

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

	Page
LIST OF TABLES	vii
LIST OF FIGURES	viii
CHAPTER	
1 INTRODUCTION AND OVERVIEW OF WORK	1
1.1 Introduction and Motivation	1
1.2 Literature Survey: Ground Robotics - State of the Field	3
1.3 Contributions - Fundamental Questions Addressed	13
1.4 Organization of Thesis	22
2 OVERVIEW OF THE DIFFERENTIAL DRIVE VEHICLE PLATFORM	24
2.1 Introduction and Overview	24
2.2 Hardware Design Requirements	24
2.3 Actuator Selection	26
2.4 Detailed DDV Build Guide	31
2.5 C^4S Requirements and Software Architecture	47
3 MATHEMATICAL PRELIMINARIES	55
3.1 Overview	55
3.2 Discretization of Linear State Space Models	55
3.3 Reference Frames, Euler Angles and Quaternions	58
3.4 Linearization of Nonlinear Systems	63
4 MODELLING AND DESIGN TRADE STUDIES FOR A DIFFERENTIAL- DRIVE MOBILE ROBOT	65
4.1 Introduction and Overview	65
4.2 Modeling of a Differential-Drive Ground Robot	66
4.2.1 Differential-Drive Robot Kinematics	68

CHAPTER	Page
4.2.2	Differential-Drive Robot Dynamics 71
4.2.3	Actuator (DC Motor) Dynamics and Parameter Estimation . 72
4.2.4	TITO LTI Model with Actuator Dynamics..... 76
4.3	Design Trade Studies 78
4.3.1	TITO LTI Model - Input/Output Variations and Analysis .. 78
4.3.2	Trade Studies for $d = 0$ 83
4.3.3	Trade Studies for $d \neq 0$ 92
5	IMPACT OF DESIGN PARAMETERS ON OUTER-LOOP: SPEED AND POSITION CONTROL PERFORMANCE..... 113
5.1	Introduction and Overview 113
5.2	DDV Design Notations and Analysis 113
5.3	Inner-Loop Decentralized Control Design and Implementation..... 121
5.4	Outer-Loop Control Design and Impact of Design Parameters: Sim- ulation and Hardware Trade Studies 131
5.4.1	Outer-Loop 1: (v, θ) Cruise Control 133
5.4.2	Outer-Loop 2: Planar (x, y) Cartesian Stabilization..... 148
6	MULTI-ROBOT FORMATION CONTROL USING RECEDING HORI- ZON OPTIMIZATION..... 164
6.1	Introduction and Overview 164
6.2	Problem Formulation: Leader-Follower Approach..... 168
6.2.1	Prediction Model 169
6.2.2	Trajectory Tracking 171
6.2.3	Control Strategy 172

CHAPTER	Page
6.3 Impact of Kinematic and Dynamic Model Constraints: Simulation and Hardware Trade Studies.....	176
7 SUMMARY AND FUTURE DIRECTIONS	191
7.1 Summary of Work	191
7.2 Directions for Future Research	191
REFERENCES	194
APPENDIX	
A ADDITIONAL HARDWARE INFORMATION	201
B MATLAB CODE	203
C ROS AND ARDUINO CODE	260

LIST OF TABLES

Table	Page
2.1 Miscellaneous Parts	40
3.1 Discretization Methods	57
4.1 DDV Nominal Parameter Definitions	67
4.2 DDV Nominal Parameter Values	68
4.3 DC Motor Parameter Values	75
5.1 Summary of Trade Studies Conducted	132
6.1 Summary of Trade Studies Conducted	167

LIST OF FIGURES

Figure	Page
2.1 DC Motor(Models 1 - 22) Characteristics	28
2.2 DC Motor(Models 1 - 22) Comparison	29
2.3 DC Motor(Models 1 - 22) Comparison	30
2.4 Selected DC Motor(Model 3) Characteristics	31
2.5 From top, Base Stations, VR Headset, Controllers	32
2.6 Vive Tracker	32
2.7 NVIDIA Jetson TX2 Module	33
2.8 Arduino Mega 2560 Module	34
2.9 Vyper DC Motor Driver	35
2.10 Wheel Encoder	35
2.11 Energizer Battery Pack	36
2.12 Hyperion LiPo Battery	37
2.13 SPST Switch	37
2.14 Metal Castor Wheel	38
2.15 Motor Bracket	38
2.16 DDV Chassis Build Materials	39
2.17 Laser Cut Components	41
2.18 3D Printed Components	42
2.19 Chassis Assembly 1	43
2.20 Chassis Assembly 2	44
2.20 Chassis Assembly 2	45
2.21 Adjustable Castor Wheel	46
2.22 Tracker with Damper	47
2.23 DDV with Vive Tracker	48

2.24	DDV Software Architecture	49
3.1	Rotation of Translation of Coordinate Axis in Two Dimensions	59
3.2	Rotation in Three Dimensions	60
4.1	DDV Visualization	70
4.2	DC Motor Speed - Voltage Dynamical Model	72
4.3	Small Scale Chassis Dynamo-meter Setup	75
4.4	DDV Dynamics $(\tau_r, \tau_l) \rightarrow (\omega_r, \omega_l)$	76
4.5	DDV Dynamics $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$	77
4.6	Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying I	84
4.7	Singular Value Response of $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ System: Varying I	84
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	85
4.9	Singular Value Response of $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ System: Varying I	86
4.10	Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying m	87
4.11	Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying m	87
4.12	Bode Magnitude Response of $P_{[e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}] \rightarrow [v, \omega]}$ System: Varying m	88
4.13	Singular Value Response of $P_{[e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}] \rightarrow [v, \omega]}$ System: Varying m	89
4.14	Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying r	90
4.15	Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying r	90
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	91
4.17	Singular Value Response of $P_{[e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}] \rightarrow [v, \omega]}$ System: Varying r	92
4.18	Dominant Pole Variation with d	93
4.19	Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying v_{eq}	94
4.20	Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying v_{eq}	95

4.21	Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[v,\omega]}$ System: Varying v_{eq}	96
4.22	Singular Value Response of $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[v,\omega]}$ System: Varying v_{eq}	96
4.23	Bode Magnitude Response of $P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$ System: Varying ω_{eq}	97
4.24	Singular Value Response of $P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$ System: Varying ω_{eq}	98
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}	99
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}	99
4.27	Bode Magnitude Response of $P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$ System: Varying d	100
4.28	Singular Value Response of $P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$ System: Varying d	101
4.29	Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[v,\omega]}$ System: Varying d	102
4.30	Singular Value Response of $psdv$ System: Varying d	102
4.31	Bode Magnitude Response of $P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$ System: Varying I	103
4.32	Singular Value Response of $P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$ System: Varying I	104
4.33	Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[v,\omega]}$ System: Varying I	105
4.34	Singular Value Response of $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[v,\omega]}$ System: Varying I	105
4.35	Bode Magnitude Response of $P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$ System: Varying m	106
4.36	Singular Value Response of $P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$ System: Varying m	107
4.37	Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[v,\omega]}$ System: Varying m	108
4.38	Singular Value Response of $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[v,\omega]}$ System: Varying m	108
4.39	Bode Magnitude Response of $P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$ System: Varying r	109
4.40	Singular Value Response of $P_{[e_{a_r},e_{a_l}]\rightarrow[\omega_r,\omega_l]}$ System: Varying r	110
4.41	Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[v,\omega]}$ System: Varying r	111
4.42	Singular Value Response of $P_{[e_{a_r}+e_{a_l},e_{a_r}-e_{a_l}]\rightarrow[v,\omega]}$ System: Varying r	111
5.1	DDV Design Notations	114
5.2	Plant Frequency Response D1 & D2	118

Figure	Page
5.3 Plant Frequency Response D3 & D4	118
5.4 Plant Singular Values D1 & D2	120
5.5 Plant Singular Values D3 & D4	120
5.6 $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ System Inner-Loop Control Block Diagram	121
5.7 $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ System Inner-Loop Control Block Diagram	121
5.8 Inner-Loop Frequency Response T_{ry} : D1 & D2	124
5.9 Inner-Loop Frequency Response T_{ry} : D3 & D4	125
5.10 Open Loop Singular Values: D1 & D2	126
5.11 Open Loop Singular Values: D3 & D4	127
5.12 Sensitivity Singular Values: D1 & D2	128
5.13 Sensitivity Singular Values: D3 & D4	129
5.14 Complementary Sensitivity Singular Values: D1 & D2	130
5.15 Complementary Sensitivity Singular Values: D3 & D4	131
5.16 Reference Trajectory Visualization	133
5.17 $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ System Outer-Loop Cruise Control Block Diagram .	134
5.18 $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ System Outer-Loop Cruise Control Block Diagram	134
5.19 $\ \theta_e\ _\infty$ vs Reference Velocity: Simulation Results	136
5.20 $\ \theta_e\ _\infty$ vs Reference Velocity: Hardware Results	136
5.21 $\ \theta_e\ _\infty$ vs Reference Velocity: Simulation Results	137
5.22 $\ \theta_e\ _\infty$ vs Reference Velocity: Hardware Results	137
5.23 $\ v_e\ _\infty$ vs Reference Velocity: Simulation Results	138
5.24 $\ v_e\ _\infty$ vs Reference Velocity: Hardware Results	138
5.25 $\ v_e\ _\infty$ vs Reference Velocity: Simulation Results	139

Figure	Page
5.26 $\ v_e\ _\infty$ vs Reference Velocity: Hardware Results	139
5.27 Control Effort vs Reference Velocity: Simulation Results	140
5.28 Control Effort vs Reference Velocity: Simulation Results	140
5.29 $\ \theta_e\ _\infty$ vs Radius of Track: Simulation Results	142
5.30 $\ \theta_e\ _\infty$ vs Radius of Track: Hardware Results	143
5.31 $\ \theta_e\ _\infty$ vs Radius of Track: Simulation Results	143
5.32 $\ \theta_e\ _\infty$ vs Radius of Track: Hardware Results	144
5.33 $\ v_e\ _\infty$ vs Radius of Track: Simulation Results	144
5.34 $\ v_e\ _\infty$ vs Radius of Track: Hardware Results	145
5.35 $\ v_e\ _\infty$ vs Radius of Track: Simulation Results	145
5.36 $\ v_e\ _\infty$ vs Radius of Track: Hardware Results	146
5.37 Control Effort vs Radius of Track: Simulation Results	146
5.38 Control Effort vs Radius of Track: Simulation Results	147
5.39 $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ System Outer-Loop Control Block Diagram	149
5.40 $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ System Outer-Loop Control Block Diagram	149
5.41 $\ x_e\ _\infty$ vs Reference Velocity: Simulation Results	150
5.42 $\ x_e\ _\infty$ vs Reference Velocity: Hardware Results	151
5.43 $\ x_e\ _\infty$ vs Reference Velocity: Simulation Results	151
5.44 $\ x_e\ _\infty$ vs Reference Velocity: Hardware Results	152
5.45 $\ y_e\ _\infty$ vs Reference Velocity: Simulation Results	152
5.46 $\ y_e\ _\infty$ vs Reference Velocity: Hardware Results	153
5.47 $\ y_e\ _\infty$ vs Reference Velocity: Simulation Results	153
5.48 $\ y_e\ _\infty$ vs Reference Velocity: Hardware Results	154
5.49 Control Effort vs Reference Velocity: Simulation Results	154

Figure	Page
5.50 Control Effort vs Reference Velocity: Simulation Results	155
5.51 $\ x_e\ _\infty$ vs Radius of Track: Simulation Results	157
5.52 $\ x_e\ _\infty$ vs Radius of Track: Hardware Results	157
5.53 $\ x_e\ _\infty$ vs Radius of Track: Simulation Results	158
5.54 $\ x_e\ _\infty$ vs Radius of Track: Hardware Results	158
5.55 $\ y_e\ _\infty$ vs Radius of Track: Simulation Results	159
5.56 $\ y_e\ _\infty$ vs Radius of Track: Hardware Results	159
5.57 $\ y_e\ _\infty$ vs Radius of Track: Simulation Results	160
5.58 $\ y_e\ _\infty$ vs Radius of Track: Hardware Results	160
5.59 Control Effort vs Radius of Track: Simulation Results	161
5.60 Control Effort vs Radius of Track: Simulation Results	161
6.1 Reference Trajectory Visualization	168
6.2 Multi-Robot Leader-Follower Formulation	169
6.3 Closed-Loop Block Diagram Representation.....	170
6.4 Desired and Actual Pose - Error Representation	171
6.5 Formulation I and II - Block Diagram Representation.....	175
6.6 $\ x_e\ _\infty$ vs Inner-Loop Bandwidth: Simulation Results	177
6.7 $\ x_e\ _\infty$ vs Inner-Loop Bandwidth: Hardware Results	178
6.8 $\ y_e\ _\infty$ vs Inner-Loop Bandwidth: Simulation Results	178
6.9 $\ y_e\ _\infty$ vs Inner-Loop Bandwidth: Hardware Results	179
6.10 $\ \theta_e\ _\infty$ vs Inner-Loop Bandwidth: Simulation Results	179
6.11 $\ \theta_e\ _\infty$ vs Inner-Loop Bandwidth: Hardware Results	180
6.12 $\ x_e\ _\infty$ vs Reference Velocity: Simulation Results	182
6.13 $\ x_e\ _\infty$ vs Reference Velocity: Hardware Results	182

Figure	Page
6.14 $\ y_e\ _\infty$ vs Reference Velocity: Simulation Results	183
6.15 $\ y_e\ _\infty$ vs Reference Velocity: Hardware Results	183
6.16 $\ \theta_e\ _\infty$ vs Reference Velocity: Simulation Results	184
6.17 $\ \theta_e\ _\infty$ vs Reference Velocity: Hardware Results	184
6.18 $\ x_e\ _\infty$ vs Radius of Track: Simulation Results	186
6.19 $\ x_e\ _\infty$ vs Radius of Track: Hardware Results	187
6.20 $\ y_e\ _\infty$ vs Radius of Track: Simulation Results	187
6.21 $\ y_e\ _\infty$ vs Radius of Track: Hardware Results	188
6.22 $\ \theta_e\ _\infty$ vs Radius of Track: Simulation Results	188
6.23 $\ \theta_e\ _\infty$ vs Radius of Track: Hardware Results	189

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's²: 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

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

²DDV - 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 decoupled⁴ 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/-

⁴absolute 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_{ar}, e_{al})) 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 funda-

mentals. 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 $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ and $P_{[e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}] \rightarrow [v, \omega]}$ 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 trials 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 decoupled⁵ 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

⁵absolute 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 ($v_{ref} = 1\text{ m/s}$), 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

⁶for 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 $v_{ref} \geq 1$

⁷good 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 track-

ing 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-

⁸good 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

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

²Albeit 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.

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

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

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

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

Note:

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.

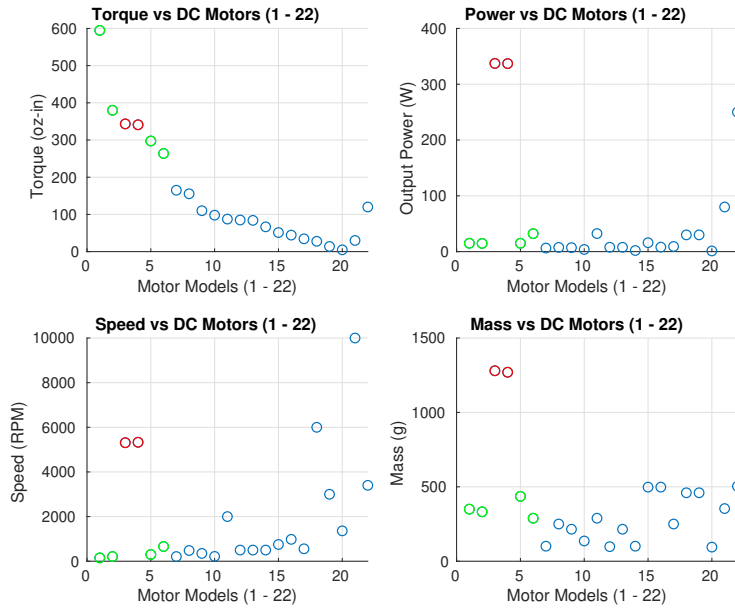


Figure 2.1: DC Motor(Models 1 - 22) Characteristics

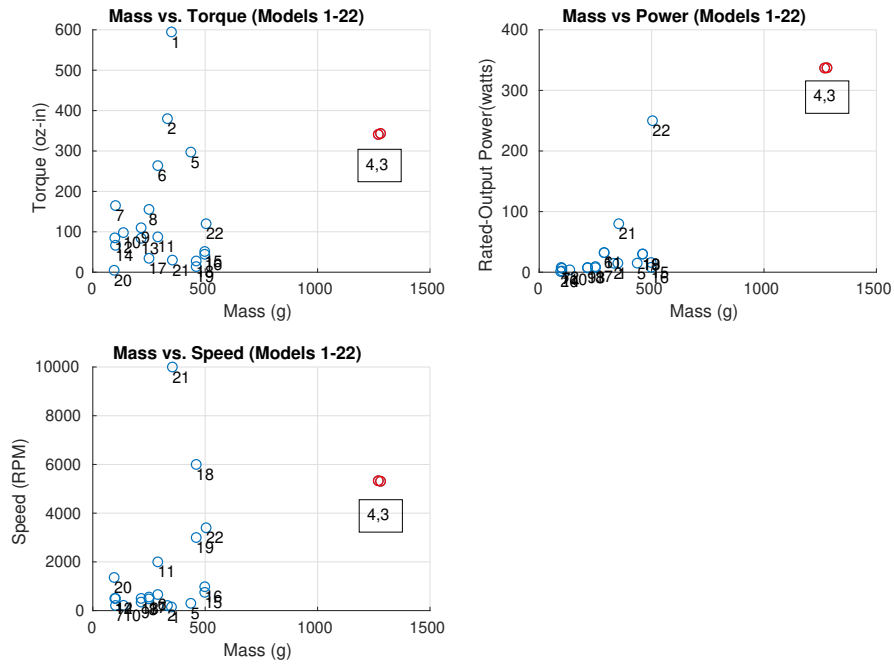


Figure 2.2: DC Motor(Models 1 - 22) Comparison

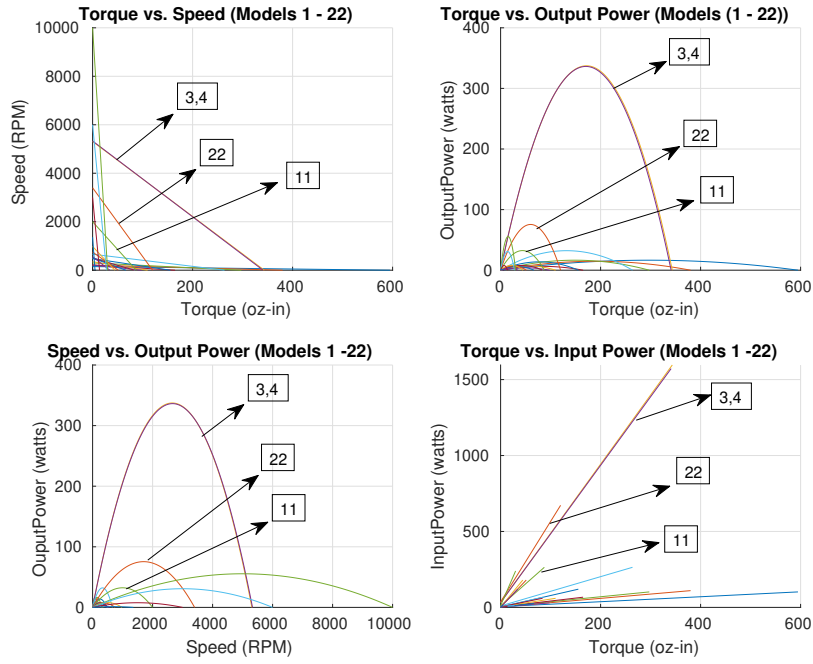


Figure 2.3: DC Motor(Models 1 - 22) Comparison

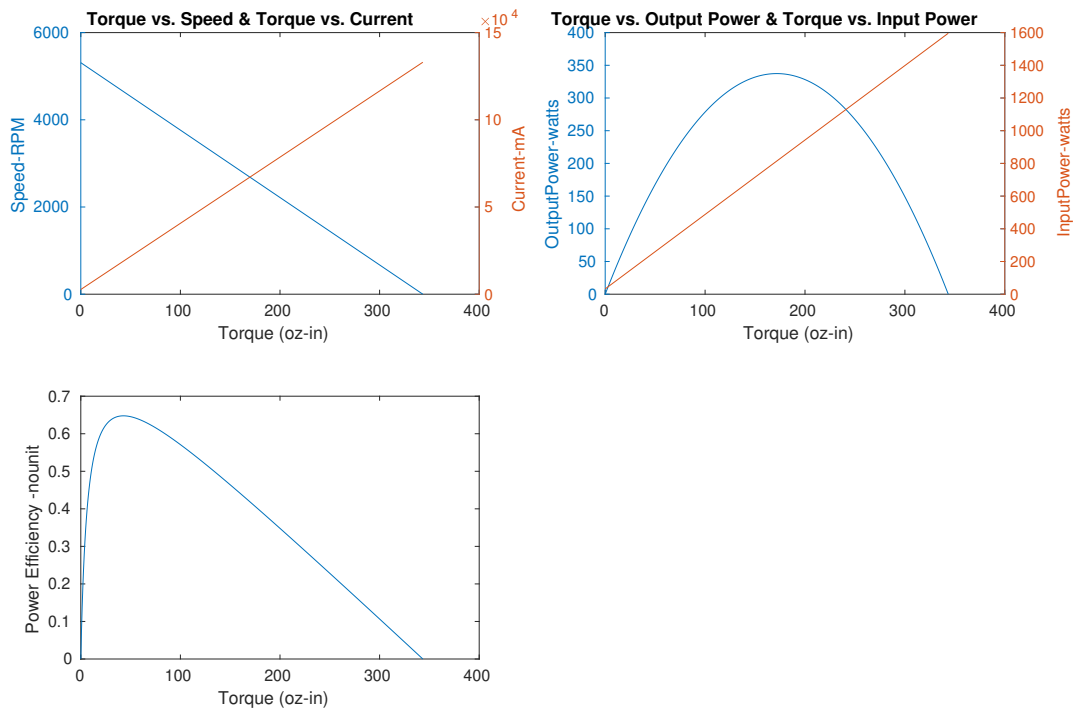


Figure 2.4: Selected DC Motor(Model 3) Characteristics

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 $5m \times 5m$, 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.

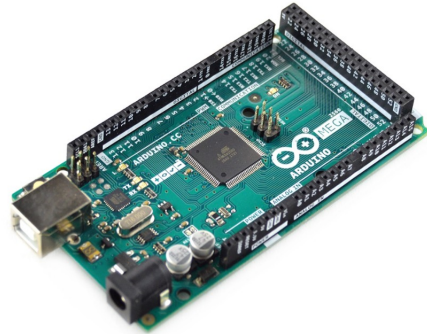


Figure 2.8: Arduino Mega 2560 Module

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 hub 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.

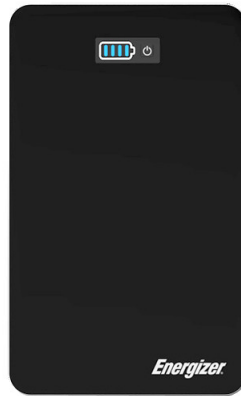


Figure 2.11: Energizer Battery Pack

7. **Hyperion LiPo Battery:** This is 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.

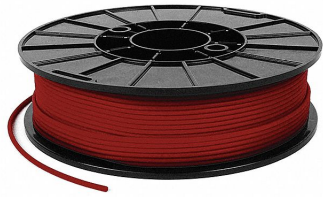


Figure 2.15: Motor Bracket

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 $62\text{cm} \times 42\text{cm} \times 0.5\text{cm}$ in order to build one DDV.



(a) PLA 3D Printer Filament



(b) Acrylic Sheets

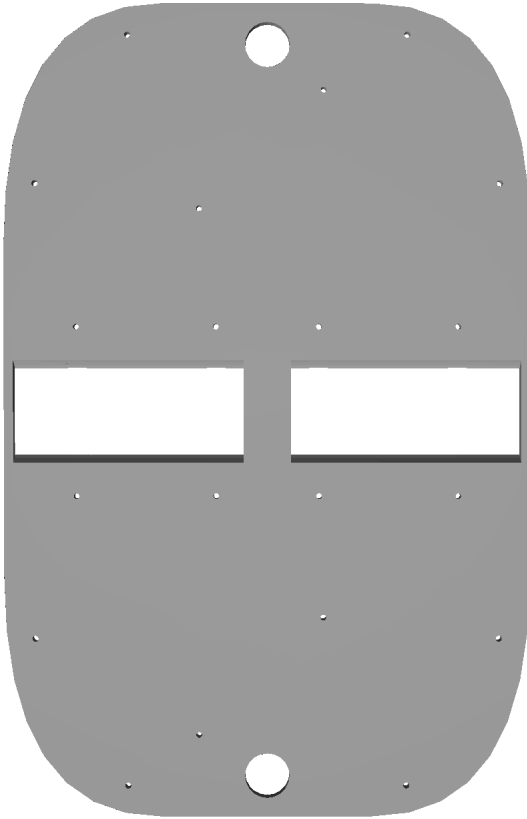
Figure 2.16: DDV Chassis Build Materials

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

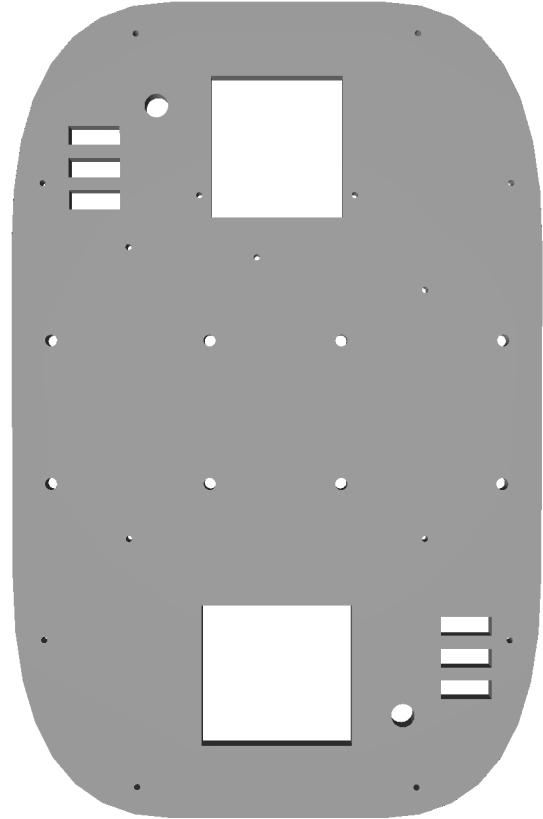
Name	Dimension	Quantity
M3 Bolt,Nut,Flat & Split Washer	30 mm	12
M3 Bolt,Nut,Flat & Split Washer	20 mm	20
M3 Bolt,Nut,Flat & Split Washer	6 mm	3
3/16" - 32 Bolt	35 mm	4
1/4" - 28 Bolt,Nut & Washer	30 mm	4
M3 Standoffs	40 mm	24
M3 Standoffs	20 mm	16
XT60 Connectors	-	2

Table 2.1: Miscellaneous Parts

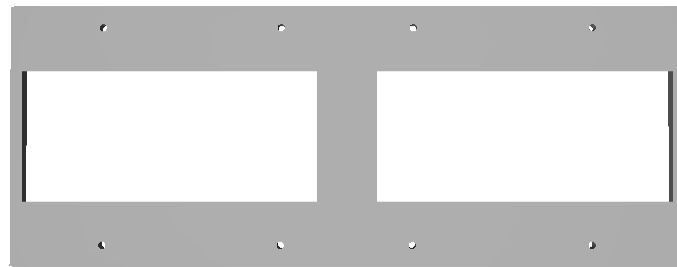
Components Assembly: Let us begin with the list of components that have to be 3D printed or laser cut out of acrylic sheets. The components shown in Figure 2.17 are made out of acrylic sheets, and the components shown in Figure 2.18 are 3D printed. The link to the .stl files of all these models is available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



(a) Base Plate(Top View)

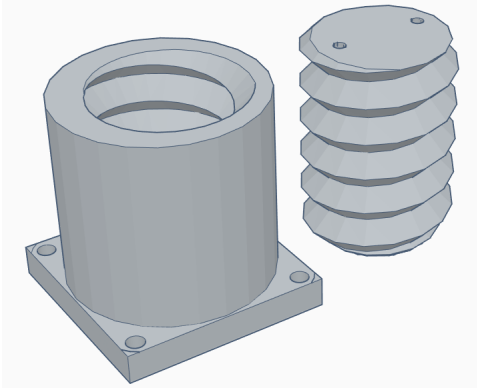


(b) Top Plate (Top View)

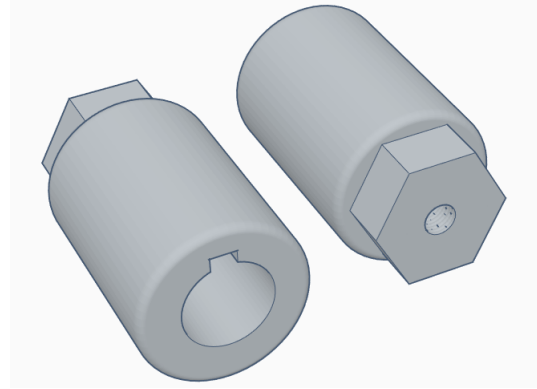


(c) Support Plate (Top View)

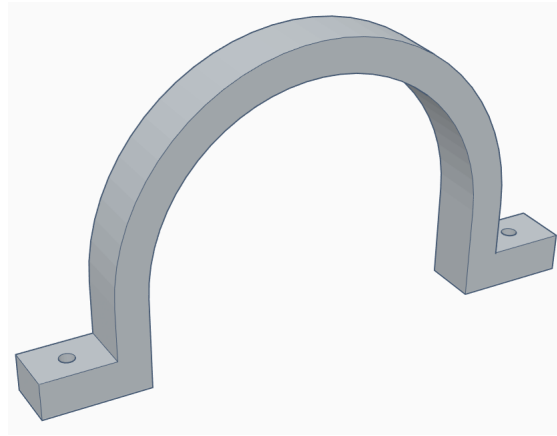
Figure 2.17: Laser Cut Components



(a) Adjustable Castor Wheel Mount
(Isometric View)



(b) Wheel Adapter
(Isometric View)

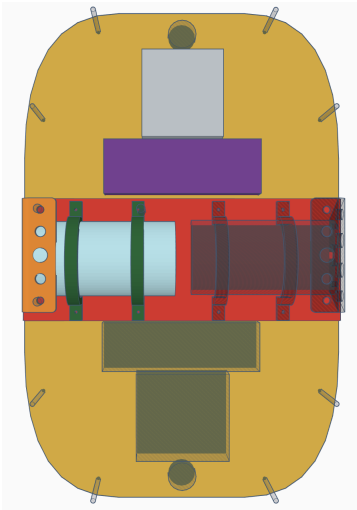


(c) Motor Clamp (Isometric View)

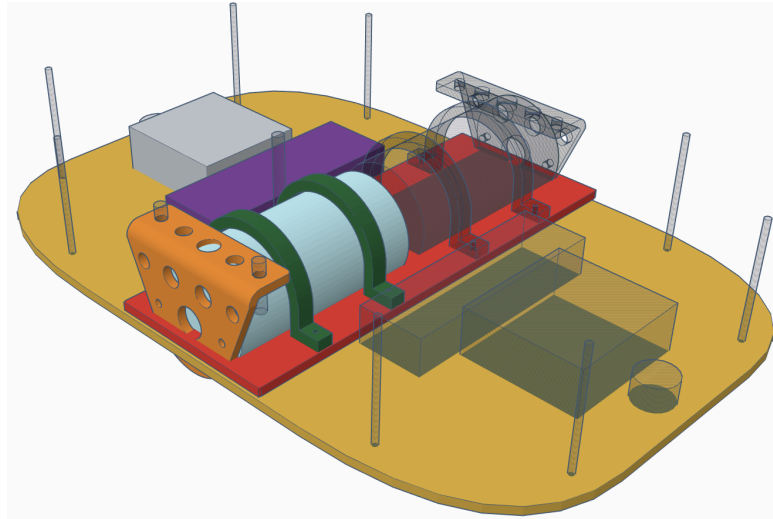
Figure 2.18: 3D Printed Components

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

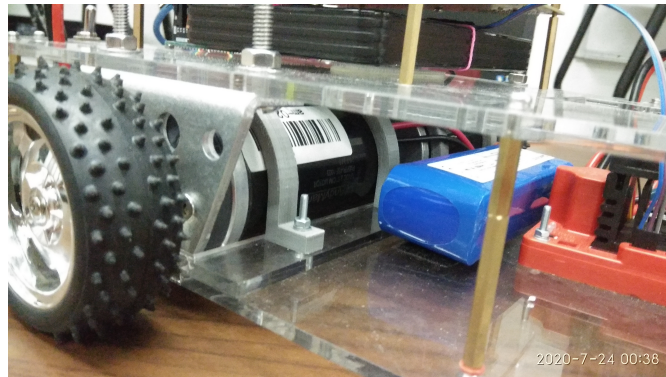
colors highlight the following components: Yellow - Base Plate; Red - Support Plate; Blue - Motor; Orange - Motor Bracket; Green - Motor Clamp; Violet - Li-Po Battery; Grey - Motor Driver. The motor driver and the Li-Po battery can either be attached using the M3 30mm bolts or Velcro strips.



(a) Top View



(b) Isometric View

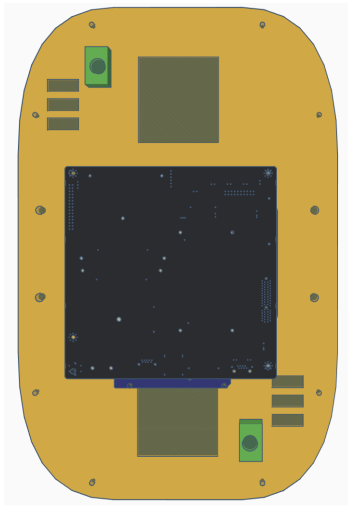


(c) Actual Assembly

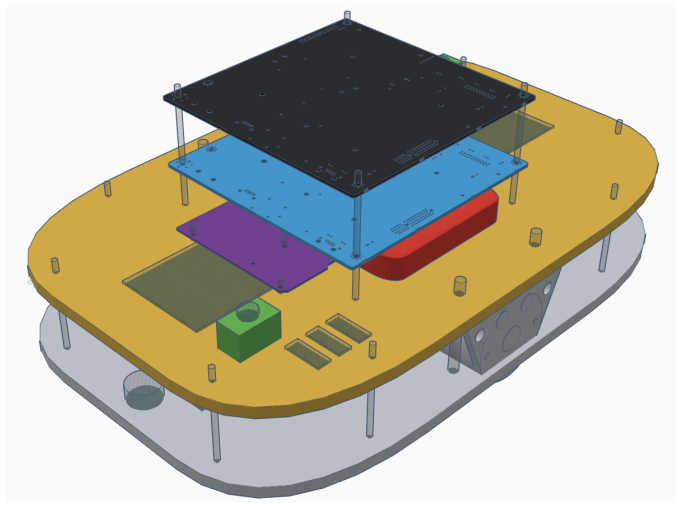
Figure 2.19: Chassis Assembly 1

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

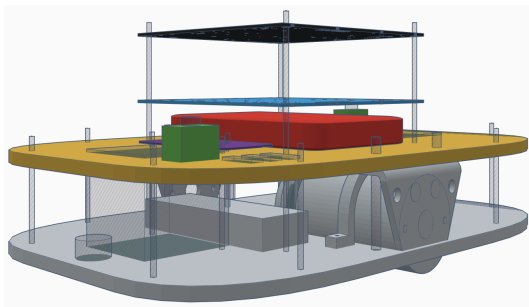
to the motor clamps using the 1/4" - 28 size bolts (Figure 2.20g) and M3 standoffs (Figure 2.20f). The SPST switch can be screwed into the base plate (Figure 2.20h), while the Energizer Li-Po battery can be fixed using Velcro strips. The Arduino Mega and the TX2 modules have to be attached using the 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



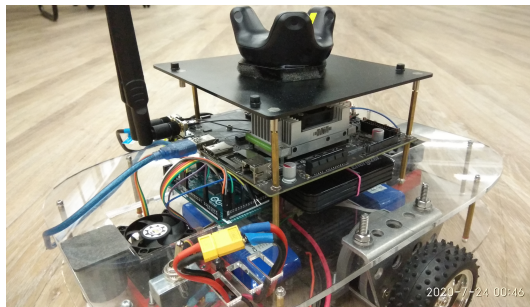
(a) Top View



(b) Isometric View

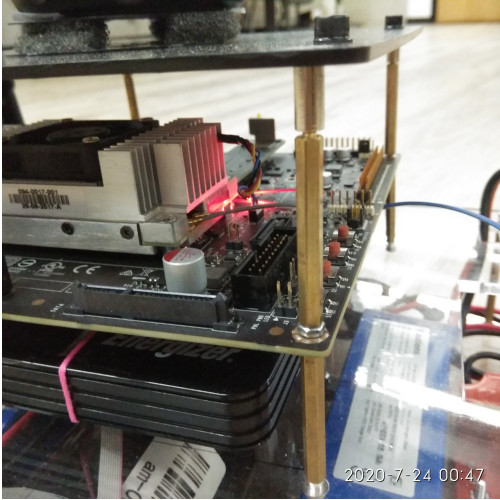


(c) Side View



(d) Actual Assembly

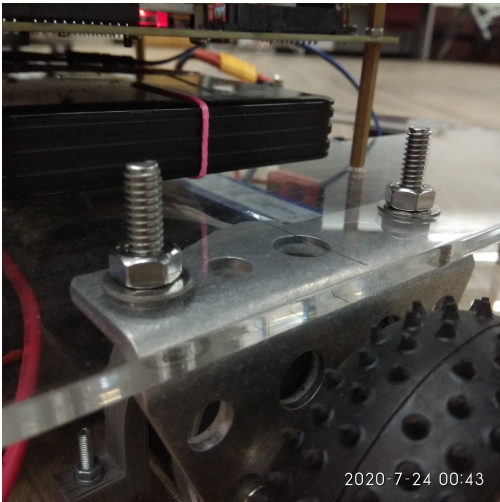
Figure 2.20: Chassis Assembly 2



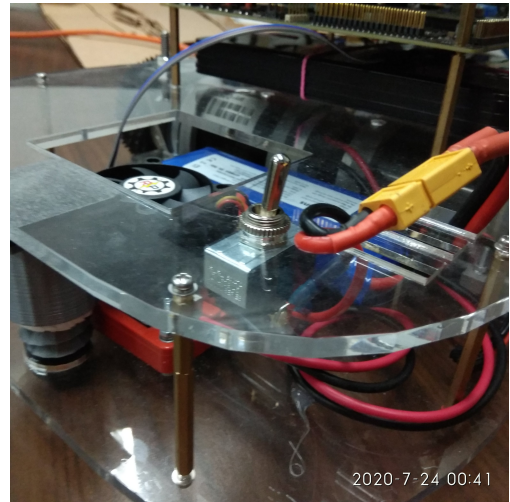
(e) M3 Standoffs(20mm, 60mm)



(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.

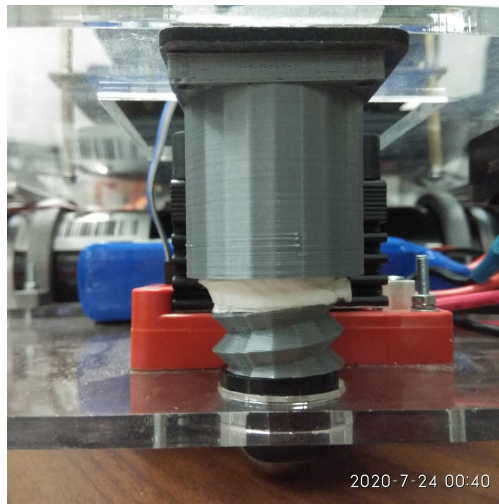


Figure 2.21: Adjustable Castor Wheel

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.



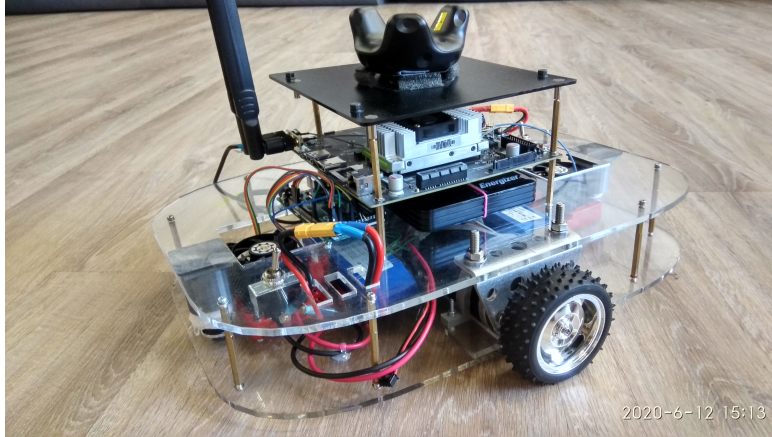
Figure 2.22: Tracker with Damper

2.5 C^4S Requirements and Software Architecture

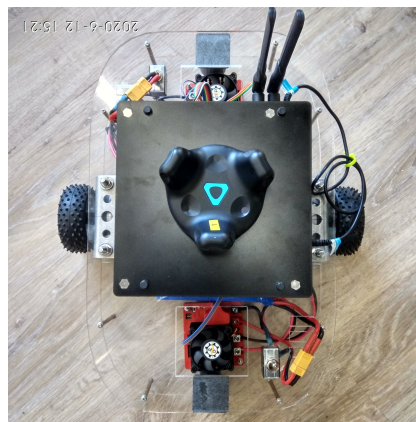
In this section, a detailed overview of the system-level architecture involved in addressing the global/local command, control, computing, communications (C^4), and sensing (S) requirements of the fleet of DDV's. Figure 2.24 describes the software architecture of the DDV, and the following discussion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

³robotic 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

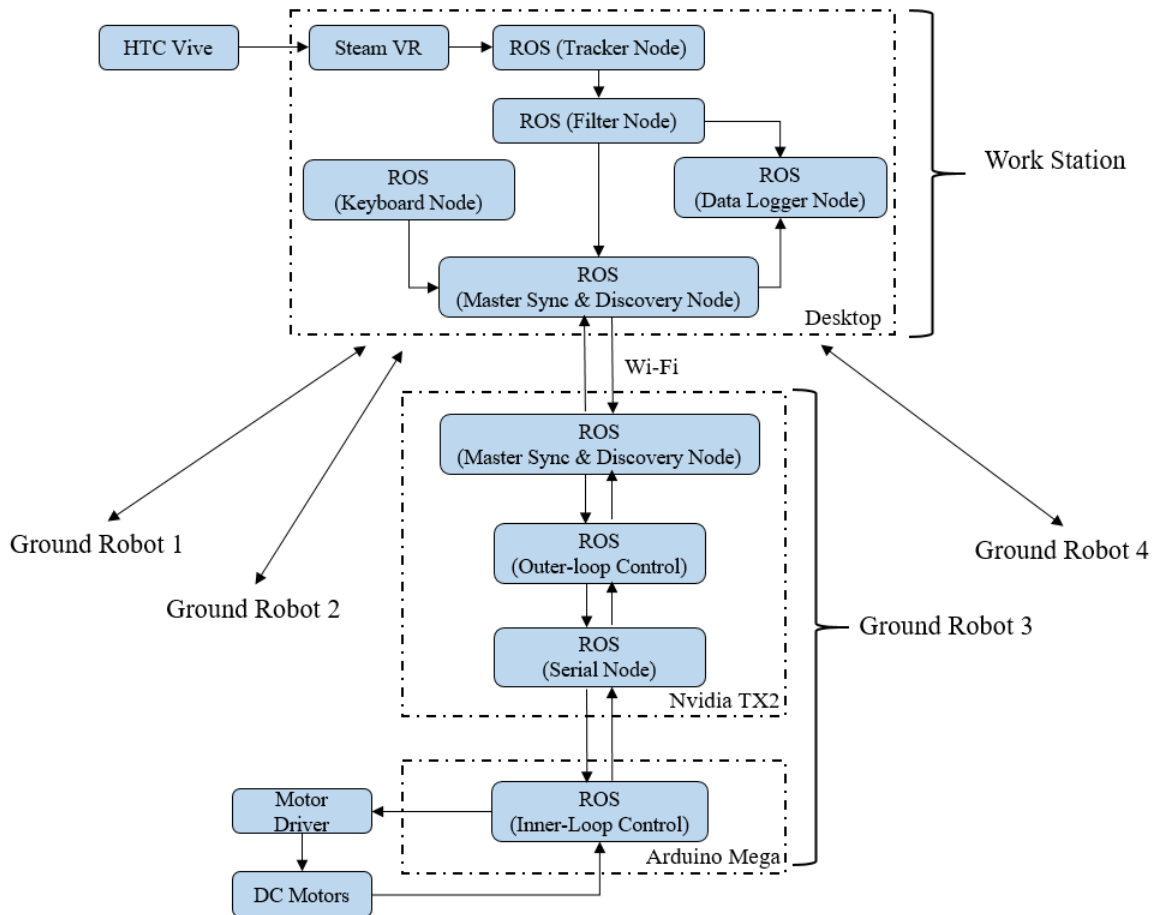


Figure 2.24: DDV Software Architecture

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.

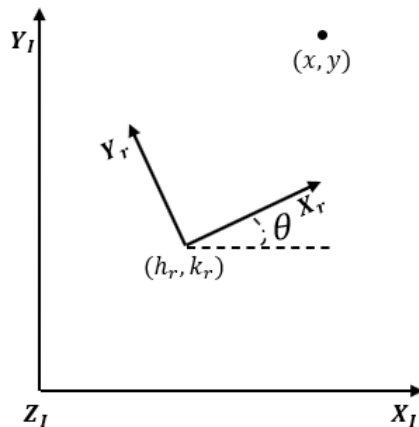


Figure 3.1: Rotation of Translation of Coordinate Axis in Two Dimensions

$$x' = (x - h_r) \cos \theta - (y - k_r) \sin \theta \quad (3.5)$$

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

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 translational 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 $\phi, \theta, \text{ and } \psi$ 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.

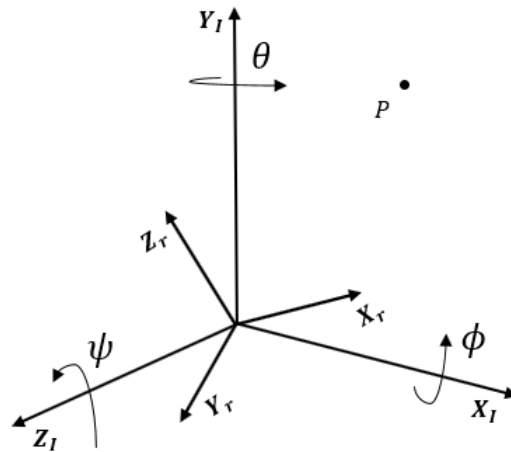


Figure 3.2: Rotation in Three Dimensions

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

¹Please 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:

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

where

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

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

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

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

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 be represented as $q_0 + q_1i + q_2j + q_3k$, 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:

$$q_{P'} = q_R q_P q_R^* \quad (3.12)$$

and here

$$q_R = q_0 + q_1i + q_2j + q_3k \quad (3.13)$$

$$q_R^* = q_0 - q_1i - q_2j - q_3k \quad (3.14)$$

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

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

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

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

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

here R is the rotation matrix from equation (3.11). Similarly, given q_R the rotation

matrix can be calculated as follows:

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

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:

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

$$y(t) = g(x(t), u(t)) \quad (3.21)$$

where, f is function mapping $\mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$, and g is a function mapping $\mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^p$. 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

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

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]:

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

$$\delta y(t) = C\delta x(t) + D\delta u(t) \quad (3.24)$$

where

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

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

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

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

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 $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$; $P_{[e_{a_r}, e_{a_l}] \rightarrow [v, \omega]}$; $P_{[e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}] \rightarrow [\omega_r, \omega_l]}$. 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. Within 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

$$-\dot{x} \sin \theta + \dot{y} \cos \theta = 0 \quad (4.1)$$

- **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.

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

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, ω) 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

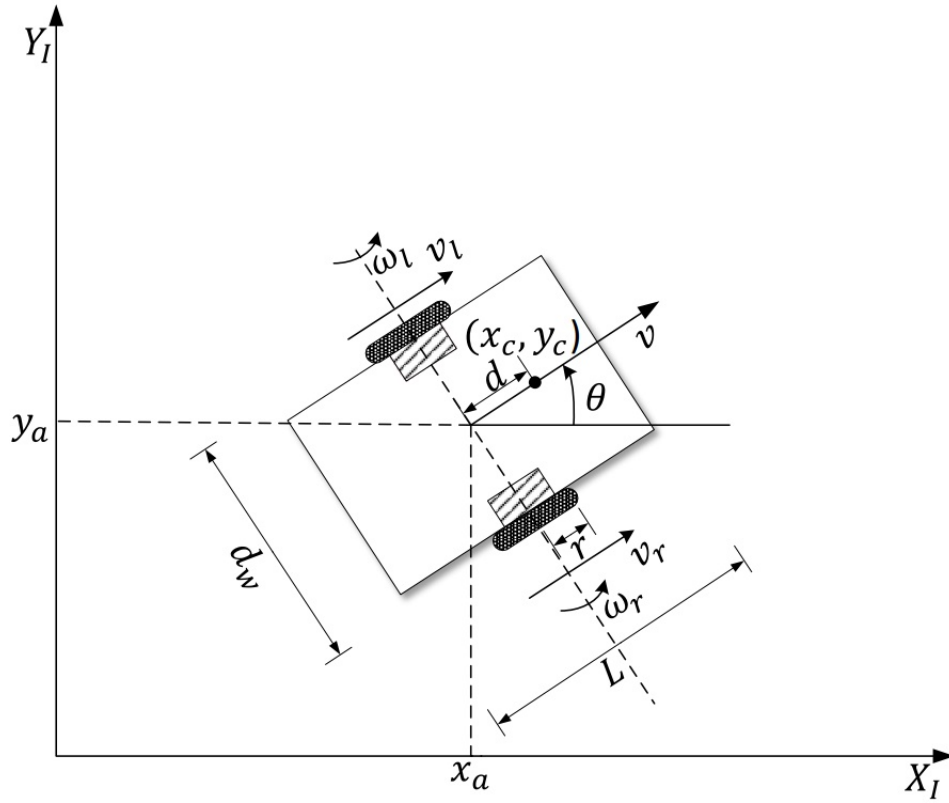


Figure 4.1: DDV Visualization

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

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = M \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \quad (4.3)$$

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

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).

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

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

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

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.

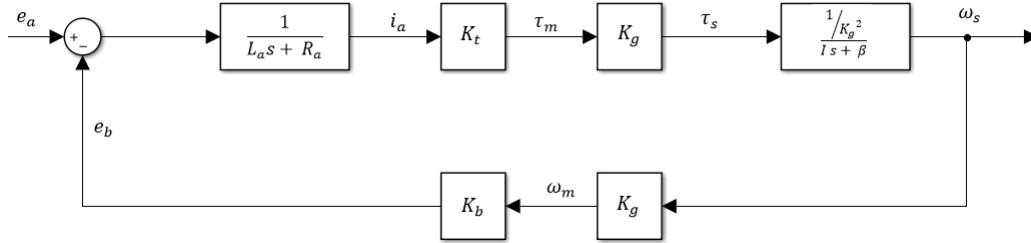


Figure 4.2: DC Motor Speed - Voltage Dynamical Model

Armature Equation:

$$e_{a,r,l} = L_a \frac{di_{a,r,l}}{dt} + R_a i_{a,r,l} + e_{b,r,l} \quad (4.8)$$

Back EMF Equation:

$$e_{b,r,l} = K_b K_g \omega_{r,l} \quad (4.9)$$

Torque Equation:

$$\tau_{b,r,l} = K_t i_{a,r,l} \quad (4.10)$$

Load Equation:

$$\tau_{r,l} = K_g \tau_{b,r,l} - K_g^2 \beta \omega_{r,l} \quad (4.11)$$

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:

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

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

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

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

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

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

From equation (4.15) it can be inferred that the bandwidth of the motor is dependent on the parameters β, 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 dynamometer 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	$1.67 \times 10^{-6} \text{ kg m}^2$	$573 \times 10^{-6} \text{ kg m}^2$	$29 \times 10^{-6} \text{ kg m}^2$
L_a	$1.729 \times 10^{-6} \text{ H}$	$13.2 \times 10^{-6} \text{ H}$	$13.2 \times 10^{-6} \text{ H}$
R_a	$3.01 \ \Omega$	$0.8 \ \Omega$	$0.8 \ \Omega$
K_b	$9.5 \times 10^{-3} \text{ V/rad/s}$	0.201 V/rad/s	0.0183 V/rad/s
K_t	$9.5 \times 10^{-3} \text{ Nm/A}$	0.201 Nm/A	0.0183 Nm/A
K_g	50	1	1
β	$3.29 \times 10^{-6} \text{ Nms}$	$7.4 \times 10^{-6} \text{ Nms}$	$110 \times 10^{-6} \text{ Nms}$

Table 4.3: DC Motor Parameter Values

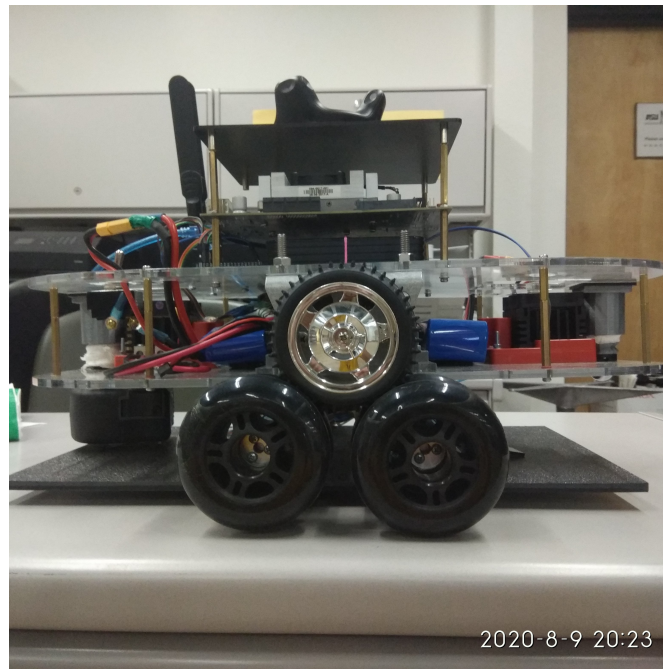


Figure 4.3: Small Scale Chassis Dynamo-meter Setup

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.

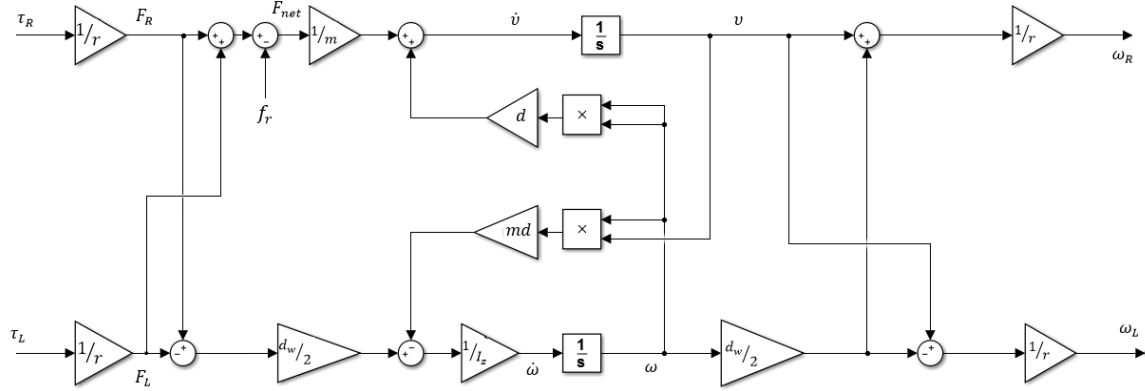


Figure 4.4: DDV Dynamics $(\tau_r, \tau_l) \rightarrow (\omega_r, \omega_l)$

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

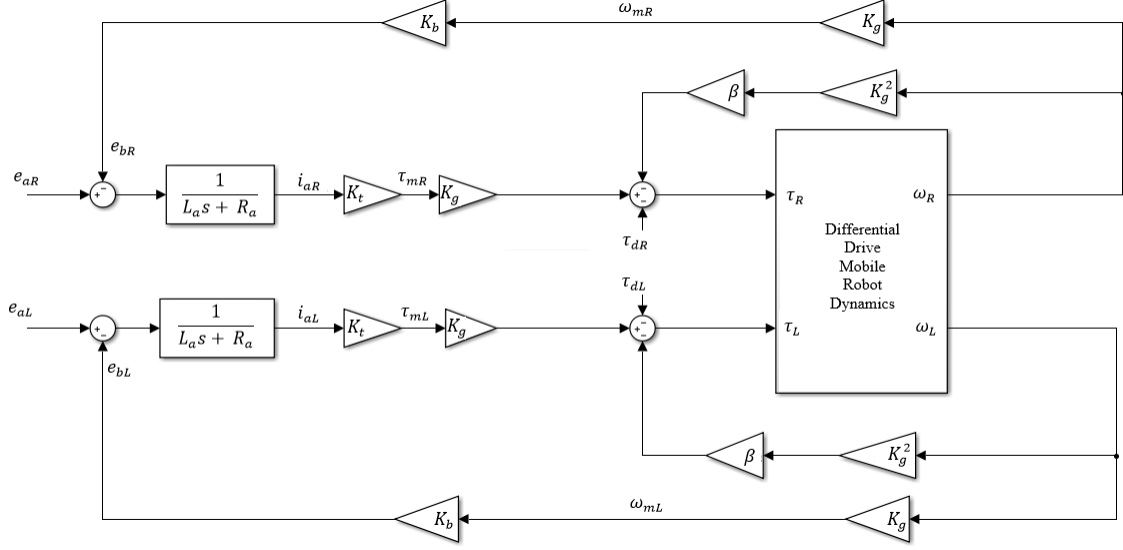


Figure 4.5: DDV Dynamics $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$

$$\dot{x} = Ax + Bu \quad y = Cx + Du \quad (4.16)$$

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

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

$$C = \begin{bmatrix} \frac{1}{r} & \frac{d_w}{2r} & 0 & 0 \\ \frac{1}{r} & \frac{-d_w}{2r} & 0 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (4.18)$$

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: $\beta, K_b, K_t, R_a, L_a$. 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:

$$P_{[e_{a_r}, e_{a_l}] \rightarrow [v, \omega]} = MP_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]} \quad (4.19)$$

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

$$M = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{d_w} & -\frac{r}{d_w} \end{bmatrix}, \quad E = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{2}{2} \end{bmatrix} \quad (4.21)$$

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

$$\dot{x} = Ax + Bu \quad y = MCx + Du \quad (4.22)$$

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

$$\dot{x} = Ax + BEu \quad y = MCx + Du \quad (4.23)$$

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

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

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

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_{ar}, e_{al}) 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

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

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

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.

$$I_{decoupling} = m\left(\frac{d_w}{2}\right)^2 \quad (4.28)$$

The consequence of this result is that

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

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

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

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 be considered while designing a DDV in order to obtain a linear and decoupled dynamical system. An aspect ratio 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 $(e_{ar}, e_{al}) \rightarrow (\omega_r, \omega_l)$ system in equation (4.24), can a similar result be obtained for the $(e_{ar}, e_{al}) \rightarrow (v, \omega)$ 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 $(e_{ar}, e_{al}) \rightarrow (v, \omega)$ cannot be decoupled by constraining the design parameters, nevertheless, let us consider the following linear transformation on the plant input

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

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

$$\begin{bmatrix} \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \frac{1}{c_{1r}} & 0 \\ 0 & -\frac{d_w}{c_{2r}} \end{bmatrix} \begin{bmatrix} \tau_r + \tau_l \\ \tau_r - \tau_l \end{bmatrix} \quad (4.31)$$

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 $(e_{ar} + e_{al}, e_{ar} - e_{al}) \rightarrow (v, \omega)$ system as well. The consequence of this result that

²more 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.

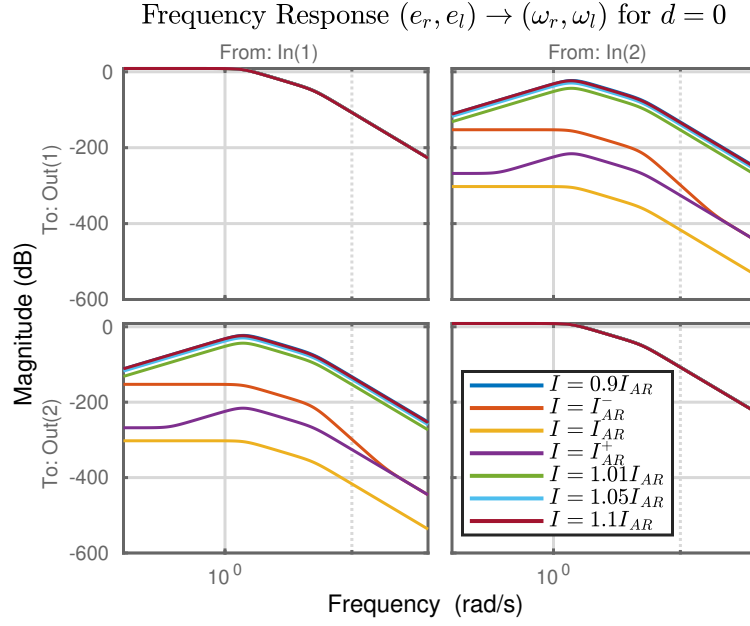


Figure 4.6: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying I

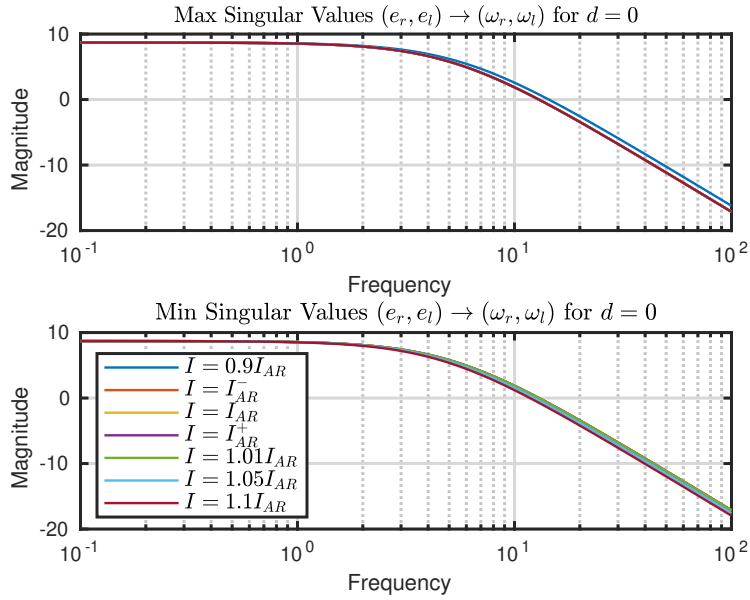


Figure 4.7: Singular Value Response of $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ System: Varying I

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

be made for the $(e_{ar}, e_{al}) \rightarrow (\omega_r, \omega_l)$ 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

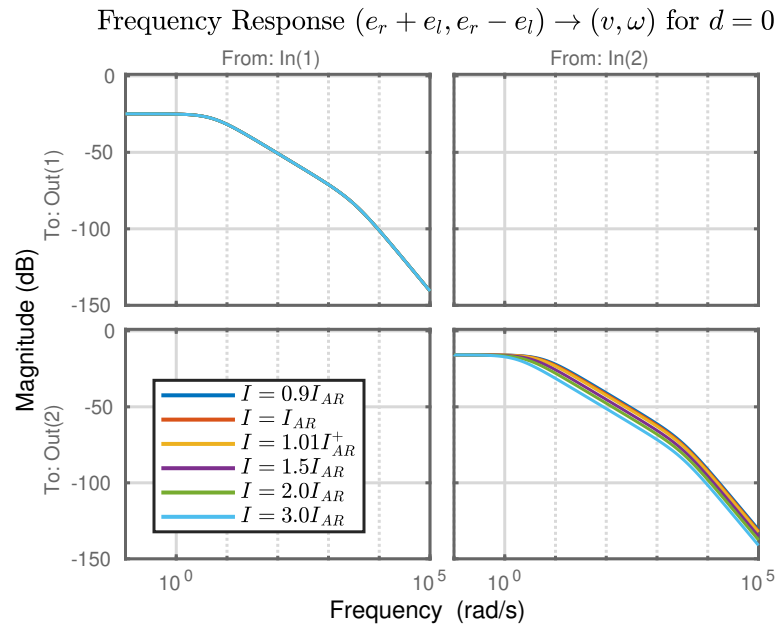


Figure 4.8: Bode Magnitude Response of $P_{[e_{ar}+e_{al}, e_{ar}-e_{al}] \rightarrow [v, \omega]}$ System: Varying I

³ $I_{AR} = I_{decoupling}$

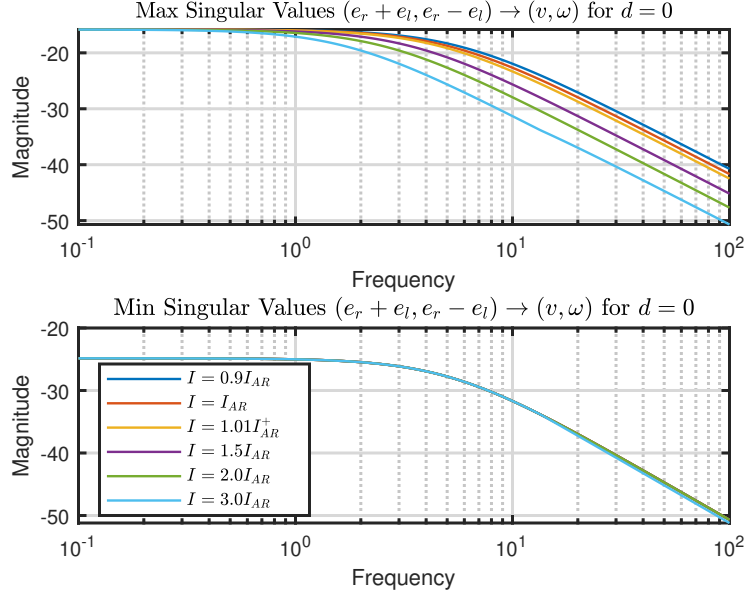


Figure 4.9: Singular Value Response of $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ System: Varying I

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

- The system is decoupled for all possible variations of I
- It can be seen 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 $(e_{a_r} + e_{a_l} \rightarrow v)$ channel is unperturbed by the variations in I , however, the response from the $(e_{a_r} - e_{a_l} \rightarrow \omega)$ 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.

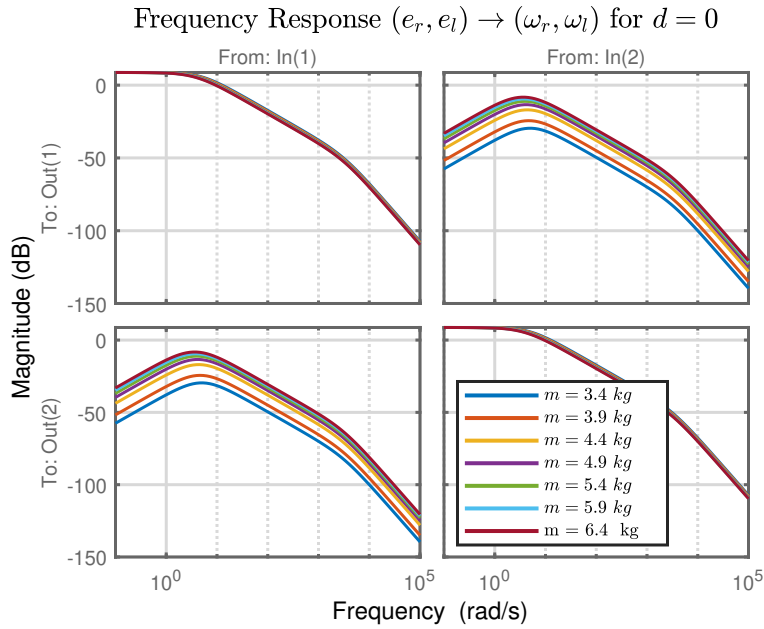


Figure 4.10: Bode Magnitude Response of $P_{[e_r, e_l] \rightarrow [\omega_r, \omega_l]}$ System: Varying m

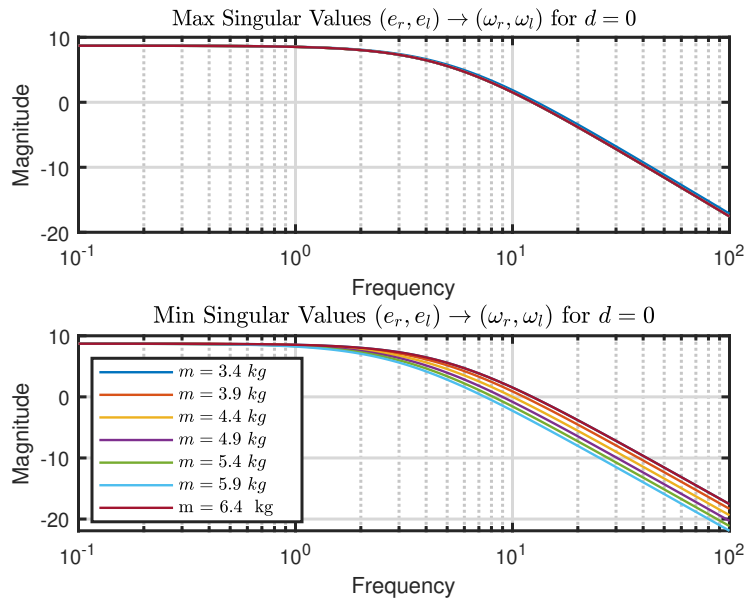


Figure 4.11: Singular Value Response of $P_{[e_r, e_l] \rightarrow [\omega_r, \omega_l]}$ System: Varying m

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

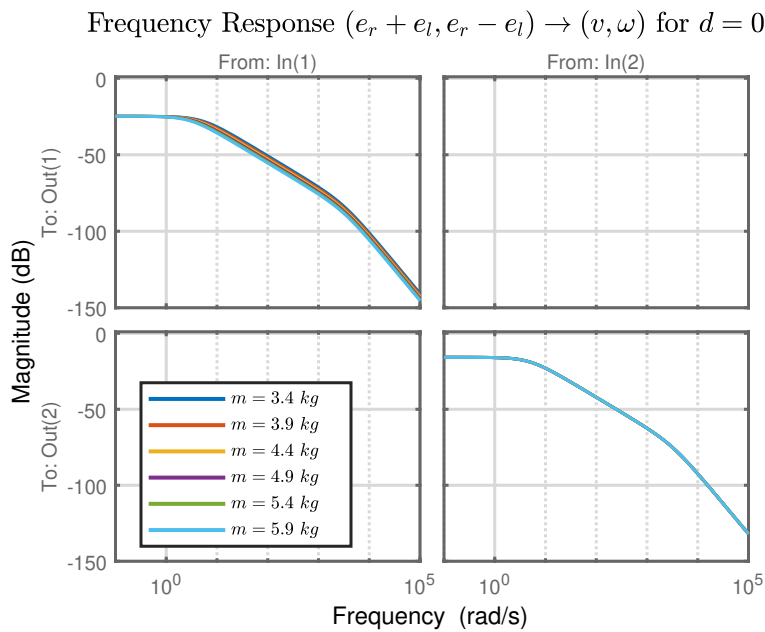


Figure 4.12: Bode Magnitude Response of $P_{[e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}] \rightarrow [v, \omega]}$ System: Varying m

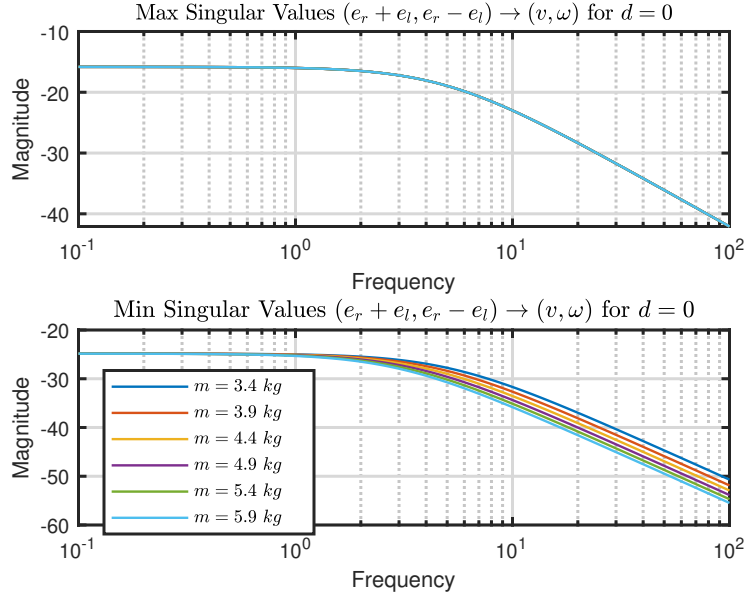


Figure 4.13: Singular Value Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}]} \rightarrow [v, \omega]$ System: Varying m

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

- The major point over here is that the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ system remains decoupled irrespective of the variation in the total mass of the system
- The response from the $(e_{a_r} - e_{a_l} \rightarrow \omega)$ channel is unperturbed by the variations in m , however, the response from $(e_{a_r} + e_{a_l} \rightarrow \checkmark)$ 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.

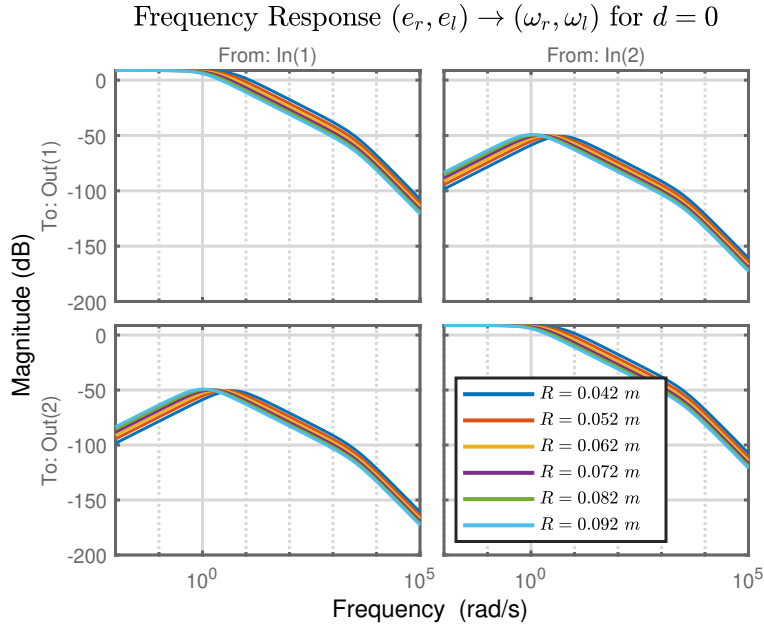


Figure 4.14: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying r

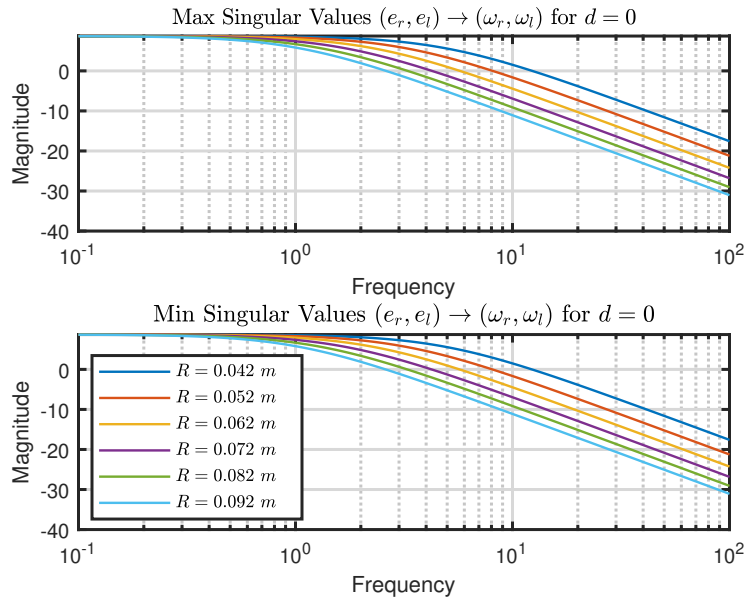


Figure 4.15: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying r

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

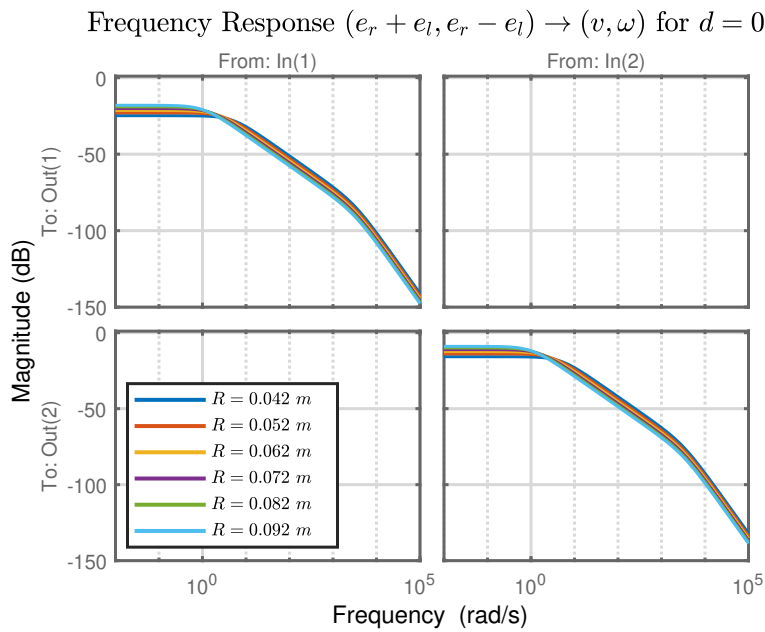


Figure 4.16: Bode Magnitude Response of $P_{[e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}] \rightarrow [v, \omega]}$ System: Varying r

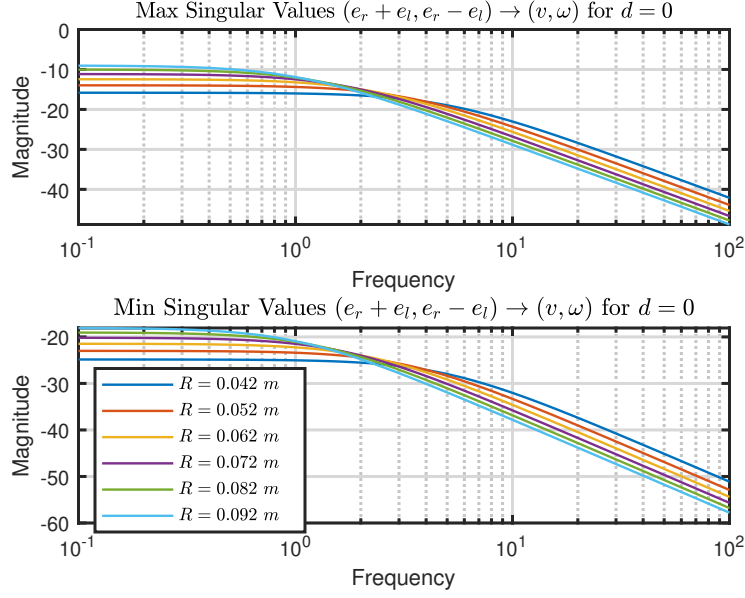


Figure 4.17: Singular Value Response of $P_{[e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}] \rightarrow [v, \omega]}$ System: Varying r

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:

- The main point to be noticed is that $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ 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 $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ 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.

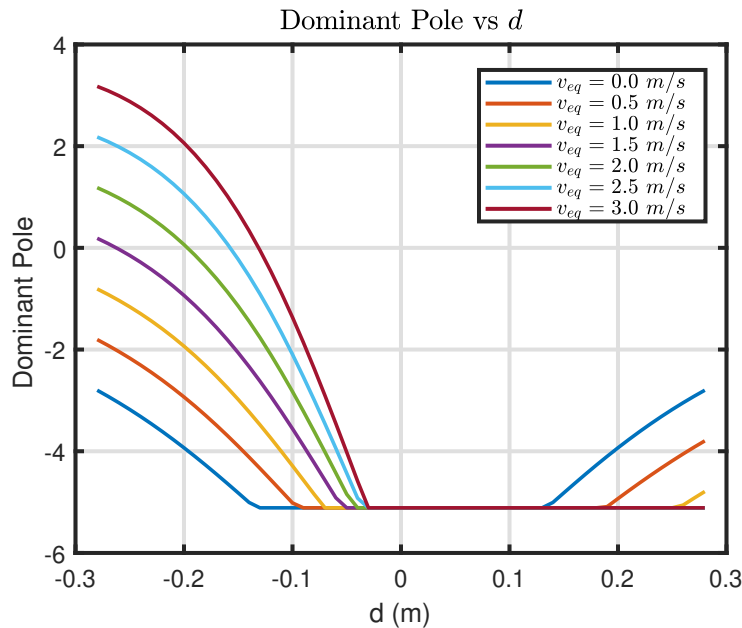


Figure 4.18: Dominant Pole Variation with d

From the above 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 $(e_{ar}, e_{al}) \rightarrow (\omega_r, \omega_l)$ and the $(e_{ar} + e_{al}, e_{ar} - e_{al}) \rightarrow (v, \omega)$ systems for the variation in v_{eq} values at $d = 0.1$ m and $\omega_{eq} = 0.8$ rad/sec.

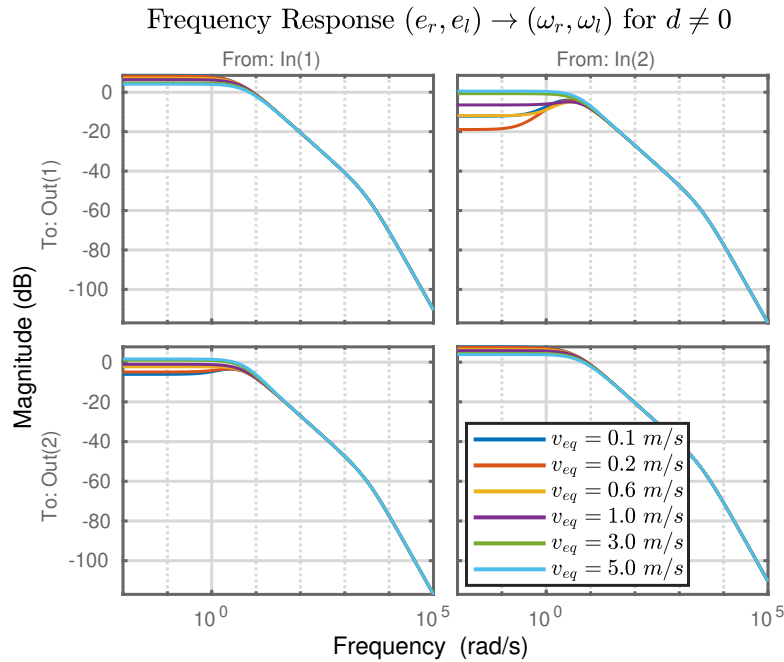


Figure 4.19: Bode Magnitude Response of $P_{[e_{ar}, e_{al}] \rightarrow [\omega_r, \omega_l]}$ System: Varying v_{eq}

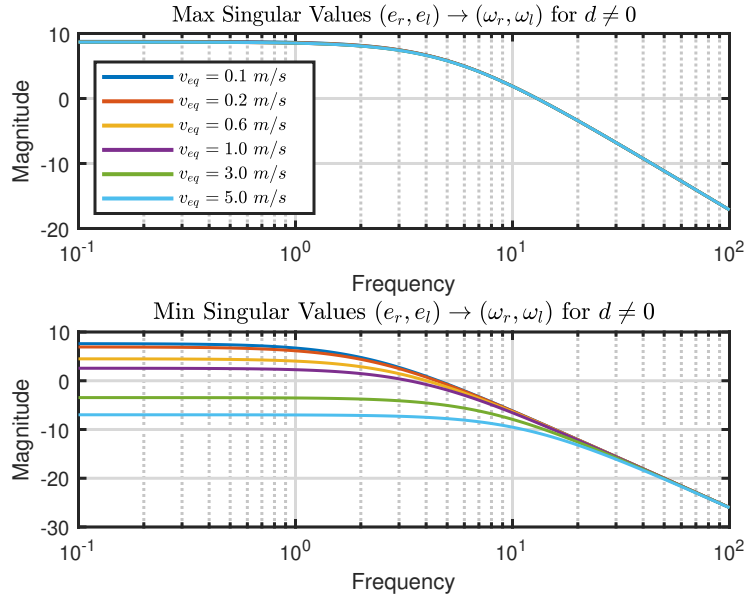


Figure 4.20: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying v_{eq}

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

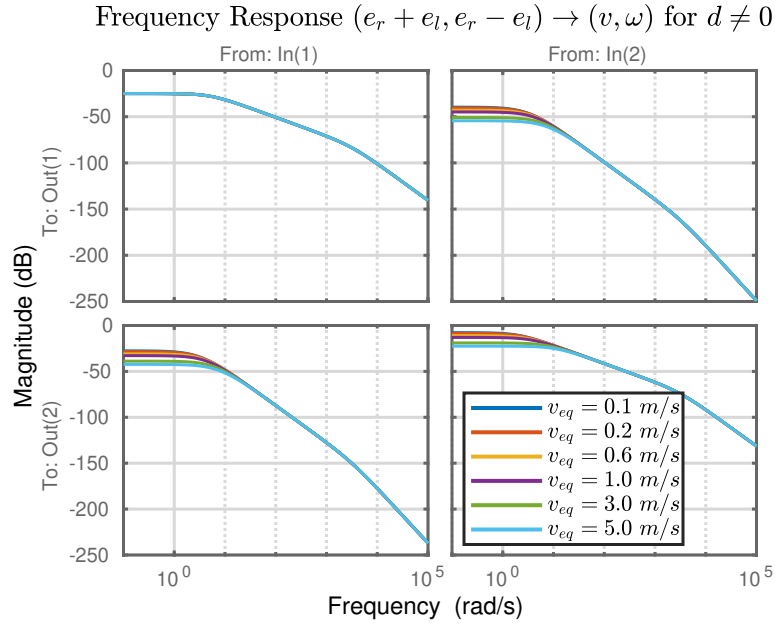


Figure 4.21: Bode Magnitude Response of $P_{[e_{ar}+e_{al}, e_{ar}-e_{al}] \rightarrow [v, \omega]}$ System: Varying v_{eq}

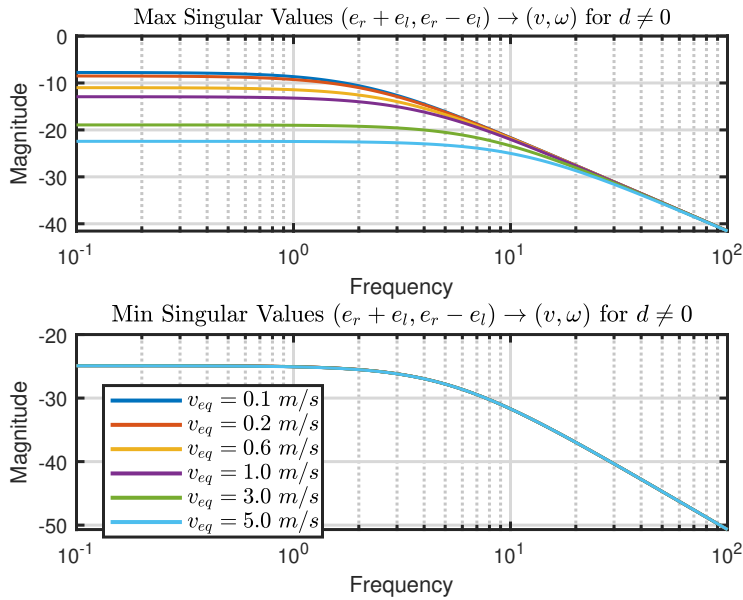


Figure 4.22: Singular Value Response of $P_{[e_{ar}+e_{al}, e_{ar}-e_{al}] \rightarrow [v, \omega]}$ System: Varying v_{eq}

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 $(e_{a_r} + e_{a_l} \rightarrow \checkmark)$ 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.

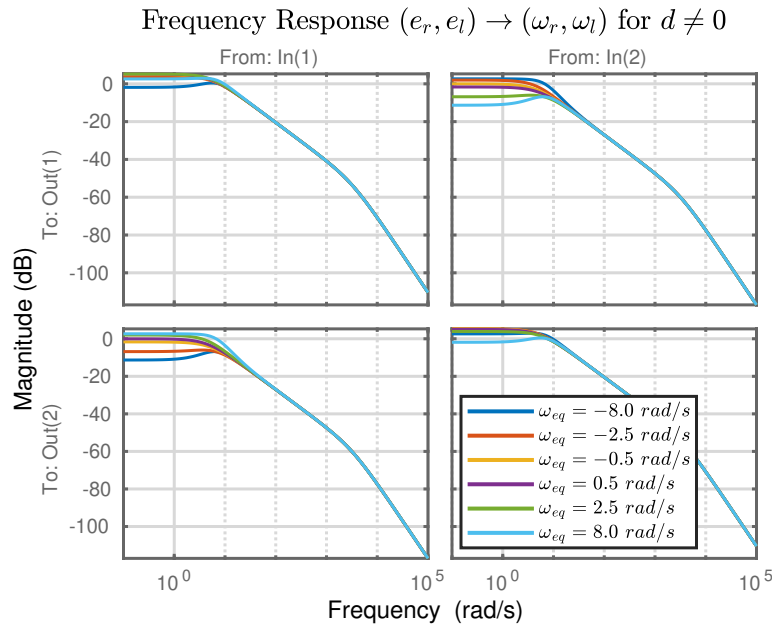


Figure 4.23: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying ω_{eq}

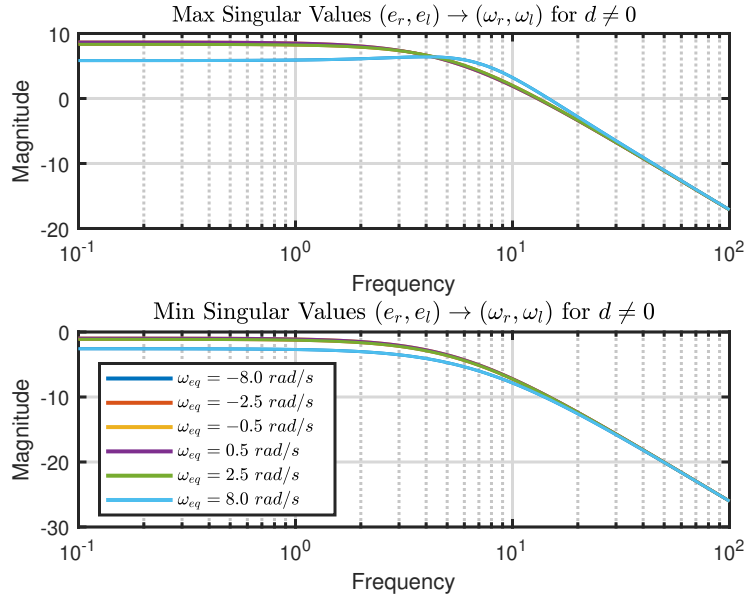


Figure 4.24: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying ω_{eq}

From the frequency response plots presented above, the following observations can be made for the $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ 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 $|\omega_{eq}|$

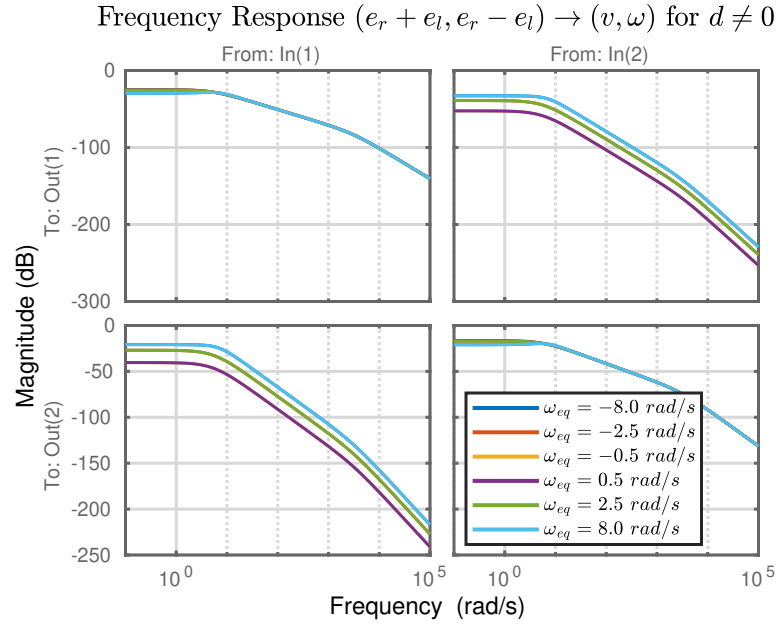


Figure 4.25: Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \rightarrow [v, \omega]}$ System: Varying ω_{eq}

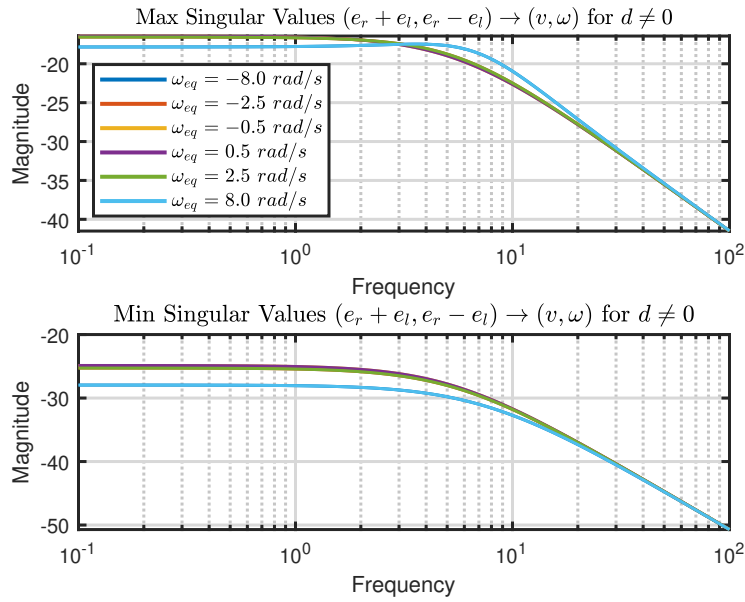


Figure 4.26: Singular Value Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \rightarrow [v, \omega]}$ System: Varying ω_{eq}

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 $w_{eq} = 0.8$ rad/sec.

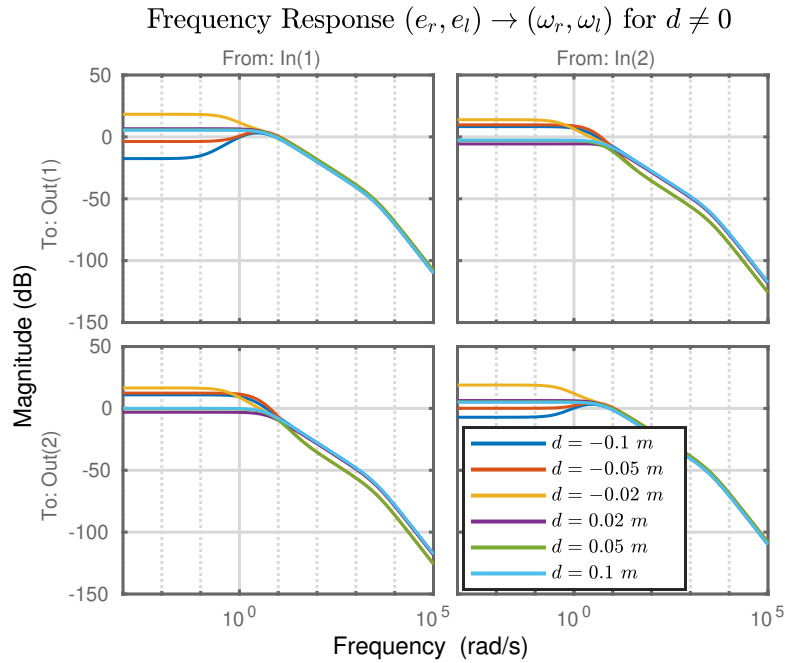


Figure 4.27: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying d

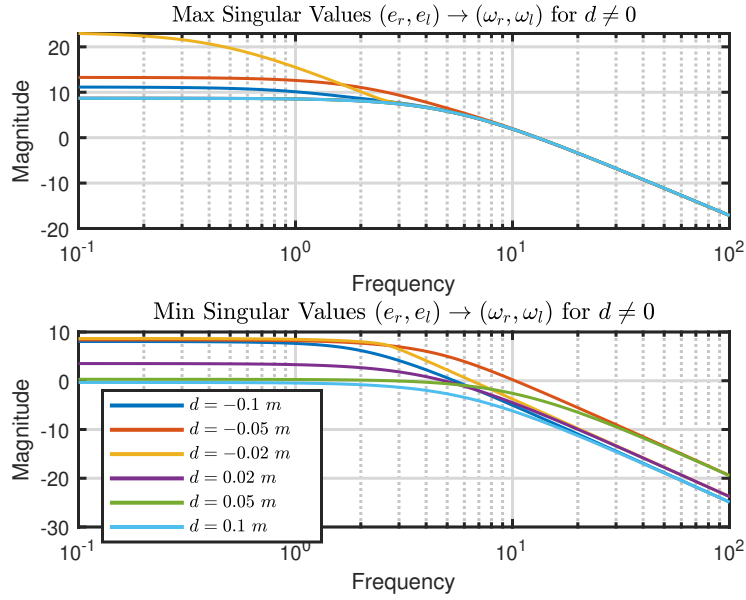


Figure 4.28: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying d

From the frequency response plots presented above, the following observations can be made for the $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ 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 $dv < 0$

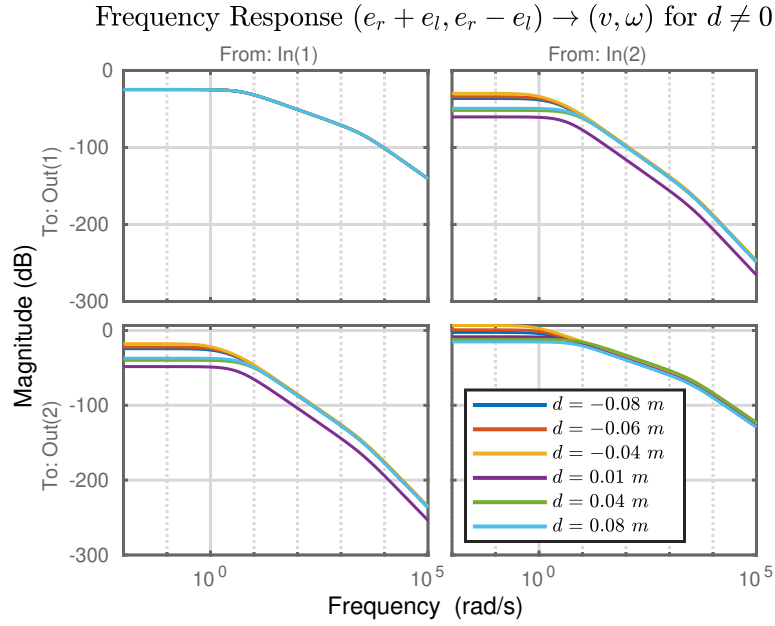


Figure 4.29: Bode Magnitude Response of $P_{[e_{ar}+e_{al}, e_{ar}-e_{al}]} \rightarrow [v, \omega]$ System: Varying d

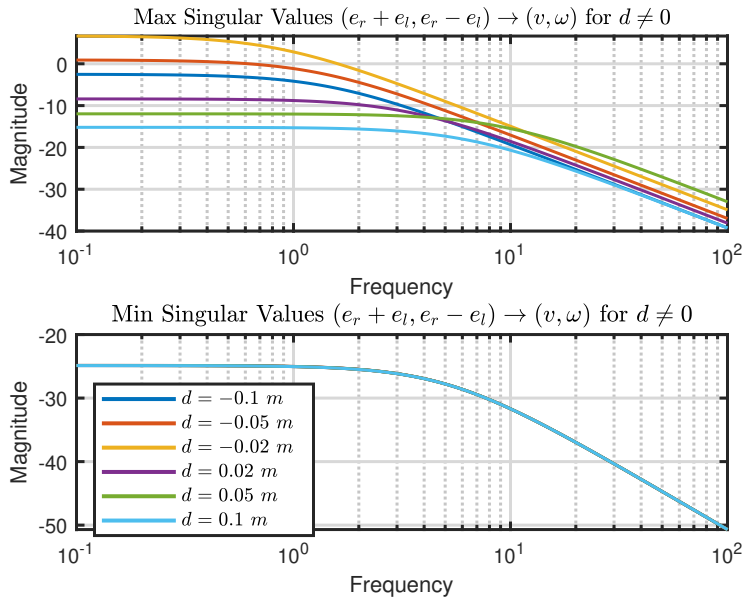


Figure 4.30: Singular Value Response of $psdv$ System: Varying d

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

be made for the $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \rightarrow [v, \omega]}$ 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 $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ plant, the diagonal elements in $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ are less susceptible to variations in d
- Almost negligible variation in the response from $(e_{a_r} + e_{a_l} \rightarrow \checkmark)$ 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 $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ and the $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ systems for the variation in I values at $v_{eq} = 2.0$ m/sec, $w_{eq} = 0.8$ rad/sec and $d = 0.1$ m.

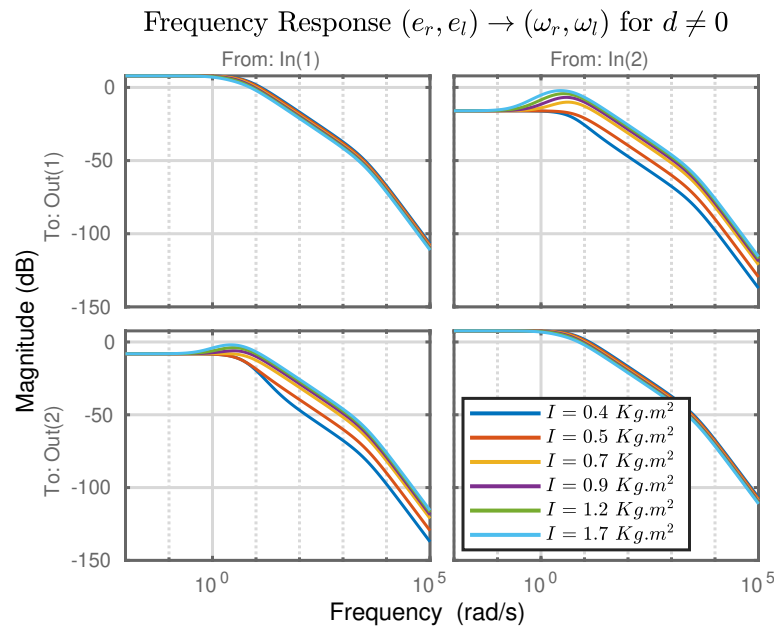


Figure 4.31: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying I

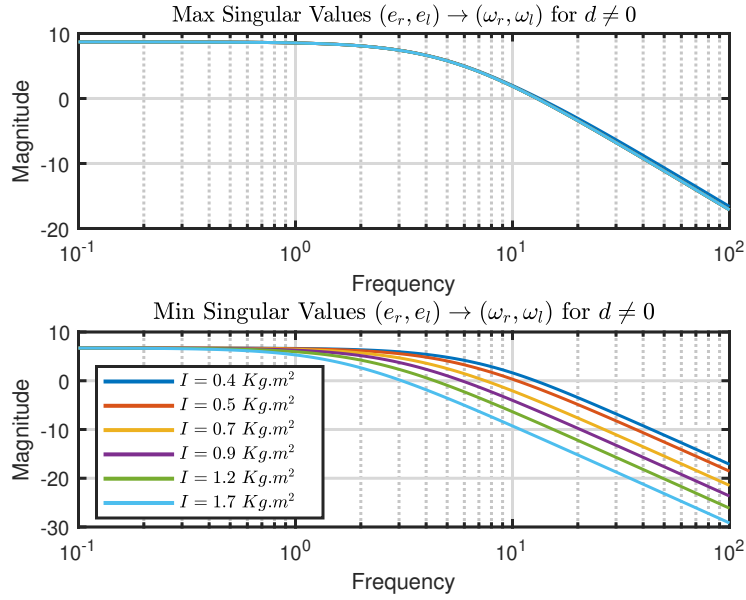


Figure 4.32: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying I

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:

- 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

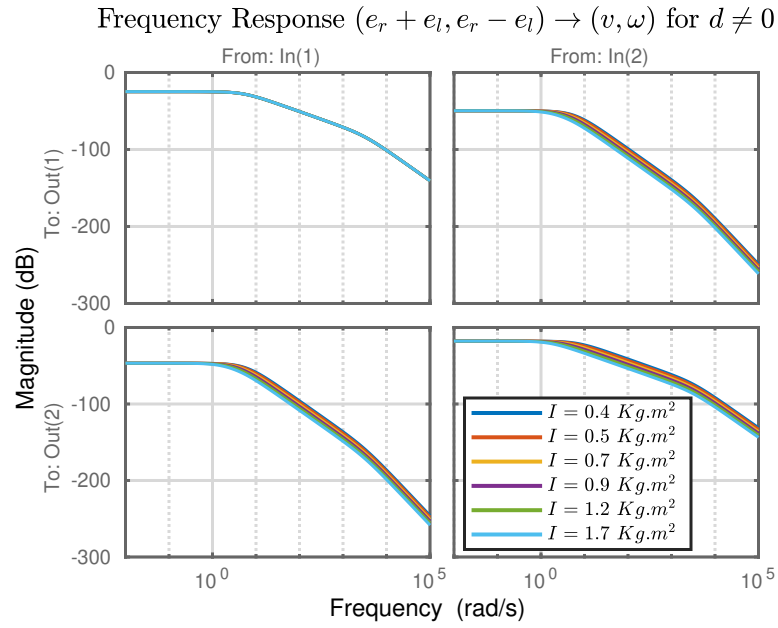


Figure 4.33: Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}]} \rightarrow [v, \omega]$ System: Varying I

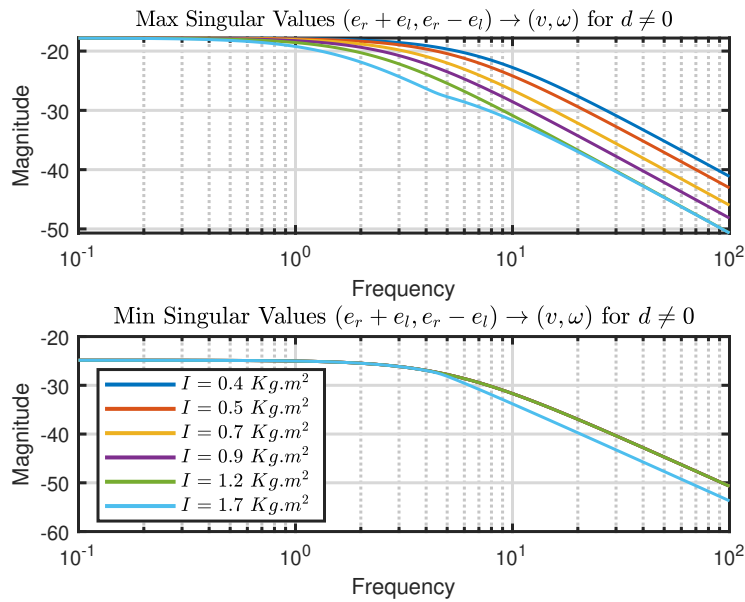


Figure 4.34: Singular Value Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}]} \rightarrow [v, \omega]$ System: Varying I

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 $(e_{a_r} + e_{a_l} \rightarrow \checkmark)$ 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.

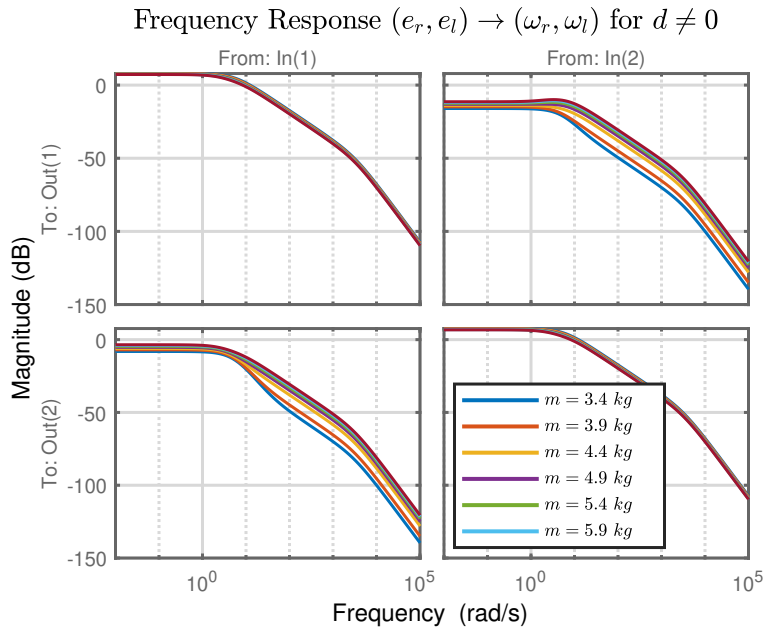


Figure 4.35: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying m

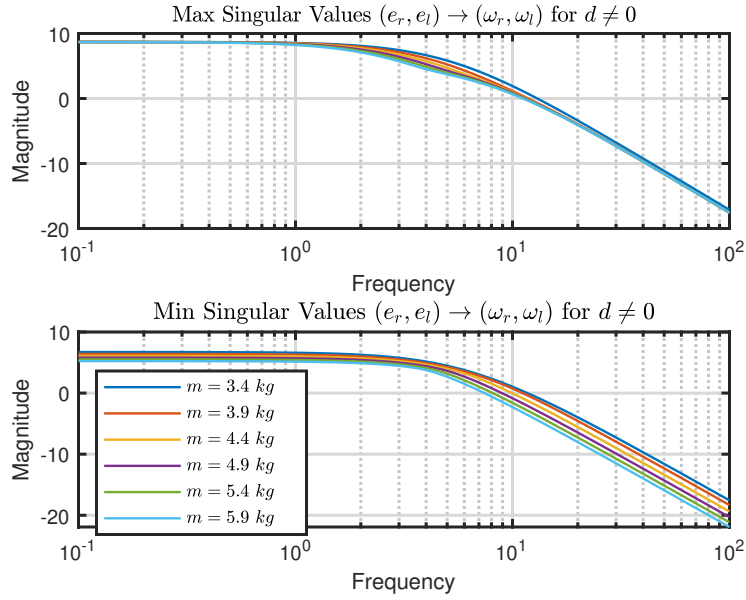


Figure 4.36: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying m

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

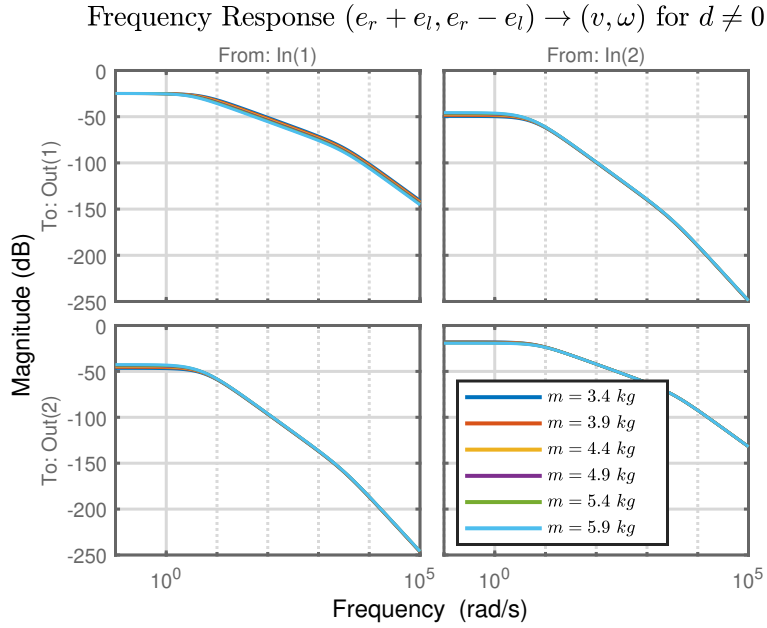


Figure 4.37: Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \rightarrow [v, \omega]}$ System: Varying m

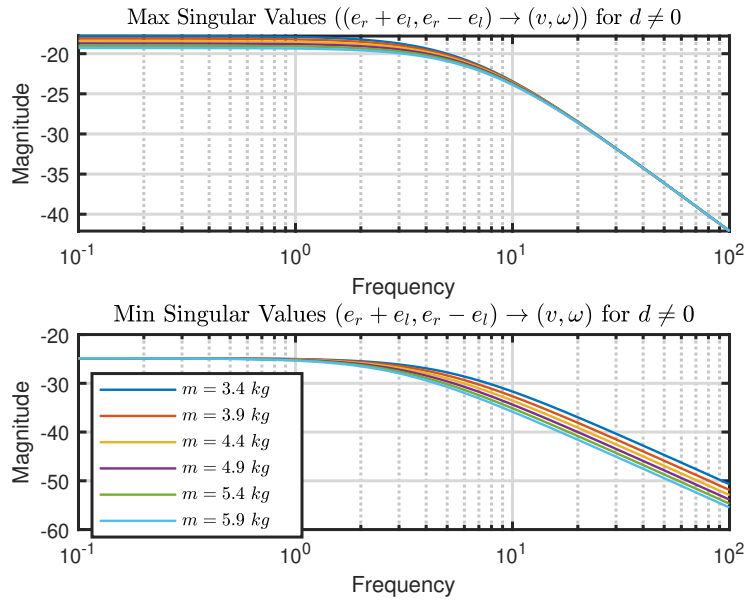


Figure 4.38: Singular Value Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}] \rightarrow [v, \omega]}$ System: Varying m

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 $(e_{a_r} + e_{a_l} \rightarrow \checkmark)$ 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 . 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 r values at $v_{eq} = 2.0$ m/sec, $w_{eq} = 0.8$ rad/sec and $d = 0.1$ m.

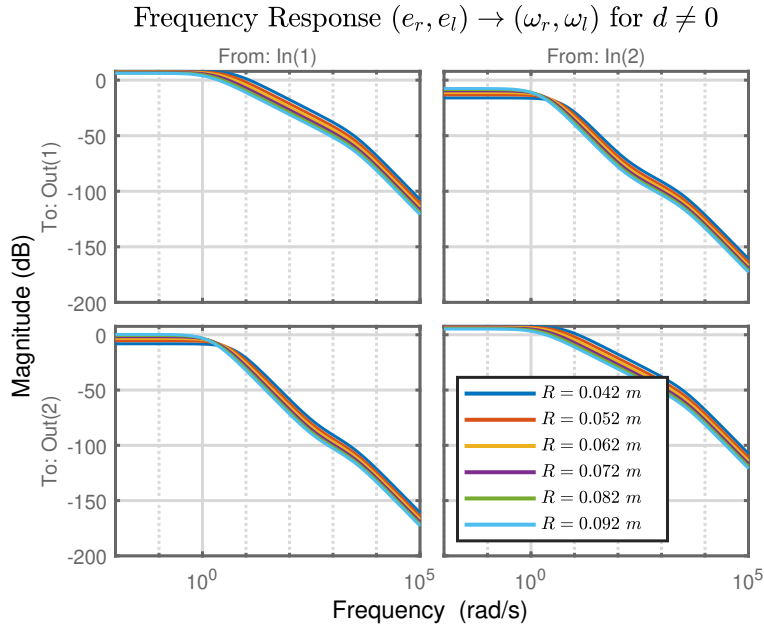


Figure 4.39: Bode Magnitude Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying r

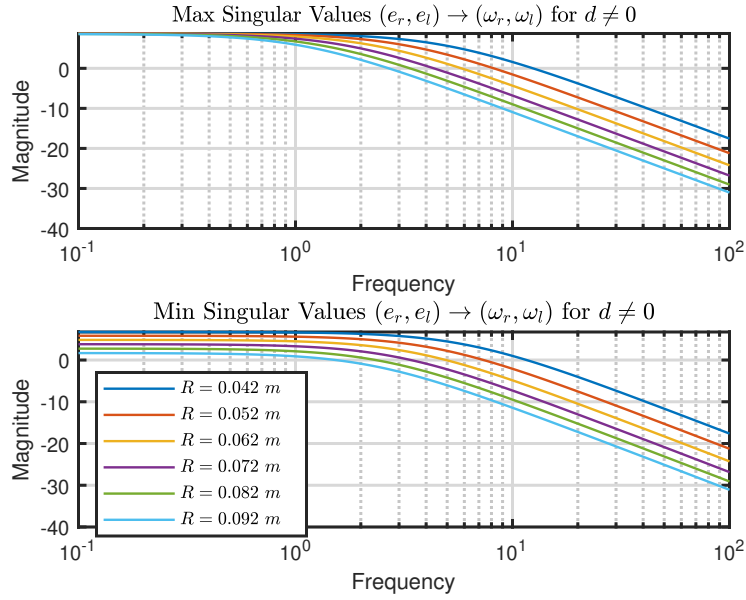


Figure 4.40: Singular Value Response of $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ System: Varying r

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

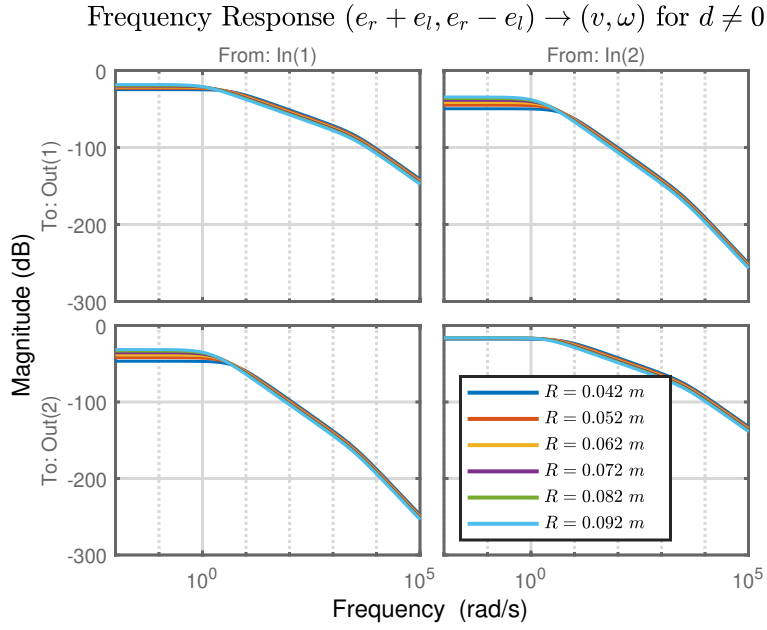


Figure 4.41: Bode Magnitude Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}]} \rightarrow [v, \omega]$ System: Varying r

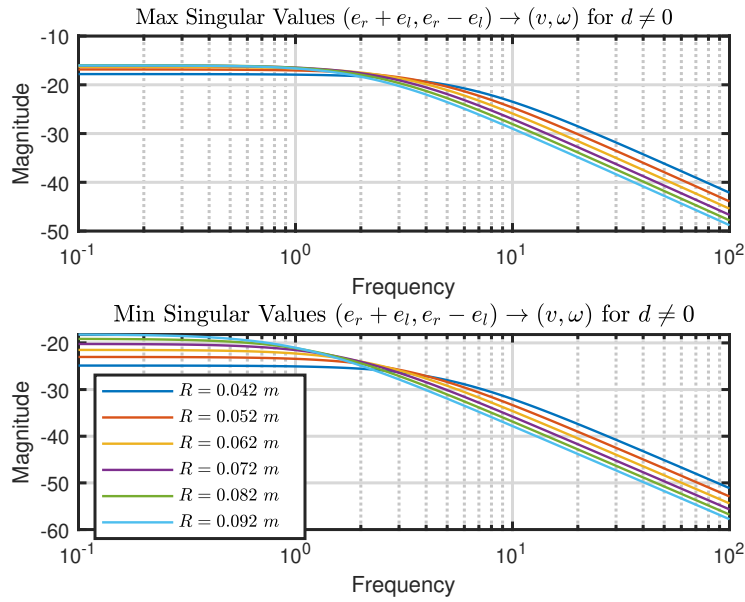


Figure 4.42: Singular Value Response of $P_{[e_{a_r}+e_{a_l}, e_{a_r}-e_{a_l}]} \rightarrow [v, \omega]$ System: Varying r

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 $(e_{a_r} + e_{a_l} \rightarrow \checkmark)$ 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.

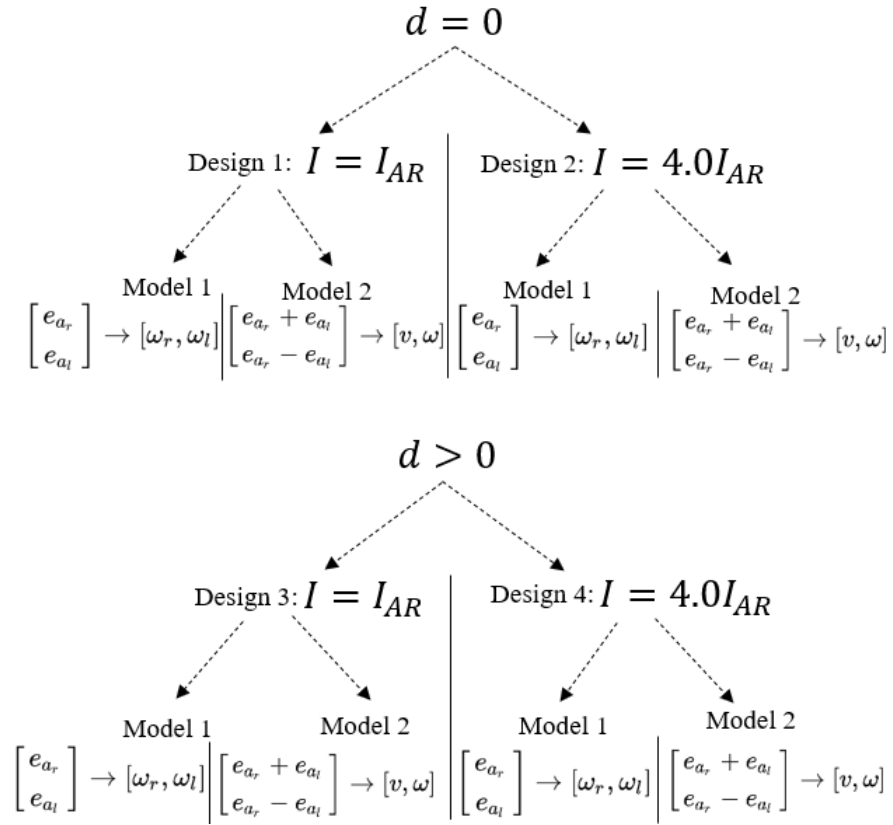


Figure 5.1: DDV Design Notations

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 $v_{eq} = 1 \text{ m/s}, \omega_{eq} = 0.8 \text{ rad/sec}^1$. 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: $D1M1 \approx D1M2 > D2M1 \approx D2M2; D3M1 \approx D3M2 > D4M1 \approx D4M2$
- Moment of Inertia: $D2M1 \approx D2M2 > D1M1 \approx D1M2; D4M1 \approx D4M2 > D3M1 \approx D3M2$
- Input-Output Coupling: $D2M1 > D1M1 \approx D2M2 \approx D1M1; D4M1 > D3M1 > D3M2 \approx D4M2$

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

PD1M1:

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

PD2M1:

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

PD3M1:

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

PD4M1:

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

PD1M2:

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

PD2M2:

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

PD3M2:

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

PD4M2:

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

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.

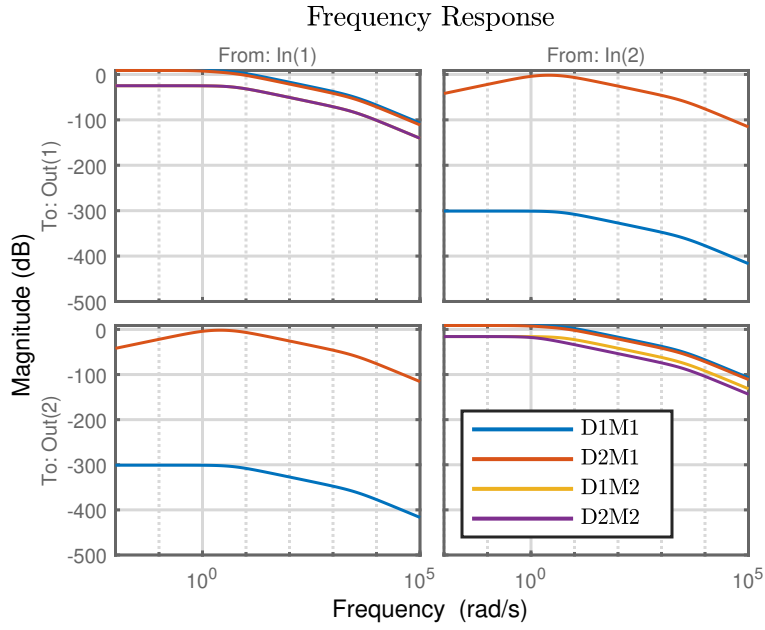


Figure 5.2: Plant Frequency Response D1 & D2

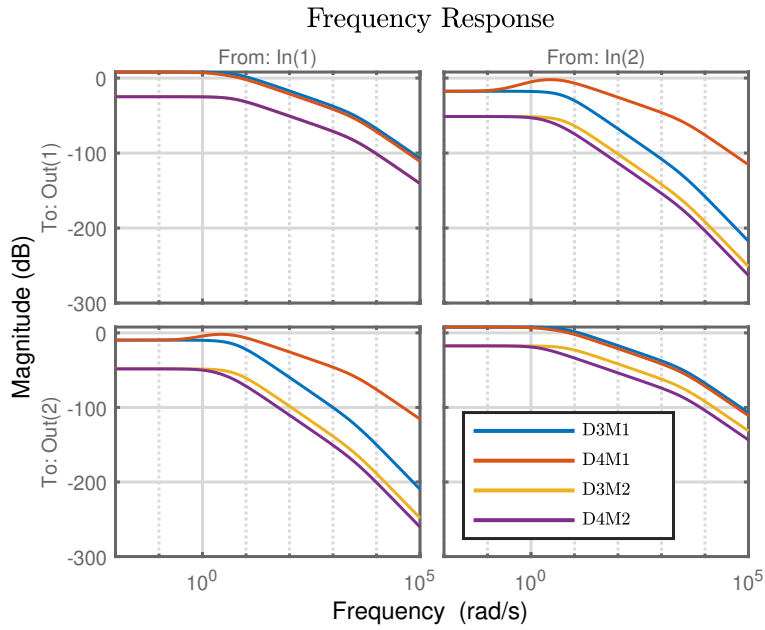


Figure 5.3: Plant Frequency Response D3 & D4

Plant Singular Values. The singular values of each of the eight designs are pre-

sented 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 $(e_{a_r} + e_{a_i} \rightarrow v)$ channel while the maximum singular value is associated with the $(e_{a_r} - e_{a_i} \rightarrow \omega)$ 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 $(e_{a_r} + e_{a_i} \rightarrow v)$ channel while the maximum singular value is associated predominantly with the $(e_{a_r} - e_{a_i} \rightarrow \omega)$ channel.

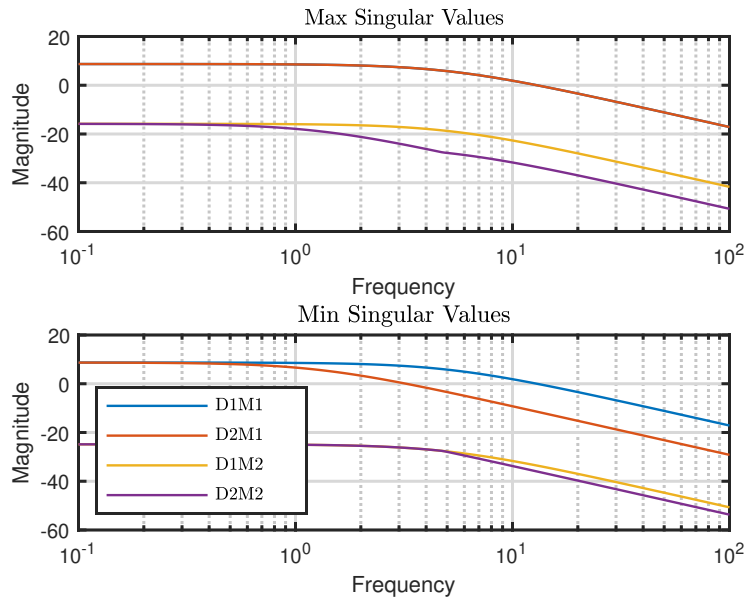


Figure 5.4: Plant Singular Values D1 & D2

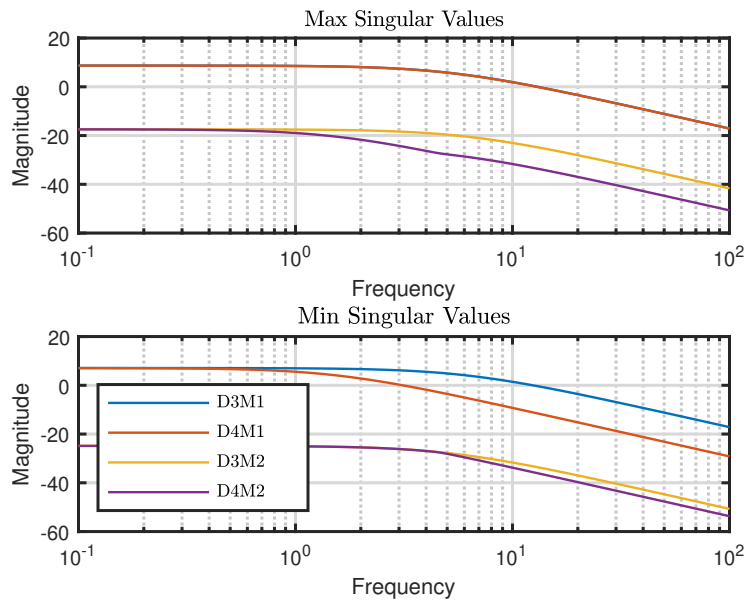


Figure 5.5: Plant Singular Values D3 & D4

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.

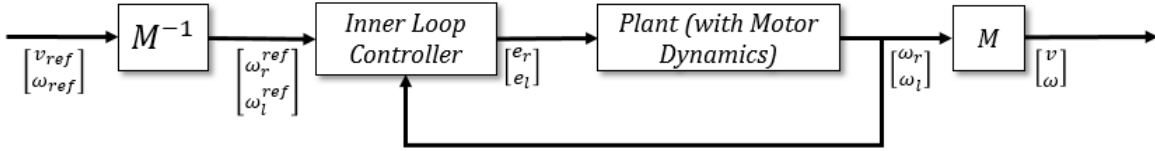


Figure 5.6: $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ System Inner-Loop Control Block Diagram

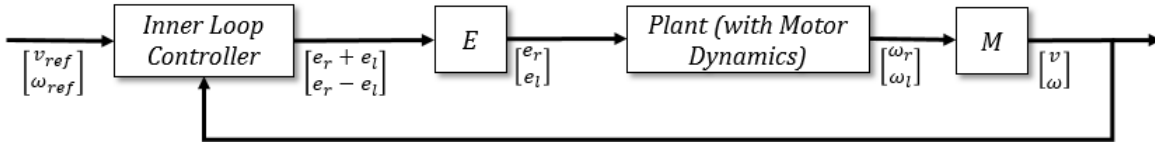


Figure 5.7: $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ System Inner-Loop Control Block Diagram

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:

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

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.

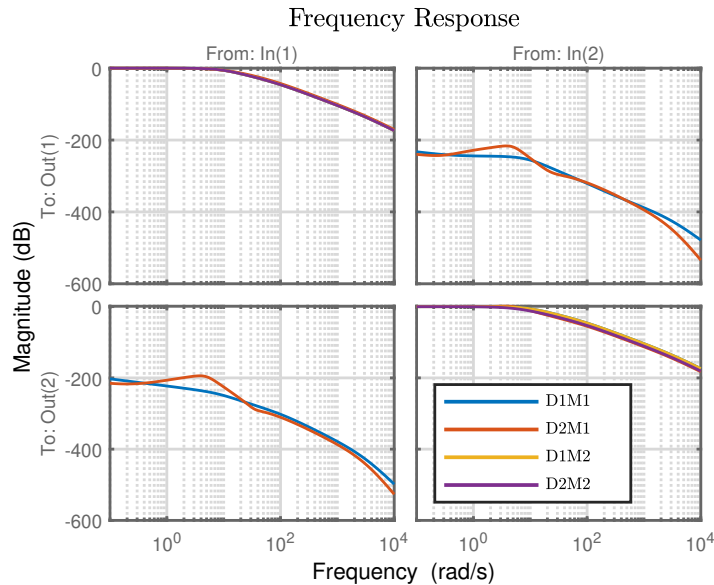


Figure 5.8: Inner-Loop Frequency Response T_{ry} : D1 & D2

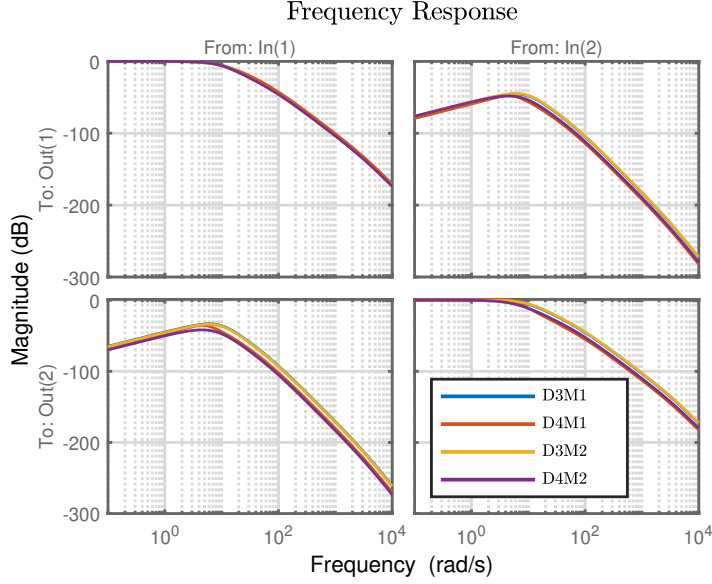


Figure 5.9: Inner-Loop Frequency Response T_{ry} : D3 & D4

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.

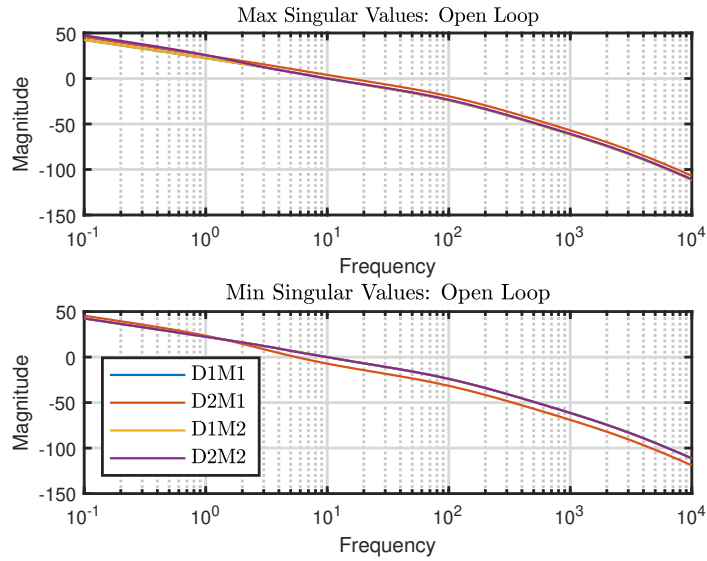


Figure 5.10: Open Loop Singular Values: D1 & D2

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.

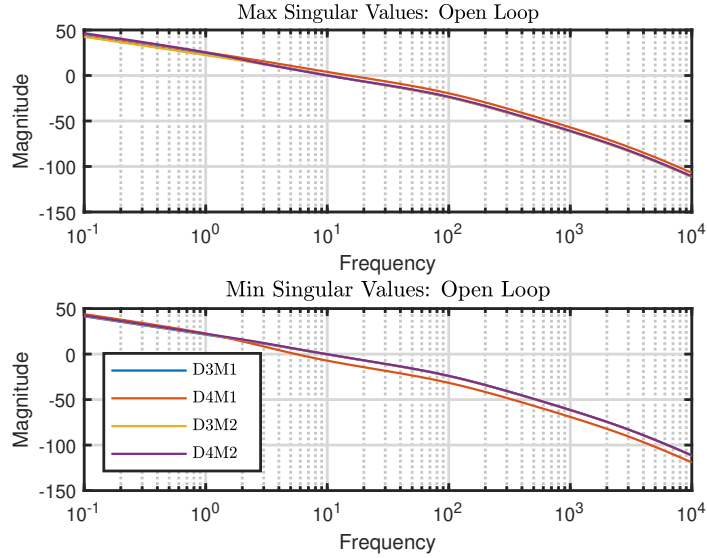


Figure 5.11: Open Loop Singular Values: D3 & D4

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.

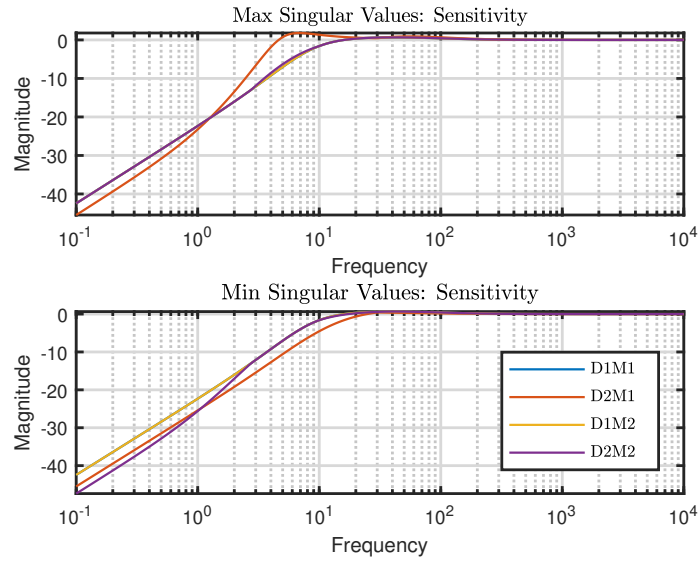


Figure 5.12: Sensitivity Singular Values: D1 & D2

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.

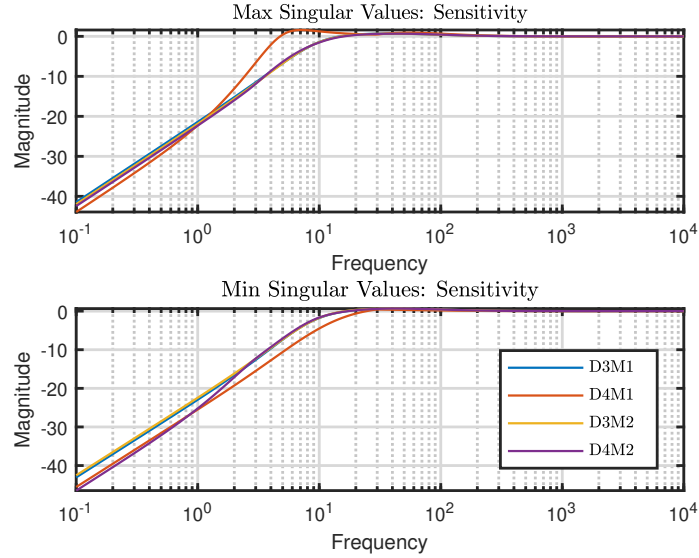


Figure 5.13: Sensitivity Singular Values: D3 & D4

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 $(e_{a_r}, e_{a_i}) \rightarrow (\omega_r, \omega_l)$ models, whereas they would differ for the $(e_{a_r} + e_{a_i}, e_{a_r} - e_{a_i}) \rightarrow (v, \omega)$ 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.

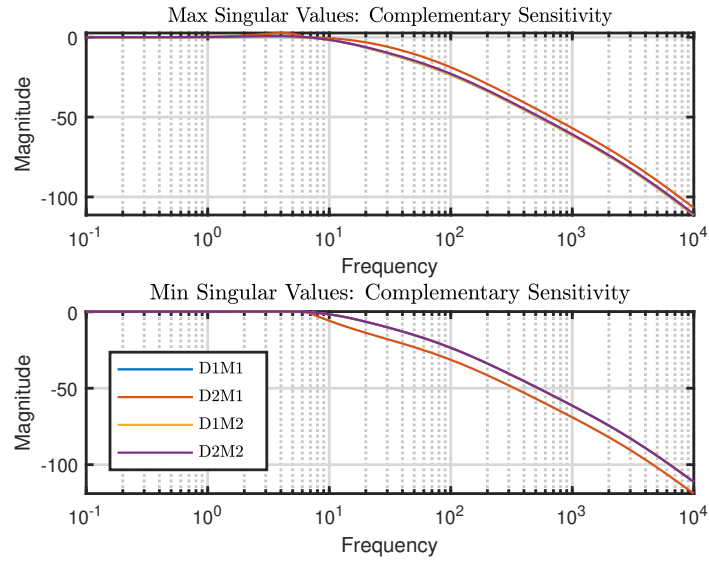


Figure 5.14: Complementary Sensitivity Singular Values: D1 & D2

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.

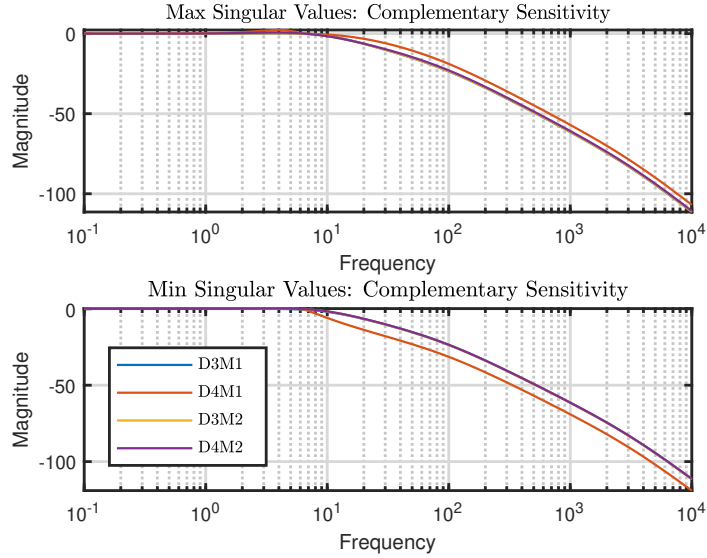


Figure 5.15: Complementary Sensitivity Singular Values: D3 & D4

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

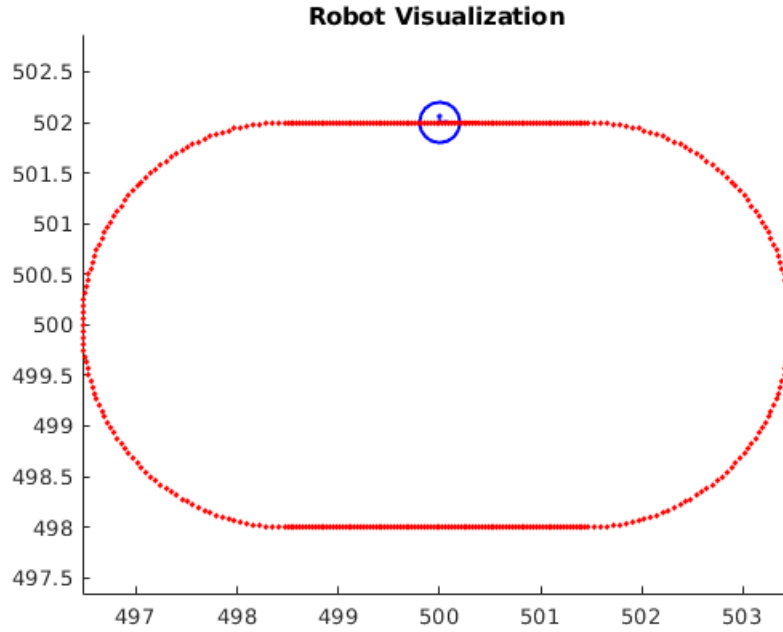


Figure 5.16: Reference Trajectory Visualization

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 $P_{[e_{a_r}, e_{a_l}] \rightarrow [\omega_r, \omega_l]}$ and $P_{[e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}] \rightarrow [v, \omega]}$ plants in the inner-loop. Here, 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.

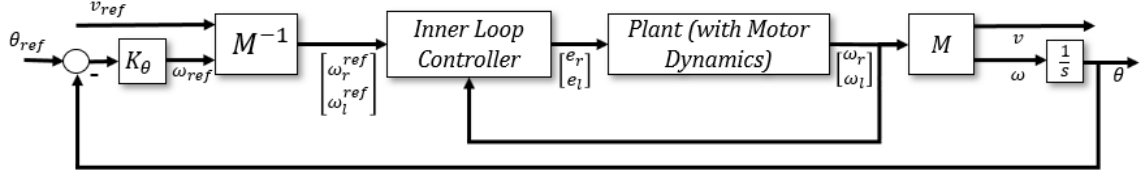


Figure 5.17: $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ System Outer-Loop Cruise Control Block Diagram

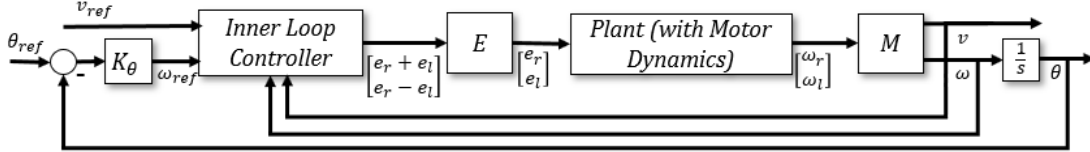


Figure 5.18: $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ System Outer-Loop Cruise Control Block Diagram

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

$$\dot{e}_\theta = -\omega \quad (5.12)$$

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

$$\omega = -k_\theta e_\theta \quad (5.13)$$

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.

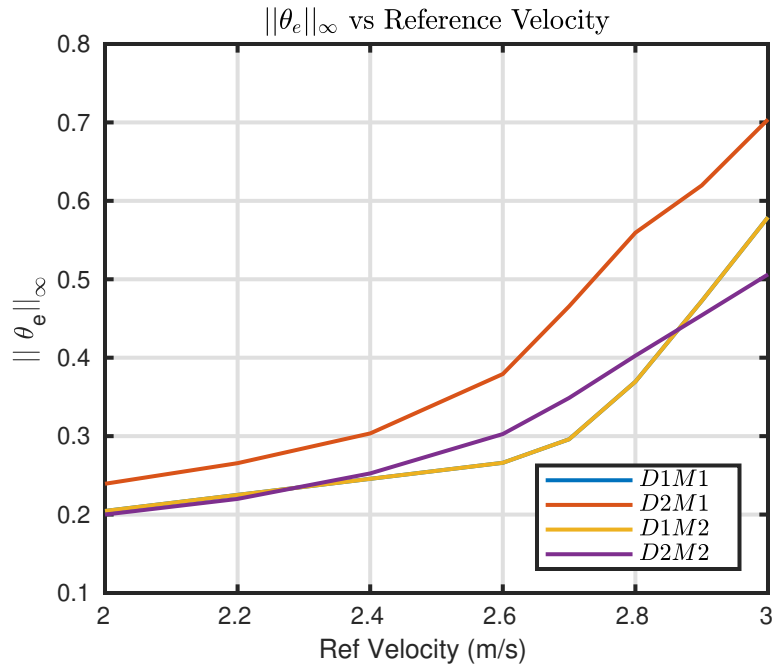


Figure 5.19: $\|\theta_e\|_\infty$ vs Reference Velocity: Simulation Results

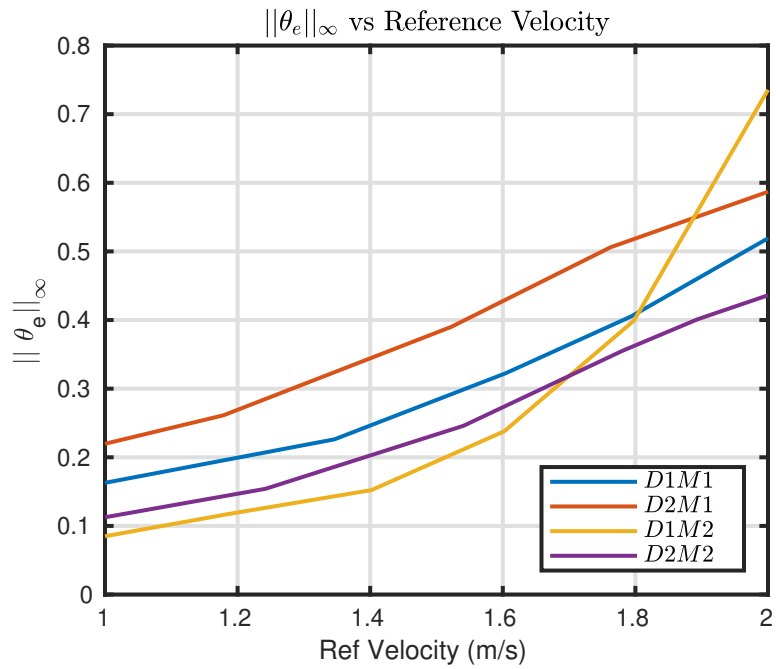


Figure 5.20: $\|\theta_e\|_\infty$ vs Reference Velocity: Hardware Results

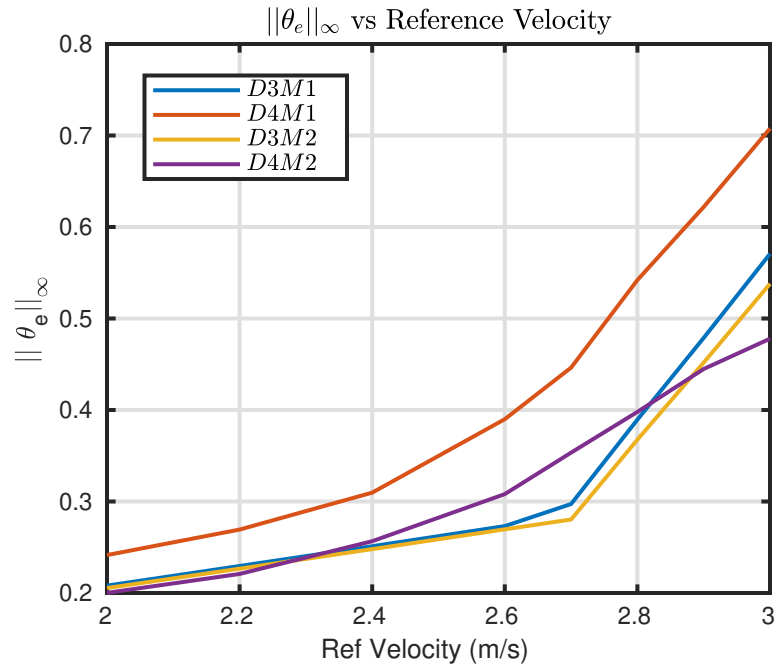


Figure 5.21: $\|\theta_e\|_\infty$ vs Reference Velocity: Simulation Results

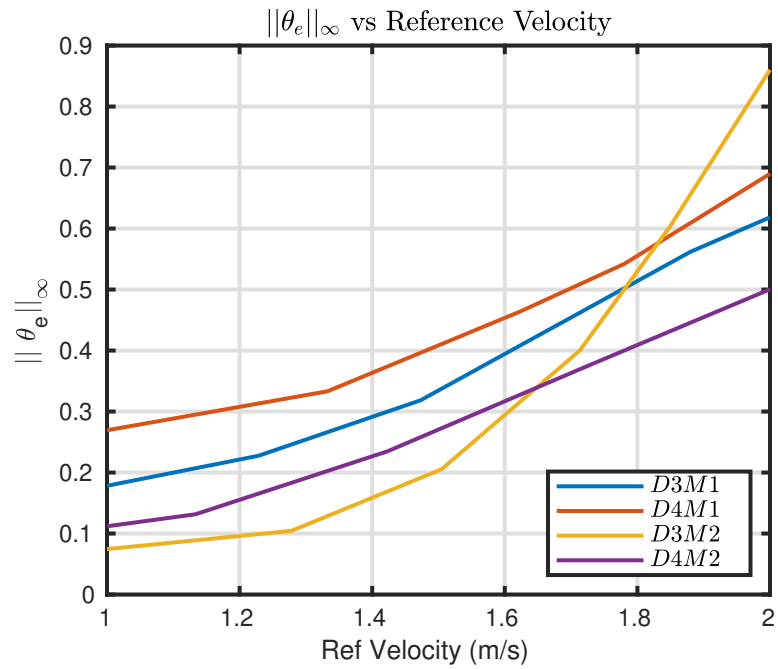


Figure 5.22: $\|\theta_e\|_\infty$ vs Reference Velocity: Hardware Results

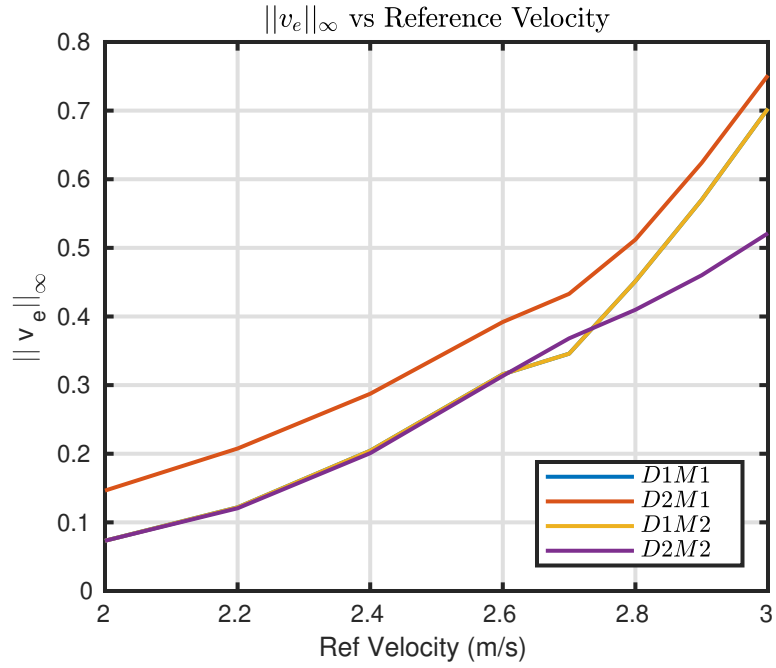


Figure 5.23: $\|v_e\|_\infty$ vs Reference Velocity: Simulation Results

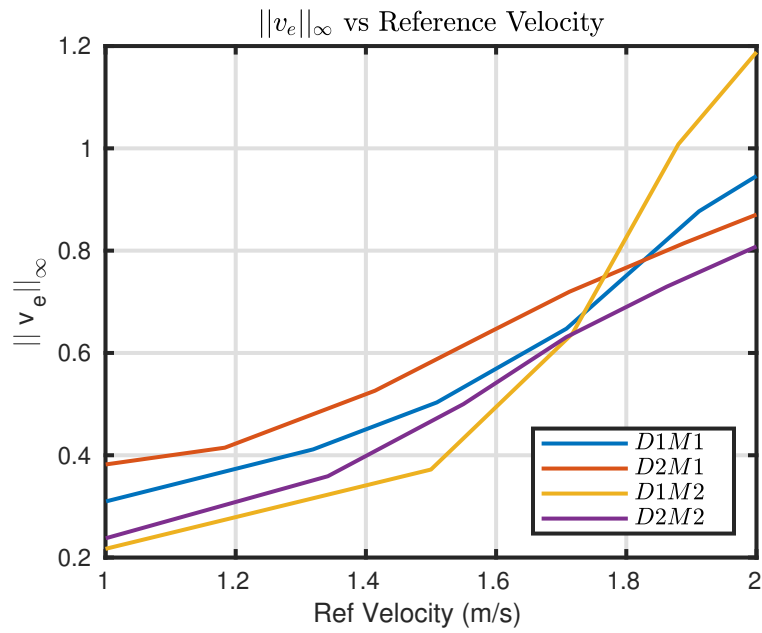


Figure 5.24: $\|v_e\|_\infty$ vs Reference Velocity: Hardware Results

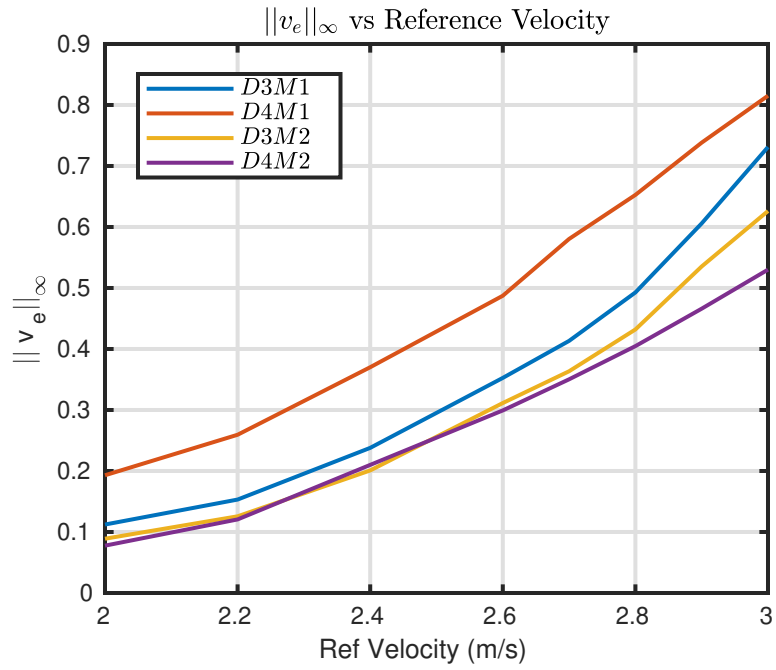


Figure 5.25: $\|v_e\|_\infty$ vs Reference Velocity: Simulation Results

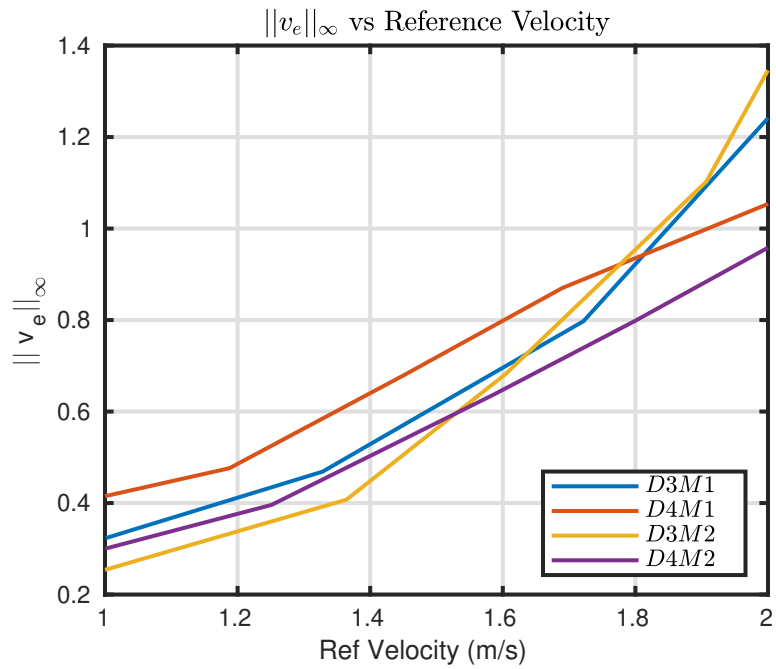


Figure 5.26: $\|v_e\|_\infty$ vs Reference Velocity: Hardware Results

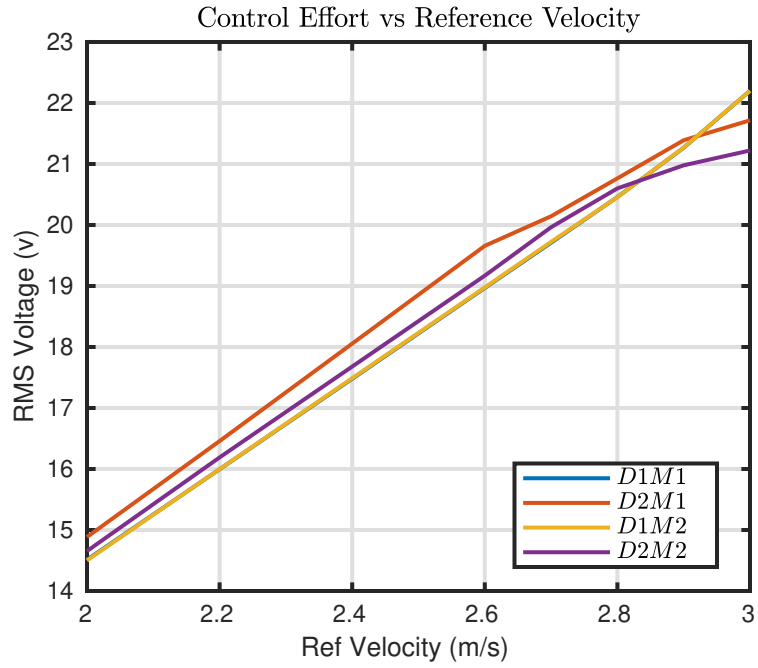


Figure 5.27: Control Effort vs Reference Velocity: Simulation Results

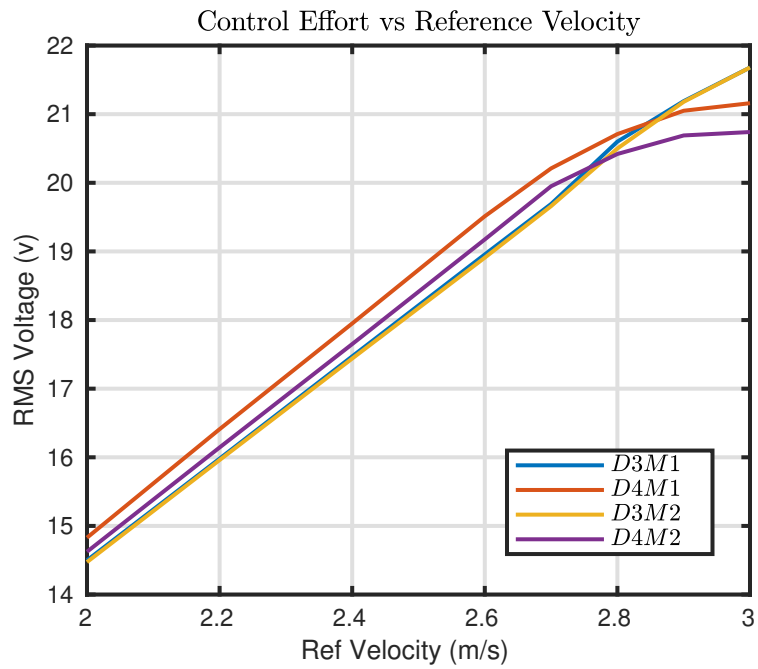


Figure 5.28: Control Effort vs Reference Velocity: Simulation Results

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 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.

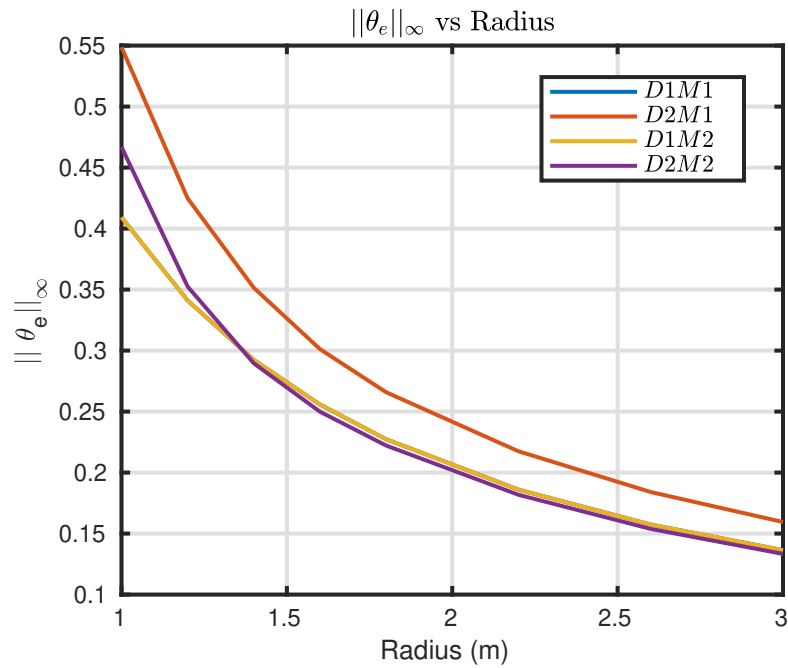


Figure 5.29: $\|\theta_e\|_\infty$ vs Radius of Track: Simulation Results

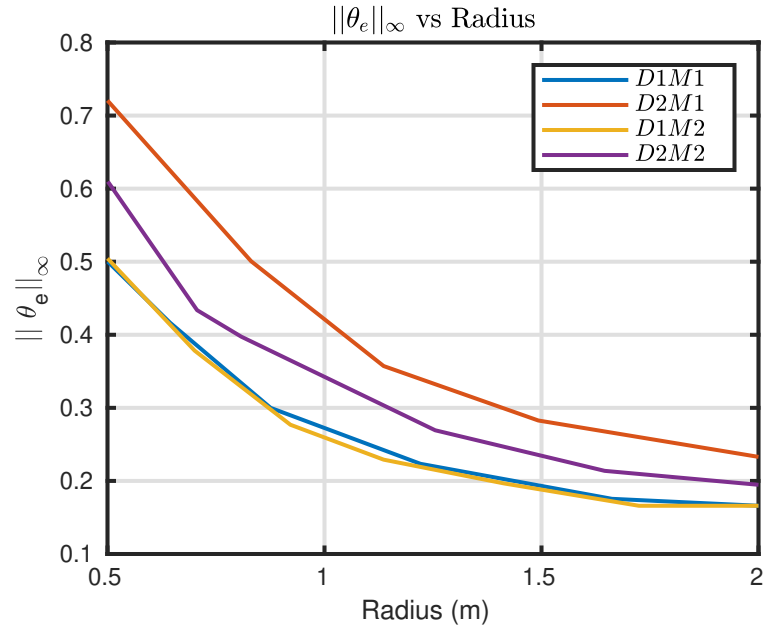


Figure 5.30: $\|\theta_e\|_\infty$ vs Radius of Track: Hardware Results

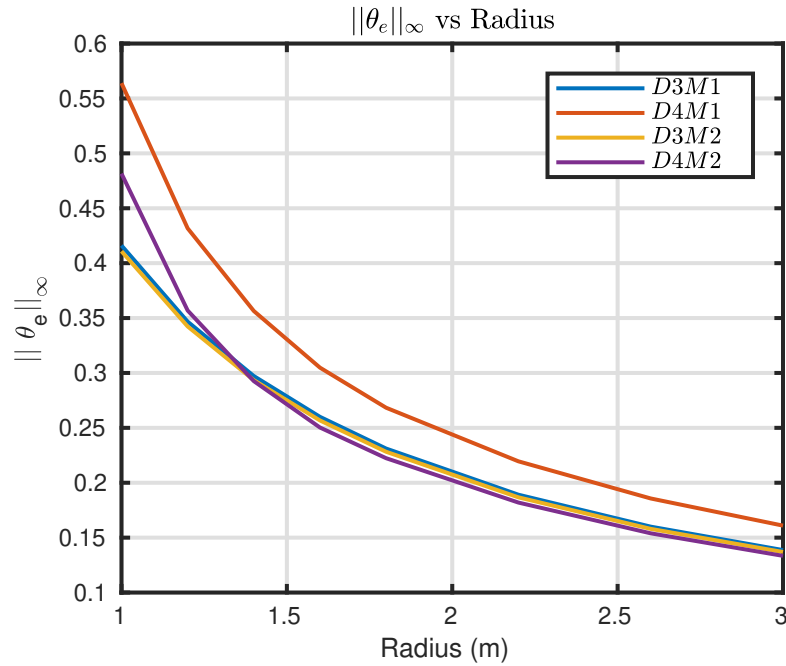


Figure 5.31: $\|\theta_e\|_\infty$ vs Radius of Track: Simulation Results

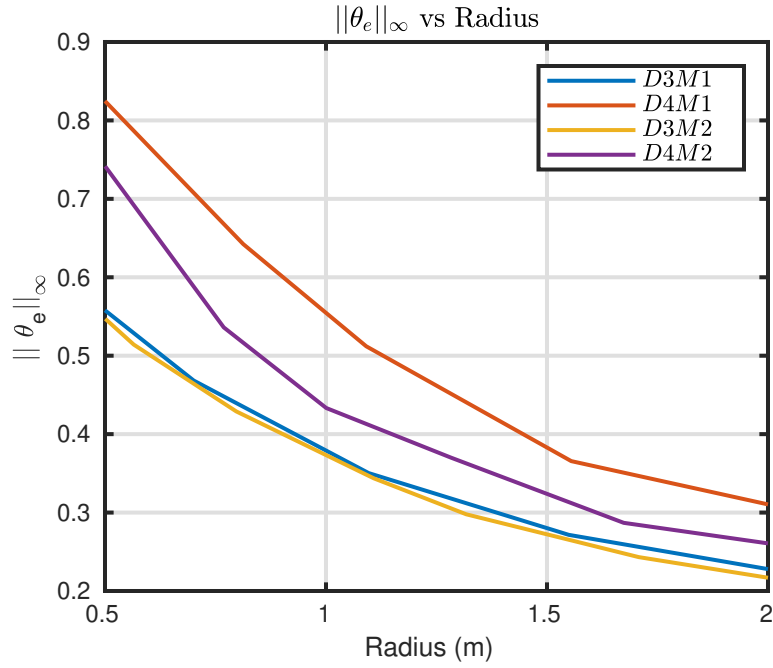


Figure 5.32: $\|\theta_e\|_\infty$ vs Radius of Track: Hardware Results

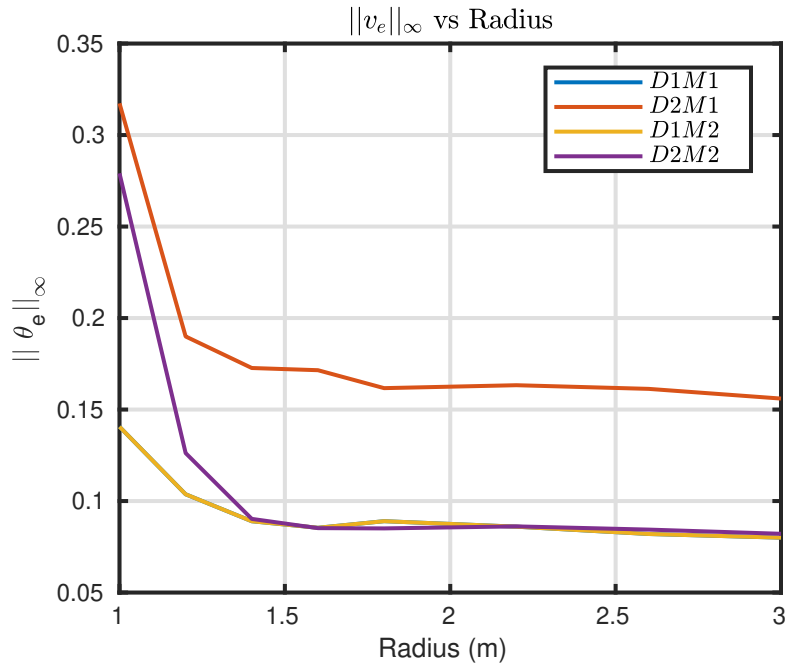


Figure 5.33: $\|v_e\|_\infty$ vs Radius of Track: Simulation Results

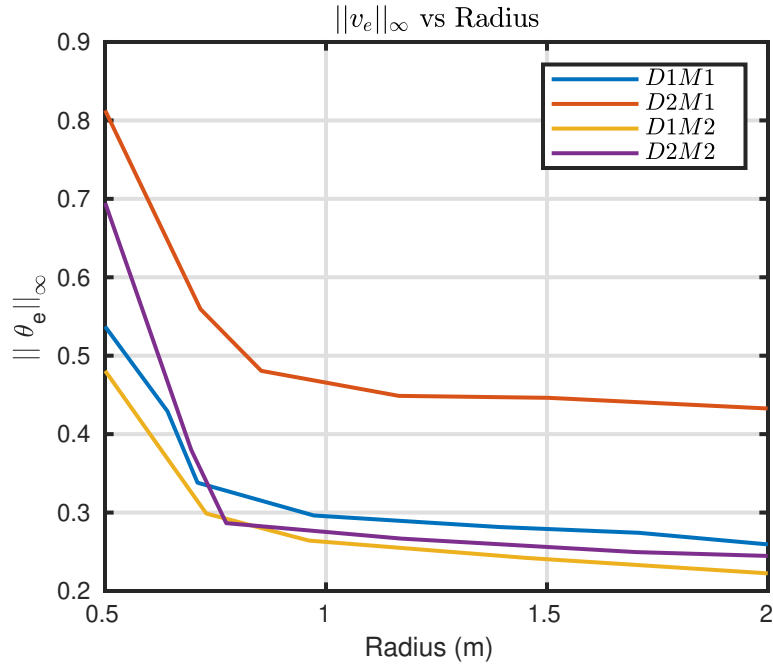


Figure 5.34: $\|v_e\|_\infty$ vs Radius of Track: Hardware Results

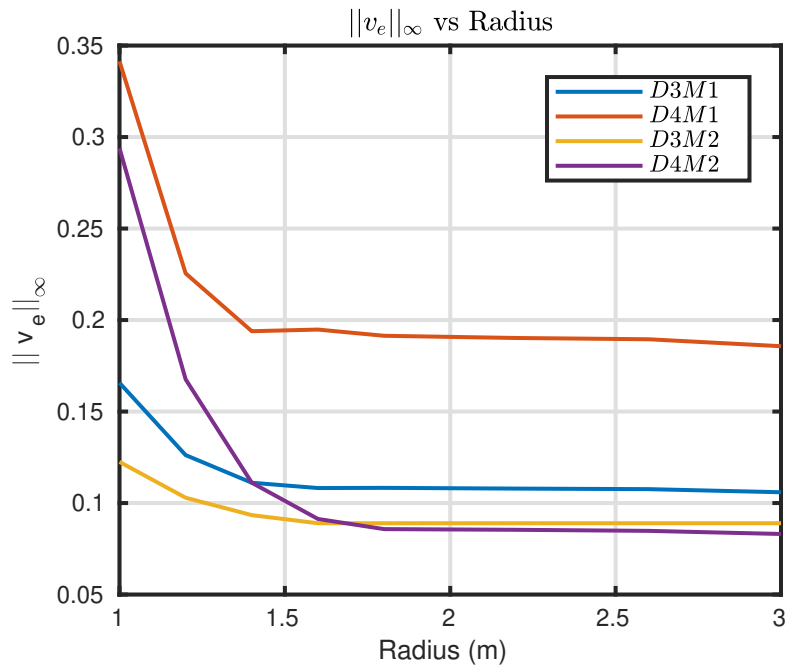


Figure 5.35: $\|v_e\|_\infty$ vs Radius of Track: Simulation Results

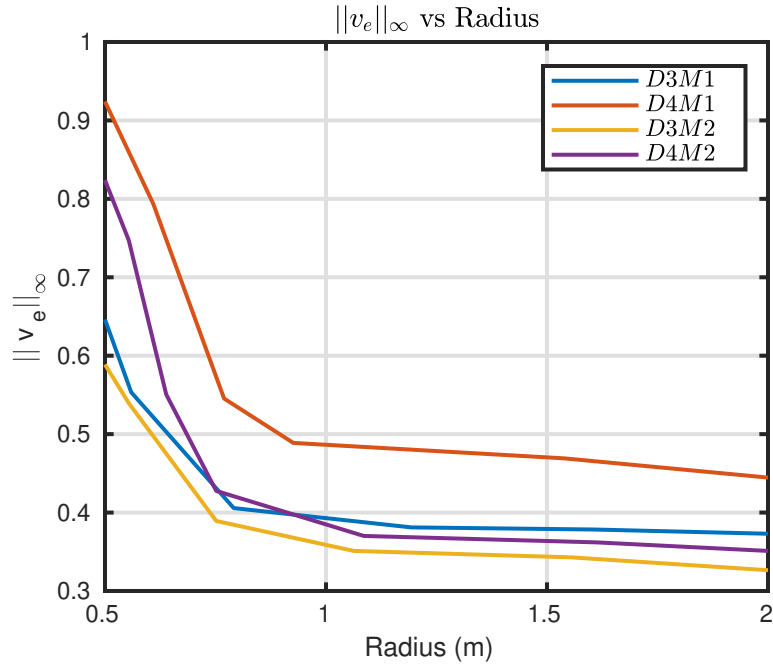


Figure 5.36: $\|v_e\|_\infty$ vs Radius of Track: Hardware Results

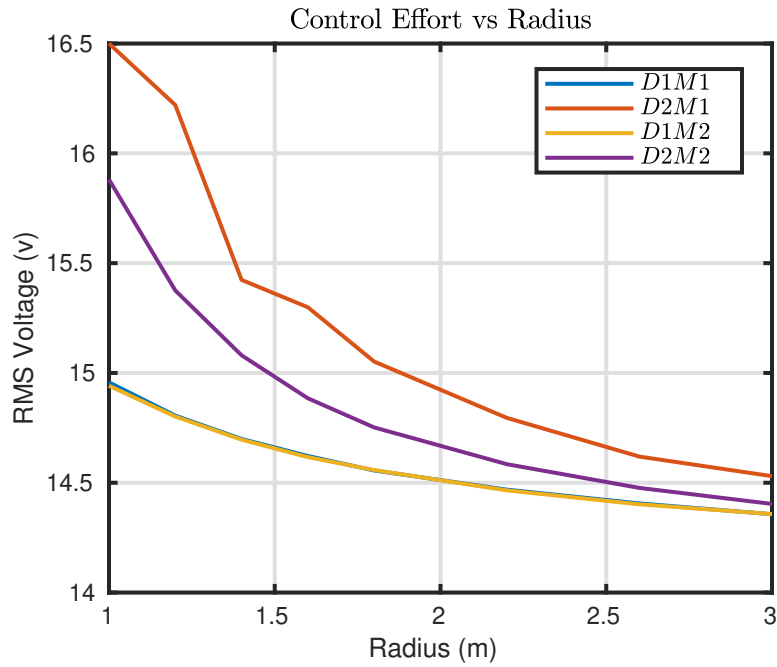


Figure 5.37: Control Effort vs Radius of Track: Simulation Results

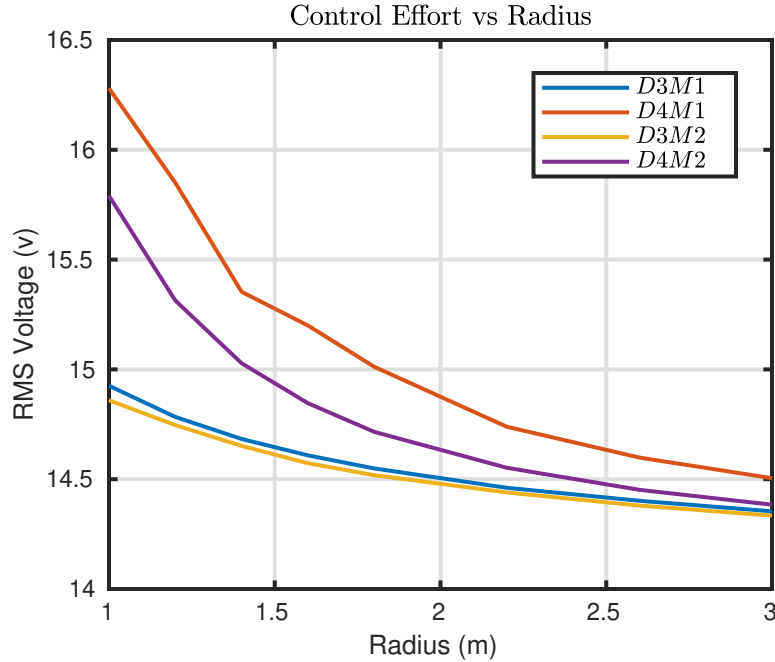


Figure 5.38: Control Effort vs Radius of Track: Simulation Results

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 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.

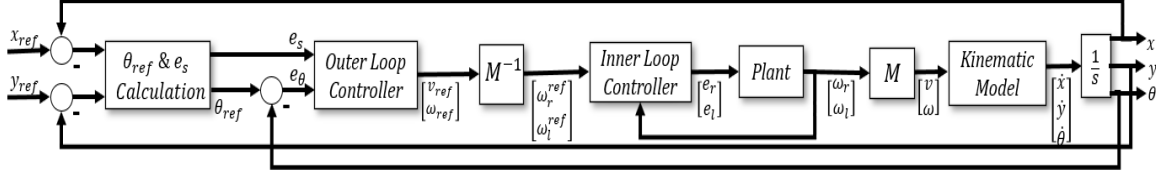


Figure 5.39: $(e_{a_r}, e_{a_l}) \rightarrow (\omega_r, \omega_l)$ System Outer-Loop Control Block Diagram

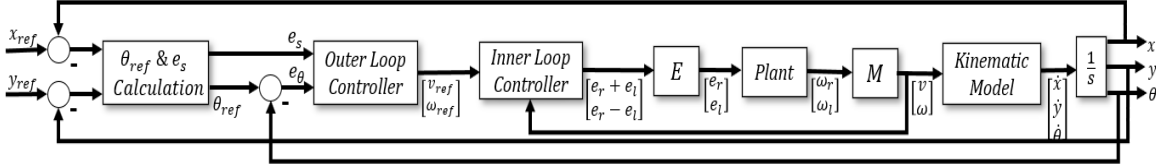


Figure 5.40: $(e_{a_r} + e_{a_l}, e_{a_r} - e_{a_l}) \rightarrow (v, \omega)$ System Outer-Loop Control Block Diagram

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

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

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

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

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

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

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.

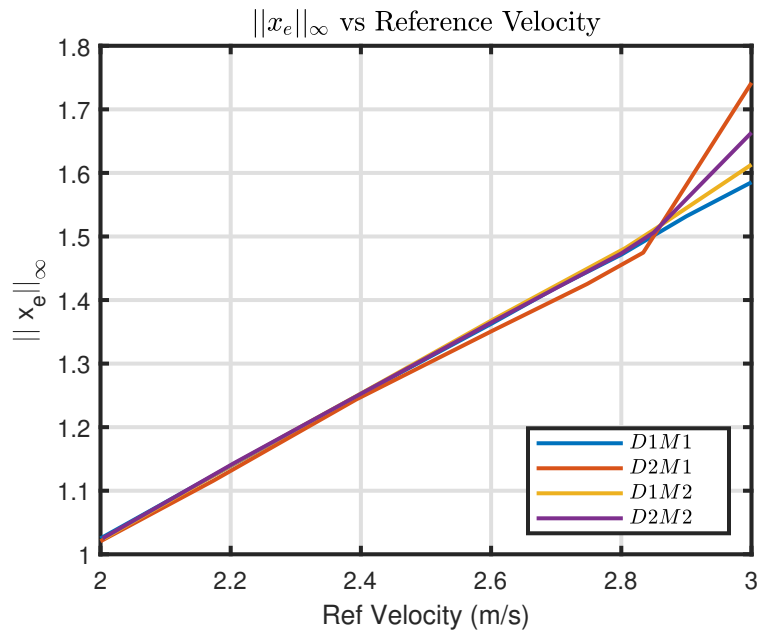


Figure 5.41: $\|x_e\|_\infty$ vs Reference Velocity: Simulation Results

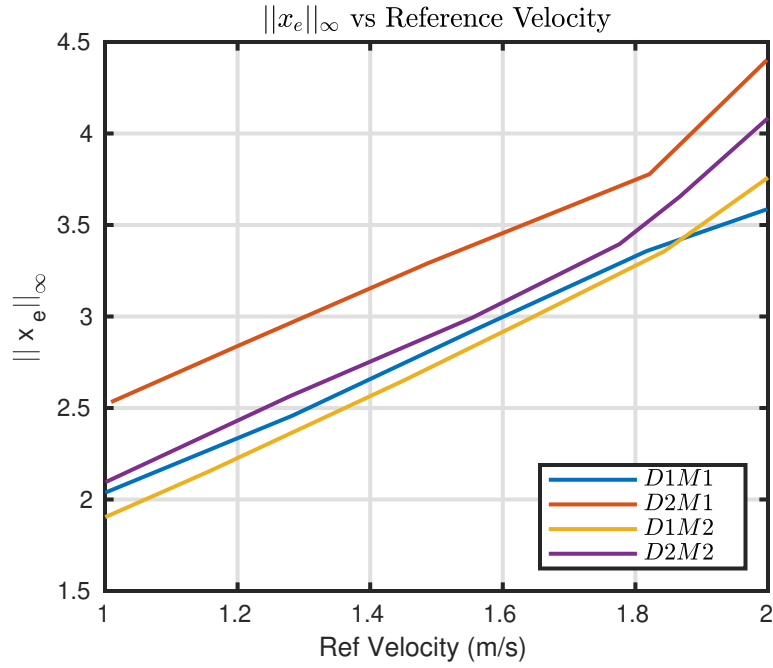


Figure 5.42: $\|x_e\|_\infty$ vs Reference Velocity: Hardware Results

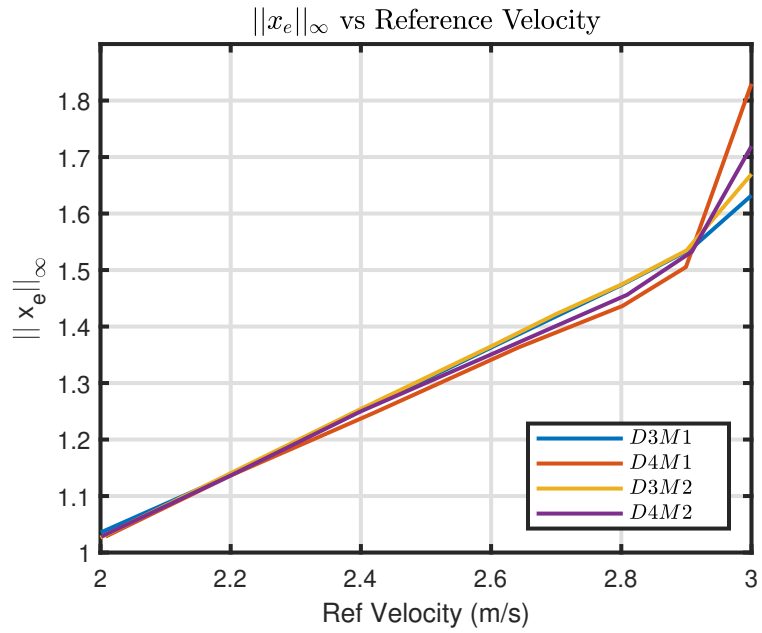


Figure 5.43: $\|x_e\|_\infty$ vs Reference Velocity: Simulation Results

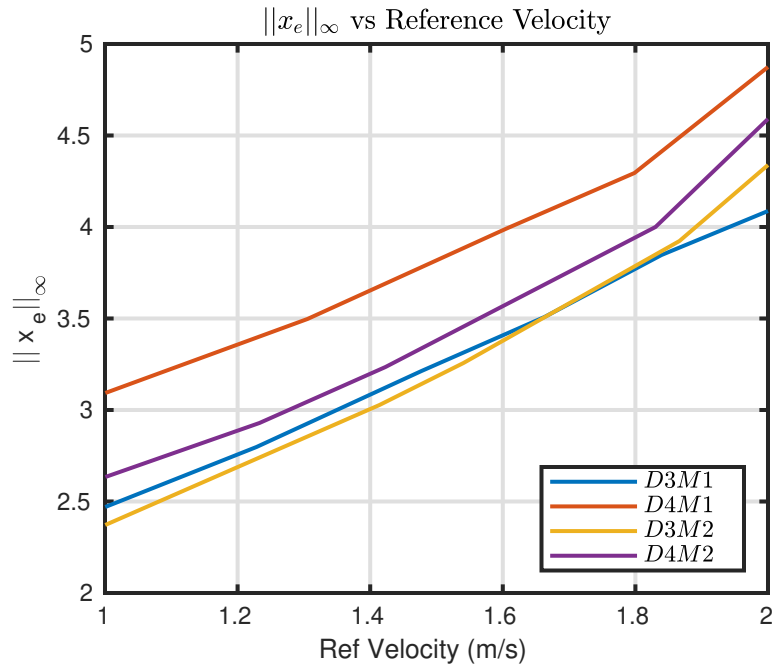


Figure 5.44: $\|x_e\|_\infty$ vs Reference Velocity: Hardware Results

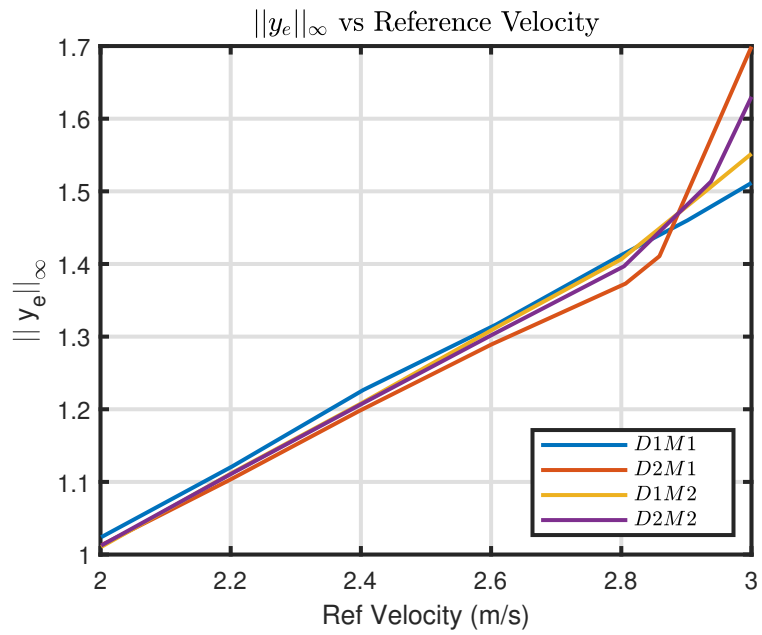


Figure 5.45: $\|y_e\|_\infty$ vs Reference Velocity: Simulation Results

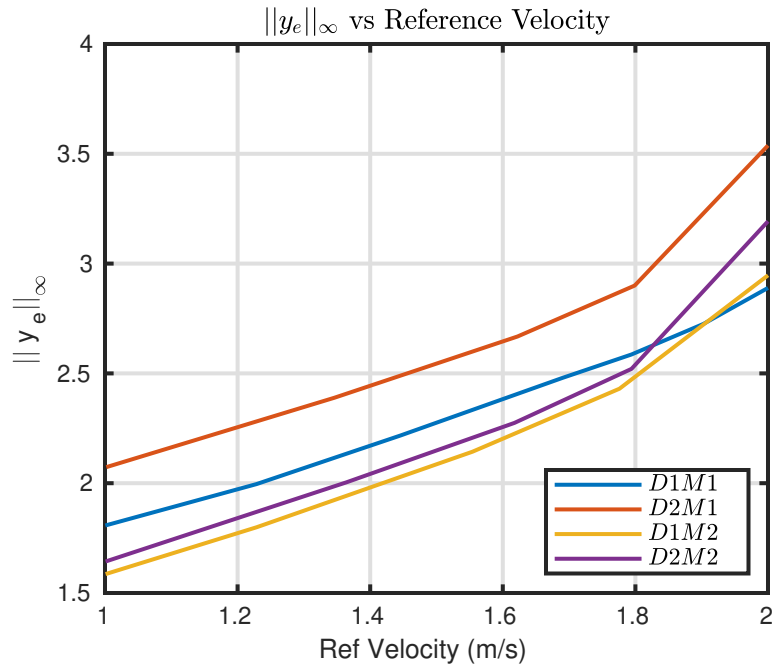


Figure 5.46: $\|y_e\|_\infty$ vs Reference Velocity: Hardware Results

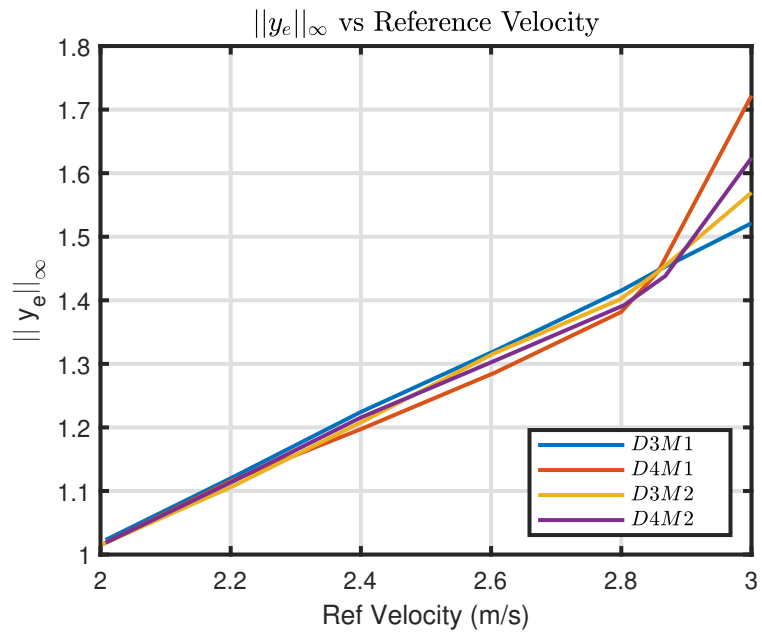


Figure 5.47: $\|y_e\|_\infty$ vs Reference Velocity: Simulation Results

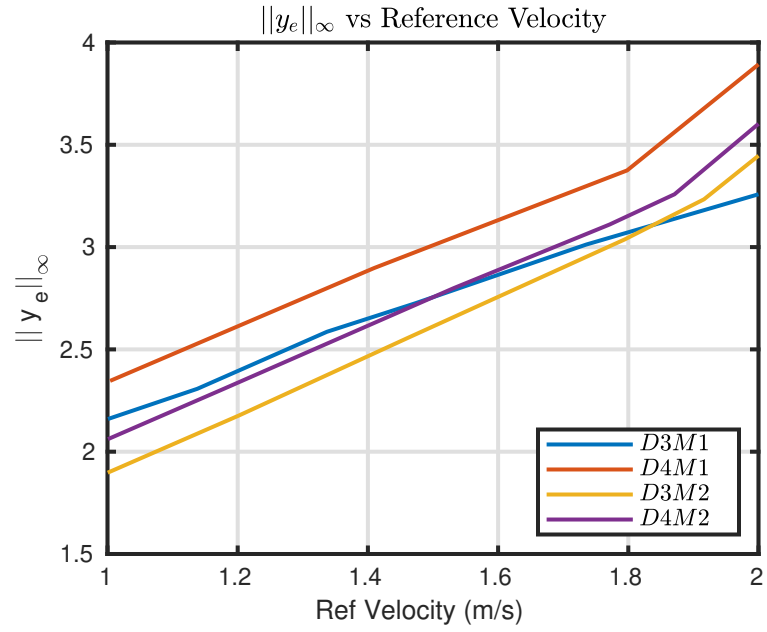


Figure 5.48: $\|y_e\|_\infty$ vs Reference Velocity: Hardware Results

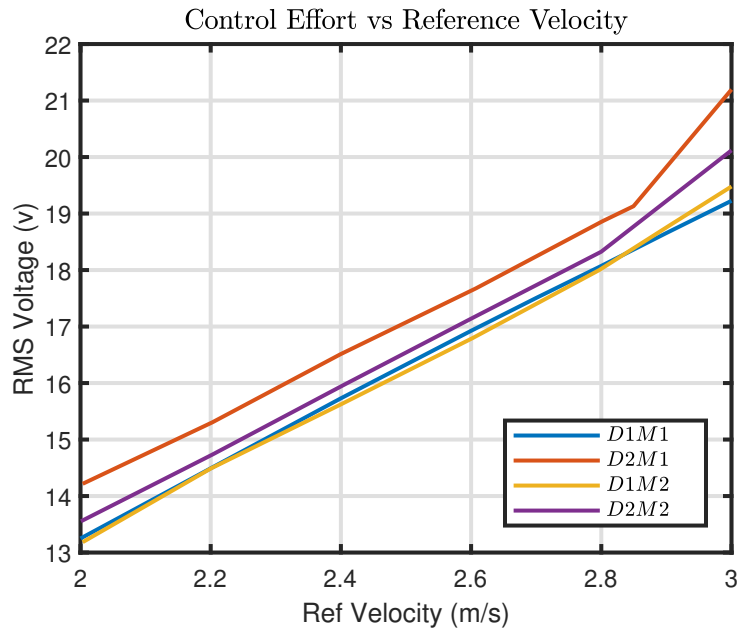


Figure 5.49: Control Effort vs Reference Velocity: Simulation Results

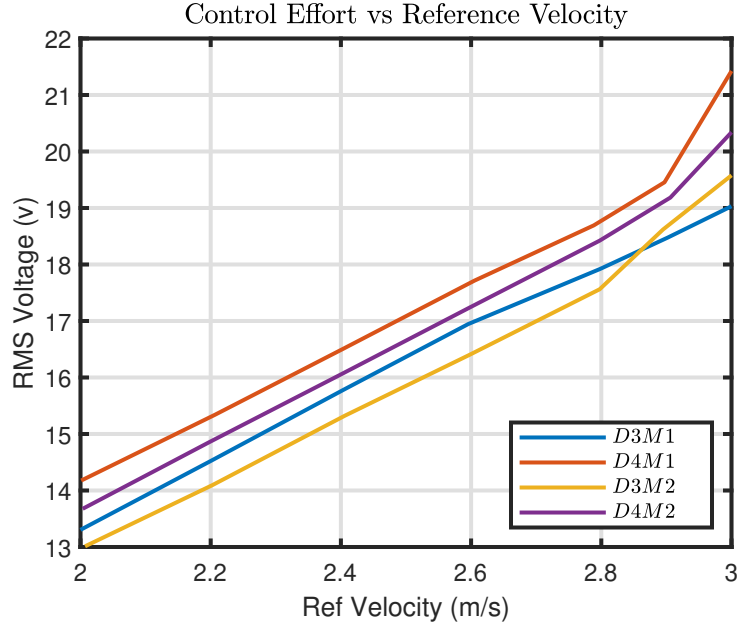


Figure 5.50: Control Effort vs Reference Velocity: Simulation Results

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 compared to 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.

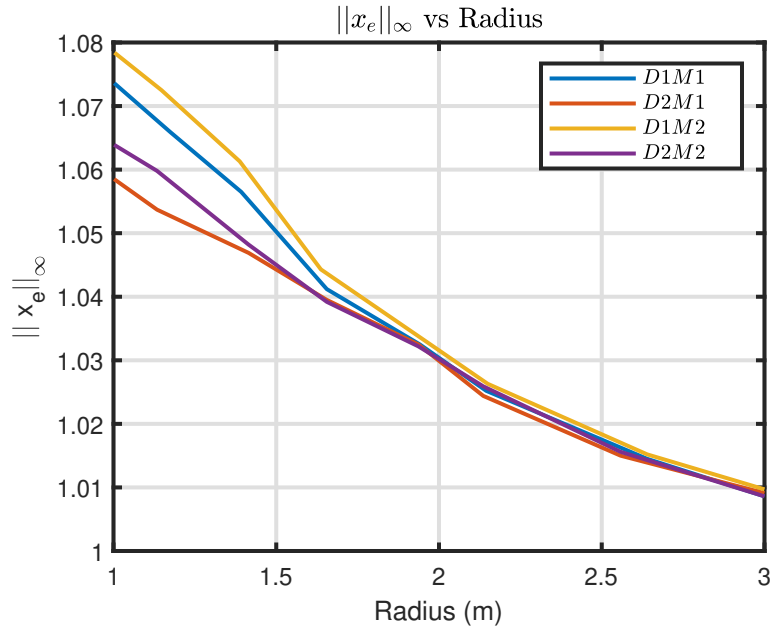


Figure 5.51: $\|x_e\|_\infty$ vs Radius of Track: Simulation Results

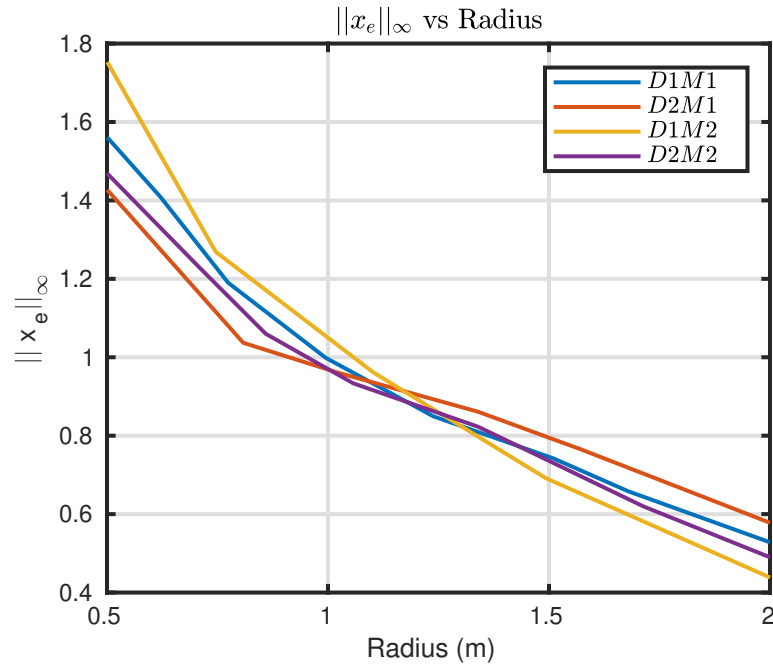


Figure 5.52: $\|x_e\|_\infty$ vs Radius of Track: Hardware Results

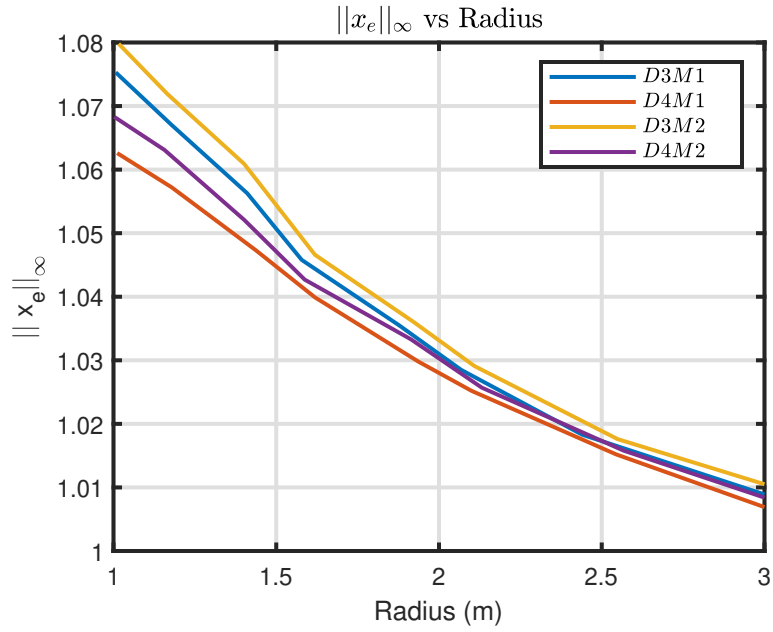


Figure 5.53: $\|x_e\|_\infty$ vs Radius of Track: Simulation Results

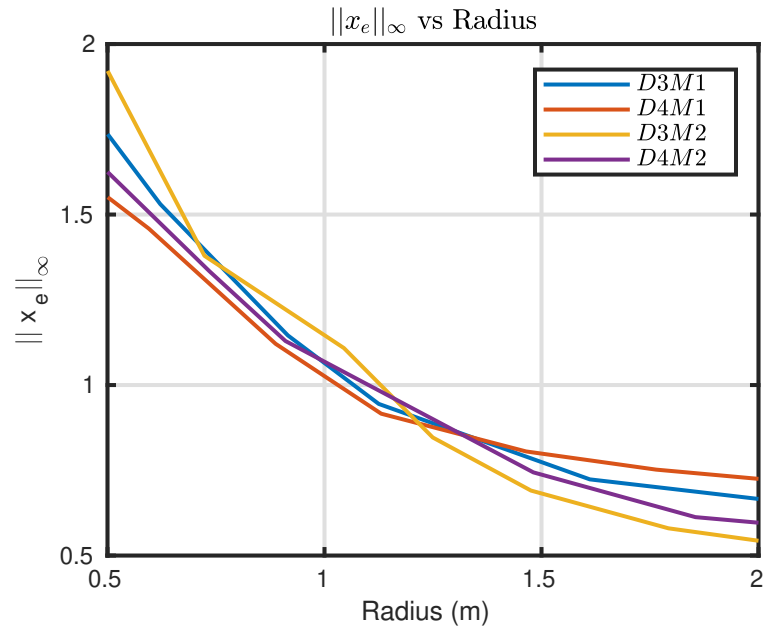


Figure 5.54: $\|x_e\|_\infty$ vs Radius of Track: Hardware Results

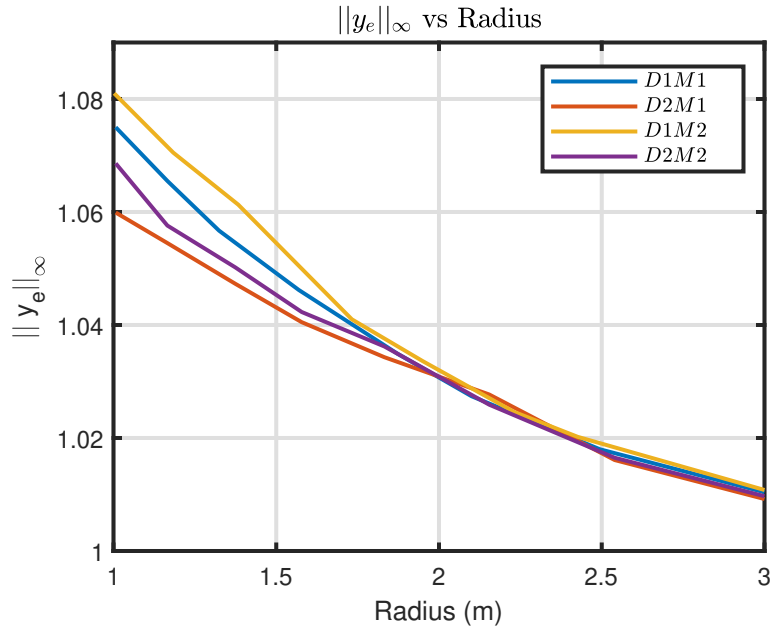


Figure 5.55: $\|y_e\|_\infty$ vs Radius of Track: Simulation Results

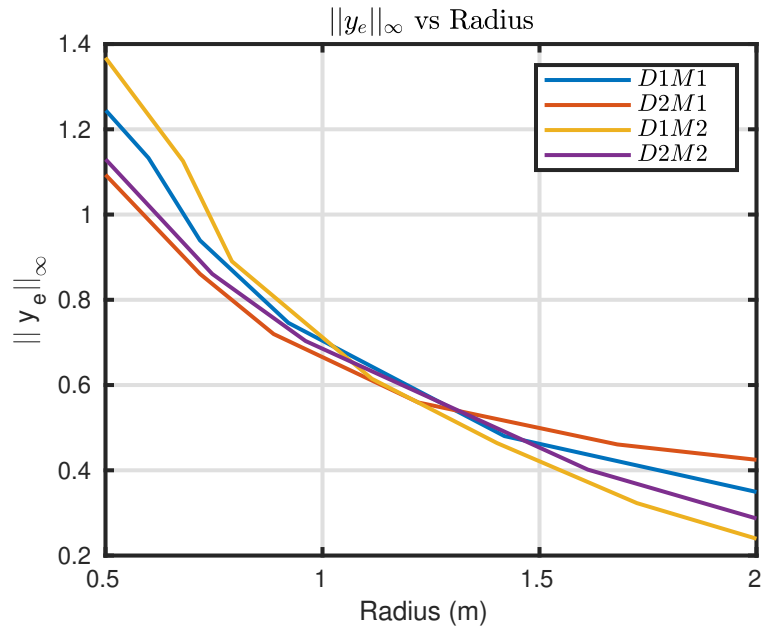


Figure 5.56: $\|y_e\|_\infty$ vs Radius of Track: Hardware Results

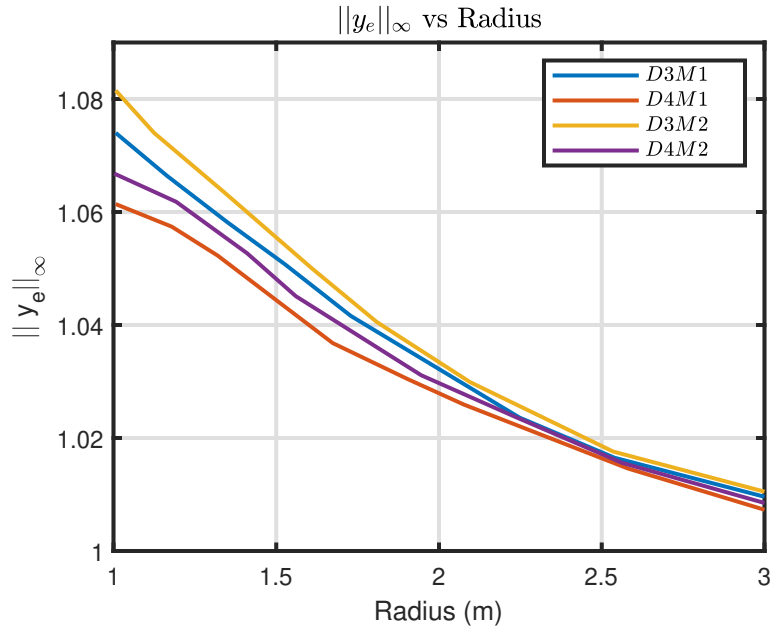


Figure 5.57: $\|y_e\|_\infty$ vs Radius of Track: Simulation Results

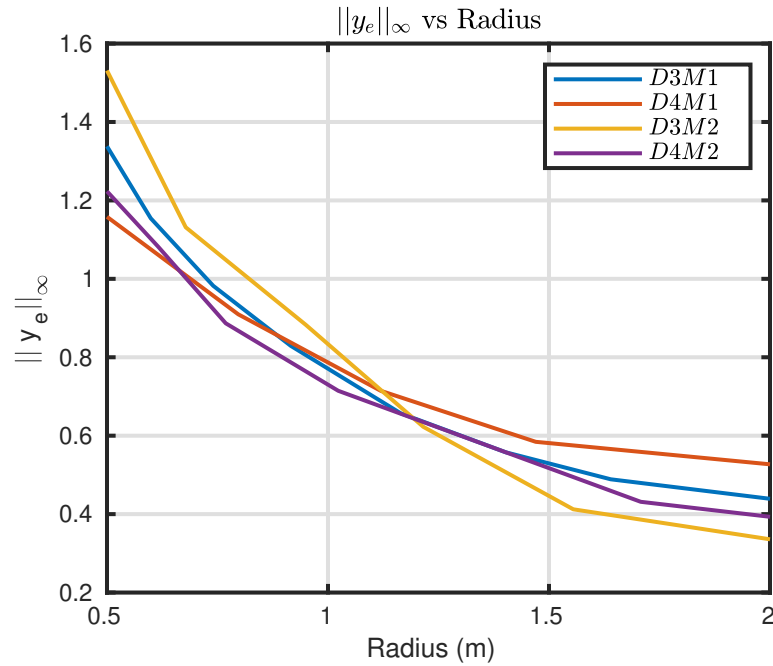


Figure 5.58: $\|y_e\|_\infty$ vs Radius of Track: Hardware Results

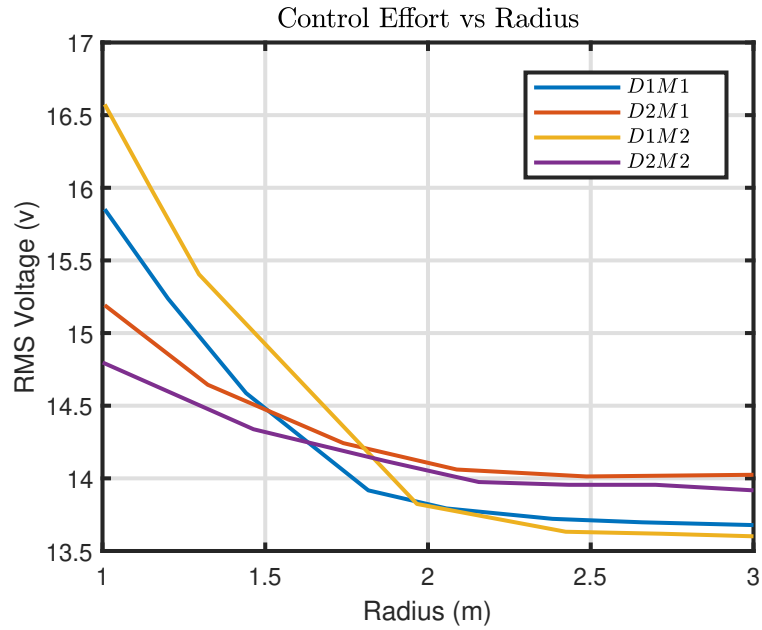


Figure 5.59: Control Effort vs Radius of Track: Simulation Results

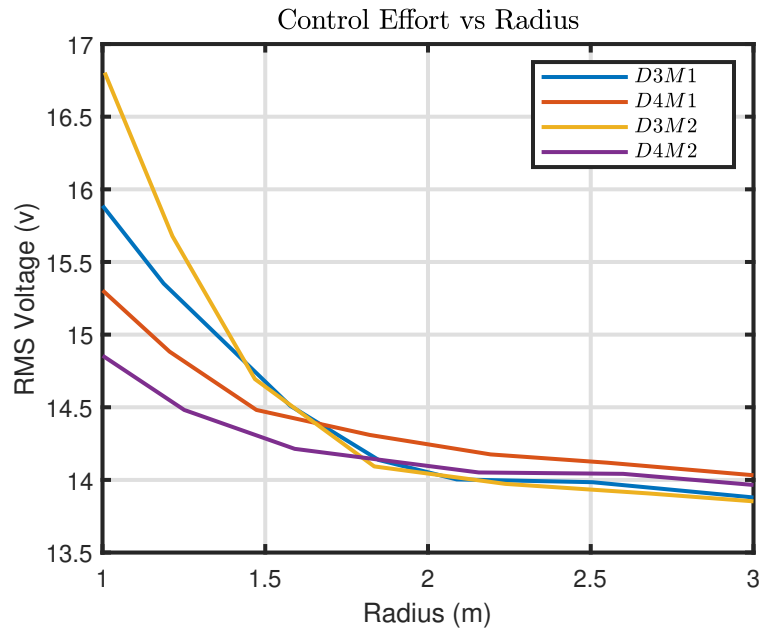


Figure 5.60: Control Effort vs Radius of Track: Simulation Results

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 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.

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

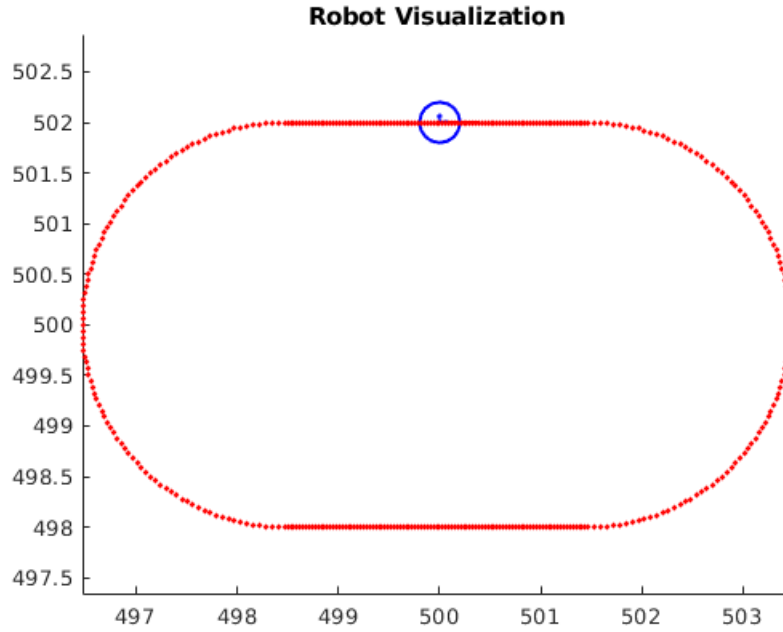


Figure 6.1: Reference Trajectory Visualization

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 $i \in \{2, 3, \dots, N\}$ 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

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

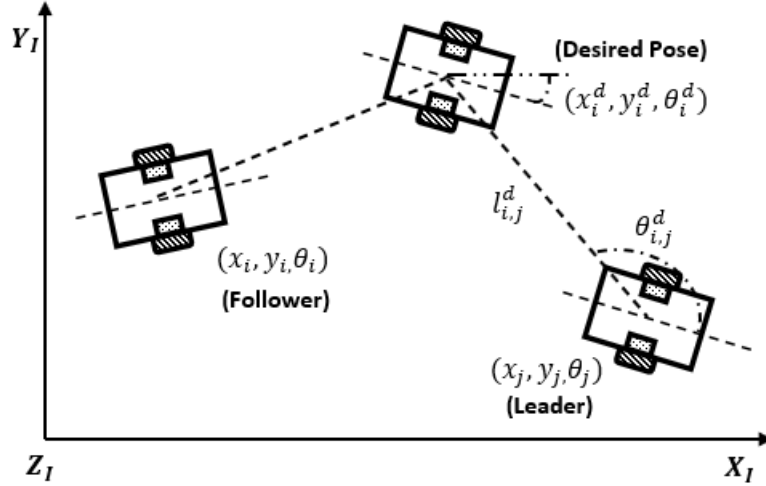


Figure 6.2: Multi-Robot Leader-Follower Formulation

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:

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

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

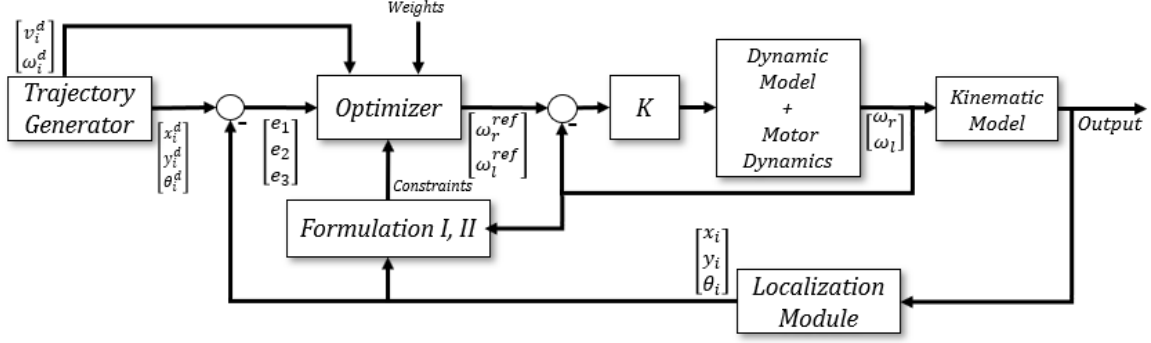


Figure 6.3: Closed-Loop Block Diagram Representation

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 width⁴ 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

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

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

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

For nominal plant parameters i.e. $d = 0$, the inner-loop control system can be approximated as a first order transfer function $[\omega_{r,i}, \omega_{l,i}] = \text{diag}\left(\frac{B_i}{s+B_i}, \frac{B_i}{s+B_i}\right)[\omega_{r,i}^{ref}, \omega_{l,i}^{ref}]$, 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

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

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

⁴the 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) .

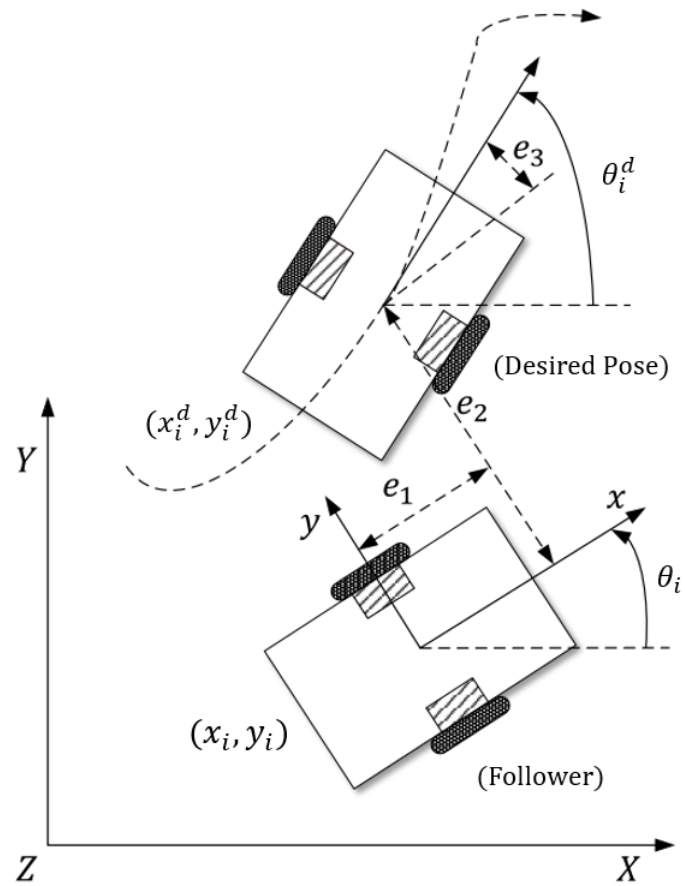


Figure 6.4: Desired and Actual Pose - Error Representation

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

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

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

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

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

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:

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

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

$$\dot{e}_3 = -\frac{r}{d_w} \omega_{r,i} + \frac{r}{d_w} \omega_{l,i} \quad (6.15)$$

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).

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

$$\text{subject to} \quad (6.17)$$

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.

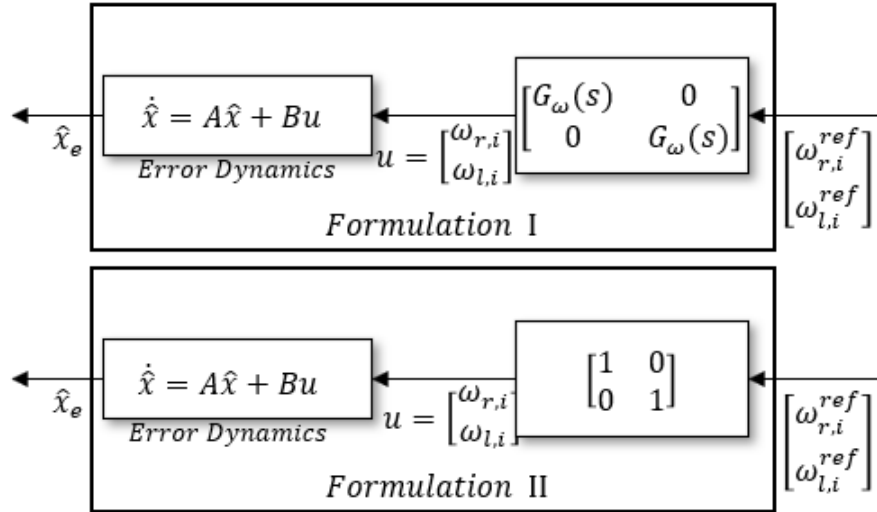


Figure 6.5: Formulation I and II - Block Diagram Representation

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 tuning the controller for the desired tracking performance, the final parameters are as follow:

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

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.

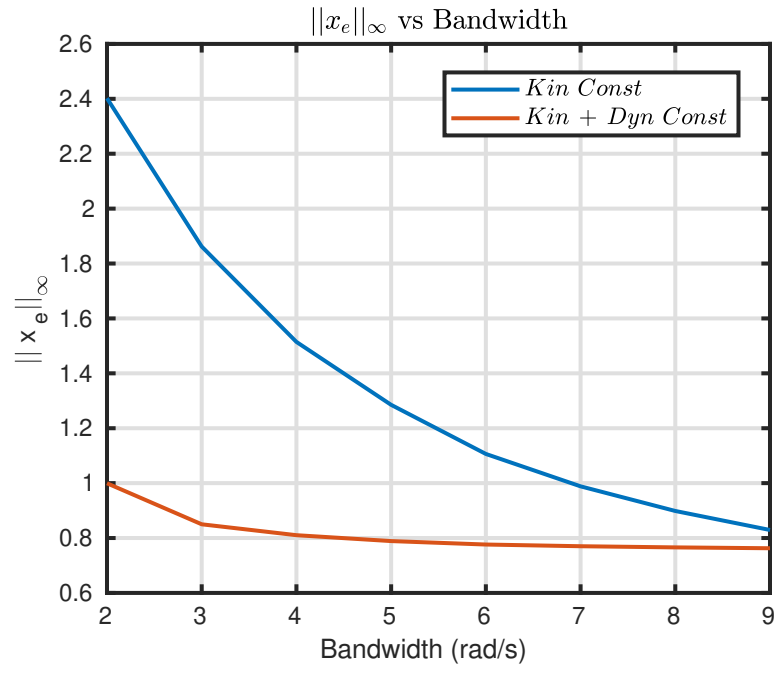


Figure 6.6: $\|x_e\|_\infty$ vs Inner-Loop Bandwidth: Simulation Results

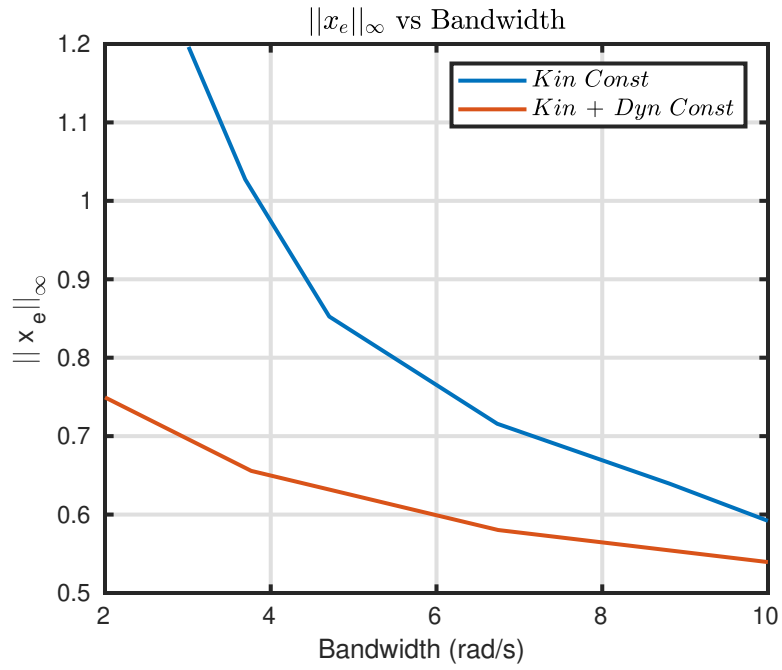


Figure 6.7: $\|x_e\|_\infty$ vs Inner-Loop Bandwidth: Hardware Results

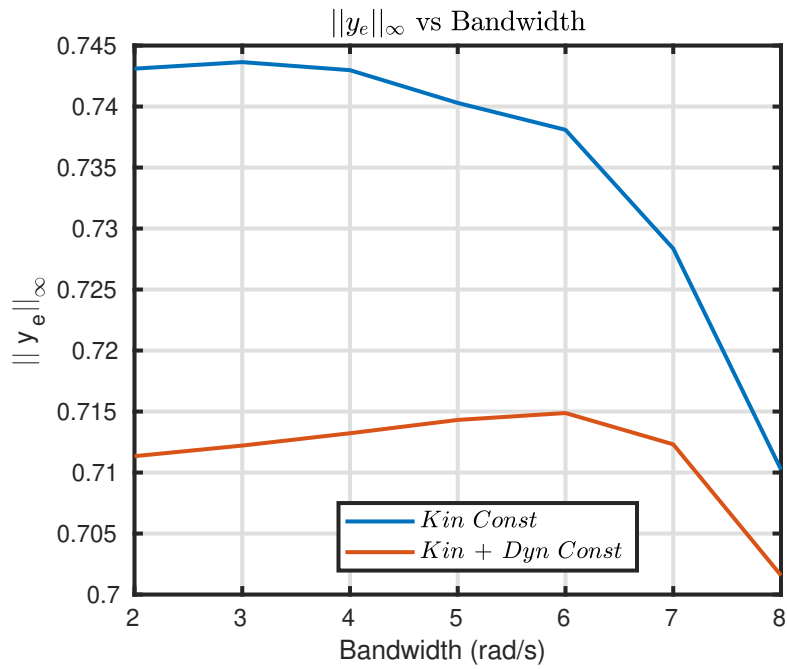


Figure 6.8: $\|y_e\|_\infty$ vs Inner-Loop Bandwidth: Simulation Results

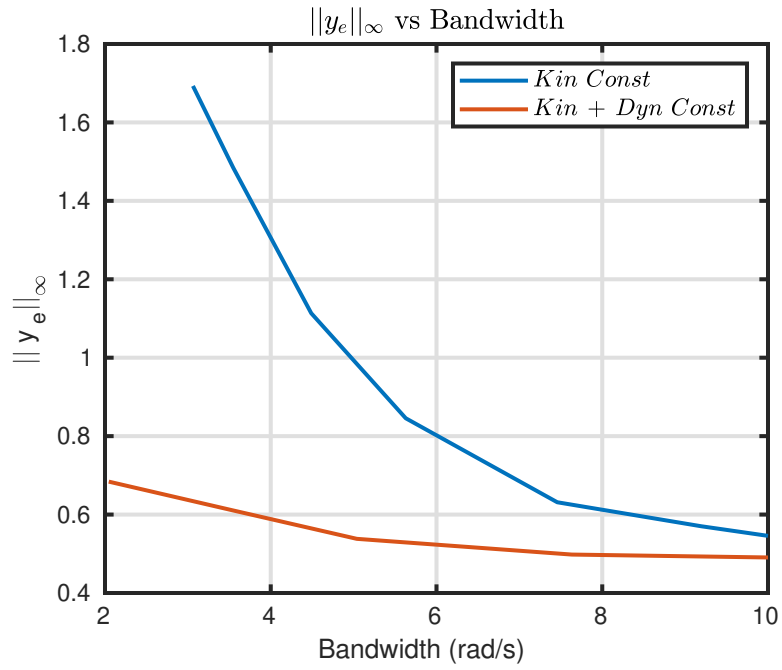


Figure 6.9: $\|y_e\|_\infty$ vs Inner-Loop Bandwidth: Hardware Results

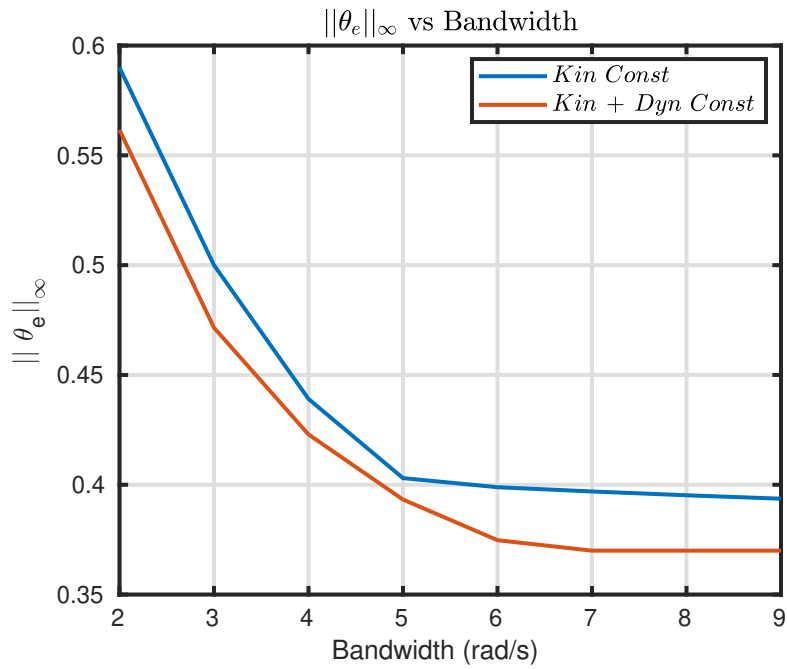


Figure 6.10: $\|\theta_e\|_\infty$ vs Inner-Loop Bandwidth: Simulation Results

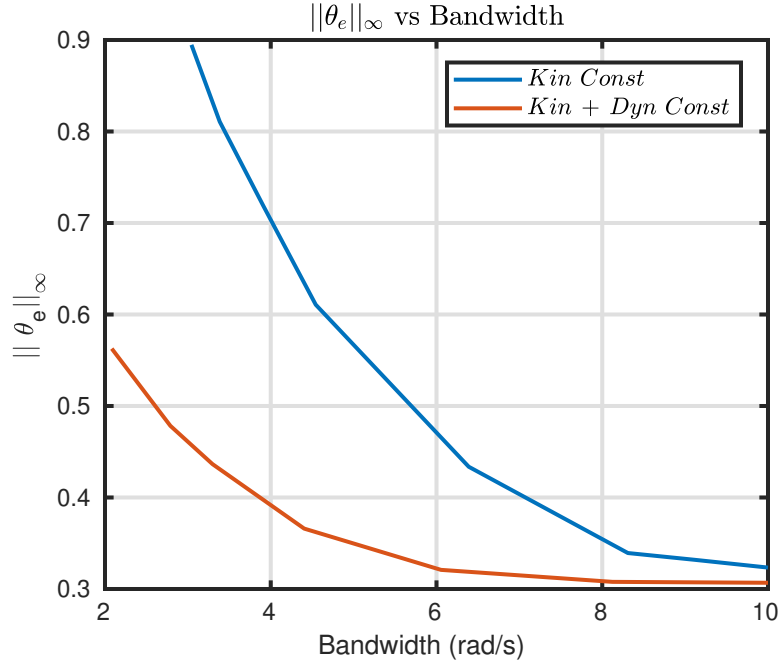


Figure 6.11: $\|\theta_e\|_\infty$ vs Inner-Loop Bandwidth: Hardware Results

Varying Inner-Loop Bandwidth B_i . 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.

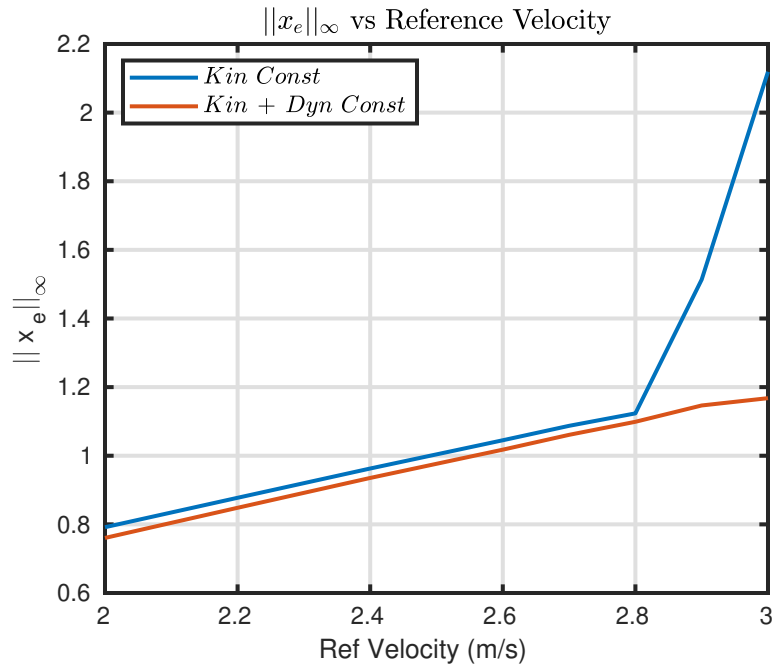


Figure 6.12: $\|x_e\|_\infty$ vs Reference Velocity: Simulation Results

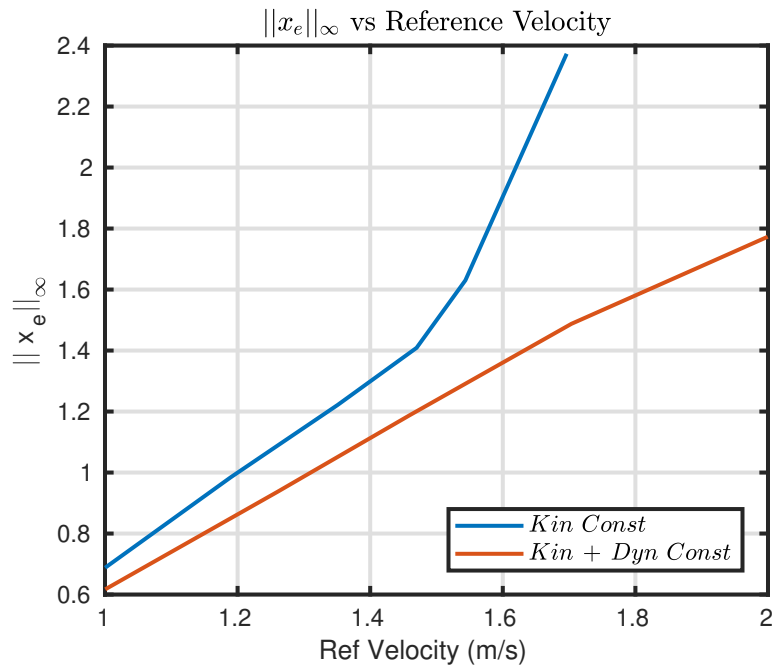


Figure 6.13: $\|x_e\|_\infty$ vs Reference Velocity: Hardware Results

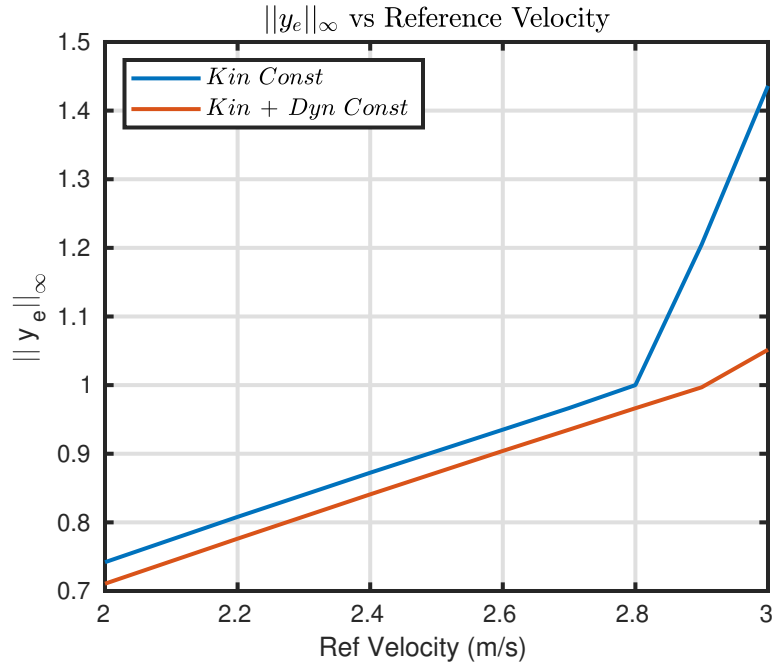


Figure 6.14: $\|y_e\|_\infty$ vs Reference Velocity: Simulation Results

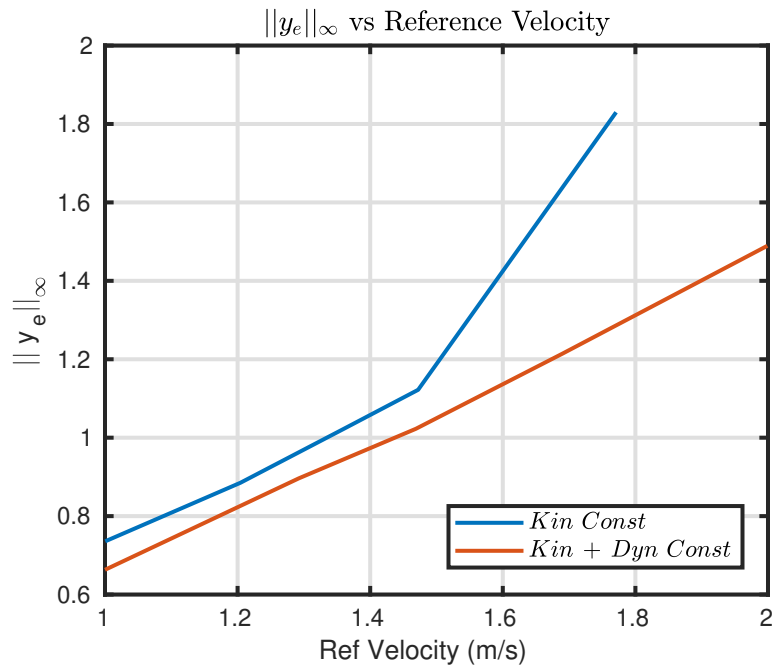


Figure 6.15: $\|y_e\|_\infty$ vs Reference Velocity: Hardware Results

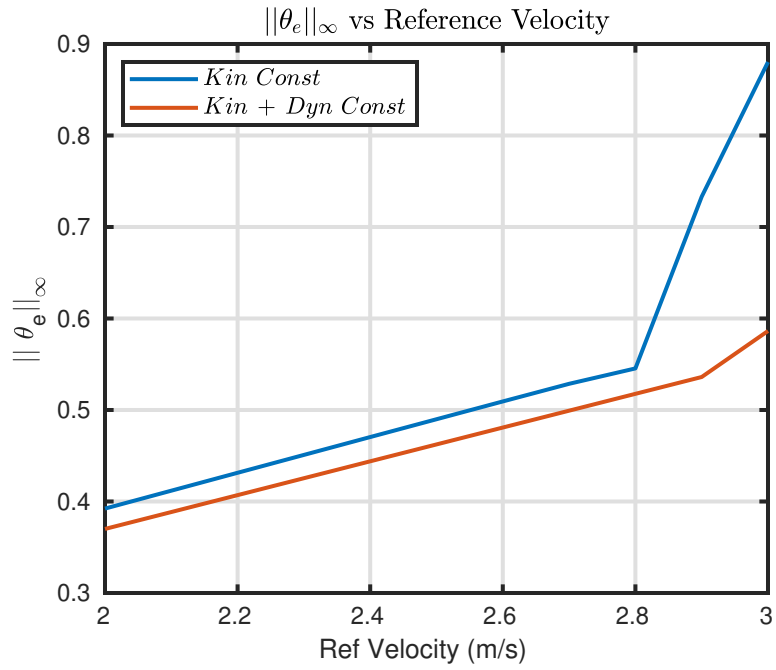


Figure 6.16: ||θ_e||_∞ vs Reference Velocity: Simulation Results

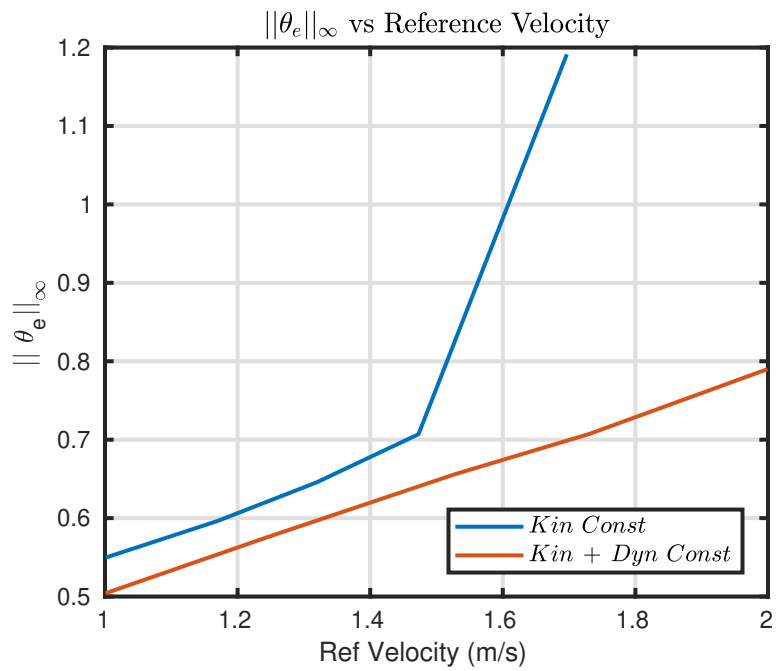


Figure 6.17: ||θ_e||_∞ vs Reference Velocity: Hardware Results

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.

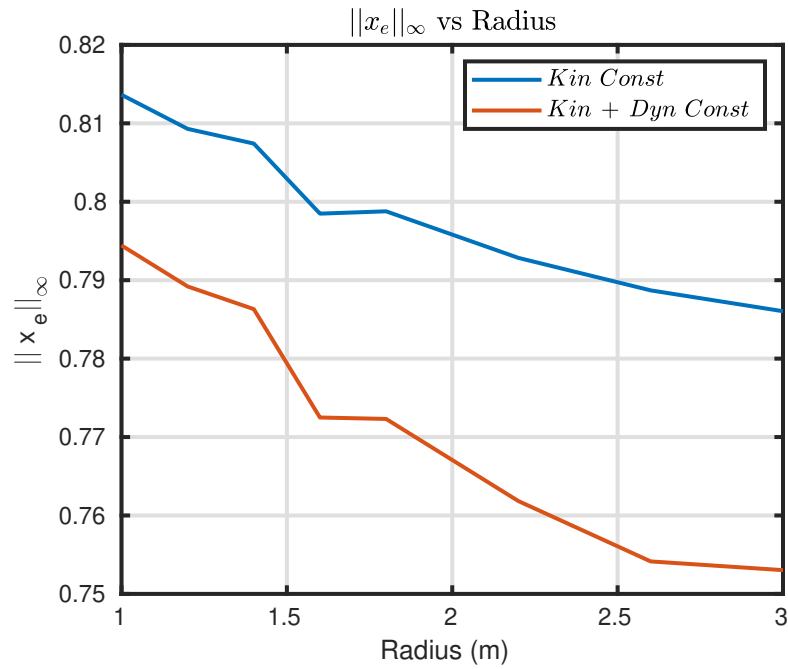


Figure 6.18: $\|x_e\|_\infty$ vs Radius of Track: Simulation Results

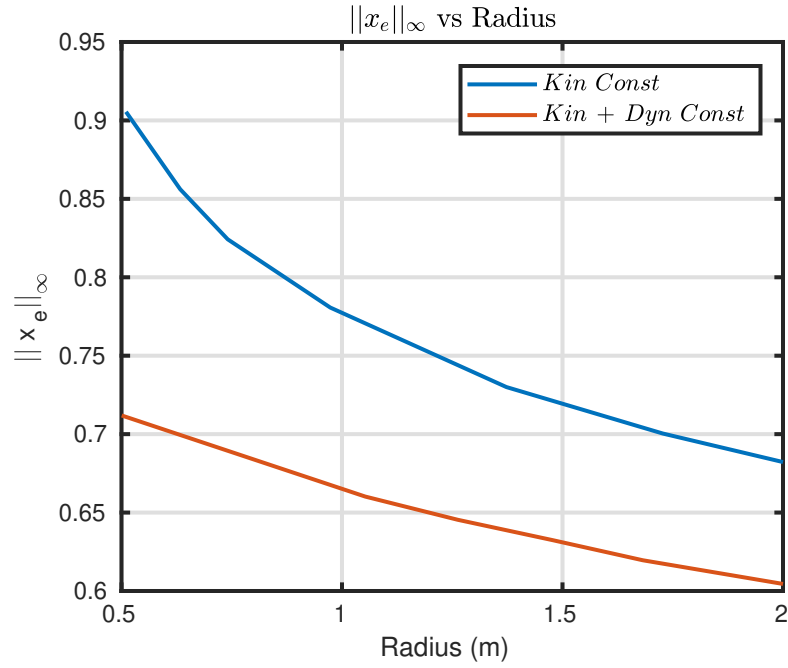


Figure 6.19: $\|x_e\|_\infty$ vs Radius of Track: Hardware Results

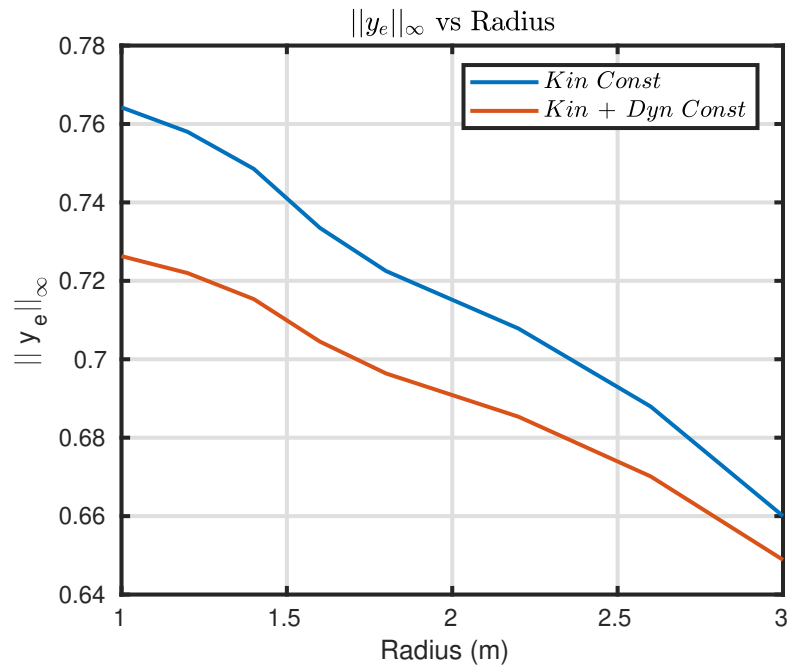


Figure 6.20: $\|y_e\|_\infty$ vs Radius of Track: Simulation Results

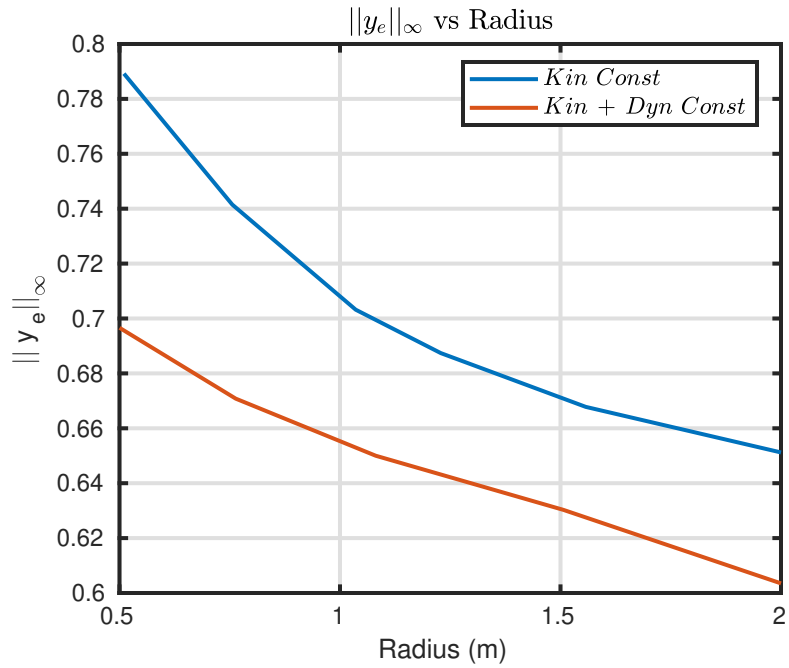


Figure 6.21: $\|y_e\|_\infty$ vs Radius of Track: Hardware Results

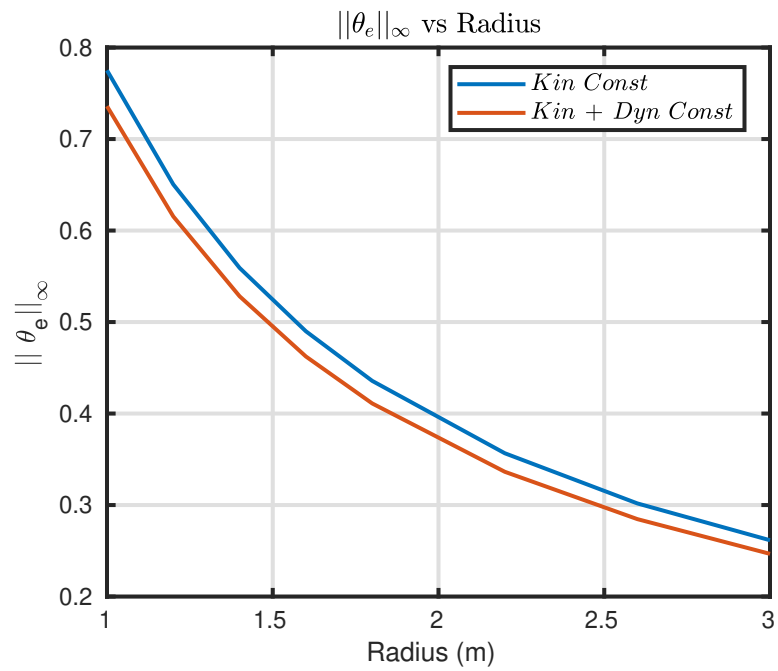


Figure 6.22: $\|\theta_e\|_\infty$ vs Radius of Track: Simulation Results

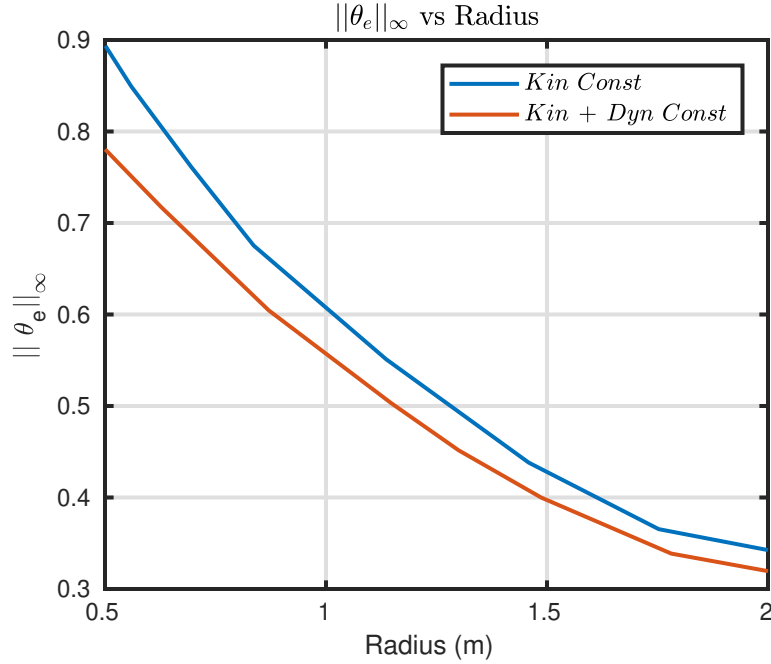


Figure 6.23: $\|\theta_e\|_\infty$ vs Radius of Track: Hardware Results

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 ($P_{[e_{ar}, e_{al}] \rightarrow [\omega_r, \omega_l]}$, $P_{[e_{ar}, e_{al}] \rightarrow [v, \omega]}$, and $P_{[e_{ar} + e_{al}, e_{ar} - e_{al}] \rightarrow [v, \omega]}$). 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 signifi-

cantly 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.

REFERENCES

- [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] Alrifaae, B., J. Maczijekowski 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. Häggglund, 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 nonholonomic 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

```

1 % Trade Studies at d = 0
2 clc
3 close all
4 clear all
5 s = tf([1 0],[1]);
6 md = 0; m = 3.4; % if d = 0;
7 %% Different Plant Models with the respective parameters as input
8 % at d = 0
9 % Plant model from e_r, e_l to W_r, W_l decoupled
10 % Plant model from (e_r + e_l), (e_r-e_l) to V, W decoupled
11 %%
12 % Plant model from e_r, e_l to W_r, W_l
13 % Singular and Bode Plots for different values of I
14 %(including the I_AR conditions)
15 d = 0; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
16 L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 /\
17 %has to be chosen based on the corresponding AR value (AR_calculation.m)
18 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values /\
19 %has to be chosen based on the corresponding AR value
20 I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
21 %[max,min] = Imaxmin(d,Iw,L,md,dw);
22 Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)
23
24 I = [0.424999999999 0.42500 0.425000000001 0.4292 0.4462 0.4675 0.3825];
25
26 P1 = Plantww(d, Veq, Weq, dw, Iw, I(1), L, md,R);
27 P2 = Plantww(d, Veq, Weq, dw, Iw, I(2), L, md,R);
28 P3 = Plantww(d, Veq, Weq, dw, Iw, I(3), L, md,R);
29 P4 = Plantww(d, Veq, Weq, dw, Iw, I(4), L, md,R);
30 P5 = Plantww(d, Veq, Weq, dw, Iw, I(5), L, md,R);
31 P6 = Plantww(d, Veq, Weq, dw, Iw, I(6), L, md,R);
32 P7 = Plantvw(d, Veq, Weq, dw, Iw, I(7), L, md,R);
33
34 figure;
35 bodemag(P7);
36 grid on;
37 h_axes = findobj(gcf, 'type', 'axes');
38 xlabel('Frequency','FontSize',12);
39 ylabel('Magnitude','FontSize',12);
40 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
41 % size and brightness of grid and size of x & y axis numbers
42 title('Frequency Response for $ d = 0 $ ','FontWeight','bold',...
43 'FontSize',14,'Interpreter','latex')
44
45 h_line = findobj(gcf, 'type', 'line');
46 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
47
48
49 %% Singular Values Plot
50 winit = -1;
51 wfin = 2;
52 nwpts = 200;
53 w = logspace(winit,wfin,nwpts);
54 P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w); P4 = sigma(P4,w);
55 P5 = sigma(P5,w);
56 P6 = sigma(P6,w); P7 = sigma(P7,w);
57 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);

```

```

58 P4 = 20*log10(P4);
59 P5 = 20*log10(P5);
60 P6 = 20*log10(P6); P7 = 20*log10(P7);
61 figure;
62 subplot(2,1,1);
63 semilogx(w, P7(1,:), w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:),...
64 w, P5(1,:), w, P6(1,:))
65 %clear sv
66 grid on;
67 h_axes = findobj(gcf, 'type', 'axes');
68 xlabel('Frequency','FontSize',12);
69 ylabel('Magnitude','FontSize',12);
70 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
71 % size and brightness of grid and size of x & y axis numbers
72 title(...
73 'Max Singular Values $ (e_r,e_l)\rightarrow(\omega_r,\omega_l) $ for $
74 d = 0 $', 'FontWeight','bold', 'FontSize',14, 'Interpreter','latex')
75
76 h_line = findobj(gcf, 'type', 'line');
77 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
78
79 subplot(2,1,2);
80 semilogx(w, P7(2,:), w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:),...
81 w, P5(2,:), w, P6(2,:))
82 %clear sv
83 grid on;
84 h_axes = findobj(gcf, 'type', 'axes');
85 xlabel('Frequency','FontSize',12);
86 ylabel('Magnitude','FontSize',12);
87 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
88 % size and brightness of grid and size of x & y axis numbers
89 title(...
90 'Min Singular Values $ (e_r,e_l)\rightarrow(\omega_r,\omega_l) $ for $
91 d = 0 $', 'FontWeight','bold', 'FontSize',12, 'Interpreter','latex')
92
93 h_line = findobj(gcf, 'type', 'line');
94 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
95
96
97
98
99 % Put legend and enhance appearance
100 % Legend bug with subscript, use '\_' instead of '_'
101 [hL,hObj]=legend({'$I = 0.9I_{AR}$', '$I = I_{AR}^{-}$', '$I = I_{AR}$', ...
102 '$I = I_{AR}^{+}$', '$I = 1.01I_{AR}$', '$I = 1.05I_{AR}$', ...
103 '$I = 1.1I_{AR}$'}, 'Interpreter','latex');
104 hTL=findobj(hObj, 'type', 'Text'); %
105 set(hTL, 'FontSize',11); % font size for letters in legend
106 hTL=findobj(hObj, 'type', 'line'); %
107 set(hTL, 'LineWidth',1.2); % thickness of lines in legend
108 set(hL, 'FontSize',1, 'Position',[0.5 0.5 0.25 0.26]);
109 % distance between lines in legend [x,y,width, height]
110
111
112 %%
113 % Plant model from (e_r + e_l), (e_r-e_l) to V, W
114 % Singular and Bode Plots for different values of I

```

```

115 %(including the I_AR conditions)
116 md = 0; m = 3.4; % if d = 0;
117 Plant1 = Plantsdv(d, Veq, Weq, dw, Iw, I_AR, L, md,R)
118
119
120 I = [0.3825 0.42560 0.4675 0.6375 0.8500 1.2750];
121
122 P1 = Plantsdv(d, Veq, Weq, dw, Iw, I(1), L, md,R);
123 P2 = Plantsdv(d, Veq, Weq, dw, Iw, I(2), L, md,R);
124 P3 = Plantsdv(d, Veq, Weq, dw, Iw, I(3), L, md,R);
125 P4 = Plantsdv(d, Veq, Weq, dw, Iw, I(4), L, md,R);
126 P5 = Plantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R);
127 P6 = Plantsdv(d, Veq, Weq, dw, Iw, I(6), L, md,R);
128
129 figure;
130 bodemag(P1,P2,P3,P4,P5,P6);
131 grid on;
132 h_axes = findobj(gcf, 'type', 'axes');
133 xlabel('Frequency','FontSize',12);
134 ylabel('Magnititude','FontSize',12);
135 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
136 % size and brightness of grid and size of x & y axis numbers
137 title(...
138 'Frequency Response  $(e_r + e_l, e_r - e_l) \rightarrow (v, \omega)$  for
139  $d = 0$ ', 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
140
141 h_line = findobj(gcf, 'type', 'line');
142 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
143
144 % Put legend and enhance appearance
145 % Legend bug with subscript, use '\_' instead of '-'
146 [hL,hObj]=legend({'$I = 0.9I\_{AR}$', '$I = I\_{AR}$', ...
147 '$I = 1.0I\_{AR}^+$', '$I = 1.5I\_{AR}$', '$I = 2.0I\_{AR}$', ...
148 '$I = 3.0I\_{AR}$'}, 'Interpreter', 'latex');
149 hTL=findobj(hObj, 'type', 'Text'); %
150 set(hTL, 'FontSize',11); % font size for letters in legend
151 hTL=findobj(hObj, 'type', 'line'); %
152 set(hTL, 'LineWidth',2); % thickness of lines in legend
153 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.25 0.24]);
154 % distance between lines in legend [x,y,width, height]
155
156 %% Singular Values Plot
157 winit = -1;
158 wfin = 2;
159 nwpts = 200;
160 w = logspace(winit,wfin,nwpts);
161 P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w); P4 = sigma(P4,w);
162 P5 = sigma(P5,w);
163 P6 = sigma(P6,w);
164 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
165 P4 = 20*log10(P4);
166 P5 = 20*log10(P5);
167 P6 = 20*log10(P6);
168 figure;
169 subplot(2,1,1);
170 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:), ...
171 w, P6(1,:))

```

```

172 %clear sv
173 grid on;
174 h_axes = findobj(gcf, 'type', 'axes');
175 xlabel('Frequency','FontSize',12);
176 ylabel('Magnitude','FontSize',12);
177 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
178 % size and brightness of grid and size of x & y axis numbers
179 title(...
180 'Max Singular Values $ (e_r + e_l, e_r - e_l)\rightarrow(v, \omega)$
181 for $d=0$, 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
182
183 h_line = findobj(gcf, 'type', 'line');
184 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
185
186 subplot(2,1,2);
187 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:), ...
188 w, P6(2,:))
189 %clear sv
190 grid on;
191 h_axes = findobj(gcf, 'type', 'axes');
192 xlabel('Frequency','FontSize',12);
193 ylabel('Magnitude','FontSize',12);
194 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
195 % size and brightness of grid and size of x & y axis numbers
196 title(...
197 'Min Singular Values $ (e_r + e_l, e_r - e_l)\rightarrow(v, \omega)$ for
198 $d=0$, 'FontWeight', 'bold', 'FontSize',12, 'Interpreter', 'latex')
199
200 h_line = findobj(gcf, 'type', 'line');
201 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
202
203
204
205
206 % Put legend and enhance appearance
207 % Legend bug with subscript, use '\_' instead of '_'
208 [hL,hObj]=legend({'$I = 0.9I_{AR}$', '$I = I_{AR}$', ...
209 '$I = 1.01I_{AR}^+$', '$I = 1.5I_{AR}$', '$I = 2.0I_{AR}$', ...
210 '$I = 3.0I_{AR}$'}, 'Interpreter', 'latex');
211 hTL=findobj(hObj, 'type', 'Text'); %
212 set(hTL, 'FontSize',11); % font size for letters in legend
213 hTL=findobj(hObj, 'type', 'line'); %
214 set(hTL, 'LineWidth',1.2); % thickness of lines in legend
215 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.25 0.24]);
216 % distance between lines in legend [x,y,width, height]
217
218
219
220
221
222 %%
223 % Plant model from e_r, e_l to W_r, W_l
224 % Singular and Bode Plots for different values of m
225 %(variations in total mass without changing I_w)
226
227 % Bode Plot
228

```

```

229 md = 0; m = 3.4; % if d = 0;
230 d = 0; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
231 L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 /\
232 %has to be chosen based on the corresponding AR value (AR_calculation.m)
233 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values /\
234 %has to be chosen based on the corresponding AR value
235 I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
236 [max,min] = I_maxmin(d,Iw,L,md,dw);
237 Plant1 = Planttw(d, Veq, Weq, dw, Iw, I_AR, L, md,R)
238
239 I = [0.424999999999 0.42500 0.425000000001 0.4292 0.4462 0.4675 0.3825];
240
241
242 P1 = newPlanttw(d, Veq, Weq, dw, Iw, I(5), L, md,R,m);
243 P2 = newPlanttw(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+0.5);
244 P3 = newPlanttw(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1);
245 P4 = newPlanttw(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1.5);
246 P5 = newPlanttw(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2);
247 P6 = newPlanttw(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2.5);
248 P7 = newPlanttw(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+3);
249
250 figure;
251 bodemag(P1,P2,P3,P4,P5,P6,P7);
252 grid on;
253 h_axes = findobj(gcf, 'type', 'axes');
254 xlabel('Frequency','FontSize',12);
255 ylabel('Magnitude','FontSize',12);
256 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
257 % size and brightness of grid and size of x & y axis numbers
258 title(...
259 'Frequency Response $ (e_r,e_l)\rightarrow(\omega_r,\omega_l) $ for
260 $ d = 0 $ ', 'FontWeight','bold', 'FontSize',14, 'Interpreter','latex')
261
262 h_line = findobj(gcf, 'type', 'line');
263 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
264
265 % Put legend and enhance appearance
266 % Legend bug with subscript, use '\_' instead of '_'
267 [hL,hObj]=legend({'$m = 3.4 \ kg$', '$m = 3.9 \ kg$', '$m = 4.4 \ kg$', ...
268 '$m = 4.9 \ kg$', '$m = 5.4 \ kg$', '$m = 5.9 \ kg$', 'm = 6.4 \ kg'}, ...
269 'Interpreter','latex');
270 hTL=findobj(hObj, 'type', 'Text'); %
271 set(hTL, 'FontSize',11); % font size for letters in legend
272 hTL=findobj(hObj, 'type', 'line'); %
273 set(hTL, 'LineWidth',2); % thickness of lines in legend
274 set(hL, 'FontSize',1, 'Position',[0.5 0.5 0.25 0.26]);
275 % distance between lines in legend [x,y,width, height]
276
277 %% Singular Values Plot
278 winit = -1;
279 wfin = 2;
280 nwpts = 200;
281 w = logspace(winit,wfin,nwpts);
282 P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w); P4 = sigma(P4,w);
283 P5 = sigma(P5,w);
284 P6 = sigma(P6,w); P7 = sigma(P7,w);
285 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);

```

```

286 P4 = 20*log10(P4);
287 P5 = 20*log10(P5);
288 P6 = 20*log10(P6); P7 = 20*log10(P7);
289 figure;
290 subplot(2,1,1);
291 semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), ...
292 w, P5(1,:), w, P6(1,:),w,P7(1,:))
293 %clear sv
294 grid on;
295 h_axes = findobj(gcf, 'type', 'axes');
296 xlabel('Frequency','FontSize',12);
297 ylabel('Magnititude','FontSize',12);
298 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
299 % size and brightness of grid and size of x & y axis numbers
300 title(...
301 'Max Singular Values  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for...
302 $ d=0$', 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
303
304 h_line = findobj(gcf, 'type', 'line');
305 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
306
307 subplot(2,1,2);
308 semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), ...
309 w, P5(2,:), w, P6(2,:),w,P7(1,:))
310 %clear sv
311 grid on;
312 h_axes = findobj(gcf, 'type', 'axes');
313 xlabel('Frequency','FontSize',12);
314 ylabel('Magnititude','FontSize',12);
315 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
316 % size and brightness of grid and size of x & y axis numbers
317 title(...
318 'Min Singular Values $ (e_r, e_l) \rightarrow (\omega_r, \omega_l) $ for
319 $ d = 0 $', 'FontWeight', 'bold', 'FontSize',12, 'Interpreter', 'latex')
320
321 h_line = findobj(gcf, 'type', 'line');
322 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
323
324
325
326
327 % Put legend and enhance appearance
328 % Legend bug with subscript, use '\_' instead of '_'
329 [hL,hObj]=legend({'$m = 3.4 \ kg$', '$m = 3.9 \ kg$', '$m = 4.4 \ kg$', ...
330 '$m = 4.9 \ kg$', '$m = 5.4 \ kg$', '$m = 5.9 \ kg$', 'm = 6.4 \ kg'}, ...
331 'Interpreter', 'latex');
332 hTL=findobj(hObj, 'type', 'Text'); %
333 set(hTL, 'FontSize',11); % font size for letters in legend
334 hTL=findobj(hObj, 'type', 'line'); %
335 set(hTL, 'LineWidth',1.2); % thickness of lines in legend
336 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.25 0.26]);
337 % distance between lines in legend [x,y,width, height]
338
339
340 %%
341 % Plant model from e_r, e_l to W_r, W_l
342 % Singular and Bode Plots for different values of R

```



```

343
344 % Bode Plot
345
346 % change in R results in change in IW, however, no significant
347 % difference is observed
348
349 md = 0; m = 3.4; % if d = 0;
350 d = 0; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
351 L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
352 %has to be chosen based on the corresponding AR value (AR_calculation.m)
353 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\\
354 %has to be chosen based on the corresponding AR value
355 I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
356 [max,min] = I_maxmin(d,Iw,L,md,dw);
357 Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)
358
359 I = [0.424999999999 0.42500 0.425000000001 0.4292 0.4462 0.4675 0.3825];
360
361 R = 0.042; m_wheel = 0.096;
362 rm = 0.0248 ; m_motor = 0.224;
363 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
364 I = I_Newcalculation(0,Iw,L,md,dw);
365
366 P1 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
367 R = R+0.01;
368 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
369 I = I_Newcalculation(0,Iw,L,md,dw);
370 P2 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
371 R = R+0.01;
372 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
373 I = I_Newcalculation(0,Iw,L,md,dw);
374 P3 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
375 R = R+0.01;
376 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
377 I = I_Newcalculation(0,Iw,L,md,dw);
378 P4 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
379 R = R+0.01;
380 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
381 I = I_Newcalculation(0,Iw,L,md,dw);
382 P5 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
383 R = R+0.01;
384 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
385 I = I_Newcalculation(0,Iw,L,md,dw);
386 P6 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
387
388
389 R = 0.042; Iw = 1.67e-06;
390 figure;
391 bodemag(P1,P2,P3,P4,P5,P6);
392 grid on;
393 h_axes = findobj(gcf, 'type', 'axes');
394 xlabel('Frequency','FontSize',12);
395 ylabel('Magnitude','FontSize',12);
396 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
397 % size and brightness of grid and size of x & y axis numbers
398 title(...
399 'Frequency Response $ (e_r,e_l)\rightarrow(\omega_r,\omega_l) $ for

```

```

400 $ d = 0 $ ', 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex')
401
402 h_line = findobj(gcf, 'type', 'line');
403 set(h_line, 'LineWidth', 1.5); % Lines with thicker width for plots
404
405 % Put legend and enhance appearance
406 % Legend bug with subscript, use '\_' instead of '_'
407 [hL, hObj]=legend({'$R = 0.042 \ m$', '$R = 0.052 \ m$', ...
408 '$R = 0.062 \ m$', '$R = 0.072 \ m$', '$R = 0.082 \ m$', ...
409 '$R = 0.092 \ m$'}, 'Interpreter', 'latex');
410 hTL=findobj(hObj, 'type', 'Text'); %
411 set(hTL, 'FontSize', 11); % font size for letters in legend
412 hTL=findobj(hObj, 'type', 'line'); %
413 set(hTL, 'LineWidth', 2); % thickness of lines in legend
414 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.25 0.26]);
415 % distance between lines in legend [x,y,width, height]
416
417 %% Singular Values Plot
418 winit = -1;
419 wfin = 2;
420 nwpts = 200;
421 w = logspace(winit, wfin, nwpts);
422 P1 = sigma(P1, w); P2 = sigma(P2, w); P3 = sigma(P3, w);
423 P4 = sigma(P4, w); P5 = sigma(P5, w);
424 P6 = sigma(P6, w);
425 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
426 P4 = 20*log10(P4); P5 = 20*log10(P5);
427 P6 = 20*log10(P6);
428 figure;
429 subplot(2,1,1);
430 semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:), ...
431 w, P6(1,:))
432 %clear sv
433 grid on;
434 h_axes = findobj(gcf, 'type', 'axes');
435 xlabel('Frequency', 'FontSize', 12);
436 ylabel('Magnitude', 'FontSize', 12);
437 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
438 % size and brightness of grid and size of x & y axis numbers
439 title(...
440 'Max Singular Values $ (e_r, e_l)\rightarrow(\omega_r, \omega_l) $ for
441 $ d = 0 $ ', 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex')
442
443 h_line = findobj(gcf, 'type', 'line');
444 set(h_line, 'LineWidth', 1.2); % Lines with thicker width for plots
445
446 subplot(2,1,2);
447 semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:), ...
448 w, P6(2,:))
449 %clear sv
450 grid on;
451 h_axes = findobj(gcf, 'type', 'axes');
452 xlabel('Frequency', 'FontSize', 12);
453 ylabel('Magnitude', 'FontSize', 12);
454 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
455 % size and brightness of grid and size of x & y axis numbers
456 title(...

```

```

457 'Min Singular Values $ (e_r, e_l)\rightarrow(\omega_r, \omega_l) $ for
458 $ d = 0 $', 'FontWeight', 'bold', 'FontSize', 12, 'Interpreter', 'latex')
459
460 h_line = findobj(gcf, 'type', 'line');
461 set(h_line, 'LineWidth', 1.2); % Lines with thicker width for plots
462
463
464
465
466 % Put legend and enhance appearance
467 % Legend bug with subscript, use '\_' instead of '_'
468 [hL, hObj]=legend({'$R = 0.042 \ m$', '$R = 0.052 \ m$', ...
469 '$R = 0.062 \ m$', '$R = 0.072 \ m$', '$R = 0.082 \ m$', ...
470 '$R = 0.092 \ m$'}, 'Interpreter', 'latex');
471 hTL=findobj(hObj, 'type', 'Text'); %
472 set(hTL, 'FontSize', 11); % font size for letters in legend
473 hTL=findobj(hObj, 'type', 'line'); %
474 set(hTL, 'LineWidth', 1.2); % thickness of lines in legend
475 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.25 0.26]);
476 % distance between lines in legend [x,y,width, height]
477 %%
478 % Plant model from e_r + e_l, e_r - e_l to V,W
479 % Singular and Bode Plots for different values of m
480 %(variations in total mass without changing I_w)
481
482 % Bode Plot
483
484 md = 0; m = 3.4; % if d = 0;
485 d = 0; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
486 L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\
487 % has to be chosen based on the corresponding AR value
488 % (AR_calculation.m)
489 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\
490 % has to be chosen based on the corresponding AR value
491 I_AR = I_ARcalculation(d, Iw, L, A, R, dw);
492 [max, min] = Imaxmin(d, Iw, L, md, dw);
493 Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md, R)
494
495 I = [0.424999999999 0.42500 0.425000000001 0.4292 0.4462 0.4675 0.3825];
496
497
498 P1 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md, R, m);
499 P2 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md, R, m+0.5);
500 P3 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md, R, m+1);
501 P4 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md, R, m+1.5);
502 P5 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md, R, m+2);
503 P6 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md, R, m+2.5);
504 P7 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md, R, m+3);
505
506 figure;
507 bodemag(P1, P2, P3, P4, P5, P6);
508 grid on;
509 h_axes = findobj(gcf, 'type', 'axes');
510 xlabel('Frequency', 'FontSize', 12);
511 ylabel('Magnitude', 'FontSize', 12);
512 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
513 % size and brightness of grid and size of x & y axis numbers

```

```

514 title(...
515 'Frequency Response  $(e_r + e_l, e_r - e_l) \rightarrow (v, \omega)$  for
516  $\delta = 0$ ', 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex')
517
518 h_line = findobj(gcf, 'type', 'line');
519 set(h_line, 'LineWidth', 1.5); % Lines with thicker width for plots
520
521 % Put legend and enhance appearance
522 % Legend bug with subscript, use '\_' instead of '_'
523 [hL, hObj]=legend({'$m = 3.4 \ kg$', '$m = 3.9 \ kg$', '$m = 4.4 \ kg$', ...
524 '$m = 4.9 \ kg$', '$m = 5.4 \ kg$', '$m = 5.9 \ kg$'}, 'Interpreter', ...
525 'latex');
526 hTL=findobj(hObj, 'type', 'Text'); %
527 set(hTL, 'FontSize', 11); % font size for letters in legend
528 hTL=findobj(hObj, 'type', 'line'); %
529 set(hTL, 'LineWidth', 2); % thickness of lines in legend
530 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.25 0.26]);
531 % distance between lines in legend [x,y,width, height]
532
533 %% Singular Values Plot
534 winit = -1;
535 wfin = 2;
536 nwpts = 200;
537 w = logspace(winit, wfin, nwpts);
538 P1 = sigma(P1, w); P2 = sigma(P2, w); P3 = sigma(P3, w);
539 P4 = sigma(P4, w); P5 = sigma(P5, w);
540 P6 = sigma(P6, w); P7 = sigma(P7, w);
541 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
542 P4 = 20*log10(P4); P5 = 20*log10(P5);
543 P6 = 20*log10(P6); P7 = 20*log10(P7);
544 figure;
545 subplot(2,1,1);
546 semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), ...
547 w, P5(1,:), w, P6(1,:))
548 %clear sv
549 grid on;
550 h_axes = findobj(gcf, 'type', 'axes');
551 xlabel('Frequency', 'FontSize', 12);
552 ylabel('Magnitude', 'FontSize', 12);
553 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
554 % size and brightness of grid and size of x & y axis numbers
555 title(...
556 'Max Singular Values  $(e_r + e_l, e_r - e_l) \rightarrow (v, \omega)$  for
557  $\delta = 0$ ', 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex')
558
559 h_line = findobj(gcf, 'type', 'line');
560 set(h_line, 'LineWidth', 1.2); % Lines with thicker width for plots
561
562 subplot(2,1,2);
563 semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:), ...
564 w, P6(2,:));
565 %clear sv
566 grid on;
567 h_axes = findobj(gcf, 'type', 'axes');
568 xlabel('Frequency', 'FontSize', 12);
569 ylabel('Magnitude', 'FontSize', 12);
570 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);

```

```

571 % size and brightness of grid and size of x & y axis numbers
572 title(...
573 'Min Singular Values  $(e_r + e_l, e_r - e_l) \rightarrow (v, \omega)$  for
574  $d = 0$ ', 'FontWeight','bold','FontSize',12, 'Interpreter','latex');
575
576 h_line = findobj(gcf, 'type', 'line');
577 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
578
579
580
581
582 % Put legend and enhance appearance
583 % Legend bug with subscript, use '\_' instead of '_'
584 [hL,hObj]=legend({'$m = 3.4 \ kg$', '$m = 3.9 \ kg$', '$m = 4.4 \ kg$', ...
585 '$m = 4.9 \ kg$', '$m = 5.4 \ kg$', '$m = 5.9 \ kg$'}, 'Interpreter', ...
586 'latex');
587 hTL=findobj(hObj, 'type', 'Text'); %
588 set(hTL, 'FontSize',11); % font size for letters in legend
589 hTL=findobj(hObj, 'type', 'line'); %
590 set(hTL, 'LineWidth',1.2); % thickness of lines in legend
591 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.25 0.26]);
592 % distance between lines in legend [x,y,width, height]
593
594 %%
595 % Plant model from  $e_r + e_l$ ,  $e_r + e_l$  to V, W
596 % Singular and Bode Plots for different values of R
597
598 % Bode Plot
599
600 % change in R results in change in IW, however, no significant
601 % difference is observed
602
603 md = 0; m = 3.4; % if d = 0;
604 d = 0; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
605 L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\
606 % has to be chosen based on the corresponding AR value
607 % (AR_calculation.m)
608 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\
609 % has to be chosen based on the corresponding AR value
610 I_AR = I_ARcalculation(d, Iw, L, A, R, dw);
611 [max, min] = Imaxmin(d, Iw, L, md, dw);
612 Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md, R)
613
614 I = [0.424999999999 0.42500 0.425000000001 0.4292 0.4462 0.4675 0.3825];
615
616 R = 0.042; m_wheel = 0.096;
617 rm = 0.0248 ; m_motor = 0.224;
618 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
619 I = INewcalculation(0, Iw, L, md, dw);
620
621 P1 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md, R, m);
622 R = R+0.01;
623 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
624 I = INewcalculation(0, Iw, L, md, dw);
625 P2 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md, R, m);
626 R = R+0.01;
627 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;

```

```

628 I = I.Newcalculation(0,Iw,L,md,dw);
629 P3 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
630 R = R+0.01;
631 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
632 I = I.Newcalculation(0,Iw,L,md,dw);
633 P4 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
634 R = R+0.01;
635 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
636 I = I.Newcalculation(0,Iw,L,md,dw);
637 P5 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
638 R = R+0.01;
639 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
640 I = I.Newcalculation(0,Iw,L,md,dw);
641 P6 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md,R,m);
642
643
644 R = 0.042; Iw = 1.67e-06;
645 figure;
646 bodemag(P1,P2,P3,P4,P5,P6);
647 grid on;
648 h_axes = findobj(gcf, 'type', 'axes');
649 xlabel('Frequency','FontSize',12);
650 ylabel('Magnitude','FontSize',12);
651 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
652 % size and brightness of grid and size of x & y axis numbers
653 title(...
654 'Frequency Response  $(e_r + e_l, e_r - e_l) \rightarrow(v, \omega)$  for
655  $d = 0$  ', 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
656
657 h_line = findobj(gcf, 'type', 'line');
658 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
659
660 % Put legend and enhance appearance
661 % Legend bug with subscript, use '\_' instead of '_'
662 [hL,hObj]=legend({'$R = 0.042 \ m$', '$R = 0.052 \ m$', ...
663 '$R = 0.062 \ m$', '$R = 0.072 \ m$', '$R = 0.082 \ m$', ...
664 '$R = 0.092 \ m$'}, 'Interpreter', 'latex');
665 hTL=findobj(hObj, 'type', 'Text'); %
666 set(hTL, 'FontSize',11); % font size for letters in legend
667 hTL=findobj(hObj, 'type', 'line'); %
668 set(hTL, 'LineWidth',2); % thickness of lines in legend
669 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.25 0.26]);
670 % distance between lines in legend [x,y,width, height]
671
672 %% Singular Values Plot
673 winit = -1;
674 wfin = 2;
675 nwpts = 200;
676 w = logspace(winit,wfin,nwpts);
677 P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
678 P4 = sigma(P4,w); P5 = sigma(P5,w);
679 P6 = sigma(P6,w);
680 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
681 P4 = 20*log10(P4); P5 = 20*log10(P5);
682 P6 = 20*log10(P6);
683 figure;
684 subplot(2,1,1);

```

```

685 semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
686 w, P6(1,:))
687 %clear sv
688 grid on;
689 h_axes = findobj(gcf, 'type', 'axes');
690 xlabel('Frequency','FontSize',12);
691 ylabel('Magnititude','FontSize',12);
692 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
693 % size and brightness of grid and size of x & y axis numbers
694 title(...
695 'Max Singular Values $ (e_r + e_l, e_r - e_l)\rightarrow(v, \omega)$for
696 $d = 0$', 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
697
698 h_line = findobj(gcf, 'type', 'line');
699 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
700
701 subplot(2,1,2);
702 semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
703 w, P6(2,:))
704 %clear sv
705 grid on;
706 h_axes = findobj(gcf, 'type', 'axes');
707 xlabel('Frequency','FontSize',12);
708 ylabel('Magnititude','FontSize',12);
709 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
710 % size and brightness of grid and size of x & y axis numbers
711 title(...
712 'Min Singular Values $(e_r + e_l, e_r - e_l)\rightarrow(v, \omega)$ for
713 $d = 0$', 'FontWeight', 'bold', 'FontSize',12, 'Interpreter', 'latex')
714
715 h_line = findobj(gcf, 'type', 'line');
716 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
717
718
719
720
721 % Put legend and enhance appearance
722 % Legend bug with subscript, use '\_' instead of '_'
723 [hL,hObj]=legend({'$R = 0.042 \ m$', '$R = 0.052 \ m$',...
724 '$R = 0.062 \ m$', '$R = 0.072 \ m$', '$R = 0.082 \ m$',...
725 '$R = 0.092 \ m$'}, 'Interpreter', 'latex');
726 hTL=findobj(hObj, 'type', 'Text'); %
727 set(hTL, 'FontSize',11); % font size for letters in legend
728 hTL=findobj(hObj, 'type', 'line'); %
729 set(hTL, 'LineWidth',1.2); % thickness of lines in legend
730 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.25 0.26]);
731 % distance between lines in legend [x,y,width, height]

1 % Trade Studies at d ~ = 0
2 clc
3 close all
4 clear all
5 s = tf([1 0],[1]);
6 md = 0; m = 3.4; % if d = 0;
7 %% Different Plant Models with the respective parameters as input
8 % at d = 0

```

```

9 % Plant model from e_r, e_l to W_r, W_l decoupled
10 % Plant model from (e_r + e_l), (e_r-e_l) to V, W decoupled
11 %%
12 % Plant model from e_r, e_l to W_r, W_l
13 % Singular and Bode Plots for different values of Veq
14 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
15 L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
16 %has to be chosen based on the corresponding AR value (AR_calculation.m)
17 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\\
18 %has to be chosen based on the corresponding AR value
19 I = I.Newcalculation(d,Iw,L,md,dw);
20 [max,min] = Imaxmin(d,Iw,L,md,dw);
21 Plant1 = Plantww(d, Veq, Weq, dw, Iw, I, L, md,R);
22
23 Veq = [0.1 0.2 0.6 1 3 5];
24
25 P1 = Plantww(d, Veq(1), Weq, dw, Iw, I, L, md,R);
26 P2 = Plantww(d, Veq(2), Weq, dw, Iw, I, L, md,R);
27 P3 = Plantww(d, Veq(3), Weq, dw, Iw, I, L, md,R);
28 P4 = Plantww(d, Veq(4), Weq, dw, Iw, I, L, md,R);
29 P5 = Plantww(d, Veq(5), Weq, dw, Iw, I, L, md,R);
30 P6 = Plantww(d, Veq(6), Weq, dw, Iw, I, L, md,R);
31
32 figure;
33 bodemag(P1,P2,P3,P4,P5,P6);
34 grid on;
35 h_axes = findobj(gcf, 'type', 'axes');
36 xlabel('Frequency','FontSize',12);
37 ylabel('Magnititude','FontSize',12);
38 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
39 % size and brightness of grid and size of x & y axis numbers
40 title(...
41 'Frequency Response $ (e_r,e_l)\rightarrow(\omega_r,\omega_l)$ for
42 $d \neq 0$, 'FontWeight','bold','FontSize',14, 'Interpreter','latex')
43
44 h_line = findobj(gcf, 'type', 'line');
45 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
46
47 % Put legend and enhance appearance
48 % Legend bug with subscript, use '\_' instead of '_'
49 [hL,hObj]=legend({'$v_{eq} = 0.1 \ m/s$', '$v_{eq} = 0.2 \ m/s$', ...
50 '$v_{eq} = 0.6 \ m/s$', '$v_{eq} = 1.0 \ m/s$', ...
51 '$v_{eq} = 3.0 \ m/s$', '$v_{eq} = 5.0 \ m/s$'}, 'Interpreter','latex');
52 hTL=findobj(hObj, 'type', 'Text'); %
53 set(hTL, 'FontSize', 11); % font size for letters in legend
54 hTL=findobj(hObj, 'type', 'line'); %
55 set(hTL, 'LineWidth', 2); % thickness of lines in legend
56 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.26 0.24]);
57 % distance between lines in legend [x,y,width, height]
58
59 %% Singular and Bode Plots for different values of Weq
60 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
61 L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
62 %has to be chosen based on the corresponding AR value (AR_calculation.m)
63 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\\
64 %has to be chosen based on the corresponding AR value
65 I = I.Newcalculation(d,Iw,L,md,dw);

```



```

66 [max,min] = Imaxmin(d,Iw,L,md,dw);
67
68 Weq = [-8.0 -2.5 -0.5 0.5 2.5 8.0];
69
70 P1 = Plantww(d, Veq, Weq(1), dw, Iw, I, L, md,R);
71 P2 = Plantww(d, Veq, Weq(2), dw, Iw, I, L, md,R);
72 P3 = Plantww(d, Veq, Weq(3), dw, Iw, I, L, md,R);
73 P4 = Plantww(d, Veq, Weq(4), dw, Iw, I, L, md,R);
74 P5 = Plantww(d, Veq, Weq(5), dw, Iw, I, L, md,R);
75 P6 = Plantww(d, Veq, Weq(6), dw, Iw, I, L, md,R);
76
77 figure;
78 bodemag(P1,P2,P3,P4,P5,P6);
79 grid on;
80 h_axes = findobj(gcf, 'type', 'axes');
81 xlabel('Frequency','FontSize',12);
82 ylabel('Magnitude','FontSize',12);
83 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
84 % size and brightness of grid and size of x & y axis numbers
85 title(...
86 'Frequency Response  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
87  $d \neq 0$ ','FontWeight','bold','FontSize',14, 'Interpreter','latex')
88
89 h_line = findobj(gcf, 'type', 'line');
90 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
91
92 % Put legend and enhance appearance
93 % Legend bug with subscript, use '\-' instead of '-'
94 [hL,hObj]=legend({' $\omega = -8.0 \text{ rad/s}$ ',...
95 ' $\omega = -2.5 \text{ rad/s}$ ', ' $\omega = -0.5 \text{ rad/s}$ ',...
96 ' $\omega = 0.5 \text{ rad/s}$ ', ' $\omega = 2.5 \text{ rad/s}$ ',...
97 ' $\omega = 8.0 \text{ rad/s}$ '}, 'Interpreter','latex');
98 hTL=findobj(hObj, 'type', 'Text'); %
99 set(hTL, 'FontSize',10); % font size for letters in legend
100 hTL=findobj(hObj, 'type', 'line'); %
101 set(hTL, 'LineWidth',2); % thickness of lines in legend
102 set(hL, 'FontSize',1, 'Position',[0.5 0.5 0.28 0.24]);
103 % distance between lines in legend [x,y,width, height]
104
105 %% %% Singular and Bode Plots for different values of d
106 % the behaviour in the bode plots can be associated with the dominant
107 % pole variation wrt to d
108 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
109 L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\
110 % has to be chosen based on the corresponding AR value
111 % (AR_calculation.m)
112 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\
113 % has to be chosen based on the corresponding AR value
114 I = I.Newcalculation(d,Iw,L,md,dw);
115 [max,min] = Imaxmin(d,Iw,L,md,dw);
116
117 d = [-0.09 -0.08 -0.04 0.04 0.08 0.09];
118 I = [I.Newcalculation(d(1),Iw,L,md,dw) I.Newcalculation(d(2),Iw,L,md,dw)
119 I.Newcalculation(d(3),Iw,L,md,dw) I.Newcalculation(d(4),Iw,L,md,dw)
120 I.Newcalculation(d(5),Iw,L,md,dw)
121 I.Newcalculation(d(6),Iw,L,md,dw)];
122

```

```

123 P1 = Plantww(d(1), Veq, Weq, dw, Iw, I(1), L, md,R);
124 P2 = Plantww(d(2), Veq, Weq, dw, Iw, I(2), L, md,R);
125 P3 = Plantww(d(3), Veq, Weq, dw, Iw, I(3), L, md,R);
126 P4 = Plantww(d(4), Veq, Weq, dw, Iw, I(4), L, md,R);
127 P5 = Plantww(d(5), Veq, Weq, dw, Iw, I(5), L, md,R);
128 P6 = Plantww(d(6), Veq, Weq, dw, Iw, I(6), L, md,R);
129
130 figure;
131 bodemag(P1,P2,P3,P4,P5,P6);
132 grid on;
133 h_axes = findobj(gcf, 'type', 'axes');
134 xlabel('Frequency','FontSize',12);
135 ylabel('Magnititude','FontSize',12);
136 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
137 % size and brightness of grid and size of x & y axis numbers
138 title(...
139 'Frequency Response  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
140  $\omega \neq 0$ ', 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex');
141
142 h_line = findobj(gcf, 'type', 'line');
143 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
144
145 % Put legend and enhance appearance
146 % Legend bug with subscript, use '\_' instead of '_'
147 [hL,hObj]=legend({' $\omega = -0.1 \text{ m}$ ', ' $\omega = -0.05 \text{ m}$ ', ...
148 ' $\omega = -0.02 \text{ m}$ ', ' $\omega = 0.02 \text{ m}$ ', ' $\omega = 0.05 \text{ m}$ ', ' $\omega = 0.1 \text{ m}$ '}, ...
149 'Interpreter', 'latex');
150 hTL=findobj(hObj, 'type', 'Text'); %
151 set(hTL, 'FontSize',10); % font size for letters in legend
152 hTL=findobj(hObj, 'type', 'line'); %
153 set(hTL, 'LineWidth',2); % thickness of lines in legend
154 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.26 0.24]);
155 % distance between lines in legend [x,y,width, height]
156
157 %% plot of dominant pole vs d
158 figure;
159 load('dpole_2_1.mat');
160 d = -0.28:0.01:0.28
161 plot(d,h(1,:),d,h(2,:),d,h(3,:),d,h(4,:),d,h(5,:),d,h(6,:),d,h(7,:));
162 grid on;
163 h_axes = findobj(gcf, 'type', 'axes');
164 xlabel('d (m)','FontSize',12);
165 ylabel('Dominant Pole ','FontSize',12);
166 set(h_axes, 'LineWidth',2, 'FontSize',12, 'GridAlpha',0.15);
167 % size and brightness of grid and size of x & y axis numbers
168 title('Dominant Pole vs  $\omega$ ', 'FontWeight', 'bold', 'FontSize',14, ...
169 'Interpreter', 'latex')
170
171 h_line = findobj(gcf, 'type', 'line');
172 set(h_line, 'LineWidth',1.8); % Lines with thicker width for
173 % plots
174 [hL,hObj]=legend({' $\omega_{eq} = 0.0 \text{ m/s}$ ', ' $\omega_{eq} = 0.5 \text{ m/s}$ ', ...
175 ' $\omega_{eq} = 1.0 \text{ m/s}$ ', ' $\omega_{eq} = 1.5 \text{ m/s}$ ', ' $\omega_{eq} = 2.0 \text{ m/s}$ ', ...
176 ' $\omega_{eq} = 2.5 \text{ m/s}$ ', ' $\omega_{eq} = 3.0 \text{ m/s}$ '}, 'Interpreter', 'latex');
177 hTL=findobj(hObj, 'type', 'Text'); %
178 set(hTL, 'FontSize',10); % font size for letters in legend
179 hTL=findobj(hObj, 'type', 'line'); %

```

```

180 set(hTL, 'LineWidth', 2); % thickness of lines in legend
181 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.27 0.24]);
182 % distance between lines in legend [x,y,width, height]
183
184 %% %% Singular and Bode Plots for different values of I
185 % the behaviour in the bode plots can be associated with the dominant
186 % pole variation wrt to d
187 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
188 L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
189 %has to be chosen based on the corresponding AR value (AR_calculation.m)
190 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\\
191 %has to be chosen based on the corresponding AR value
192 I = I.Newcalculation(d, Iw, L, md, dw);
193 [max, min] = Imaxmin(d, Iw, L, md, dw);
194
195 I = [0.4 0.5 0.7 0.9 1.2 1.7];
196
197 P1 = Plantww(d, Veq, Weq, dw, Iw, I(1), L, md, R);
198 P2 = Plantww(d, Veq, Weq, dw, Iw, I(2), L, md, R);
199 P3 = Plantww(d, Veq, Weq, dw, Iw, I(3), L, md, R);
200 P4 = Plantww(d, Veq, Weq, dw, Iw, I(4), L, md, R);
201 P5 = Plantww(d, Veq, Weq, dw, Iw, I(5), L, md, R);
202 P6 = Plantww(d, Veq, Weq, dw, Iw, I(6), L, md, R);
203
204 figure;
205 bodemag(P1, P2, P3, P4, P5, P6);
206 grid on;
207 h_axes = findobj(gcf, 'type', 'axes');
208 xlabel('Frequency', 'FontSize', 12);
209 ylabel('Magnitude', 'FontSize', 12);
210 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
211
212 % size and brightness of grid and size of x & y axis numbers
213 title(...
214 'Frequency Response  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
215  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex');
216
217 h_line = findobj(gcf, 'type', 'line');
218 set(h_line, 'LineWidth', 1.5); % Lines with thicker width for plots
219
220 % Put legend and enhance appearance
221 % Legend bug with subscript, use '\_' instead of '_'
222 [hL, hObj]=legend({'$I = 0.4 \ Kg.m^2$', '$I = 0.5 \ Kg.m^2$', ...
223 '$I = 0.7 \ Kg.m^2$', '$I = 0.9 \ Kg.m^2$', '$I = 1.2 \ Kg.m^2$', ...
224 '$I = 1.7 \ Kg.m^2$'}, 'Interpreter', 'latex');
225 hTL=findobj(hObj, 'type', 'Text'); %
226 set(hTL, 'FontSize', 10); % font size for letters in legend
227 hTL=findobj(hObj, 'type', 'line'); %
228 set(hTL, 'LineWidth', 2); % thickness of lines in legend
229 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.27 0.24]);
230 % distance between lines in legend [x,y,width, height]
231
232
233 %%
234 % Plant model from e_r, e_l to W_r, W_l
235 % Singular and Bode Plots for different values of m
236 %(variations in total mass without changing I-w)

```

```

237
238 % Bode Plot
239
240 md = 0; m = 3.4; % if d = 0;
241 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
242 L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 /\
243 %has to be chosen based on the corresponding AR value (AR_calculation.m)
244 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values /\
245 %has to be chosen based on the corresponding AR value
246 I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
247 [max,min] = I_maxmin(d,Iw,L,md,dw);
248 Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)
249
250 I = [0.424999999999 0.42500 0.425000000001 0.4292 0.4462 0.4675 0.3825];
251
252
253 P1 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m);
254 P2 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+0.5);
255 P3 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1);
256 P4 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1.5);
257 P5 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2);
258 P6 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2.5);
259 P7 = newPlantww(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+3);
260
261 figure;
262 bodemag(P1,P2,P3,P4,P5,P6,P7);
263 grid on;
264 h_axes = findobj(gcf, 'type', 'axes');
265 xlabel('Frequency','FontSize',12);
266 ylabel('Magnititude','FontSize',12);
267 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
268 % size and brightness of grid and size of x & y axis numbers
269 title(...
270 'Frequency Response  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
271  $d \neq 0$ ','FontWeight','bold','FontSize',14, 'Interpreter','latex');
272
273 h_line = findobj(gcf, 'type', 'line');
274 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
275
276 % Put legend and enhance appearance
277 % Legend bug with subscript, use '\_' instead of '_'
278 [hL,hObj]=legend({'$m = 3.4 \ kg$', '$m = 3.9 \ kg$', '$m = 4.4 \ kg$', ...
279 '$m = 4.9 \ kg$', '$m = 5.4 \ kg$', '$m = 5.9 \ kg$'}, 'Interpreter', ...
280 'latex');
281 hTL=findobj(hObj, 'type', 'Text'); %
282 set(hTL, 'FontSize',11); % font size for letters in legend
283 hTL=findobj(hObj, 'type', 'line'); %
284 set(hTL, 'LineWidth',2); % thickness of lines in legend
285 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.25 0.26]);
286
287 %% Singular Values Plot
288 winit = -1;
289 wfin = 2;
290 nwpts = 200;
291 w = logspace(winit,wfin,nwpts);
292 P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
293 P4 = sigma(P4,w); P5 = sigma(P5,w);

```

```

294 P6 = sigma(P6,w); P7 = sigma(P7,w);
295 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
296 P4 = 20*log10(P4); P5 = 20*log10(P5);
297 P6 = 20*log10(P6); P7 = 20*log10(P7);
298 figure;
299 subplot(2,1,1);
300 semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
301 w, P6(1,:))
302 %clear sv
303 grid on;
304 h_axes = findobj(gcf, 'type', 'axes');
305 xlabel('Frequency','FontSize',12);
306 ylabel('Magnititude','FontSize',12);
307 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
308 % size and brightness of grid and size of x & y axis numbers
309 title(...
310 'Max Singular Values  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
311  $\$d \neq 0\$, 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
312
313 h_line = findobj(gcf, 'type', 'line');
314 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
315
316 subplot(2,1,2);
317 semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
318 w, P6(2,:))
319 %clear sv
320 grid on;
321 h_axes = findobj(gcf, 'type', 'axes');
322 xlabel('Frequency','FontSize',12);
323 ylabel('Magnititude','FontSize',12);
324 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
325 % size and brightness of grid and size of x & y axis numbers
326 title(...
327 'Min Singular Values  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
328  $\$d \neq 0\$, 'FontWeight', 'bold', 'FontSize',12, 'Interpreter', 'latex')
329
330 h_line = findobj(gcf, 'type', 'line');
331 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
332
333
334
335
336 % Put legend and enhance appearance
337 % Legend bug with subscript, use '\_' instead of '_'
338 [hL,hObj]=legend({'$m = 3.4 \ kg$', '$m = 3.9 \ kg$', '$m = 4.4 \ kg$', ...
339 '$m = 4.9 \ kg$', '$m = 5.4 \ kg$', '$m = 5.9 \ kg$'}, 'Interpreter', ...
340 'latex');
341 hTL=findobj(hObj, 'type', 'Text'); %
342 set(hTL, 'FontSize',11); % font size for letters in legend
343 hTL=findobj(hObj, 'type', 'line'); %
344 set(hTL, 'LineWidth',1.2); % thickness of lines in legend
345 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.25 0.26]);
346 % distance between lines in legend [x,y,width, height]
347
348
349 %%
350 % Plant model from e_r, e_l to W_r, W_l$$ 
```

```

351 % Singular and Bode Plots for different values of R
352
353 % Bode Plot
354
355 % change in R results in change in IW, however, no significant
356 % difference is observed
357
358 md = 0; m = 3.4; % if d = 0;
359 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
360 L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
361 %has to be chosen based on the corresponding AR value (ARcalculation.m)
362 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\\
363 %has to be chosen based on the corresponding AR value
364 I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
365 [max,min] = I_maxmin(d,Iw,L,md,dw);
366 Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)
367
368 I = [0.424999999999 0.42500 0.425000000001 0.4292 0.4462 0.4675 0.3825];
369
370 R = 0.042; m_wheel = 0.096;
371 rm = 0.0248 ; m_motor = 0.224;
372 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
373 I = I_Newcalculation(0,Iw,L,md,dw);
374
375 P1 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
376 R = R+0.01;
377 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
378 I = I_Newcalculation(0,Iw,L,md,dw);
379 P2 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
380 R = R+0.01;
381 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
382 I = I_Newcalculation(0,Iw,L,md,dw);
383 P3 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
384 R = R+0.01;
385 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
386 I = I_Newcalculation(0,Iw,L,md,dw);
387 P4 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
388 R = R+0.01;
389 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
390 I = I_Newcalculation(0,Iw,L,md,dw);
391 P5 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
392 R = R+0.01;
393 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
394 I = I_Newcalculation(0,Iw,L,md,dw);
395 P6 = newPlantww(d, Veq, Weq, dw, Iw, I, L, md,R,m);
396
397
398 R = 0.042; Iw = 1.67e-06;
399 figure;
400 bodemag(P1,P2,P3,P4,P5,P6);
401 grid on;
402 h_axes = findobj(gcf, 'type', 'axes');
403 xlabel('Frequency','FontSize',12);
404 ylabel('Magnitude','FontSize',12);
405 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
406 % size and brightness of grid and size of x & y axis numbers
407 title(...)

```

```

408 'Frequency Response  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
409  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex');
410
411 h_line = findobj(gcf, 'type', 'line');
412 set(h_line, 'LineWidth', 1.5); % Lines with thicker width for plots
413
414 % Put legend and enhance appearance
415 % Legend bug with subscript, use '\_' instead of '_'
416 [hL, hObj]=legend({'$R = 0.042 \ m$', '$R = 0.052 \ m$', ...
417 '$R = 0.062 \ m$', '$R = 0.072 \ m$', '$R = 0.082 \ m$', ...
418 '$R = 0.092 \ m$'}, 'Interpreter', 'latex');
419 hTL=findobj(hObj, 'type', 'Text'); %
420 set(hTL, 'FontSize', 11); % font size for letters in legend
421 hTL=findobj(hObj, 'type', 'line'); %
422 set(hTL, 'LineWidth', 2); % thickness of lines in legend
423 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.25 0.26]);
424 % distance between lines in legend [x,y,width, height]
425
426 %% Singular Values Plot
427 winit = -1;
428 wfin = 2;
429 nwpts = 200;
430 w = logspace(winit, wfin, nwpts);
431 P1 = sigma(P1, w); P2 = sigma(P2, w); P3 = sigma(P3, w);
432 P4 = sigma(P4, w); P5 = sigma(P5, w);
433 P6 = sigma(P6, w);
434 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
435 P4 = 20*log10(P4); P5 = 20*log10(P5);
436 P6 = 20*log10(P6);
437 figure;
438 subplot(2,1,1);
439 semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:), ...
440 w, P6(1,:))
441 %clear sv
442 grid on;
443 h_axes = findobj(gcf, 'type', 'axes');
444 xlabel('Frequency', 'FontSize', 12);
445 ylabel('Magnitude', 'FontSize', 12);
446 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
447 % size and brightness of grid and size of x & y axis numbers
448 title(...
449 'Max Singular Values  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
450  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex')
451
452 h_line = findobj(gcf, 'type', 'line');
453 set(h_line, 'LineWidth', 1.2); % Lines with thicker width for plots
454
455 subplot(2,1,2);
456 semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:), ...
457 w, P6(2,:))
458 %clear sv
459 grid on;
460 h_axes = findobj(gcf, 'type', 'axes');
461 xlabel('Frequency', 'FontSize', 12);
462 ylabel('Magnitude', 'FontSize', 12);
463 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
464 % size and brightness of grid and size of x & y axis numbers

```

```

465 title(...)
466 'Min Singular Values  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
467  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize', 12, 'Interpreter', 'latex')
468
469 h_line = findobj(gcf, 'type', 'line');
470 set(h_line, 'LineWidth', 1.2); % Lines with thicker width for plots
471
472
473
474
475 % Put legend and enhance appearance
476 % Legend bug with subscript, use '\_' instead of '_'
477 [hL, hObj]=legend({'$R = 0.042 \ m$', '$R = 0.052 \ m$', ...
478 '$R = 0.062 \ m$', '$R = 0.072 \ m$', '$R = 0.082 \ m$', ...
479 '$R = 0.092 \ m$'}, 'Interpreter', 'latex');
480 hTL=findobj(hObj, 'type', 'Text'); %
481 set(hTL, 'FontSize', 11); % font size for letters in legend
482 hTL=findobj(hObj, 'type', 'line'); %
483 set(hTL, 'LineWidth', 1.2); % thickness of lines in legend
484 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.25 0.26]);
485 % distance between lines in legend [x,y,width, height]

1 % Trade Studies at  $d \sim 0$ 
2 %continuation of the ppt_2.1.d.nonzero with the singular value plots
3 clc
4 close all
5 clear all
6 s = tf([1 0], [1]);
7 md = 0; m = 3.4;
8 winit = -1;
9 wfin = 2;
10 nwpts = 200;
11 w = logspace(winit, wfin, nwpts);
12 %% Different Plant Models with the respective parameters as input
13 % at  $d = 0$ 
14 % Plant model from  $e_r, e_l$  to  $W_r, W_l$  decoupled
15 % Plant model from  $(e_r + e_l), (e_r - e_l)$  to  $V, W$  decoupled
16 %%
17 % Plant model from  $e_r, e_l$  to  $W_r, W_l$ 
18 % Singular and Bode Plots for different values of  $Ve_q$ 
19 d = 0.1;  $Ve_q = 2$ ;  $We_q = 0.8$ ; % in m/s max value is 0.14 for hardware
20 L = 0.3536;  $dw = L/\sqrt{2}$ ;  $R = 0.042$ ; % default values  $L = 0.3536$  //\
21 %has to be chosen based on the corresponding AR value (AR_calculation.m)
22  $Iw = 1.67e-06$ ;  $A = m + 2*Iw/(R*R)$ ; % default values //\
23 %has to be chosen based on the corresponding AR value
24 I = INewcalculation(d, Iw, L, md, dw);
25 [max, min] = Imaxmin(d, Iw, L, md, dw);
26 Plant1 = Plantww(d,  $Ve_q$ ,  $We_q$ , dw, Iw, I, L, md, R);
27
28  $Ve_q = [0.1 0.2 0.6 1 3 5]$ ;
29
30 P1 = Plantww(d,  $Ve_q(1)$ ,  $We_q$ , dw, Iw, I, L, md, R);
31 P2 = Plantww(d,  $Ve_q(2)$ ,  $We_q$ , dw, Iw, I, L, md, R);
32 P3 = Plantww(d,  $Ve_q(3)$ ,  $We_q$ , dw, Iw, I, L, md, R);
33 P4 = Plantww(d,  $Ve_q(4)$ ,  $We_q$ , dw, Iw, I, L, md, R);
34 P5 = Plantww(d,  $Ve_q(5)$ ,  $We_q$ , dw, Iw, I, L, md, R);

```



```

35 P6 = Plantww(d, Veq(6), Weq, dw, Iw, I, L, md,R);
36
37 P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
38 P4 = sigma(P4,w); P5 = sigma(P5,w);
39 P6 = sigma(P6,w);
40 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
41 P4 = 20*log10(P4); P5 = 20*log10(P5);
42 P6 = 20*log10(P6);
43
44 figure;
45 subplot(2,1,1);
46 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
47 w, P6(1,:))
48 %clear sv
49 grid on;
50 h_axes = findobj(gcf, 'type', 'axes');
51 xlabel('Frequency','FontSize',12);
52 ylabel('Magnitude','FontSize',12);
53 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
54 % size and brightness of grid and size of x & y axis numbers
55 title(...
56 'Max Singular Values  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
57  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
58
59 h_line = findobj(gcf, 'type', 'line');
60 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
61
62 subplot(2,1,2);
63 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
64 w, P6(2,:))
65 %clear sv
66 grid on;
67 h_axes = findobj(gcf, 'type', 'axes');
68 xlabel('Frequency','FontSize',12);
69 ylabel('Magnitude','FontSize',12);
70 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
71 % size and brightness of grid and size of x & y axis numbers
72 title(...
73 'Min Singular Values  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
74  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize',12, 'Interpreter', 'latex')
75
76 h_line = findobj(gcf, 'type', 'line');
77 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
78
79 % Put legend and enhance appearance
80 % Legend bug with subscript, use '\_' instead of '-'
81 [hL,hObj]=legend({'$v_{eq} = 0.1 \ m/s$', '$v_{eq} = 0.2 \ m/s$', ...
82 '$v_{eq} = 0.6 \ m/s$', '$v_{eq} = 1.0 \ m/s$', '$v_{eq} = 3.0 \ m/s$', ...
83 '$v_{eq} = 5.0 \ m/s$'}, 'Interpreter', 'latex');
84 hTL=findobj(hObj, 'type', 'Text'); %
85 set(hTL, 'FontSize',11); % font size for letters in legend
86 hTL=findobj(hObj, 'type', 'line'); %
87 set(hTL, 'LineWidth',2); % thickness of lines in legend
88 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.26 0.24]);
89 % distance between lines in legend [x,y,width, height]
90
91 %% Singular and Bode Plots for different values of Weq

```

```

92 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
93 L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
94 %has to be chosen based on the corresponding AR value (AR_calculation.m)
95 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\\
96 %has to be chosen based on the corresponding AR value
97 I = I_Newcalculation(d, Iw, L, md, dw);
98 [max, min] = Imaxmin(d, Iw, L, md, dw);
99
100 Weq = [-8.0 -2.5 -0.5 0.5 2.5 8.0];
101
102 P1 = Plantww(d, Veq, Weq(1), dw, Iw, I, L, md, R);
103 P2 = Plantww(d, Veq, Weq(2), dw, Iw, I, L, md, R);
104 P3 = Plantww(d, Veq, Weq(3), dw, Iw, I, L, md, R);
105 P4 = Plantww(d, Veq, Weq(4), dw, Iw, I, L, md, R);
106 P5 = Plantww(d, Veq, Weq(5), dw, Iw, I, L, md, R);
107 P6 = Plantww(d, Veq, Weq(6), dw, Iw, I, L, md, R);
108
109 P1 = sigma(P1, w); P2 = sigma(P2, w); P3 = sigma(P3, w);
110 P4 = sigma(P4, w); P5 = sigma(P5, w);
111 P6 = sigma(P6, w);
112 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
113 P4 = 20*log10(P4); P5 = 20*log10(P5);
114 P6 = 20*log10(P6);
115
116 figure;
117 subplot(2,1,1);
118 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:), ...
119 w, P6(1,:))
120 %clear sv
121 grid on;
122 h_axes = findobj(gcf, 'type', 'axes');
123 xlabel('Frequency', 'FontSize', 12);
124 ylabel('Magnitude', 'FontSize', 12);
125 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
126 % size and brightness of grid and size of x & y axis numbers
127 title(...
128 'Max Singular Values  $e_r, e_l$   $\rightarrow$   $\omega_r, \omega_l$  for
129  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex')
130
131 h_line = findobj(gcf, 'type', 'line');
132 set(h_line, 'LineWidth', 1.5); % Lines with thicker width for plots
133
134 subplot(2,1,2);
135 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:), ...
136 w, P6(2,:))
137 %clear sv
138 grid on;
139 h_axes = findobj(gcf, 'type', 'axes');
140 xlabel('Frequency', 'FontSize', 12);
141 ylabel('Magnitude', 'FontSize', 12);
142 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
143 % size and brightness of grid and size of x & y axis numbers
144 title(...
145 'Min Singular Values  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
146  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize', 12, 'Interpreter', 'latex')
147
148 h_line = findobj(gcf, 'type', 'line');

```

```

149 set(h_line, 'LineWidth',1.5);           % Lines with thicker width for plots
150
151 % Put legend and enhance appearance
152 % Legend bug with subscript, use '\_' instead of '_'
153 [hL,hObj]=legend({'$\omega_{eq} = -8.0 \ rad/s$',...
154 '$\omega_{eq} = -2.5 \ rad/s$', '$\omega_{eq} = -0.5 \ rad/s$',...
155 '$\omega_{eq} = 0.5 \ rad/s$', '$\omega_{eq} = 2.5 \ rad/s$',...
156 '$\omega_{eq} = 8.0 \ rad/s$'}, 'Interpreter', 'latex');
157 hTL=findobj(hObj, 'type', 'Text');      %
158 set(hTL, 'FontSize',10);                % font size for letters in legend
159 hTL=findobj(hObj, 'type', 'line');      %
160 set(hTL, 'LineWidth',2);                % thickness of lines in legend
161 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.28 0.24]);
162 % distance between lines in legend [x,y,width, height]
163
164 %% %% Singular and Bode Plots for different values of d
165 % the behaviour in the bode plots can be associated with the dominat...
166 pole
167 % variation wrt to d
168 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
169 L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\
170 %has to be chosen based on the corresponding AR value (AR_calculation.m)
171 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\
172 %has to be chosen based on the corresponding AR value
173 I = I.Newcalculation(d, Iw, L, md, dw);
174 [max, min] = Imaxmin(d, Iw, L, md, dw);
175
176 d = [-0.09 -0.08 -0.04 0.04 0.08 0.09];
177 I = [I.Newcalculation(d(1), Iw, L, md, dw) I.Newcalculation(d(2), Iw, L, md, dw)
178      I.Newcalculation(d(3), Iw, L, md, dw) I.Newcalculation(d(4), Iw, L, md, dw)
179      I.Newcalculation(d(5), Iw, L, md, dw)
180      I.Newcalculation(d(6), Iw, L, md, dw)];
181
182 P1 = Plantww(d(1), Veq, Weq, dw, Iw, I(1), L, md, R);
183 P2 = Plantww(d(2), Veq, Weq, dw, Iw, I(2), L, md, R);
184 P3 = Plantww(d(3), Veq, Weq, dw, Iw, I(3), L, md, R);
185 P4 = Plantww(d(4), Veq, Weq, dw, Iw, I(4), L, md, R);
186 P5 = Plantww(d(5), Veq, Weq, dw, Iw, I(5), L, md, R);
187 P6 = Plantww(d(6), Veq, Weq, dw, Iw, I(6), L, md, R);
188
189 P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
190 P4 = sigma(P4,w); P5 = sigma(P5,w);
191 P6 = sigma(P6,w);
192 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
193 P4 = 20*log10(P4); P5 = 20*log10(P5);
194 P6 = 20*log10(P6);
195
196 figure;
197 subplot(2,1,1);
198 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:)...
199 w, P6(1,:))
200 %clear sv
201 grid on;
202 h_axes = findobj(gcf, 'type', 'axes');
203 xlabel('Frequency', 'FontSize',12);
204 ylabel('Magnititude', 'FontSize',12);
205 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);

```

```

206 % size and brightness of grid and size of x & y axis numbers
207 title(...
208 'Max Singular Values  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  $ for
209 $d \neq 0$, 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex')
210
211 h_line = findobj(gcf, 'type', 'line');
212 set(h_line, 'LineWidth', 1.5); % Lines with thicker width for plots
213
214 subplot(2,1,2);
215 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:), ...
216 w, P6(2,:))
217 %clear sv
218 grid on;
219 h_axes = findobj(gcf, 'type', 'axes');
220 xlabel('Frequency', 'FontSize', 12);
221 ylabel('Magnitude', 'FontSize', 12);
222 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
223 % size and brightness of grid and size of x & y axis numbers
224 title(...
225 'Min Singular Values  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  $ for
226 $d \neq 0$, 'FontWeight', 'bold', 'FontSize', 12, 'Interpreter', 'latex')
227
228 h_line = findobj(gcf, 'type', 'line');
229 set(h_line, 'LineWidth', 1.5); % Lines with thicker width for plots
230
231 % Put legend and enhance appearance
232 % Legend bug with subscript, use '\_' instead of '_'
233 [hL, hObj]=legend({'$d = -0.1 \ m$', '$d = -0.05 \ m$', ...
234 '$d = -0.02 \ m$', '$d = 0.02 \ m$', '$d = 0.05 \ m$', '$d = 0.1 \ m$'}, ...
235 'Interpreter', 'latex');
236 hTL=findobj(hObj, 'type', 'Text'); %
237 set(hTL, 'FontSize', 10); % font size for letters in legend
238 hTL=findobj(hObj, 'type', 'line'); %
239 set(hTL, 'LineWidth', 2); % thickness of lines in legend
240 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.26 0.24]);
241 % distance between lines in legend [x,y,width, height]
242
243 %% %% Singular and Bode Plots for different values of I
244 % the behaviour in the bode plots can be associated with the dominant
245 % pole variation wrt to d
246 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
247 L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\
248 %has to be chosen based on the corresponding AR value (AR_calculation.m)
249 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\
250 %has to be chosen based on the corresponding AR value
251 I = I.Newcalculation(d, Iw, L, md, dw);
252 [max, min] = Imaxmin(d, Iw, L, md, dw);
253
254 I = [0.4 0.5 0.7 0.9 1.2 1.7];
255
256 P1 = Plantww(d, Veq, Weq, dw, Iw, I(1), L, md, R);
257 P2 = Plantww(d, Veq, Weq, dw, Iw, I(2), L, md, R);
258 P3 = Plantww(d, Veq, Weq, dw, Iw, I(3), L, md, R);
259 P4 = Plantww(d, Veq, Weq, dw, Iw, I(4), L, md, R);
260 P5 = Plantww(d, Veq, Weq, dw, Iw, I(5), L, md, R);
261 P6 = Plantww(d, Veq, Weq, dw, Iw, I(6), L, md, R);
262

```

```

263 P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
264 P4 = sigma(P4,w); P5 = sigma(P5,w);
265 P6 = sigma(P6,w);
266 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
267 P4 = 20*log10(P4); P5 = 20*log10(P5);
268 P6 = 20*log10(P6);
269
270 figure;
271 subplot(2,1,1);
272 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
273 w, P6(1,:))
274 %clear sv
275 grid on;
276 h_axes = findobj(gcf, 'type', 'axes');
277 xlabel('Frequency','FontSize',12);
278 ylabel('Magnitude','FontSize',12);
279 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
280 % size and brightness of grid and size of x & y axis numbers
281 title(...
282 'Max Singular Values  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
283  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
284
285 h_line = findobj(gcf, 'type', 'line');
286 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
287
288 subplot(2,1,2);
289 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
290 w, P6(2,:))
291 %clear sv
292 grid on;
293 h_axes = findobj(gcf, 'type', 'axes');
294 xlabel('Frequency','FontSize',12);
295 ylabel('Magnitude','FontSize',12);
296 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
297 % size and brightness of grid and size of x & y axis numbers
298 title(...
299 'Min Singular Values  $(e_r, e_l) \rightarrow (\omega_r, \omega_l)$  for
300  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize',12, 'Interpreter', 'latex')
301
302 h_line = findobj(gcf, 'type', 'line');
303 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
304
305 % Put legend and enhance appearance
306 % Legend bug with subscript, use '\_' instead of '_'
307 [hL,hObj]=legend({'$I = 0.4 \ Kg.m^2$', '$I = 0.5 \ Kg.m^2$', ...
308 '$I = 0.7 \ Kg.m^2$', '$I = 0.9 \ Kg.m^2$', '$I = 1.2 \ Kg.m^2$', ...
309 '$I = 1.7 \ Kg.m^2$'}, 'Interpreter', 'latex');
310 hTL=findobj(hObj, 'type', 'Text'); %
311 set(hTL, 'FontSize',10); % font size for letters in legend
312 hTL=findobj(hObj, 'type', 'line'); %
313 set(hTL, 'LineWidth',2); % thickness of lines in legend
314 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.27 0.24]);
315 % distance between lines in legend [x,y,width, height]
316
317 %%
318 % Plant model from  $e_r + e_l$ ,  $e_r - e_l$  to V,W
319 % Singular and Bode Plots for different values of m

```

```

320 $(variations in total mass without changing I_w)
321
322 % Bode Plot
323
324 md = 0; m = 3.4; % if d = 0;
325 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
326 L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 /\
327 %has to be chosen based on the corresponding AR value (AR_calculation.m)
328 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values /\
329 %has to be chosen based on the corresponding AR value
330 I_AR = I_ARcalculation(d,Iw,L,A,R,dw);
331 [max,min] = Imaxmin(d,Iw,L,md,dw);
332 Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md,R)
333
334 I = [0.424999999999 0.42500 0.425000000001 0.4292 0.4462 0.4675 0.3825];
335
336
337 P1 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m);
338 P2 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+0.5);
339 P3 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1);
340 P4 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+1.5);
341 P5 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2);
342 P6 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+2.5);
343 P7 = newPlantsdv(d, Veq, Weq, dw, Iw, I(5), L, md,R,m+3);
344
345 figure;
346 bodemag(P1,P2,P3,P4,P5,P6);
347 grid on;
348 h_axes = findobj(gcf, 'type', 'axes');
349 xlabel('Frequency','FontSize',12);
350 ylabel('Magnitude','FontSize',12);
351 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
352 % size and brightness of grid and size of x & y axis numbers
353 title(...
354 'Frequency Response  $(e_r + e_l, e_r - e_l)$   $\rightarrow$  for
355  $d \neq 0$ , 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
356
357 h_line = findobj(gcf, 'type', 'line');
358 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
359
360 % Put legend and enhance appearance
361 % Legend bug with subscript, use '\_' instead of '_'
362 [hL,hObj]=legend({'$m = 3.4 \ kg$', '$m = 3.9 \ kg$', '$m = 4.4 \ kg$', ...
363 '$m = 4.9 \ kg$', '$m = 5.4 \ kg$', '$m = 5.9 \ kg$'}, 'Interpreter', ...
364 'latex');
365 hTL=findobj(hObj, 'type', 'Text'); %
366 set(hTL, 'FontSize',11); % font size for letters in legend
367 hTL=findobj(hObj, 'type', 'line'); %
368 set(hTL, 'LineWidth',2); % thickness of lines in legend
369 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.25 0.26]);
370 % distance between lines in legend [x,y,width, height]
371
372 %% Singular Values Plot
373 winit = -1;
374 wfin = 2;
375 nwpts = 200;
376 w = logspace(winit,wfin,nwpts);

```

```

377 P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
378 P4 = sigma(P4,w); P5 = sigma(P5,w);
379 P6 = sigma(P6,w); P7 = sigma(P7,w);
380 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
381 P4 = 20*log10(P4); P5 = 20*log10(P5);
382 P6 = 20*log10(P6); P7 = 20*log10(P7);
383 figure;
384 subplot(2,1,1);
385 semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
386 w, P6(1,:))
387 %clear sv
388 grid on;
389 h_axes = findobj(gcf, 'type', 'axes');
390 xlabel('Frequency','FontSize',12);
391 ylabel('Magnitude','FontSize',12);
392 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
393 % size and brightness of grid and size of x & y axis numbers
394 title(...
395 'Max Singular Values  $((e_r + e_l, e_r - e_l) \rightarrow (v, \omega))$  for
396  $\delta \neq 0$ ', 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
397
398 h_line = findobj(gcf, 'type', 'line');
399 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
400
401 subplot(2,1,2);
402 semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
403 w, P6(2,:))
404 %clear sv
405 grid on;
406 h_axes = findobj(gcf, 'type', 'axes');
407 xlabel('Frequency','FontSize',12);
408 ylabel('Magnitude','FontSize',12);
409 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
410 % size and brightness of grid and size of x & y axis numbers
411 title(...
412 'Min Singular Values  $(e_r + e_l, e_r - e_l) \rightarrow (v, \omega)$  for
413  $\delta \neq 0$ ', 'FontWeight', 'bold', 'FontSize',12, 'Interpreter', 'latex')
414
415 h_line = findobj(gcf, 'type', 'line');
416 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
417
418
419
420
421 % Put legend and enhance appearance
422 % Legend bug with subscript, use '\_' instead of '_'
423 [hL,hObj]=legend({'$m = 3.4 \ kg$', '$m = 3.9 \ kg$', '$m = 4.4 \ kg$', ...
424 '$m = 4.9 \ kg$', '$m = 5.4 \ kg$', '$m = 5.9 \ kg$'}, 'Interpreter', ...
425 'latex');
426 hTL=findobj(hObj, 'type', 'Text'); %
427 set(hTL, 'FontSize',11); % font size for letters in legend
428 hTL=findobj(hObj, 'type', 'line'); %
429 set(hTL, 'LineWidth',1.2); % thickness of lines in legend
430 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.25 0.26]);
431 % distance between lines in legend [x,y,width, height]
432
433 %%

```

```

434 % Plant model from e_r+ e_l, e_r+ e_l to V, W
435 % Singular and Bode Plots for different values of R
436
437 % Bode Plot
438
439 % change in R results in change in IW, however, no significant
440 % difference is observed
441
442 md = 0; m = 3.4; % if d = 0;
443 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
444 L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 /\
445 %has to be chosen based on the corresponding AR value (AR_calculation.m)
446 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values /\
447 %has to be chosen based on the corresponding AR value
448 I_AR = I_ARcalculation(d, Iw, L, A, R, dw);
449 [max, min] = I_maxmin(d, Iw, L, md, dw);
450 Plant1 = Plantww(d, Veq, Weq, dw, Iw, I_AR, L, md, R)
451
452 I = [0.424999999999 0.42500 0.425000000001 0.4292 0.4462 0.4675 0.3825];
453
454 R = 0.042; m_wheel = 0.096;
455 rm = 0.0248 ; m_motor = 0.224;
456 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
457 I = I_Newcalculation(0, Iw, L, md, dw);
458
459 P1 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md, R, m);
460 R = R+0.01;
461 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
462 I = I_Newcalculation(0, Iw, L, md, dw);
463 P2 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md, R, m);
464 R = R+0.01;
465 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
466 I = I_Newcalculation(0, Iw, L, md, dw);
467 P3 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md, R, m);
468 R = R+0.01;
469 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
470 I = I_Newcalculation(0, Iw, L, md, dw);
471 P4 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md, R, m);
472 R = R+0.01;
473 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
474 I = I_Newcalculation(0, Iw, L, md, dw);
475 P5 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md, R, m);
476 R = R+0.01;
477 Iw = 0.5*m_motor*rm*rm + 0.5*m_wheel*R*R;
478 I = I_Newcalculation(0, Iw, L, md, dw);
479 P6 = newPlantsdv(d, Veq, Weq, dw, Iw, I, L, md, R, m);
480
481
482 R = 0.042; Iw = 1.67e-06;
483 figure;
484 bodemag(P1, P2, P3, P4, P5, P6);
485 grid on;
486 h_axes = findobj(gcf, 'type', 'axes');
487 xlabel('Frequency', 'FontSize', 12);
488 ylabel('Magnitude', 'FontSize', 12);
489 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
490 % size and brightness of grid and size of x & y axis numbers

```



```

491 title(...
492 'Frequency Response  $(e_r + e_l, e_r - e_l) \rightarrow (v, \omega)$  for
493  $\$d \neq 0\$$ ', 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex')
494
495 h_line = findobj(gcf, 'type', 'line');
496 set(h_line, 'LineWidth', 1.5); % Lines with thicker width for plots
497
498 % Put legend and enhance appearance
499 % Legend bug with subscript, use '\_' instead of '_'
500 [hL,hObj]=legend({'$R = 0.042 \ m$', '$R = 0.052 \ m$', ...
501 '$R = 0.062 \ m$', '$R = 0.072 \ m$', '$R = 0.082 \ m$', ...
502 '$R = 0.092 \ m$'}, 'Interpreter', 'latex');
503 hTL=findobj(hObj, 'type', 'Text'); %
504 set(hTL, 'FontSize', 11); % font size for letters in legend
505 hTL=findobj(hObj, 'type', 'line'); %
506 set(hTL, 'LineWidth', 2); % thickness of lines in legend
507 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.25 0.26]);
508 % distance between lines in legend [x,y,width, height]
509
510 %% Singular Values Plot
511 winit = -1;
512 wfin = 2;
513 nwpts = 200;
514 w = logspace(winit, wfin, nwpts);
515 P1 = sigma(P1, w); P2 = sigma(P2, w); P3 = sigma(P3, w);
516 P4 = sigma(P4, w); P5 = sigma(P5, w);
517 P6 = sigma(P6, w);
518 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
519 P4 = 20*log10(P4); P5 = 20*log10(P5);
520 P6 = 20*log10(P6);
521 figure;
522 subplot(2,1,1);
523 semilogx( w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:), ...
524 w, P6(1,:))
525 %clear sv
526 grid on;
527 h_axes = findobj(gcf, 'type', 'axes');
528 xlabel('Frequency', 'FontSize', 12);
529 ylabel('Magnitude', 'FontSize', 12);
530 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
531 % size and brightness of grid and size of x & y axis numbers
532 title(...
533 'Max Singular Values  $(e_r + e_l, e_r - e_l) \rightarrow (v, \omega)$  for
534  $\$d \neq 0\$$ ', 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex')
535
536 h_line = findobj(gcf, 'type', 'line');
537 set(h_line, 'LineWidth', 1.2); % Lines with thicker width for plots
538
539 subplot(2,1,2);
540 semilogx( w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:), ...
541 w, P6(2,:))
542 %clear sv
543 grid on;
544 h_axes = findobj(gcf, 'type', 'axes');
545 xlabel('Frequency', 'FontSize', 12);
546 ylabel('Magnitude', 'FontSize', 12);
547 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);

```

```

548 % size and brightness of grid and size of x & y axis numbers
549 title(...)
550 'Min Singular Values  $(e_r + e_l, e_r - e_l) \rightarrow (v, \omega)$  for
551  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize', 12, 'Interpreter', 'latex')
552
553 h_line = findobj(gcf, 'type', 'line');
554 set(h_line, 'LineWidth', 1.2); % Lines with thicker width for plots
555
556
557
558
559 % Put legend and enhance appearance
560 % Legend bug with subscript, use '\_' instead of '_'
561 [hL, hObj]=legend({'$R = 0.042 \ m$', '$R = 0.052 \ m$', ...
562 '$R = 0.062 \ m$', '$R = 0.072 \ m$', '$R = 0.082 \ m$', ...
563 '$R = 0.092 \ m$'}, 'Interpreter', 'latex');
564 hTL=findobj(hObj, 'type', 'Text'); %
565 set(hTL, 'FontSize', 11); % font size for letters in legend
566 hTL=findobj(hObj, 'type', 'line'); %
567 set(hTL, 'LineWidth', 1.2); % thickness of lines in legend
568 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.25 0.26]);
569 % distance between lines in legend [x,y,width, height]

1 % Trade Studies at  $d \sim 0$  Contd.
2 clc
3 close all
4 clear all
5 s = tf([1 0], [1]);
6 md = 0; m = 3.4; % if d = 0;
7 %% Different Plant Models with the respective parameters as input
8 % at d = 0
9 % Plant model from  $e_r, e_l$  to  $W_r, W_l$  decoupled
10 % Plant model from  $(e_r + e_l), (e_r - e_l)$  to  $V, W$  decoupled
11 %%
12 % Plant model from  $(e_r + e_l), (e_r - e_l)$  to  $V, W$  decoupled
13 % Singular and Bode Plots for different values of  $Ve_q$ 
14 d = 0.1;  $Ve_q = 0$ ;  $We_q = 0.8$ ; % in m/s max value is 0.14 for hardware
15 L = 0.3536;  $dw = L/\sqrt{2}$ ;  $R = 0.042$ ; % default values  $L = 0.3536$  //\
16 %has to be chosen based on the corresponding AR value (AR-calculation.m)
17  $Iw = 1.67e-06$ ;  $A = m + 2*Iw/(R*R)$ ; % default values //\
18 %has to be chosen based on the corresponding AR value
19 I = INewcalculation(d, Iw, L, md, dw);
20 [max, min] = Imaxmin(d, Iw, L, md, dw);
21 Plant1 = Plantsdv(d,  $Ve_q$ ,  $We_q$ , dw, Iw, I, L, md, R);
22
23  $Ve_q = [0.1 0.2 0.6 1 3 5]$ ;
24
25 P1 = Plantsdv(d,  $Ve_q(1)$ ,  $We_q$ , dw, Iw, I, L, md, R);
26 P2 = Plantsdv(d,  $Ve_q(2)$ ,  $We_q$ , dw, Iw, I, L, md, R);
27 P3 = Plantsdv(d,  $Ve_q(3)$ ,  $We_q$ , dw, Iw, I, L, md, R);
28 P4 = Plantsdv(d,  $Ve_q(4)$ ,  $We_q$ , dw, Iw, I, L, md, R);
29 P5 = Plantsdv(d,  $Ve_q(5)$ ,  $We_q$ , dw, Iw, I, L, md, R);
30 P6 = Plantsdv(d,  $Ve_q(6)$ ,  $We_q$ , dw, Iw, I, L, md, R);
31
32 figure;
33 bodemag(P1, P2, P3, P4, P5, P6);

```

```

34 grid on;
35 h_axes = findobj(gcf, 'type', 'axes');
36 xlabel('Frequency','FontSize',12);
37 ylabel('Magnitide','FontSize',12);
38 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
39 % size and brightness of grid and size of x & y axis numbers
40 title(...
41 'Frequency Response  $(e_r + e_l, e_r - e_l)$ \rightarrow( $v, \omega$ )$ for
42  $d \neq 0$ $', 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
43
44 h_line = findobj(gcf, 'type', 'line');
45 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
46
47 % Put legend and enhance appearance
48 % Legend bug with subscript, use '\_' instead of '_'
49 [hL,hObj]=legend({'$v_{eq} = 0.1 \ m/s$', '$v_{eq} = 0.2 \ m/s$', ...
50 '$v_{eq} = 0.6 \ m/s$', '$v_{eq} = 1.0 \ m/s$', ...
51 '$v_{eq} = 3.0 \ m/s$', '$v_{eq} = 5.0 \ m/s$'}, 'Interpreter', 'latex');
52 hTL=findobj(hObj, 'type', 'Text'); %
53 set(hTL, 'FontSize',11); % font size for letters in legend
54 hTL=findobj(hObj, 'type', 'line'); %
55 set(hTL, 'LineWidth',2); % thickness of lines in legend
56 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.26 0.24]);
57 % distance between lines in legend [x,y,width, height]
58
59 %% Singular and Bode Plots for different values of Weq
60 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
61 L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\
62 %has to be chosen based on the corresponding AR value (AR.calculation.m)
63 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\
64 %has to be chosen based on the corresponding AR value
65 I = I.Newcalculation(d, Iw, L, md, dw);
66 [max, min] = Imaxmin(d, Iw, L, md, dw);
67
68 Weq = [-8.0 -2.5 -0.5 0.5 2.5 8.0];
69
70 P1 = Plantsdv(d, Veq, Weq(1), dw, Iw, I, L, md, R);
71 P2 = Plantsdv(d, Veq, Weq(2), dw, Iw, I, L, md, R);
72 P3 = Plantsdv(d, Veq, Weq(3), dw, Iw, I, L, md, R);
73 P4 = Plantsdv(d, Veq, Weq(4), dw, Iw, I, L, md, R);
74 P5 = Plantsdv(d, Veq, Weq(5), dw, Iw, I, L, md, R);
75 P6 = Plantsdv(d, Veq, Weq(6), dw, Iw, I, L, md, R);
76
77 figure;
78 bodemag(P1,P2,P3,P4,P5,P6);
79 grid on;
80 h_axes = findobj(gcf, 'type', 'axes');
81 xlabel('Frequency','FontSize',12);
82 ylabel('Magnitide','FontSize',12);
83 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
84 % size and brightness of grid and size of x & y axis numbers
85 title(...
86 'Frequency Response  $(e_r + e_l, e_r - e_l)$ \rightarrow( $v, \omega$ )$ for
87  $d \neq 0$ $ ', 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
88
89 h_line = findobj(gcf, 'type', 'line');
90 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots

```

```

91
92 % Put legend and enhance appearance
93 % Legend bug with subscript, use '\_' instead of '_'
94 [hL,hObj]=legend({'$\omega_{eq} = -8.0 \ rad/s$',...
95 '$\omega_{eq} = -2.5 \ rad/s$', '$\omega_{eq} = -0.5 \ rad/s$',...
96 '$\omega_{eq} = 0.5 \ rad/s$', '$\omega_{eq} = 2.5 \ rad/s$',...
97 '$\omega_{eq} = 8.0 \ rad/s$'}, 'Interpreter', 'latex');
98 hTL=findobj(hObj, 'type', 'Text'); %
99 set(hTL, 'FontSize', 10); % font size for letters in legend
100 hTL=findobj(hObj, 'type', 'line'); %
101 set(hTL, 'LineWidth', 2); % thickness of lines in legend
102 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.28 0.24]);
103 % distance between lines in legend [x,y,width, height]
104
105 %% %% Singular and Bode Plots for different values of d
106 % the behaviour in the bode plots can be associated with the dominant
107 % pole variation wrt to d
108 d = 0.1; Weq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
109 L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\
110 %has to be chosen based on the corresponding AR value (AR_calculation.m)
111 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\
112 %has to be chosen based on the corresponding AR value
113 I = I_Newcalculation(d, Iw, L, md, dw);
114 [max,min] = I_maxmin(d, Iw, L, md, dw);
115
116 d = [-0.08 -0.07 -0.06 0.01 0.04 0.08];
117 I = [I_Newcalculation(d(1), Iw, L, md, dw) I_Newcalculation(d(2), Iw, L, md, dw)
118 I_Newcalculation(d(3), Iw, L, md, dw) I_Newcalculation(d(4), Iw, L, md, dw)
119 I_Newcalculation(d(5), Iw, L, md, dw)
120 I_Newcalculation(d(6), Iw, L, md, dw)];
121
122 P1 = Plantsdv(d(1), Veq, Weq, dw, Iw, I(1), L, md, R);
123 P2 = Plantsdv(d(2), Veq, Weq, dw, Iw, I(2), L, md, R);
124 P3 = Plantsdv(d(3), Veq, Weq, dw, Iw, I(3), L, md, R);
125 P4 = Plantsdv(d(4), Veq, Weq, dw, Iw, I(4), L, md, R);
126 P5 = Plantsdv(d(5), Veq, Weq, dw, Iw, I(5), L, md, R);
127 P6 = Plantsdv(d(6), Veq, Weq, dw, Iw, I(6), L, md, R);
128
129 figure;
130 bodemag(P1,P2,P3,P4,P5,P6);
131 grid on;
132 h_axes = findobj(gcf, 'type', 'axes');
133 xlabel('Frequency', 'FontSize', 12);
134 ylabel('Magnitude', 'FontSize', 12);
135 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
136 % size and brightness of grid and size of x & y axis numbers
137 title(...
138 'Frequency Response $(e_r + e_l, e_r - e_l)\rightarrow(v, \omega)$ for
139 $\neq 0$', 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex')
140
141 h_line = findobj(gcf, 'type', 'line');
142 set(h_line, 'LineWidth', 1.5); % Lines with thicker width for plots
143
144 % Put legend and enhance appearance
145 % Legend bug with subscript, use '\_' instead of '_'
146 [hL,hObj]=legend({'$d = -0.08 \ m$', '$d = -0.06 \ m$',...
147 '$d = -0.04 \ m$', '$d = 0.01 \ m$', '$d = 0.04 \ m$',...

```

```

148 '$d = 0.08 \ m$}', 'Interpreter', 'latex');
149 hTL=findobj(hObj, 'type', 'Text'); %
150 set(hTL, 'FontSize', 10); % font size for letters in legend
151 hTL=findobj(hObj, 'type', 'line'); %
152 set(hTL, 'LineWidth', 2); % thickness of lines in legend
153 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.26 0.24]);
154 % distance between lines in legend [x,y,width, height]
155
156 %% plot of dominant pole vs d
157 figure;
158 load('dpole_3-1.mat');
159 plot(h(1,:), h(2,:));
160 grid on;
161 h_axes = findobj(gcf, 'type', 'axes');
162 xlabel('d (m)', 'FontSize', 12);
163 ylabel('Dominant Pole ', 'FontSize', 12);
164 set(h_axes, 'LineWidth', 2, 'FontSize', 12, 'GridAlpha', 0.15);
165 % size and brightness of grid and size of x & y axis numbers
166 title('Dominant Pole vs $d$', 'FontWeight', 'bold', 'FontSize', 14, ...
167 'Interpreter', 'latex')
168
169 h_line = findobj(gcf, 'type', 'line');
170 set(h_line, 'LineWidth', 1.8); % Lines with thicker width for
171 % plots
172
173
174 %% %% Singular and Bode Plots for different values of I
175 % the behaviour in the bode plots can be associated with the dominant
176 % pole variation wrt to d
177 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
178 L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\\
179 %has to be chosen based on the corresponding AR value (AR_calculation.m)
180 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\\
181 %has to be chosen based on the corresponding AR value
182 I = I_Newcalculation(d, Iw, L, md, dw);
183 [max, min] = Imaxmin(d, Iw, L, md, dw);
184
185 I = [0.4 0.5 0.7 0.9 1.2 1.7];
186
187 P1 = Plantsdv(d, Veq, Weq, dw, Iw, I(1), L, md, R);
188 P2 = Plantsdv(d, Veq, Weq, dw, Iw, I(2), L, md, R);
189 P3 = Plantsdv(d, Veq, Weq, dw, Iw, I(3), L, md, R);
190 P4 = Plantsdv(d, Veq, Weq, dw, Iw, I(4), L, md, R);
191 P5 = Plantsdv(d, Veq, Weq, dw, Iw, I(5), L, md, R);
192 P6 = Plantsdv(d, Veq, Weq, dw, Iw, I(6), L, md, R);
193
194 figure;
195 bodemag(P1, P2, P3, P4, P5, P6);
196 grid on;
197 h_axes = findobj(gcf, 'type', 'axes');
198 xlabel('Frequency', 'FontSize', 12);
199 ylabel('Magnitude', 'FontSize', 12);
200 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
201 % size and brightness of grid and size of x & y axis numbers
202 title(...
203 'Frequency Response $(e_r + e_l, e_r - e_l) \rightarrow(v, \omega)$ for
204 '$d \neq 0$', 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex')

```

```

205
206 h_line = findobj(gcf, 'type', 'line');
207 set(h_line, 'LineWidth',1.5);           % Lines with thicker width for plots
208
209 % Put legend and enhance appearance
210 % Legend bug with subscript, use '\_' instead of '_'
211 [hL,hObj]=legend({'$I = 0.4 \ Kg.m^2$', '$I = 0.5 \ Kg.m^2$', ...
212 '$I = 0.7 \ Kg.m^2$', '$I = 0.9 \ Kg.m^2$', '$I = 1.2 \ Kg.m^2$', ...
213 '$I = 1.7 \ Kg.m^2$'}, 'Interpreter', 'latex');
214 hTL=findobj(hObj, 'type', 'Text');      %
215 set(hTL, 'FontSize', 10);              % font size for letters in legend
216 hTL=findobj(hObj, 'type', 'line');      %
217 set(hTL, 'LineWidth', 2);              % thickness of lines in legend
218 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.27 0.24]);
219 % distance between lines in legend [x,y,width, height]

1  % Trade Studies at  $d \sim 0$  Contd.
2
3  clc
4  close all
5  clear all
6  s = tf([1 0],[1]);
7  md = 0; m = 3.4;
8  winit = -1;
9  wfin = 2;
10 nwpts = 200;
11 w = logspace(winit,wfin,nwpts);
12 %% Different Plant Models with the respective parameters as input
13 % at  $d = 0$ 
14 % Plant model from  $e_r$ ,  $e_l$  to  $W_r$ ,  $W_l$  decoupled
15 % Plant model from  $(e_r + e_l)$ ,  $(e_r - e_l)$  to  $V$ ,  $W$  decoupled
16 %%
17 % Plant model from  $(e_r + e_l)$ ,  $(e_r - e_l)$  to  $V$ ,  $W$  decoupled
18 % Singular and Bode Plots for different values of  $Ve_q$ 
19 d = 0.1;  $Ve_q = 2$ ;  $We_q = 0.8$ ; % in m/s max value is 0.14 for hardware
20 L = 0.3536;  $dw = L/\sqrt{2}$ ;  $R = 0.042$ ; % default values  $L = 0.3536$  //\\
21 %has to be chosen based on the corresponding AR value (AR_calculation.m)
22  $I_w = 1.67e-06$ ;  $A = m + 2*I_w/(R*R)$ ; % default values //\\
23 %has to be chosen based on the corresponding AR value
24 I = I_Newcalculation(d,  $I_w$ , L, md, dw);
25 [max,min] = Imaxmin(d,  $I_w$ , L, md, dw);
26 Plant1 = Plantww(d,  $Ve_q$ ,  $We_q$ , dw,  $I_w$ , I, L, md, R);
27
28  $Ve_q = [0.1 0.2 0.6 1 3 5]$ ;
29
30 P1 = Plantsdv(d,  $Ve_q(1)$ ,  $We_q$ , dw,  $I_w$ , I, L, md, R);
31 P2 = Plantsdv(d,  $Ve_q(2)$ ,  $We_q$ , dw,  $I_w$ , I, L, md, R);
32 P3 = Plantsdv(d,  $Ve_q(3)$ ,  $We_q$ , dw,  $I_w$ , I, L, md, R);
33 P4 = Plantsdv(d,  $Ve_q(4)$ ,  $We_q$ , dw,  $I_w$ , I, L, md, R);
34 P5 = Plantsdv(d,  $Ve_q(5)$ ,  $We_q$ , dw,  $I_w$ , I, L, md, R);
35 P6 = Plantsdv(d,  $Ve_q(6)$ ,  $We_q$ , dw,  $I_w$ , I, L, md, R);
36
37 P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
38 P4 = sigma(P4,w); P5 = sigma(P5,w);
39 P6 = sigma(P6,w);
40 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);

```

```

41 P4 = 20*log10(P4); P5 = 20*log10(P5);
42 P6 = 20*log10(P6);
43
44 figure;
45 subplot(2,1,1);
46 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:), ...
47 w, P6(1,:))
48 %clear sve_r, e_l to W_r, W_l
49 grid on;
50 h_axes = findobj(gcf, 'type', 'axes');
51 xlabel('Frequency','FontSize',12);
52 ylabel('Magnitude','FontSize',12);
53 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
54 % size and brightness of grid and size of x & y axis numbers
55 title(...
56 'Max Singular Values   $(e_r + e_l, e_r - e_l) \rightarrow (\nu, \omega)$  for
57  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
58
59 h_line = findobj(gcf, 'type', 'line');
60 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
61
62 subplot(2,1,2);
63 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:), ...
64 w, P6(2,:))
65 %clear sv
66 grid on;
67 h_axes = findobj(gcf, 'type', 'axes');
68 xlabel('Frequency','FontSize',12);
69 ylabel('Magnitude','FontSize',12);
70 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
71 % size and brightness of grid and size of x & y axis numbers
72 title(...
73 'Min Singular Values   $(e_r + e_l, e_r - e_l) \rightarrow (\nu, \omega)$  for
74  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize',12, 'Interpreter', 'latex')
75
76 h_line = findobj(gcf, 'type', 'line');
77 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
78
79 % Put legend and enhance appearance
80 % Legend bug with subscript, use '\_' instead of '_'
81 [hL,hObj]=legend({'$v_{eq} = 0.1 \ m/s$', '$v_{eq} = 0.2 \ m/s$', ...
82 '$v_{eq} = 0.6 \ m/s$', '$v_{eq} = 1.0 \ m/s$', '$v_{eq} = 3.0 \ m/s$', ...
83 '$v_{eq} = 5.0 \ m/s$'}, 'Interpreter', 'latex');
84 hTL=findobj(hObj, 'type', 'Text'); %
85 set(hTL, 'FontSize',11); % font size for letters in legend
86 hTL=findobj(hObj, 'type', 'line'); %
87 set(hTL, 'LineWidth',2); % thickness of lines in legend
88 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.26 0.24]);
89 % distance between lines in legend [x,y,width, height]
90
91 %% Singular and Bode Plots for different values of Weq
92 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
93 L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\
94 %has to be chosen based on the corresponding AR value (AR_calculation.m)
95 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\
96 %has to be chosen based on the corresponding AR value
97 I = I.Newcalculation(d, Iw, L, md, dw);

```

```

98 [max,min] = Imaxmin(d,Iw,L,md,dw);
99
100 Weq = [-8.0 -2.5 -0.5 0.5 2.5 8.0];
101
102 P1 = Plantsdv(d, Veq, Weq(1), dw, Iw, I, L, md,R);
103 P2 = Plantsdv(d, Veq, Weq(2), dw, Iw, I, L, md,R);
104 P3 = Plantsdv(d, Veq, Weq(3), dw, Iw, I, L, md,R);
105 P4 = Plantsdv(d, Veq, Weq(4), dw, Iw, I, L, md,R);
106 P5 = Plantsdv(d, Veq, Weq(5), dw, Iw, I, L, md,R);
107 P6 = Plantsdv(d, Veq, Weq(6), dw, Iw, I, L, md,R);
108
109 P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
110 P4 = sigma(P4,w); P5 = sigma(P5,w);
111 P6 = sigma(P6,w);
112 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
113 P4 = 20*log10(P4); P5 = 20*log10(P5);
114 P6 = 20*log10(P6);
115
116 figure;
117 subplot(2,1,1);
118 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:), ...
119 w, P6(1,:))
120 %clear sv
121 grid on;
122 h_axes = findobj(gcf, 'type', 'axes');
123 xlabel('Frequency','FontSize',12);
124 ylabel('Magnitude','FontSize',12);
125 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
126 % size and brightness of grid and size of x & y axis numbers
127 title(...
128 'Max Singular Values  $(e_r + e_l, e_r - e_l) \rightarrow (\omega)$  for
129  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
130
131 h_line = findobj(gcf, 'type', 'line');
132 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
133
134 subplot(2,1,2);
135 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:), ...
136 w, P6(2,:))
137 %clear sv
138 grid on;
139 h_axes = findobj(gcf, 'type', 'axes');
140 xlabel('Frequency','FontSize',12);
141 ylabel('Magnitude','FontSize',12);
142 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
143 % size and brightness of grid and size of x & y axis numbers
144 title(...
145 'Min Singular Values  $(e_r + e_l, e_r - e_l) \rightarrow (\omega)$  for
146  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize',12, 'Interpreter', 'latex')
147
148 h_line = findobj(gcf, 'type', 'line');
149 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
150
151 % Put legend and enhance appearance
152 % Legend bug with subscript, use '\_' instead of '_'
153 [hL,hObj]=legend({' $\omega_{eq} = -8.0 \text{ rad/s}$ ',...
154 ' $\omega_{eq} = -2.5 \text{ rad/s}$ ', ' $\omega_{eq} = -0.5 \text{ rad/s}$ ',...

```



```

155 '$\omega_{eq} = 0.5 \ rad/s$', '$\omega_{eq} = 2.5 \ rad/s$', ...
156 '$\omega_{eq} = 8.0 \ rad/s$'}', 'Interpreter', 'latex');
157 hTL=findobj(hObj, 'type', 'Text'); %
158 set(hTL, 'FontSize', 10); % font size for letters in legend
159 hTL=findobj(hObj, 'type', 'line'); %
160 set(hTL, 'LineWidth', 2); % thickness of lines in legend
161 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.28 0.24]);
162 % distance between lines in legend [x,y,width, height]
163
164 %% %% Singular and Bode Plots for different values of d
165 % the behaviour in the bode plots can be associated with the dominant
166 % pole variation wrt to d
167 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
168 L = 0.3536; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 //\
169 %has to be chosen based on the corresponding AR value (AR_calculation.m)
170 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values //\
171 %has to be chosen based on the corresponding AR value
172 I = I.Newcalculation(d, Iw, L, md, dw);
173 [max, min] = I.maxmin(d, Iw, L, md, dw);
174
175 d = [-0.08 -0.07 -0.06 0.01 0.04 0.08];
176 I = [I.Newcalculation(d(1), Iw, L, md, dw) I.Newcalculation(d(2), Iw, L, md, dw)
177      I.Newcalculation(d(3), Iw, L, md, dw) I.Newcalculation(d(4), Iw, L, md, dw)
178      I.Newcalculation(d(5), Iw, L, md, dw)
179      I.Newcalculation(d(6), Iw, L, md, dw)];
180
181 P1 = Plantsdv(d(1), Veq, Weq, dw, Iw, I(1), L, md, R);
182 P2 = Plantsdv(d(2), Veq, Weq, dw, Iw, I(2), L, md, R);
183 P3 = Plantsdv(d(3), Veq, Weq, dw, Iw, I(3), L, md, R);
184 P4 = Plantsdv(d(4), Veq, Weq, dw, Iw, I(4), L, md, R);
185 P5 = Plantsdv(d(5), Veq, Weq, dw, Iw, I(5), L, md, R);
186 P6 = Plantsdv(d(6), Veq, Weq, dw, Iw, I(6), L, md, R);
187
188 P1 = sigma(P1, w); P2 = sigma(P2, w); P3 = sigma(P3, w);
189 P4 = sigma(P4, w); P5 = sigma(P5, w);
190 P6 = sigma(P6, w);
191 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
192 P4 = 20*log10(P4); P5 = 20*log10(P5);
193 P6 = 20*log10(P6);
194
195 figure;
196 subplot(2, 1, 1);
197 semilogx(w, P1(1, :), w, P2(1, :), w, P3(1, :), w, P4(1, :), w, P5(1, :), ...
198 w, P6(1, :))
199 %clear sv
200 grid on;
201 h_axes = findobj(gcf, 'type', 'axes');
202 xlabel('Frequency', 'FontSize', 12);
203 ylabel('Magnitude', 'FontSize', 12);
204 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
205 % size and brightness of grid and size of x & y axis numbers
206 title(...
207 'Max Singular Values $(e_r + e_l, e_r - e_l)\rightarrow(v, \omega)$ for
208 $\neq 0$', 'FontWeight', 'bold', 'FontSize', 14, 'Interpreter', 'latex')
209
210 h_line = findobj(gcf, 'type', 'line');
211 set(h_line, 'LineWidth', 1.5); % Lines with thicker width for plots

```

```

212
213 subplot(2,1,2);
214 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
215 w, P6(2,:))
216 %clear sv
217 grid on;
218 h_axes = findobj(gcf, 'type', 'axes');
219 xlabel('Frequency','FontSize',12);
220 ylabel('Magnititude','FontSize',12);
221 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
222 % size and brightness of grid and size of x & y axis numbers
223 title(...
224 'Min Singular Values  $(e_r + e_l, e_r - e_l) \rightarrow (\nu, \omega)$  for
225  $d \neq 0$ ','FontWeight','bold','FontSize',12, 'Interpreter','latex')
226
227 h_line = findobj(gcf, 'type', 'line');
228 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
229
230 % Put legend and enhance appearance
231 % Legend bug with subscript, use '\_' instead of '_'
232 [hL,hObj]=legend({'$d = -0.1 \ m$', '$d = -0.05 \ m$',...
233 '$d = -0.02 \ m$', '$d = 0.02 \ m$', '$d = 0.05 \ m$',...
234 '$d = 0.1 \ m$'}, 'Interpreter','latex');
235 hTL=findobj(hObj, 'type', 'Text'); %
236 set(hTL, 'FontSize',10); % font size for letters in legend
237 hTL=findobj(hObj, 'type', 'line'); %
238 set(hTL, 'LineWidth',2); % thickness of lines in legend
239 set(hL, 'FontSize',1, 'Position',[0.5 0.5 0.26 0.24]);
240 % distance between lines in legend [x,y,width, height]
241
242 %% %% Singular and Bode Plots for different values of I
243 % the behaviour in the bode plots can be associated with the dominat
244 % pole variation wrt to d
245 d = 0.1; Veq = 2; Weq = 0.8; % in m/s max value is 0.14 for hardware
246 L = 1; dw = L/sqrt(2); R = 0.042; % default values L = 0.3536 /\
247 %has to be chosen based on the corresponding AR value (AR_calculation.m)
248 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values /\
249 %has to be chosen based on the corresponding AR value
250 I = I.Newcalculation(d, Iw, L, md, dw);
251 [max,min] = I.maxmin(d, Iw, L, md, dw);
252
253 I = [0.4 0.5 0.7 0.9 1.2 1.7];
254
255 P1 = Plantsdv(d, Veq, Weq, dw, Iw, I(1), L, md, R);
256 P2 = Plantsdv(d, Veq, Weq, dw, Iw, I(2), L, md, R);
257 P3 = Plantsdv(d, Veq, Weq, dw, Iw, I(3), L, md, R);
258 P4 = Plantsdv(d, Veq, Weq, dw, Iw, I(4), L, md, R);
259 P5 = Plantsdv(d, Veq, Weq, dw, Iw, I(5), L, md, R);
260 P6 = Plantsdv(d, Veq, Weq, dw, Iw, I(6), L, md, R);
261
262 P1 = sigma(P1,w); P2 = sigma(P2,w); P3 = sigma(P3,w);
263 P4 = sigma(P4,w); P5 = sigma(P5,w);
264 P6 = sigma(P6,w);
265 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
266 P4 = 20*log10(P4); P5 = 20*log10(P5);
267 P6 = 20*log10(P6);
268

```

```

269 figure;
270 subplot(2,1,1);
271 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:), w, P5(1,:),...
272 w, P6(1,:))
273 %clear sv
274 grid on;
275 h_axes = findobj(gcf, 'type', 'axes');
276 xlabel('Frequency','FontSize',12);
277 ylabel('Magnititude','FontSize',12);
278 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
279 % size and brightness of grid and size of x & y axis numbers
280 title(...
281 'Max Singular Values  $(e_r + e_l, e_r - e_l) \rightarrow (v, \omega)$  for
282  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize',14, 'Interpreter', 'latex')
283
284 h_line = findobj(gcf, 'type', 'line');
285 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
286
287 subplot(2,1,2);
288 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:), w, P5(2,:),...
289 w, P6(2,:))
290 %clear sv
291 grid on;
292 h_axes = findobj(gcf, 'type', 'axes');
293 xlabel('Frequency','FontSize',12);
294 ylabel('Magnititude','FontSize',12);
295 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
296 % size and brightness of grid and size of x & y axis numbers
297 title(...
298 'Min Singular Values  $(e_r + e_l, e_r - e_l) \rightarrow (v, \omega)$  for
299  $d \neq 0$ ', 'FontWeight', 'bold', 'FontSize',12, 'Interpreter', 'latex')
300
301 h_line = findobj(gcf, 'type', 'line');
302 set(h_line, 'LineWidth',1.5); % Lines with thicker width for plots
303
304 % Put legend and enhance appearance
305 % Legend bug with subscript, use '\_' instead of '_'
306 [hL,hObj]=legend({' $I = 0.4 \text{ Kg.m}^2$ ', ' $I = 0.5 \text{ Kg.m}^2$ ',...
307 ' $I = 0.7 \text{ Kg.m}^2$ ', ' $I = 0.9 \text{ Kg.m}^2$ ', ' $I = 1.2 \text{ Kg.m}^2$ ',...
308 ' $I = 1.7 \text{ Kg.m}^2$ '}, 'Interpreter', 'latex');
309 hTL=findobj(hObj, 'type', 'Text'); %
310 set(hTL, 'FontSize',10); % font size for letters in legend
311 hTL=findobj(hObj, 'type', 'line'); %
312 set(hTL, 'LineWidth',2); % thickness of lines in legend
313 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.27 0.24]);
314 % distance between lines in legend [x,y,width, height]

1 % Inner-Loop frequency response plots
2 clc
3 close all
4 clear all
5 s = tf([1 0],[1]);
6 md = 0; m = 3.4; % if d = 0;
7 %% Different Plant Models with the respective parameters as input
8 % at d = 0
9 % Plant model from e_r, e_l to W_r, W_l decoupled

```

```

10 d1 = 0; d2 = 0.08; Veq = 2; Weq = 0.8; % independent of Veq and Weq
11 L = 1; dw = 1/sqrt(2); R = 0.042; % default values L = 0.3536 /\
12 %has to be chosen based on the corresponding AR value (AR_calculation.m)
13 Iw = 1.67e-06; A = m + 2*Iw/(R*R); % default values /\
14 %has to be chosen based on the corresponding AR value
15 I1 = 0.4250; % I_AR
16 I2 = 1.7;% 4.0I_AR
17 %[max,min] = Imaxmin(d2,Iw,L,md,dw)
18 PlantD1M1 = Plantww(d1, Veq, Weq, dw, Iw, I1, L, md,R); I1 = 0.42560;
19 PlantD1M2 = Plantsdv(d1, Veq, Weq, dw, Iw, I1, L, md,R); I1 = 0.4250;
20
21 PlantD2M1 = Plantww(d1, Veq, Weq, dw, Iw, I2, L, md,R);
22 PlantD2M2 = Plantsdv(d1, Veq, Weq, dw, Iw, I2, L, md,R);
23
24 PlantD3M1 = Plantww(d2, Veq, Weq, dw, Iw, I1, L, md,R);
25 PlantD3M2 = Plantsdv(d2, Veq, Weq, dw, Iw, I1, L, md,R);
26
27
28 PlantD4M1 = Plantww(d2, Veq, Weq, dw, Iw, I2, L, md,R);
29 PlantD4M2 = Plantsdv(d2, Veq, Weq, dw, Iw, I2, L, md,R);
30
31 %bodemag(PlantD1M2, PlantD2M2)
32 PD1M1 = minreal(zpk(tf(PlantD1M1)));
33 PD1M2 = minreal(zpk(tf(PlantD1M2)));
34
35 PD2M1 = minreal(zpk(tf(PlantD2M1)));
36 PD2M2 = minreal(zpk(tf(PlantD2M2)));
37
38 PD3M1 = minreal(zpk(tf(PlantD3M1)));
39 PD3M2 = minreal(zpk(tf(PlantD3M2)));
40
41 PD4M1 = minreal(zpk(tf(PlantD4M1)));
42 PD4M2 = minreal(zpk(tf(PlantD4M2)));
43
44 KD1M1 = [0.6435 + 4.8633/s 0; 0 0.6435 + 4.8633/s]*(100/(s+100));
45 KD1M2 = [30.6428 + 231.5868/s 0; 0 10.8341 + 81.8799/s]*(100/(s+100));
46
47 KD2M1 = [1.0713 + 6.9057/s 0; 0 1.0713 + 6.9057/s]*(100/(s+100));
48 KD2M2 = [30.6428 + 231.5868/s 0; 0 46.5661 + 144.8851/s]*(100/(s+100));
49
50 KD3M1 = [0.63861 + 5.2248/s 0; 0 0.63861 + 5.2248/s]*(100/(s+100));
51 KD3M2 = [30.6428 + 231.5868/s 0; 0 10.6159 + 94.2519/s]*(100/(s+100));
52
53 KD4M1 = [1.0666 + 6.9621/s 0; 0 1.0666 + 6.9621/s]*(100/(s+100));
54 KD4M2 = [30.6428 + 231.5868/s 0; 0 46.3477 + 157.2715/s]*(100/(s+100));
55
56
57 WD1M1 = [ (4.8633/0.6435)/(s+ 4.8633/0.6435) 0;...
58 0 (4.8633/0.6435)/(s+(4.8633/0.6435))]
59 WD1M2 = [ (231.5868/30.6428)/(s + 231.5868/30.6428) 0;...
60 0 (81.8799/10.8341)/(s + 81.8799/10.8341)]
61
62 WD2M1 = [ (6.9057/1.0713)/(s + 6.9057/1.0713) 0;...
63 0 (6.9057/1.0713)/(s + 6.9057/1.0713)]
64 WD2M2 = [ (231.5868/30.6428)/(s+231.5868/30.6428) 0;...
65 0 (144.8851/46.5661)/(s+144.8851/46.5661)]
66

```

```

67 WD3M1 = [ (5.2248/0.63861)/(s+5.2248/0.63861) 0;...
68 0 (5.2248/0.63861)/(s+5.2248/0.63861) ]
69 WD3M2 = [ (231.5868/30.6428)/(s+231.5868/30.6428) 0;...
70 0 (94.2519/10.6159)/(s+94.2519/10.6159) ]
71
72 WD4M1 = [ (6.9621/1.0666)/(s+6.9621/1.0666) 0;...
73 0 (6.9621/1.0666)/(s+6.9621/1.0666) ]
74 WD4M2 = [ (231.5868/30.6428)/(s+231.5868/30.6428) 0;...
75 0 (157.2715/46.3477)/(s+157.2715/46.3477) ]
76
77
78 LD1M1 = PD1M1*KD1M1;
79 LD1M2 = PD1M2*KD1M2;
80
81 LD2M1 = PD2M1*KD2M1;
82 LD2M2 = PD2M2*KD2M2;
83
84 LD3M1 = PD3M1*KD3M1;
85 LD3M2 = PD3M2*KD3M2;
86
87 LD4M1 = PD4M1*KD4M1;
88 LD4M2 = PD4M2*KD4M2;
89 %%
90 SD1M1 = (eye(2) + LD1M1)^-1;
91 SD1M2 = (eye(2) + LD1M2)^-1;
92
93 SD2M1 = (eye(2) + LD2M1)^-1;
94 SD2M2 = (eye(2) + LD2M2)^-1;
95
96 SD3M1 = (eye(2) + LD3M1)^-1;
97 SD3M2 = (eye(2) + LD3M2)^-1;
98
99 SD4M1 = (eye(2) + LD4M1)^-1;
100 SD4M2 = (eye(2) + LD4M2)^-1;
101
102 %% complimentary sensitivity
103 CD1M1 = LD1M1*(eye(2) + LD1M1)^-1;
104 CD1M2 = LD1M2*(eye(2) + LD1M2)^-1;
105
106 CD2M1 = LD2M1*(eye(2) + LD2M1)^-1;
107 CD2M2 = LD2M2*(eye(2) + LD2M2)^-1;
108
109 CD3M1 = LD3M1*(eye(2) + LD3M1)^-1;
110 CD3M2 = LD3M2*(eye(2) + LD3M2)^-1;
111
112 CD4M1 = LD4M1*(eye(2) + LD4M1)^-1;
113 CD4M2 = LD4M2*(eye(2) + LD4M2)^-1;
114
115 %% Try
116 inM = ([1/R dw/(2*R); 1/R -dw/(2*R)]);
117 M = [R/2 R/2; R/dw -R/dw];
118
119 TD1M1 = M*(LD1M1*(eye(2) + LD1M1)^-1)*WD1M1*inM;
120 TD1M2 = CD1M2*WD1M2;
121
122 TD2M1 = M*(LD2M1*(eye(2) + LD2M1)^-1)*WD2M1*inM;
123 TD2M2 = CD2M2*WD2M2;

```

```

124
125 TD3M1 = M*(LD3M1*(eye(2) + LD3M1)^-1)*WD3M1*inM;
126 TD3M2 = CD3M2*WD3M2;
127
128 TD4M1 = M*(LD4M1*(eye(2) + LD4M1)^-1)*WD4M1*inM;
129 TD4M2 = CD4M2*WD4M2;
130
131 %% Tru
132 TRD1M1 = (KD1M1*(eye(2) + LD1M1)^-1)*WD1M1*inM;
133 TRD1M2 = KD1M2*SD1M2*WD1M2;
134
135 TRD2M1 = (KD2M1*(eye(2) + LD2M1)^-1)*WD2M1*inM;
136 TRD2M2 = KD2M2*SD2M2*WD2M2;
137
138 TRD3M1 = (KD3M1*(eye(2) + LD3M1)^-1)*WD3M1*inM;
139 TRD3M2 = KD3M2*SD3M2*WD3M2;
140
141 TRD4M1 = (KD4M1*(eye(2) + LD4M1)^-1)*WD4M1*inM;
142 TRD4M2 = KD4M2*SD4M2*WD4M2;
143
144
145 %% Open Loop
146 winit = -1;
147 wfin = 4;
148 nwpts = 200;
149 w = logspace(winit,wfin,nwpts);
150 P1 = sigma(LD1M1,w); P2 = sigma(LD2M1,w); P3 = sigma(LD1M2,w);
151 P4 = sigma(LD2M2,w);
152 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
153 P4 = 20*log10(P4);
154 figure;
155 subplot(2,1,1);
156 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
157 %clear sv
158 grid on;
159 h_axes = findobj(gcf, 'type', 'axes');
160 xlabel('Frequency','FontSize',12);
161 ylabel('Magnitude','FontSize',12);
162 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
163 % size and brightness of grid and size of x & y axis numbers
164 title('Max Singular Values: Open Loop','FontWeight','bold',...
165 'FontSize',14, 'Interpreter','latex')
166
167 h_line = findobj(gcf, 'type', 'line');
168 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
169
170 subplot(2,1,2);
171 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
172 %clear sv
173 grid on;
174 h_axes = findobj(gcf, 'type', 'axes');
175 xlabel('Frequency','FontSize',12);
176 ylabel('Magnitude','FontSize',12);
177 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
178 % size and brightness of grid and size of x & y axis numbers
179 title('Min Singular Values: Open Loop','FontWeight','bold',...
180 'FontSize',12, 'Interpreter','latex')

```

```

181
182 h_line = findobj(gcf, 'type', 'line');
183 set(h_line, 'LineWidth',1.2);           % Lines with thicker width for plots
184
185
186 % Put legend and enhance appearance
187 % Legend bug with subscript, use '\_' instead of '_'
188 [hL,hObj]=legend({'D1M1', 'D2M1', 'D1M2', 'D2M2'}, 'Interpreter','latex');
189 hTL=findobj(hObj, 'type', 'Text');      %
190 set(hTL, 'FontSize',11);                % font size for letters in legend
191 hTL=findobj(hObj, 'type', 'line');      %
192 set(hTL, 'LineWidth',1.2);              % thickness of lines in legend
193 set(hL, 'FontSize',1, 'Position',[0.5 0.5 0.2 0.2]);
194 % distance between lines in legend [x,y,width, height]
195
196 %%
197 winit = -1;
198 wfin = 4;
199 nwpts = 200;
200 w = logspace(winit,wfin,nwpts);
201 P1 = sigma(LD3M1,w); P2 = sigma(LD4M1,w); P3 = sigma(LD3M2,w);
202 P4 = sigma(LD4M2,w);
203 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
204 P4 = 20*log10(P4);
205 figure;
206 subplot(2,1,1);
207 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
208 %clear sv
209 grid on;
210 h_axes = findobj(gcf, 'type', 'axes');
211 xlabel('Frequency','FontSize',12);
212 ylabel('Magnitude','FontSize',12);
213 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
214 % size and brightness of grid and size of x & y axis numbers
215 title('Max Singular Values: Open Loop','FontWeight','bold',...
216 'FontSize',14, 'Interpreter','latex')
217
218 h_line = findobj(gcf, 'type', 'line');
219 set(h_line, 'LineWidth',1.2);           % Lines with thicker width for plots
220
221 subplot(2,1,2);
222 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
223 %clear sv
224 grid on;
225 h_axes = findobj(gcf, 'type', 'axes');
226 xlabel('Frequency','FontSize',12);
227 ylabel('Magnitude','FontSize',12);
228 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
229 % size and brightness of grid and size of x & y axis numbers
230 title('Min Singular Values: Open Loop','FontWeight','bold',...
231 'FontSize',12, 'Interpreter','latex')
232
233 h_line = findobj(gcf, 'type', 'line');
234 set(h_line, 'LineWidth',1.2);           % Lines with thicker width for plots
235
236
237 % Put legend and enhance appearance

```

```

238 % Legend bug with subscript, use '\_' instead of '_'
239 [hL,hObj]=legend({'D3M1','D4M1','D3M2','D4M2'},'Interpreter','latex');
240 hTL=findobj(hObj,'type','Text'); %
241 set(hTL,'FontSize',11); % font size for letters in legend
242 hTL=findobj(hObj,'type','line'); %
243 set(hTL,'LineWidth',1.2); % thickness of lines in legend
244 set(hL,'FontSize',1,'Position',[0.5 0.5 0.2 0.2]);
245 % distance between lines in legend [x,y,width,height]
246 %% Sensitivity
247
248 winit = -1;
249 wfin = 4;
250 nwpts = 200;
251 w = logspace(winit,wfin,nwpts);
252 P1 = sigma(SD1M1,w); P2 = sigma(SD2M1,w); P3 = sigma(SD1M2,w);
253 P4 = sigma(SD2M2,w);
254 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
255 P4 = 20*log10(P4);
256 figure;
257 subplot(2,1,1);
258 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
259 %clear sv
260 grid on;
261 h_axes = findobj(gcf, 'type', 'axes');
262 xlabel('Frequency','FontSize',12);
263 ylabel('Magnitude','FontSize',12);
264 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
265 % size and brightness of grid and size of x & y axis numbers
266 title('Max Singular Values: Sensitivity','FontWeight','bold',...
267 'FontSize',14,'Interpreter','latex')
268
269 h_line = findobj(gcf, 'type', 'line');
270 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
271
272 subplot(2,1,2);
273 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
274 %clear sv
275 grid on;
276 h_axes = findobj(gcf, 'type', 'axes');
277 xlabel('Frequency','FontSize',12);
278 ylabel('Magnitude','FontSize',12);
279 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
280 % size and brightness of grid and size of x & y axis numbers
281 title('Min Singular Values: Sensitivity','FontWeight','bold',...
282 'FontSize',12,'Interpreter','latex')
283
284 h_line = findobj(gcf, 'type', 'line');
285 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
286
287
288 % Put legend and enhance appearance
289 % Legend bug with subscript, use '\_' instead of '_'
290 [hL,hObj]=legend({'D1M1','D2M1','D1M2','D2M2'},'Interpreter','latex');
291 hTL=findobj(hObj,'type','Text'); %
292 set(hTL,'FontSize',11); % font size for letters in legend
293 hTL=findobj(hObj,'type','line'); %
294 set(hTL,'LineWidth',1.2); % thickness of lines in legend

```



```

295 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.2 0.2]);
296 % distance between lines in legend [x,y,width, height]
297
298 %%
299
300 winit = -1;
301 wfin = 4;
302 nwpts = 200;
303 w = logspace(winit, wfin, nwpts);
304 P1 = sigma(SD3M1, w); P2 = sigma(SD4M1, w); P3 = sigma(SD3M2, w);
305 P4 = sigma(SD4M2, w);
306 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
307 P4 = 20*log10(P4);
308 figure;
309 subplot(2,1,1);
310 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
311 %clear sv
312 grid on;
313 h_axes = findobj(gcf, 'type', 'axes');
314 xlabel('Frequency', 'FontSize', 12);
315 ylabel('Magnitude', 'FontSize', 12);
316 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
317 % size and brightness of grid and size of x & y axis numbers
318 title('Max Singular Values: Sensitivity', 'FontWeight', 'bold', ...
319 'FontSize', 14, 'Interpreter', 'latex')
320
321 h_line = findobj(gcf, 'type', 'line');
322 set(h_line, 'LineWidth', 1.2); % Lines with thicker width for plots
323
324 subplot(2,1,2);
325 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
326 %clear sv
327 grid on;
328 h_axes = findobj(gcf, 'type', 'axes');
329 xlabel('Frequency', 'FontSize', 12);
330 ylabel('Magnitude', 'FontSize', 12);
331 set(h_axes, 'LineWidth', 1.5, 'FontSize', 10, 'GridAlpha', 0.18);
332 % size and brightness of grid and size of x & y axis numbers
333 title('Min Singular Values: Sensitivity', 'FontWeight', 'bold', ...
334 'FontSize', 12, 'Interpreter', 'latex')
335
336 h_line = findobj(gcf, 'type', 'line');
337 set(h_line, 'LineWidth', 1.2); % Lines with thicker width for plots
338
339
340 % Put legend and enhance appearance
341 % Legend bug with subscript, use '\_' instead of '_'
342 [hL, hObj]=legend({'D3M1', 'D4M1', 'D3M2', 'D4M2'}, 'Interpreter', 'latex');
343 hTL=findobj(hObj, 'type', 'Text'); %
344 set(hTL, 'FontSize', 11); % font size for letters in legend
345 hTL=findobj(hObj, 'type', 'line'); %
346 set(hTL, 'LineWidth', 1.2); % thickness of lines in legend
347 set(hL, 'FontSize', 1, 'Position', [0.5 0.5 0.2 0.2]);
348 % distance between lines in legend [x,y,width, height]
349
350 %% Complimentary Sensitivity
351

```

```

352 winit    = -1;
353 wfin     = 4;
354 nwpts    = 200;
355 w        = logspace(winit,wfin,nwpts);
356 P1 = sigma(CD1M1,w); P2 = sigma(CD2M1,w); P3 = sigma(CD1M2,w);
357 P4 = sigma(CD2M2,w);
358 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
359 P4 = 20*log10(P4);
360 figure;
361 subplot(2,1,1);
362 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
363 %clear sv
364 grid on;
365 h_axes = findobj(gcf, 'type', 'axes');
366 xlabel('Frequency','FontSize',12);
367 ylabel('Magnitude','FontSize',12);
368 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
369 % size and brightness of grid and size of x & y axis numbers
370 title('Max Singular Values: Complementary Sensitivity','FontWeight',...
371 'bold','FontSize',14, 'Interpreter','latex')
372
373 h_line = findobj(gcf, 'type', 'line');
374 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
375
376 subplot(2,1,2);
377 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
378 %clear sv
379 grid on;
380 h_axes = findobj(gcf, 'type', 'axes');
381 xlabel('Frequency','FontSize',12);
382 ylabel('Magnitude','FontSize',12);
383 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
384 % size and brightness of grid and size of x & y axis numbers
385 title('Min Singular Values: Complementary Sensitivity','FontWeight',...
386 'bold','FontSize',12, 'Interpreter','latex')
387
388 h_line = findobj(gcf, 'type', 'line');
389 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
390
391
392 % Put legend and enhance appearance
393 % Legend bug with subscript, use '\_' instead of '_'
394 [hL,hObj]=legend({'D1M1', 'D2M1', 'D1M2', 'D2M2'}, 'Interpreter','latex');
395 hTL=findobj(hObj, 'type', 'Text'); %
396 set(hTL, 'FontSize',11); % font size for letters in legend
397 hTL=findobj(hObj, 'type', 'line'); %
398 set(hTL, 'LineWidth',1.2); % thickness of lines in legend
399 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.2 0.2]);
400 % distance between lines in legend [x,y,width, height]
401
402 %%
403
404 winit    = -1;
405 wfin     = 4;
406 nwpts    = 200;
407 w        = logspace(winit,wfin,nwpts);
408 P1 = sigma(CD3M1,w); P2 = sigma(CD4M1,w); P3 = sigma(CD3M2,w);

```

```

409 P4 = sigma(CD4M2,w);
410 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
411 P4 = 20*log10(P4);
412 figure;
413 subplot(2,1,1);
414 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
415 %clear sv
416 grid on;
417 h_axes = findobj(gcf, 'type', 'axes');
418 xlabel('Frequency','FontSize',12);
419 ylabel('Magnitude','FontSize',12);
420 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
421 % size and brightness of grid and size of x & y axis numbers
422 title('Max Singular Values: Complementary Sensitivity','FontWeight',...
423 'bold','FontSize',14, 'Interpreter','latex')
424
425 h_line = findobj(gcf, 'type', 'line');
426 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
427
428 subplot(2,1,2);
429 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
430 %clear sv
431 grid on;
432 h_axes = findobj(gcf, 'type', 'axes');
433 xlabel('Frequency','FontSize',12);
434 ylabel('Magnitude','FontSize',12);
435 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
436 % size and brightness of grid and size of x & y axis numbers
437 title('Min Singular Values: Complementary Sensitivity','FontWeight',...
438 'bold','FontSize',12, 'Interpreter','latex')
439
440 h_line = findobj(gcf, 'type', 'line');
441 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
442
443
444 % Put legend and enhance appearance
445 % Legend bug with subscript, use '\_' instead of '_'
446 [hL,hObj]=legend({'D3M1','D4M1','D3M2','D4M2'}, 'Interpreter','latex');
447 hTL=findobj(hObj, 'type', 'Text'); %
448 set(hTL, 'FontSize',11); % font size for letters in legend
449 hTL=findobj(hObj, 'type', 'line'); %
450 set(hTL, 'LineWidth',1.2); % thickness of lines in legend
451 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.2 0.2]);
452 % distance between lines in legend [x,y,width, height]
453
454 %% Try
455 figure;
456 bodemag(TD1M1, TD2M1, TD1M2, TD2M2, w);
457 grid on;
458 h_axes = findobj(gcf, 'type', 'axes');
459 xlabel('Frequency','FontSize',12);
460 ylabel('Magnitude','FontSize',12);
461 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
462 % size and brightness of grid and size of x & y axis numbers
463 title('Frequency Response','FontWeight','bold','FontSize',14, ...
464 'Interpreter','latex')
465

```

```

466 h_line = findobj(gcf, 'type', 'line');
467 set(h_line, 'LineWidth',1.5);           % Lines with thicker width for plots
468
469 % Put legend and enhance appearance
470 % Legend bug with subscript, use '\_' instead of '_'
471 [hL,hObj]=legend({'D1M1','D2M1','D1M2','D2M2'},'Interpreter','latex');
472 hTL=findobj(hObj,'type','Text');       %
473 set(hTL,'FontSize',11);                 % font size for letters in legend
474 hTL=findobj(hObj,'type','line');       %
475 set(hTL,'LineWidth',2);                % thickness of lines in legend
476 set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.2]);
477 % distance between lines in legend [x,y,width, height]
478
479
480 figure;
481 bodemag(TD3M1,TD4M1,TD3M2,TD4M2,w);
482 grid on;
483 h_axes = findobj(gcf, 'type', 'axes');
484 xlabel('Frequency','FontSize',12);
485 ylabel('Magnitude','FontSize',12);
486 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18);
487 % size and brightness of grid and size of x & y axis numbers
488 title('Frequency Response','FontWeight','bold','FontSize',14, ...
489 'Interpreter','latex')
490
491 h_line = findobj(gcf, 'type', 'line');
492 set(h_line, 'LineWidth',1.5);           % Lines with thicker width for plots
493
494 % Put legend and enhance appearance
495 % Legend bug with subscript, use '\_' instead of '_'
496 [hL,hObj]=legend({'D3M1','D4M1','D3M2','D4M2'},'Interpreter','latex');
497 hTL=findobj(hObj,'type','Text');       %
498 set(hTL,'FontSize',11);                 % font size for letters in legend
499 hTL=findobj(hObj,'type','line');       %
500 set(hTL,'LineWidth',2);                % thickness of lines in legend
501 set(hL,'FontSize',1,'Position',[0.5 0.5 0.25 0.2]);
502 % distance between lines in legend [x,y,width, height]
503
504 %% Tru
505
506 winit = -1;
507 wfin = 4;
508 nwpts = 200;
509 w = logspace(winit,wfin,nwpts);
510 P1 = sigma(TRD1M1,w); P2 = sigma(TRD2M1,w); P3 = sigma(TRD1M2,w);
511 P4 = sigma(TRD2M2,w);
512 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
513 P4 = 20*log10(P4);
514 figure;
515 subplot(2,1,1);
516 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
517 %clear sv
518 grid on;
519 h_axes = findobj(gcf, 'type', 'axes');
520 xlabel('Frequency','FontSize',12);
521 ylabel('Magnitude','FontSize',12);
522 set(h_axes,'LineWidth',1.5,'FontSize',10,'GridAlpha',0.18); ...

```

```

523 % size and brightness of grid and size of x & y axis numbers
524 title('Max Singular Values: Tru','FontWeight','bold','FontSize',14,...
525 'Interpreter','latex')
526
527 h_line = findobj(gcf, 'type', 'line');
528 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
529
530 subplot(2,1,2);
531 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
532 %clear sv
533 grid on;
534 h_axes = findobj(gcf, 'type', 'axes');
535 xlabel('Frequency','FontSize',12);
536 ylabel('Magnitude','FontSize',12);
537 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
538 % size and brightness of grid and size of x & y axis numbers
539 title('Min Singular Values: Tru','FontWeight','bold','FontSize',12,...
540 'Interpreter','latex')
541
542 h_line = findobj(gcf, 'type', 'line');
543 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
544
545
546 % Put legend and enhance appearance
547 % Legend bug with subscript, use '\_' instead of '_'
548 [hL,hObj]=legend({'D1M1', 'D2M1', 'D1M2', 'D2M2'}, 'Interpreter','latex');
549 hTL=findobj(hObj, 'type', 'Text'); %
550 set(hTL, 'FontSize',11); % font size for letters in legend
551 hTL=findobj(hObj, 'type', 'line'); %
552 set(hTL, 'LineWidth',1.2); % thickness of lines in legend
553 set(hL, 'FontSize',1, 'Position', [0.5 0.5 0.2 0.2]);
554 % distance between lines in legend [x,y,width, height]
555 winit = -1;
556
557 %%
558 wfin = 4;
559 nwpts = 200;
560 w = logspace(winit,wfin,nwpts);
561 P1 = sigma(TRD3M1,w); P2 = sigma(TRD4M1,w); P3 = sigma(TRD3M2,w);
562 P4 = sigma(TRD4M2,w);
563 P1 = 20*log10(P1); P2 = 20*log10(P2); P3 = 20*log10(P3);
564 P4 = 20*log10(P4);
565 figure;
566 subplot(2,1,1);
567 semilogx(w, P1(1,:), w, P2(1,:), w, P3(1,:), w, P4(1,:))
568 %clear sv
569 grid on;
570 h_axes = findobj(gcf, 'type', 'axes');
571 xlabel('Frequency','FontSize',12);
572 ylabel('Magnitude','FontSize',12);
573 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
574 % size and brightness of grid and size of x & y axis numbers
575 title('Max Singular Values: Tru','FontWeight','bold','FontSize',14,...
576 'Interpreter','latex')
577
578 h_line = findobj(gcf, 'type', 'line');
579 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots

```

```

580
581 subplot(2,1,2);
582 semilogx(w, P1(2,:), w, P2(2,:), w, P3(2,:), w, P4(2,:))
583 %clear sv
584 grid on;
585 h_axes = findobj(gcf, 'type', 'axes');
586 xlabel('Frequency','FontSize',12);
587 ylabel('Magnititude','FontSize',12);
588 set(h_axes, 'LineWidth',1.5, 'FontSize',10, 'GridAlpha',0.18);
589 % size and brightness of grid and size of x & y axis numbers
590 title('Min Singular Values: Tru','FontWeight','bold','FontSize',12,...
591 'Interpreter','latex')
592
593 h_line = findobj(gcf, 'type', 'line');
594 set(h_line, 'LineWidth',1.2); % Lines with thicker width for plots
595
596
597 % Put legend and enhance appearance
598 % Legend bug with subscript, use '\_' instead of '_'
599 [hL,hObj]=legend({'D3M1','D4M1','D3M2','D4M2'}, 'Interpreter','latex');
600 hTL=findobj(hObj, 'type', 'Text'); %
601 set(hTL, 'FontSize',11); % font size for letters in legend
602 hTL=findobj(hObj, 'type', 'line'); %
603 set(hTL, 'LineWidth',1.2); % thickness of lines in legend
604 set(hL, 'FontSize',1, 'Position',[0.5 0.5 0.2 0.2]);
605 % distance between lines in legend [x,y,width, height]

1 % dominant pole vs d for increasing v_eq
2 clc
3 close all
4 clear all
5
6 loop = 3; % 1 - PID design %2 Wr,Wl design %3 ICC Paper Plant
7 s = tf([1 0],[1]);
8 md = 0; % if d = 0;
9 %% Different Plant Models with the respective parameters as input
10
11 % Plant model from e_r, e_l to W_r, W_l with coupling
12 % Plant model from e_r, e_l to W_r, W_l decoupled
13 % Plant model from e_r, e_l to V, W with coupling
14 % Plant model from (e_r + e_l), (e_r-e_l) to V, W decoupled
15 %%
16 % Plant model from e_r, e_l to W_r, W_l (with coupling)
17 Veq = 1.0; % in m/s
18 Weq = 0.0; % in m/s
19 d = 0.28; % in m/s max value is 0.14 for hardware
20 L = 0.3536; dw = L*sqrt(2); R = 0.042; % default values L = 0.3536//\
21 %has to be chosen based on the corresponding AR value (AR_calculation.m)
22 Iw = 1.67e-06;% default values /\
23 %has to be chosen based on the corresponding AR value
24 I = I_cal(d,Iw,L,md,dw);
25 [ma,mi] = Imaxmin(d,Iw,L,md,dw);
26 i = 0; kk = 1; h = ones(7,57)
27 for j = 0: 0.5:3
28     Veq = j;
29 for d = -0.28:0.01:0.28

```

```

30     i= i+1;
31     I = I_cal(d,Iw,L,md,dw);
32     Plant1 = Plantvw(d, Veq, Weq, dw, Iw, I, L, md,R);
33     dpole(i) = max(real(pole(Plant1)));
34 end
35 h(kk, :)=dpole;
36 kk = kk +1;
37 i = 0;
38 end
39 d = -0.28:0.01:0.28;
40 plot(d,h(1, :),d,h(2, :),d,h(3, :),d,h(4, :),d,h(5, :),d,h(6, :),d,h(7, :))

1 % e_r, e_l to V,W Plant
2 function [Plant] = Plantvw(d, Veq, Weq, dw, Iw, I, L, md, R)
3
4 % Plant model from e_r, e_l to V, W
5
6 s = tf([1 0],[1]);
7 z = tf('z');
8
9 %% Plant Model
10 m = 3.4; mc = 2.76 + md; I = I ; Iw = Iw; L = L; d = d; dw = dw;
11 Weq = Weq; Veq = Veq;
12 Ao = m + 2*Iw/(R^2); Bo = I + (2*Iw*dw^2)/R^2;
13 % Linear Plant without actuator - Motor Torque to V,W
14
15 % Linear Plant without actuator - Motor Torque to Wr, Wl
16 Tmain2 = [(1/(s*R*R))*((1/Ao)+(L*L/Bo)) ...
17 (1/(s*R*R))*((1/Ao)-(L*L/Bo)); ...
18 (1/(s*R*R))*((1/Ao)-(L*L/Bo)) (1/(s*R*R))*((1/Ao)+(L*L/Bo))];
19
20 % Right Motor Actuator Dynamics - voltage to motor torque
21 Kt = 0.0337; Kg = 50; Kb = Kt; B = 1.3023e-04; La = 22.8e-06; Ra = 2.9;
22
23 % Left Motor Actuator Dynamics - voltage to motor torque
24 Kt = 0.0046; Kg = 50; Kb = Kt; Beta = 2.29e-06; La =1.729e-03;Ra = 5.51;
25
26 A = m + 2*Iw/(R*R); B = I + dw*dw*Iw/(2*R*R);
27
28 % ULTIMATE - CORRECT representation of Plant using State Space
29 %form derived on Sept 21st, 2019
30 As =[-2*Beta*Kg*Kg/(A*R*R) 2*mc*d*Weq/A (Kt*Kg/(R*A)) (Kt*Kg/(A*R));
31 -mc*d*Weq/B (-mc*d*Veq/B)-(Beta*dw*dw*Kg*Kg/(2*R*R*B)) ...
32 dw*Kt*Kg/(2*R*B) -dw*Kt*Kg/(2*B*R) ;
33 -Kb*Kg/(La*R) -Kb*Kg*dw/(2*La*R) -Ra/La 0 ;
34 -Kb*Kg/(La*R) Kb*Kg*dw/(2*La*R) 0 -Ra/La ];
35 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];
36 Mains = ss(As,Bs,Cs,Ds);
37 MainTfs = minreal(tf(Mains)); Mains = ss(MainTfs);
38 MainTfs = minreal(zpk(MainTfs));
39
40 % Alternate representation of Plant from Lin's thesis ea -r,l to Wr,Wl
41 % H1 = Kt/(La*m*R*R*s*s + (Ra*m*R*R + 2*La*B)*s + (2*Kb*Kt + 2*Ra*B));
42 % H2 = dw*dw*Kt/(I*La*R*R*s*s + (I*Ra*R*R + dw*dw*La*B)*s +
43 % (Kb*Kt*L*L + dw*dw*Ra*B));
44 % Plant3 = [H1+0.5*H2 H1-0.5*H2; H1-0.5*H2 H1+0.5*H2];

```

```

45 %Plant3 = minreal(zpk(Plant3));
46 Plant = Mains;
47
48 end

1 % e_r, e_l to W_r, W_l Plant
2 function [Plant] = Plantww(d, Veq, Weq, dw, Iw, I, L, md,R)
3
4 % Plant model from e_r, e_l to W_r, W_l
5
6 R = R; dw = dw;
7
8 Temp = Plantvw(d,Veq,Weq, dw, Iw, I,L,md,R);
9 %MainTfs = ([1/R dw/(2*R); 1/R -dw/(2*R)])*Temp; % This code is for
10 %obtaining the transfer function represntation of the plant
11 %MainTfs = minreal(tf(MainTfs)); MainTfs = minreal(zpk(MainTfs));
12
13 Temp.C = ([1/R dw/(2*R); 1/R -dw/(2*R)])*Temp.C;
14 Plant = Temp;
15
16 end

1 % (e_r + e_l), (e_r-e_l) to V, W Plant
2 function [Plant] = Plantsdv(d, Veq, Weq, dw, Iw, I, L, md,R)
3
4 % Plant model from (e_r + e_l), (e_r-e_l) to V, W
5
6 dw = dw;
7
8 Temp = Plantvw(d,Veq,Weq,dw,Iw,I,L,md,R);
9 %MainTfs = Temp*[0.5 0.5; 0.5 -0.5]; % this code is for transfer
10 %function representation of the plant
11 %MainTfs = minreal(tf(MainTfs)); MainTfs = minreal(zpk(MainTfs));
12
13 Temp.B = Temp.B*[0.5 0.5; 0.5 -0.5];
14 Plant = Temp;
15
16 end

1 % I as a function of d; d is varied by shifting m_c to new location (db)
2 function [I] = I_cal(d,Iw,L,md,dw)
3 % I as a function of d, dw
4 % range of d is -0.03 to 0.03
5 m = 3.4;
6 mc = 2.76; % mass without motors
7 mw = (m - mc)*0.5 ; % mass of individual motor and wheel
8 %dw = 0.25*167;
9 % the Li-ion battery, camera and the Lipo battery are shifted to match
10 % the new d value, so no new mass is being added md = 0 always
11 %% Iw estimation (max and min value estimation)
12
13 %rw = 0.1; m_wheel = 0.181;
14 %rm = 0.0248 ; m_motor = 0.096;

```



```

15 %maxval = 0.5*m_motor*rm*rm + 0.5*m_wheel*rw*rw;
16 % minval = maxval/8;
17
18 %% Ic calculation
19
20 m_plate = 0.411;
21
22
23 db = d*m/(mc);
24
25
26 Ic = mc*(db - d)*(db - d);
27
28
29 %% I approximation
30
31 I_approx = (1/12)*m*(L*L + dw*dw); % in order to verify
32 %if the calculated I is right or wrong
33
34 %% I original
35
36 % Iw = wheel+motor moment of inertia about wheel axel
37 I = Ic + (mc+md)*d*d + 2*mw*dw*dw + Iw;
38
39
40 end

1 %I as a function of d; d is varied by shifting camera and LiPo battery
2 %location
3 function [I] = I.Newcalculation(d,Iw,L,md,dw)
4 % I as a function of d, dw
5 % range of d is -0.03 to 0.03
6 m = 3.4;
7 mc = 2.76; % mass without motors
8 mw = (m - mc)*0.5 ; % mass of individual motor and wheel
9 %dw = 0.25*167;
10 % the Li-ion battery, camera and the Lipo battery are shifted to match
11 % the new d value, so no new mass is being added md = 0 always
12 %% Iw estimation (max and min value estimation)
13
14 %rw = 0.042; m_wheel = 0.181;
15 %rm = 0.0248 ; m_motor = 0.096;
16 %maxval = 0.5*m_motor*rm*rm + 0.5*m_wheel*rw*rw;
17 %minval = maxval/8;
18
19 %% Ic calculation
20 %
21 % plate
22 Ic = md*(L/2)^2; % initial value, md is the
23 %additional mass that has to be added to manipulate d,
24 m_plate = 0.411;
25
26
27 Ic = Ic + (2/12)*m_plate*(L*L + dw*dw) + 2*m_plate*d*d;
28 % for the two acrylic sheets
29 % 3d print

```

```

30 % m3d = 0.055;
31 % l3d = 0.079;
32 % Ic = Ic + 2*m3d*l3d*l3d;
33 % Nvidia
34 m_nvidia = 0.4;
35 l_nvidia = 0.12*dw;
36 Ic = Ic + m_nvidia*((l_nvidia^2) + (d^2));
37 m_bat = 0.492;
38 m_lipo = 0.185;
39 m_cam = 0.077;
40 db = d*mc/(m_bat + m_cam + m_lipo);
41
42 % battery
43 m_bat = 0.492;
44 l_cam = 0.404*L;
45 l = 0.108;
46 w = 0.101;5
47 Ic = Ic + (1/12)*m_bat*(l*l + w*w) + m_bat*(db - d)*(db - d);
48 % camera
49 m_cam = 0.077;
50 l_cam = 0.404*L;
51 Ic = Ic + m_cam*(db - d)*(db - d);
52 % LiPo
53 m_lipo = 0.185;
54 l_lipo = 0.404*L;
55 l = 0.10;
56 w = 0.034;
57 Ic = Ic + m_lipo*(db - d)*(db - d) + (1/12)*m_lipo*(l*l + w*w);
58 % arduino + motor shield
59 m_ard_shield = 0.062;
60 L_ard_shield = 0.033;
61 Ic = Ic + m_ard_shield*(L_ard_shield)^2; % doesn't really matter
62 %% I approximation
63
64 I_approx = (1/12)*m*(L*L + dw*dw); % in order to verify if
65 %the calculated I is right or wrong
66 %% I original
67
68 % Iw = wheel+motor moment of inertia about wheel axel
69 I = Ic + (mc+md)*d*d + 2*mw*dw*dw + Iw;
70
71 end

```

APPENDIX C
ROS AND ARDUINO CODE

```

1
2 // Description: Arduino code for generic PI inner-loop
3 #include <Servo.h>
4 #include <math.h>
5 #include <ros.h>
6 #include <ros/time.h>
7 #include <tf/tf.h>
8 #include <tf/transform_broadcaster.h>
9 #include <nav_msgs/Odometry.h>
10 #include <geometry_msgs/Vector3.h>
11 #include <geometry_msgs/Vector3Stamped.h>
12 #include <geometry_msgs/Twist.h>
13 #include <std_msgs/Int8.h>
14 #include <Adafruit_Sensor.h>
15 #include <Adafruit_BNO055.h>
16 #include <utility/imuMaths.h>
17
18 Servo left; // create servo object to control right motor
19 Servo right; // create servo object to control left motor
20
21 unsigned long Time=0; // Starting time
22 unsigned long lastMilli = 0;
23 double td = 0.0095; // T = 0.01 sec (100 hz)
24 unsigned long sample_time= td*1000*0.1 ;
25
26 double wd ; // Desired angular speed of COM about ICC(Instantaneous
27 //center of curvature)
28 double vd ; // Desired longitudinal speed of center of mass
29
30 double wR; // present angular speed of right motor
31 double wL; // present angular speed of left motor
32 double wRp=0.0; // previous angular speed right motor
33 double wLp=0.0; // previous angular speed left motor
34 double wLn; // average angular speed (wL + wLp)/2
35 double wRn; // average angular speed (wR + wRp)/2
36
37 double CPR = 1024.0; // encoder counts per revolution
38 double LdVal = 0.0;
39 double RdVal = 0.0;
40 long Lcount; // Present Encoder value
41 long Rcount; // Present Encoder value
42 long Lcount_last=0; // Previous encoder value
43 long Rcount_last=0; // Previous encoder value
44
45 double Radius = 0.04; // Change it (radius of wheel) 0.045
46 double Length = 0.32; // Change it (distance between wheels) 0.555 0.308
47
48 double wdr = 0; // Desired angular speed of right wheel using wd &
49 // vd prefilter parameter x_{n+1}
50 double wdl = 0; // Desired angular speed of left wheel using wd &
51 //vd prefilter parameter x_{n+1}
52 double wdr_p= 0.0; // prefilter parameter x_{n} for right motor
53 double wdl_p= 0.0; // prefilter parameter x_{n} for left motor
54 double wrf; // prefilter parameter y_{n+1} for right motor
55 double wlf; // prefilter parameter y_{n+1} for left motor
56 double wrf_p= 0.0; // prefilter paramter y_{n} for right motor
57 double wlf_p= 0.0; // prefilter parameter y_{n} for left motor'

```

```

58
59 double CR;          // Controller output y- $\{n+2\}$  Right motor
60 double CR_p=0.0;   // Controller output y- $\{n+1\}$  Right motor
61 double CR_pp=0.0;  // Controller output y- $\{n\}$  Right motor
62 double CR_ppp=0.0;
63 double CR_pppp=0.0;
64 double CL;          // Controller output y- $\{n+2\}$  Left motor
65 double CL_p=0.0;   // Controller output y- $\{n+1\}$  Left motor
66 double CL_pp=0.0;  // Controller output y- $\{n\}$  Left motor
67 double CL_ppp=0.0;
68 double CL_pppp=0.0;
69
70 double Lerror;     // Lerror = wlf(output of prefilter/ reference speed)
71 // - wLn.....or... Controller input x- $\{n+2\}$ 
72 double Lerror_p = 0.0; // Controller input x- $\{n+1\}$ 
73 double Lerror_pp = 0.0; // Controller input x- $\{n\}$ 
74 double Lerror_ppp = 0.0;
75 double Lk = 0.0;
76 double Rerror;    // Rerror = wrf(output of prefilter/ reference speed)
77 // - wRn....or.....Controller input x- $\{n+2\}$ 
78 double Rerror_p = 0.0; // Controller input x- $\{n+1\}$ 
79 double Rerror_pp = 0.0; // Controller input x- $\{n\}$ 
80 double Rerror_ppp = 0.0;
81 double Rk = 0.0;
82
83 double Lx = 0.0; // left - integrator anti-windup
84 double Rx = 0.0; // right - integrator anti-windup
85
86 double PWMR; // Controller output for right motor
87 double PWML; // Controller output for left motor
88 int val; // input to the motors
89
90 double A ;          // Controller gain kp of
91 //  $K = (kp + ki/s) * (100/(s+100))$ 
92 double B ;          // Controller gain ki
93 double C ;          // Controller gain kp of
94 //  $K = (kp + ki/s) * (100/(s+100))$ 
95 double Kp = 0.5; double Ki; double Kd; long long EN; long long DE;
96 long long DE1;
97 double ta = 1/1260;
98 double Po = 0.0; double scale;
99 double g = 1.0; double z; // Controller gain ki
100 double alpha = 200.0; // Roll off parameter alpha
101 double h ; // prefilter parameter  $z = ki/kp$ 
102 //obtained from  $K = (g(s+z)/s)*(100/(s+100))$ 
103 // for PD controller double b1; double b0; double c1; double c0;
104 // double A;
105
106
107 // Subscriber call back to /cmd_vel
108 void twist_message_cmd(const geometry_msgs::Twist& msg)
109 {
110     wdr = msg.linear.x ;
111     wdl = msg.angular.x ;
112     //wdl = wdr;
113     //if (wdr == 0) g = 0.0;
114     //else g = 1.0;

```

```

115     g = msg.linear.z;
116     //z = msg.angular.z;
117 }
118
119
120 // Node handle
121 ros::NodeHandle arduino_nh ;
122
123 //geometry_msgs::Twist msg ;
124 //geometry_msgs::Vector3Stamped rpm_msg;
125 geometry_msgs::Twist rpm_msg ;
126
127
128 // Publisher of the right and left wheel angular velocities
129 //ros::Publisher pub("robot_2/arduino_vel", &rpm_msg); // Add robot_*
130 ros::Publisher pub("arduino_vel", &rpm_msg);
131
132 // Subscriber of the reference velocities coming from the outerloop
133 ros::Subscriber<geometry_msgs::Twist> sub("cmd_vel",...
134 &twist_message_cmd);
135 //ros::Subscriber<std_msgs::Int8> sub2("emergency_stop", &callback );
136
137
138 // Left Encoder
139 #define LH_ENCODER_A PK0 // pin A8 (PCINT16)
140 #define LH_ENCODER_B PK1 // pin A9 (PCINT17)
141 static long left_ticks = 0L;
142 volatile bool LeftEncoderBSet ;
143
144 // Right Encoder
145 #define RH_ENCODER_A PB0 // Digital pin 53 (PCINT 0)
146 #define RH_ENCODER_B PB1 // Digital pin 52 (PCINT 1)
147 static long right_ticks = 0L;
148 volatile bool RightEncoderBSet ;
149
150 #define LEFT 0
151 #define RIGHT 1
152
153 static const int8_t ENC_STATES [] = {0,1,-1,0,-1,0,0,1,1,0,0,-1,0,-1,...
154 1,0};
155 //encoder lookup table
156
157 /* Interrupt routine for LEFT encoder, taking care of actual counting */
158 ISR (PCINT2_vect) // pin change interrupts for port K (A8,A9)
159 {
160     static uint8_t enc_last=0;
161     enc_last <<=2; //shift previous state two places
162     enc_last |= (PINK & (3 << 0)) >> 0 ;
163     left_ticks -= ENC_STATES[(enc_last & 0x0f)]; // changed from -ve to
164     // +ve after interchanging the M1A and M1B wires
165 }
166
167 /* Interrupt routine for RIGHT encoder, taking care of actual
168 // counting */
169 ISR (PCINT0_vect) // pin change interrupts for port J
170 // (Digital pin 14,15)
171 {

```

```

172  static uint8_t enc_last=0;
173  enc_last <<=2; //shift previous state two places
174  enc_last |= (PINB & (3 << 0)) >> 0 ;
175  right_ticks -= ENC_STATES[(enc_last & 0x0f)]; // changed from -ve to
176  // +ve after interchanging the M1A and M1B wires
177  }
178
179  void SetupEncoders()
180  {
181  // Initializing the encoder pins as input pins
182
183  // set as inputs DDRD(pins 0-7) , DDRC(A0-A5)
184  // (The Port D Data Direction Register – read/write)
185  DDRK &= ~(1<<LH_ENCODER_A); // PK0 pin A8
186  DDRK &= ~(1<<LH_ENCODER_B); // PK1 pin A9
187  DDRB &= ~(1<<RH_ENCODER_A); // Digital pin 53 (PB0)
188  DDRB &= ~(1<<RH_ENCODER_B); // Digital pin 52 (PB1)
189
190  /* Pin to interrupt map:
191  * D0-D7 = PCINT 16-23 = PCIR2 = PD = PCIE2 = pcmsk2
192  * D8-D13 = PCINT 0-5 = PCIR0 = PB = PCIE0 = pcmsk0
193  * A0-A5 (D14-D19) = PCINT 8-13 = PCIR1 = PC = PCIE1 = pcmsk1
194  */
195
196  /*
197  For Atmega 2560 pin change interrupt enable flags
198  PCIE2 : PCINT23-16
199  PCIE1 : PCINT15-8
200  PCIE0 : PCINT7-0
201  */
202
203  // tell pin change mask to listen to left encoder pins and right pins
204  PCMSK2 |= (1 << LH_ENCODER_A)|(1 << LH_ENCODER_B);
205  PCMSK0 |= (1 << RH_ENCODER_A)|(1 << RH_ENCODER_B);
206
207  // enable PCINT1 and PCINT2 interrupt in the general interrupt mask
208  // the Pin Change Interrupt Enable flags have to be set in the PCICR
209  // register. These are bits PCIE0, PCIE1 and PCIE2 for the groups of
210  // pins PCINT7..0, PCINT14..8 and PCINT23..16 respectively
211  PCICR |= (1 << PCIE0) | (1 << PCIE2);
212  //PCICR |= (1 << PCIE2) ;
213  }
214
215
216  void setup() {
217  // put your setup code here, to run once:
218  Serial.begin(115200);
219
220  // initialize the encoders
221  SetupEncoders();
222
223  // attach servo to pin 51,11
224  left.attach(51);
225  right.attach(43);
226
227  // Arduino node
228  arduino_nh.initNode() ;

```

```

229
230 //broadcaster.init(arduino_nh) ; //added
231
232 arduino_nh.getHardware()->setBaud(115200);
233 arduino_nh.advertise(pub); // setting up subscriptions
234 arduino_nh.subscribe(sub); // setting up publications
235
236 }
237
238 void loop() {
239
240     if (millis() - Time > sample_time)
241     {
242         Time = millis() ;
243
244         // Update Motors with corresponding speed and send speed values
245         //through serial port
246
247         UpdateMotors(vd, wd);
248         publish_data();
249         arduino_nh.spinOnce();
250
251     }
252
253 }
254
255 // UPDATE MOTORS
256 void UpdateMotors(double vd, double wd)
257 {
258
259     //Prefilter
260     wrf = ( (td*h)*wdr + (td*h)*wdr_p - (td*h - 2)*wrf_p )/(2 + td*h);
261     wlf = ( (td*h)*wdl + (td*h)*wdl_p - (td*h - 2)*wlf_p )/(2 + td*h);
262     wrf_p = wrf;
263     wlf_p = wlf;
264     wdr_p = wdr;
265     wdl_p = wdl;
266
267
268     // Encoder counts
269     Lcount = left_ticks ;
270     Rcount = right_ticks ;
271     LdVal = (double) -(Rcount - Rcount_last)/(td) ; // Counts per second
272     // simple interchange to match notation
273     RdVal = (double) (Lcount - Lcount_last)/(td) ; // Counts per second
274     // simple interchange to match notation
275     Lcount_last = Lcount;
276     Rcount_last = Rcount;
277
278     // Present angular velocities
279     wL = (LdVal/CPR)*(2*3.14159) ; // rads/sec
280     wR = (RdVal/CPR)*(2*3.14159) ; // rads/sec
281
282     wLn = (wL + wLp)/2.0; // avg with previous values to make it even
283     // smoother
284     wRn = (wR + wRp)/2.0;
285

```



```

286  wLp = wL; // saving present angular velocities to be used in the next
287  // loop
288  wRp = wR; // saving present angular velocities to be used in the next
289  // loop
290
291  Error = wdr - wRn ; // error (ref - present) pre-filterer - should be
292  // added to prevent overshoot, reduces the jerk
293  Lerror = wdl - wLn ; // error (ref - present)
294
295  // Inner loop controller PID
296
297  CL = CL_p + 0.259*Error - 0.2166*Error_p;
298
299  CR = CR_p + 0.259*Error - 0.2166*Error_p;
300
301
302  CL_pppp = CL_ppp;
303  CL_ppp = CL_pp;
304  CL_pp = CL_p;
305
306  CR_pppp = CR_ppp;
307  CR_ppp = CR_pp;
308  CR_pp = CR_p;
309
310  Lerror_ppp = Lerror_pp;
311  Lerror_pp = Lerror_p;
312  Lerror_p = Lerror;
313  Rerror_ppp = Rerror_pp;
314  Rerror_pp = Rerror_p;
315  Rerror_p = Rerror;
316
317  if (CL < 0) CL = 0;
318  if (CL > 100) CL = 100;
319  if (CR < 0) CR = 0;
320  if (CR > 100) CR = 100;
321
322  CL_p = CL;
323  CR_p = CR;
324
325  CL = CL + 1570;
326  CR = CR + 1570;
327  left.writeMicroseconds (CL*g);
328  right.writeMicroseconds (CR*g);
329
330
331 }
332
333 void publish_data(){
334
335  rpm_msg.linear.x = wRn;//right angularVelocity;
336  rpm_msg.linear.y = wLn;//right angularVelocity;
337  rpm_msg.linear.z = Time;
338  rpm_msg.angular.x = CL;
339  rpm_msg.angular.y = CR;
340  rpm_msg.angular.z = g;
341  pub.publish(&rpm_msg);
342  //Serial.println(Time);

```

```
343 }
344 }
```

```
1
2 // Description: ROS Node to send Pose data from ground station
3 // (HTC Vive) to NVIDIA TX2s
4 #include "ros/ros.h"
5 #include "std_msgs/String.h"
6 #include "std_msgs/Int8.h"
7 #include <tf/tf.h>
8 #include<geometry_msgs/Vector3.h>
9 #include<geometry_msgs/Vector3Stamped.h>
10 #include<geometry_msgs/Twist.h>
11 #include<geometry_msgs/Point.h>
12 #include<geometry_msgs/PoseWithCovarianceStamped.h>
13 #include<sensor_msgs/Joy.h>
14 #include <sstream>
15 #include <iostream>
16 #include <fstream>
17
18
19 class readData{
20     public:
21         readData();
22     private:
23         ros::NodeHandle n;
24         ros::Publisher pub;
25         ros::Subscriber sub;
26         ros::Publisher pub2;
27         ros::Subscriber sub2;
28         void callBack(const geometry_msgs::PoseWithCovarianceStamped::
29             ConstPtr& msg);
30         void callBack2(const geometry_msgs::PoseWithCovarianceStamped::
31             ConstPtr& msg);
32         geometry_msgs::Point tracker1;
33         geometry_msgs::Point tracker2;
34 };
35
36 readData::readData(){
37     sub = n.subscribe<geometry_msgs::PoseWithCovarianceStamped>
38     ("/vive/LHR_D254C151_pose", 1000, &readData::callBack,this);
39     sub2 = n.subscribe<geometry_msgs::PoseWithCovarianceStamped>
40     ("/vive/LHR_90C2F95A_pose", 1000, &readData::callBack2,this);
41     pub = n.advertise<geometry_msgs::Point>("tracker_1",1000);
42     pub2 = n.advertise<geometry_msgs::Point>("tracker_2",1000);
43 }
44
45 void readData::callBack(const geometry_msgs::
46 PoseWithCovarianceStamped::ConstPtr& msg){tf::Quaternion q1(
47     msg->pose.pose.orientation.x,
48     msg->pose.pose.orientation.y,
49     msg->pose.pose.orientation.z,
50     msg->pose.pose.orientation.w);
51     tf::Quaternion q2(0.707, 0.000, 0.000, 0.707);
52     tf::Matrix3x3 m(q2*q1);
53     double roll, pitch, yaw;
```

```

54     static double yaw_prev = 0, relYaw = 0;
55         m.getRPY(roll, pitch, yaw);
56         // converts the Quaternion to Euler angles
57     if ((std::abs(yaw - yaw_prev)) > 3.141) {
58         if (yaw > yaw_prev) relYaw = relYaw - (2*3.141 -
59             std::abs(yaw - yaw_prev));
60         else relYaw = relYaw + (2*3.141 - std::abs(yaw - yaw_prev));
61         //if (yaw < yaw_prev) relYaw = relYaw +
62             //(2*3.141 - std::abs(yaw - yaw_prev));
63     }
64     else {
65         relYaw = relYaw + (yaw - yaw_prev);
66     }
67
68     yaw_prev = yaw;
69     tracker1.x = -(msg->pose.pose.position.x);
70     tracker1.y = msg->pose.pose.position.z;
71     tracker1.z = relYaw;
72     pub.publish(tracker1);
73 }
74 void readData::callBack2(const geometry_msgs::
75 PoseWithCovarianceStamped::ConstPtr& msg){tf::Quaternion q1(
76     msg->pose.pose.orientation.x,
77     msg->pose.pose.orientation.y,
78     msg->pose.pose.orientation.z,
79     msg->pose.pose.orientation.w);
80     tf::Quaternion q2(0.707, 0.000, 0.000, 0.707);
81     tf::Matrix3x3 m(q2*q1);
82     double roll, pitch, yaw;
83     m.getRPY(roll, pitch, yaw);
84     tracker2.x = -(msg->pose.pose.position.x);
85     tracker2.y = msg->pose.pose.position.z;
86     tracker2.z = yaw;
87     pub2.publish(tracker2);
88 }
89     //return 0;
90
91
92 int main(int argc, char **argv)
93 {
94     ros::init(argc, argv, "vive_data_send");
95
96     //TeleopJoy teleop_turtle;
97     readData dude;
98
99     ros::spin();
100
101     return 0;
102 }

```

```

1
2 //Description: ROS Node for Cruise Control
3 //Fri 22 May 2020 12:17:46 AM MST
4 #include "ros/ros.h"
5 #include "ros/time.h"
6 #include "std_msgs/Int8.h"

```

```

7 #include "std_msgs/String.h"
8 #include "std_msgs/Float64MultiArray.h"
9 #include <cmath>
10 #include <tf/tf.h>
11 #include<geometry_msgs/Vector3.h>
12 #include<geometry_msgs/Vector3Stamped.h>
13 #include<geometry_msgs/Twist.h>
14 #include<geometry_msgs/Point.h>
15 #include<geometry_msgs/PoseWithCovarianceStamped.h>
16 #include<sensor_msgs/Joy.h>
17 #include <sstream>
18 #include <iostream>
19 #include <fstream>
20
21 double Radius = 0.039;
22 double Length = 0.324;
23
24 class readData{
25 public:
26     readData();
27 private:
28     ros::NodeHandle n;
29     ros::Publisher pub; // publish to cmd vel: wr, wl
30     ros::Publisher pub2; // publish the experiment simulation data: x,y,
31     // theta,v,w,Vref,theta_ref
32     ros::Subscriber sub; // subscribe to keyboard
33     ros::Subscriber sub2; // subscribe to tracker_1: x,y,theta
34     void callBack(const geometry_msgs::Twist::ConstPtr& msg);
35     // subscribe to the keyboard
36     void callBack2(const geometry_msgs::Point::ConstPtr& msg);
37     // subscribe to the tracker_1
38     geometry_msgs::Twist vel; std_msgs::Float64MultiArray expData;
39     double wr; double wl; // for now the values are going to be in
40     // micro seconds to test the rpm of the motor and log the data
41     int cruise = 0; int initial = 0;
42     double x_i; double y_i; double theta_i; double x_f; double y_f;
43     double theta_f;
44     double v_ref; double w_ref; double theta_ref; double theta_err;
45     double k_theta = 5.0;
46     std::string line; std::string sV_ref; std::string sTheta_ref;
47     std::ifstream ifile
48     {"~/smannel/catkin_ws/src/highBW/src/Cruise_05r.csv"};
49 };
50
51 readData::readData(){
52     sub2 = n.subscribe<geometry_msgs::Point>("/tracker_1", 1,
53     &readData::callBack2,this);
54     sub = n.subscribe<geometry_msgs::Twist>("/keyboard",10,
55     &readData::callBack,this);
56     pub = n.advertise<geometry_msgs::Twist>("cmd_vel",1);
57     pub2 = n.advertise<std_msgs::Float64MultiArray>("exp_data",10000);
58 }
59
60 void readData::callBack(const geometry_msgs::Twist::ConstPtr& msg){
61     if (msg->linear.x == 2) cruise = 1;
62     else {
63         cruise = 0;

```

```

64     initial = 0; // resets the initial to record the initial values
65     // of x,y,theta
66 }
67 }
68
69 void readData::callBack2(const geometry_msgs::Point::ConstPtr& msg){
70     if (cruise == 1){
71         if (initial == 0){
72             x_i = msg->x;
73             y_i = msg->y;
74             theta_i = msg->z;
75             initial = 1;
76         }
77         x_f = (msg->x - x_i)*cos(theta_i) + (msg->y - y_i)*sin(theta_i)
78         + 0.0;
79         // add the starting value of the robot instead of 500
80         y_f = -(msg->x - x_i)*sin(theta_i) + (msg->y - y_i)*cos(theta_i)
81         + 0.0;
82         theta_f = msg->z - theta_i;
83
84         if (std::getline(ifile, line)) { // read the current line
85             std::istringstream iss{line}; // construct a string stream
86             // from line
87             std::getline(iss, sTheta_ref, ',');
88             std::getline(iss, sV_ref, ',');
89
90             //ROS_INFO("%s\n", sV_ref.c_str());
91             v_ref = std::stod(sV_ref);
92             theta_ref = std::stod(sTheta_ref);
93         }
94
95         // outerloop code
96         theta_err = theta_ref - theta_f;
97         w_ref = k_theta * theta_err;
98
99         wr = (2*v_ref + Length*w_ref)/(2*Radius);
100         wl = (2*v_ref - Length*w_ref)/(2*Radius);
101
102         vel.linear.x = wr;
103         vel.angular.x = wl;
104         vel.linear.z = 1;
105         pub.publish(vel); // cmd_vel to the inner loop
106
107         expData.data = { x_f, y_f, theta_f, v_ref, theta_ref};
108         pub2.publish(expData);
109     }
110 }
111 else {
112     wr = 0.0;
113     wl = 0.0;
114     vel.linear.z = 0;
115     vel.linear.x = wr;
116     vel.angular.x = wl;
117     pub.publish(vel); // cmd_vel to the inner loop
118 }
119 }
120

```

```

121 }
122 }
123
124 int main(int argc, char **argv){
125     ros::init(argc, argv, "Cruise");
126
127     //TeleopJoy teleop_turtle;
128     readData dude;
129
130     ros::spin();
131
132     return 0;
133 }

1
2 //Description: ROS Node for writing data into a .csv file
3 #include "ros/ros.h"
4 #include "ros/time.h"
5 #include "std_msgs/String.h"
6 #include "std_msgs/Int8.h"
7 #include "std_msgs/Float64.h"
8 #include "std_msgs/Float64MultiArray.h"
9 #include <cmath>
10 #include <tf/tf.h>
11 #include<geometry_msgs/Vector3.h>
12 #include<geometry_msgs/Vector3Stamped.h>
13 #include<geometry_msgs/Twist.h>
14 #include<geometry_msgs/Point.h>
15 #include<geometry_msgs/PoseWithCovarianceStamped.h>
16 #include<sensor_msgs/Joy.h>
17 #include <sstream>
18 #include <iostream>
19 #include <fstream>
20
21 double Radius = 0.039; // Change it (radius of wheel) 0.045
22 double Length = 0.324; // Change it (distance between the wheels (dw)
23 int emergency = 0;
24
25 class emergencyStop{ // stop the robot if any axes of the
26 //joystick is moved
27     public:
28         emergencyStop();
29     private:
30         ros::NodeHandle n;
31         ros::Subscriber sub;
32         void callback(const sensor_msgs::Joy::ConstPtr& joy);
33
34 };
35 emergencyStop::emergencyStop(){
36     sub = n.subscribe<sensor_msgs::Joy>("joy", 10, &emergencyStop::
37     callback, this);
38 }
39
40 void emergencyStop::callback(const sensor_msgs::Joy::ConstPtr& joy){
41     if (joy->axes[1] + joy->axes[2] + joy->axes[3] != 0)
42         emergency = 1;

```

```

43     }
44
45 class readData{
46     public:
47     readData();
48     private:          // emergency stop feature programmed in the robot3
49     // module incase of high current
50     void callBack(const geometry_msgs::Twist::ConstPtr& msg);
51     void callBack2(const std_msgs::Float64MultiArray::ConstPtr& msg);
52     void dataWrite(const geometry_msgs::Twist::ConstPtr& msg);
53     geometry_msgs::Twist vel;
54     std::string filename =
55     "/home/smanne1/catkin_ws/src/highBW/matlab/robot1/arduino.csv";
56     std::string filename2 =
57     "/home/smanne1/catkin_ws/src/highBW/matlab/robot1/cruiseData.csv";
58     int i; double vdf; double wdf; double wdr; double wdl; double Rwdr;
59     double Rwdl;
60     ros::NodeHandle n;
61     ros::Subscriber sub;
62     ros::Subscriber sub2;
63
64 };
65
66     readData::readData(){
67     sub = n.subscribe<geometry_msgs::Twist>("arduino_vel", 10,
68     &readData::callBack, this);
69     sub2 = n.subscribe<std_msgs::Float64MultiArray>(
70     "exp_dataRecord", 10000, &readData::callBack2, this);
71     i = 0;
72     }
73
74 void readData::callBack(const geometry_msgs::Twist::ConstPtr& msg){
75     dataWrite(msg);
76 }
77
78
79 void readData::dataWrite(const geometry_msgs::Twist::ConstPtr& msg){
80     //vdf = msg->linear.y;
81     //wdf = msg->angular.y;
82     //wdr = (2*vdf + Length*wdf)/(2*Radius);    // actual angular
83     //velocities
84     //wdl = (2*vdf - Length*wdf)/(2*Radius);
85
86     //Rwdr = (2*(msg->linear.x) + Length*(msg->angular.x))/(2*Radius);
87     // reference angular velocities
88     //Rwdl = (2*(msg->linear.x) - Length*(msg->angular.x))/(2*Radius);
89
90     vdf = (msg->linear.x + msg->linear.y)*Radius/2;
91     wdf = (msg->linear.x - msg->linear.y)*Radius/Length;
92
93     std::ofstream myfile;
94     ROS_INFO("printing data");
95     myfile.open(filename.c_str(), std::ios::app);
96     myfile << " Right_Angular_Vel " << msg->linear.x <<
97     " Left_Angular_Vel " << msg->linear.y;
98     myfile << " Time " << msg->linear.z << " Ref_Right " <<
99     msg->angular.x;

```

```

100         myfile << " Ref_Left " << msg->angular.y << "\n";
101 //         myfile << " Position_x " << msg->linear.z << " Position_y "
102 //<< msg->angular.z << "\n";
103         myfile.close();
104         //return 0;
105     }
106
107 void readData::callBack2(const std_msgs::Float64MultiArray::
108     ConstPtr& msg){
109     std::ofstream myfile2;
110     ROS_INFO("printing data");
111     myfile2.open(filename2.c_str(), std::ios::app);
112     myfile2 << " Position_x " << msg->data[0] << " Position_y "
113     << msg->data[1];
114     myfile2 << " Theta " << msg->data[2] << " Linear.Velocity "
115     << msg->data[3];
116     myfile2 << " Angular.Velocity " << msg->data[4] << " Time "
117     << msg->data[5];
118     myfile2 << " X_ref " << msg->data[6] << " Y_ref " <<
119     msg->data[7] << "\n";
120 //         myfile << " Position_x " << msg->linear.z << " Position_y "
121 //<< msg->angular.z << "\n";
122     myfile2.close();
123 }
124
125
126 int main(int argc, char **argv)
127 {
128     ros::init(argc, argv, "ground_station_data_receive.Vive2");
129
130     //emergencyStop delta;
131     readData dude;
132
133     ros::spin();
134
135     return 0;
136 }

```

```

1
2 // Description: ROS Node for generic outer-loop
3 //Fri 22 May 2020 12:17:46 AM MST
4 #include "ros/ros.h"
5 #include "ros/time.h"
6 #include "std_msgs/Int8.h"
7 #include "std_msgs/String.h"
8 #include "std_msgs/Float64MultiArray.h"
9 #include <cmath>
10 #include <tf/tf.h>
11 #include<geometry_msgs/Vector3.h>
12 #include<geometry_msgs/Vector3Stamped.h>
13 #include<geometry_msgs/Twist.h>
14 #include<geometry_msgs/Point.h>
15 #include<geometry_msgs/PoseWithCovarianceStamped.h>
16 #include<sensor_msgs/Joy.h>
17 #include <sstream>
18 #include <iostream>

```



```

19 #include <fstream>
20
21 class readData{
22     public:
23     readData();
24     private:
25     ros::NodeHandle n;
26     ros::Publisher pub; // publish to cmd vel: wr, wl
27     ros::Publisher pub2; // publish the experiment simulation data: x,y,
28     // theta,v,w,Vref,theta_ref
29     ros::Subscriber sub; // subscribe to keyboard
30     ros::Subscriber sub2; // subscribe to tracker_1: x,y,theta
31     void callBack(const geometry_msgs::Twist::ConstPtr& msg);
32     // subscribe to the keyboard
33     void callBack2(const geometry_msgs::Point::ConstPtr& msg);
34     // subscribe to the tracker_1
35     geometry_msgs::Twist vel; std_msgs::Float64MultiArray expData;
36     double wr; double wl; // for now the values are going to be
37     //in micro seconds to test the rpm of the motor and log the data
38     int cruise = 0; int initial = 0;
39     double x_i; double y_i; double theta_i; double x_f; double y_f;
40     double theta_f;
41     double v_ref; double theta_ref; std::string line; std::
42     string sV_ref;
43     std::string sTheta_ref;
44     std::ifstream ifile {
45         "/home/smannel/catkin_ws/src/highBW/src/Cruise.csv"};
46 };
47
48 readData::readData(){
49     sub2 = n.subscribe<geometry_msgs::Point>("/tracker_1", 1,
50     &readData::callBack2,this);
51     sub = n.subscribe<geometry_msgs::Twist>("/keyboard",10,
52     &readData::callBack,this);
53     pub = n.advertise<geometry_msgs::Twist>("cmd_vel",1);
54     pub2 = n.advertise<std_msgs::Float64MultiArray>("exp_data",10000);
55 }
56
57 void readData::callBack(const geometry_msgs::Twist::ConstPtr& msg){
58     if (msg->linear.x == 2) cruise = 1;
59     else {
60         cruise = 0;
61         initial = 0; // resets the initial to record the initial values
62         // of x,y,theta
63     }
64     vel.linear.x = msg->linear.x*(51/2);
65     vel.angular.x = msg->angular.z;
66     if (vel.linear.x < 0) vel.linear.x = 0;
67     if (vel.angular.x < 0) vel.angular.x = 0;
68     pub.publish(vel);
69 }
70
71 void readData::callBack2(const geometry_msgs::Point::ConstPtr& msg){
72     if (cruise == 1){
73         if (initial == 0){
74             x_i = msg->x;
75             y_i = msg->y;

```

```

76         theta_i = msg->z;
77         initial = 1;
78     }
79     x_f = (msg->x - x_i)*cos(theta_i) + (msg->y - y_i)*sin(theta_i)
80     + 0.0;
81     // add the starting value of the robot instead of 500
82     y_f = -(msg->x - x_i)*sin(theta_i) + (msg->y - y_i)*cos(theta_i)
83     + 0.0;
84     theta_f = msg->z - theta_i;
85
86
87
88     expData.data = { x_f, y_f, theta_f, v.ref, theta.ref};
89     pub2.publish(expData);
90 }
91 }
92
93 int main(int argc, char **argv){
94     ros::init(argc, argv, "tesla1");
95
96     //TeleopJoy teleop_turtle;
97     readData dude;
98     ros::spin();
99
100     return 0;
101 }

```

```

1 //Description: ROS Node for Planar Cartesian Stabilization
2 //Fri 22 May 2020 12:17:46 AM MST
3 #include "ros/ros.h"
4 #include "ros/time.h"
5 #include "std_msgs/Int8.h"
6 #include "std_msgs/String.h"
7 #include "std_msgs/Float64MultiArray.h"
8 #include <cmath>
9 #include <tf/tf.h>
10 #include<geometry_msgs/Vector3.h>
11 #include<geometry_msgs/Vector3Stamped.h>
12 #include<geometry_msgs/Twist.h>
13 #include<geometry_msgs/Point.h>
14 #include<geometry_msgs/PoseWithCovarianceStamped.h>
15 #include<sensor_msgs/Joy.h>
16 #include <sstream>
17 #include <iostream>
18 #include <fstream>
19
20 double Radius = 0.039;
21 double Length = 0.324;
22
23 class readData{
24     public:
25     readData();
26     private:
27     ros::NodeHandle n;
28     ros::Publisher pub; // publish to cmd vel: wr, wl
29     ros::Publisher pub2; // publish the experiment simulation data: x,y,

```

```

30 // theta,v,w,Vref,theta_ref
31   ros::Subscriber sub; // subscribe to keyboard
32   ros::Subscriber sub2; // subscribe to tracker_1: x,y,theta
33   void callBack(const geometry_msgs::Twist::ConstPtr& msg);
34   // subscribe to the keyboard
35   void callBack2(const geometry_msgs::Point::ConstPtr& msg);
36   // subscribe to the tracker_1
37   double fcn(double xp,double yp); // to compute the theta_err
38   // (refer matlab Cartesian code)
39   geometry_msgs::Twist vel; std_msgs::Float64MultiArray expData;
40   double wr; double wl; // for now the values are going to be in
41   // micro seconds to test the rpm of the motor and log the data
42   int cruise = 0; int initial = 0;
43   double x_i; double y_i; double theta_i; double x_f; double y_f;
44   double theta_f;
45   double v_ref; double w_ref; double theta_ref; double x_ref;
46   double y_ref;
47   double d_err; double theta_err; double k_theta = 6.0;
48   double k_v = 1.0;
49   double xp; double yp; // used by function 'fcn'
50   std::string line; std::string sX_ref; std::string sY_ref;
51   std::ifstream ifile
52   {"/home/smanuel/catkin_ws/src/highBW/src/Cartesian.BW.csv"};
53 };
54
55 readData::readData(){
56   sub2 = n.subscribe<geometry_msgs::Point>("/tracker_1", 1,
57   &readData::callBack2,this);
58   sub = n.subscribe<geometry_msgs::Twist>("/keyboard",10,
59   &readData::callBack,this);
60   pub = n.advertise<geometry_msgs::Twist>("cmd_vel",1);
61   pub2 = n.advertise<std_msgs::Float64MultiArray>("exp_data",10000);
62 }
63
64 void readData::callBack(const geometry_msgs::Twist::ConstPtr& msg){
65   if (msg->linear.x == 2) cruise = 1;
66   else {
67     cruise = 0;
68     initial = 0; // resets the initial to record the initial values
69     // of x,y,theta
70   }
71 }
72
73 void readData::callBack2(const geometry_msgs::Point::ConstPtr& msg){
74   if (cruise == 1){
75     if (initial == 0){
76       x_i = msg->x;
77       y_i = msg->y;
78       theta_i = msg->z;
79       initial = 1;
80     }
81     x_f = (msg->x - x_i)*cos(theta_i) + (msg->y - y_i)*sin(theta_i)
82     + 500.0;
83     // add the starting value of the robot instead of 500
84     y_f = -(msg->x - x_i)*sin(theta_i) + (msg->y - y_i)*cos(theta_i)
85     + 501.5;
86     theta_f = msg->z - theta_i;

```

```

87
88     if (std::getline(ifile, line)) { // read the current line
89         std::istringstream iss{line}; // construct a string stream
90         // from line
91         std::getline(iss, sX_ref, ',');
92         std::getline(iss, sY_ref, ',');
93
94         //ROS_INFO("%s\n", sV_ref.c_str());
95         x_ref = std::stod(sX_ref);
96         y_ref = std::stod(sY_ref);
97     }
98
99     // outerloop code
100    xp = x_ref - x_f;
101    yp = y_ref - y_f;
102
103    theta_ref = fcn(xp, yp);
104    theta_err = theta_ref - theta_f;
105
106    v_ref = sqrt(pow(xp,2) + pow(yp,2))*cos(theta_err)*k_v;
107    w_ref = k_theta * theta_err;
108
109    wr = (2*v_ref + Length*w_ref)/(2*Radius);
110    wl = (2*v_ref - Length*w_ref)/(2*Radius);
111
112    vel.linear.x = wr;
113    vel.angular.x = wl;
114    vel.linear.z = 1;
115    pub.publish(vel); // cmd_vel to the inner loop
116
117    expData.data = { x_f, y_f, theta_f, x_ref, y_ref};
118    pub2.publish(expData);
119
120 }
121 else {
122     wr = 0.0;
123     wl = 0.0;
124     vel.linear.z = 0;
125     vel.linear.x = wr;
126     vel.angular.x = wl;
127     pub.publish(vel); // cmd_vel to the inner loop
128
129 }
130
131 }
132 }
133
134 double readData::fcn(double xp, double yp){
135     static double xo = 1.0, yo = 0.0, theta = 0.0;
136     double a1,b1,c1,dtheta,sign;
137
138     // calculate the value of theta
139     a1 = sqrt( pow(xp,2) + pow(yp,2) );
140     b1 = sqrt( pow(xo,2) + pow(yo,2) );
141     c1 = sqrt( pow(xp - xo, 2) + pow(yp - yo ,2) );
142
143     if ((a1 != 0) && (b1 != 0))

```

```

144         dtheta = acos(std::max(std::min( (pow(a1,2) + pow(b1,2) -
145         pow(c1,2))/(2*a1*b1) ,1.0),-1.0));
146     else
147         dtheta = 0.0;
148
149     // calculate the direction of rotation
150     sign = 1.0;
151     if (dtheta != 0){
152         sign = (-yo/xo)*xp + yp;
153         if (sign != 0){
154             if ((xo > 0) && (yo > 0))
155                 sign = sign/std::abs(sign);
156             else if ((xo < 0) && (yo < 0))
157                 sign = -sign/std::abs(sign);
158             else if ((xo > 0) && (yo < 0))
159                 sign = sign/std::abs(sign);
160             else if ((xo < 0) && (yo > 0))
161                 sign = -sign/std::abs(sign);
162             else if ((xo == 0) && (yo >= 0)){
163                 if (xp != 0)
164                     sign = -xp/std::abs(xp);
165                 else
166                     sign = 0;
167             }
168             else if ((xo == 0) && (yo <= 0)){
169                 if (xp != 0)
170                     sign = xp/std::abs(xp);
171                 else
172                     sign = 0;
173             }
174             else if ((xo >= 0)&&(yo == 0)){
175                 if (yp != 0)
176                     sign = yp/std::abs(yp);
177                 else
178                     sign = 0;
179             }
180             else if ((xo <= 0) && (yo == 0)){
181                 if (yp != 0)
182                     sign = -yp/std::abs(yp);
183                 else
184                     sign = 0;
185             }
186         }
187     }
188
189     xo = xp; yo = yp;
190     theta = theta + dtheta*sign;
191     return theta;
192 }
193
194
195
196 int main(int argc, char **argv){
197     ros::init(argc, argv, "Cartesian");
198
199     //TeleopJoy teleop_turtle;
200     readData dude;

```

```

201
202     ros::spin();
203
204     return 0;
205 }

1
2 //Description: ROS Node for calculating (v,w) from pose values
3 //(x,y,theta)
4 #include "ros/ros.h"
5 #include "ros/time.h"
6 #include "std_msgs/String.h"
7 #include "std_msgs/Int8.h"
8 #include "std_msgs/Float64MultiArray.h"
9 #include <cmath>
10 #include <tf/tf.h>
11 #include<geometry_msgs/Vector3.h>
12 #include<geometry_msgs/Vector3Stamped.h>
13 #include<geometry_msgs/Twist.h>
14 #include<geometry_msgs/Point.h>
15 #include<geometry_msgs/PoseWithCovarianceStamped.h>
16 #include<sensor_msgs/Joy.h>
17 #include <sstream>
18 #include <iostream>
19 #include <fstream>
20
21 int buffer_length = 50;
22 std::deque<double> filterbuffer_v(buffer_length,0.0);
23 std::deque<double> filterbuffer_w(buffer_length,0.0);
24
25 class readData{
26     public:
27         readData();
28     private:
29         ros::NodeHandle n;
30         ros::Publisher pub;
31         ros::Subscriber sub;
32         //ros::Subscriber sub2;
33         void callBack(const std_msgs::Float64MultiArray::ConstPtr& msg);
34         void callBack2(const geometry_msgs::Twist::ConstPtr& key);
35         geometry_msgs::Twist vel;
36         geometry_msgs::Point pre_msg;
37         std_msgs::Float64MultiArray exp_dataRecord;
38         double time; double expTime = 0.0;
39         double ts = 1.0/105.0;
40         double pre_time2 = 0.0;
41         double pre_time = 0.0;
42         double inA; double inB;
43         double yaw = 0.0; // this is absolute yaw angle
44 };
45
46
47 // using Joy will cause serious problems – when joy is not publishing on
48 // to the topic
49 readData::readData(){
50     sub = n.subscribe<std_msgs::Float64MultiArray>("exp_data", 10000,

```

```

51     &readData::callBack, this);
52     pub = n.advertise<std_msgs::Float64MultiArray>("exp_dataRecord",
53     10000);
54     //sub2 = n.subscribe<geometry_msgs::Twist>("/keyboard",10,
55     //&readData::callBack,this);
56     pre_msg.x = 0.0; pre_msg.y = 0.0; pre_msg.z = 0.0;
57     }
58
59 void readData::callBack(const std_msgs::Float64MultiArray::
60 ConstPtr& msg){
61
62     time = ros::Time::now().toSec();
63     vel.linear.y =
64     sqrt(pow((msg->data[0] - pre_msg.x)/(time - pre_time),2) +
65     pow((msg->data[1] - pre_msg.y)/(time - pre_time),2));
66     // the next four conditions are not necessary, they are covered
67     // by the last 4, all the if conditions are not necessary as
68     // well, by default the vel.linear.y is always positive
69
70     // convert the theta from relative to absolute
71     if (!((msg->data[2]) > -3.141) && (msg->data[2] < 3.141))){
72     // it theta is out of -pi to pi range enter the loop
73         yaw = msg->data[2];
74         if (msg->data[2] > 0){
75             while(!((yaw > -3.141) && (yaw < 3.141)))
76                 yaw = yaw - (2*3.141);
77         }
78         if (msg->data[2] < 0){
79             while(!((yaw > -3.141) && (yaw < 3.141)))
80                 yaw = yaw + (2*3.141);
81         }
82     }
83     else yaw = msg->data[2];
84
85
86     if ( ((msg->data[0] - pre_msg.x)>0) && ((msg->data[1] -
87     pre_msg.y)==0)){
88         if (std::abs(yaw) > (3.141/2))
89             vel.linear.y = vel.linear.y;
90         else
91             vel.linear.y = -vel.linear.y;
92     }
93     else if ( ((msg->data[0] - pre_msg.x)==0) && ((msg->data[1] -
94     pre_msg.y)<0))
95     {
96         if (yaw > (0))
97             vel.linear.y = vel.linear.y;
98         else
99             vel.linear.y = -vel.linear.y;
100    }
101    else if ( ((msg->data[0] - pre_msg.x)<0) && ((msg->data[1] -
102    pre_msg.y)==0))
103    {
104        if (std::abs(yaw)<(3.141/2))
105            vel.linear.y = vel.linear.y;
106        else
107            vel.linear.y = -vel.linear.y;

```

```

108     }
109     else if (((msg->data[0] - pre_msg.x)==0)&&((msg->data[1] -
110     pre_msg.y)>0))
111     {
112         if (yaw < (0))
113             vel.linear.y = vel.linear.y;
114         else
115             vel.linear.y = -vel.linear.y;
116     }
117     else if (((msg->data[0] - pre_msg.x)>0)&&((msg->data[1] -
118     pre_msg.y)>0))
119     {
120         if ((yaw < (-3.141/2))&&(yaw > (-3.141/1)))
121             vel.linear.y = vel.linear.y;
122         else
123             vel.linear.y = -vel.linear.y;
124     }
125     else if (((msg->data[0] - pre_msg.x)>0)&&((msg->data[1] -
126     pre_msg.y)<0))
127     {
128         if ((yaw > (3.141/2))&&(yaw < (3.141/1)))
129             vel.linear.y = vel.linear.y;
130         else
131             vel.linear.y = -vel.linear.y;
132     }
133     else if (((msg->data[0] - pre_msg.x)<0)&&((msg->data[1] -
134     pre_msg.y)<0))
135     {
136         if ((yaw > (0))&&(yaw < (3.141/2)))
137             vel.linear.y = vel.linear.y;
138         else
139             vel.linear.y = -vel.linear.y;
140     }
141     else if (((msg->data[0] - pre_msg.x)<0)&&((msg->data[1] -
142     pre_msg.y)>0))
143     {
144         if ((yaw < (0))&&(yaw > (-3.141/2)))
145             vel.linear.y = vel.linear.y;
146         else
147             vel.linear.y = -vel.linear.y;
148     }
149
150
151     filterbuffer_v.push_front(vel.linear.y); // buffer implementation
152     double sum_v = 0.0;
153     for (int i = 0; i < buffer_length; i++){
154         sum_v = sum_v + filterbuffer_v[i];
155     }
156     vel.linear.y = sum_v/(buffer_length); // buffer end
157     filterbuffer_v.pop_back();
158     pre_msg.x = msg->data[0];
159     pre_msg.y = msg->data[1];
160     pre_time = time;
161     if (std::abs(vel.linear.y) < 0.015)
162     vel.linear.y = 0;
163
164

```



```

165     //if ((msg->data[2]*pre_msg.z)>0)    // this conditions is
166     // required when the range of theta is from -pi to pi and it's
167     // absolute. Now it's relative to the starting value i.e. it
168     // keeps increasing or decreasing from the starting value.
169     //{
170     time = ros::Time::now().toSec();
171     vel.angular.y = (msg->data[2] - pre_msg.z)/(time - pre_time2);
172     filterbuffer_w.push_front(vel.angular.y);
173     double sum_w = 0.0;
174     for (int i = 0; i < buffer_length; i++){
175     sum_w = sum_w + filterbuffer_w[i];
176     }
177     vel.angular.y = sum_w/(buffer_length);
178     filterbuffer_w.pop_back();
179     pre_msg.z = msg->data[2];
180     pre_time2 = time;
181     //}
182     //else {
183     //pre_msg.z = msg->data[2];
184     //}
185
186     if (std::abs(vel.angular.y) < 0.01)
187     vel.angular.y = 0;
188
189
190     exp_dataRecord.data = {msg->data[0], msg->data[1], msg->data[2],
191     -vel.linear.y, vel.angular.y, expTime, msg->data[3],
192     msg->data[4]};
193     expTime = expTime + ts;
194
195     pub.publish(exp_dataRecord);
196
197 }
198
199 void readData::callBack2(const geometry_msgs::Twist::ConstPtr& key){
200
201 }
202
203 int main(int argc, char **argv)
204 {
205     ros::init(argc, argv, "ground_station_innerLoop2");
206
207     //TeleopJoy teleop_turtle;
208     readData dude;
209
210     ros::spin();
211
212     return 0;
213 }

```