher pub2; // publish the experiment simulation data: x,y,
    // theta,v,w,Vref,theta_ref
        ros::Subscriber sub; // subscribe to keyboard
    ros::Subscriber sub2; // subscribe to tracker_1: x,y,theta
        void callBack(const geometry_msgs::Twist::ConstPtr& msg);
        // subscribe to the keyboard
    void callBack2(const geometry_msgs::Point::ConstPtr& msg);
    // subscribe to the tracker_1
    geometry_msgs::Twist vel; std_msgs::Float64MultiArray expData;
        double wr; double wl; // for now the values are going to be in
        // micro seconds to test the rpm of the motor and log the data
    int cruise = 0; int initial = 0;
    double x_i; double y_i; double theta_i; double x_f; double y_f;
    double theta_f;
    double v_ref; double w_ref; double theta_ref; double theta_err;
    double k_theta = 5.0;
    std::string line; std::string sV_ref; std::string sTheta_ref;
    std::ifstream ifile
    {"/home/smanne1/catkin_ws/src/highBW/src/Cruise_05r.csv"};
};

readData::readData(){
    sub2 = n.subscribe<geometry_msgs::Point>("/tracker_1", 1,
    &readData::callBack2,this);
    sub = n.subscribe<geometry_msgs::Twist>("/keyboard",10,
    &readData::callBack,this);
    pub = n.advertise<geometry_msgs::Twist>("cmd_vel",1);
    pub2 = n.advertise<std_msgs::Float64MultiArray>("exp_data",10000);
}

void readData::callBack(const geometry_msgs::Twist::ConstPtr& msg){
    if (msg->linear.x == 2) cruise = 1;
    else {
        cruise = 0;
```

```
        initial = 0;  // resets the initial to record the initial values
        // of x,y,theta
    }
}

void readData::callBack2(const geometry_msgs::Point::ConstPtr& msg){
    if (cruise == 1){
        if (initial == 0){
            x_i = msg->x;
            y_i = msg->y;
            theta_i = msg->z;
            initial = 1;
        }
        x_f = (msg->x - x_i)*cos(theta_i) + (msg->y - y_i)*sin(theta_i)
        + 0.0;
        // add the starting value of the robot instead of 500
        y_f = -(msg->x - x_i)*sin(theta_i) + (msg->y - y_i)*cos(theta_i)
        + 0.0;
        theta_f = msg->z - theta_i;

        if (std::getline(ifile, line)) { // read the current line
            std::istringstream iss{line}; // construct a string stream
            // from line
            std::getline(iss, sTheta_ref, ',');
            std::getline(iss, sV_ref,',');

            //ROS_INFO("%s\n", sV_ref.c_str());
            v_ref = std::stod(sV_ref);
            theta_ref = std::stod(sTheta_ref);
        }

        // outerloop code
        theta_err = theta_ref - theta_f;
        w_ref = k_theta * theta_err;

        wr = (2*v_ref + Length*w_ref)/(2*Radius);
            wl = (2*v_ref - Length*w_ref)/(2*Radius);

        vel.linear.x = wr;
        vel.angular.x = wl;
        vel.linear.z = 1;
        pub.publish(vel);   // cmd_vel to the inner loop

        expData.data = { x_f, y_f, theta_f, v_ref, theta_ref};
        pub2.publish(expData);

    }
    else {
        wr = 0.0;
        wl = 0.0;
        vel.linear.z = 0;
        vel.linear.x = wr;
        vel.angular.x = wl;
        pub.publish(vel);   // cmd_vel to the inner loop

    }

```

```cpp

}

int main(int argc, char **argv){
    ros::init(argc, argv, "Cruise");

    //TeleopJoy teleop_turtle;
    readData dude;

    ros::spin();

    return 0;
}
```

```cpp

//Description: ROS Node for writing data into a .csv file
#include "ros/ros.h"
#include "ros/time.h"
#include "std_msgs/String.h"
#include "std_msgs/Int8.h"
#include "std_msgs/Float64.h"
#include "std_msgs/Float64MultiArray.h"
#include <cmath>
#include <tf/tf.h>
#include<geometry_msgs/Vector3.h>
#include<geometry_msgs/Vector3Stamped.h>
#include<geometry_msgs/Twist.h>
#include<geometry_msgs/Point.h>
#include<geometry_msgs/PoseWithCovarianceStamped.h>
#include<sensor_msgs/Joy.h>
#include <sstream>
#include <iostream>
#include <fstream>

double Radius = 0.039; // Change it (radius of wheel) 0.045
double Length = 0.324; // Change it (distance between the wheels (dw)
int emergency = 0;

class emergencyStop{       // stop the robot if any axes of the
//joystick is moved
        public:
     emergencyStop();
    private:
     ros::NodeHandle n;
     ros::Subscriber sub;
     void callBack(const sensor_msgs::Joy::ConstPtr& joy);

};
    emergencyStop::emergencyStop(){
     sub = n.subscribe<sensor_msgs::Joy>("joy", 10, &emergencyStop::
     callBack,this);
    }

    void emergencyStop::callBack(const sensor_msgs::Joy::ConstPtr& joy){
     if (joy->axes[1] + joy->axes[2] + joy->axes[3] != 0)
      emergency = 1;
```

```
    }

class readData{
    public:
     readData();
    private:        // emergency stop feature programmed in the robot3
    // module incase of high current
     void callBack(const geometry_msgs::Twist::ConstPtr& msg);
     void callBack2(const std_msgs::Float64MultiArray::ConstPtr& msg);
     void dataWrite(const geometry_msgs::Twist::ConstPtr& msg);
     geometry_msgs::Twist vel;
     std::string filename =
     "/home/smanne1/catkin_ws/src/highBW/matlab/robot1/arduino.csv";
     std::string filename2 =
     "/home/smanne1/catkin_ws/src/highBW/matlab/robot1/cruiseData.csv";
     int i; double vdf; double wdf; double wdr; double wdl; double Rwdr;
     double Rwdl;
     ros::NodeHandle n;
     ros::Subscriber sub;
     ros::Subscriber sub2;

};

    readData::readData(){
    sub = n.subscribe<geometry_msgs::Twist>("arduino_vel", 10,
    &readData::callBack,this);
        sub2 = n.subscribe<std_msgs::Float64MultiArray>(
        "exp_dataRecord", 10000,&readData::callBack2,this);
    i = 0;
    }

void readData::callBack(const geometry_msgs::Twist::ConstPtr& msg){
    dataWrite(msg);
}


void readData::dataWrite(const geometry_msgs::Twist::ConstPtr& msg){
     //vdf = msg->linear.y;
     //wdf = msg->angular.y;
     //wdr = (2*vdf + Length*wdf)/(2*Radius);    // actual angular
     //velocities
     //wdl = (2*vdf - Length*wdf)/(2*Radius);

     //Rwdr = (2*(msg->linear.x) + Length*(msg->angular.x))/(2*Radius);
     // reference angular velocities
     //Rwdl = (2*(msg->linear.x) - Length*(msg->angular.x))/(2*Radius);

     vdf = (msg->linear.x + msg->linear.y)*Radius/2;
     wdf = (msg->linear.x - msg->linear.y)*Radius/Length;

     std::ofstream myfile;
         ROS_INFO("printing data");
     myfile.open(filename.c_str(), std::ios::app);
         myfile << " Right_Angular_Vel " << msg->linear.x <<
         " Left_Angular_Vel " << msg->linear.y;
         myfile << " Time " << msg->linear.z << " Ref_Right " <<
         msg->angular.x;
```

```
         myfile << " Ref_Left " << msg->angular.y << "\n";
//         myfile << " Position_x " << msg->linear.z << " Position_y "
//<< msg->angular.z << "\n";
     myfile.close();
     //return 0;
}

void readData::callBack2(const std_msgs::Float64MultiArray::
    ConstPtr& msg){
     std::ofstream myfile2;
         ROS_INFO("printing data");
     myfile2.open(filename2.c_str(), std::ios::app);
         myfile2 << " Position_x " << msg->data[0] << " Position_y "
         << msg->data[1];
         myfile2 << " Theta " << msg->data[2] << " Linear_Velocity "
         << msg->data[3];
         myfile2 << " Angular_Velocity " << msg->data[4] << " Time "
         << msg->data[5];
         myfile2 << " X_ref " << msg->data[6] << " Y_ref " <<
         msg->data[7] << "\n";
//         myfile << " Position_x " << msg->linear.z << " Position_y "
//<< msg->angular.z << "\n";
     myfile2.close();
}


int main(int argc, char **argv)
{
 ros::init(argc, argv, "ground_station_data_receive_Vive2");

 //emergencyStop delta;
 readData dude;

 ros::spin();

 return 0;
}
```

```

// Description: ROS Node for generic outer-loop
//Fri 22 May 2020 12:17:46 AM MST
#include "ros/ros.h"
#include "ros/time.h"
#include "std_msgs/Int8.h"
#include "std_msgs/String.h"
#include "std_msgs/Float64MultiArray.h"
#include <cmath>
#include <tf/tf.h>
#include<geometry_msgs/Vector3.h>
#include<geometry_msgs/Vector3Stamped.h>
#include<geometry_msgs/Twist.h>
#include<geometry_msgs/Point.h>
#include<geometry_msgs/PoseWithCovarianceStamped.h>
#include<sensor_msgs/Joy.h>
#include <sstream>
#include <iostream>
```

```
#include <fstream>

class readData{
    public:
    readData();
    private:
    ros::NodeHandle n;
        ros::Publisher pub;  // publish to cmd vel: wr, wl
    ros::Publisher pub2; // publish the experiment simulation data: x,y,
    // theta,v,w,Vref,theta_ref
        ros::Subscriber sub; // subscribe to keyboard
    ros::Subscriber sub2; // subscribe to tracker_1: x,y,theta
        void callBack(const geometry_msgs::Twist::ConstPtr& msg);
        // subscribe to the keyboard
    void callBack2(const geometry_msgs::Point::ConstPtr& msg);
    // subscribe to the tracker_1
    geometry_msgs::Twist vel; std_msgs::Float64MultiArray expData;
        double wr; double wl; // for now the values are going to be
        //in micro seconds to test the rpm of the motor and log the data
    int cruise = 0; int initial = 0;
    double x_i; double y_i; double theta_i; double x_f; double y_f;
    double theta_f;
    double v_ref; double theta_ref; std::string line; std::
    string sV_ref;
    std::string sTheta_ref;
    std::ifstream ifile {
        "/home/smanne1/catkin_ws/src/highBW/src/Cruise.csv"};
};

readData::readData(){
    sub2 = n.subscribe<geometry_msgs::Point>("/tracker_1", 1,
    &readData::callBack2,this);
    sub = n.subscribe<geometry_msgs::Twist>("/keyboard",10,
    &readData::callBack,this);
    pub = n.advertise<geometry_msgs::Twist>("cmd_vel",1);
    pub2 = n.advertise<std_msgs::Float64MultiArray>("exp_data",10000);
}

void readData::callBack(const geometry_msgs::Twist::ConstPtr& msg){
    if (msg->linear.x == 2) cruise = 1;
    else {
        cruise = 0;
        initial = 0;  // resets the initial to record the initial values
        // of x,y,theta
    }
    vel.linear.x = msg->linear.x*(51/2);
    vel.angular.x = msg->angular.z;
    if (vel.linear.x < 0) vel.linear.x = 0;
    if (vel.angular.x < 0) vel.angular.x = 0;
    pub.publish(vel);
}

void readData::callBack2(const geometry_msgs::Point::ConstPtr& msg){
    if (cruise == 1){
        if (initial == 0){
            x_i = msg->x;
            y_i = msg->y;
```

```
            theta_i = msg->z;
            initial = 1;
        }
        x_f = (msg->x - x_i)*cos(theta_i) + (msg->y - y_i)*sin(theta_i)
        + 0.0;
        // add the starting value of the robot instead of 500
        y_f = -(msg->x - x_i)*sin(theta_i) + (msg->y - y_i)*cos(theta_i)
        + 0.0;
        theta_f = msg->z - theta_i;



        expData.data = { x_f, y_f, theta_f, v_ref, theta_ref};
        pub2.publish(expData);
    }
}

int main(int argc, char **argv){
    ros::init(argc, argv, "tesla1");

    //TeleopJoy teleop_turtle;
    readData dude;
    ros::spin();

    return 0;
}
```

```
//Description: ROS Node for Planar Cartesian Stabilization
//Fri 22 May 2020 12:17:46 AM MST
#include "ros/ros.h"
#include "ros/time.h"
#include "std_msgs/Int8.h"
#include "std_msgs/String.h"
#include "std_msgs/Float64MultiArray.h"
#include <cmath>
#include <tf/tf.h>
#include<geometry_msgs/Vector3.h>
#include<geometry_msgs/Vector3Stamped.h>
#include<geometry_msgs/Twist.h>
#include<geometry_msgs/Point.h>
#include<geometry_msgs/PoseWithCovarianceStamped.h>
#include<sensor_msgs/Joy.h>
#include <sstream>
#include <iostream>
#include <fstream>

double Radius = 0.039;
double Length = 0.324;

class readData{
    public:
    readData();
    private:
    ros::NodeHandle n;
        ros::Publisher pub;  // publish to cmd vel: wr, wl
    ros::Publisher pub2; // publish the experiment simulation data: x,y,
```

```
    // theta,v,w,Vref,theta_ref
        ros::Subscriber sub; // subscribe to keyboard
    ros::Subscriber sub2; // subscribe to tracker_1: x,y,theta
    void callBack(const geometry_msgs::Twist::ConstPtr& msg);
    // subscribe to the keyboard
    void callBack2(const geometry_msgs::Point::ConstPtr& msg);
    // subscribe to the tracker_1
    double fcn(double xp,double yp);  // to compute the theta_err
    // (refer matlab Cartesian code)
    geometry_msgs::Twist vel; std_msgs::Float64MultiArray expData;
        double wr; double wl; // for now the values are going to be in
    // micro seconds to test the rpm of the motor and log the data
    int cruise = 0; int initial = 0;
    double x_i; double y_i; double theta_i; double x_f; double y_f;
    double theta_f;
    double v_ref; double w_ref; double theta_ref; double x_ref;
    double y_ref;
    double d_err; double theta_err; double k_theta = 6.0;
    double k_v = 1.0;
    double xp; double yp; // used by function 'fcn'
        std::string line; std::string sX_ref; std::string sY_ref;
    std::ifstream ifile
    {"/home/smanne1/catkin_ws/src/highBW/src/Cartesian_BW.csv"};
};

readData::readData(){
    sub2 = n.subscribe<geometry_msgs::Point>("/tracker_1", 1,
    &readData::callBack2,this);
    sub = n.subscribe<geometry_msgs::Twist>("/keyboard",10,
    &readData::callBack,this);
    pub = n.advertise<geometry_msgs::Twist>("cmd_vel",1);
    pub2 = n.advertise<std_msgs::Float64MultiArray>("exp_data",10000);
}

void readData::callBack(const geometry_msgs::Twist::ConstPtr& msg){
    if (msg->linear.x == 2) cruise = 1;
    else {
        cruise = 0;
        initial = 0;  // resets the initial to record the initial values
    // of x,y,theta
    }
}

void readData::callBack2(const geometry_msgs::Point::ConstPtr& msg){
    if (cruise == 1){
        if (initial == 0){
            x_i = msg->x;
            y_i = msg->y;
            theta_i = msg->z;
            initial = 1;
        }
        x_f = (msg->x - x_i)*cos(theta_i) + (msg->y - y_i)*sin(theta_i)
        + 500.0;
        // add the starting value of the robot instead of 500
        y_f = -(msg->x - x_i)*sin(theta_i) + (msg->y - y_i)*cos(theta_i)
        + 501.5;
        theta_f = msg->z - theta_i;
```

```
        if (std::getline(ifile, line)) { // read the current line
            std::istringstream iss{line}; // construct a string stream
            // from line
            std::getline(iss, sX_ref, ',');
            std::getline(iss, sY_ref,',');

            //ROS_INFO("%s\n", sV_ref.c_str());
            x_ref = std::stod(sX_ref);
            y_ref = std::stod(sY_ref);
        }

        // outerloop code
        xp = x_ref - x_f;
        yp = y_ref - y_f;

        theta_ref = fcn(xp, yp);
        theta_err = theta_ref - theta_f;

        v_ref = sqrt(pow(xp,2) + pow(yp,2))*cos(theta_err)*k_v;
        w_ref = k_theta * theta_err;

        wr = (2*v_ref + Length*w_ref)/(2*Radius);
            wl = (2*v_ref - Length*w_ref)/(2*Radius);

        vel.linear.x = wr;
        vel.angular.x = wl;
        vel.linear.z = 1;
        pub.publish(vel);   // cmd_vel to the inner loop

        expData.data = { x_f, y_f, theta_f, x_ref, y_ref};
        pub2.publish(expData);

    }
    else {
        wr = 0.0;
        wl = 0.0;
        vel.linear.z = 0;
        vel.linear.x = wr;
        vel.angular.x = wl;
        pub.publish(vel);   // cmd_vel to the inner loop

    }


}

double readData::fcn(double xp, double yp){
    static double xo = 1.0, yo = 0.0, theta = 0.0;
    double a1,b1,c1,dtheta,sign;

// calculate the value of theta
    a1 = sqrt( pow(xp,2) + pow(yp,2));
    b1 = sqrt( pow(xo,2) + pow(yo,2));
    c1 = sqrt( pow(xp - xo, 2) + pow(yp - yo ,2));

    if ((a1 != 0) && (b1 != 0))
```

```
        dtheta = acos(std::max(std::min( (pow(a1,2) + pow(b1,2) -
        pow(c1,2))/(2*a1*b1) ,1.0),-1.0));
    else
        dtheta = 0.0;

// calculate the direction of rotation
    sign = 1.0;
    if (dtheta != 0){
        sign = (-yo/xo)*xp + yp;
        if (sign != 0){
            if ((xo > 0) && (yo > 0))
                sign = sign/std::abs(sign);
            else if ((xo < 0) && (yo < 0))
                        sign = -sign/std::abs(sign);
                else if ((xo > 0) && (yo < 0))
                        sign = sign/std::abs(sign);
                else if ((xo < 0) && (yo > 0))
                        sign = -sign/std::abs(sign);
                else if ((xo == 0) && (yo >= 0)){
                        if (xp != 0)
                            sign = -xp/std::abs(xp);
                        else
                            sign = 0;
                        }
                else if ((xo == 0) && (yo <= 0)){
                        if (xp != 0)
                            sign = xp/std::abs(xp);
                        else
                            sign = 0;
                        }
                else if ((xo >= 0)&&(yo == 0)){
                        if (yp != 0)
                            sign = yp/std::abs(yp);
                        else
                            sign = 0;
                        }
                else if ((xo <= 0) && (yo == 0)){
                        if (yp != 0)
                            sign = -yp/std::abs(yp);
                        else
                            sign = 0;
                    }
            }
        }

    xo = xp; yo = yp;
    theta = theta + dtheta*sign;
    return theta;

}


int main(int argc, char **argv){
    ros::init(argc, argv, "Cartesian");

    //TeleopJoy teleop_turtle;
    readData dude;
```

```cpp

    ros::spin();

    return 0;
}
```

```cpp

//Description: ROS Node for calculating (v,w) from pose values
//(x,y,theta)
#include "ros/ros.h"
#include "ros/time.h"
#include "std_msgs/String.h"
#include "std_msgs/Int8.h"
#include "std_msgs/Float64MultiArray.h"
#include <cmath>
#include <tf/tf.h>
#include<geometry_msgs/Vector3.h>
#include<geometry_msgs/Vector3Stamped.h>
#include<geometry_msgs/Twist.h>
#include<geometry_msgs/Point.h>
#include<geometry_msgs/PoseWithCovarianceStamped.h>
#include<sensor_msgs/Joy.h>
#include <sstream>
#include <iostream>
#include <fstream>

int buffer_length = 50;
std::deque<double> filterbuffer_v(buffer_length,0.0);
std::deque<double> filterbuffer_w(buffer_length,0.0);

class readData{
    public:
     readData();
    private:
     ros::NodeHandle n;
     ros::Publisher pub;
     ros::Subscriber sub;
     //ros::Subscriber sub2;
     void callBack(const std_msgs::Float64MultiArray::ConstPtr& msg);
     void callBack2(const geometry_msgs::Twist::ConstPtr& key);
     geometry_msgs::Twist vel;
     geometry_msgs::Point pre_msg;
     std_msgs::Float64MultiArray exp_dataRecord;
     double time; double expTime = 0.0;
     double ts = 1.0/105.0;
     double pre_time2 = 0.0;
     double pre_time = 0.0;
     double inA; double inB;
     double yaw = 0.0;  // this is absolute yaw angle
};


// using Joy will cause serious problems -- when joy is not publishing on
// to the topic
readData::readData(){
    sub = n.subscribe<std_msgs::Float64MultiArray>("exp_data", 10000,
```

```
    &readData::callBack,this);
    pub = n.advertise<std_msgs::Float64MultiArray>("exp_dataRecord",
    10000);
    //sub2 = n.subscribe<geometry_msgs::Twist>("/keyboard",10,
    //&readData::callBack,this);
    pre_msg.x = 0.0; pre_msg.y = 0.0; pre_msg.z = 0.0;
    }

void readData::callBack(const std_msgs::Float64MultiArray::
ConstPtr& msg){

        time = ros::Time::now().toSec();
        vel.linear.y =
        sqrt(pow(((msg->data[0] - pre_msg.x)/(time - pre_time)),2) +
        pow(((msg->data[1] - pre_msg.y)/(time - pre_time)),2));
        // the next four conditions are not ncessary, they are covered
        // by the last 4, all the if conditions are not necessary as
        // well, by default the vel.linear.y is always positive

        // convert the theta from relative to absolute
        if (!(((msg->data[2]) > -3.141) && (msg->data[2] < 3.141))){
        // it theta is out of -pi to pi range enter the loop
            yaw = msg->data[2];
            if (msg->data[2] > 0){
                while(!((yaw > -3.141) && (yaw < 3.141)))
                    yaw = yaw - (2*3.141);
            }
            if (msg->data[2] < 0){
                while(!((yaw > -3.141) && (yaw < 3.141)))
                    yaw = yaw + (2*3.141);
            }
        }
        else yaw = msg->data[2];


        if ( ((msg->data[0] - pre_msg.x)>0) && ((msg->data[1] -
        pre_msg.y)==0)){
            if (std::abs(yaw) > (3.141/2))
                vel.linear.y = vel.linear.y;
            else
                vel.linear.y = -vel.linear.y;
        }
        else if (((msg->data[0] - pre_msg.x)==0)&&((msg->data[1] -
        pre_msg.y)<0))
        {
            if (yaw > (0))
                vel.linear.y = vel.linear.y;
            else
                vel.linear.y = -vel.linear.y;
        }
        else if (((msg->data[0] - pre_msg.x)<0)&&((msg->data[1] -
        pre_msg.y)==0))
        {
            if (std::abs(yaw)<(3.141/2))
                vel.linear.y = vel.linear.y;
            else
                vel.linear.y = -vel.linear.y;
```

```
        }
        else if (((msg->data[0] - pre_msg.x)==0)&&((msg->data[1] -
        pre_msg.y)>0))
        {
            if (yaw < (0))
                vel.linear.y = vel.linear.y;
            else
                vel.linear.y = -vel.linear.y;
        }
        else if (((msg->data[0] - pre_msg.x)>0)&&((msg->data[1] -
        pre_msg.y)>0))
        {
            if ((yaw < (-3.141/2))&&(yaw > (-3.141/1)))
                vel.linear.y = vel.linear.y;
            else
                vel.linear.y = -vel.linear.y;
        }
        else if (((msg->data[0] - pre_msg.x)>0)&&((msg->data[1] -
        pre_msg.y)<0))
        {
            if ((yaw > (3.141/2))&&(yaw < (3.141/1)))
                vel.linear.y = vel.linear.y;
            else
                vel.linear.y = -vel.linear.y;
        }
        else if (((msg->data[0] - pre_msg.x)<0)&&((msg->data[1] -
        pre_msg.y)<0))
        {
            if ((yaw > (0))&&(yaw < (3.141/2)))
                vel.linear.y = vel.linear.y;
            else
                vel.linear.y = -vel.linear.y;
        }
        else if (((msg->data[0] - pre_msg.x)<0)&&((msg->data[1] -
        pre_msg.y)>0))
        {
            if ((yaw < (0))&&(yaw > (-3.141/2)))
                vel.linear.y = vel.linear.y;
            else
                vel.linear.y = -vel.linear.y;
        }


        filterbuffer_v.push_front(vel.linear.y);// buffer implementation
        double sum_v = 0.0;
        for (int i = 0; i < buffer_length; i++){
        sum_v = sum_v + filterbuffer_v[i];
        }
        vel.linear.y = sum_v/(buffer_length);   // buffer end
        filterbuffer_v.pop_back();
        pre_msg.x = msg->data[0];
        pre_msg.y = msg->data[1];
        pre_time = time;
        if (std::abs(vel.linear.y) < 0.015)
        vel.linear.y = 0;
```

```
        //if ((msg->data[2]*pre_msg.z)>0)    // this conditions is
        // required when the range of theta is from -pi to pi and it's
        // absolute. Now it's relative to the starting value i.e. it
        // keeps increasing or decreasing from the starting value.
        //{
        time = ros::Time::now().toSec();
        vel.angular.y = (msg->data[2] - pre_msg.z)/(time - pre_time2);
        filterbuffer_w.push_front(vel.angular.y);
        double sum_w = 0.0;
        for (int i = 0; i < buffer_length; i++){
        sum_w = sum_w + filterbuffer_w[i];
        }
        vel.angular.y = sum_w/(buffer_length);
        filterbuffer_w.pop_back();
        pre_msg.z = msg->data[2];
        pre_time2 = time;
        //}
        //else {
        //pre_msg.z = msg->data[2];
        //}

        if (std::abs(vel.angular.y) < 0.01)
        vel.angular.y = 0;


        exp_dataRecord.data = {msg->data[0], msg->data[1], msg->data[2],
        -vel.linear.y, vel.angular.y, expTime, msg->data[3],
        msg->data[4]};
        expTime = expTime + ts;

        pub.publish(exp_dataRecord);

}

void readData::callBack2(const geometry_msgs::Twist::ConstPtr& key){

}

int main(int argc, char **argv)
{
 ros::init(argc, argv, "ground_station_innerLoop2");

 //TeleopJoy teleop_turtle;
 readData dude;

 ros::spin();

 return 0;
}
```