POLITECNICO DI TORINO

Master Degree in Aerospace Engineering

Master Degree Thesis

Modelling and Control of a Skid-Steering Mobile Robot for Indoor Trajectory Tracking Applications

A framework for the performance evaluation of PID and Sliding Mode controllers for trajectory tracking of a non-holonomic wheeled mobile robot in indoor environment

Supervisors
Prof. Giorgio Guglieri
Prof.ssa Elisa Capello

Candidate
Davide Locardi
matricola: 249043

External Supervisor
Assistant Prof. Daniele Sartori
Shanghai Jiao Tong University

April 2020

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

This thesis was written with the supervision of Assistant Prof. Daniele Sartori from the department of Electronic Information and Electrical Engineering of the Shanghai Jiao Tong University, China.

北斗导航创新研究院
Shanghai Beidou Research Institute

The experimental tests on the real Clearpath Husky platform were made in collaboration with the Shanghai BeiDou Research Institute in the indoor Robotics Testing and Training Area located in West Hongqiao, Shanghai, China.

Dedicated to my family and friends.

Abstract

The present Master thesis addresses the problem of the modelling and control of a non-holonomic Unmanned Ground Vehicle (UGV) for trajectory tracking application. Much effort has been put by the scientific community in the research and development of new modelling methods and in the study of new techniques aimed at finding suitable control laws for this category of vehicle. All this interest is due both by the high number of possible applications for which these platforms are ideal. For example, all those characterised by Dull, Dirty and Dangerous (D^3) tasks that have made them gain great fame even in military or civilian fields. Non-holonomic UGVs, and in particular Four-Wheeled Skid-Steering Mobile Robots (4W-SSMRs) which is that vehicle class directly analysed by this thesis, are characterised by highly non-linear behaviours, especially concerning wheel-soil interaction, which requires advanced design methods. This work aims to provide a valid comparison between two different control techniques through numerical simulations and experimental tests.

The vehicle dynamics, along with its linearisation, is first derived using the Euler-Lagrange approach and the problem of determining the resistive forces due to the wheel-soil interaction has been overcome by proposing an advanced friction model not yet used for this type of applications. Furthermore, the major system uncertainties have been resolved exploiting a system identification process based on evolutionary algorithm.

After that, two controllers have been designed and implemented: the straightforward Proportional Integrative Derivative (PID) controller and the more advanced Sliding Mode Control (SMC) strategy. Both of them have been analysed from a theoretical point of view, validated through numerical simulations with the developed mathematical model and experimentally tested and compared on the real robotic platform on four different trajectories in the indoor test facility of the Shanghai BeiDou Research Institute (BRI). Besides, the optimal parameter tuning of both controllers was done following a heuristic approach based on a Differential Evolution (DE) algorithm.

The PID approach, thanks to its ease of implementation and being known by the whole research community, has served as a yardstick for the most advanced controller. In particular, its structure is composed of two different branches for longitudinal and latero-directional control. Also, for improving its tracking performance, a feed-forward loop has been implemented for speed control. Nevertheless, a trade-off between robustness and performance is necessary when dealing with this standard control technique, being subject to performance limitations, especially when dealing with non-linear systems affected by uncertainties and parametric perturbations.

The robustness problem has tried to be solved by the adoption of the more advanced Variable Structure Control (VSC) theory which, thanks to the application of a non-linear high-frequency switching control, alters the dynamics of the non-linear system under analysis forcing it into a *sliding motion* along a cross-section of the system's normal behaviour, allowing us to consider a reduced-order model of plant dynamics. Besides, another advantage is that of having low sensitivity to matched uncertainty. In this master thesis is employed a multi-variable sliding mode controller developed on a new switching function

design in which the lateral and the angular error are coupled in a sliding surface leading to the convergence of both variables. Besides, the chattering problem deriving from the discontinuity of the infinitely fast switching control architecture is suppressed by the use of a continuous quasi sliding mode approach. It is demonstrated that the achieved configuration guarantees satisfying performance also when applied to a complete non-linear model affected by uncertainties and parametric perturbations. Due to the SMC potentiality in dealing with complex and non-linear systems, such as the Skid-Steering Mobile Robots (SSMRs), satisfying results have been achieved even when the non-linear model has been perturbed by sensor noise and parametric uncertainties.

A final confrontation among the two controllers applied to the Clearpath Husky platform shows that PID and SMC are comparable at low tracking velocities in terms of nominal performance. However, when the control performance demand is increased, the implemented PID architecture always resulted inferior.

The simulation scenario, the mobile robot plant and the proposed controllers have been developed using MATLAB R2019b and Simulink which, thanks to the numerous toolboxes, have made that work easy to implement. Particular importance was gained by the Robot Operating System (ROS) toolbox which made possible to greatly simplify the interface between the developed controllers and the Husky platforms, allowing us to carry out the experimental tests.

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Acronyms

D^3 Dull, Dirty and Dangerous

4W-SSMR Four-Wheeled Skid-Steering Mobile Robot

ABC Artificial Bee Colony

ADAS Advanced Driver-Assistance System

AGV Automated Guided Vehicle

AI Artificial Intelligence

BDS BeiDou Navigation Satellite System

BRI BeiDou Research Institute

CBRNE Chemical, Biological, Radiological, Nuclear, Explosive

CMU Carnegie Mellon University

CoG Centre of Gravity

DARPA Defense Advanced Research Projects Agency

DE Differential Evolution

DIMEAS Department of Mechanical and Aerospace Engineering

EA Evolutionary Algorithm

EKF Extended Kalman Filter

EOD Explosive Ordnance Disposal

GA Genetic Algorithm

GLONASS GLObal NAvigation Satellite System

GPS Global Positioning System

HIL Hardware in the Loop

ICR Instantaneous Center of Rotation

IFR International Federation of Robotics

IMU Inertial Measurement Unit

IP International Protection

ISN Institute for Sensing and Navigation

JPL Jet Propulsion Laboratory

LAAS Laboratory for Analysis and Architecture of Systems

LIDAR Laser Imaging Detection and Ranging

LQ Linear Quadratic

LQR Linear Quadratic Regulator

LTI Linear Time Invariant

MER Mars Exploration Rover

MGI McKinsey Global Institute

MIMO Multi Input Multi Output

MSc Master of Science

NASA National Aeronautics and Space Administration

PBIL Population Based Incremental Learning

PID Proportional Integrative Derivative

PSO Particle Swarm Optimization

RADAR Radio Detection and Ranging

RC Remote Control

RMS Root Mean Square

ROS Robot Operating System

RSTA Reconnaissance, Surveillance and Target Acquisition

RTS Realis Tracking System

SA Simulated Annealing

SAR Search and Rescue

SBAS Satellite-based Augmentation Systems

SIL Software in the Loop

SISO Single Input Single Output

SJTU Shanghai Jiao Tong University

SLAM Simultaneous Localization and Mapping

SMC Sliding Mode Control

SONAR Sound Navigation and Ranging

SRI Stanford Research Institute

SSMR Skid-Steering Mobile Robot

STRIPS Stanford Research Institute Problem Solver

TCP Transmission Control Protocol

UAV Unmanned Aerial Vehicle

UDP User Datagram Protocol

UGV Unmanned Ground Vehicle

USV Unmanned Surface Vehicle

UUV Unmanned Underwater Vehicle

UWB Ultra-Wideband

UXO Unexploded Ordnance

VR Virtual Reality

VRPN Virtual-Reality Peripheral Network

VSC Variable Structure Control

VSCS Variable Structure Control System

WMR Wheeled Mobile Robot

ZOH Zero Order Hold

List of Symbols

$\boldsymbol{A}$ Constraint vector

e_{bend} Total bending energy

c_i Local curvature

$\mathscr{S}_g\,(X,\ Y,\ Z)$ Global coordinate system

$\mathscr{S}_l\,(x,\ y,\ z)$ Local coordinate system

$\mathscr{S}$ Generic reference system

CR Crossover rate

D Number of parameter to be optimized

DC Direct current

δv_i General deviation

df Objective function relative error threshold

d_{haus} Hausdorff distance

$delay$ Commands delay

d_{ratio} Distance ratio

d_{rms} RMS of minimum distances

d_T Generic euclidean distance

$\boldsymbol{E}$ Input transformation matrix

E Kinetic energy

$e_{r_{bend}}$ Total bending energy ratio

$e\,(t)$ Generic signal error

$\boldsymbol{\eta}$ Kinematic control input vector

$\Phi,\ \Theta,\ \Psi$ General Euler angles [rad]

$\boldsymbol{F}$ Vector of resultant active forces and torques

F_x Resulting active force

F_{att} Active force/torque applied to the system

FDJ Dynamic friction force/torque

FF Effective friction force/torque

F_f General friction force

FF Feed-forward gain

f_{opt} Cost function used in the optimization process

FSJ Static friction force/torque

F_y Resulting resistive force

g Gravity acceleration: 9.81 $\mathrm{m/s^2}$

$\boldsymbol{\Gamma}$ Active torque and force vector

$a,\ b,\ W$ Mobile robot geometric constants

I Mobile robot moment of inertia [$\mathrm{kg\,m^2}$]

I_m Motor inertia

$\boldsymbol{\mathcal{X}}_{i,j}$ Generic ith individual at generation j

K_D PID derivative gain

K_I PID integral gain

K_i General viscous coefficient

K_P PID proportional gain

k_s Signum approximation accuracy

K_w Wheels torque controller proportional gain

$\mathcal{L}$ Lagrangian

$\boldsymbol{\lambda}$ Vector of Lagrange multipliers

$\boldsymbol{\mathcal{X}}_{min}$ Search space lower parameter bounds

$\boldsymbol{M}$ Inertia matrix

m Vehicle mass

M Resulting active torque

F Mutation scale factor

M_r Resulting resistive moment

μ_c Coulomb friction coefficient

μ_{cy} Lateral friction coefficient

$\boldsymbol{\mathcal{V}}_{i,j}$ Generic ith mutant vector at generation j

μ_v Viscous friction coefficient

N Generic normal force

NG Number of generations

NP Number of parameter vectors in one population

nr Number of vectors used for mutation

$\boldsymbol{q}$ UGV state vector

$\boldsymbol{R}$ Vector of resultant reactive forces and torque

r Effective rolling radius

$\boldsymbol{R_{lg}}$ local-global rotation matrix

$\boldsymbol{R_{21}}$ Complete rotation matrix

$[\boldsymbol{\Phi}]$, $[\boldsymbol{\Theta}]$, $[\boldsymbol{\Psi}]$ Elementary rotation matrices associated with general Euler angles

R_x Resulting rolling resistance

$\boldsymbol{S}$ General coordinates transformation matrix

dt Simulink fixed step time

T Generic Trajectory

τ_1 F adaptation parameter

τ_2 CR adaptation parameter

$\boldsymbol{\tau}$ Torque control input vector

$\boldsymbol{\tau}_{max}$ Torque limits

ϑ Heading angle

$\boldsymbol{\mathcal{U}}_{i,j}$ Generic ith trial vector at generation j

U Potential energy

$u_c\,(t)$ Generic control signal

$\boldsymbol{\mathcal{X}}_{max}$ Search space upper parameter bounds

u_x Longitudinal rolling coefficient

$\mathcal{V}\,(\boldsymbol{x}, t)$ Lyapunov function candidate

$\boldsymbol{V}$ UGV body velocity vector

ω_z Angular component of the velocity vector $\boldsymbol{V}$ along body axes [rad/s]

v_x, v_y Linear components of the velocity vector $\boldsymbol{V}$ along body axes [m/s]

$\bar{v}_i$ General equilibrium point

$\dot{\boldsymbol{q}}$ UGV global velocity vector

v Generic wheel linear velocity

$\boldsymbol{\omega_w}$ Wheels angular velocity vector

ξ Gradient descent constant slope

XTE Cross-track error

Acknowledgements

I wish to thank all the people that directly or indirectly helped me with the realization of this project, starting from Prof. Giorgio Guglieri from Politecnico di Torino and Assistant Prof. Daniele Sartori from Shanghai Jiao Tong University for the big opportunity to develop this work and for their infinite availability. A big thank you goes to Filippo Arbinolo, with whom I shared this experience in Shanghai and who, together with Manfredi di Rovasenda, Gabriele Ermacora and Giovanni Serito helped to give a sense of normalcy after immersion in an entirely different culture, such as Chinese one. I thank my older friends Sean, Victor, Samuele and Alessandro for their constant support; Alex, Luigi, Luca, Roberto and all the fantastic people known in these five years who have contributed, each in its way, to lightening and making this journey unique. Finally, I would like to thank my parents, Alessandro and Cristina, for their unconditional support, for having always believed in me and for having constantly represented the model to be aspired to, making me the person who I am.

Preface

The current Master thesis is realized within the Department of Mechanical and Aerospace Engineering (DIMEAS) of the Polytechnic of Turin for the fulfilment of the Master of Science (MSc) degree in Aerospace Engineering. The research project was carried out for a total of four months at Shanghai Jiao Tong University (SJTU) Institute for Sensing and Navigation (ISN) and thanks to them collaboration with BeiDou Research Institute, it was possible to test the obtained results in their indoor test facility, also located in Shanghai, China.

Both of these universities are active in a wide range of research projects related to unmanned vehicles. The first one, thanks to its aerospace background mostly focusses on the development of unmanned aircraft with an emphasis on their control system. An example is the research carried out by DRAFT student team, which aims to increase the autonomy of today's Unmanned Aerial Vehicle (UAV) technology by developing innovative solution focussed on artificial intelligence. The ISN, managed by the Department of Electronic Information and Electrical Engineering, through the *Key Laboratory of Navigation and Location-based Services* and the *Shanghai Key Laboratory of Intelligent Sensing and Recognition* mainly researches on robotics, navigation and localization services, radar detection and remote sensing. Some of the on-going projects include the realisation of a next-generation Inertial Navigation Computation able to improve the accuracy of an ideal IMU up to a meter error per hour, the development of a 3D vehicle motion planning and obstacle avoidance algorithm for cluttered environments and the implementation of a millimetre-wave Simultaneous Localization and Mapping (SLAM) for Advanced Driver-Assistance System (ADAS) applications.

The aim of this project, which during the mobility period was carried out in collaboration with another student also coming from the Polytechnic of Turin, was the design and the implementation of three different control laws for an unmanned wheeled mobile robot. The request under this work was first to collaborate on the development of a classic PID approach to acquire confidence with the control technique applied to this particular class of vehicles and to assess its performances, necessary as a reference for the subsequent comparison with an advanced control method on which each of us had to specialize: a SMC strategy and a Linear Quadratic Regulator (LQR) controller. The idea was to provide an ultimate confrontation among the three different techniques independent from the type of controller adopted or from the WMR mathematical model used for the tests.

By going into detail, the specialization of the present research was made on the sliding mode control theory because this approach allows us to consider a reduced-order modelling of plant dynamics and moreover gives the advantage of having low sensitivity to

matched uncertainty. After each controller was identified, the vehicle dynamics was first derived. Then the following phase consisted in the study and definition of the control laws mathematical basis. The theoretical foundations derived was generally taken from existing work, but was elaborated and combined to guarantee an innovative contribution for every case. For example, the main problems regarding the mathematical modelling of this category of vehicles, which are the determination of the resistive forces due to the wheel-soil interaction and the necessity to deal with system uncertainties, have been overcome respectively by proposing an advanced friction model not yet used for this type of applications, see Section 2.2.2, and an evolutionary algorithm based heuristic identification process, see Section 3.4.

Regarding the SMC, as explained in Section 4.4.3, a new switching function design is proposed in which, the lateral and the angular error are coupled. The realization of numerical simulations was used to validate the proposed approaches and finally, the controllers were tested and compared on the real robotic platform. The combination of expertise achieved from the different research fields of the two universities allowed us to overcome various problems that arose during the progress of the project, in particular concerning our limited knowledge of robotics. Different practical problems were also solved in the experimental test phase, which leads to the controllers' parameters adjustment. Unfortunately, the coincidence of the test period with the lunar new year holidays and with COVID-19 outbreak in mainland China resulted in a prolonged university and indoor test facility lockdown, which caused our consequent repatriation thus limiting the number of experimental tests carried out.

The thesis is structured as follows: Chapter 1 introduces the UGV technology and defines the tasks addressed in this work, going from the SSMR control problem to the proposed control techniques, suggesting different solutions emphasizing their contribution to state of the art. Chapter 2 describes the developed linear and non-linear mathematical models of the mobile robot adopted for the definition and the testing of the controllers. In Chapter 3, the experimental set-up adopted in this thesis is described, going from the BeiDou Research Institute indoor test facility to the Clearpath Husky testing platform and ending with the benefits and drawbacks of the exploited frameworks and tools. Besides the experimental set-up, part of the chapter is dedicated to the description of the parameter estimation method used for dealing with some model uncertainties. Chapters 4 and 5 respectively address the design process followed by the results of the software simulations of the proposed controllers and their final experimental validation and comparison made on the physical platform. The problems encountered and the adopted solutions are here illustrated, and the main results of the preliminary tests are presented. Finally, Chapter 6 draws the conclusions and suggests possible improvements that can be developed in future works. The thesis ends with one appendices, Appendix A, which describes and reports the pseudo-code of the self-adaptive evolutionary algorithm used for both parameters estimation and optimal tuning of the controllers' gains.

Chapter 1

Introduction

Everyday robotics finds more and more applications in the real world, replacing man in those jobs that until now have been carried out manually. Their use, from agriculture and mining operations, up to the use inside factories and hospitals, is increasing performance, efficiency and safety in all tasks otherwise considered too dull, dirty or dangerous for manual work.

Robotics has achieved its greatest success to date in the world of industrial manufacturing. According to a recent study carried out by McKinsey Global Institute (MGI) in collaboration with International Federation of Robotics (IFR), in 2017 the industrial robotics market grew by 19%, reaching USD 48 billion, see Fig. 1.1, and, thanks to the reduction in production costs, the increasing variety of models, the ease of integration, the increase in the cost of labour and ever greater technical skills achievable, it was estimated to continue double-digit grow at least through to 2021 [1].

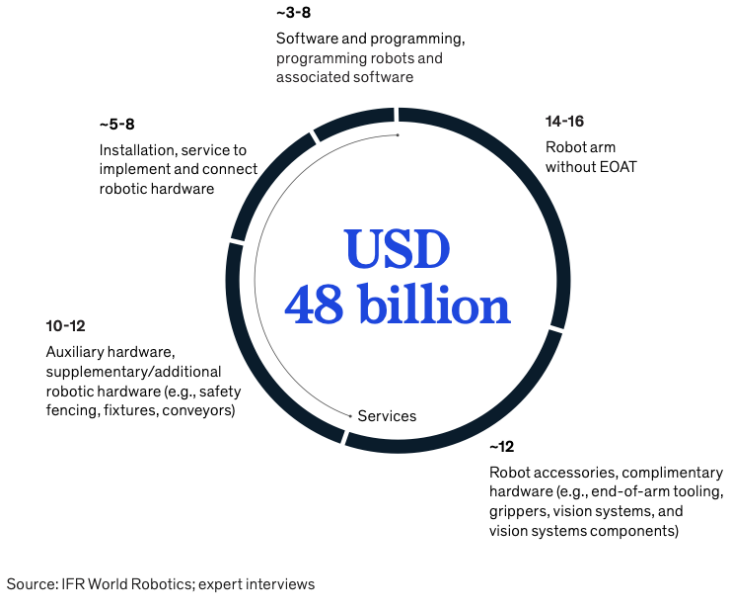


Figure 1.1: Robot systems 2017 market breakdown revenues.

Thanks to the continuous technological evolution, the robotic arm can move with high speed and precision to perform repetitive activities such as painting or welding. However, despite their great success, these robots suffer from one major disadvantage: the lack of mobility. A fixed manipulator has a limited range of motion which depends on where it is positioned in the assembly line. On the contrary, a mobile robot can travel through the production plant, applying its talents flexibly wherever it is most effective. This great advantage, however, entails the need to have a control system that allows the robot to navigate from a point A to a point B and to perform its tasks efficiently and safely.

1.1 Overview on UGV Technology

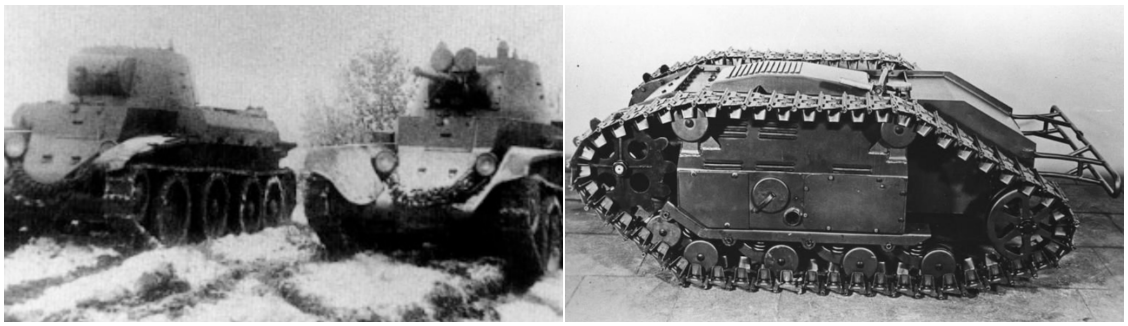
UGVs together with their marine-based[1] and aerial-based[2] counterparts, are part of mobile robotics. This acronym comprehends a broad range of *powered vehicle that does not carry a human operator, can be operated autonomously or remotely while in contact with the ground, can be expendable or recoverable, and can carry a lethal or non-lethal payload. Ballistic or semi-ballistic vehicles, cruise missiles, artillery projectiles, torpedoes, mines, satellites, and unattended sensors (with no form of propulsion) are not considered unmanned vehicles* [2].

The first documented UGV appeared in October 1921 [3, 342-343] when World Wide Wireless magazine published the news that Captain R.E. Vaughn, chief of the McCook Field radio section, controlled a car through radio signals, showing the possibilities of Remote Control (RC) locomotion and it was thought this immature technology could someday be adapted in the military field. The interest in this category of vehicles has its origins in the military environment during World War II. The Soviet Union was the first to apply this technology; they developed radio-controlled TT-26 tanks, each of which was controlled by another tank about $500 \div 1500$ m away, see Fig. 1.2a. The *"tele-tanks"* were armed with machine guns, flame-throwers, or loaded with explosives that could be detonated remotely behind enemy lines. On the other side of the conflict, the Germans also used remote assistance on land vehicles developing the *"Beetle"* tank, see Fig. 1.2b, which was not considered a great success as it travelled very slowly, and had a low ground clearance. Besides, the main unit was connected to a small control box via a triple-strand of telephone wires that were vulnerable to entanglement and damage [4].

After the war, the attention shifted to the automation of these vehicles. In universities, teams of researchers focused on how the robot could generate a response based on the surrounding environment and no longer following the command of an external operator. The first fully autonomous UGV was developed by William Walter at the Burden Neurological Institute in Bristol [5]. The robot, called *"Elsie"*, see Fig. 1.3a, was able to respond to contact with external objects and to perceive light through photoelectric sensors. Relying on these to navigate, they used two vacuum tube amplifiers that guided the relays to control the steering and drive the engines.

[1] Unmanned Underwater Vehicles (UUVs) and Unmanned Surface Vehicles (USVs)

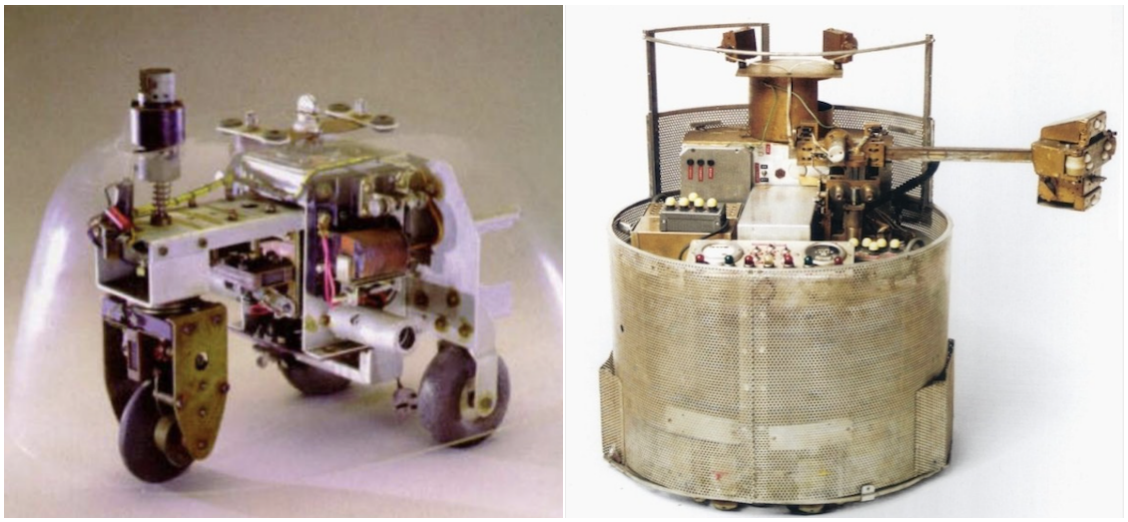
[2] UAVs



(a) URSS *"Tele-tank"* (1941). (b) German *"Beetle"* (1940).

Figure 1.2: The First Autonomous Ground Vehicles.

The next more sophisticated autonomous vehicle appeared in the early 1960s, researchers at the Johns Hopkins University Applied Physics Lab created *"The Beast"*, see Fig. 1.3b. Controlled by dozen of transistors the Hopkins Beast wandered the University's hallways centring itself using Sound Navigation and Ranging (SONAR) until its batteries run low. Then it would seek black wall outlets with special photocell optics, and plug itself in by feel with its special recharging arm [4].



(a) William Walter's *"Elsie"* (1949). (b) Johns Hopkins *"Beast"* (early 1960s).

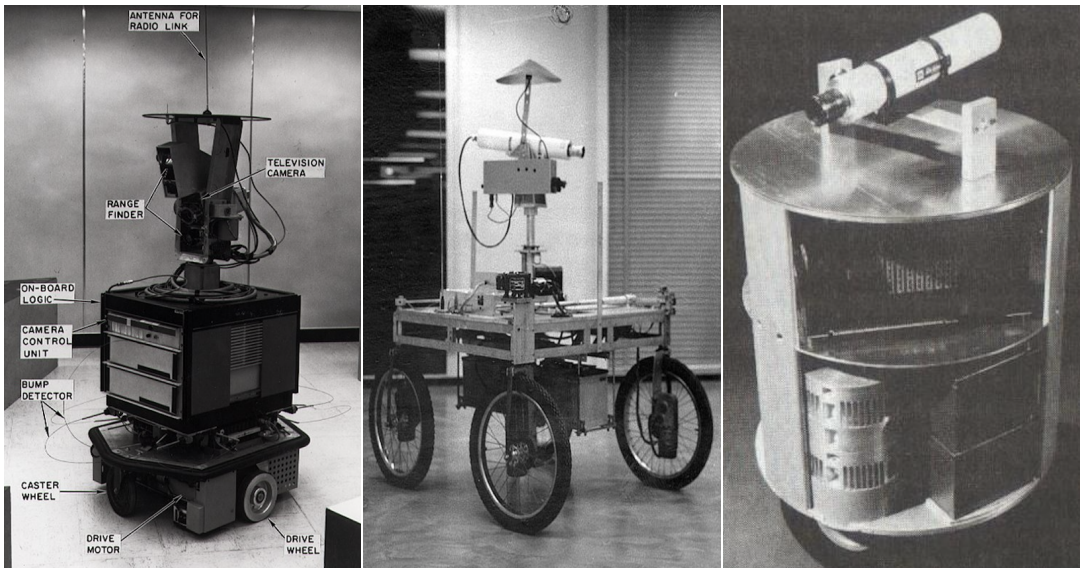
Figure 1.3: The First Autonomous Ground Vehicles.

Later in this decade Stanford University and Stanford Research Institute (SRI) developed the first computer-controlled autonomous UGVs: "Cart" and "Shakey". The primary purpose of these projects was to study the processes for perception, communication, decision making and the real-time control of a robotic system interacting with a complex environment [6].

In the same period, a second and slightly different research trend started. The aim

was to solve the problem of locomotion of robotic vehicles on rough terrain. The goal of the work was the design and study of the kinematics and dynamics of multi-legged robots. The main research thrust was aimed at designing the structure and coordinating the legs. Work was done on four-legged structures, for example, the General Electric *Quadruped* [7] and on biped systems, but the most popular test benches were the six-legged ones. In 1980 a six-legged robot, the *Hexapoda*, began to be built at the University of Paris VII, Laboratoire de Robotique et Intelligence Artificielle [8], but the problems that remained to be solved to obtain satisfactory control over this type of locomotion were mainly related to the limited computing power and the reduced environmental perception [8, 9].

The first major WMR development effort, called *Shakey*, see Fig. 1.4a, was created in 1966 as a research study on Artificial Intelligence (AI) for the Defense Advanced Research Projects Agency (DARPA) at the SRI [6]. It was a platform on wheels equipped with an adjustable TV camera for capturing images of its environment, ultrasound range finder for sensing its distance from walls and other objects and bump detectors for detecting the collisions with the external environment. It was connected via RF to his mainframe computer which performed navigation and exploration tasks.



(a) SRI's *"Shakey"* (1966). (b) Stanford *"Cart"* (1963). (c) CMU *"Rover"* (1981).

Figure 1.4: The First Autonomous Ground Vehicles.

Shakey's development has led to numerous results that had a far-reaching impact on the fields of robotics, computer science and AI. The most notable include the development of the $A^\star$ search algorithm [10], which is widely used in path-finding[3] and graphs traversal[4]; the Visibility Graph method [6, 11], used for find the shortest Euclidean

[3]Is the plotting, by a computer application, of the shortest route between two points. This field of research is based heavily on Dijkstra's algorithm for finding the shortest path on a weighted graph.

[4]Also known as graph search, refers to the process of visiting each vertex in a graph.

paths between obstacles in the plane; and the Stanford Research Institute Problem Solver (STRIPS) [10,12], an automated planner who generated plans of actions that Shakey had to perform within a mission. Shakey had a list of pre-set actions inside his planner. These actions ranged from travelling from one location to another, turning the light switches on and off, opening and closing the doors. The robot photographed its surroundings and then, through STRIPS, planned a route to the objective position that avoided obstacles by carrying out the necessary actions. The solver's task was to find a composition of operations that transforms a given initial world model into one that satisfies an objective condition. Today, $A^\star$ search algorithm is used in many applications including language analysis, route calculation and interactive computer games; while the STRIPS laid the foundation for today's AI planners [10].

From 1961 to 1981 a similar project at Stanford University, nicknamed Stanford *"Cart"*, see Fig. 1.4b, caught on. It began as a research vehicle for remote Moon missions [13] during the space race against the Soviet Union that, in the same period, was developing the Lunokhod 1 Moon rover.[5] After President John F. Kennedy's announcement on September 12 1962, of the U.S. human-crewed mission to the Moon, the project of the unmanned lunar rover was put off and turned into a test bench for autonomous vehicle research. In particular, from 1973 to 1981, Dr. Hans Moravec led the Cart project at the Stanford University's AI Lab, exploring the problems of navigation and obstacle avoidance using a stereo vision system based on a single TV camera that sent images to an external mainframe for processing. The extraction of the characteristics and the correlation between images allowed the reconstruction of a 3D model of the scene, which was used to plan an obstacle-free route to destination [13,14].

In the second half of the '70s, at the Jet Propulsion Laboratory (JPL), a very ambitious project was underway destined to be an autonomous system for planetary exploration: the *Mars ROVER*. Perception systems, based on cameras and range finders, and decision-making were the central parts of the project. Furthermore, path-finding was obtained using an algorithm $PATH^\star$, similar to $A^\star$ [15].

The Shakey and Cart programs showed some weaknesses. The systems were very slow, taking several dozens of minutes to travel 1 meter. This was caused by technological limitations related to environmental perception systems, which did not reliably see simple polygonal objects making visual navigation fragile in certain conditions, in particular when there was not sufficiently high contrast; and to the available computing power, with consequent excessively long times required for tests. In addition, the minimal nature of robot hardware has precluded economic solutions for most basic functions as well [16].

Between the 1960s and 1970s, significant advances in the technological field led to the development of microprocessors and integrated circuits, to higher computational power,

[5]Lunokhod 1 was the first successful rover to explore another world. It arrived on the Moon on November 17, 1970, on the Luna 17 lander. Led by remote control operators in the Soviet Union, it travelled more than 10 kilometres. Lunokhod 1 held the durability record for space rovers for more than 30 years until a new record was set by the National Aeronautics and Space Administration (NASA) Opportunity Mars rover during the Mars Exploration Rover (MER) mission in 2004.

to new communication techniques and to the development of new materials. Additionally, satellite tracking has become routine since the Global Positioning System (GPS) was declassified in 1983 and it is possible to realistically predict a series of real-world applications, in addition to those supporting scientific research, such as intervention robots that operate in hostile or extremely dangerous environments, or day-to-day machines in highly automated factories. Similarly, advances in signal and data processing techniques have resulted in the sophisticated estimation, optimisation, tracking and machine learning techniques which, in turn, have allowed the advancement of automation.

In this context in 1981 Moravec moved at Carnegie Mellon University (CMU) where continued his work on a new mobile robotic test bench: the CMU *"Rover"*, see Fig. 1.4c, a multi-sensor three wheel omnidirectional platform which utilizes a new distributed microprocessor architecture to provide an high level closed-loop control and sensory functions [16]. CMU became a major leader in mobile robot research during the 1980s, in particular with its *Navlab* vehicle for the results obtained about perception and navigation in outdoor environments [17].

In France, the *Hilare* project, see Fig. 1.5, began at Laboratory for Analysis and Architecture of Systems (LAAS) in 1977: it was considered the first European mobile robot capable of autonomously negotiating an unknown environment. The goal of the project was to carry out comprehensive research on word perception and modelling, path planning, navigation and decision making. To implement a series of its high-level functions, it makes use of expert systems[6] and of a variety of sensors, such as optical encoders, laser range finder, ultrasonic sensors and vision cameras, that can send information to the multilevel computer and decision-making system [18].

From the data collected by these sensors, the on-board computer system was able to calculate the position information. The idea was to use redundant position sensing to obtain complementary data, thereby improving the accuracy and precision of the overall measurements. This structure allows the robot to model its universe, build one possible plan, then develop and execute it [19]. Its modularity allowed for incremental changes, which then resulted in HILARE 2 and HILARE 2bis, a trailer-pulling wheeled robot eventually built in 1992 [20]. The development of the HILARE project helped to consolidate what are today's subsystems of any mobile wheeled robot [21]:

- Locomotion;
- Human-Robot Interaction;
- Communication;
- Planning, Perception and Navigation;
- Power Source;
- Learning and Adaptation.

[6] In artificial intelligence, an expert system is a computer system that emulates the decision-making ability of a human expert. Expert systems are designed to solve complex problems by reasoning through bodies of knowledge, represented mainly as if-then rules rather than through conventional procedural code.

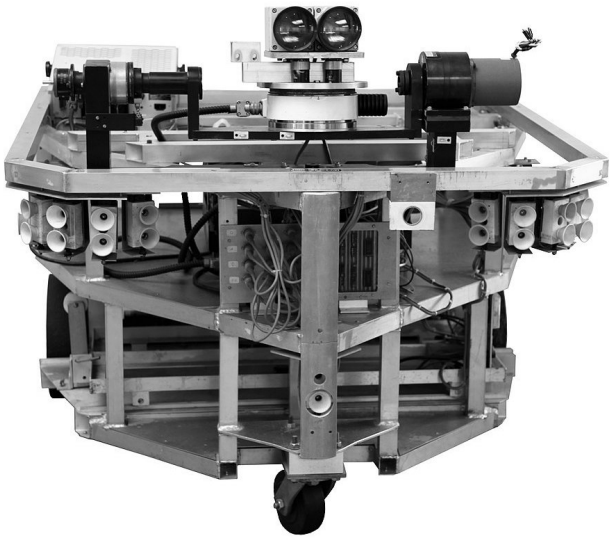


Figure 1.5: LAAS *Hilare 1* (1977).

This evolution clearly shows that mobile robots are intended for much more than moving from one point to another. However, when it comes to controlling their movement, intended as the combination of path execution and actuation, it is necessary having a control scheme. A general control scheme could be the one shown in Fig. 1.6, which is a simplified version of what is presented in [22].

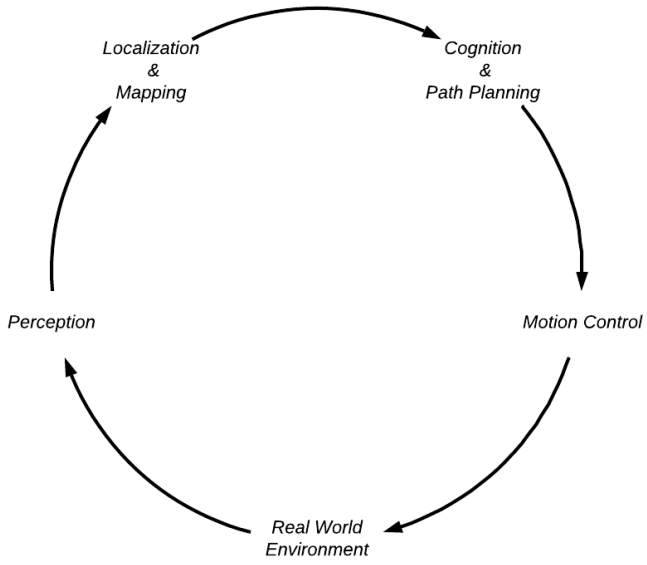


Figure 1.6: Wheeled Mobile Robot Control Scheme.

In the case of wheeled robots, research focuses on three main areas:

- Manoeuvrability;
- Controllability;
- Stability.

These are all influenced by the combination of two physical properties: wheel types and geometry. Unlike cars, which are mostly designed for a very highly standardised environment, represented by the road network, mobile robots are designed for applications in a wide variety of scenarios. For that reason, cars share similar wheel configurations because there is only one design that maximises manoeuvrability, controllability and stability on their standard environment: the asphalt road. However, no wheel configuration maximises these qualities for the variety of environments faced by mobile robots. This is why there are a large number of wheel configurations for that class of vehicles. Few WMR use the car's Ackermann configuration due to its poor handling and stability on rough terrain, except of course for those designed for paved roads applications.

Mobile robot wheels are generally of four types:

- Standard;
- Castor;
- Swedish;
- Spherical.

The two on the right offer more degrees of freedom but are more difficult to produce. The total number of wheels used directly influences the overall stability; conventionally, static stability is obtained with a minimum of three wheels and the centre of gravity of the robot lying inside the ground footprint described by the wheels contact points. However, most designs employ two, three, four or six wheels. Furthermore, taking into account the presence or absence of a steering system, it is possible to obtain the designs shown in Figure 1.7.

Regarding manoeuvrability, there are omnidirectional robots that can move in any direction at any time. Clearly, it requires the use of Swedish or spherical wheels. Other architectures can still do the same, but first, they must rotate on their vertical axis, sometimes without even changing the footprint of the terrain, as in the case of two-wheel differential drive vehicles. The less manoeuvrable models are those that use Ackermann steering characterised by a larger turning radius.

In general, there is an inversely proportional relationship between manoeuvrability and controllability. For instance, omnidirectional robots are the most difficult to control. Imagine what is necessary to drive a robot in a straight line, in the case of an Ackermann system, it is sufficient to centre and lock the wheels in position. In contrast, for differential drive robots, in which each wheel is independently driven, it is required that the wheels follow the same speed profile, a difficult task due to differences in transmissions, wheels

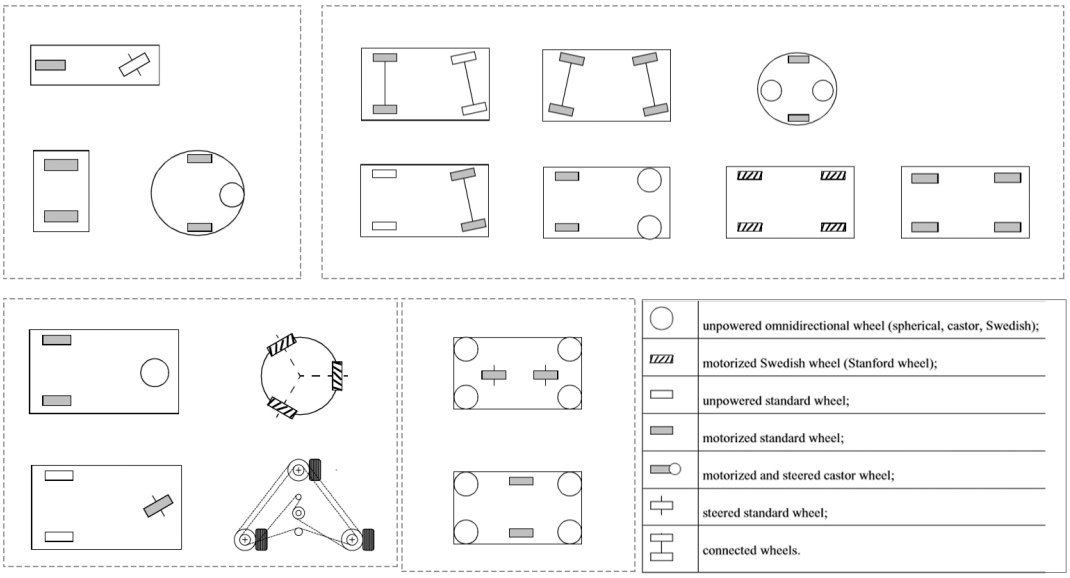


Figure 1.7: UGV possible wheel configurations.

and in the interaction with the ground. The differential transmission is used in the absence of a steering system and is often related to another type of control architecture known as skid-steering. The operation principle at the base of skid-steering is similar to the one of the tanks crawlers. When this design is used on rough terrain, manoeuvrability increases significantly, with a harmful effect on controllability. This type of architecture will be further analysed in Section 1.2.

For the purpose of this work, it is necessary to introduce another block from the control scheme illustrated in Figure 1.6: perception. The perception of the real-world environment occurs through the following actions:

- Sensing;

- Information Extraction and Interpretation;

and depends on the robot's sensors which in turn depend on its applications. The most common are [23]: Encoder, IMU, GPS, Laser Imaging Detection and Ranging (LIDAR), Radio Detection and Ranging (RADAR) and SONAR.

Since multiple of the aforementioned sensors can be mounted on a mobile robot at the same time, statistical sensor fusion techniques, such as Kalam filtering, are used to combine multiple sensor readings and obtain more accurate estimations of a single measurement. In particular, SLAM technique, which is a crucial point for autonomous navigation, is a problematic area of application of high-dimensional non-linear filtering problems and, without sensor fusion, it would not be possible. On the other hand, another localisation technique is used when autonomy is not necessary: dead reckoning is the process of determining one's position uniquely using information about its initial condition,

speed and acceleration data sensed by the use of encoders or IMU. The major problem of this technique is the error propagation, due to continuous state integrations, if the data is not corrected using observations from the outside world, for example, via LIDAR or GPS. But with well-tuned sensor fusion algorithms implementation, reasonable accuracy can be achieved.

From what has been stated, it is easy to realise that the mobile robotics design process involves the integration of many branches of engineering, making it as interdisciplinary as possible. Until the early 2000s, however, it was extremely challenging to find experts who were also willing to manage the complex software infrastructures used by robots due to its deep layered structure and the excessive time to be spent for setting up the hardware. For this reason, many robotic researchers solved some of those issues by presenting a wide variety of frameworks to manage complexity and facilitate rapid prototyping of software for experiments resulting in several robotic software such as Player [24] or CARMEN [25]. In 2007, Willow Garage funded a Stanford University project born out of the desire to create a universal robotic framework that aimed to standardise and simplify the complex robot interfaces. It took the name of ROS [26], and now has become the most trending and popular robotic framework having a vast collection of tools and libraries that are natively supported by over 80 commercial robots. ROS promotes code reuse with different hardware by providing a large number of libraries and tools for 3D visualisation, recording experiments and playing back data off-line. Regular updates enable the users to obtain, build, write, test and run ROS code; additionally, thanks to its simple architecture, ROS allows reusing code from numerous other open-source projects, such as Stage [27], Gazebo [28], OpenCV and more.

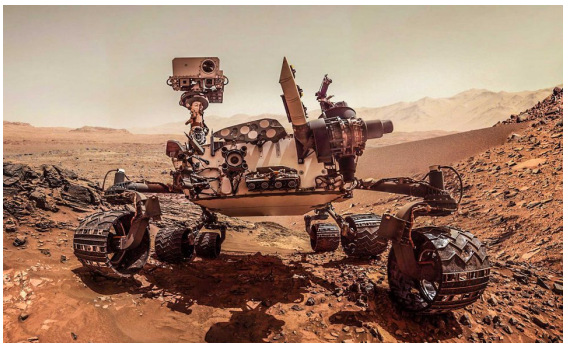
Without ROS, its counterparts, low-cost technology and advances in miniaturisation, sensors, computer processing, signal and image processing, communication techniques and materials science, robotics would not have progressed as it has done in recent decades. Today's applications of unmanned ground vehicles cover several industries; some notable examples are:

- Civilian and commercial applications [29, 30], like:
  - Manufacturing;
  - Industrial Automated Guided Vehicles (AGVs), see Fig. 1.8a;
  - Agriculture;
  - Mine exploration;
  - Outdoor exploration;
  - Volcanology;
  - Home appliances;
  - Hospital and nursing;
  - Self-driving Cars;
- Space exploration [31], such as Planetary rovers, see Fig. 1.8b;
- Emergency response [32], like:

  - Search and Rescue (SAR);
  - Fire fighting, see Fig. 1.8c;
  - Nuclear response;

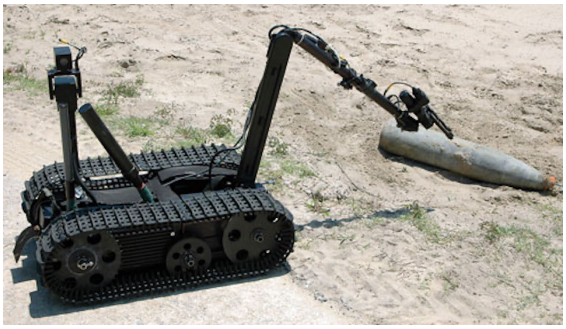
- Military applications [2, 21, 33], like:
  - Explosive Ordnance Disposal (EOD), see Fig. 1.8d;
  - Unexploded Ordnance (UXO) clearance;
  - Reconnaissance, Surveillance and Target Acquisition (RSTA);
  - Chemical, Biological, Radiological, Nuclear, Explosive (CBRNE) reconnaissance;
  - Close range field support;
  - Logistics.

(a) Comau *"Agile1500"* industrial AGV.



(b) NASA *"Curiosity"* Mars rover.

(c) SHARK Robotics *"Colossus"*.



(d) Foster-Miller *"TALON"*.

Figure 1.8: Unmanned Ground Vehicles application examples.

1.2 Skid-Steer Control Problem and Proposed Solutions

Many of today's external applications of UGVs described in the previous section 1.1, such as terrain navigation and exploration, SAR, defence, involve the use of SSMR because the

absence of the steering mechanism, like differential drive robots, makes it mechanically robust [34,35] and able to move on rough terrain with good manoeuvrability [36,37]. Like tracked vehicles, with which it shares many properties [35, 38], steering is provided controlling the relative velocities of the left and right sides drives. However, this locomotion scheme, which requires wheels slippage for steering, makes it challenging to develop an accurate kinematic and dynamic mathematical model or a robust tracking control scheme capable of describing the robot movement, due to the continuous change in the interaction forces between tires and soil. It results very difficultly for the skid-steering kinematics to predict the exact movement of the vehicle only from its control inputs. For this reason, kinematic models with pure rolling and no-slip hypotheses for non-holonomic wheeled vehicles cannot be applied in this case [38]. Besides, other disadvantages are that movement tends to be energy inefficient, difficult to control and for wheeled vehicles, tires tend to wear out more quickly [34, 39].

Even though there is a great amount of research on dynamic modelling and tracking control of differential-driven mobile robots that are under the non-holonomic constraint of zero lateral velocity, such as unicycles or car-like robots [40–42], the counterpart research on skid-steered mobile robots is less frequently reported. Some studies have presented a dynamic model for a 4W-SSMR and in [36] a non-holonomic constraint between the robot's lateral velocity and yaw rate was considered, assuming perfect knowledge of the wheel-to-ground interaction. In [34], a simple Coulomb friction model was used to describe the wheel-ground interaction, and a non-linear feedback controller was designed for trajectory tracking. A general 2D dynamic model was developed in [43, 44]. This model, unlike the others proposed, is based on the functional relationship between shear stress and shear displacement. But dynamic models for skid-steering can be too expensive for real-time control and dead–reckoning.

Authors in [45, 46] considered the kinematic relationship between driving and vehicle speeds but without taking into account the main slip effects. Wheel slip has been proven to have a crucial role in the kinematic and dynamic modelling of skid-steering mobile robots. The slip information provides a connection between the rotational speed of the wheel and the linear movement of the mobile platform. In [47] an estimate of the slip from the actual inertial readings was performed using an EKF and a kinematic model of the vehicle correlates the slip parameters with the track speeds. In [48], an experimental method was developed to determine the sliding ratios, and the sliding coefficients of the tracks were modelled as an exponential function of the radius of the traversed path.

In [49, 50] the authors followed a different approach adopting an additional trailer to study the kinematic relationship for SLAM applications. They concluded that an ideal differential-drive kinematic model could not be used for skid-steering robots.

In the work reported in [38, 39], a geometric analogy with an ideal differential-driven wheeled mobile robot was studied and validated experimentally for both tracked and skid-steered vehicles. These correspond to the position of the ideal differential drive wheels for a given terrain. The assumption is based on the fact that the Instantaneous Center of Rotations (ICRs) values of the treads depend on the dynamics, but they are located within a bounded area at moderate speeds. A group of constant kinematic parameters were derived as optimized values for the tread ICRs on the plane. These values vary

with the mobile robot velocity and with the curvature of the path. Hence, authors in [51] further experimentally described the relationship between the tread ICRs values and the curvature of the path and the speed of the vehicle with a derived approximation function.

All of these authors agree in the fact that motion control for this type of vehicles is a very complicated problem. Furthermore, most published works are based on simulated results and used complex vehicle dynamics or on-line parameters adaptation that may result too costly for real-time implementation. For this reason, the proposed work adopts a simplified bi-dimensional non-linear mathematical model inspiring at the results proposed by [37] and [52], considering constant, given the low speeds and accelerations in analysis, the local x-position of the ICR. Furthermore, the wheel-ground interaction is modelled as rolling friction for the longitudinal resistance, while for the lateral resistance is implemented an enhanced dry friction model, proposed for the first time in [53], that is able to select the correct sign of the friction force according to the direction of the actuation speed and, employing a zero-crossing detection algorithm, to distinguish between the stick and slip conditions, to evaluate the possible stopping of the element previously in motion, to maintain the stationary condition correctly and to evaluate the possible breakaway of the previously stopped element.

1.3 PID Control Related Work and Contribution

PID controllers are still widely used in motion control of mobile robots and for controlling industrial systems in general because their excellent performance in a wide range of operating conditions, their straightforward architectures and their familiarity in the control community, makes them very simple and easy to implement [61]. However, its ability to cope with some complex model characteristics such as non-linearities, external disturbances and time-varying parameters is known to be very poor. For this reason, traditional PID gains design commonly requires a compromise between system robustness and performance.

Over the years various design methods[7] have been proposed to determine the PID controller parameters. Some methods use information about the open-loop step response, e.g. the Coon-Cohen reaction curve method [62]; other methods use some knowledge of the Nyquist curve of the plant, e.g. the Ziegler-Nichols frequency response method [63] or the gain-phase margin approach [64]. The simple regulation laws used to determine the controller gains has made these methods suitable for the implementation in some commercial auto-tuner algorithms. However, since the small amount of information on the dynamic behaviour of the system used, these methods often do not provide good tuning. For example, Ziegler Nichols method may give high overshoots, highly oscillatory and longer settling time for a high order system and Coon-Cohen method is only valid for systems having S-shaped plant step response [65]. However, they have been widely used as heuristic methods for determining PID controller parameters. Evolutions of these methods have been proposed to improve these techniques as the Åström-Hägglund phase

[7] With the term *design method* it is meant the determination of the three parameters of the controller which are the proportional, integral, and derivative gains.

margin method [66] and the refined Ziegler-Nichols method [67]; however, in some circumstances, these methods do not produce satisfactory closed-loop responses for example when a system has a large time delay, resulting in a very oscillatory closed-loop response. The major limits of implementation of these methods are the need of having the transfer function of the plant. To overcome these limitations, various methods have been developed to obtain optimal PID parameters ranging from conventional pole placement [61] to the implementation of a variety of AI techniques such as Artificial Bee Colony (ABC) optimization algorithm [68], Simulated Annealing (SA), Population Based Incremental Learning (PBIL), Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and DE algorithms [69].

Recently, GA has been widely adopted in the search for optimal PID parameters thanks to its ease of implementation and its high adaptability as demonstrated in [70]. In [71], the authors applied a GA to derive the optimal gains of a PID controller used in a trajectory taking application, demonstrating the improved performance in terms of control precision and speed of convergence compared to the conventional tuning method. Although the performance effectiveness of GA in looking for a globally optimal solution, some studies have highlighted shortcomings in the algorithm's performances such as poor premature convergence, loss of the best solution found and no absolute assurance that the algorithm will find a global optimum [72].

These limitations are overcome by the DE algorithm, which was designed to meet the requirements of the practical minimization technique [73], therefore the algorithm has attracted great attention within the scientific community. In [74], authors studied the performance of DE, GA and PSO in optimizing the PID controller for in-position control of a manipulator concluding that DE is generally more robust than other methods and not suffers from local minima problems such as GA. In [75] authors applied the DE algorithm in setting up the PID controller for the trajectory tracking control of quad-rotors. The simulation results show that the controlled system has a satisfactory response and that it is able to deal with the aircraft coupled dynamics, thus demonstrating the effectiveness of the proposed optimization method.

The method here illustrated follows the multi-loop structure. The inner loop, dedicated to speed control, it is divided into two branches for controlling the linear and the angular velocity of the rover; furthermore, his work is aided by a feed-forward branch for improving the velocity tracking. The reference variables are the velocity errors. The outer loop is dedicated to the position control of the WMR. It is divided into three branches in which the control variables are the position errors. Given the number of parameters, an optimization algorithm is used to speed up the tuning process. In essence, the selected algorithm is DE due to its implementation simplicity, also ensuring excellent results in a variety of problems. In particular, an improved version based on the work of [76, 77] called jDE was adopted, characterized by self-adaptive control parameters which give flexibility to the algorithm: the mutation control parameters evolve together with the candidate individuals obtaining a flexible DE architecture that does not require an additional tuning of its parameters. Gains are chosen on one hand to obtain adequate stability and performance and on the other hand, to optimize closed-loop performances in terms of step response and waypoint tracking. In particular, a range in which the PID gains give

a stable and acceptable response is first identified by trial and error. Then the jDE algorithm is applied to the user-specified range to find the optimal gain values. Adopting this approach, since the PID parameters searching area is restricted due to the control engineer sense, the convergence time of the jDE algorithm and consequently the time to get the optimized set of PID gains is drastically shortened.

1.4 Sliding Mode Control Related Work and Contribution

During the past decades, some investigation focused on non-linear controllers design for implementing the stability of those systems for which an approximation to linear model or the design of a conventional linear controller may results too complicated or inaccurate. One of the fields which is generating greater interest in this research branch is that of Variable Structure Control Systems (VSCSs), in particular, the SMC technique due to its robustness to system uncertainties [78] making it an effective candidate for mobile robot control applications.

In [79, 80], a sliding mode control strategy shows, from the simulations results, the robustness of the controller against the system uncertainties. However, since the control was only applied at torque level, it ensures only the reverence velocity tracking generated by a separated kinematic controller. In [81], the authors designed a kinematic sliding mode controller for a WMR following a local coordinate transformation. The control law proposed, based on reaching law approach, demonstrates a good tracking performance, making the initial posture error converge to zero as the time approaches infinity, and its superiority over the classical PID. However, the controller was described in an implicit form which is inconvenient for practical implementation on a real platform. The authors in [82] used a feedback linearisation approach for applying the sliding mode controller to a mobile robot. Even if excellent results have been obtained for both the tracking and the regulation tasks, from a piratical point of view that solutions would be difficult to implement due to the authors' choice of considering the active force generated by the rover drivetrain as one of the system states. That choice, necessary to satisfy the feedback linearisation, would inevitably increase the system complexity and computational cost. A different approach has been taken in [83] where the SMC gain parameters have been tuned employing a genetic algorithm to follow the desired trajectory satisfactorily, obtaining a high tracking efficiency even in the presence of disturbances.

Yang and Kim proposed in [84] a robust sliding mode control strategy for a two-wheeled mobile robot trajectory tracking purposes. Although the proposed control law asymptotically stabilised the mobile platform to the desired trajectory, it was affected by chattering problems due to hardware limitation, not exhibiting the expected tracking performances. The chattering drawback has been analysed in detail in [85], where authors experimentally compared four different control laws based on the reaching-law method, and in [86], where an exponential sliding mode is proposed for reducing the chattering phenomenon.

In this context, the sliding mode control algorithm introduced in this thesis proposes a new switching function design, similar to the one suggested in [87] but introducing the dependency of the x-error derivative in its longitudinal term for achieving a better tracking

of the linear velocity, and an enhanced exponential reaching law based on a quasi-sliding mode approach, for forcing the state to reach the switching manifolds faster when the switching function is large reducing the reaching phase and at the same time reducing the chattering effect tanks to the substitution of the signum function with a continuous relay function. Furthermore, the jDE algorithm is applied to a user-specified range for tuning the controller gains optimally obtaining a high tracking efficiency.

1.5 Trajectory Evaluation Metrics

Since Robotic systems in the last decades are gradually finding more and more applications, ranging from everyday life to industry applications, the evaluation of their performances is gaining significant attention also by companies interested in developing robotics applications for the general public. In this context plenty of research activities are pointing to define evaluation frameworks for the assessment of the robot performances ranging from the comparison of different motion planners based on statistical measurements of the performed trajectories [54] to the development of more complex laws that make possible the so popular robotic competitions like the *DARPA Grand Challenge* [55] or the *Robocup@Home* [56] because able to compare at the same time the performances of different robotic assets deployed in the same environmental conditions.

Considering now our problem, that is finding a method to compare the goodness of different trajectory tracking controllers, it is clear that a similar framework can be adopted. Taking inspiration from the work done by [57], in which the autonomous navigation performances of the mobile robot, considered as a *black-box*, are assessed trough several metrics based on external observations. The purpose of the proposed approach is to adapt some of these metrics for comparing the simulated/real robot trajectory defined by the controllers with the reference one.

Because of the efficiency and the accuracy of a control system are generally measured exploiting different quantities that for some reasons provide complementary pieces of information, the key point in this work consists in comparing the proposed metrics and later combining them into a cost function to obtain a general evaluation for a specific controller architecture. Being inspired by the standard closed-loop specifications, i.e. the overshoot could be linked to the Hausdorff distance and the settling time to the total bending energy, a list of performance indices focussing on trajectory tracking tasks are formally expressed as follows:

- **Distance ratio:** The distance ratio d_{ratio} metrics directly compares the travelled length of the robot and the reference trajectory. Considering d_T as a generic euclidean distance defined as the sum of each segment length composing the trajectory, the distance ratio is defined as:

$$d_{ratio} = \frac{d_{T_{rob}}}{d_{T_{ref}}} \tag{1.1}$$

- **Hausdorff distance:** The Hausdorff distance d_{haus}, broadly employed in image matching and handwriting recognition applications, gives an indication of the maximum separation between two trajectories $T_1(t)$, $T_2(t)$ [58], see fig.1.9. It is evalu-

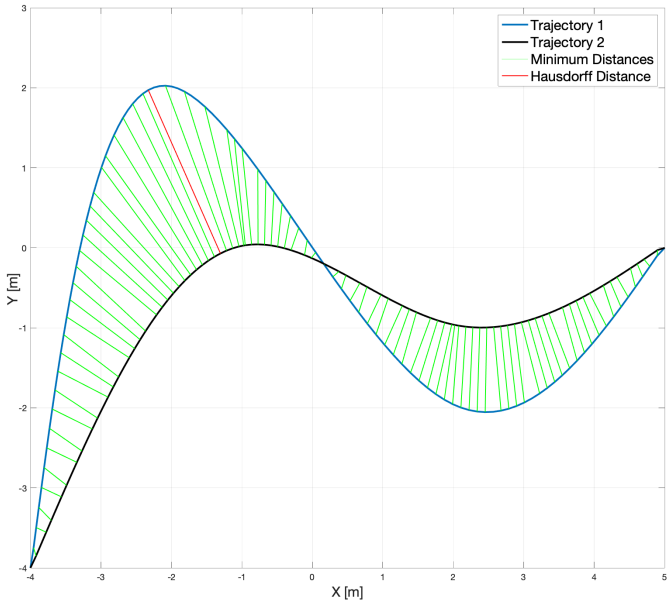


Figure 1.9: Hausdorff distances between two trajectories.

ated as the maximum among the minimum distances from the points in one curve to the points in the other curve. Given two trajectories $T_{ref}(t)$ and $T_{rob}(t)$, having respectively n and m elements, the Hausdorff distance between the two curves can be calculated as:

$$d_{haus} = \max\left\{\min\left\{d_{T_{ref}(t_i)} - d_{T_{rob}(t_j)}\right\}\right\} \quad \text{where, } i = 1,\ldots,n,\ j = 1,\ldots,m \tag{1.2}$$

- **Root Mean Square (RMS) of minimum distances:** The overall separation between two trajectories $T_1(t)$, $T_2(t)$ can be calculated as the RMS of the minimum distances from the points in one curve to the points in another other curve so, given the trajectories $T_{ref}(t)$ and $T_{rob}(t)$, the minimum distance between any two points of each trajectory, and consequently its RMS, can be calculated as:

$$d_{rms} = RMS\left\{\min\left\{d_{T_{ref}(t_i)} - d_{T_{rob}(t_j)}\right\}\right\} \quad \text{where, } i = 1,\ldots,n,\ j = 1,\ldots,m \tag{1.3}$$

- **Total bending energy:** The total bending energy e_{bend} can be understood as the energy necessary to bend a rod to the desired shape so, applied to a trajectory, could represents a measure of efficiency indicating how *'curvy'* the overall trajectory is [59, 60]. Defining c_i as the local curvature along the discrete trajectory of n elements each long d_{T_i} , the total bending energy is defined as:

$$e_{bend} = \sum_{i=0}^{n} c_i^2 d_{T_i} \tag{1.4}$$

The analysed metrics are selected to assess the trajectory efficiency and the vehicle capability to follow the desired trajectory. These include absolute parameters which describe the characteristics of the trajectory and will be later applied to compare the measured robot simulated/experimental trajectory with the theoretical one. Furthermore, the cost function obtained combining these indices will be used in the optimisation process of the controllers' parameters in which the selected algorithm tries to find, generation after generation, the controllers' gains that minimise an objective function f_{opt} until a stop criterion is satisfied or the final generation is reached. A more detailed description of the operation of the optimisation algorithm is given in Section 3.4 and Appendix A.

Chapter 2

Mathematical Model

A complete mobile robot mathematical model is defined by three sets of differential equations that govern its dynamics. These describe the forces and moments acting on it and its orientation towards a reference frame. In this chapter, a 2D-simplified non-linear model is presented along with its linearisation and state-space representation. As usual, the model has been decoupled into an independent kinematic and dynamic formulation. In order to obtain the kinematic one, the non-holonomic constraint of the vehicle will be considered and derived using the chassis geometrical properties and a transformation between the coordinate systems. The dynamical properties, on the other hand, will be derived using the Euler-Lagrange equations. Some mathematical tools necessary to fully understand the equations proposed are introduced first. In essence, these include the reference frames and the Euler angles used. Finally, the mobile platform adopted for simulations and experimental tests is introduced, in particular, the *Clearpath Husky* mobile robot which represents the test bench for the controllers proposed in this thesis.

2.1 Mathematical Tools Overview

2.1.1 Reference Frames

A reference frame consists of an abstract coordinate system and the set of physical reference points that uniquely represent the position and orientation of a dynamic system; in this case, the mobile robot. Due to the great variety of existing reference frames, an introduction to those used in this work is necessary for the sake of clarity.

Local Frame

With Local Frame, we refer to those generic body axes that have origin in the mobile robot Centre of Gravity (CoG). They are defined as follows: x_l and z_l lie in the robot plane of symmetry, with x_l generally parallel to the car body reference line and directed towards the rover nose, z_l is directed from the lower to the upper surface of the shell; the y_l axis is selected so that the coordinate frame is right-handed, see 2.1. The local frame is fixed with respect to the mobile robot; therefore, it is an inertial reference frame: the

moments of inertia of the rover calculated within this frame do not change during the motion.

Global Frame

With Global Frame, we refer to those generic Cartesian axes fixed in a determined origin. They are defined as follows: X_g and Y_g lie on a plane parallel to the ground, and the Z_g axis is selected so that the coordinate frame is right-handed, see 2.1. The global frame is fixed with respect to the external environment; therefore, it is a non-inertial reference frame: the moments of inertia of the rover calculated within this frame change during the motion.

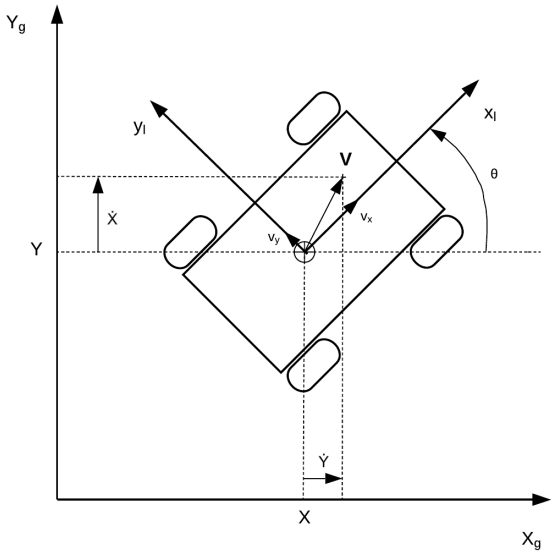


Figure 2.1: Global and local reference frames.

2.1.2 Euler Angles

The Euler angles are a mathematical tool used to define the orientation of a reference frame with respect to another. Indicated as $\Phi,\ \Theta,\ \Psi$, they represent three independent angular rotations necessary to align two reference frames. For example, considering the two coordinate systems $\mathscr{S}_1\,(X_1,\ Y_1,\ Z_1)$ and $\mathscr{S}_2\,(X_2,\ Y_2,\ Z_2)$ represented in Figure 2.2, the rotations[1] $\Phi,\ \Theta,\ \Psi$ aligns $\mathscr{S}_2$ to $\mathscr{S}_1$.

According to this description, Euler angles can also be used to define the transformation of the components of a generic vector between two reference frames through the elementary rotation matrix. Assuming that $[F_{X_1}, F_{Y_1}, F_{Z_1}]^T$ is a generic vector of the

[1]The rotations are performed in sequence and the order of the rotations is fixed.

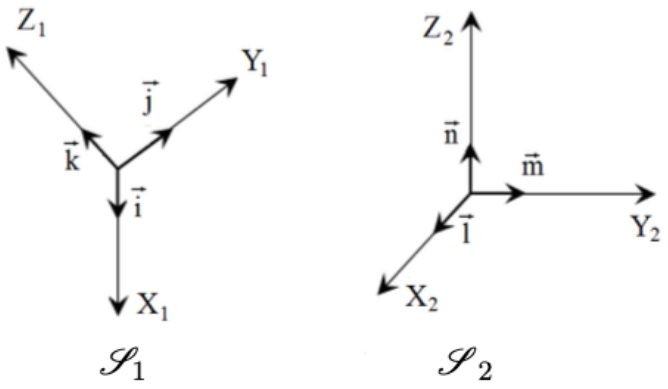


Figure 2.2: Generic reference frames.

coordinate system $\mathscr{S}_1$, the relationship with the corresponding vector $[F_{X_2}, F_{Y_2}, F_{Z_2}]^T$ in the final coordinate system $\mathscr{S}_2$ is:

$$\begin{bmatrix} F_{X_2} \\ F_{Y_2} \\ F_{Z_2} \end{bmatrix} = \left[\boldsymbol{R_{21}}\right] \begin{bmatrix} F_{X_1} \\ F_{Y_1} \\ F_{Z_1} \end{bmatrix} \tag{2.1}$$

where $\boldsymbol{R_{21}} = [\boldsymbol{\Phi}]\,[\boldsymbol{\Theta}]\,[\boldsymbol{\Psi}]$ represent the matrix of complete transformation and $[\boldsymbol{\Phi}]$, $[\boldsymbol{\Theta}]$, $[\boldsymbol{\Psi}]$ are the elementary rotation matrices defined as:

$$[\boldsymbol{\Phi}] = \begin{bmatrix} cos\Phi & -sin\Phi & 0 \\ sin\Phi & cos\Phi & 0 \\ 0 & 0 & 1 \end{bmatrix},\ [\boldsymbol{\Theta}] = \begin{bmatrix} cos\Theta & 0 & sin\Theta \\ 0 & 1 & 0 \\ -sin\Theta & 0 & cos\Theta \end{bmatrix},\ [\boldsymbol{\Psi}] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\Psi & -sin\Psi \\ 0 & sin\Psi & cos\Psi \end{bmatrix}$$

Since the elementary matrices are orthogonal, also $\boldsymbol{R_{21}}$ is orthogonal, so $\boldsymbol{R_{21}^{-1}} = \boldsymbol{R_{12}} = \boldsymbol{R_{21}^T}$. This allows to define the inverse transformation as:

$$\begin{bmatrix} F_{X_1} \\ F_{Y_1} \\ F_{Z_1} \end{bmatrix} = \left[\boldsymbol{R_{21}^T}\right] \begin{bmatrix} F_{X_2} \\ F_{Y_2} \\ F_{Z_2} \end{bmatrix} \tag{2.2}$$

Global-Local Transformation

Is now illustrated the Euler angle used in the present work representing the rotation necessary to transform the components of a generic vector between the reference frames described in Section 2.1.1 or to align the two reference frames.

In order to project the local frame $\mathscr{S}_l\,(x,\ y,\ z)$ to the global frame $\mathscr{S}_g\,(X,\ Y,\ Z)$, reported in Fig. 2.1, it is necessary one rotation of f magnitude ϑ about z_l so the corresponding Euler angle is:

$$\Phi = \vartheta \tag{2.3}$$

and the local-global rotation matrix $\boldsymbol{R_{lg}}$ is:

$$\boldsymbol{R_{lg}} = \begin{bmatrix} cos\vartheta & -sin\vartheta & 0 \\ sin\vartheta & cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

2.2 Non-linear Mathematical Model

The non-linear mobile robot mathematical model is here presented for trajectory tracking application and built starting from the dynamic and kinematic characteristics of the mobile platform.

According to the conclusions derived in section 1.2, without loss of generality, the following assumptions are considered:

1. The centre of mass of the robot lies on the car body reference line and is fixed during motion;
2. The vehicle is rigid and moves on a horizontal plane.
3. The four wheels are always in contact with the ground surface;
4. There is point contact between the wheel and the ground;
5. Each side's two wheels rotate at the same speed;
6. The vehicle speed is below 1 m/s;
7. The longitudinal wheel slippage is neglected;
8. The tire lateral force is function of its vertical load.

2.2.1 Kinematic Model

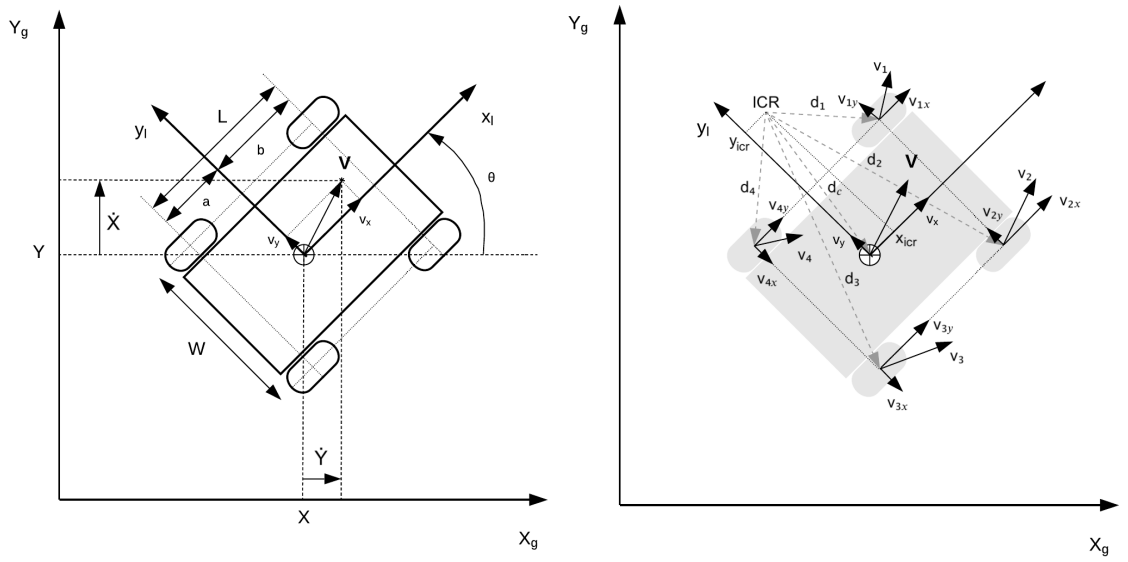
Considering the vehicle allowed to move only on a two dimensional, see figure 2.3a, is possible to define the state vector describing the generalized coordinates of the robot[2] and the body velocity vector respectively as $\boldsymbol{q} = [X, Y, \vartheta]$ and $\boldsymbol{V} = [v_x,\ v_y,\ \omega_z]^T$, with $v_x, v_y,\ \omega_z$ determining respectively the longitudinal, lateral and angular velocity of the vehicle.

Accordingly if we consider the body velocity vector it is possible to derive, in the case of planar motion, the global velocity vector[3] $\dot{\boldsymbol{q}} = \left[\dot{X},\ \dot{Y},\ \dot{\vartheta}\right]^T$ by means of the rotation matrix defined in eq. 2.4:

$$\dot{\boldsymbol{q}} = \begin{bmatrix} cos\vartheta & -sin\vartheta & 0 \\ sin\vartheta & cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{V} = R_{lg}\boldsymbol{V} \tag{2.5}$$

[2]I.e., the CoG position, X and Y , and the orientation ϑ of the local coordinate frame with respect to the inertial frame.

[3]In case of planar motion the relationship $\dot{\vartheta} = \omega_z$ is verified.



(a) Free-body kinematics. (b) Wheel velocities relationships.

Figure 2.3: Kinematics of SSMR.

Since equation 2.5 describes free-body kinematics and does not impose any restriction on the in-plane movement of the SSMR, it is necessary to analyse the relationship between the speeds of the wheels and the local velocities.

Neglecting, for the sake of simplicity, the thickness of the wheel and assuming it is in contact with the plane at point P_i, see fig. 2.4, we suppose that the wheels rotates with an angular velocity $\omega_i(t)$, where $i = 1,2,\ldots,4$ represents the i-th wheel. For skid-steer robots, in contrast to most wheeled vehicles, the lateral velocity v_{iy}, is generally non-zero; because the mechanical structure of the SSMR makes lateral skidding necessary for changing the orientation of the vehicle. Therefore the wheels are tangent to the path only when $\omega = 0$, i.e. when the robot moves on a straight line.

Whereas we neglect any longitudinal slip between the wheels and the surface, according to this assumption based on work done in [88], the following relation can be developed:

$$v_{ix} = r_i \omega_i \tag{2.6}$$

where v_{ix} is the longitudinal component of the total velocity vector v_i of the i-th wheel expressed in the local frame and r_i denotes the so-called *effective rolling radius*[4] of that wheel.

Considering the whole mobile platform, with reference to the figure 2.3b, it is possible to express the following relationship between the position of the ICR and the local velocities vectors components with respect to the local frame:

$$\frac{\|\boldsymbol{v_i}\|}{\|\boldsymbol{d_i}\|} = \frac{\|\boldsymbol{V}\|}{\|\boldsymbol{d_c}\|} = |\omega_z| \tag{2.7}$$

[4]It represents ratio of the linear velocity of the wheel centre to the angular velocity of the wheel.

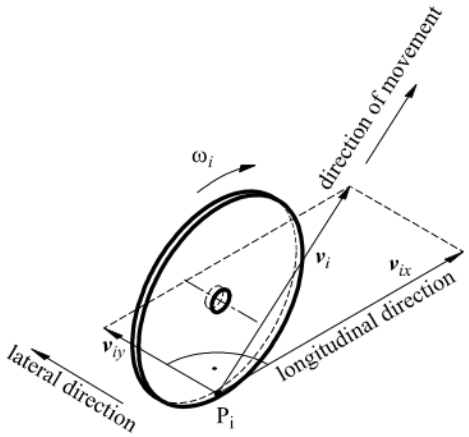


Figure 2.4: Velocities of one wheel.

where $\|\cdot\|$ represent the euclidean norm.

Defining the local frame ICR coordinates as:

$$ICR\,(x_{ICR}, y_{ICR}) = (-d_{xc}, -d_{yc}) \tag{2.8}$$

it is possible to rewrite 2.7 as:

$$\frac{v_x}{y_{ICR}} = -\frac{v_y}{x_{ICR}} = \omega_z \tag{2.9}$$

furthermore, the coordinates between the ICR and the i-th wheel satisfy the following relationships:

$$\begin{aligned} d_{1y} &= d_{4y} = d_{cy} + \frac{W}{2} \\ d_{2y} &= d_{3y} = d_{cy} - \frac{W}{2} \\ d_{1x} &= d_{2x} = d_{cx} + b \\ d_{3x} &= d_{4x} = d_{cx} - a \end{aligned} \tag{2.10}$$

where $a,\ b,\ W$ are geometric constants of the mobile robot.

The relationships between wheel velocities can be obtained from eqs. 2.7 and 2.10:

$$\begin{aligned} v_L &= v_{1x} = v_{4x} \\ v_R &= v_{2x} = v_{3x} \\ v_F &= v_{1y} = v_{2y} \\ v_B &= v_{3y} = v_{4y} \end{aligned} \tag{2.11}$$

where:

- v_L and v_R represent the longitudinal coordinates of the left and right wheels velocities;

- v_F and v_B represent the lateral coordinates of the front and rear wheels velocities.

Combining eqs. (2.7)–(2.11) it is possible to obtain the relationship between the wheel and the robot velocities:

$$\begin{bmatrix} v_L \\ v_R \\ v_F \\ v_B \end{bmatrix} = \begin{bmatrix} 1 & -\frac{W}{2} \\ 1 & \frac{W}{2} \\ 0 & -x_{ICR} + b \\ 0 & -x_{ICR} - a \end{bmatrix} \begin{bmatrix} v_x \\ \omega_z \end{bmatrix} \tag{2.12}$$

moreover, considering for each wheel the same effective radius $r_i = r$, from eqs. 2.6 and 2.11, we can write:

$$\boldsymbol{\omega_w} = \begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix} = \frac{1}{r} \begin{bmatrix} v_L \\ v_R \end{bmatrix} \tag{2.13}$$

where ω_l and ω_r are respectively the angular velocities of the left and right wheels.

Finally, it is possible to derive a relation between the angular wheel velocities and the velocities of the robot; starting from eqs. 2.12 and 2.13 we can introduce a new control input at kinematic level $\boldsymbol{\eta} \in \mathbb{R}^2$:

$$\boldsymbol{\eta} = \begin{bmatrix} v_x \\ \omega_z \end{bmatrix} = r \begin{bmatrix} \frac{\omega_L + \omega_R}{2} \\ \\ \frac{\omega_R - \omega_L}{W} \end{bmatrix} \tag{2.14}$$

In order to complete the kinematic model of the SSMR, a velocity constraint must be introduced:

$$v_y + x_{ICR}\omega_z = 0 \tag{2.15}$$

The last equation it was introduced for the first time in [37]: it is a not integrable equation that imposes the null motion along the local y-direction. In other words, it represents a non-holonomic constraint which bounds the x-position of the ICR in the robot wheelbase avoiding severe loss of stability due to elevate skidding.

Equation 2.15 can be rewritten in the Pfaffian form introducing the constraint vector $\boldsymbol{A} \in \mathbb{R}^3$ and using eq. 2.5:

$$\begin{bmatrix} -sin\vartheta & cos\vartheta & x_{ICR} \end{bmatrix} \begin{bmatrix} \dot{X} & \dot{Y} & \dot{\vartheta} \end{bmatrix}^T = \boldsymbol{A}(\boldsymbol{q})\,\dot{\boldsymbol{q}} = 0 \tag{2.16}$$

Since the generalized velocity vector $\dot{\boldsymbol{q}}$ is always in the null space of $\boldsymbol{A}$, introducing a general coordinates transformation matrix $\boldsymbol{S} \in \mathbb{R}^{3x2}$ we can write:

$$\dot{\boldsymbol{q}} = \boldsymbol{S}(\boldsymbol{q})\,\boldsymbol{\eta} \tag{2.17}$$

where:

$$\boldsymbol{S}^T(\boldsymbol{q})\,\boldsymbol{A}^T(\boldsymbol{q}) = 0 \tag{2.18}$$

and:

$$\boldsymbol{S}(\boldsymbol{q}) = \begin{bmatrix} cos\vartheta & x_{ICR}sin\vartheta \\ sin\vartheta & -x_{ICR}cos\vartheta \\ 0 & 1 \end{bmatrix} \tag{2.19}$$

Since $dim\,(\boldsymbol{\eta}) = 2 < dim\,(\boldsymbol{q}) = 3$, eq. 2.17 describes the kinematics of an under-actuated robot.

Considering the angular velocity ω as control input has an advantage over the choice to use v_y like in [37] because adopting this choice it is possible to have the SSMR kinematic model without the knowledge of the y component of the ICR in the local frame which can be only obtained experimentally.

2.2.2 Dynamic Model

As concluded in [37] the dynamic properties of SSMR play a significant role because of the interaction between wheels and ground: reactive friction forces are usually much higher than forces resulting from inertia when the robot is changing its orientation. The dynamic properties of SSMR, for this reason, influences its motion much more than for vehicles under the pure rolling assumption. Therefore is here presented a simplified dynamic model of the SSMR useful for control purposes of the real mobile platform.

The SSMR dynamic equation can be obtained from Euler-Lagrange principle with Lagrange multipliers to include the non-holonomic constraint 2.15. The Euler-Lagrange equation is:

$$\boldsymbol{\Gamma} = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} \tag{2.20}$$

where $\boldsymbol{\Gamma}$ denotes the active torques and forces vector without considering any external force and $\mathcal{L} = E - U$ is the Lagrangian of the system defined as the sum of kinetic and potential energy. Due to assumption 2 and 3, the potential energy U is equal to 0, so in our case the Lagrangian coincides with the kinetic energy of the robot:

$$\mathcal{L}\,(\dot{\boldsymbol{q}}, \boldsymbol{q}) = E\,(\dot{\boldsymbol{q}}, \boldsymbol{q}) \tag{2.21}$$

Neglecting the kinetic energy of the wheels for sake of simplicity, the following equation describing the kinetic energy of the vehicle can be developed:

$$E = \frac{1}{2} m \boldsymbol{v}^T \boldsymbol{v} + \frac{1}{2} I \omega_z^2 \tag{2.22}$$

where I is the moment of inertia of the robot about its CoG. Since:

$$\boldsymbol{v}^T \boldsymbol{v} = v_x^2 + v_y^2 = \dot{X}^2 + \dot{Y}^2$$

eq. 2.22 can be rewritten in:

$$E = \frac{1}{2} m \left(\dot{X}^2 + \dot{Y}^2\right) + \frac{1}{2} I \dot{\vartheta}^2 \tag{2.23}$$

The inertial forces can be obtained from the partial derivative of kinetic energy and its time-derivative as:

$$\frac{d}{dt}\left(\frac{\partial E}{\partial \dot{\boldsymbol{q}}}\right) = \begin{bmatrix} m\ddot{X} \\ m\ddot{Y} \\ I\ddot{\vartheta} \end{bmatrix} = \boldsymbol{M}\ddot{\boldsymbol{q}} \tag{2.24}$$

where $\boldsymbol{M} \in \mathbb{R}^{3x3}$ denotes the constant, diagonal, positive definite inertia matrix:

$$\boldsymbol{M}(\boldsymbol{q}) = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix} \tag{2.25}$$

Before deriving the resistive forces, with reference to fig. 2.5, we first describe the wheel-ground interaction. The active and reactive forces, respectively F_{xi} and N_i are functions of the wheel torque and the gravity load, in addition F_{xi} is linearly dependent on the wheel control input τ_i:

$$F_{xi} = \frac{\tau_i}{r} \tag{2.26}$$

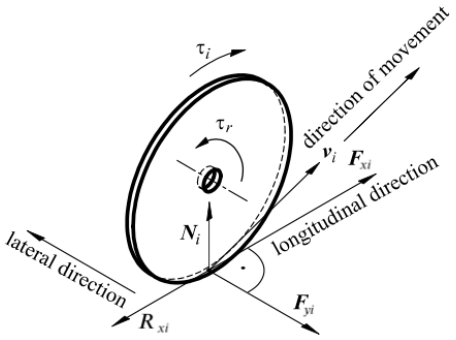


Figure 2.5: Forces acting on one wheel.

Assuming that the vertical loads N_i acts from the surface to the wheel [88], neglecting additional dynamic properties like the kinetic energy of the wheels, see fig. 2.6, the following equilibrium equation can be obtained:

$$\begin{aligned} N_4 a &= N_1 b \\ N_2 b &= N_3 a \\ \sum_{i=1}^{4} N_i &= mg \end{aligned} \tag{2.27}$$

where m and g represents respectively the vehicle mass, considered distributed homogeneously, and the gravity acceleration. Since the longitudinal mid-line symmetry, it is possible to state:

$$\begin{aligned} N_1 = N_2 &= \frac{a}{2(a+b)} mg \\ N_3 = N_4 &= \frac{b}{2(a+b)} mg \end{aligned} \tag{2.28}$$

Assuming R_{xi} derive from the rolling resistant moment τ_{ri}, this vector can be treated as rolling resistance; while the vector F_{yi}, that denotes the lateral reactive force, based

on [37] can be regarded as friction force. Since modelling the friction is quite complicated due to its highly non-linearities and its many variables dependencies, in most it is used a simplified approximation describing the friction F_f as a superposition of Coulomb and viscous friction:

$$F_f(v) = \mu_c N sgn(v) + \mu_v v \tag{2.29}$$

where N is the force perpendicular to the surface, μ_c and μ_v denote respectively the coefficients of Coulomb and viscous friction. Since the mobile robot velocity v is relatively low, especially during lateral slippage, it is possible to neglect the viscous term $\mu_v v$ due to the relation $\mu_c N \gg \mu_v v$, simplifying the model.

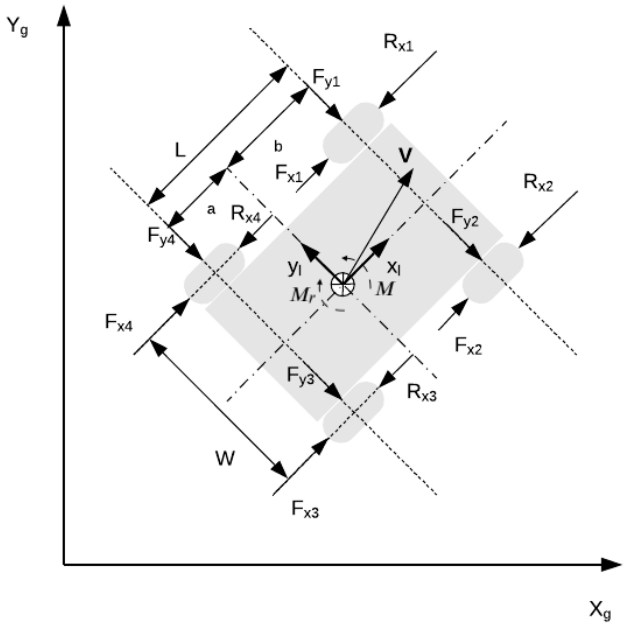


Figure 2.6: Active and resistive forces of the vehicle.

Based on the preview considerations the longitudinal rolling resistance and the lateral friction force can be written as:

$$\begin{aligned} R_{xi} &= u_{xi} mg \; sgn(v_{xi}) \\ F_{yi} &= \mu_{cyi} mg \; sgn(v_{yi}) \end{aligned} \tag{2.30}$$

where u_x is the longitudinal rolling coefficient and μ_{cy} is the lateral friction coefficient.

Consequently, the energy dissipating forces namely, according to fig. 2.6, the resulting rolling resistance R_x, the resultant reactive force F_y and moment M_r around the CoG

expressed in the local frame:

$$\begin{aligned} R_x\left(\dot{\boldsymbol{q}}\right) &= \sum_{i=1}^{4} R_{xi}\left(v_{xi}\right) \\ F_y\left(\dot{\boldsymbol{q}}\right) &= \sum_{i=1}^{4} F_{yi}\left(v_{yi}\right) \\ M_r\left(\dot{\boldsymbol{q}}\right) &= b\sum_{i=1,2} F_{yi}\left(v_{yi}\right) - a\sum_{i=3,4} F_{yi}\left(v_{yi}\right) + \frac{W}{2}\left(\sum_{i=2,3} R_{xi}\left(v_{xi}\right) - \sum_{i=1,4} R_{xi}\left(v_{xi}\right)\right) \end{aligned} \tag{2.31}$$

for defining the vector of resultant resistive forces and torque $\boldsymbol{R} \in \mathbb{R}^3$:

$$\boldsymbol{R}\left(\dot{\boldsymbol{q}}\right) = \begin{bmatrix} R_x\left(\dot{\boldsymbol{q}}\right) cos\vartheta - F_y\left(\dot{\boldsymbol{q}}\right) sin\vartheta \\ R_x\left(\dot{\boldsymbol{q}}\right) sin\vartheta + F_y\left(\dot{\boldsymbol{q}}\right) cos\vartheta \\ M_r\left(\dot{\boldsymbol{q}}\right) \end{bmatrix} \tag{2.32}$$

The resultant active force F_x and torque M generated by the robot's actuators can be expressed in the local frame as:

$$\begin{aligned} F_x &= \sum_{i=1}^{4} F_{xi} \\ M &= \frac{W}{2}\left(\sum_{i=2,3} F_{xi} - \sum_{i=1,4} F_{xi}\right) \end{aligned} \tag{2.33}$$

that compose the vector of active forces $\boldsymbol{F} \in \mathbb{R}^3$:

$$\boldsymbol{F} = \begin{bmatrix} F_x cos\vartheta \\ F_x sin\vartheta \\ M \end{bmatrix} \tag{2.34}$$

Introducing a new control input vector $\boldsymbol{\tau} \in \mathbb{R}^2$ based on the active torques, defined as:

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_L \\ \tau_R \end{bmatrix} = \begin{bmatrix} \tau_1 + \tau_4 \\ \tau_2 + \tau_3 \end{bmatrix} \tag{2.35}$$

and combining 2.35 with eqs. 2.26, 2.33 and 2.34, is possible to rewrite the active force vector as:

$$\boldsymbol{F} = \boldsymbol{E}\left(\boldsymbol{q}\right)\boldsymbol{\tau} \tag{2.36}$$

where $\boldsymbol{E} \in \mathbb{R}^{3x2}$ is the input transformation matrix defined as:

$$\boldsymbol{E}\left(\boldsymbol{q}\right) = \frac{1}{r}\begin{bmatrix} cos\vartheta & cos\vartheta \\ sin\vartheta & sin\vartheta \\ -\frac{W}{2} & \frac{W}{2} \end{bmatrix} \tag{2.37}$$

Starting from eqs. 2.24,2.32, 2.36, and neglecting the centrifugal loads due to assumption 6, a first dynamic model is obtained:

$$\boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{R}(\dot{\boldsymbol{q}}) = \boldsymbol{E}(\boldsymbol{q})\,\boldsymbol{\tau} \tag{2.38}$$

It can be noted that this equation does not include the non-holonomic constraint 2.15 yet so describes only the dynamics of a free body. For imposing the constraint, a vector of Lagrange multipliers $\boldsymbol{\lambda}$, is introduced:

$$\boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{R}(\dot{\boldsymbol{q}}) = \boldsymbol{E}(\boldsymbol{q})\,\boldsymbol{\tau} + \boldsymbol{A}^T(\boldsymbol{q})\,\boldsymbol{\lambda} \tag{2.39}$$

However, to eliminate the unknown vector of Lagrange multipliers $\boldsymbol{\lambda}$, we left multiply equation 2.39 by the matrix $\boldsymbol{S}^T(\boldsymbol{q})$ defined in 2.19 and using the kinematic constraint propriety 2.18, it is possible to rewrite the dynamic model in the generalized coordinates $\boldsymbol{q}$:

$$\boldsymbol{S}^T(\boldsymbol{q})\,\boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{S}^T(\boldsymbol{q})\,\boldsymbol{R}(\dot{\boldsymbol{q}}) = \boldsymbol{S}^T(\boldsymbol{q})\,\boldsymbol{E}(\boldsymbol{q})\,\boldsymbol{\tau} \tag{2.40}$$

For control purposes it would be more suitable and more clear to express 2.40 in terms of the local velocity vector $\boldsymbol{\eta}$. After taking the time derivative of 2.17, we obtain:

$$\ddot{\boldsymbol{q}} = \dot{\boldsymbol{S}}(\boldsymbol{q})\,\boldsymbol{\eta} + \boldsymbol{S}(\boldsymbol{q})\,\dot{\boldsymbol{\eta}} \tag{2.41}$$

and replacing it in 2.40 the dynamic equations become:

$$\bar{\boldsymbol{M}}\dot{\boldsymbol{\eta}} + \bar{\boldsymbol{C}}\boldsymbol{\eta} + \bar{\boldsymbol{R}} = \bar{\boldsymbol{E}}\boldsymbol{\tau} \tag{2.42}$$

where:

$$\begin{aligned}
\bar{\boldsymbol{M}} &= \boldsymbol{S}^T(\boldsymbol{q})\,\boldsymbol{M}(\boldsymbol{q})\,\boldsymbol{S}(\boldsymbol{q}) \\
\bar{\boldsymbol{C}} &= \boldsymbol{S}^T(\boldsymbol{q})\,\boldsymbol{M}(\boldsymbol{q})\,\dot{\boldsymbol{S}}(\boldsymbol{q}) \\
\bar{\boldsymbol{R}} &= \boldsymbol{S}^T(\boldsymbol{q})\,\boldsymbol{R}(\dot{\boldsymbol{q}}) \\
\bar{\boldsymbol{E}} &= \boldsymbol{S}^T(\boldsymbol{q})\,\boldsymbol{E}(\boldsymbol{q})
\end{aligned} \tag{2.43}$$

By explaining equation 2.42 as a function of the control variables $\boldsymbol{\eta}$, it can be noted the presence of a direct dependence with the active torque vector $\boldsymbol{\tau}$. Since we have no direct information related to engine dynamics, to solve the model we assumed that the motor can always provide the angular velocity required by the motor velocity servo-controller. In such conditions, the motor dynamics can be described by a first order system defined as follows:

$$\dot{\omega}_i = K_w\left(\omega_{i_{ref}} - \omega_i\right) \tag{2.44}$$

It is also well known that the motor dynamics can be described by the equation:

$$\tau_i = I_m\dot{\omega}_i \tag{2.45}$$

where I_m represent the motor inertia. Combining equations 2.44 and 2.45, we obtain the following relation defining the torque control input:

$$\tau_i = I_m K_w\left(\omega_{i_{ref}} - \omega_i\right) \tag{2.46}$$

If we do not take into account any gear/transmission and electrical issue typical of DC motors, it is possible to state that $K_w = \frac{1}{I_m}$, therefore the control torque is simply provided by the difference between the desired and measured wheel angular velocities; in addition the relationship between the wheel angular velocity and the vehicle linear velocity is stated by eq. 2.14.

It must be remembered that such a model is justified only if the motor can always provide the desired torque. That assumption can be considered verified because of the low test loads and low-speed conditions. Moreover, any elasticity effect due to the motor belt has been taken into account by the model.

Enhanced Dry Friction Model

This section is dedicated to the description and implementation of an innovative dry friction model developed by Borello, *et al.*, in [53], able to overcome the characteristic limits of the classical Coulomb model and its other variants and evolutions proposed in literature [89, 90], maintaining consistency, robustness and ease of integration in complex problems.

The main advantages deriving from the model under analysis are the abilities to:

- Select the correct friction force/torque sign as a function of the sensed actuation rate;
- Distinguish between the sticking condition and the slipping one assigning respectively two different values to the friction force/torque: FSJ for static condition and FDJ for dynamic one;
- Evaluate the eventual stop of the previously moving element;
- Assesses the eventual breakaway of the element in standstill condition;
- Correctly keep the standstill element in a standstill position;
- Takes into account the presence of eventual mechanical end-stops, supposing a completely inelastic shock.

With reference to the graphic representation in figure 2.7, the model can be generically mathematically formulated as follows:

$$FF = \begin{cases} -F_{att} & \text{if } v = 0 \wedge |F_{att}| \leq FSJ \\ -sgn\left(F_{att}\right) FSJ & \text{if } v = 0 \wedge |F_{att}| > FSJ \\ -FDJ & \text{if } v \neq 0 \end{cases} \tag{2.47}$$

where FF is friction force/torque actually calculated coming as model output, FSJ is the static friction force/torque, FDJ the dynamic one and F_{att} is the active force/torque applied to the system.

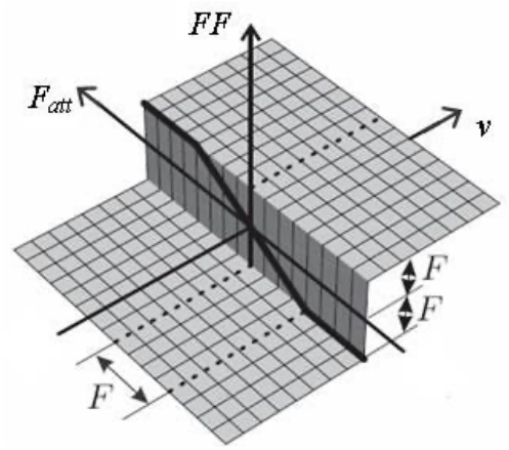


Figure 2.7: Borello dry friction model graphic representation.

In order to avoid numerical instability phenomena, similar to those manifested for example in the Karnopp model [89], during the execution of the model inside of simulation programs, the mechanical system stop is imposed, through a zero crossing detection algorithm, whenever linear or angular speed changes sign during the integration step:

$$\begin{aligned} v(t_{i+1}) &= 0 \quad \text{if} \quad v(t_{i+1})\, v(t_i) \leq 0 \\ &\text{or} \\ \dot{\vartheta}(t_{i+1}) &= 0 \quad \text{if} \quad \dot{\vartheta}(t_{i+1})\, \dot{\vartheta}(t_i) \leq 0 \end{aligned}$$

In case the imposed stop would not result to be correct the imbalance between active and resistive forces/torques acting on the system would cause its proper runaway at the next integration step. This control represents the fundamental innovation with respect the other models cited, giving to this method robustness and accuracy.

Another key point of the proposed model is the ability to distinguish aiding and opposing conditions, by comparing the load sign with the actuation speed one and choosing, step after step, the right parameters. A block diagram representation of the model is reported in figure 2.8, where it is possible to distinguish the static/dynamic friction and velocity reset branches.

Although the aforementioned model is easy to implement within a simulation program, applying it to the case under analysis results complex at a structural level, due to the matrix formulation of the SSMR dynamic model, and at computational cost since the presence of 4 wheels would require the application of the presented friction model to each wheel. For these reasons, it was decided to modify the structure of the dynamic model by simplifying and writing it through the equations of motion in Newtonian form.

With reference to figure 2.6, due to assumption 5 and eq. 2.31, the equation of motion can be written in the local frame as:

$$\begin{aligned} ma_x &= 2F_{x1} + 2F_{x2} - R_x(\dot{\boldsymbol{q}}) \\ ma_y &= F_y(\dot{\boldsymbol{q}}) \\ I\ddot{\vartheta} &= W(F_{x2} - F_{x1}) - M_r(\dot{\boldsymbol{q}}) \end{aligned} \tag{2.48}$$

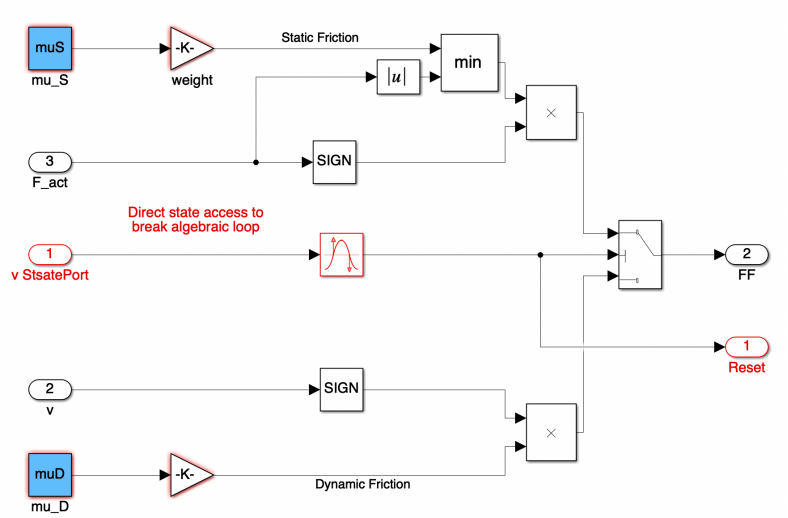


Figure 2.8: Borello dry friction model block diagram.

Introducing now 2 additional assumption, we consider the mobile robot symmetric with respect of its y axis, i.e. $a = b$, and we modify the non-holonomic constraint 2.15 imposing $x_{ICR} = 0$, id est the x position of the instantaneous centre of rotation lies on the CoG line. As result the new non-holonomic constraint became:

$$\dot{y} = 0 \tag{2.49}$$

that imposes the mobile robot no-lateral skidding. Consequently, $a_y = 0$ and we can rewrite the equation of motion as:

$$\begin{aligned} ma_x &= 2F_{x1} + 2F_{x2} - R_x\left(\dot{\boldsymbol{q}}\right) \\ I\ddot{\vartheta} &= W\left(F_{x2} - F_{x1}\right) - M_r\left(\dot{\boldsymbol{q}}\right) \end{aligned} \tag{2.50}$$

This formulation allows us to keep the longitudinal and lateral branches separated and therefore easily apply the proposed friction model for the computation of the reactive moment $M_r\left(\dot{\boldsymbol{q}}\right)$. With reference to figure 2.8 and to eqs. 2.47 and 2.50 we can now write the following variables as:

$$\begin{aligned} FF &= M_r\left(\dot{\boldsymbol{q}}\right) \\ F_{att} &= W\left(F_{x2} - F_{x1}\right) \\ v &= \dot{\vartheta} \end{aligned} \tag{2.51}$$

completing the friction model.

2.3 Linear State-Space Model

Even though the non-linear mathematical model provides a powerful tool for studying the SSMR dynamics and performances in all operating conditions, in some occasions the

high accuracy provided may not be necessary, and a simpler model is preferred. Such simplification has been obtained linearising the simplified equation of motion defined by eq. 2.50, which, in this case, were chosen without the improved friction model introduced in Section 2.2.2 as it is not suitable for the process here presented.

Since its friction forces represent the only non linearities of the system, the Jacobian linearisation around a near-zero operating point is not possible due to the presence of the sign function discontinuous first derivative. To cope with this discontinuity, it is possible to approximate the signum function by means of an arctangent one as proposed by [34]:

$$sgn\,(x) = \frac{2}{\pi}\arctan\,(k_s x) \tag{2.52}$$

where $k_s \gg 1$ is a constant representing the accuracy of the approximation according to the following relation:

$$\lim_{k_s \to \infty} \frac{2}{\pi}\arctan\,(k_s x) = sgn\,(x) \tag{2.53}$$

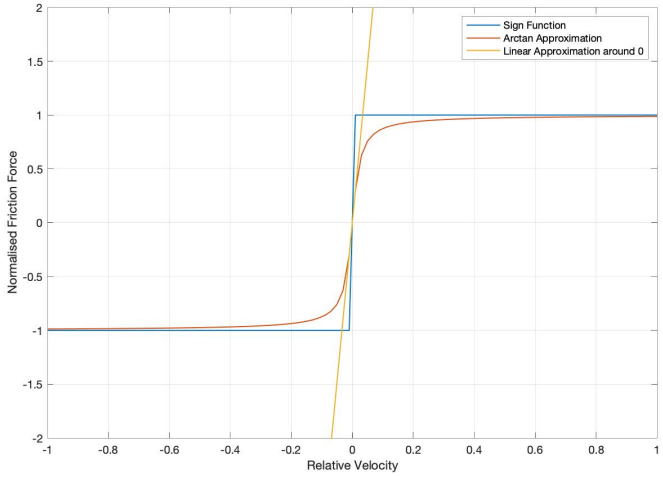


Figure 2.9: Friction approximation with Arctan and Hyper-Viscous models

Replacing 2.52 in 2.30 it is possible to rewrite the friction forces acting of each wheel:

$$\begin{aligned} R_{xi} &= u_{xi} mg \frac{2}{\pi}\arctan\,(k_{sx} v_{xi}) \\ F_{yi} &= \mu_{cyi} mg \frac{2}{\pi}\arctan\,(k_{sy} v_{yi}) \end{aligned} \tag{2.54}$$

and their Taylor expansion about an equilibrium point $\bar{v}_i$:

$$\begin{aligned} R_{xi} &\approx R_{xi}\,(\bar{v}_{xi}) + \left.\frac{\partial R}{\partial v_{xi}}\right|_{v_{xi}=\bar{v}_{xi}} \delta v_{xi} + \mathcal{O}\,(v_{xi}) \\ F_{yi} &\approx F_{yi}\,(\bar{v}_{yi}) + \left.\frac{\partial F}{\partial v_{yi}}\right|_{v_{yi}=\bar{v}_{yi}} \delta v_{yi} + \mathcal{O}\,(v_{yi}) \end{aligned} \tag{2.55}$$

where δv_i are the deviations defined as: $\delta v_i = v_i - \bar{v}_i$

Neglecting the higher order terms and making the partial derivatives explicit:

$$\begin{aligned} R_{xi} &\approx R_{xi}\left(\bar{v}_{xi}\right) + \mu_{xi}\frac{2k_{sx}}{\pi\left[1+\left(k_{sx}\bar{v}_{xi}\right)^2\right]}\delta v_{xi} \\ F_{yi} &\approx F_{yi}\left(\bar{v}_{yi}\right) + \mu_{cyi}\frac{2k_{sy}}{\pi\left[1+\left(k_{sy}\bar{v}_{yi}\right)^2\right]}\delta v_{yi} \end{aligned} \tag{2.56}$$

Choosing $\bar{v}_{xi} = \bar{v}_{yi} = 0$ as equilibrium point means $R_{xi}\left(\bar{v}_{xi}\right) = F_{yi}\left(\bar{v}_{yi}\right) = 0$ because there is no friction at zero velocity so, it is possible to rewrite eq. 2.56 as:

$$\begin{aligned} R_{xi} &\approx \frac{2}{\pi}k_{sx}u_{xi}mgv_{xi} \\ F_{yi} &\approx \frac{2}{\pi}k_{sy}\mu_{cyi}mgv_{yi} \end{aligned} \tag{2.57}$$

Considering the friction coefficients constant on a defined surface it is notable the similarity of eq. 2.57 with the form $K_i\dot{x}$ representing a viscous friction. Introducing the general viscous coefficient K_i it is possible to rewrite eq. 2.57 as:

$$\begin{aligned} R_{xi} &\approx K_{ui}v_{xi} \\ F_{yi} &\approx K_{yi}v_{yi} \end{aligned} \tag{2.58}$$

The formulation obtained is analogous to what is generally called hyper-viscous friction model [53], it presumes that viscous effects dominate on any other form of resistance in the range of application, resulting on the linear dependency of the resulting resistive force from the mobile robot speed as depicted in figure 2.9.

After having linearised the model, a state-space representation is now proposed because, providing the mathematical model of the system as a set of input, output and state variables related first-order differential equations, represents a simple and valid approach for the design and analysis of a feedback control system. Let's now define a set of notions necessary to better understand the following modelling process:

- **System state:** With reference to a dynamic system, the concept of *state* refers [61] to those minimum number of variables, named *state variables*, which, together with their initial value and with the input signals, are necessary to fully describe the system and its response at any time $t \geq t_0$.

- **State equations:** Represent the set of n decoupled first-order differential equations, composed by both state variables $x_i\left(t\right)$ and system inputs $u_i\left(t\right)$, which mathematically describe the system. In the general case the form of the n state equations is:

$$\begin{aligned} \dot{x_1} &= f_1\left(\boldsymbol{x},\ \boldsymbol{u},\ t\right) \\ &\vdots \\ \dot{x_n} &= f_n\left(\boldsymbol{x},\ \boldsymbol{u},\ t\right) \end{aligned} \tag{2.59}$$

for $i = 1, \ldots, n$. Limiting the attention to LTI systems, equations 2.59 become a set of n coupled first-order linear differential equations with constant coefficients, which, for a LTI system with r inputs, become:

$$\begin{aligned} \dot{x_1} &= a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + b_{11}u_1 + \cdots + b_{1r}u_r \\ \dot{x_2} &= a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n + b_{21}u_1 + \cdots + b_{2r}u_r \\ &\vdots \\ \dot{x_n} &= a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n + b_{n1}u_1 + \cdots + b_{nr}u_r \end{aligned} \tag{2.60}$$

where the constants $a_{i,j}$ and $b_{i,j}$ describe the system. Adopting a matrix form, system 2.60 can be written as:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \vdots \\ \dot{x_n} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \vdots \\ \dot{x_n} \end{bmatrix} + \begin{bmatrix} b_{11} & \ldots & b_{1r} \\ b_{21} & \ldots & b_{2r} \\ \vdots & \ddots & \vdots \\ b_{n1} & \ldots & b_{nr} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_r \end{bmatrix} \tag{2.61}$$

whose compact form is:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{2.62}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u} \in \mathbb{R}^r$ is the input vector, $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is a square constant matrix of coefficients $a_{i,j}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times r}$ is the input constant matrix of coefficients $b_{i,j}$.

- **Output equations:** The term *output* means any system variable of interest. Since all of the system variables can be represented as a linear combination of the states and the inputs of the system, a generic output of a nth order system with r inputs may be written as:

$$\begin{aligned} y_1 &= c_{11}x_1 + c_{12}x_2 + \cdots + c_{1n}x_n + d_{11}u_1 + \cdots + d_{1r}u_r \\ y_2 &= c_{21}x_1 + c_{22}x_2 + \cdots + c_{2n}x_n + d_{21}u_1 + \cdots + d_{2r}u_r \\ &\vdots \\ y_n &= c_{m1}x_1 + c_{m2}x_2 + \cdots + c_{mn}x_n + d_{m1}u_1 + \cdots + d_{mr}u_r \end{aligned} \tag{2.63}$$

or in matrix form as:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \ldots & c_{1n} \\ c_{21} & c_{22} & \ldots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \ldots & c_{mn} \end{bmatrix} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \vdots \\ \dot{x_n} \end{bmatrix} + \begin{bmatrix} d_{11} & \ldots & d_{1r} \\ d_{21} & \ldots & d_{2r} \\ \vdots & \ddots & \vdots \\ d_{m1} & \ldots & d_{mr} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_r \end{bmatrix} \tag{2.64}$$

whose compact form is:

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \tag{2.65}$$

where $\boldsymbol{y} \in \mathbb{R}^n$ is the output vector, $\boldsymbol{C} \in \mathbb{R}^{m \times n}$ is a constant matrix of coefficients $c_{i,j}$ that weight the state variables and $\boldsymbol{D} \in \mathbb{R}^{m \times r}$ is a constant matrix of coefficients $d_{i,j}$ that weight the system inputs, which for many physical system, including the one here analysed, is null leading to the form:

$$\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} \tag{2.66}$$

Knowing the basis of the state-space representation approach, is now possible deriving that formulation for our SSMR simplified mathematical model, trying to express it as the system:

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \end{cases} \tag{2.67}$$

In order to do so, it is possible to consider Eq. 2.14 as our input, obtaining:

$$\boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} v_x \\ \omega_z \end{bmatrix} = r \begin{bmatrix} \frac{\omega_L + \omega_R}{2} \\ \\ \frac{\omega_R - \omega_L}{W} \end{bmatrix} \tag{2.68}$$

from which, is possible to obtain the speeds of the wheels as:

$$\begin{aligned} \omega_L &= u_1 - u_2 \frac{W}{2} \\ \omega_R &= u_1 + u_2 \frac{W}{2} \end{aligned} \tag{2.69}$$

From Eqs. 2.46 and 2.69 is possible to obtain the control torque as:

$$\begin{cases} \tau_L = \left(u_1 - u_2 \frac{W}{2} - \omega_{L_{ref}}\right) K_w \\ \tau_R = \left(u_1 + u_2 \frac{W}{2} R - \omega_{R_{ref}}\right) K_w \end{cases} \tag{2.70}$$

also, expressing the reference wheel speeds as

$$\begin{cases} \omega_{L_{ref}} = \dot{x} - \dot{\vartheta} \frac{W}{2} \\ \omega_{R_{ref}} = \dot{x} + \dot{\vartheta} \frac{W}{2} \end{cases} \tag{2.71}$$

Eq. 2.70 become:

$$\begin{cases} \tau_L = \left(u_1 - u_2 \frac{W}{2} - \dot{x} + \dot{\vartheta} \frac{W}{2}\right) K_w \\ \tau_R = \left(u_1 + u_2 \frac{W}{2} R - \dot{x} - \dot{\vartheta} \frac{W}{2}\right) K_w \end{cases} \tag{2.72}$$

Starting from the assumptions proposed in Section 2.3, we can rewrite the simplified model defined by Eq. 2.50 as:

$$\begin{aligned} \ddot{x} &= \frac{1}{m} \left[(\tau_R + \tau_L) \frac{1}{r} - \dot{x} m g K_{u_x} \right] \\ \ddot{\vartheta} &= \frac{1}{J_z} \left[(\tau_R - \tau_L) \frac{W^2}{4r} - \dot{\vartheta} m g \frac{W}{2} K_\vartheta \right] \end{aligned} \tag{2.73}$$

and substituting Eq. 2.72 we finally obtain:

$$\begin{aligned} \ddot{x} &= \dot{x} \left(-\frac{2K_w}{rm} - g K_{u_x} \right) + \frac{2K_w}{rm} u_1 \\ \ddot{\vartheta} &= \dot{\vartheta} \left(-\frac{K_w W^2}{2 r J_z} - \frac{mg}{J_z} K_\vartheta \frac{W}{2} \right) + \frac{K_w W^2}{2 r J_z} u_2 \end{aligned} \tag{2.74}$$

Lets now make a change of variables. Introducing a set of *phase variables* it is possible to reduce the order of the system:

$$\begin{cases} x_1 = x \rightarrow \dot{x}_1 = x_2 \\ x_2 = \dot{x} \rightarrow \dot{x}_2 = \ddot{x} \quad \text{so} \quad \dot{x}_2 = x_2\left(-\frac{2K_w}{rm} - mgK_{u_x}\right) + \frac{2K_w}{rm}u_1 \\ x_3 = \vartheta \rightarrow \dot{x}_3 = x_4 \\ x_4 = \dot{\vartheta} \rightarrow \dot{x}_4 = \ddot{\vartheta} \quad \text{so} \quad \dot{x}_4 = x_4\left(-\frac{K_w W^2}{2rJ_z} - \frac{mg}{J_z}K_\vartheta\frac{W}{2}\right) + \frac{K_w W^2}{2rJ_z}u_2 \end{cases} \tag{2.75}$$

Converting now the system derived from 2.75 in its matrix form, the state-space representation of the model is obtained:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2K_w}{rm} - mgK_{u_x} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{K_w W^2}{2rJ_z} - \frac{mg}{J_z}K_\vartheta\frac{W}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{2K_w}{rm} & 0 \\ 0 & 0 \\ 0 & \frac{K_w W^2}{2rJ_z} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.76}$$

Regarding the output equation, since the output variables required by the control system are the same as the state variables and matrix $\boldsymbol{D}$ is considered null, we can simply write:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{2.77}$$

Writing Eqs. 2.76 and 2.77 in in their compact form, we obtain the final formulation:

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \end{cases} \tag{2.78}$$

Taking a close look to equation 2.76 it is not difficult to notice that the effects of the two control parameters are decoupled. Indeed the input u_1 only affects the linear motion of the WMR, while the input u_2 only affects its angular response. It is therefore possible, tanks to the assumptions made in Section 2.3, on which the whole linear formulation is based, to consider the Multi Input Multi Output (MIMO) system as two decoupled Single Input Single Output (SISO) systems thus simplifying the control system analysis.

Chapter 3

Experimental Set-Up and Model Tuning

This chapter takes a closer look to the SSMR mathematical models proposed in Chapter 2 describing their implementation within the Simulink simulation environment and the tuning necessary to faithfully simulate the behaviour of the Clearpath Husky testing platform. For this reason, the first part of this chapter is dedicated to explain which was the experimental set-up used for the development of this thesis: going from the indoor testing facility of the BeiDou Research Institute in Shanghai to the description of the Husky rover and of its interaction with ROS. After that, a heuristic approach, based on a self-adaptive Differential Evolution algorithm, for determining the unknown parameters is applied to each model and the comparison of their resulting performances, to check the simulation accuracy obtainable, is carried out. Finally, the last pages are dedicated to the Gazebo robotics simulator accuracy analysis concluding with the explanation of why it was not used during the project.

3.1 BRI Indoor Test Facility

The experimental tests presented in this thesis were carried out at the Robotic Testing and Training Area of BeiDou Research Institute Shanghai division; institute that deals with the development of the Chinese satellite navigation system. The BeiDou[1] project was officially initiated in 2000, with its first experimental satellite constellation named BeiDou-1, composed of 3 satellites offering limited coverage and navigation services mainly for users in China and neighbouring regions. Decommissioned towards the end of 2012, it was replaced by the second generation of the system, officially called BeiDou Navigation Satellite System (BDS), offering services to customers in the Asia-Pacific region. Starting from March 2015, China launched the third and current generation BeiDou system, namely

[1]It is named after the Big Dipper asterism, which is known in Chinese as 北斗 (pinyin: Běidǒu) that literally means *'Northern Dipper'*. This name was given by ancient Chinese astronomers to the seven brightest stars of the Ursa Major constellation, set of stars historically used in navigation to locate the North Star.

BeiDou-3, for global coverage constellation. To this day, BeiDou-3 counts 33 satellites, the latest of which launched on December 16, 2019. The last two satellites meant to complete its fleet are due to become operational by June 2020. When fully completed, BeiDou will provide an alternative global navigation satellite system to United States GPS, Russian GLObal NAvigation Satellite System (GLONASS) or European Galileo systems, and is expected to be more accurate than these, reaching millimetre-level accuracy after Satellite-based Augmentation Systems (SBAS) implementation.[2] By 2020, its market value is estimated at 400 billion RMB, approximately 52 billion euros.

According to a 2018 report conducted by the China Satellite Navigation Office, the axes of development for BDS are multiple, ranging from mass applications such as smartphones enhanced localization and factory-installed car navigation systems, but also to much broader sectors such as smart cities, transportation, public security, disaster alleviation and relief, agriculture and meteorological detection. It aims to push China towards a leadership position in key technological sectors through the implementation in the new industry 4.0 sector of a series of technological innovations aimed at improving industrial productivity. These include cloud computing, artificial intelligence and more broadly smart automation like drone delivery or automated warehouse logistics.

The Shanghai BRI division located at West Hongqiao, see Figure 3.1b was developed along those axes, focusing on the development of innovative technologies for UAVs and UGVs.

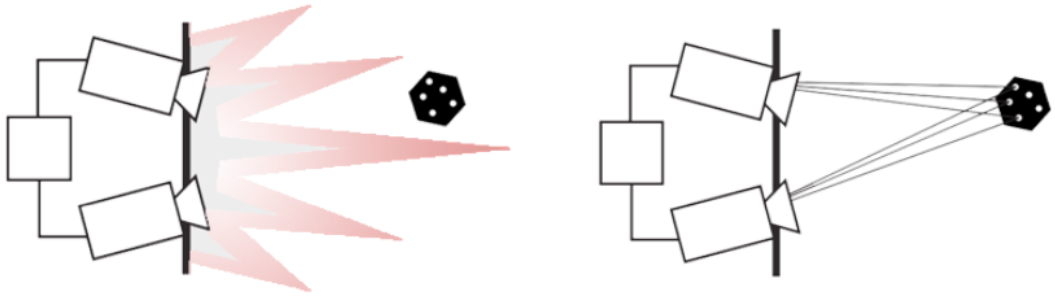
(a) Robotics Testing and Training Area. (b) West Hongqiao facility aerial view.

Figure 3.1: BeiDou Research Institute.

About a quarter of the fifth floor of building 1, the left oval-shaped one shown in Figure 3.1b, hosts the institute's Robotics Testing and Training Area, a 500 squared meters indoor laboratory equipped with one of the world's most comprehensive infrared-marker multi-camera localization system, counting 62 REALis motion tracking Realis Tracking System (RTS)1000 cameras, which specifications are summarized in Table 3.1, and a Ultra-Wideband (UWB) positioning system. This technology establishes state of the art in indoor localization, achieving a tenth of millimetre accuracy. The system employs high precision infra-red cameras tracking a set of retro-reflective markers mounted on the

[2]For more information the reader is referred to consult BeiDou website.

target, as represented in Figure 3.2.



(a) The object is lit using near IR light. (b) Retro-reflective marker reflect back.

Figure 3.2: Representation of IR marker multi-camera localization system.

Table 3.1: RTS1000 camera specifications.

PERFORMANCES	
IMMAGE RESOLUTION	1280×1024 @ 210 FPS
FIELD OF VIEW	$58° \times 48°$
LENSES	Fixed focus $F\#1.8 \times 5.5$ mm
LEDS	14 high power LEDs
MEASUREMENT FREQUENCY	4.8 ms
MAXIMUM DISTANCE OF THE OBJECT	13 m
SIZE AND WEIGHT	
DIMENSIONS	$90 \times 91 \times 70.5$ mm
WEIGHT	0.55 kg
MATERIAL	Alluminium alloy
POWER SYSTEM AND INTERFACING	
POWER SUPPLY	POE + DC 12V
DATA INTERFACE	Gigabit RJ54
OUTPUT	2D markers location

3.2 Clearpath Husky

The test bench used in this thesis for carrying out the experimental test and for validating the simulations results is Clearpath HUSKY™A200 robot base developed by CLEARPATH ROBOTICS™; it is a medium-sized robotic development platform globally trusted by hundreds of researchers and engineers. Due to the numerous research papers published using it as the test set-up, Husky provides a well-tried benchmark for establishing new robot research and development efforts.

Its large payload capacity and power systems can accommodate a large variety of customised payloads, and it is plug-and-play compatible with a wide range of accessories and sensors for meeting research needs, such as stereo cameras, LIDAR, GPS, and IMUs. Even industrial manipulators can be easily interfaced with the platform for teleoperated or autonomous manipulation tasks. The Husky's rugged construction and high-torque maintenance-free drivetrain built out of durable materials with very few moving parts make it suitable to deal with challenging real-world terrain. Furthermore, Husky was the first robotics platform to support ROS from its factory settings, making it easy to use and to integrate with the existing research carried out by the ROS community.

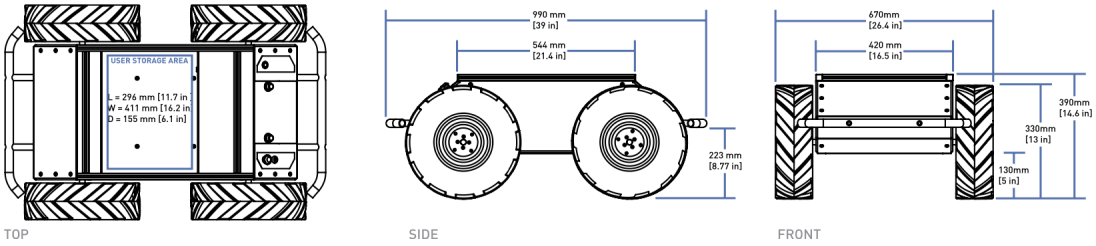


Figure 3.3: Husky robot base dimensions.

By going into detail, Husky is a 50 kg lightweight all-terrain vehicle with an International Protection (IP) rating of IP55 [91] that fully protect it against splash water and dust; thus avoiding interferences with equipment correct operations.

Due to its generous ground clearance and a width of almost 1m, see Fig. 3.3 and table 3.2, the mobile robot is able to easily get over large obstacles and thanks to its 330mm off-road tires with four-wheel high-torque drive, it is suitable for traversing harsh terrains and for operating with confidence in sand, snow, mud and dirt. Moreover its skid-steer configuration, similarly to tracked vehicles, give it excellent traction on the surface making it particularly suited for rough terrains. However, the performance of the vehicle strongly depends on the ground it is crossing, and speed of the vehicle since skidding can cause unpredictable power requirements because of terrain irregularities and non-linear tire–soil interaction. Its left and right pairs of wheel are mechanically linked, and each pair is actuated by an electric motor; however, their specifications were not made available by the producer.

In terms of performances, the robot base can take a maximum payload of 75 kg over out-door terrains, reaching speeds of up to 1.0 m/s, and being able to climb slopes of up to 45°.

Table 3.2: Husky base specifications.

SIZE AND WEIGHT	
EXTERNAL DIMENSIONS (LxWxH)	$990 \times 670 \times 390\ mm$
INTERNAL DIMENSIONS	$296 \times 411 \times 155\ mm$
WEIGHT	$50\ kg$
WHEEL RADIUS	$330\ mm$
GROUND CLEARANCE	$130\ mm$
SPEED AND PERFORMANCE	
MAX. PAYLOAD	$75\ kg$
ALL-TERRAIN PAYLOAD	$20\ kg$
MAX SPEED	$1.0\ m/s$
DRIVETRAIN / DRIVE POWER	4×4 Zero-Maintenance
MAX CLIMB GRADE	45°
MAX TRAVERSAL GRADE	30°
BATTERY AND POWER SYSTEM	
BATTERY CHEMISTRY	Sealed Lead Acid
CAPACITY	$24\ V, 20\ Ah$
RUNTIME - STANDBY	$8\ Hours$
RUNTIME - NOMINAL USAGE	$3\ Hours$
CHARGE TIME	$4\ Hours$
USER POWER	$5\ V, 12\ V, 24\ V$ Fused at $5\ A$ each. $192\ W$ total available power.
INTERFACING AND COMMUNICATION	
CONTROL MODES	Direct voltage, wheel speed, and kinematic velocity.
FEEDBACK	Battery voltage, motor currents, wheel odometry, and control system output.
COMMUNICATION	$RS232$ @ 115200 $baud$
ENCODERS	Quadrature: 78000 $pulses/m$
DRIVERS AND APIs	ROS, C++, and Python.
ENVIRONMENTAL	
OPERATING AMBIENT TEMPERATURE	$-10\ to$ 40 °C Not in direct sunlight
STORAGE TEMPERATURE	$-40\ to$ 60 °C
RATING	IP 55

The A200 Husky is equipped with several sensors and hardware placed inside the ample storage area. In addition, some conversion kits can be purchased for mounting sensors on an external structure placed above the aluminium rail of the robot base.

The robot is powered by a single 24 V, 20 Ah sealed Lead-Acid battery, capable of delivering 1800 W, placed in the battery compartment on the rear chassis. The power outputs provided are 5 V, 12 V, and 24 V fused at 5 A each and allows the robot to operate continuously for 3 h.

The Husky base is equipped with high-resolution quadrature encoders with 78000 $pulse/m$ that gives it improved state estimation and dead reckoning capabilities. It multiple control modes, such as direct voltage, wheel speed or kinematic velocity, provides flexibility for

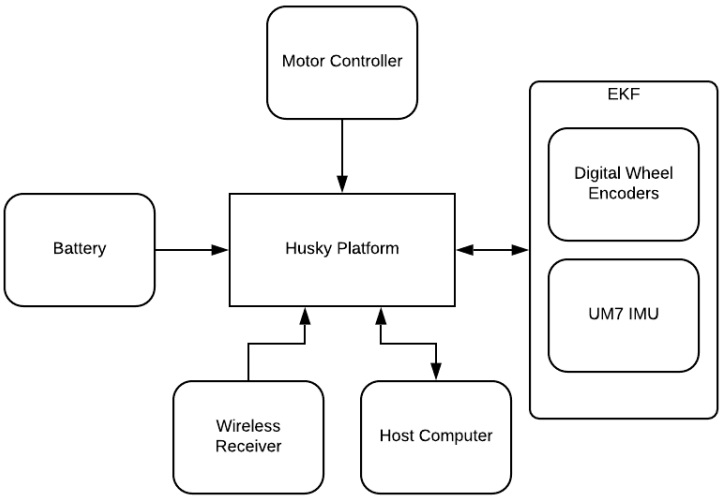


Figure 3.4: Husky Platform block diagram

specific research objectives and application requirements. In addition to wheel encoders, an UM7 Orientation Sensor containing gyroscopes, accelerometers and magnetometers, provides 3D orientation expressed in quaternion, acceleration and rate of turn to the mobile robot. These outputs generated by the two sensors are combined in an EKF used to estimate the position of the robot knowing only its previous state, the applied input and the sensor measurements, giving the localisation system improved dead reckoning capabilities. Without it would be severally compromised by the skid phenomenon. The EKF relies on the ROS *localization package*, developed by the Clearpath, with which it is possible to directly edit the filter parameters that, for the purpose of this work were left as default.

In terms of communication, the Husky A200 base provides an RS232 Connector for serial port communication, which interfaces with the hardware via a USB-serial adapter that we have connected to an USB Port Hub enabling communication to the outside. This equipment enables USB connection of the robot to an external computer receiving feedback data for on-line and off-line analysis. The host computer chosen to process all data and run the developed control system is a *XPS 15 9570* with an *Intel Core i7-8750H 2.20 GHz* running on Linux 18.04 LTS and using ROS Melodic Morenia as middleware.

The mobile robot can be controlled in two ways: via a remote controller giving linear and angular velocity direct commands or via the on-board computer using ROS as middleware for giving commands with different levels of authority. The commands natively supported with these control interfaces are of two types, plus a third one implemented as part of the research work:

1. **Linear and angular velocity:** the robot has no autonomy and the wheel speeds are regulated and maintained by an embedded controller which characteristics are

not made available by the producer. This is the only type of command which can be generated by a remote.

2. **Target pose[3]:** once communicated, the robot tries to reach it generating corresponding speed commands for traversing a known environment. If the robot is equipped with the required sensors, it can also navigate autonomously in an unknown environment mapping its surroundings for localising itself. Once it founds its location, the robot auto-generates sequential intermediate target poses and the associated speed commands for reaching them.

3. **Reference Trajectory:** command not supported natively, can be interpreted only by mean of the control system developed in this research. The method in which the trajectory is processed is similar to that used for autonomous navigation. In essence, sequential intermediate target poses are feed in the control system, but this time every pose is associated with a time stamp.

3.2.1 Robot Operating System Interface

Despite the existence of many different robotic frameworks that were developed in the last decade, the significant advantages of ROS, such as hardware abstraction, low-level device control, implementation of commonly-used functionalities, message and package management, made it the most trending and popular robotic framework.

The A200 Husky is fully integrated with the robot operating system framework tanks to the Clearpath's driver for the robot base and fortunately, ROS provides seamless integration of external sensors thus we make use of the UM7 driver available by the ROS community for the IMU and EKF integration. Furthermore, other packages have been used, in particular, the tele-operation and the Virtual-Reality Peripheral Network (VRPN) packages respectively for directly controlling the robot via joystick sending velocity commands and for receiving the ground truth of the rover via the RTS. Tele-operation is particularly useful to park the robot or to position it in a specific place, i.e. at the initial point of the trajectory. For this porpoise, a Logitech joystick is used, which can communicate with the robot via a wireless receiver connected in the USB hub inside the robot. The commands published by the tele-operation node have the highest priority, second only to the emergency stop button mounted on the rover chassis, meaning that it can be used for overriding any command and safely recover remote control of the vehicle.

The implementation of such packages creates small executables: the ROS nodes, see Fig. 3.5, which expose library functionalities to ROS. When booting the robot, since the whole system is the result of an integration of several different modules, it will automatically check which components are plugged in and start their respective driver; which become responsible for publishing sensor data in dedicated ROS topics. The robot's communication light on the rear panel will change from red to green, indicating that the robot operating system is up and has established communications with the Husky.

[3]A pose indicates the coordinates of a point in space plus its orientation; following the convention reported in chapter 2, a generic pose can be expressed as a three-element vector $[X,\ Y,\ \vartheta]$.

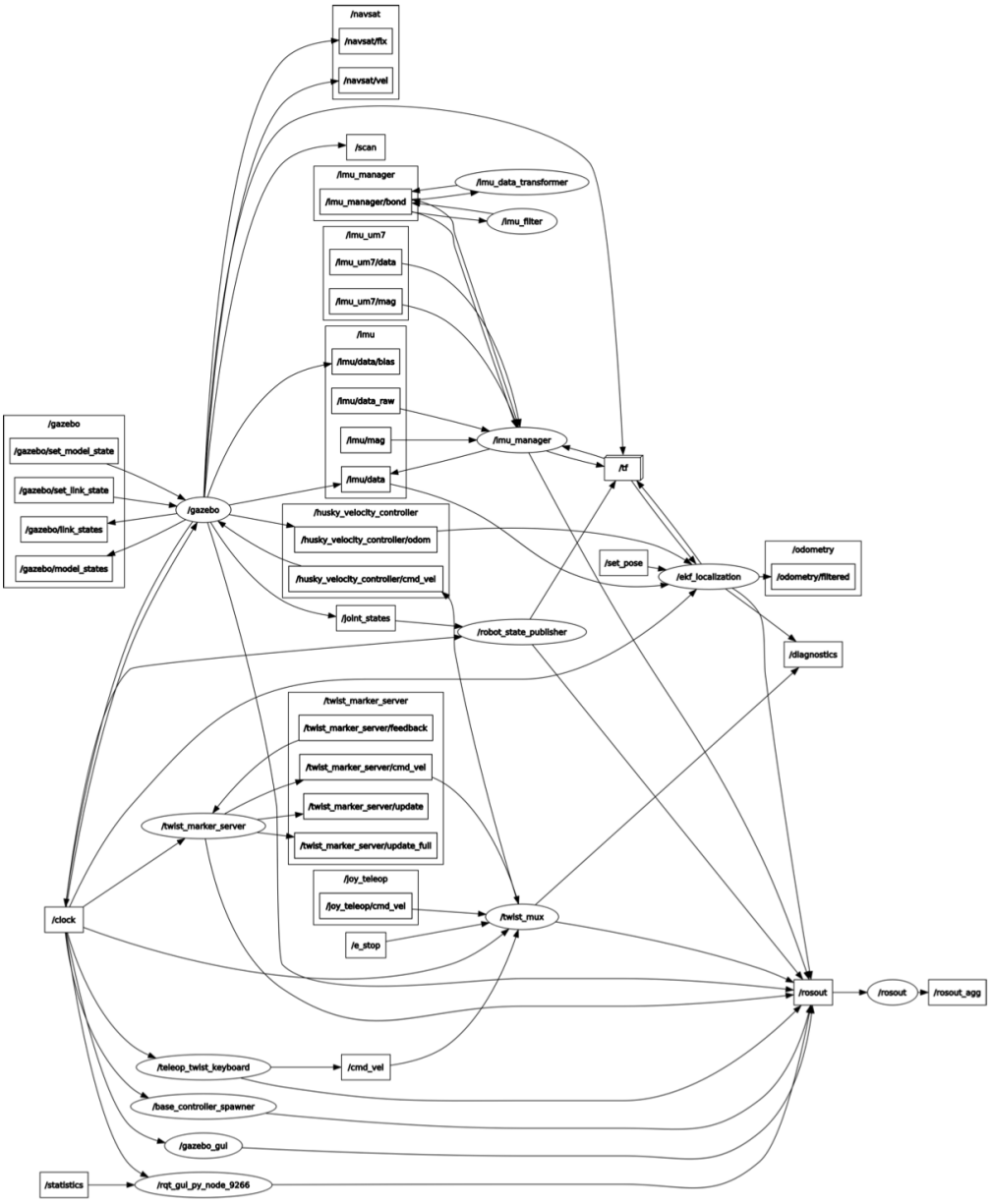


Figure 3.5: ROS rqt graph[4]of the Husky robot used during tests

The controllers have been developed in MATLAB/Simulink R2019b and using the stable distribution ROS Melodic and the MathWorks ROS toolbox it was possible to test the controllers on the mobile platform by publishing the commands in the *cmd_vel* topic and receiving the feedback by subscribing respectively at the *odometry_filtered* and

[4]The presence of the node */gazebo* is due to the fact that the image was taken during a session on Gazebo simulator.

vrpn_ client_ node/pose[5] topics. Moreover, some bash scripts[6] has been written to automate some steps during simulations, such as running the executable for connecting at the VRPN client at the beginning of the simulations or calling the service for set/reset the initial position between two simulations. All the other data useful for post-processing is saved in rosbags[7] and exported in the MATLAB workspace.

3.2.2 Default Mobile Platform Internal Controller

The Husky mobile platform drivers, which communicate with the motors, are in the *husky_ node* that continuously exchange data to a series of topics. One of these is *cmd_ vel*, which accepts a translational velocity and a rotational velocity vector for command purposes. When data is published on this topic, *husky_ node* extracts the x velocity v_x and z angular velocity ω and feeds these into the *diff_ drive_ controller* for:

- Converting the robot reference velocities into wheel speeds;
- Actuating the wheels via the electric motors;
- Controlling the wheel speeds;

It is a generic controller used for differentially steered devices, and handles both body velocity to wheel speed conversions and dead reckoning odometry calculations, using the following equations:

$$\begin{aligned} \omega_L &= \frac{1}{w_r r}\left(v_x - \frac{W w_s \omega}{2}\right) \\ \omega_R &= \frac{1}{w_r r}\left(v_x + \frac{W w_s \omega}{2}\right) \\ v_x &= \frac{w_r r\,(\omega_R + \omega_L)}{2} \\ v_y &= 0 \\ \omega &= \frac{1}{W w_s}\left[w_r r\,(\omega_R - \omega_L)\right] \end{aligned} \tag{3.1}$$

where w_r and w_s are respectively the *wheel radius multiplier* and *wheel separation multiplier*; these two constants are used to reduce the error deriving from having considered the mobile robot as differential drive and therefore to take into account the presence of skidding. Since the internal architecture and parameters of this build-in controller are not made available to the public, Section 3.4 propose a system identification method to cope with this uncertainties.

[5]For receiving this data the on-board computer and the host computer on which the RTS client is run must be connected at the same Wi-Fi network.

[6]These scripts are called by executing operating system commands from the MATLAB command line.

[7]Bags are typically created by a tool like rosbag, which subscribe to one or more ROS topics, and store the serialised message data in a file as it is received.

3.3 Simulink Models Implementation

The following paragraphs deals with the Simulink implementation of the tree SSMR mathematical models proposed in Chapter 2, namely:

1. The non-linear model with Coulomb friction formulation.
2. The non-linear model with enhanced friction formulation.
3. The LTI state-space model with hyper-viscous friction formulation.

Regardless of the mathematical formulation with which they were derived, each model shares some characteristics, starting from, see Fig 3.6, the definition of the input-output scheme:

- **Inputs:**
  - u_cmd: commanded linear velocity.
  - $omega\_cmd$: commanded angular velocity.
- **Outputs:**
  - q: mobile robot pose in the global reference frame, with components $[X,\ Y,\ theta]$.
  - dq: mobile robot velocity in the global reference frame, with components $[dX,\ dY,\ dtheta]$.
  - eta: mobile robot velocity in the local reference frame, with components $[dx,\ dheta]$.

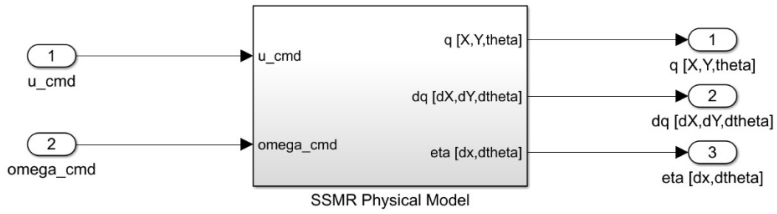


Figure 3.6: Input-Output representations of the proposed SSMR Simulink models.

Entering the *SSMR Physical Model* block represented if Fig. 3.7, apart from the *Husky Plant* block which containing the mathematical formulation of the model differs in each individual variant, it is possible to identify other common components, namely:

- **Saturation blocks:** needed for limiting the upper and lower values of the commands u_cmd and $omega\_cmd$. Regarding the Husky platform these values respectively are bounded in the ranges $[-1,\ 1]$ m/s and $[-2,\ 2]$ rad/s.

- **Delay blocks:** needed for simulating the delay on each command caused by the internal processing unit during the transmission and conversion of the signals. This is one of the values covered by the system identification process subsequently presented in Section 3.4.
- **Wheel Torque Controller block:** Needed for simulating the feedback control loop that regulates and converts the velocity inputs into wheel torque commands. As seen in Fig. 3.8 the controller, based on equation 2.46 presented in Section 2.2.2, is composed by an upper branch which converts the saturated and delayed commands in reference wheel velocities, and a lower one which converts the feedbacked local velocity vector into the actual wheel speeds. The two branches are subtracted, and the result is fed to a proportional gain and saturated for simulating the mechanical limits of the electric wheel motors. The two proportional gains here introduced are the other two values affected by the identification process.

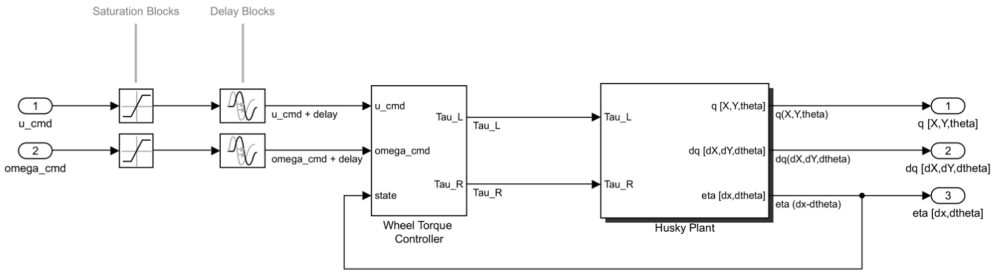


Figure 3.7: Insight on *SSMR Physical Model* Simulink block.

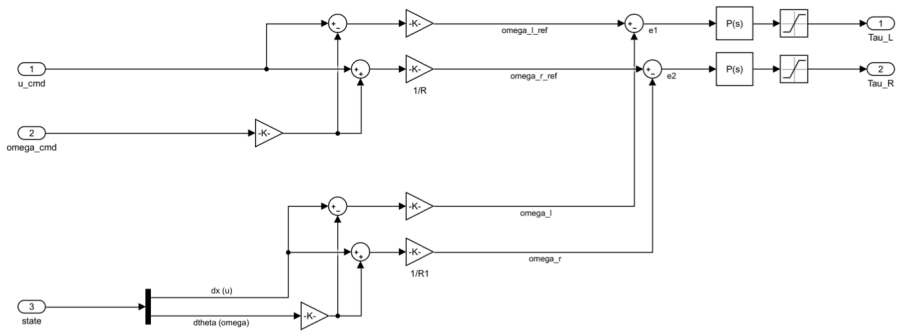


Figure 3.8: Insight on *Wheel Torque Controller* Simulink block.

3.3.1 Non-linear Model with Coulomb Friction Formulation

The model depicted in figure 3.9 is the block-diagram representation of the dynamic system defined by Eq. 2.42. Since each component is easily identifiable within the model, this representation does not require further explanation.

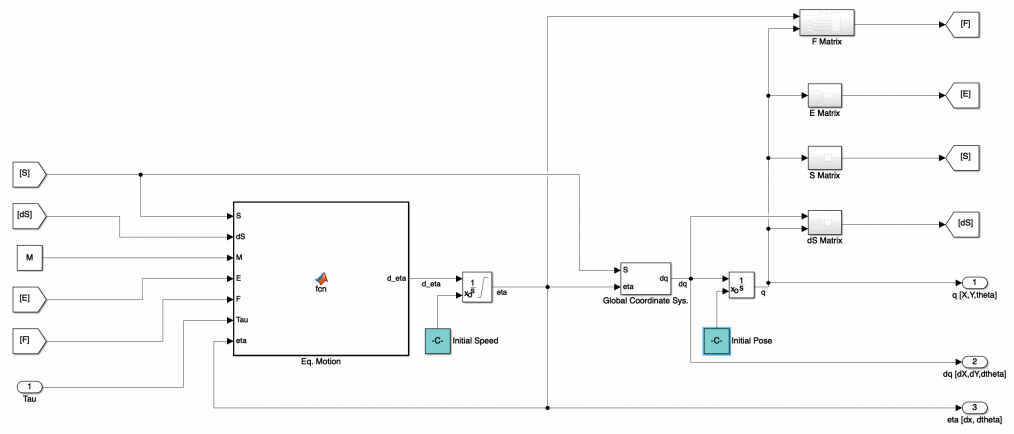


Figure 3.9: Insight on Non-linear *Husky Plant* with Coulomb friction Simulink block.

3.3.2 Non-linear Model with enhanced Friction Formulation

Figure 3.10 defines the block-diagram representation of the simplified non-linear dynamical model deriving from the simplifications made in Section 2.2.2 for obtaining the system equations 2.50. The upper branch, composed by the green blocks, defines the longitudinal dynamics of the WMR, while the lower one, composed by the blue blocks, computes its latero-directional behaviour.

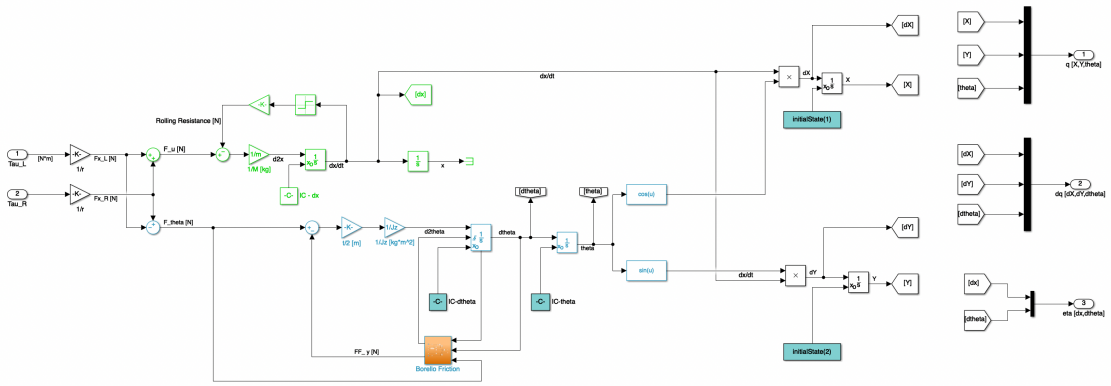


Figure 3.10: Insight on Non-linear *Husky Plant* with enhanced friction Simulink block.

The reason why a further non-linear model of the aircraft has been proposed, also containing greater simplifications, lies in the possibility to implement an enhanced friction model, whose properties and mathematical formulation has been presented in Section 2.2.2, that is believed to bring significant improvements in the behaviour of the mobile robot model, especially regarding the wheel-soil interaction problem. The model is implemented via the orange block within the latero-directional loop; its structure, that can be seen from Fig. 2.8 is based on eq. 2.47 and receives three different inputs:

- v *StatePort*: In this case represents the rover angular velocity ω computed by the

state port of the integrator positioned between $\ddot{\vartheta}$ and ϑ. That velocity must be taken from the state port to avoid the occurrence of algebraic loops that stops the simulation program. This quantity is used in the zero-crossing detection algorithm that resets the angular velocity, i.e. every time a change in its sign is detected, the value of ω is set to zero.

- v: Namely ω, the standard output of the integrator described above, whose sign is used to define the direction of the static resistive force.
- F_act: Represent the resultant of the active forces, whose sign is used to define the direction of the static resistive force.

3.3.3 LTI State-Space Model with Hyper-viscous Friction Formulation

Finally, the block-diagram representation of the system State-Space formulation obtained in Section 2.3, and based on eqs. 2.76 and 2.77 is presented in figure 3.11. It is pointed out that all the computations made after the demultiplexer are needed only to express the result in global coordinates.

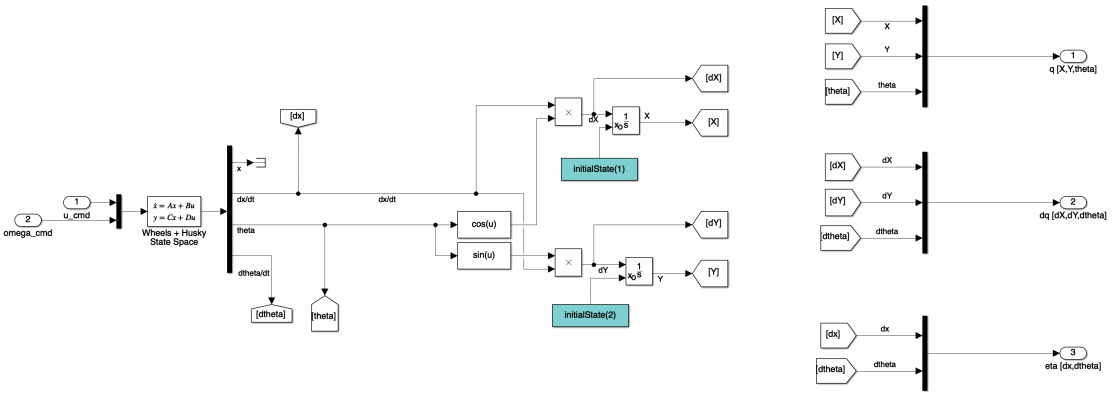


Figure 3.11: Insight on LTI State-Space *Husky Plant* Simulink block.

3.3.4 Sensor Noise Modelling

As presented in Section 3.2, the Clearpath Husky used for experimental tests, for locate itself is space relied on five different sensors, four wheel encoders and a IMU, whose data is interpreted and merged in an EKF. This signal, being generated by physical sensors, is intrinsically characterised by background noise, which, if not properly assessed, can significantly affect the performance of the developed controllers. One example could be the derivative action of a PID controller which, if not tuned accounting for noise, can bring the system to unstable behaviours when tested in real scenarios. For this reason, the following section proposes a simple approach to modelling the sensor noise; however, its analysis, which can be considered a discipline by itself, is outside the scope of this thesis.

To understand the properties of the noise affecting the signals generated by the UGV sensors, respectively its position $\boldsymbol{q}\,(X,\ Y,\ \vartheta)$ and velocity $\boldsymbol{\eta}\,(v_x,\ \omega)$ vectors, the EKF output was measured in different scenarios, defined in Table 3.3, and then replicated via MATLAB and Simulink.

Table 3.3: Test cases employed for the EKF sensor noise analysis.

Case №	$\boldsymbol{u_c}$ $\boldsymbol{[m/s]}$	$\boldsymbol{\omega_c}$ $\boldsymbol{[rad/s]}$
1	0.25	0
2	0.50	0
3	0	0.25
4	0	0.50

The process adopted for each test case was the following:

1. The measurements of the interested signals were converted into *timeseries* objects.
2. Each timeseries was high-pass filtered to keep only the noise component of the signal. The high-pass filter settings are shown in Table 3.4.

Table 3.4: High-pass filter settings.

Setting	**Value**	
Passband frequency	10.0	Hz
Trasnition band steepness	0.85	
Stopband Attenuation	60.0	dB

3. The noise variance was evaluated via the MATLAB *var* command.

The obtained variances, summarized in Table 3.5, are then used to set up the Simulink random number blocks whose output is summed to the original signals to replicate their noise component, as in the example reported for the position signal in Figure 3.12.

Table 3.5: Test cases employed for the EKF sensor noise analysis.

Parameter	**Variance**	**Parameter**	**Variance**
X	8.8786×10^{-6}	v_x	5.0115×10^{-5}
Y	7.9259×10^{-12}	ω	3.5298×10^{-5}
ϑ	1.2941×10^{-8}		

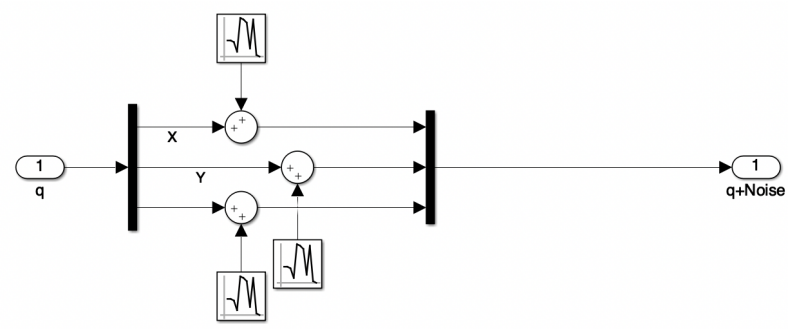
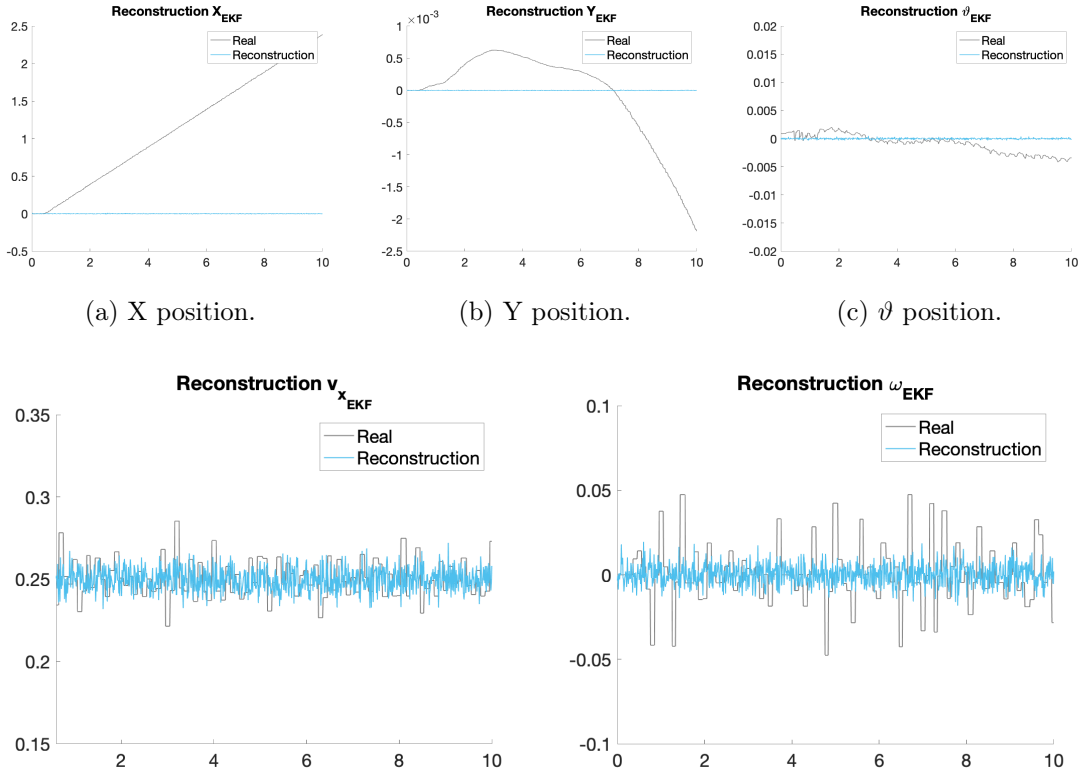


Figure 3.12: $\boldsymbol{q}$ vector sensor noise component within the SSMR Simulink model.

An example of the results achieved in the test case №1 is presented in Figure 3.13.



(a) X position. (b) Y position. (c) ϑ position.

(d) Linear velocity. (e) Angular velocity.

Figure 3.13: EKF output reconstruction compared to real one for test case №1.

3.3.5 External Disturbances Modelling

The external disturbances acting on the Husky UGV during the experimental tests within the test facility laboratory environment can be considered very limited but, to better test

the control architectures to external uncertainties, they were included in the Simulink model. The disturbance has been chosen to affect the angular acceleration of the rover and was simulated using a random number block with a variance, determined by trial and error, set to 25. Whose output was summed to the angular acceleration $\ddot{\vartheta}$, as can be seen from Figure 3.14.

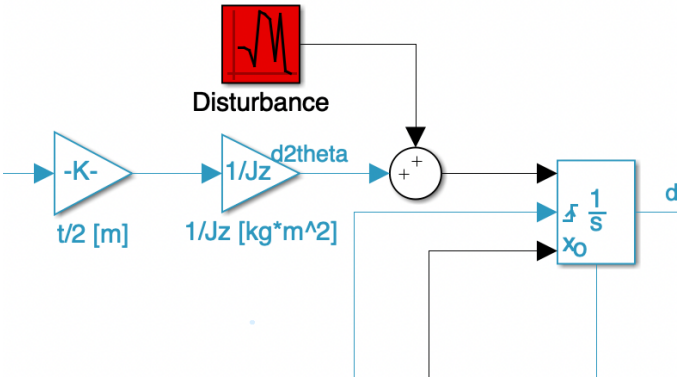


Figure 3.14: External disturbance implementation on the SSMR Simulink model.

3.3.6 Model Discretization

In order to simulate the real behaviour of the Husky platform, the Simulink model representation has been discretised following a Zero Order Hold (ZOH) approach. The ZOH Simulink block introduces the discretisation by holding its input for the sample period specified. With regards to the Clearpath Husky, multiple signals need to be transmitted or received at a specific frequency, as shown in Table 5.1. These are the commands, and the odometry and ground truth respectively generated by the on-board EKF and the laboratory RTS.

3.4 Unknown Parameters Estimation: an Evolutionary Approach

From what emerges from the previous section, the presence of 6 uncertain parameters, i.e. two delays in the control action, two proportional gains in the *Wheel Torque Controller* and the longitudinal and lateral friction coefficients, does not guarantee the reliability of the results deriving from the simulations of the proposed models, necessitating a trial and error approach that would slow down the development of the controllers and affect their performance and reliability. For this reason, a heuristic approach for the determination of these parameters is proposed here. The selected method is based on the use of an optimisation algorithm belonging to the class of Evolutionary Algorithm (EA): the Differential Evolution. In particular, an adaptive version named *jDE/best/2*, characterised by a gradient descent self adaptability of its control parameters and by the use of the two-difference vector perturbation *best/2* mutation strategy. The following only contains

the description and the results of the global optimisation process; therefore, the reader is referred to Appendix A which contains a detailed description of the proposed algorithm and its pseudo-code.

The underlying idea behind this process requires:

1. A reliable reference response dataset. To obtain that, a real Husky was controlled with a sequence of known commands. Since it was tested in the known laboratory environment presented in Section 3.1, its exact response was tracked and measured tanks to the RTS.

2. The execution of numerous simulations. To do this, the DE algorithm was implemented into a recursive MATLAB script so that, by executing a large number of Simulink runs, it could experiment with different values of the unknown parameters until the simulated behaviour would mimic the real one.

By following this approach, and by testing the results on more than one sample trajectory, it has been possible to find the correct combination of the unknown parameters which guarantees the most faithful simulation model. Obviously, the single parameters could not be exact if taken alone, but their combination is. Furthermore, this type of optimisation, which aims at replicating a real, measured result, allows to include in the output parameters the effects of other nearly exact or totally unaccounted physical quantities, if not of entire dynamic effects.

The generated MATLAB script has been divided to allow faster reconfiguration and better recognition of each task into a main file and two sub-functions:

- ***main*:** in which all the required parameters are initialized and the final optimized results are saved. Its pseudo-code is reported in Algorithm 3.1.

Algorithm 3.1 Parameters estimation *main* file.

Input: Reference trajectory, SSMR properties and algorithm control parameters

Output: Optimized unknown parameters

1: Import reference trajectory and related commands
2: Set SSMR properties, initial conditions, Simulink solver parameters
▷ Begin jDE initialization
3: Initialise unknown parameters as **global** variables and declare its search bounds
4: Set objective function weights
5: Set population number NP
6: Set maximum generations NG
7: Initialize scale factor F
8: Initialize crossover rate CR

▷ *Continued in the next page*

Algorithm 3.1 – *Continued from previous page*

```
Set objective function threshold df
                                                    ▷ Start optimization
procedure jDE optimization
    Call jDE_best_2_desc function
    Receive optimized parameters
end procedure
Save and print optimization results
```

- ***jDE_best_2_desc*:** in which the mutation, crossover and selection processes are executed. Is the core of the jDE algorithm, as we can se from the listing in Algorithm 3.2.

Algorithm 3.2 Parameter estimation *jDE_best_2_desc* function.

Input: Control parameters from *main*
Output: Optimized unknown parameters

```
Receive control parameters from main
Initialize jDE control parameters
for each nth parameter to be optimized do
    Define initial population of target vector
end for
while number of generations ≤ NG AND relative difference between the objective functions of two subsequent generations ≤ df do
    for each element of the population do
        Find four random indexes within NP
        Generate rand_F in the range [0, 1]
                                                    ▷ Perform Mutation
        procedure Define F given the relationship between rand_F(1) and τ1
            Define F                                ▷ Equation (A.9)
        end procedure
        Define the mutant vector                    ▷ Equation (A.4)
    end for
    for each element of the population do
        Generate rand_CR in the range [0, 1]
                                                    ▷ Perform Crossover
        procedure Define CR given the relationship between rand_CR(1) and τ2
            Define CR                               ▷ Equation (A.10)
        end procedure
```

▷ *Continued in the next page*

Algorithm 3.2 – *Continued from previous page*

```
        Define the trial vector                                  ▷ Equation (A.7)
    end for
    for each element of the population do
        procedure SIMULATION RUN AND SELECTION
            Call eval_obj_fun function for both sample vector and trial vectors
            Receive objective functions results
                                                                 ▷ Perform Selection
            if objective function improves using the trial vector then
                Keep trial vector
                Evaluate relative improvement of objective function
            end if
        end procedure
    end for
    Move to next generation
end while
Return optimized unknown parameters
```

- ***eval_obj_fun*:** which receives, see Algorithm 3.3, the parameter vector obtained from the optimization algorithm, uses it to run a Simulink simulation with whom results evaluates the objective function and returns it to the jDE function for the selection process.

Algorithm 3.3 Parameter estimation *eval_obj_fun* function.

Input: Trial vectors and simulation parameters from *jDE_best_2_desc*
Output: Objective function results

```
Receive the simulation parameters and trial vectors from jDE_best_2_desc function
                                                                 ▷ Parse results
Run Simulation
Evaluate RMS of minimum distances                                ▷ Equation (1.3)
Evaluate Hausdorff distance                                      ▷ Equation (1.2)
Evaluate bending energy                                          ▷ Equation (1.4) If needed
Evaluate objective function                                      ▷ Equation (3.2)
Return objective function result
```

For the optimisation process of the Husky rover physical models, it was chosen to employ the following quantities to assess the fitness of the trial and target vector during the selection process. The cost function f_{opt} to be minimised has bees selected as the combination of the features of the evaluated trajectory derived from some of the metrics

presented in Section 1.5:

$$f_{opt} = k_d d_{ratio} + k_{haus} d_{haus} + k_{rms} d_{rms} \tag{3.2}$$

where the constant values $k_d,\ k_{haus},\ k_{rms}$ have been chosen, see Table 3.6, for guarantee a balanced effect of the evaluated features on the cost function. In particular, looking more closely at each component of the function, it is possible to express some considerations. Regarding the RMS of the minimum distances and the Hausdorff distances, the greater the accuracy of the mathematical model, the closer the two metrics should be to zero. Then, talking about the distance ratio, in a perfect simulation, its value equals 1.

Table 3.6: Objective function gains for the mathematical models optimization.

Parameters	**Values**	**Parameters**	**Values**
$\boldsymbol{k_d}$	5	$\boldsymbol{k_{haus}}$	8
$\boldsymbol{k_{rms}}$	20		

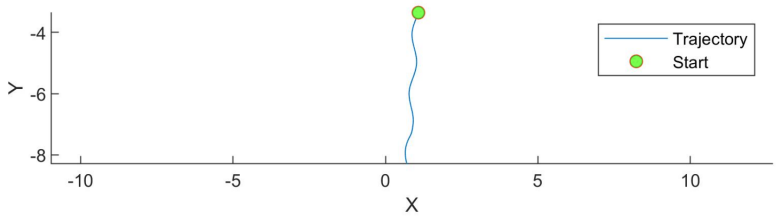
3.4.1 Sample Trajectories

Four different reference trajectories have been tested in the BRI robotics laboratory. To standardise each test hast been crested a Python ROS node able to execute a predefined series of commands based on the tracking of linear and angular velocity. In particular, because the major interest was to assess the dynamic effects which arise during steering, it has been decided to command constant linear velocity and, once the rover had reached its linear target speed, to introduce an angular sinusoidal component too, whose amplitude and frequency features were changed between tests to obtain different responses and therefore consider different cases. In particular, the cases characterised by higher frequencies or amplitudes has also been considered to assess the more evident non-linear effects which significantly changes the vehicle response. The parameters of the the four trajectories are listed in Table 3.7 and a graphical example of the commands provided for trajectory number 2 is present in Figure 3.15.

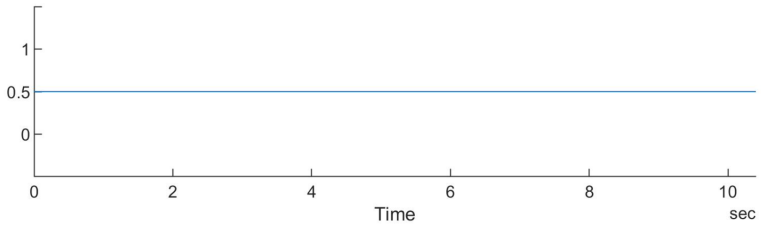
Table 3.7: Sample trajectories parameters.

№	$\boldsymbol{u_{cmd}}$ $\boldsymbol{[m/s]}$	$\boldsymbol{\omega_{cmd}}$ $\boldsymbol{[rad/s]}$	$\boldsymbol{f}$ $\boldsymbol{[Hz]}$	$\boldsymbol{A}$ $\boldsymbol{[m]}$
1	0.50	$A\sin(\omega t)$	0.50	1.00
2	0.50	$A\sin(\omega t)$	0.25	0.25
3	0.50	$A\sin(\omega t)$	0.25	0.50
4	0.50	$A\sin(\omega t)$	0.25	0.75

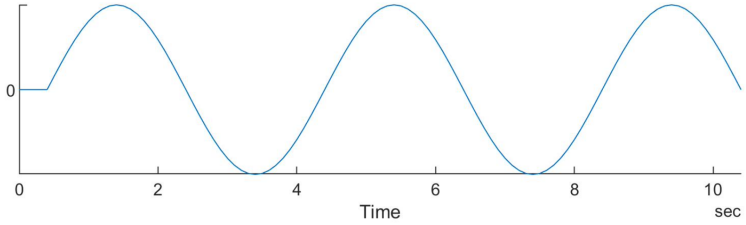
The Husky position was measured using the RTS system and its data, together with the rover internal measurements, were registered using the rosbag functionality provided



(a) Odometry.



(b) Linear velocity command.

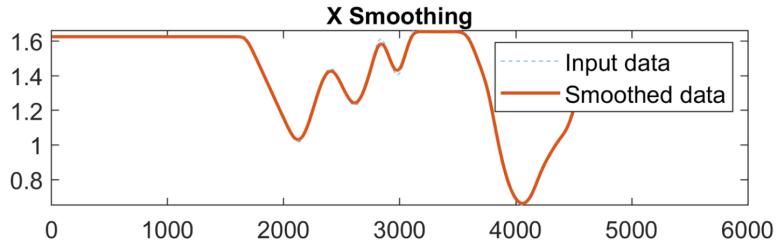


(c) Angular velocity command.

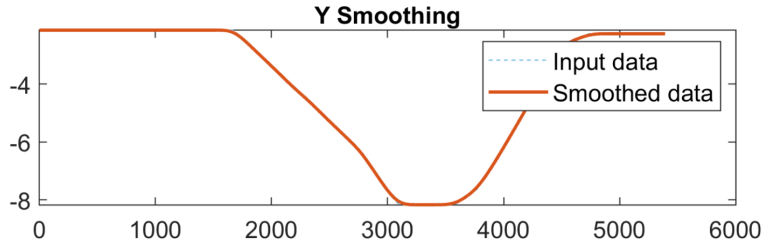
Figure 3.15: Sample trajectory №2.

by ROS. The rosbag was then processed in MATLAB by cropping and smoothing its data to remove some sensors' noise. The smoothing process relied on the native *smoothdata* MATLAB function, using a Gaussian-weighted moving average with a *SmoothingFactor*[8] of 0.01. In the case here presented, a very low *SmoothingFactor* was adopted for removing only the underlying sensor noise. It was concluded that more intense smoothing would have cut through some important parameters variations. An example of the low impact smoothing applied is visible in Figure 3.16.

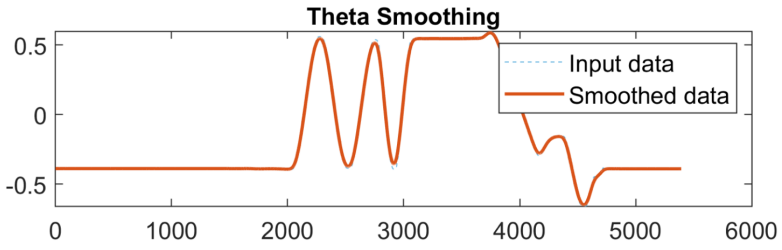
[8]Represent a parameter that determines the size of the window considered by the algorithm at each data point. In essence, a higher value produces a more significant smoothing.



(a) Along linear X.



(b) Along Y.



(c) Along ϑ.

Figure 3.16: Example of the RTS data after smoothing procedure.

3.4.2 Optimization of the Non-linear Model with Coulomb Friction Formulation

Concerning the non-linear mathematical model with the Coulomb friction formulation, the following parameters had to be determined:

- Wheel torque controllers proportional gain K_w. Their value are considered equal on each branch.
- Rolling resistance coefficient u_x.
- Dynamic lateral friction coefficient between the wheels and the ground μ_{cy_d}.

As for the linear and angular velocity commands delays, namely $delay_u$ and $delay_{omega}$, they were manually determined by observing the response curves of the rover. While the

maximum torque generated by the electric motors was found using the Husky specifications published by Clearpath Robotics [91].

By instructing the EA, adopting the configuration parameters reported in Table 3.8 which also contains the control parameters of the jDE algorithm defined in Appendix A, to replicate the fourth trajectory described in Table 3.7, since it represents the most common application case of the Husky UGV, the algorithm convergence was achieved after 1500 iterations obtaining the results reported in Table 3.9.

Table 3.8: Configuration parameters for non-linear model optimization.

Parameters	**Values**	**Parameters**	**Values**
F_{init}	0.75	CR_{init}	0.80
F_l	0.10	F_u	1.00
CR_{min}	0.00	CR_{max}	1.00
τ_1	0.10	τ_2	0.10
nr	4.0	ξ	0.20
df	0.001	dt	0.001
NP	50.0	NG	300.0

Table 3.9: Non-linear model unknown parameters results.[9]

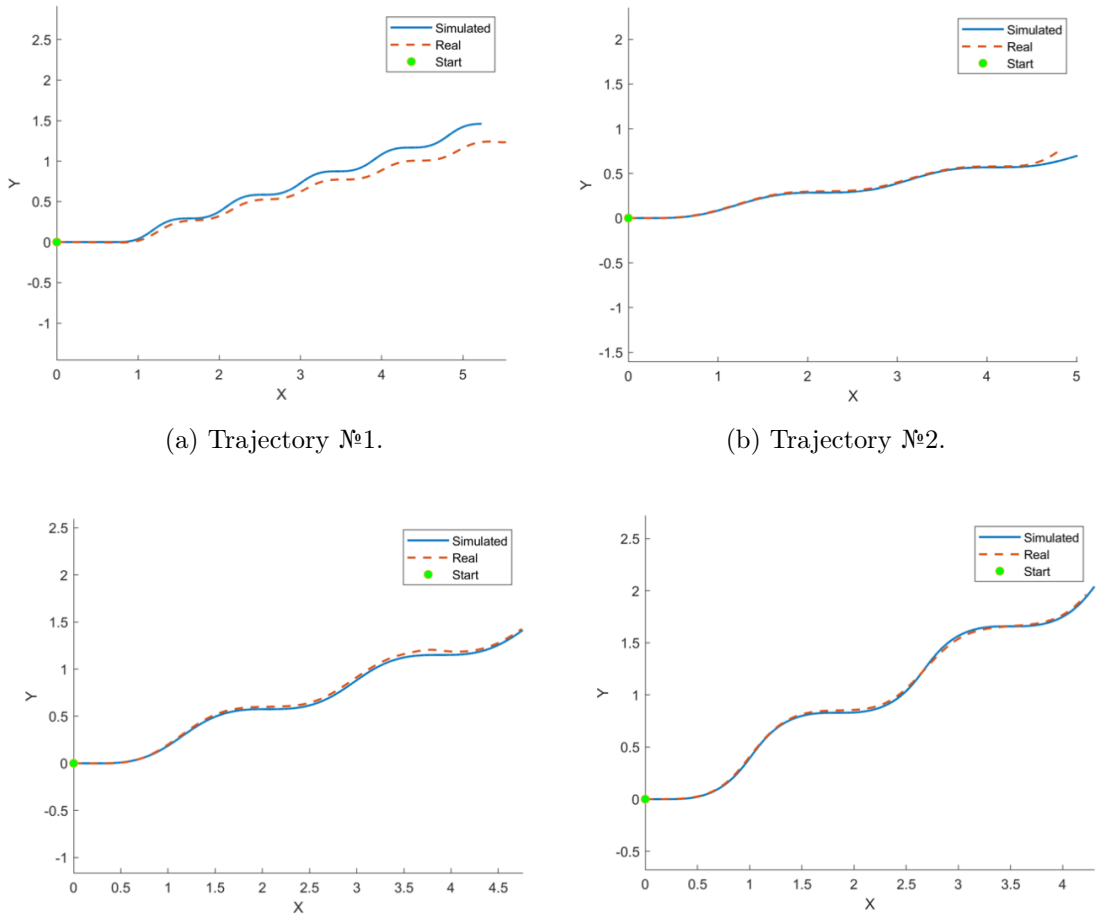
Parameters	**Search Boundaries**	**Results**	
$delay_u$	–	0.22	s
$delay_\omega$	–	0.22	s
$\boldsymbol{\tau}_{max}$	–	$\pm$7.0	N m
u_x	[0.0, 0.1]	0.0727	m
μ_{cy}	[0.1, 0.2]	0.0254	
K_w	[20, 200]	56.75	

These parameters were then tested on the other remaining sample trajectories giving the results in Table 3.10 and depicted in Figure 3.17.

[9]It should be specified that these results were obtained after a short refinement procedure in which the objective function weights, defined in Table 3.6, were adjusted until they guaranteed the best results, and the parameters search boundaries were gradually reduced as well to ensure faster convergence.

Table 3.10: Non-linear model optimization objective functions results over the four sample trajectories.

№	d_{ratio}	d_{haus}	d_{rms}
1	0.9666	0.3726	0.1416
2	1.0408	0.1039	0.0217
3	0.9998	0.0557	0.0252
4[10]	1.0312	0.0250	0.0146



(a) Trajectory №1.

(b) Trajectory №2.

(c) Trajectory №3.

(d) Trajectory №4.[10]

Figure 3.17: Non-linear model optimization trajectories results.

[10]Trajectory used for jDE optimization.

Comparing the results, the following two considerations can be drawn:

1. The trajectory on which the optimisation was made, namely trajectory №4, is simulated with a higher degree of realism. Assertion confirmed by its objective function coefficients result in Table 3.10 and its trajectory represented in Figure 3.17d. The values of d_{rms} and d_{haus} have an order of magnitude of 1×10^{-2}, while d_{ratio} shows that the simulated trajectory is only 3.12% longer than the real one, leading to a satisfactory result.

2. Trajectory №1 is the one that most diverges from reality. This lies in two reasons: first, it is the case with the higher frequency and amplitude of the angular velocity command, which means it is the case with the greatest non-linearities. Secondly, it was concluded that the electric motors deliver a slightly different torque, leading to some unexpected behaviours which are almost invisible in the other more linear trajectories. This will be clearer while analysing the experimental results of the trajectory tracking controllers in Chapter 5.

3.4.3 Optimization of the Non-linear Model with enhanced Friction Formulation

Analysing now the version of the non-linear mathematical formulation containing the enhanced friction model, the following four parameters had to be determined:

- Wheel torque controllers proportional gain K_w. Their value are still considered equal on each branch.
- Rolling resistance coefficient u_x.
- Static lateral friction coefficient between the wheels and the ground μ_{cy_s}.
- Dynamic lateral friction coefficient between the wheels and the ground μ_{cy_d}.

The commands delays, and the maximum wheel torque K_w, were kept identical to the previous case, as well as the other configuration parameters that are reported in Table 3.8 and the fitness function gains of Table 3.6. After a little adjustment of the search boundaries, the same optimisation procedure used in Section 3.4.2 was carried out on trajectory №4 coming to the convergence of the algorithm after 2500 iterations and obtaining the results reported in Table 3.11

The obtained parameters were then tested on the other remaining sample trajectories giving the results in Table 3.12 and depicted in Figure 3.18c.

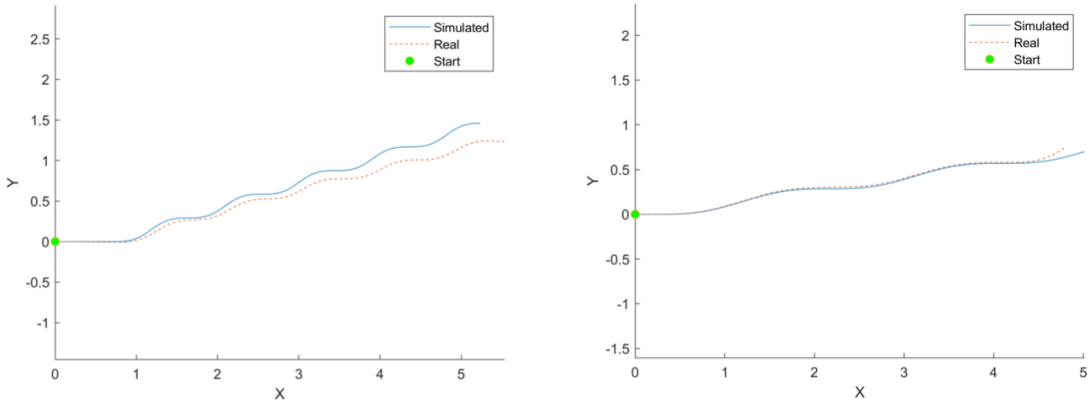
Comparing the results, the considerations made in the previous case in Section 3.4.2 are still valid. It is necessary to specify though, that since the distinction between static and dynamic lateral friction is introduced by means of the advanced model, the search boundaries of μ_{cy_s} and μ_{cy_d} have to be chosen so that they didn't overlap, to avoid the risk of running into numerical problems during simulations.

Table 3.11: Non-linear model with enhanced friction formulation unknown parameters results.

Parameters	**Search Boundaries**	**Results**	
$delay_u$	–	0.22	s
$delay_\omega$	–	0.22	s
τ_{max}	–	$\pm$7.0	N m
u_x	[0.05, 0.20]	0.1282	m
μ_{cy_s}	[0.08, 0.15]	0.0972	
μ_{cy_d}	[0.0, 0.07]	0.0409	
K_w	[60, 120]	118.04	

Table 3.12: Non-linear model with enhanced friction formulation optimization objective functions results over the four sample trajectories.

№	d_{ratio}	d_{haus}	d_{rms}
1	0.9668	0.3720	0.1420
2	1.0411	0.1049	0.0222
3	1.0000	0.0561	0.0252
4[11]	1.0312	0.0255	0.0145

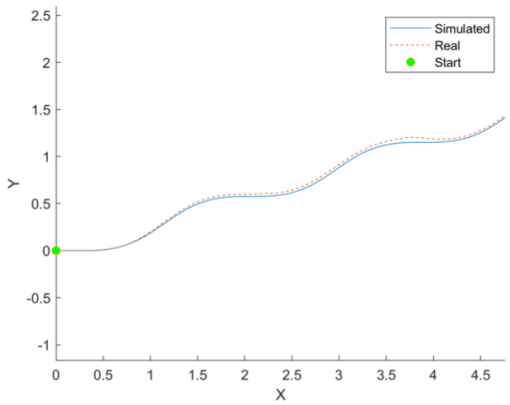


(a) Trajectory №1.

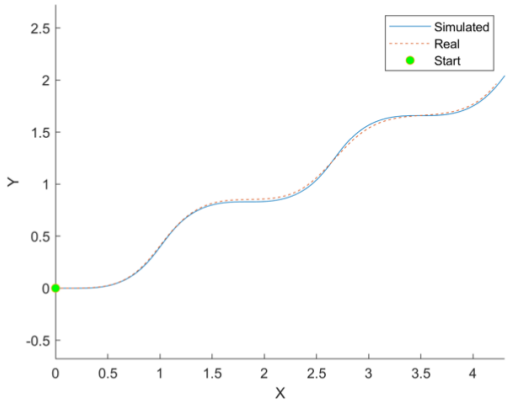
(b) Trajectory №2.

Figure 3.18: Non-linear model with enhanced friction formulation optimization trajectories results. (*continued*)

[11] Trajectory used for jDE optimization.



(c) Trajectory №3.



(d) Trajectory №4.[11]

Figure 3.18 – *Continued from previous page.*

3.4.4 Optimization of the LTI State-Space Model with Hyper-viscous Friction Formulation

Finally, considering the linear state-space formulation, the following three parameters had to be determined:

- Wheel torque controllers proportional gain K_w. Their value are still considered equal on each branch.
- Longitudinal viscous coefficient K_{u_x}.
- Lateral viscous coefficient K_ϑ.

The commands delays, and the maximum wheel torque K_w, were kept identical to the other previous case, as well as the other configuration parameters that are reported in Table 3.8 and the fitness function gains of Table 3.6. After a little adjustment of the search boundaries, the same optimisation procedure used in Sections 3.4.2 and 3.4.3 was carried out on trajectory №4 coming to the convergence of the algorithm after 5750 iterations and obtaining the results reported in Table 3.13

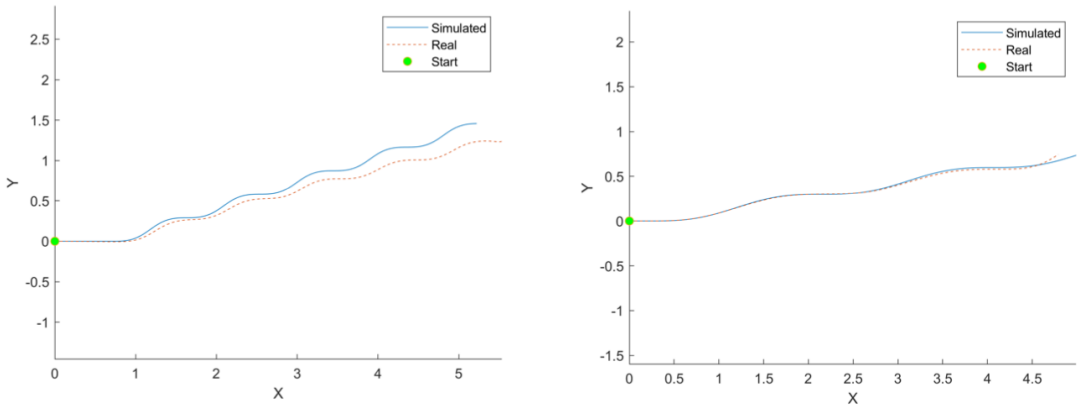
The obtained parameters were then tested on the other remaining sample trajectories giving the results in Table 3.14 and depicted in Figure 3.19.

Table 3.13: State-space model with hyper-viscous friction formulation unknown parameters results.

Parameters	**Search Boundaries**	**Results**	
$delay_u$	–	0.22	s
$delay_\omega$	–	0.22	s
$\boldsymbol{\tau}_{max}$	–	$\pm$7.0	N m
K_{u_x}	[0.01, 0.03]	0.0277	
K_ϑ	[0.00, 0.03]	0.0298	
K_w	[80, 300]	229.58	

Table 3.14: State-space model with hyper-viscous friction formulation optimization objective functions results over the four sample trajectories.

№	$\boldsymbol{d_{ratio}}$	$\boldsymbol{d_{haus}}$ [m]	$\boldsymbol{d_{rms}}$ [m]
1	0.9641	0.3793	0.1427
2	1.0382	0.0677	0.0154
3	0.9972	0.0451	0.0176
4[12]	1.0285	0.0269	0.0150

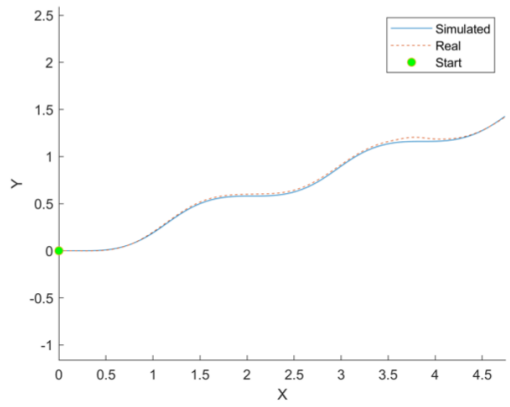


(a) Trajectory №1.

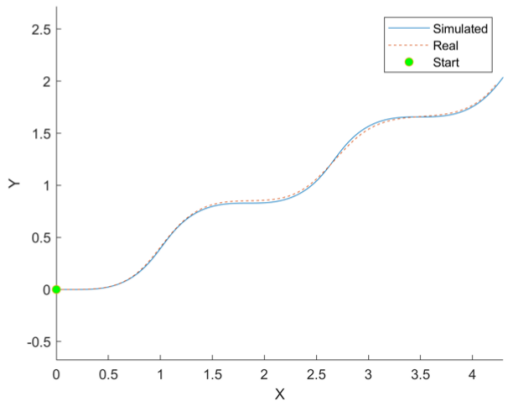
(b) Trajectory №2.

Figure 3.19: State-space model with hyper-viscous friction formulation optimization trajectories results. (*continued*)

[12]Trajectory used for jDE optimization.



(c) Trajectory №3.



(d) Trajectory №4.[12]

Figure 3.19 – *Continued from previous page.*

Comparing the results, the considerations made in the first case in Section 3.4.2 are still valid.

3.5 Mathematical Models Comparison

By comparing the accuracy of the three mathematical models, it is clear that the results are very similar to each other, as evidenced in Table 3.15. In addition none of the models can correctly represent the trajectory №1, see Figures 3.17a, 3.18a and 3.19a. As already explained, that is due to unaccounted non-linearities and wheel motor imperfections. As expected, despite having unknown parameters in common, i.e. wheel torque proportional gain and friction coefficients, their optimised value changes from case to case, as the DE algorithm looks for the best suitable combination of parameters, instead of their isolated exact values.

Table 3.15: Mathematical models comparison over trajectory №4.

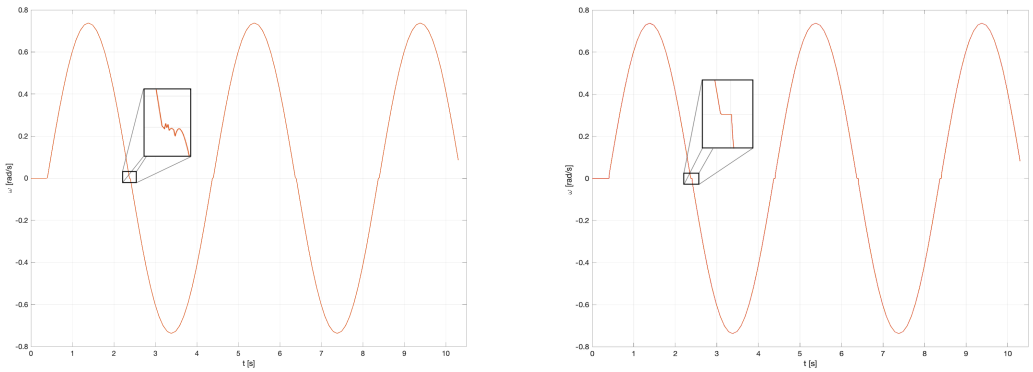
Model	d_{ratio}	d_{haus} [m]	d_{rms} [m]
LTI hyper-viscous friction	1.0285	0.0269	0.0150
Non-linear Coulomb friction	1.0312	0.0250	0.0146
Non-linear improved friction	1.0314	0.0255	0.0144

The linear model appears to be a fitting simplification of the non-linear counterparts. It was therefore used to assess the initial response of the proposed controllers for finding the stability boundaries of the parameters to be optimised. The non-linear ones, on the other hand, are required for accurate simulations. By going beyond the trajectories comparison, the most significant differences can be appreciated by looking at the angular velocity curves and the simulation time. Referring to Figure 3.20, it possible to notice that

the angular velocity trend ω differ from one model to the other. The model with Coulomb friction formulation, see Figure 3.20a, exhibits numerical artefacts near the velocity sign reversal area, while the improved friction formulation, see Figure 3.20b, is much more faithful as it includes the stiction phenomenon.

Regarding the simulation time, the Coulomb friction model requires more time to be executed; on average, it takes 2.18 seconds to simulate the fourth sample trajectory, due to its complexity. While the improved friction one, with all its simplifying assumptions, only takes about 0.50 seconds. A reduction of 77% is of great importance when the simulation has to be repeated numerous times, as in the case of the controllers optimisation via EA.

Given all the considerations above, only the non-linear model with the improved friction formulation was used to design and test the controllers.



(a) Coulomb friction formulation.

(b) Enhanced friction formulation.

Figure 3.20: Angular velocity trends on trajectory №4.

3.6 Gazebo Simulator Accuracy Analysis

Gazebo simulator, greatly used by the robotics community, was initially thought to be the best simulation environment for testing the trajectory controllers with. It relies on well-known physics engines, natively supports the Husky model developed by Clearpath Robotics, and can interface with Simulink via ROS. After thorough analysis and repeated attempts, however, it was concluded that its outputs were far away from reality, an issue not easily mended. This section will give a brief overview of the tests and consequent unsatisfactory results, which contributed to the final decision to fully replicate the model within Simulink and abandon Gazebo. The considerations here presented are only valid for Gazebo 9 and the ROS distribution Melodic Morenia.

Gazebo 7 supports four physics engine: ODE, Bullet, Simbody and DART. However, only one of them is compatible with ROS: ODE, which, not so coincidentally, is the default engine and also the most outdated. The creator's website (`www.ode.org`) defines ODE as an open-source, high-performance library for simulating rigid body dynamics with applications ranging from games to 3D animation tools and industrial simulators. At first, the

only configuration deemed necessary was related to the ground-wheel friction coefficients, that had to replicate the BRI testing facility conditions. Initial tests highlighted deeper problems, only solvable by thoroughly studying the user manual. This resulted in the selection of the parameters[13] whose tuning would have impacted the robot dynamics the most:

- ***iters*:** The number of iterations for each simulation step. A higher number produces greater accuracy at a performance cost.
- ***max_step_size*:** Maximum time step size at which every system in the simulation can interact with the states of the world.
- ***contact_surface_layer*:** The depth of the surface layer around all geometry objects. Contacts are allowed to sink into the surface layer up to the given depth before coming to rest.
- ***contact_max_correcting_velocity*:** The maximum correcting velocities allowed when resolving contacts.
- ***max_contacts*:** Maximum number of contacts allowed between two entities, in this case, wheels and ground.
- ***friction_model*:** The type of friction model used by ODE, it can be chosen between the pyramid, box or cone model.
- ***cfm*:** Constrain force mixing parameter, used to smooth the contact between two surfaces.
- ***Friction Coefficients*:** Namely μ and μ_2, the first determines the friction force along the first direction of the pyramid model, determined by the parameter ***fdir1***, while the second along the direction perpendicular to ***fdir1***.

The parameters were tested by creating a MATLAB script, summarised in Algorithm 3.4, which allowed to effortlessly reconfigure Gazebo, to output a predefined set of commands and to post-process the data recorder by ROS.

Algorithm 3.4 MATLAB algorithm used to test Gazebo parameters.

Input:

Output:

1: Run a Linux script that opens Gazebo and executes all the launch le required

▷ *Continued in the next page*

[13] As the detailed explanation of each parameter would be outside the scope of this thesis, the reader is advised to refer to the on-line documentation, reachable through this link, for further information.

Algorithm 3.4 – *Continued from previous page*

Run a Linux script that resets the Husky pose and velocity
Get from the user the commands to test (u_c and ω_c)
for $i \leftarrow 1$, number of test cases **do**
    Execute the script that loads the parameters of the ith case
    Create a ROS publisher
    Create a ROS Service Client
    Execute the script that starts a rosbag
    for $j \leftarrow 1$, $t_{cmd} \times f_{cmd}$ **do**
        Create a ROS message with the required u_c
        Create a ROS message with the required ω_c
        Publish the commands via the ROS publisher
        Wait for the duration of the commands ($1/f_{cmd}$)
    end for
    Create a ROS message with $u_c = \omega_c = 0$
    Stop the rover by publishing the previous cmd
    Call the ROS Service Client that calls the */gazebo/get_physics_properties* service
    Parse its output to read the Gazebo configuration parameters
    Stop and save the rosbag
end for
Shut-down ROS
Close Gazebo

The scripts used to switch between physics pro le relied on the command: ***gz physics –profile [preset_name]*** where *preset_name* the field is related to the preset coded into the world file launched by the command ***husky_empty_world.launch*** or any other launch file within the *husky_gazebo* package. The XML world file, in turn, was structured as follows:

Listing 3.1: Gazebo world file example.

```
<?xml version="1.0" ?>
<sdf version='1.6'>
 <world name='default'>
  <model name='ground_plane'>
   <link name='link'>
    <collision name='collision'>
     <surface>
      <friction>
       <ode>
        <mu>0.250000</mu>
        <mu2>0.150000</mu2>
       </ode>
      </friction>
     </surface>
    </collision>
```

```
16      </model>
17
18      <physics name='preset1' default='1' type='ode'>
19       <max_step_size>...</max_step_size>
20       <real_time_update_rate>...</real_time_update_rate>
21       <max_contacts>...</max_contacts>
22       <ode>
23        <solver>
24         <type>quick</type>
25         <iters>...</iters>
26         <sor>...</sor>
27         <friction_model>...</friction_model>
28        </solver>
29        <constraints>
30         <contact_surface_layer>...</contact_surface_layer>
31         <contact_max_correcting_vel>...</contact_max_correcting_vel>
32         <cfm>...</cfm>
33         <erp>...</erp>
34        </constraints>
35       </ode>
36      </physics>
37
38      <physics name='preset2' default='0' type='ode'>...</physics>
39      <!--other physics presets-->
40    </world>
41  </sdf>
```

The the combination of parameters that better represented the experimental observations are shown in Table 3.16.

Table 3.16: ODE physics engine main parameters.

Parameters	**Values**	**Parameters**	**Values**
iters	400	*max_step_size*	0.001
contact_max_correcting_velocity	100	*contact_surface_layer*	0.001
μ	0.25	*max_contacts*	20.0
μ_2	0.15	*cfm*	0.0
friction_model	cone		

Despite many attempts, it was not possible to achieve satisfactory results. Figures 3.21 to 3.24 show the measured response of the rover to the same inputs used for test trajectory №3 defined in Table 3.7, which should produce a sine-wave in the XY-plane, preceded by a straight path. The tests were conducted using the parameters presented in Table 3.16 and the three runs presented differ for the friction model employed: respectively pyramid, box and cone model. The time axes should be taken as reference as they present some synchronisation problems between the devices used for record the rosbags and the one used for the simulation and post-processing of the data which couldn't be addressed at the time of the experiments.

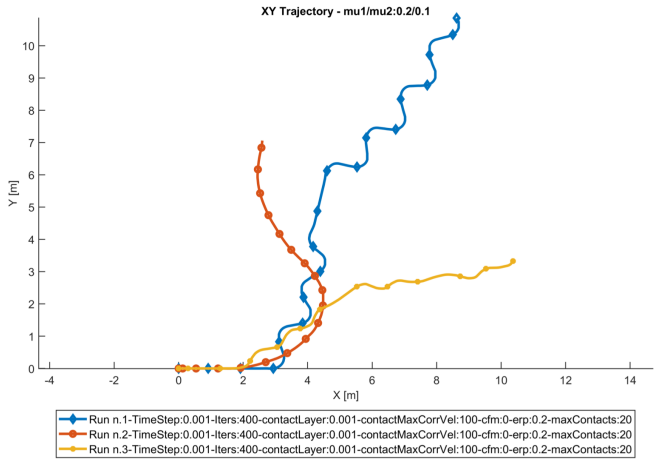
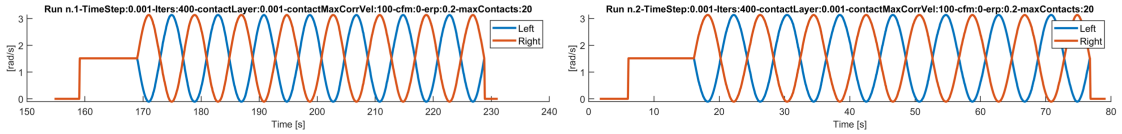
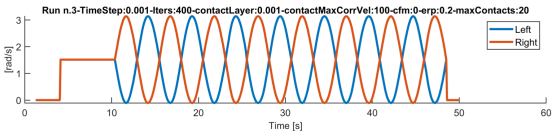


Figure 3.21: XY frame trajectory of the three runs.

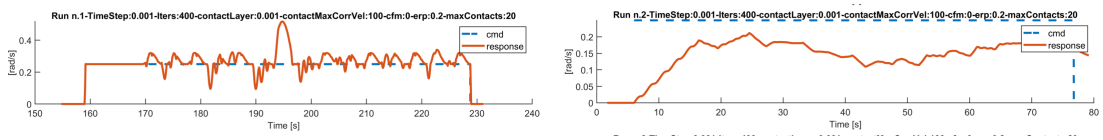


(a) Run1: pyramid model (b) Run2: box model.



(c) Run3: cone model.

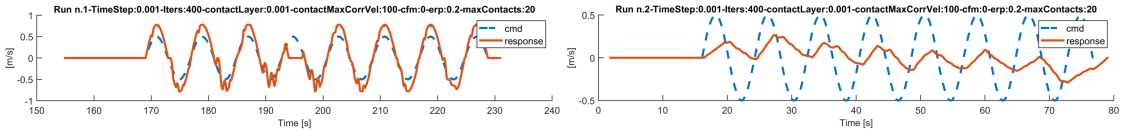
Figure 3.22: Husky's wheel speeds in Gazebo simulator.



(a) Run1: pyramid model (b) Run2: box model.

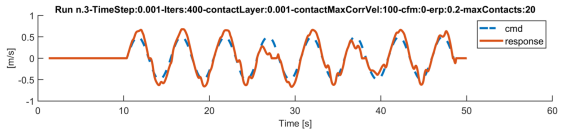
(c) Run3: cone model.

Figure 3.23: Husky's linear velocity in Gazebo simulator.



(a) Run1: pyramid model

(b) Run2: box model.



(c) Run3: cone model.

Figure 3.24: Husky's angular velocity in Gazebo simulator.

By analysing the responses, it is evident that:

- The three friction model, despite being approximations of the same physical description, give very different results.
- The box friction model is, by far, the worst. Nevertheless, the other two models produce very unreliable results, being far off from reality which was presented in the previous sections.
- At some point during the first and second runs, the simulated robot drives in a straight line even though the wheel speeds are identical in modulus and opposite in sign; behaviour which is not physically possible unless some external disturbances had arisen. This is clearly not the case of the simulation here reported.
- The angular velocity fluctuations, concerning the reference signal shown in Figure 3.24, follow an unusual pattern and are accompanied by sudden linear accelerations. It is possible to see that in this cases the linear velocity is greater than the reference speed of about $40 - 50\%$.

These phenomena are far from the real response witnessed during the preliminary tests at BRI. Not being able to solve such type of issues, it was concluded that replicating the mathematical model of the vehicle within the MATLAB/Simulink environment would have been a better choice.

Chapter 4

Trajectory Tracking Controllers

This chapter deals with the presentation and design of the control architectures developed within this research. After a brief initial introduction on the WMRs motion control and the reference trajectories generation, a PID and a SMC controllers are introduced, explaining their characteristics and the mathematical theories on which are based, and then developed in a trajectory tracking perspective, proposing where possible innovative approaches aiming to improve their performance or solve their characteristic problems. One of these, common to both architectures, is a heuristic approach to the gains tuning carried out using the jDE algorithm described in Appendix A. Them final implementation in a numerical simulation environment allows us to assess their tracking performance and then decide if are reliable enough to be implemented on the real Husky platform.

4.1 Motion Control Problem for WMR

Control problems involving mobile robots have attracted considerable attention in the control community, leading, in recent years, to the development of different solving methods that can be classified into the following three categories:

- **Sensor-based control:** This first category of controllers approaches the navigation problem emphasizing the autonomy of the vehicles deployed in dynamic unknown environments [92]. Since in this application the WMR has to traverse an unstructured environment, i.e. that can change over time, it must completely rely on its on-board sensors to cope with the constant changes in the surrounding dynamic environment. Most of the reported projects dealing with this approach are based on intelligent control schemes, such as fuzzy logic [93] and neural network [94]. Furthermore, to correctly plan the movement of the mobile robot, the controller must also be able to estimate the movement of obstacles and predict the configuration of the environment by means of sensory information from the external environment. However, since the rover to do this responds to its surroundings reactively or reflectively, the trajectory performed may not be globally optimized.

- **Path Planning and Execution:** Decomposing the navigation problem this two instances, a collision-free path is first generated based on a well-known map of the

environment in which the WMR must be deployed, and then executed by the robot that will have a priori knowledge of its surroundings. The planning procedures make use of certain optimization algorithms based on minimal time, minimal distance or minimal energy performance index, able to generate a collision-free path according to the environmental map space-time relations [95]. The mobile robot then must follow the planned path employing a path-following controller.

- **Trajectory Tracking:** The last category follows a motion control approach in which the desired trajectory has to be accurately tracked. With the term trajectory tracking, we mean the need for the vehicle to follow a time parametrized reference. This type of problem has been investigated for years and now is well understood for the fully-actuated system, for which can be found satisfactory solutions in the most non-linear advanced-control textbooks. Regarding under-actuated[1] vehicles, in whose category the skid-steering mobile robots are included, unfortunately, this problem is still a very active topic of research. Some solution have been proposed, ranging from the most common linearisation and feedback linearisation methods [96, 97] to the more advanced Lyapunov based control laws [98].

Concerning this thesis, we try to asses the trajectory tracking main control objective, i.e. the guidance of the state trajectories of the dynamical systems along a desired reference trajectory employing two different controller architectures based on a feedback action. In particular, the objective of both controllers is based on the zeroing of the tracking error, that is, considering a time-varying reference or desired trajectory $\boldsymbol{x}_d(t)$ and the controlled system's state trajectory $\boldsymbol{x}_r(t)$ both defined in the time interval $t_0 \leq t \leq t_{end}$, to minimize the relative error $\boldsymbol{e}(t) = \boldsymbol{x}_d(t) - \boldsymbol{x}_r(t)$ for each t by performing a control action both in speed and position. The closer the controlled state trajectory follows the reference one, the better the control target is achieved until an ideal case is reached characterized by the coincidence the two trajectories for every time instant.

By going into detail, lets consider that the state trajectories are expressed by a three element vector with components $(x,\ y,\ \vartheta)$, with reference to figure 4.1 it is possible to define the trajectory tracking error, according to the coordinate change which is the inverse formulation of 2.4, as the difference between the reference and the robot poses:

$$\boldsymbol{e} = \begin{bmatrix} x_e \\ y_e \\ \vartheta_e \end{bmatrix} = \begin{bmatrix} \cos\vartheta_r & \sin\vartheta_r & 0 \\ -\sin\vartheta_r & \cos\vartheta_r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d - x_r \\ y_d - y_r \\ \vartheta_d - \vartheta_r \end{bmatrix} \tag{4.1}$$

The aim of the trajectory tracking controller is to find the control input $\boldsymbol{u}$ that, under a random initial condition, makes the tracking error $\boldsymbol{e}$ bounded and:

$$\lim_{t\to\infty} \|\boldsymbol{e}\| = 0 \tag{4.2}$$

[1] With under-actuated, it is meant that class of vehicles that have fewer actuators than state variables to be tracked.

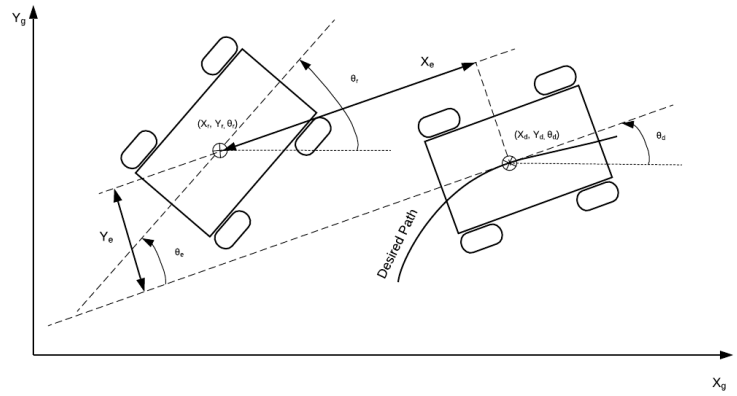


Figure 4.1: Representation of the Trajectory Tracking Problem.

4.2 Reference Trajectories Generation

A trajectory, as already mentioned, is a path containing an explicit function of time. Thus, being expressed as an equation in form $T = f(\boldsymbol{x},\ t)$, it is clear that for obtaining smooth movement, the trajectory must be twice differentiable to give a continuous velocity and acceleration profile. As a result, curve fitting techniques are an integral part of trajectory planning, such as the use linear splines, B-splines[2] or cubic splines [99].

A simple planning technique could be based on the assumption that the WMR is omnidirectional and able to flawlessly execute every trajectory. However, in the real applications, these assumptions are often not valid: planning a trajectory which disregards the robot constraints could have a serious impact on the ability of the robot to track the reference path [100]. Whereby for planning our feasible reference trajectory, dynamic and kinematic robot constrains has been considered such as upper-velocity bounds and non-holonomic constraint. The proposed trajectories are defined as follows:

1. A straight line at 45° given by the equations:

$$\begin{array}{llll} X & = v_{ref} t \cos\vartheta_d & \dot{X} & = v_{ref} \cos\vartheta_d \\ Y & = v_{ref} t \sin\vartheta_d & \dot{Y} & = v_{ref} \sin\vartheta_d \\ \vartheta_d & = 45^\circ & \dot{\vartheta}_d & = 0 \end{array} \tag{4.3}$$

   where v_{ref} is the constant linear reference velocity and t is the simulation time.

2. A circumference with radius r of 1.5 m given by the equations:

$$\begin{array}{llll} X & = r\cos\frac{v_{ref}}{r}t & \dot{X} & = -v_{ref}\sin\frac{v_{ref}}{r}t \\ Y & = r\sin\frac{v_{ref}}{r}t & \dot{Y} & = v_{ref}\cos\frac{v_{ref}}{r}t \\ \vartheta_d & = \frac{\pi}{2} + \frac{v_{ref}}{r}t & \dot{\vartheta}_d & = \frac{v_{ref}}{r} \end{array} \tag{4.4}$$

[2]Is a particular spline function that has minimal support with respect to a given degree, smoothness, and domain partition.

3. A sinusoidal trajectory with initial straight segment at 0° generated with the equations:

   - if $t \leq 1.5$ s

   $$\begin{aligned} X &= v_{ref} t \\ Y &= 0 \\ \vartheta_d &= 0^\circ \end{aligned} \tag{4.5}$$

   - if $t > 1.5$ s

   $$\begin{aligned} X &= v_{ref} t \\ Y &= 0.5 \sin\left(0.1\pi\left(t - 1.5\right)\right) \\ \vartheta_d &= atan2\left[\left(Y - Y_f\right), \left(X - X_f\right)\right] \end{aligned} \tag{4.6}$$

   where the desired orientation is obtained through the *MATLAB* function $atan2\left(Y,\ X\right)$ which returns the four-quadrant inverse tangent of Y and X which in turn are calculated as the difference between the current position and that at the previous integration step; and the desired velocity is obtained for simplicity deriving the position signal via the Simulink derivative block.

4. A complex trajectory composed by different turns generated with $\mathcal{C}^2$ continuous piecewise Bézier curves starting from a $Theta^\star$ path [95]. The two trajectory components $(X,\ Y)$ are loaded in MATLAB via an Excel sheet and by means of the lookup table Simulink block it is possible to obtain, by cubic interpolation, a reference point for any integration step so as to avoid discontinuity problems which lead to not negligible oscillations in position error. The third component, ϑ, of the reference trajectory and the desired velocity are also obtained this time through the *MATLAB* function $atan2\left(Y,\ X\right)$, which graphic representation is shown in Figure 4.2, and by deriving the position signal via the Simulink derivative block. This selection process can be described by the algorithm proposed in Algorithm 4.1, where the lines 13 to 16 describe the function that imposes the simulation stop when the last waypoint is reached.

Algorithm 4.1 Waypoint selection algorithm.

Input: Waypoint xlsx file, reference velocity
Output: Reference waypoint

1: Import a series of waypoints from a xlsx file
2: **if** the waypoints don't include a time coordinate **then**
3: Ask the user a reference velocity
4: **for** each waypoint **do**
5: Evaluate the time coordinate so that the average constant velocity required to move from the previous waypoint to the one considered is equal to the reference velocity input by the user

▷ *Continued in the next page*

Algorithm 4.1 – *Continued from previous page*

end for
end if
for each waypoint **do**
    Use $atan2$ function to evaluate the angle ϑ of the segment that connects the current waypoint to the next one
end for
while $stopCondition \leftarrow 1$ **do**
    $t \leftarrow$ simulation time
    if $t == t_{end}$ **then**
        Select the last waypoint as target
        $stopCondition = 1$
    else
        Find two waypoints $WP_n\,(X_n,\ Y_n,\ t_n)$ and $WP_{n+1}\,(X_{n+1},\ Y_{n+1},\ t_{n+1})$ so that $t_n \leq t \leq t_{n+1}$
        Linearly interpolate the $X,\ Y,\ t$ coordinate between the two selected waypoints
    end if
    Send the reference waypoint to the control system
end while

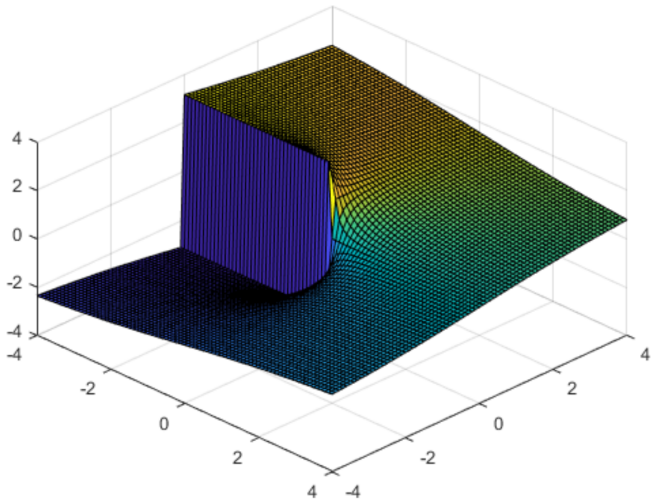


Figure 4.2: Matlab $atan2$ function graphic representation.[3]

[3]Image taken from the MathWorks official website.

4.3 PID Control

The Proportional Integrative Derivative control technique was first developed by Minorsky [101] in the early Twenties for a maritime application to automatic ship steering but, its greatest success is due to its mechanical implementation in the electronic analogue controllers, making this control strategy very successful in the industrial field. Furthermore, its mathematical simplicity and straightforward implementation allowed this old approach to gain great popularity, and to keep it up to the present day, also for applications different than industrial ones, ranging from aviation to robotics.

4.3.1 Definitions and Preliminaries

The PID is based on a feedback architecture in which its underlying principle is to regulate the signal error $e(t)$ between the measured and the desired state using appropriate gains that modify the control action $u_c(t)$ according to the error signal proprieties. The basic controller implementation and its tuning starting point is the introduction in the control loop of a proportional gain K_P which modify the control action in:

$$u_c(t) = K_P e(t) \tag{4.7}$$

The only proportional action is not always sufficient as it can lead to the emergence of a steady-state error with respect to a constant reference signal or to the susceptibility of the system to external disturbances. Moreover, an excessively increase in the gain value, in attempt to reduce the generated error or the system's rising time, can lead to the creation of instability, in particular for high-order systems.

To overcome these limitations, in support of the proportional action, an integral gain K_I can be added. Its main task is to cancel the steady-state error generating a contribution proportional to the time evolution of the integral of the signal error:

$$u_c(t) = K_I \int_0^t e(\tau)\, d\tau \tag{4.8}$$

Since the integral term took into account the time evolution of the error, it is possible to have a control action different from zero also in presence of a null signal error. The introduction of the integral contribution improves the steady-state response tracking but, an excessive increase in the K_I value leads to a slower response for equal overshoot, or to a rise in overshoot for the same response velocity.

Finally, the last contribution can be added by a derivative term K_D which, tanks to its anticipatory behaviour increases the damping of the system response ad of consequence its stability. The derivative gain contribution to the total control action is proportional to the rate of change of the signal error:

$$u_c(t) = K_D \frac{de}{dt}(t) \tag{4.9}$$

The sum of these three terms is a linear combination of the signal error, its integral and its derivative over time, leading to the PID controller formulation in the time domain:

$$u_c(t) = K_P e(t) + K_I \int_0^t e(\tau)\, d\tau + K_D \frac{de}{dt}(t) \tag{4.10}$$

or in the frequency domain:

$$u_c(t) = K_P + \frac{K_I}{s} + K_D s \tag{4.11}$$

4.3.2 Implementation and Tuning with DE Algorithm

After the system kinematic and dynamic model are determined, and supposing that a feasible reference trajectory for the mobile robot is pre-specified by the planner described in Section 4.2, in the format of pose $\boldsymbol{q}_d = [x_d,\ y_d,\ \vartheta_d]^T$ and velocity $\boldsymbol{v}_d = [v_d,\ \omega_d]^T$, it is now possible to design a classical PID controller so that the robot will correctly track the desired trajectory. This section aims to design a stable controller that generates a command vector $\boldsymbol{u} = [u_c,\ \omega_c]^T$ in order to respect the assumptions made in chapter 2 on the form of the kinematic and dynamic eqs. (2.17) and (2.42); and also to respect, and therefore make possible its later implementation, the commands required by the physical Husky platform which, using the *cmd_ vel* topic, needs them in the linear and angular velocities format.

Starting from the trajectory tracking problem described in Section 4.1, consider the robot real position expressed as a three element vector with components $(x_r,\ y_r,\ \vartheta_r)$. With reference to figure 4.1, lets recall the expression of the error that will be useful for the mathematical formulation of the controller:

$$\boldsymbol{e} = \begin{bmatrix} x_e \\ y_e \\ \vartheta_e \end{bmatrix} = \begin{bmatrix} \cos\vartheta_r & \sin\vartheta_r & 0 \\ -\sin\vartheta_r & \cos\vartheta_r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d - x_r \\ y_d - y_r \\ \vartheta_d - \vartheta_r \end{bmatrix} \tag{4.12}$$

The aim of the trajectory tracking controller, ass seen in eq. 4.2 is to drive towards zero the state trajectory error in finite time, generating a control action dependent on the state feedback for controlling both position and velocity of the mobile robot. To achieve this target has been thought the development of a multi-loop PI controller capable of taking into account both the kinematic and dynamic aspects of the WMR to obtain better motion control performance. The global control structure, with reference to Figure 4.3, is therefore decomposed into two stages:

1. *An outer loop*: that, depending on the robot kinematics, employs a kinematic-level control action used for controlling the vehicle pose $\boldsymbol{x}_r(t)$.
2. *An inner loop*: that, depending on the robot dynamics, employs a dynamic-level control action used for controlling the vehicle linear v_r and angular ω_r velocities.

Kinematic-level Control Model

Starting from the outer-loop, the controller reference variable, i.e. the pose error, must be defined. At first, it was considered a decoupled control for lateral and longitudinal branches, adopting the polar formulation of the tracking error defined as:

$$\boldsymbol{e}_\rho = \begin{bmatrix} \rho_e \\ \vartheta_e \end{bmatrix} = \begin{bmatrix} \sqrt{(X_d - X_r)^2 + (Y_d - Y_r)^2}\ sgn\left(\sqrt{(X_d - X_r)^2 + (Y_d - Y_r)^2}\cos\beta\right) \\ \vartheta_d - \vartheta_r \end{bmatrix} \tag{4.13}$$

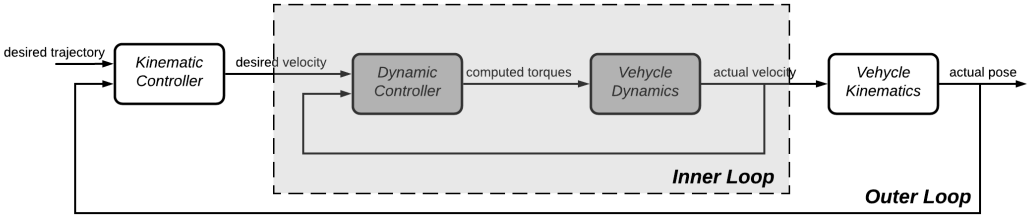


Figure 4.3: Proposed PID global control architecture.

where this particular ρ_e manipulation also takes into account the sign and β is yaw to reach the target position. But, after some initial tests, it was noticed that, being the control action decoupled, in the particular event of presence of an cross-track error, if the robot pose had been parallel to the reference trajectory, the controller would not have been able to recover that error. For this reason, a further control component that considers the cross-track error, defined as:

$$XTE = \sqrt{(X_d - X_r)^2 + (Y_d - Y_r)^2}\sin\beta \tag{4.14}$$

,has been added in the latero-directional branch remedying this problem. Adding now a *Proportional* and *Integral* term on each of these branches, we obtain the kinematic controller shown in Figure 4.4 characterized by the following mathematical formulation:

$$\boldsymbol{u}(t) = \begin{bmatrix} u(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} K_{P_\rho}\rho_e(t) + K_{I_\rho}\int_0^t \rho_e(\tau)\,d\tau \\ K_{P_\vartheta}\vartheta_e(t) + K_{P_{XTE}}XTE(t) + K_{I_\vartheta}\int_0^t \vartheta_e(\tau)\,d\tau + K_{I_{XTE}}\int_0^t XTE(\tau)\,d\tau \end{bmatrix} \tag{4.15}$$

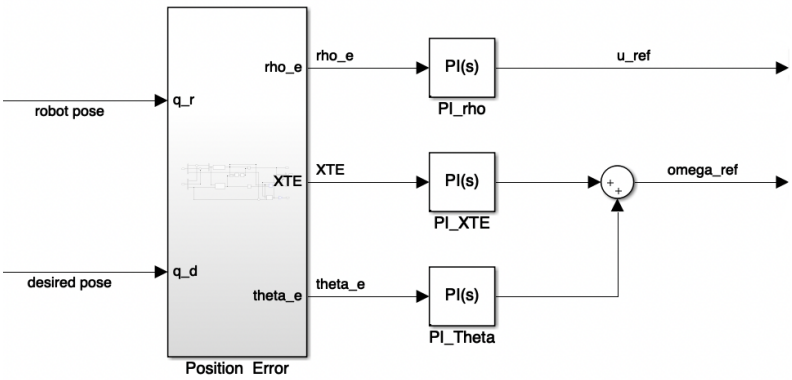


Figure 4.4: Kinematic controller block diagram representation.

Dynamic-level Control Model

Considering now the inner-loop, lets use as reference signal the outputs of eq.4.15: $\boldsymbol{u}_d(t) = \boldsymbol{u}(t)$. The state error is here simply represented as the difference between the reference and actual velocities:

$$\boldsymbol{e}_v = \begin{bmatrix} v_e \\ \omega_e \end{bmatrix} = \begin{bmatrix} u_d - u_r \\ \omega_d - \omega_r \end{bmatrix} \tag{4.16}$$

Adding now a *Proportional* and *Integral* term on the longitudinal and lateral branches, we obtain a first dynamic controller. For improving its responsiveness and velocity tracking performances, a feed forward component can be added in its architecture introducing the two gains FF_u and FF_ω obtaining the structure represented in Figure 4.5 defined by the equations:

$$\boldsymbol{u}_c(t) = \begin{bmatrix} u_c(t) \\ \omega_c(t) \end{bmatrix} = \begin{bmatrix} K_{P_u} u_e(t) + K_{I_u} \int_0^t u_e(\tau)\, d\tau + FF_u u_d \\ \\ K_{P_\omega} \omega_e(t) + K_{I_\omega} \int_0^t \omega_e(\tau)\, d\tau + FF_\omega \omega_d \end{bmatrix} \tag{4.17}$$

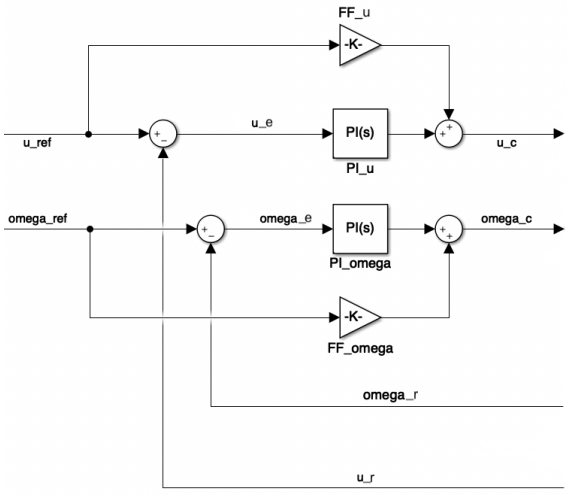


Figure 4.5: Dynamic controller block diagram representation.

Replacing eqs. (4.15) and (4.16) in eq. 4.17 the final control action, that represents the controller architecture shown in Fig. 4.6, can be obtained.

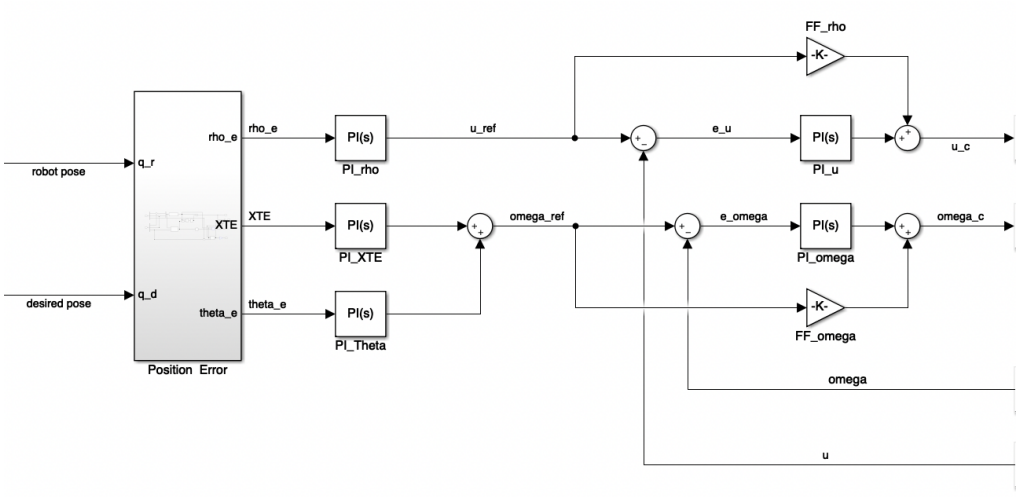
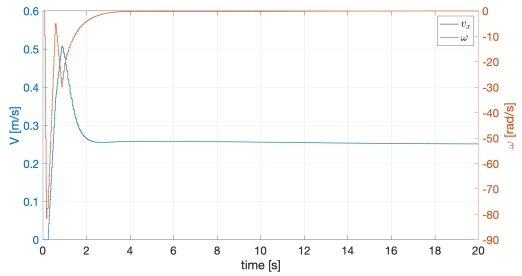


Figure 4.6: PID controller block diagram representation.

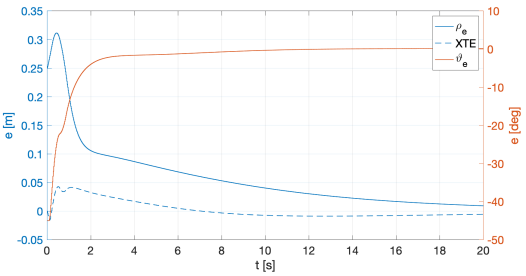
According to a first simulation on the tracking of a linear trajectory at the reference velocity of 0.25 m/s, with the robot having an initial cross-track error and different orientation compared to the first reference point, see fig. 4.7c, the tracking errors slowly converges to 0, as we can see from figure 4.7b. Even though the controller parameter have been determined roughly and manually at the values reported in table 4.1, these results demonstrated the functionality of the implemented controller, giving the green light for the optimization process.

Table 4.1: PID gains for the example trajectory in presence of initial pose error.

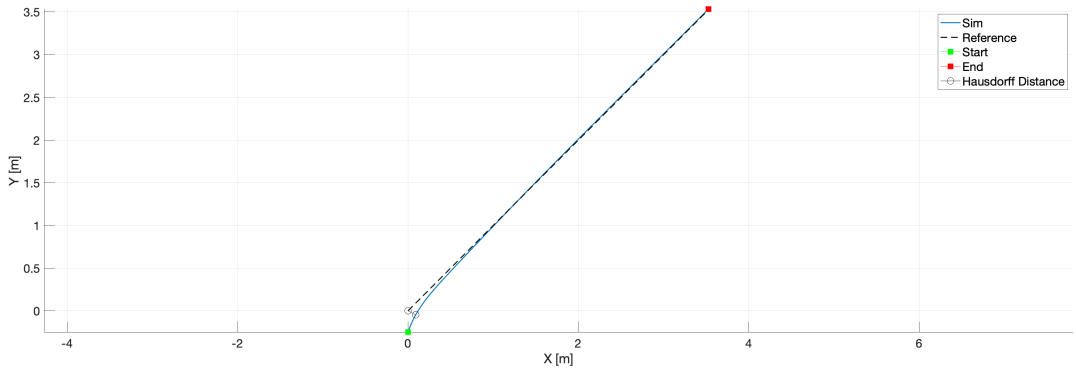
Controller	**Branch**	**Gain**	**Value**	**Gain**	**Value**
Position	**Linear**	$\boldsymbol{K_{P_\rho}}$	1.50	$\boldsymbol{K_{I_\rho}}$	0.20
		$\boldsymbol{K_{P_{XTE}}}$	5.00	$\boldsymbol{K_{I_{XTE}}}$	0.50
	Angular	$\boldsymbol{K_{P_\vartheta}}$	2.50	$\boldsymbol{K_{I_\vartheta}}$	0.0
Velocity	**Linear**	$\boldsymbol{K_{P_v}}$	0.05	$\boldsymbol{K_{I_v}}$	0.40
	Angular	$\boldsymbol{K_{P_\omega}}$	0.20	$\boldsymbol{K_{I_\omega}}$	0.20
Feed Forward	**Both**	$\boldsymbol{FF_\rho}$	0.90	$\boldsymbol{FF_\omega}$	0.50



(a) Linear and angular velocity evolution.



(b) Errors evolution.



(c) Travelled trajectory.

Figure 4.7: PID tracking results of a linear trajectory with initial pose error.

Controller Tuning and Optimization

Before further simulations can be performed, the controller's gain values must be refined. In this particular application, a manual trial and error tuning of gain parameters K_{Pi}, K_{Ii} and FF_i has to be performed, leading at the determination of 12 parameters. This can be a very time-consuming process and also them optimum tuning is not guaranteed. To overcome this difficulty, an automatic parameters tuning using the MATLAB script based on the DE algorithm, already used in Section 3.4 and described by Algorithms 3.1, 3.3 and A.1, has been employed. The operational explanation of the chosen algorithm, together with its pseudo code, is reported in Appendix A, as it is common for both of the controllers studied; while in this section, we limit to report the values of its control parameters and the results obtained in terms of controller gains and fitness function values. The controller then will be tested on the different trajectories presented in Section 4.2.

For running the jDE optimization over the selected mathematical model, not considering both external disturbances and sensor noise, after having decided its control parameters, it is necessary to define the search boundaries for each gain. To avoid too long convergence times, it is a good choice limiting the search area. For this reason, the bounds have been identified by a first trial and error approach in order to identify which were the most suitable ranges to also avoid the areas where they could have occurred

instability phenomena due to excessively high gains. Moreover, we have to define the cost function used to assess the fitness of the trial and target vector during the selection process, and consequently, the performances of the developed controllers. The cost function f_{opt} to be minimized has bees selected as the combination of the features of the evaluated trajectory derived from the metrics presented in Section 1.5:

$$f_{opt} = k_d d_{ratio} + k_{bend} e_{r_{bend}} + k_{haus} d_{haus} + k_{rms} d_{rms} \tag{4.18}$$

where $e_{r_{bend}}$ is defined as the ratio between the bending energy resulting from the simulation and the corresponding one of the reference path, both derived from Eq. 1.4, and where the constant values k_d, k_{bend}, k_{haus}, k_{rms} have been chosen, see Table 4.2, for guarantee a balanced effect of the evaluated features on the cost function.

Table 4.2: Objective function gains.

Parameters	**Values**	**Parameters**	**Values**
$\boldsymbol{k_d}$	10	$\boldsymbol{k_{haus}}$	15
$\boldsymbol{k_{bend}}$	10	$\boldsymbol{k_{rms}}$	100

By instructing the EA, adopting the configuration parameters reported in Table 4.3 which also contains the control parameters of the jDE algorithm already defined in Appendix A, to replicate the *complex trajectory* described in Section 4.2 used as test case, the values obtained through the optimization process and the corresponding optimization time, are shown in Table 4.4. It is worth noting that the feed-forward gains were set manually before starting the optimization process.

Table 4.3: Configuration parameters for non-linear model optimization.

Parameters	**Values**	**Parameters**	**Values**
F_{init}	0.75	CR_{init}	0.80
F_l	0.10	F_u	1.00
CR_{min}	0.00	CR_{max}	1.00
τ_1	0.10	τ_2	0.10
nr	4.0	ξ	0.20
df	0.001	dt	0.001
NP	100.0	NG	150.0

Table 4.4: PID optimized parameters.

Controller	Branch	Gain	Values	f_{opt}	Optimization time	Search bounds
Complex trajectory						
Position	**Linear**	K_{P_ρ}	2.8360	28.311	9.621×10^3	[0, 6]
		K_{I_ρ}	0.1146			[0, 2]
		$K_{P_{XTE}}$	5.9828			[0, 6]
		$K_{I_{XTE}}$	1.9708			[0, 2]
	Angular	K_{P_ϑ}	4.5073			[0, 6]
		K_{I_ϑ}	5.276×10^{-4}			[0, 2]
Velocity	**Linear**	K_{P_v}	0.0372			[0, 2]
		K_{I_v}	0.4769			[0, 2]
	Angular	K_{P_ω}	0.6498			[0, 2]
		K_{I_ω}	0.5481			[0, 2]
Feed Forward	**Linear**	FF_ρ	1.0			*constant*
	Angular	FF_ω	0.0			*constant*

4.3.3 Simulation Results

In this subsection, the simulation results of the PID controller are presented. Since the simulations are run in Simulink, the equations describing the trajectory planner, see Section 4.2, are transformed into their block diagram representation and cascaded to the PID and mobile robot with advanced friction ones described respectively in Sections 3.3.2 and 4.3.2, obtaining the model block diagram shown in Figure 4.8.

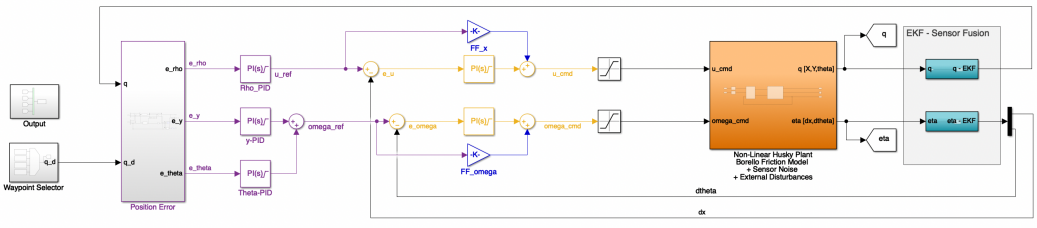
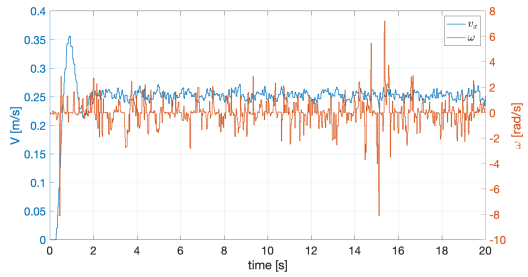


Figure 4.8: PID simulation model block diagram.

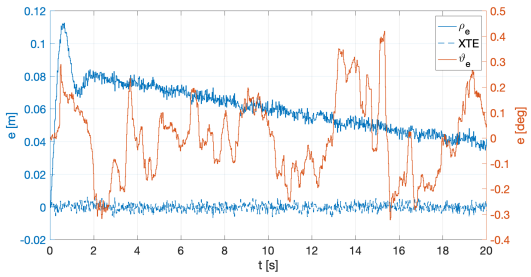
In particular, the waypoint selector block consist of a variant subsystem, containing the planners of the 4 type trajectories, which output the reference trajectory selected

by the user. The error block generates the tracking error by subtracting the robot pose from the reference one; the resulting orientation is given in the range $(-\pi,\ \pi)$. The next section is composed by the purple, yellow and blue blocks that together make the PID controller, which takes the tracking error as reference and generates the control action based on equation 4.17. Once the control signal enters the mobile robot plant, it is first transformed by the wheel controller into the desired torque by eq. 2.46 so it can enter as input in the dynamic plant, containing the enhanced friction model and some saturation blocks for limiting the generated torques and velocities at them maximum values reported in table 3.2, which obtains the actual rover pose, used as feedback, by dead-reckoning. The last blocks are used for the sensor noise generation, based on a white noise model.

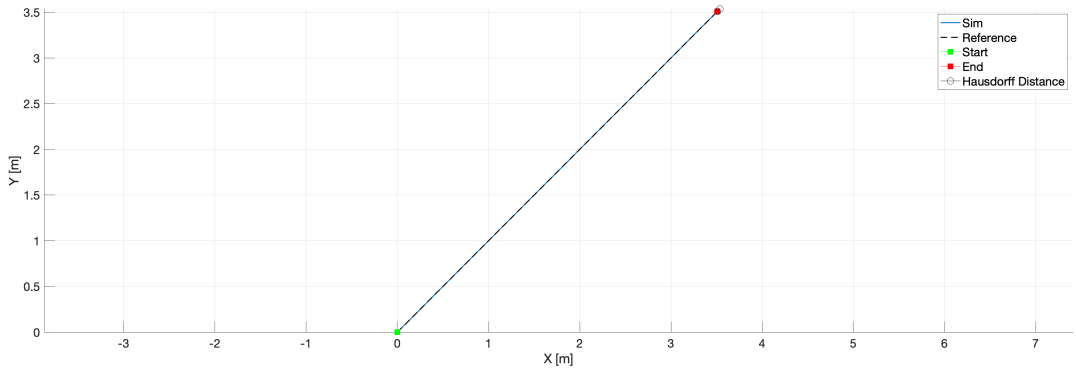
The control model is now simulated for all the trajectories described before, without any initial position error and at a reference velocity of 0.25 m/s, using the gains obtained from the first optimization process carried out on the test trajectory and reported in table 4.4. It is worth specifying that the other three trajectories were used only to assess the controller performances in different scenarios. The obtained result are represented in Figures 4.9 to 4.12 and the respective metric values are summarized in Table 4.5.



(a) Linear and angular velocities evolution.

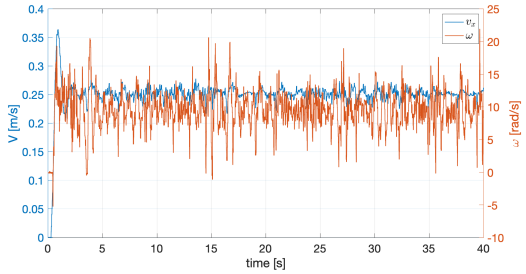


(b) Errors evolution.

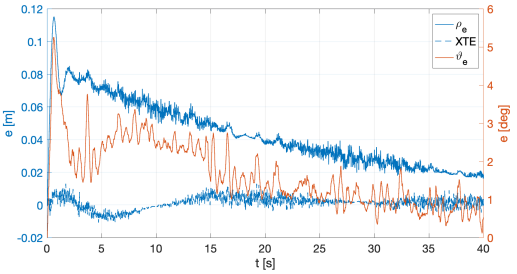


(c) Travelled trajectory.

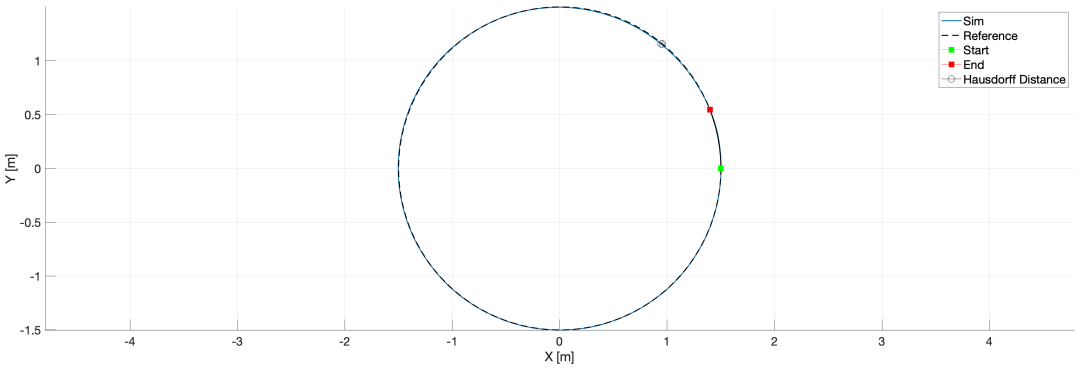
Figure 4.9: PID simulation tracking results of a linear trajectory.



(a) Linear and angular velocities evolution.

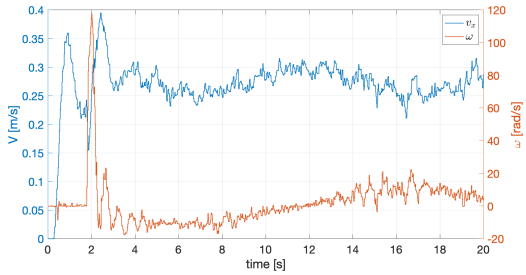


(b) Errors evolution.

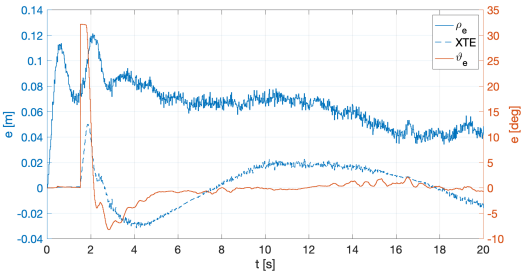


(c) Travelled trajectory.

Figure 4.10: PID simulation tracking results of a circumferential trajectory.

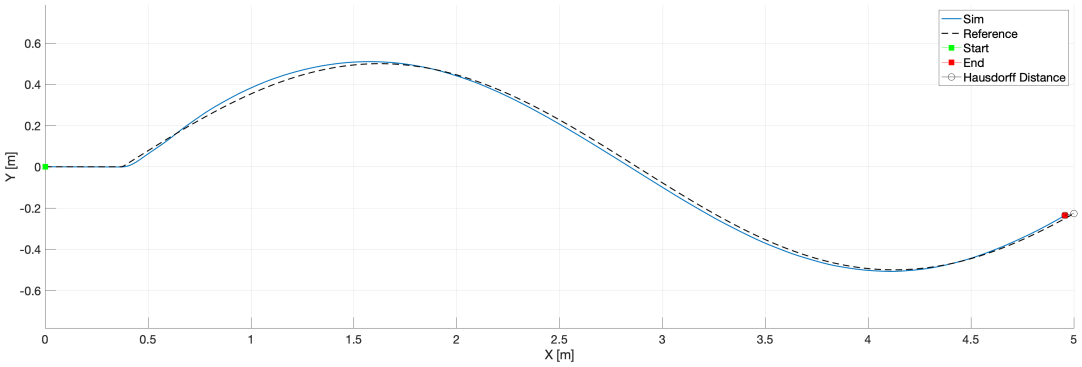


(a) Linear and angular velocities evolution.



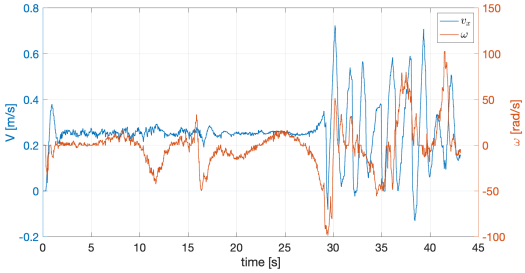
(b) Errors evolution.

Figure 4.11: PID simulation tracking results of a sinusoidal trajectory. (*continued*)

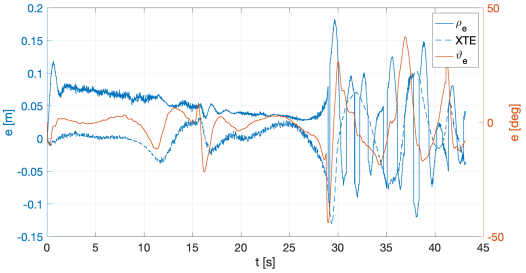


(c) Travelled trajectory.

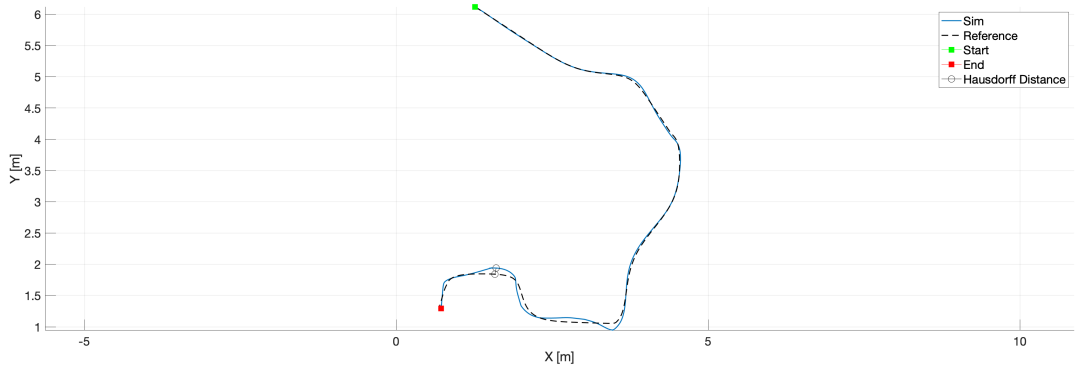
Figure 4.11 – *Continued from previous page.*



(a) Linear and angular velocities evolution.



(b) Errors evolution.



(c) Travelled trajectory.

Figure 4.12: PID simulation tracking results of a complex trajectory.

Table 4.5: PID trajectory tracking numerical simulations results.

Trajectory	d_{ratio}	d_{haus} [m]	d_{rms} [m]
Straight line	0.992	0.038	0.004
Circumference	0.999	0.013	0.006
Sinusoidal trajectory	0.996	0.045	0.015
Complex trajectory	1.034	0.101	0.034

Starting from the linear trajectory, which represents the least demanding case for the controller, it can be seen from figure 4.9a that after a settling period of about 2 second, the actual outputs perfectly tracked the desired linear velocity remaining in an acceptable range.

Interesting considerations can be made regarding errors evolution of the first three trajectories, respectively represented in Figures 4.9b, 4.10b and 4.11b, in particular regarding ρ_e which shows a convergent trend but a very slow zeroing. This is strictly dependent on the value of the integrative gain, which in this case is very low to avoid the onset of unstable responses.

Analogizing the simulation of the complex trajectory reported in 4.12, in correspondence with the more accentuated turns, high oscillations arise. This is due to the sudden increases in tracking errors. Despite this behaviour, the controller can quickly drive back the state on the tracked trajectory generating a maximum deviation of 0.101 meters, quite high for this low speed. Furthermore, the angular speed profile in Figure 4.12a seems to have quite high peaks in correspondence with the most accentuated turns, this could generate problems during the experimentation on the real Husky platform.

With regard to the metrics values shown in Table 4.5, it is worth to mention that the bending energy ratio has not been considered this time due to faulty results given by the implement of sensor noise.

4.4 Sliding Mode Control

Sliding mode control is a type of variable structure control system which control logic switches between two opposed control laws, based on the location of the state trajectory, to achieve the desired performances of the system under analysis. The control architecture relies on a switching function to chose between the two control laws, which drives the state trajectory towards a sliding surface[4] that, once is reached, traps the state trajectory and forces it to move towards the equilibrium point of the state by a *sliding movement*. Hence the name sliding mode control.

The most significant advantage of this class of controllers is its robustness, due to

[4]When the dimension of the sliding surface is greater than two, it is also called sliding manifold or hypersurface.

its shifting control structure, which makes it entirely insensitive for those uncertainties implicit in the input channel, called *matched uncertainty*, during the sliding motion [78]. Besides, once the state trajectory reaches the sliding surface, the order of the system can be considered reduced, simplifying the complexity of the system and, under suitable conditions, the controller can deal with systems in the presence of unmatched uncertainty [102]. However, the main drawback of sliding mode control is represented by the *chattering* phenomenon: the high-frequency switching motion of the state trajectory between the two opposite control laws occurring in the sliding surface bounds. This phenomenon arises in the attempt of the switching function to trap the state trajectory within the sliding surface, which is switched at an infinite frequency. However, due to physical limitations in real-world systems, directly applying the above-developed control algorithms can lead to unwanted oscillations and to heat generation or wear off mechanical parts, eventually damaging them.

4.4.1 Definitions and Preliminaries

The sliding mode is a motion, characterized by the very high-frequency switching of the control logic, which took place in any control system with a discontinuous control function or in any dynamic system presenting discontinuities in the equation of motion. To provide a clear about the key design techniques of SMC and to minimize confusion, the discussion and the further examples concentrate only on linear systems or at least on systems which are linear in the control variables.

Considering a general system characterized by the following equation:

$$\dot{x} = f(x, u, t) \tag{4.19}$$

where, $x \in \mathbb{R}^n$ and the control $u(x,t) \in \mathbb{R}^m$ with its respective components $u_i(x,t)$ has the form:

$$u_i(x,t) = \begin{cases} u_i^+(x,t), \text{ when } s_i(x) > 0 \\ u_i^-(x,t), \text{ when } s_i(x) < 0 \end{cases}, \text{ for } i = 1, \ldots, m \tag{4.20}$$

where $u_i^+(x,t)$ and $u_i^-(x,t)$ are continuous functions, some basic definitions are given as follows:

- **Switching surface:** Let $s_i(x)$ be a continuous $(n-1)$-dimensional *switching function*; since the control entries $u_i(x,t)$ undergoes discontinuity on the surface identified by the aforementioned function, $s_i(x) = 0$ is called a *switching surface* or *switching hyperplane*.

- **Sliding mode:** Let $S = \{x \ : \ s(x) = 0\}$ be a switching surface that includes the origin $x = 0$; if, for any initial state $x_0 \in S$, the condition $x(t) \in S$ is verified at every $t > t_0$, then $x(t)$ is a *sliding mode* of the system. In general, a sliding mode exists if in the vicinity of the switching surface S, the velocity vectors of the state trajectory always point towards the switching surface.

- **Sliding surface:** If a sliding mode exist on the switching surface $S = \{x \ : \ s(x) = 0\}$, i.e., if exist a trajectory, for every point in the surface S, able to reach the surface

from both sides, then the switching surface S is called *sliding surface* or *sliding manifold* depending on its dimensions.

- **Reaching condition:** The existence of a sliding mode requires the stability of the state trajectory towards the sliding surface $S = \{x \ : \ s(x) = 0\}$ at least in its neighbourhood, in other words the state trajectory must approach the surface at least asymptotically.
- **Region of attraction:** Represents the largest neighbourhood of the sliding surface $S = \{x \ : \ s(x) = 0\}$ in which the reaching condition is satisfied.
- **Reaching mode:** Represents the phase in which the state trajectory undergoes the reaching condition.

From the definitions above, it is shown that for an n^{th}-order system with m inputs there are m switching functions and $2^m - 1$ switching surfaces of different dimensions [103]:

1. m surfaces of dimension $(n-1)$defined as:

$$S_i = \{\boldsymbol{x} \ : \ s(\boldsymbol{x}) = 0\}, \quad \text{for} \ \ i = 1, \ldots, m \tag{4.21}$$

2. Considering the intersection of two surfaces S_i and S_j which generates an $(n-2)$-dimensional switching surface, the total number of such intersections equals the number of combinations of m surfaces S_i taken two at a time:

$$\binom{m}{2} = \frac{m(m-1)}{2} \tag{4.22}$$

   These surfaces can be described as:

$$S_{ij} = S_i \cap S_j, \quad \text{for} \ \ i, j = 1, \ldots, m \text{ with } i < j \tag{4.23}$$

   Figure 4.13 shows a generic interpretation of two planes S_1 and S_2 which intersection is the switching surface S_{12} represented by a line.

3. Further intersections involving multiple surfaces S_i, can be described in the same manner as in 4.23; so the intersection of k surfaces generates $\binom{m}{k}$ switching surfaces of dimension $(n-k)$.

4. There is finally a single switching surface S_E of dimension $(n-m)$ which is the intersection of the all surfaces S_i, with $i = 1, \ldots, m$, given by:

$$S_E = S_1 \cap \cdots \cap S_m \tag{4.24}$$

   This last surface can be called *eventual sliding surface*, to which the all trajectories must eventually reach.

Given this number of sliding surfaces, in such a system it is possible to have $2^m - 1$ different sliding modes which can actually begin in a number of different ways, hereafter referred to as a *switching scheme*. The number of switching schemes that exist depends on the order of entering different sliding modes, three of them are described briefly here [104]:

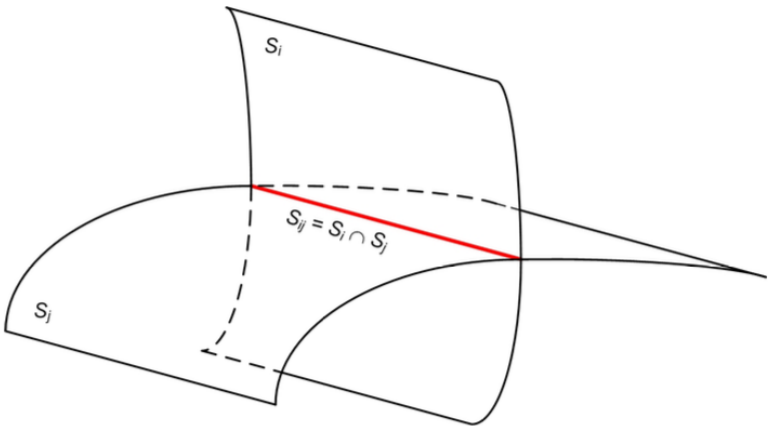


Figure 4.13: Geometric representation of two intersecting switching surfaces.

- **Fixed-Order Sliding Mode Switching:** In this scheme, as shown in Fig.4.14a, sliding modes take place in a preassigned order while the state is traversing the state space. For example, considering the initial state $\boldsymbol{x}_0$, it can move into the switching surface S_1 of dimension $(n-1)$, to the surface $S_{12} = S_1 \cap S_2$, which has dimension $(n-2)$ and so on, moving to progressively lower-dimensional sliding surfaces and eventually reaching the $(n-m)$-dimensional surface S_E:

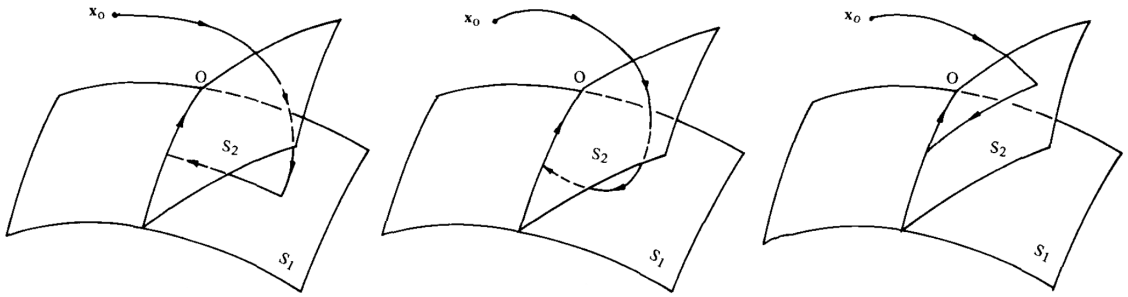
$$\boldsymbol{x}_0 \rightarrow S_1 \rightarrow (S_1 \cap S_2) \rightarrow ... \rightarrow S_E \tag{4.25}$$

  Although it is conceptually simple, the associated control system has a long transient period and the work needed for determining it is tedious and time-consuming.

- **Eventual Sliding Mode Switching:** In this scheme, as shown in Fig. 4.14b, the state is driven from any initial state $\boldsymbol{x}_0$ to the eventual sliding surface S_E on which the sliding mode takes place. On the other switching surfaces sliding modes there may or may not be obtained. This scheme is simple to implement and a smooth control can be easily obtained; however the switching scheme does not guarantee good transient characteristics.

- **Free-Order Sliding Mode Switching:** Here, the order of sliding modes are not preassigned but follows its natural evolution in a *"first-reach-first-switch"* scheme in which the state is driven to the eventual sliding surface S_E, as shown in Fig. 4.14c. This scheme, compared to the fixed-order approach, is more simple to implement, the reaching mode possesses better dynamic characteristics and the resulting control effort is smaller in magnitude. Hence, saturation is less likely to occur.

After having defined these basic concepts, it is now possible to present the most commonly used design procedures for this type of controllers. Generally, the SMC design consists of two steps: first, the sliding surface must be defined so that the design objectives are satisfied by the system performance during the sliding mode. Second, the switched control logic is designed for satisfying the reaching condition and thus ensuring



(a) Fixed-order. (b) Eventual. (c) Free-order.

Figure 4.14: Switching schemes.

the finite-time convergence of the state trajectory to the sliding surface, maintaining it there thereafter. In other word $s(x) = 0 \; \forall \; t \geq t_s$ where t_s is the time needed to the state trajectories for reaching the sliding surface.

Sliding Surface Design

The design of a sliding surface can be done following different approaches thanks to the great effort that has been put in this research topic, especially regarding the linear systems. Considering the following generic non-linear system:

$$\dot{x} = Ax + Bu \tag{4.26}$$

where $x \in \mathbb{R}^n$ and the control $u(x,t) \in \mathbb{R}^m$, and its generic sliding surface:

$$S = \{x \; : \; s(x) = 0\} \tag{4.27}$$

some of these theories are summarized as follows:

- **Canonic form:** In presence of a linear single input system, after writing it in its canonic form:

$$\begin{aligned} \dot{x}_i &= x_{i+1} \text{ for } i = 1, \ldots, n-1 \\ \dot{x}_n &= \sum_{i=1}^{n} a_i x_i + bu \end{aligned} \tag{4.28}$$

  it is possible to define the sliding surface as a linear combination of the states [105]:

$$s(x) = \Lambda x = \sum_{i=1}^{n-1} \lambda_i x_i + x_n = 0 \tag{4.29}$$

  where the λ_i coefficients of the switching function define the desired characteristics of the closed loop system after the reaching phase.

- **Coordinate transformation:** Considering a linear system written as 4.26 and supposing the existence of a non-singular transformation matrix Q:

$$QB = \begin{bmatrix} 0 \\ B_2 \end{bmatrix} \tag{4.30}$$

where $B_2 \in \mathbb{R}^{m \times m}$ is a non-singular matrix, it is possible possible to transform the system in:

$$\begin{aligned} \dot{x}_1 &= A_{11}x_1 + A_{12}x_2 \\ \dot{x}_2 &= A_{21}x_1 + A_{22}x_2 + B_2u \end{aligned} \tag{4.31}$$

where, $x_1 \in \mathbb{R}^{n-m}$, $x_2 \in \mathbb{R}^m$. Writing the switching function as [106]:

$$s\left(x\right) = \Lambda_1 x_1 + \Lambda_2 x_2 \tag{4.32}$$

and assuming Λ_2 non-singular, during the sliding mode $\Lambda_1 x_1 + \Lambda_2 x_2$ it is possible to linearly relate x_2 to x_1 so the system can be transformed in:

$$\begin{aligned} \dot{x}_1 &= A_{11}x_1 + A_{12}x_2 \\ \dot{x}_2 &= -Kx_1 \end{aligned} \tag{4.33}$$

where $K = \frac{\Lambda_1}{\Lambda_2}$ represents an $(n-m)^{th}$ order system, in which x_2 is considered as the control variable, which sliding dynamic described by:

$$\dot{x}_1 = \left(A_{11} - A_{12}K\right) x_2 \tag{4.34}$$

Following this procedure we have changed to a reduced order state feedback problem which can be solved with classical approaches, like linear quadratic methods, for determining of the value of K that satisfies our system requirements. Once K is determined, the switching function, and the associated sliding surface, takes the form:

$$s\left(x\right) = \Lambda x = \Lambda_2 \left[K,\ I\right] x \tag{4.35}$$

- **Linear Quadratic (LQ) approach:** Represents one possible improvement of the Coordinate transformation, based on the optimal choice of the value of K by minimising a quadratic cost function over infinite time interval [106]. For example, considering the problem defined by eqs. 4.33, since x_2 can be considered as the control input, it is possible to find the optimal sliding mode of the system by a LQ optimization, minimising the cost function:

$$J = \int_{t_s}^{\infty} \left(x_1^T Q_{11} x_1 + 2x_1^T Q_{12} x_2 + x_2 Q_{22} x_2\right) dt \tag{4.36}$$

For sake of simplicity, considering $Q_{11} = 0$, without loss of generality it is possible to write the optimal control x_2 as:

$$x_2 = -Q_{22}^{-1} A_{12}^T P x_1 = -Kx_1 \tag{4.37}$$

where P is a positive defined matrix solution of the Ricatti equation:

$$A_{11}^T P + P A_{11} - P A_{12} Q_{22}^{-1} A_{12}^T P = -Q_{11} \tag{4.38}$$

Considering $\Lambda_2 = I$ then the switching function 4.35, and its related sliding surface, can be written as:

$$s(x) = K x_1 + x_2 = \left[Q_{22}^{-1} A_{12}^T P,\ I \right] x \tag{4.39}$$

- **Equivalent control method:** This approach introduces the condition $\dot{s}(x) = 0$ as a further requirement for maintaining the state trajectory on the sliding surface [107]. Assuming that a sliding mode exist on the generic surface 4.27, we can find a continuous control such that under the initial condition of the state vector on this surface, it yields to zero the time derivative of the switching function along the state trajectories:

$$\dot{s}(x) = G\dot{x} = GAx + GBu = 0 \tag{4.40}$$

where the components of the matrix $G \in \mathbb{R}^{m \times n}$ represent the gradients $\frac{\partial s}{\partial x}$ of the switching function.

Assuming the existence of a solution for the system 4.40, it is possible solving for the equivalent control u_{eq} expressed as:

$$u_{eq} = -\frac{\partial s}{\partial x} A x \left(\frac{\partial s}{\partial x} B \right)^{-1} \tag{4.41}$$

where $\frac{\partial s}{\partial x} B$ is non singular.

Substituting the the equivalent control in the system 4.26, its dynamic can be written as:

$$\dot{x} = \left[I - B \frac{\partial s}{\partial x} \left(\frac{\partial s}{\partial x} B \right)^{-1} \right] A x \tag{4.42}$$

and its relative sliding surface as:

$$s(x) = Gx = 0 \tag{4.43}$$

- **Time varying surface for tracking control:** According to [107], one possible approach for designing the sliding surface of a single input system is going through the definition of the desired control bandwidth, considering a tracking problem and expressing the state trajectory x as the tracking error for control purposes, a sliding surface can be written as:

$$s(x, t) = \left(\frac{d}{dt} - \lambda \right)^{n-1} x = 0 \tag{4.44}$$

where λ is a positive constant determining the closed-loop bandwidth.

These presented are just some possible methods, others for developing linear or non-linear sliding surfaces can be found in literature, such as [108] where a frequency-shaped sliding surface, which appears as a linear operator, was designed to attenuate high frequency components in order to avoid vibrations given to the interaction between sliding mode and unmodelled system dynamics, or [109, 110] where a robust sliding hyperplanes was designed respectively via a Ricatti and Lyapunov approach.

Control Law Design

After finding a suitable sliding surface, the reachability problem must be analysed to design a control function $u : \mathbb{R}^n \to \mathbb{R}^m$ able to drive the state trajectory x towards the sliding surface and to maintain it herein. To do this, it is first necessary to define a reaching law and then it is possible to derive the control method. Some different approaches are here proposed [103]:

1. ***Reaching laws*:** The following reaching laws are valid for both SISO and MIMO systems:

   - **Direct switching function approach:** A sufficient reaching condition for make sliding mode appear is:

     $$s_i \dot{s}_i < 0 \text{ , for } i = 1, \ldots, m \tag{4.45}$$

     Even if this reaching condition is global, it do not guarantee finite reaching time. A similar sufficient condition was also proposed in [105]:

     $$\lim_{s_i \to 0^+} \dot{s}_i < 0 \quad \text{and} \quad \lim_{s_i \to 0^-} \dot{s}_i > 0 \tag{4.46}$$

     These reaching laws generates a control structure in which the individual switching surfaces and their intersection are all sliding surfaces but results but it is very difficult to implement for the multiple-input systems.

   - **Lyapunov function approach:** Choosing the Lyapunov function candidate:

     $$\mathcal{V}(\boldsymbol{x}, t) = \boldsymbol{s}^T \boldsymbol{s} \tag{4.47}$$

     a global reaching condition is given by:

     $$\dot{\mathcal{V}}(\boldsymbol{x}, t) < 0 \tag{4.48}$$

     This approach results in a control structure in which the sliding mode is guaranteed only on the intersection of the all sliding surfaces leading to the so called *eventual sliding mode*. The major drawback is that the points lying on the individuals switching surfaces s_i could not belong to the sliding surface. Furthermore finite reaching time is not guaranteed, but this problem is easily fixable by modifying 4.48 in:

     $$\dot{\mathcal{V}}(\boldsymbol{x}, t) < -\varepsilon \tag{4.49}$$

     where ε is a strictly positive value.

   - **Gao's reaching law approach:** Gao proposed in [111] a method for expressing the reaching law by differential equation, which parameters can define the dynamic qualities of the control system during the reaching mode. A practical general form of the proposed reaching law is:

     $$\dot{\boldsymbol{s}} = -Q sgn(\boldsymbol{s}) - P\boldsymbol{h}(\boldsymbol{s}) \tag{4.50}$$

where:

$$\begin{aligned} Q &= diag\left[q_1, \ldots, q_m\right], \quad \text{where} \quad q_i > 0 \quad \text{for} \quad i = 1, \ldots, m \\ P &= diag\left[p_1, \ldots, p_m\right], \quad \text{where} \quad p_i > 0 \quad \text{for} \quad i = 1, \ldots, m \\ sgn\left(\boldsymbol{s}\right) &= \left[sgn\left(s_1\right), \ldots, sgn\left(s_m\right)\right]^T \\ \boldsymbol{h}\left(\boldsymbol{s}\right) &= \left[h_1\left(s_1\right), \ldots, h_m\left(s_m\right)\right]^T \end{aligned} \tag{4.51}$$

and

$$\begin{aligned} s_i h_i\left(s\right) &> 0 \\ h_i\left(0\right) &= 0 \end{aligned} \tag{4.52}$$

Four practical special cases of 4.50 are given below:

– *Constant rate reaching*:

$$\dot{\boldsymbol{s}} = -Qsgn\left(\boldsymbol{s}\right) \tag{4.53}$$

The switching function is forced to reach the switching surface S at the constant rate $|\dot{s}_i| = -q_i$. Its simplicity makes it a valid approach but some attention must be paid in the choice of q_i because if it is too small the reaching time will be to long and, on the contrary, a too large value t will cause severe chattering.

– *Constant plus proportional rate reaching*:

$$\dot{\boldsymbol{s}} = -Qsgn\left(\boldsymbol{s}\right) - P\boldsymbol{s} \tag{4.54}$$

Adding the proportional term $-P\boldsymbol{s}$ at the previous formulation, it is possible to force the state to approach the switching surface faster when $\boldsymbol{s}$ is large. Furthermore, it can be shown that the reaching time for $\boldsymbol{x}$ to move from an initial state $\boldsymbol{x}_0$ to the switching manifold S_i is finite, and is given by:

$$T_i = \frac{1}{p_i} ln\left(\frac{p_i\left|s_i\right| + q_i}{q_i}\right), \quad \text{for} \quad i = 1, \ldots, m \tag{4.55}$$

– *Power rate reaching*:

$$\dot{s}_i = -p_i\left|s_i\right|^{\alpha} sgn\left(s_i\right), \quad \text{for} \quad i = 1, \ldots, m \tag{4.56}$$

where $0 < \alpha < 1$. By using this formulation, the reaching speed is increased when the state is far away from the switching surface, but reduced when it is in its proximity. The reaching time can be obtained integrating 4.56 from $s_i = s_{i0}$ to $s_i = 0$:

$$T_i = \frac{1}{\left(1-\alpha\right)p_i}\left|s_i\left(0\right)\right|^{\left(1-\alpha\right)}, \quad \text{for} \quad i = 1, \ldots, m \tag{4.57}$$

Thus the proposed law gives a finite time reaching and, in addition, the absence of the constant rate term $-Qsgn\left(\boldsymbol{s}\right)$ reduces the chattering.

– *Particular rate reaching*: An enhanced version of 4.56 based on a speed control relationship in the reaching phase was developed by Loh and Yeung in [112]:

$$\dot{s}_i = -p_i e^{\alpha|s_i|} sgn\left(s_i\right), \quad \text{for} \quad i = 1, \ldots, m \tag{4.58}$$

where the switching gain p_i and α are strictly positive values, and the reaching time becomes:

$$T_i = \frac{1}{\alpha p_i}\left(1 - e^{-\alpha|s_i(0)|}\right), \quad \text{for} \quad i = 1, \ldots, m \tag{4.59}$$

This approach provides a fastest reaching time and chattering reduction.

2. ***Control laws*:** After having selected the reaching equation, is now possible to determine the control law; some design processes are here presented:

- **Equivalent control augmentation method:** Even if we have already derived a control action u_{eq} based on equivalent control according to eq. 4.41, only using this equation if the initial condition $\boldsymbol{x}_0$ of the system is not on the sliding surface S the state cannot be driven towards S. One possible solution is to augment u_{eq} with a discontinuous or switched component u_N for satisfy the reaching condition obtaining:

$$u = u_{eq} + u_N \tag{4.60}$$

Combining eqs. (4.41) and (4.60) in 4.40, we obtain:

$$\dot{s}\left(x\right) = \frac{\partial s}{\partial x}\left[Ax + B\left(u_{eq} + u_N\right)\right] = \frac{\partial s}{\partial x} B u_N \tag{4.61}$$

assuming for sake of simplicity $\frac{\partial s}{\partial x} B = I$ then:

$$\dot{s}\left(x\right) = u_N \tag{4.62}$$

This condition allows an easy verification of the sufficiency conditions for the existence and reachability of a sliding mode: $s_i\left(\boldsymbol{x}\right)\dot{s}_i\left(\boldsymbol{x}\right) < 0$. Some structures of u_N are now presented below according to [113]:

– *Relay with gains*:

$$u_N = \begin{cases} -\alpha sgn\left(\boldsymbol{s}\left(\boldsymbol{x}\right)\right), & \text{if} \quad s \neq 0 \\ 0, & \text{if} \quad s = 0 \end{cases} \tag{4.63}$$

where α must be strictly greater than 0 and can be either a constant matrix or state dependent $\alpha\left(x\right)$. Can be noted that this controller meet the sufficiency condition for the existence of a sliding mode since each of its components u_{Ni} respect:

$$s_i\left(\boldsymbol{x}\right)\dot{s}_i\left(\boldsymbol{x}\right) = -\alpha s_i\left(\boldsymbol{x}\right) sgn\left(s_i\left(\boldsymbol{x}\right)\right) < 0, \quad \text{for} \quad s_i\left(\boldsymbol{x}\right) \neq 0 \tag{4.64}$$

– *Linear feedback with switching gains*:

$$u_{Ni} = \psi \boldsymbol{x} \tag{4.65}$$

where $\psi = [\psi_{ij}]$ defined as:

$$\psi_{ij} = \begin{cases} -\alpha_{ij}, & \text{if } s_i x_j > 0 \\ \beta_{ij}, & \text{if } s_i x_j < 0 \end{cases} \tag{4.66}$$

with $\alpha_{ij},\ \beta_{ij} > 0$. Again,the reaching condition is satisfied with:

$$s_i(\boldsymbol{x})\,\dot{s}_i(\boldsymbol{x}) = s_i(\boldsymbol{x})\,(\psi_{i1}x_1 + \cdots + \psi_{in}x_n) < 0 \tag{4.67}$$

– *Linear continuous feedback*:

$$u_{Ni} = -\alpha s_i(\boldsymbol{x}) \tag{4.68}$$

where α_i is a strictly positive value and the condition for the existence of a sliding mode is:

$$s_i(\boldsymbol{x})\,\dot{s}_i(\boldsymbol{x}) = -\alpha_i s_i^2(\boldsymbol{x}) < 0 \tag{4.69}$$

or more generally:

$$u_N = -L\boldsymbol{s}(\boldsymbol{x}) \tag{4.70}$$

where $L \in \mathbb{R}^{m\times m}$ is a strictly positive defined constant matrix and hence the reaching condition is:

$$\boldsymbol{s}^T(\boldsymbol{x})\,\dot{\boldsymbol{s}}(\boldsymbol{x}) = -\boldsymbol{s}^T(\boldsymbol{x})\,L\boldsymbol{s}(\boldsymbol{x}) < 0 \tag{4.71}$$

– *Univector non-linearity with scalar factor*:

$$u_N = -\frac{\boldsymbol{s}(\boldsymbol{x})}{\|\boldsymbol{s}(\boldsymbol{x})\|}\rho \tag{4.72}$$

where ρ is a strictly positive scalar. The existence condition is:

$$\boldsymbol{s}^T(\boldsymbol{x})\,\dot{\boldsymbol{s}}(\boldsymbol{x}) = -\|\boldsymbol{s}(\boldsymbol{x})\|\rho < 0 \tag{4.73}$$

- **Reaching law method:** By using the reaching law approach proposed in [111], the control can be directly obtained by computing $\boldsymbol{s}(\boldsymbol{x})$ time derivative along the reaching trajectory:

$$\dot{\boldsymbol{s}}(\boldsymbol{x}) = \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{x}}(A\boldsymbol{x} + Bu) = -Qsgn(\boldsymbol{s}) - P\boldsymbol{h}(\boldsymbol{s}) \tag{4.74}$$

thus, solving for u, the control action took the form:

$$u = -\left[\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{x}}A\boldsymbol{x} + Qsgn(\boldsymbol{s}) + P\boldsymbol{h}(\boldsymbol{s})\right]\left(\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{x}}Bu\right)^{-1} \tag{4.75}$$

The resulting sliding mode is not preassigned but follows the natural trajectory on a first-reach-first-switch scheme, so the switching takes place depending on the location of the initial state.

- **Control hierarchy method:** This approach is based on the direct switching function approach reaching law which sufficient condition for a sliding mode is $s_i \dot{s}_i < 0$, for $i = 1, \ldots, m$. The method establishes a control scheme in which the sliding modes take place in a preassigned order. Starting from the initial condition $\boldsymbol{x}_0$, the state trajectory moves progressively onto lower dimensional switching surfaces and eventually reaches the final sliding surface S_E:

$$\boldsymbol{x}_0 \rightarrow S_1 \rightarrow (S_1 \cap S_2) \rightarrow ... \rightarrow S_E \tag{4.76}$$

  The main drawback is that the control is determined by a set of complicated inequalities. For example, with reference to the system 4.26, the determination of the control u involves the solution of m pairs of inequalities:

$$\dot{s}_i(\boldsymbol{x}) = \frac{\partial s_i}{\partial \boldsymbol{x}}(A\boldsymbol{x} + Bu) = \begin{cases} > 0, & \text{if } s_i(\boldsymbol{x}) < 0 \\ < 0, & \text{if } s_i(\boldsymbol{x}) > 0 \end{cases}, \quad \text{for } i = 1, \ldots, m \tag{4.77}$$

  This process is very time consuming so this approach is seldom used.

4.4.2 Chattering Problem and its Reduction

As already mentioned, given the need for sliding mode controllers to request infinitely fast switching mechanisms, in order to guarantee the required closed-loop performances, an inevitable occurrence of oscillations in the proximity of the switching surface causes the main limitation of this type of control algorithms in real applications. This phenomenon is called *chattering* and is due, in agreement with Young [114], to two possible causes:

1. The presence of switching non-idealities, such as time delays or time constants, which characterize any implementation of switching devices, including both analogue and digital circuits, and microprocessor-based implementations as well.

2. The presence of parasitic dynamics, in series with the plant, may cause a small amplitude high-frequency oscillation to appear in the neighbourhood of the sliding surface. These dynamics are those of actuators, sensors and other high-frequency modes of the controlled process, which are often neglected in the open-loop model used for control design if the associated poles are well damped and outside the desired bandwidth of the feedback control system.

However, since the characteristic control signal discontinuity in sliding mode applications, the interactions between the parasitic dynamics and the switching action may cause non-dumped oscillations with finite amplitude and frequency which, if the switching gain is large, may lead to unpredictable system instability. Furthermore, other side effects in electronic or mechanical systems may appear due to heat generation or mechanical wearing. For this reasons, chattering effect is considered as a major obstacle for SMC to become a more appreciated control method and its reduction has been a major research objective. Some existing approaches for chattering reduction are now presented.

Boundary Layer SMC

As as the name specified by name, a boundary layer is introduced around the sliding surface. Inside that layer, as represented in fig. 4.15, the signum discontinuity in the switching function is replaced by a linear feedback gain, thus the control signal becomes continuous and chattering is avoided. As a consequence, ideal sliding mode is no longer taking place and state trajectories are confined to a vicinity of the sliding surfaces instead of exact tracking. One possible solution is the saturation function defined as:

$$sgn\left(\boldsymbol{s}\right) \approx sat\left(\frac{\boldsymbol{s}}{\varepsilon}\right) = \begin{cases} \frac{\boldsymbol{s}}{\varepsilon}, & \text{if } \left|\frac{\boldsymbol{s}}{\varepsilon}\right| \leq 1 \\ sgn\left(\frac{\boldsymbol{s}}{\varepsilon}\right), & \text{if } \left|\frac{\boldsymbol{s}}{\varepsilon}\right| > 1 \end{cases} \tag{4.78}$$

where the value ε represents the width of the area extending from the sliding surface where the state trajectory will be trapped in.

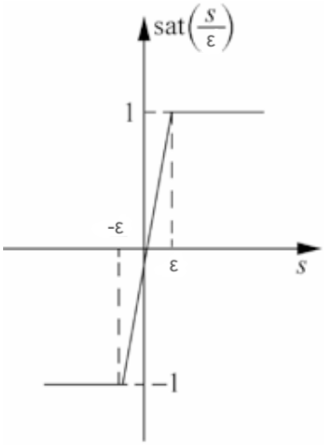


Figure 4.15: Graphic representation of saturation function.

The shortcoming of this approach is the loss of robustness inside the boundary layer, implying that uncertainties and parasitic dynamics must be carefully considered and modelled in order to avoid instability.

Quasi Sliding Mode

Similar to the boundary layer method, in order to reduce chattering, the approximation is done by means of a continuous sigmoid function defined as:

$$sgn\left(\boldsymbol{s}\right) \approx \frac{\boldsymbol{s}}{|\boldsymbol{s}| + \varepsilon} \tag{4.79}$$

where ε is a positive constant defined in the interval $0 < \varepsilon < 1$.

SMC with Variable Gain Switching

Another class of techniques is to keep the signum function unchanged and replace the constant switching gain Q with a variable one function of the sliding variable $\boldsymbol{s}$, one

possible choice can be:

$$Q(\boldsymbol{s}) = \frac{1}{\left(1 + \frac{1}{|\boldsymbol{s}|}\right) e^{-|\boldsymbol{s}|}} \tag{4.80}$$

As $\boldsymbol{s}$ becomes zero, the value of switching gain becomes:

$$\lim_{s \to 0} Q(s) \approx \lim_{s \to 0} \frac{s}{s+1} = 0 \tag{4.81}$$

hence the switching gain becomes zero at sliding surface (where s = 0), and chattering will be eliminated at steady state.

SMC with Variable Bandwidth Filter

Another possible solution to reduce chattering is to apply a filter to the control input. The control law is same as in the case of conventional SMC with the constant switching gain Q, but a filter with variable bandwidth, function of the sliding variable $\boldsymbol{s}$, is used instead. The filter transfer function can be implemented as follows:

$$G(\boldsymbol{s}) \frac{1}{1 + s\tau_f} \tag{4.82}$$

where the bandwidth is $\omega_f = k|s|$ and $\tau_f = \frac{1}{\omega_f}$. The bandwidth will be large during transient period, since the magnitude of s is large during transient period, and zero during steady state for the same reason.

Observer-based Sliding Mode Control

This more advanced approach uses an asymptotic state observers to construct an high frequency by pass loop [114]. The control is therefore discontinuous only with respect to the observer variables making chattering only localized inside the observer loop which, as we can see in fig. 4.16, bypasses the plant. However, this approach assumes that an

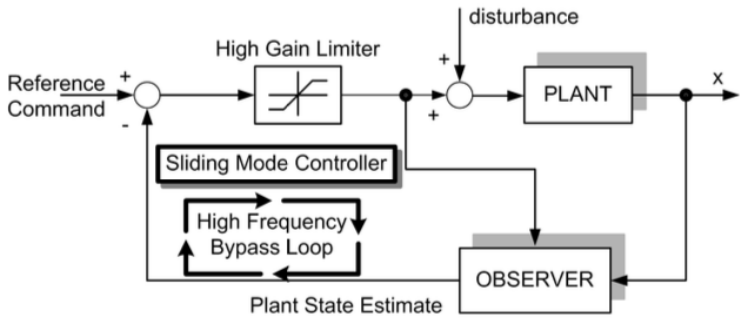


Figure 4.16: Observer based sliding mode control block diagram.

asymptotic observer can indeed be designed such that the observation error converges to zero asymptotically.

High-order Sliding Mode Control

The last proposed solution is to introduce a control action which is a function of higher order time derivatives of the sliding variable s. For example, authors proposed in [115] a second order sliding mode approach that allows the definition of a discontinuous control $\dot{u}$ capable of steering both the sliding variable $\boldsymbol{s}$ and its time derivative $\dot{\boldsymbol{s}}$ to zero. in this way that the plant input u is a continuous function and thus chattering can be avoided. The major difficulty of this approach is that there is no general method for tuning the parameters which characterize the various algorithms.

4.4.3 Implementation and Tuning with DE Algorithm

After the system kinematic and dynamic model is determined, and supposing that a feasible reference trajectory for the mobile robot is pre-specified by the planner defined in Section 4.2, in the format of pose $\boldsymbol{q}_d = [x_d,\ y_d,\ \vartheta_d]^T$ and velocity $\boldsymbol{v}_d = [v_d,\ \omega_d]^T$, it is now possible to design a robust sliding mode controller, based on some suitable methods proposed in the above sub-section so that the robot will correctly track the desired trajectory under a large class of disturbances. This section aims to design a stable controller that generates a command vector $\boldsymbol{u} = [u_c,\ \omega_c]^T$ to respect the assumptions made in chapter 2 on the form of the kinematic and dynamic eqs. (2.17) and (2.42), and also to respect, and therefore make possible its later implementation, the commands required by the physical Husky platform which, utilizing the *cmd_ vel* topic, needs them in the linear and angular velocities format.

Starting from the trajectory tracking problem described in Section 4.1, lets consider that the robot real position is expressed by a three element vector with components $(x_r,\ y_r,\ \vartheta_r)$. With reference to figure 4.1 and to the trajectory tracking error expressed by eq. 4.1, it is possible to obtain the corresponding error derivatives as:

$$\begin{aligned} \dot{x}_e &= -v_r + v_d \cos\vartheta_e + y_e\omega_r \\ \dot{y}_e &= v_d \sin\vartheta_e - x_e\omega_r \\ \dot{\vartheta}_e &= \omega_d - \omega_r \end{aligned} \tag{4.83}$$

The equation 4.83 is a two-input non-linear system, of which the switching function is difficult to be designed so, in this thesis we propose a new switching function design in which the lateral error y_e, and the angular error ϑ_e are internally coupled with each other in a sliding surface leading to convergence of both variables. For that purpose the following switching functions are proposed:

$$\boldsymbol{s} = \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} \dot{x}_e + kx_e \\ \vartheta_e + \arctan\left(v_d y_e\right) \end{bmatrix} \tag{4.84}$$

where k is a positive constant parameter.

To verify that the function is a good candidate for a sliding mode application, i.e. drives the state trajectory to the sliding surfaces, we have to check the reachability condition. If for s_1 it is clear that if it converges to zero, trivially x_e converges to zero, for s_2 this conclusion is not so immediate. Lets chose a Lyapunov candidate such as:

$$\mathcal{V} = \frac{1}{2}y_e^2 \tag{4.85}$$

Knowing that, for any $x \in \mathbb{R}$ and $|x| < \infty$, exist a function:

$$\phi(x) = x \sin(\arctan x) \geq 0 \tag{4.86}$$

when x_e equals zero, the sliding function $\boldsymbol{s} \to 0$ if:

$$\vartheta_e = -\arctan(v_d y_e) \tag{4.87}$$

Explaining the time derivative of $\mathcal{V}$:

$$\dot{\mathcal{V}} = y_e \dot{y}_e = y_e (v_d \sin \vartheta_e - x_e \omega_r) \tag{4.88}$$

and substituting the condition 4.87 can be obtained:

$$\dot{\mathcal{V}} = -y_e x_e \omega_r - y_e v_d \sin(\arctan(y_e v_d)) \tag{4.89}$$

Hence, remembering that $x_e = 0$ and applying the propriety derived from eq. 4.86 to equation 4.89, the reachability condition for s_2 is verified:

$$\dot{\mathcal{V}} \leq 0 \tag{4.90}$$

So it ca be true that when $s_1 \to 0$ and $s_2 \to 0$ consequently $x_e \to 0$ and ϑ_e converges to $\arctan(y_e v_d)$, so it comes with $y_e \to 0$ and $\vartheta_e \to 0$.

Having designed the switching function is now possible to start deriving the control law from the choice of the reaching law. A practical general form of reaching law is, as described above, the constant plus proportional rate reaching in which the introduction of a proportional term $-P\boldsymbol{s}$ makes the state trajectory approach the switching surface faster. This is based on its simplicity, but a major drawback occurs: the presence of the discontinuous signum function causes the chattering phenomenon due to unmodelled dynamics. In this thesis, a quasi sliding mode approach is used to weaken the chattering; so the improved reaching law can be described as:

$$\dot{s}_i = -Q_i s_i - P_i \frac{s_i}{|s_i| + \varepsilon_i}, \quad \text{for} \quad i = 1,2 \tag{4.91}$$

where Q_i and P_i are positive constants and ε_i is bounded in the interval $0 < \varepsilon < 1$.

From the time derivation of equation 4.84 we can write:

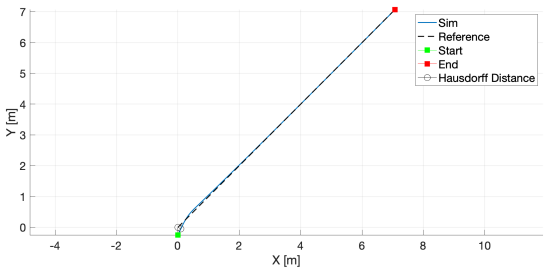
$$\begin{aligned} \dot{s}_1 &= -\dot{v}_r + \dot{v}_d \cos \vartheta_e - v_d \sin \vartheta_e \dot{\vartheta}_e + \dot{y}_e \omega_r + y_e \dot{\omega}_r + k \dot{x}_e \\ \dot{s}_2 &= \omega_d - \omega_r + \frac{y_e}{1 + (y_e v_d)^2} \dot{v}_d + \frac{v_d}{1 + (y_e v_d)^2} (v_d \sin \vartheta_e - x_e \omega_r) \end{aligned} \tag{4.92}$$

Replacing now $\dot{s}_i$ with 4.91 it is possible to solve the resulting system for $\dot{v}_r$ and ω_r obtaining the following control law:

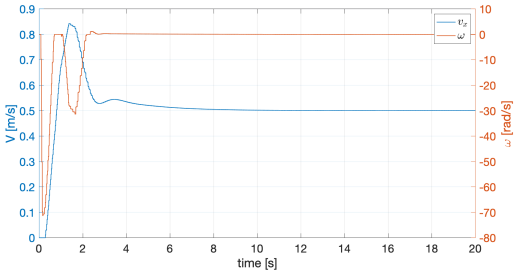
$$\boldsymbol{u} = \begin{bmatrix} \dot{v}_{r=c} \\ \omega_{r=c} \end{bmatrix} = \begin{bmatrix} Q_1 s_1 + P_1 \frac{s_1}{|s_1| + \varepsilon} + \dot{v}_d \cos \vartheta_e - v_d \sin \vartheta_e \dot{\vartheta}_e + \dot{y}_e \omega_r + y_e \dot{\omega}_r + k \dot{x}_r \\ \dfrac{Q_2 s_2 + P_2 \frac{s_2}{|s_2| + \varepsilon} + \omega_d + \frac{y_e}{1 + (y_e v_d)^2} \dot{v}_d + \frac{v_d}{1 + (y_e v_d)^2} v_d \sin \vartheta_e}{1 + \frac{v_d}{1 + (y_e v_d)^2} x_e} \end{bmatrix} \tag{4.93}$$

It can be noted that the linear control action obtained is written as the time derivative of the quantity we were looking for so, to be able to implement it, it will be necessary to integrate it.

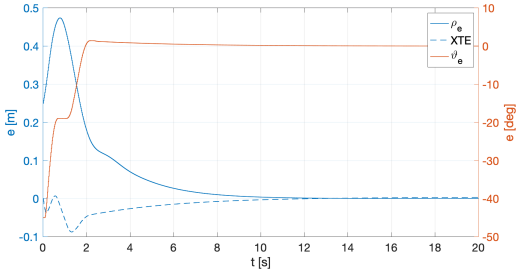
According to a first simulation on the tracking of a linear trajectory at the reference velocity of 0.5 m/s, with the robot having an initial cross-track error and different orientation, see Figure 4.17a, the tracking errors converges to 0 in about 10 s, see Figure 4.17c and the state trajectories reach the sliding surfaces in about 3 s as we can see respectively from Figure 4.17d. Even though the controller parameter have been determined



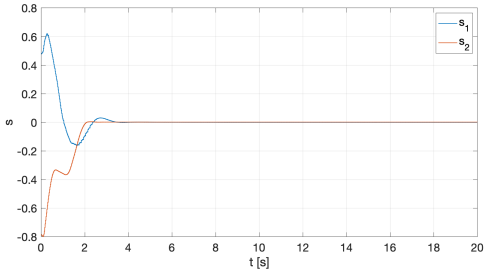
(a) Travelled trajectory.



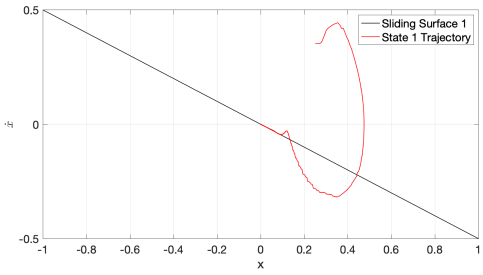
(b) Linear and angular velocities evolution.



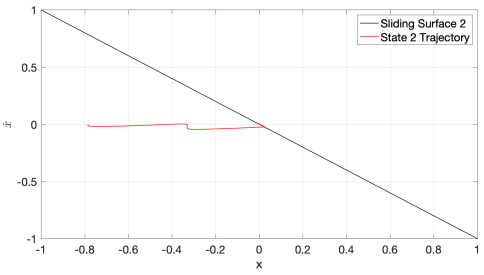
(c) Errors evolution.



(d) Switching functions convergence



(e) s_1 state trajectory convergence.



(f) s_2 state trajectory convergence.

Figure 4.17: SMC simulation tracking results of a linear trajectory with initial pose error.

roughly and manually at the values reported in Table 4.6, these results demonstrated the functionality of the implemented controller.

Before further simulations can be performed, the sliding mode controller's gain values must be refined. In this particular application, a manual trial and error tuning of

Table 4.6: SMC parameters for the example trajectory in presence of initial pose error.

Gains and Parameters	**Values**	**Gains and Parameters**	**Values**
$\boldsymbol{Q_1}$	1.8	$\boldsymbol{P_1}$	0.01
$\boldsymbol{Q_2}$	0.9	$\boldsymbol{P_2}$	0.6
$\boldsymbol{k}$	0.5	$\boldsymbol{\varepsilon}$	0.1

gain parameters Q_i, P_i and k have to be performed, leading at the determination of 5 parameters. This can be a very time-consuming process and also them optimum tuning is not guaranteed. To overcome this difficulty, the automatic tuning of parameters using a heuristic GA approach has been employed. The operational explanation of the chosen algorithm, together with its pseudo code, is reported in Appendix A, as it is common for both of the controllers studied; while in this section, we limit to report the values of its control parameters and the results obtained in terms of controller gains, convergence times and fitness function values. As done for the PID controller, the SMC will be optimised on the test *complex trajectory* and later tested on the other different trajectories presented in Section 4.2 for the sake of completeness. The optimization process does not take into account any external disturbance of sensor noise. The control parameters of the optimization algorithm and the cost function gains have been left the same as those used for PID tuning reported in respectively in Tables 4.2 and 4.3. The values obtained through the optimization process and the corresponding search bounds are shown in table 4.7.

Table 4.7: SMC optimized parameters.

Gains and Parameters	**Values**	$\boldsymbol{f_{opt}}$	**Optimization time**	**Search bounds**
Complex trajectory				
$\boldsymbol{Q_1}$	1.5756			[0, 2]
$\boldsymbol{Q_2}$	0.4759			[0, 2]
$\boldsymbol{P_1}$	0.0103			[0, 2]
$\boldsymbol{P_2}$	0.4009	21.6290	2.1340×10^4	[0, 2]
$\boldsymbol{k}$	0.5662			[0, 1]
$\boldsymbol{\varepsilon}$	0.1			constant

4.4.4 Simulation Results

In this subsection, the simulation results of the proposed controller are presented. The simulations are run in Simulink, so the equations describing the trajectory planner, see Section 4.2, the sliding mode controller and the plant of the mobile robot are transformed

into the model block diagrams shown in figure 4.18. In particular, the waypoint selector

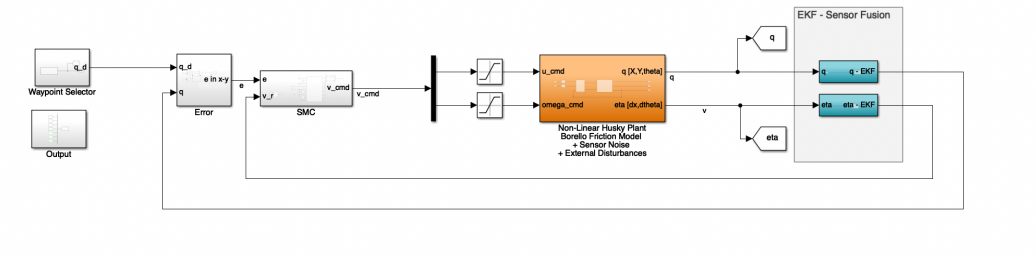


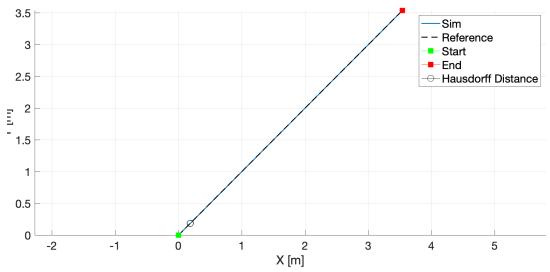
Figure 4.18: SMC simulation model block diagram.

block consists of a variant subsystem, containing the planners of the 4 type trajectories described above, which output the reference trajectory selected by the user. The error block generates the tracking error by subtracting the robot pose from the reference one; the resulting orientation is given in the range $(-\pi, \pi)$. The next block is the sliding mode controller, which takes the tracking error as reference and generates the control action based on equation 4.93. Once the control signal enters the mobile robot plant, it is first transformed by the wheel controller into the desired torque by eq. 2.46 so it can enter as input in the dynamic plant, containing the enhanced friction model and some saturation blocks for limiting the generated torques and velocities at them maximum values reported in table 3.2, which obtains the rover actual pose, used as feedback, by dead-reckoning. The last blocks are used for the sensor noise generation, based on white noise model.

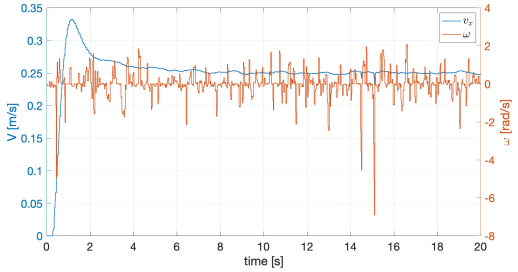
The control model is simulated for all the trajectories described before, without any initial position error and at a reference velocity of 0.25 m/s, using the gains obtained from the first optimization process reported in table 4.7.

Starting from the linear trajectory, which represents the least demanding case for the controller, it can be seen from figure 4.19 that, after a simulation of 20 seconds and taken away an initial settling period of about 4 second caused by the delay between the command and the response in which the state reaches the sliding surface, the actual outputs perfectly tracked the desired values, see Fig. 4.19d. From figure 4.19b, we can observe that the velocity profiles perfectly track the desired ones, and the command v_x remains in an acceptable range. Furthermore, once the initial error of e_x is recovered, we can observe from Fig. 4.19c that it is maintained around zero for the remaining part of the simulation.

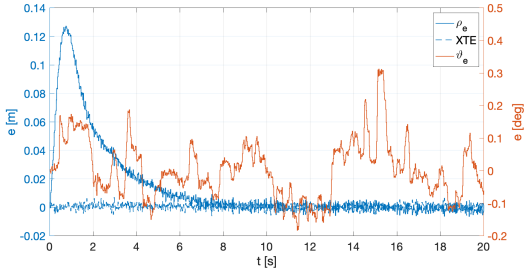
Interesting considerations can be made regarding the simulation of the complex trajectory reported in 4.22, in correspondence with the more accentuated turns, the state trajectory is driven away from the sliding surface, see Fig. 4.22d this is due to the sudden increases in the tracking errors. Despite this behaviour, the controller is able to quickly drive back the state on the sliding surface without too many oscillations. Furthermore, it is noted that the maximum deviation from the trajectory, indicated by the Hausdorff distance, is approximately 0.07 meters, demonstrating the correct functioning of the controller even in more complex cases. However, by analysing the speed profiles, should be



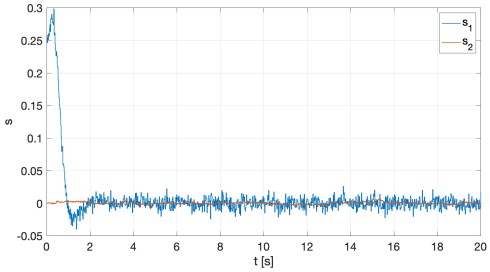
(a) Travelled trajectory.



(b) Linear and angular velocities evolution.

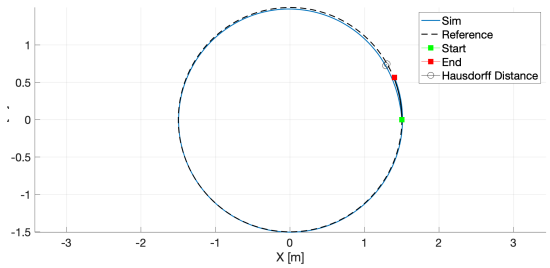


(c) Errors evolution.

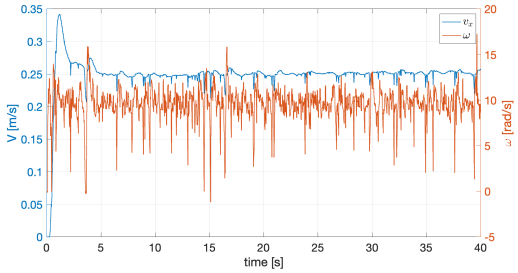


(d) Switching functions convergence

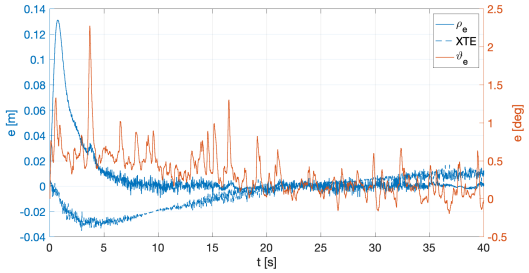
Figure 4.19: SMC tracking results of a linear trajectory.



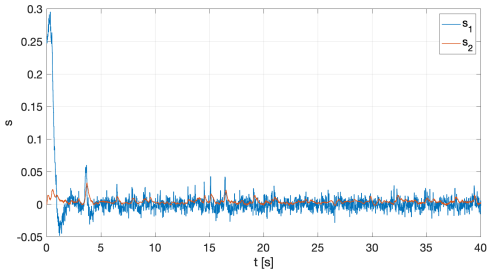
(a) Travelled trajectory.



(b) Linear and angular velocities evolution.

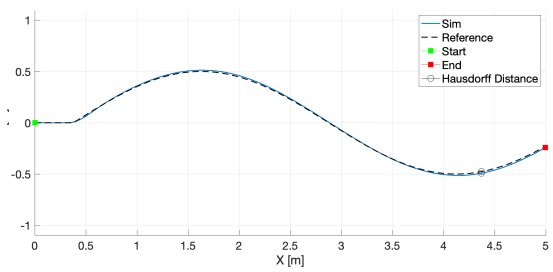


(c) Errors evolution.

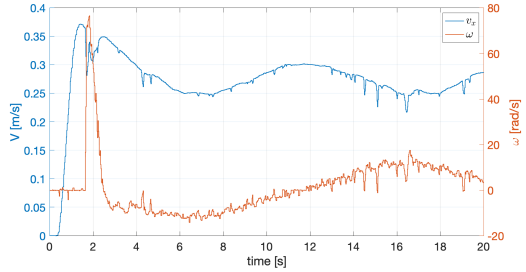


(d) Switching functions convergence

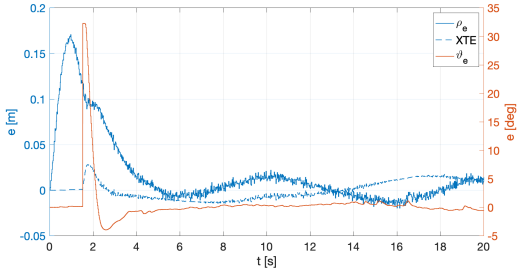
Figure 4.20: SMC tracking results of a circumferential trajectory.



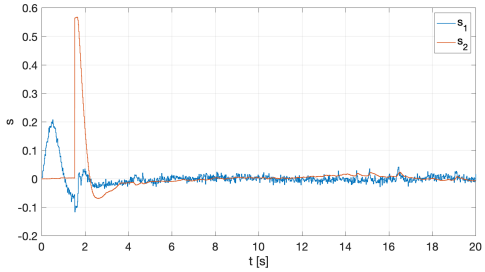
(a) Travelled trajectory.



(b) Linear and angular velocities evolution.

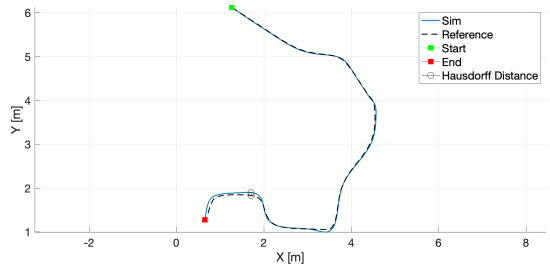


(c) Errors evolution.

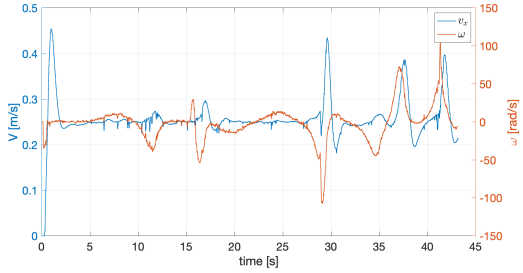


(d) Switching functions convergence

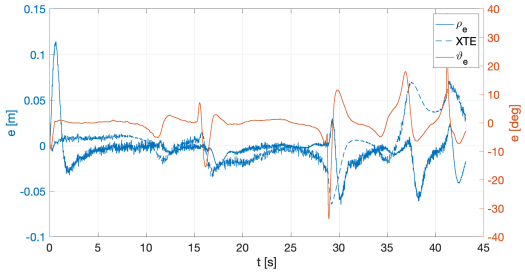
Figure 4.21: SMC tracking results of a sinusoidal trajectory.



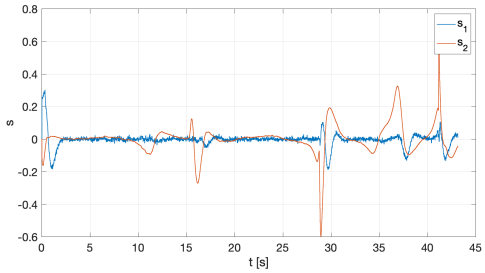
(a) Travelled trajectory.



(b) Linear and angular velocities evolution.



(c) Errors evolution.



(d) Switching functions convergence

Figure 4.22: SMC tracking results of a complex trajectory.

pointed out that the high angular velocity peaks in correspondence with the most accentuated turns could generate problems during the experimentation on the real rover thus making it necessary to additionally adjust the values of the gains.

Table 4.8: SMC trajectory tracking results.

Trajectory	d_{ratio}	d_{haus} [m]	d_{rms} [m]
Straight line	1.000	0.006	0.002
Circumference	0.996	0.031	0.014
Sinusoidal trajectory	1.002	0.016	0.010
Complex trajectory	1.021	0.071	0.033

The values of sliding variables for every trajectory, shown in figs. 4.19d, 4.20d, 4.21d and 4.22d, are very close to zero, confirming that the control law is working correctly. It must be restated that the values of the sliding variable will not be perfectly equal to zero during the sliding mode. It can be seen from Figure 4.21d by zooming the graph since the signum function is replaced with the sigmoid function. Therefore, the values of sliding variable are only trapped in the vicinity of the sliding surface where $s = 0$. Can be concluded that the results obtained from simulating the control law modelled in Simulink verify that the developed control logic is working properly. This modelled controller is now ready to be tested on the real platform.

Chapter 5

Experimental Results

The previous chapter illustrates the design of the proposed controller architectures, them implementation and the results of the numerical simulations that shown their control capability of the proposed WMR model, also in presence of sensor noise, external parametric disturbances and system uncertainties. This chapter deals with the preparation and the fulfilment of the experimental tests carried on a physical Clearpath Husky mobile platform at BRI indoor robotic testing facility. The controllers are first interfaced to the real platform and to the VRPN[1] client for having access to the ground truth of the rover measured by the RTS, explaining the problems encountered and the solutions proposed. Then the two control architectures are tested on different trajectories, and finally, the obtained results are compared. It is worth remembering that the experimental results here presented are incomplete, for the same reasons highlighted at the beginning of this thesis: this research was abruptly stopped by COVID-19 outbreak in mainland China at the end of January 2020. Therefore, the last and most promising computer simulations couldn't be tested on the real rover.

5.1 Model Configuration

To achieve the final objective of this thesis, namely to test the developed control laws directly on a physical mobile platform, it is first necessary to adapt the presented model to this occurrence, removing the part corresponding to the mathematical modelling of the WMR, as it is no longer necessary because replaced by its physical counterpart, and setting up a communication interface that allows real-time exchange of data between the two different actors. The controllers will then directly command the rover, basing their control action on the feedback signals measured by real sensors mounted on the vehicle. Thanks to the use of the ROS communication interface, which enables the different parts of a robotic system to discover each other and to send and receive data between them, it was

[1]The Virtual-Reality Peripheral Network is a device-independent, network-based interface for accessing Virtual Reality (VR) peripherals, such as the Realis Motion Tracking System (RTS) used in this research, over Transmission Control Protocol (TCP) or User Datagram Protocol (UDP), providing a unified interface to this input devices.

possible to carry out this operation without having to resort to other more complex and time-consuming methods, like C/C++ code generation and implementation on a control board. Besides, by means of the MathWorks ROS Toolbox and the ROS-enabled Husky UGV, it was possible to directly interface the Matlab/Simulink environment with the mobile robot, creating a network of ROS nodes thanks to special functions or blocks useful for importing, analysing and reproducing the data recorded in the rosbag, and also connecting to a live ROS network to access ROS messages.

The main changes made to the structure of the simulation program are shown as follows:

- **Command velocity publisher:** Replaces the part that represented the inputs of the mathematical model of the vehicle, generating, see Figure 5.1, a message of type *geometry_ msgs/Twist* sampled at 20 Hz. The message is then compiled via the *Bus Assignment* block, which assigns the commands u_c and ω_c from the controller to a bus coding them respectively as *Linear.X* and *Angular.Z*, and finally sent via the *Publisher* to the */cmd_ vel* ROS topic.

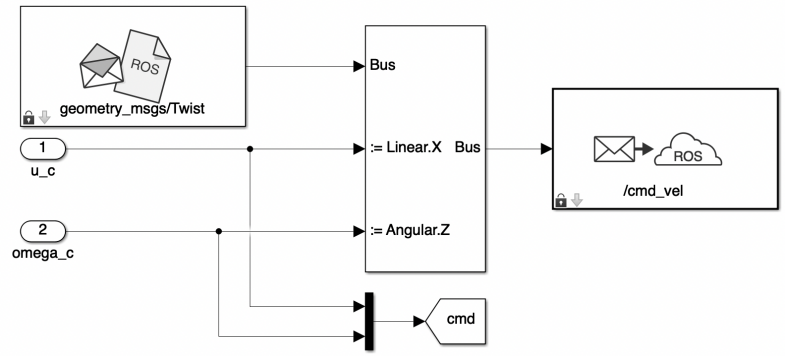
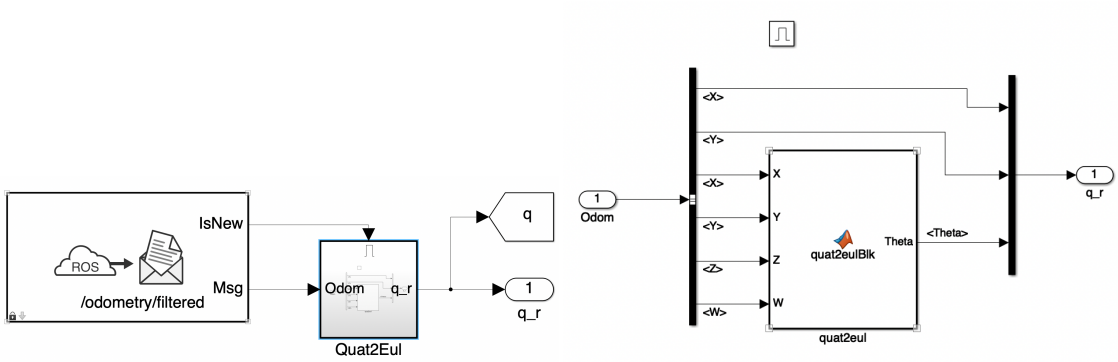


Figure 5.1: Block diagram representation of the command velocity publisher.

- **Odometry subscriber:** Replaces the part of the mathematical model that computed the WMR odometry by, see Figure 5.2a, receiving a message of type *nav_ msgs /Odometry* sampled at 50 Hz from the subscription, by means of a *Subscriber* block, to the */odometry_ filtered* ROS topic. The received message is then unpacked by a *Bus Selector* block which extracts the values of position and orientation, this expressed in quaternions, computed by the Husky EKF and codified respectively as signals *Pose.Pose.Position.X, Y* and *Pose.Pose.Orientation.X, Y, Z, W*. As we can see from Figure 5.2b, the Husky orientation is then transformed in its Euler angles representation by means of the MATLAB function *quat2eul*, thus obtaining the robot actual pose in the required format to be feed-backed to the controller.

- **Velocity subscriber:** This component replaces the part of the mathematical model that computed the rover actual velocity. With reference to Figure 5.3 , its functioning is completely analogous to the odometry subscriber one but the message of type *nav_ msgs/Odometry*, sampled also at 50 Hz, is this time received from the



(a) Subscriber structure. (b) Odometry transformation.

Figure 5.2: Block diagram representation of the odometry subscriber.

/husky_velocity_controller/odom ROS topic. The Husky linear v_r and angular ω_r actual velocity values extracted by the *Bus Selector* according respectively to the coding *Twist.Twist.Linear.X* and *Twist.Twist.Angular.Z*, are then feed-backed to the controller without any further transformation because already in the required format.

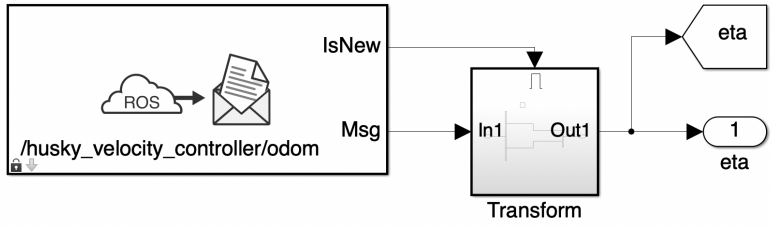
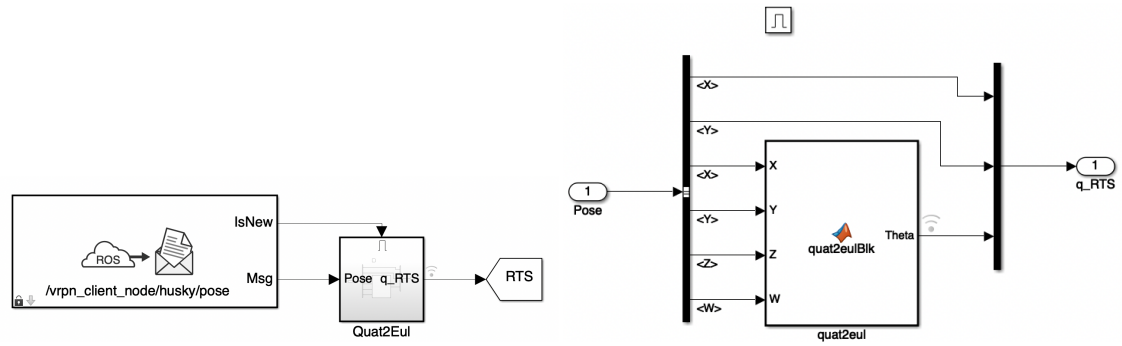


Figure 5.3: Block diagram representation of the velocity subscriber.

- **RTS pose subscriber:** This block diagram do not replaces any part of the mathematical model but is introduced for receiving the ground truth of the Husky platform measured by the motion tracking system of the laboratory. As can be seen from Figure 5.4a, completely analogous to other subscribers, the robot real pose is received by the as a message of type *geometry_msgs/PoseStamped*, sampled at 120 Hz from the */vrpn_client_node/husky/pose* ROS topic. To be able to compare the Husky ground truth with its odometry, after extracting the content of the message through the *Bus Selector* block, in the same way as for odometry, the obtained orientation expressed in quaternion, see Figure 5.4b, is transformed in its Euler angles representation by the aforementioned *quat2eul* MATLAB function. The obtained pose is sent directly to the MATLAB workspace for post-processing operations.

It should be specified that the sampling time used in the respective blocks was not chosen randomly but dictated by the characteristics of the sensors or systems, and therefore necessary to obtain valid simulation results. The components or systems relating to



(a) Subscriber structure. (b) RTS pose transformation.

Figure 5.4: Block diagram representation of the RTS pose subscriber.

the specified sampling times are shown in Table 5.1.

Table 5.1: Sampling times and relative system components.

Components	**Sampling Time** $[Hz]$
RTS	120
EKF	50
CmdVel	20

By applying the aforementioned changes to the models, we are ready to execute the test on the Husky physical platform.

5.1.1 Encountered Problems

Before assessing the experimental results, it is good to describe, for the sake of clarity, the problems encountered during this phase and the proposed solutions:

- ***IMU frame mismatch*:** The Husky EKF computes the vehicle pose merging the measurements coming from the IMU and the wheel encoders within the local reference frame, whose values are the basis of the feedback action. Preliminary tests on the mobile platform showed a mismatch in the IMU and the mobile robot reference frame. The first was rotated of 180° around X causing incorrect reading of speeds and accelerations by the EKF, generating unexpected responses. The problem was solved by modifying the vehicle configuration files loaded by ROS at rover start-up, reported in Listing 5.1

Listing 5.1: Husky IMU setup file.

```
1 # Mark location of self so that robot_upstart knows where to
      find the setup file.
2 export ROBOT_SETUP=/etc/ros/setup.bash
3
```

```
4          # Setup robot upstart jobs to use the IP from the network
               bridge.
5          # export ROBOT_NETWORK=br0
6          # Insert extra platform-level environment variables here. The
               six hashes below are a marker
7          # for scripts to insert to this file.
8
9          export HUSKY_IMU_XYZ='0 -0.15 0.065'
10         export HUSKY_IMU_RPY='3.14 0 0'
11         export HUSKY_TOP_PLATE_ENABLED=true
12
13         ######
14         # Pass through to the main ROS workspace of the system.
15         source /opt/ros/melodic/setup.bash
```

- ***RTS frame orientation mismatch*:** The motion tracking system measures the actual position of the vehicle by inserting it into a pre-setted reference system which, for the sake of simplicity, was used as the global reference frame for the experimental tests carried out. Initial tests have shown that although the RTS correctly measured the displacements within the generated coordinate system, the angular component ϑ, which represents the actual orientation of the rover, had an error of 90° compared to what was expected. Not being able to act directly within the tracking system settings for technical reasons, it was decided to solve the problem within the simulation environment by subtracting the a value of $\frac{\pi}{2}$ from each measurement in question.

- ***Discontinuity in RTS orientation measurements*:** Since the motion tracking system provides the Husky orientation in the interval $(-\pi,\ \pi)$, every time the rover crosses this discontinuity the controller receives an unexpected angular error which causes an unpredicted unwanted response. For this reason, it was decided to create a Simulink block named *Remove Theta Discontinuity* containing a function, reported in Listing 5.2, that eliminated this discontinuity by adding or subtracting the quantity 2π to the ridden value depending on the crossing direction.

Listing 5.2: Remove Theta Discontinuity matlab function.

```
1          function theta = fcn(buffer)
2
3          global i_phase
4
5          theta_prev = buffer(1);
6          theta_now = buffer(2);
7
8          if (theta_prev-theta_now) > 6         % -pi -> pi crossing
9          i_phase = i_phase+1;
10         elseif abs(theta_prev-theta_now) > 6     % pi -> -pi crossing
11         i_phase = i_phase-1;
12         end
13
14         theta = theta_now+i_phase*2*pi;
```

- ***RTS inaccuracies*:** Although the motion tracking system installed in the testing facility was characterised by a very high measurements accuracy, see Section 3.1,

a first inspection of the simulations data noticed the presence of not negligible oscillations which however did not reflect the actual behaviour of the vehicle. This problem has been identified in the presence of grey areas in the laboratory sector used for testing which, given the not negligible dimensions of the reference trajectories, represented the only possible choice. Since the only way to remedy the problem was to reposition the motion capture cameras, obviously not possible for technical reasons, it was decided to accept these inaccuracies by trying to reduce it as much as possible during the post-processing; representing the ground truth of the rover, these are fundamental measurements for evaluating the performance of the controllers.

- ***Simulation rate synchronization*:** Since the Simulink environment is characterised by an execution pace for which one simulation-second is completed in a few wall clock time milliseconds, testing the controller directly on a real mobile platform through this simulation environment would produce results that have no physical correlation since the time to which the controller refers and the one to which the vehicle is subject are not synchronised. To overcome this problem, the *simulation pace* block was introduced in the model block diagram, which allows to run the simulation at a slower pace by imposing unitary equality between simulation time and wall clock time. The simulation pace is implemented by putting the entire MATLAB thread to sleep until it must run again to keep up the pace. Following this approach is now possible to observe the simulation results and understand the system behaviour, identify design issues and demonstrate its near real-time behaviour.

- ***Husky EKF pose initialization*:** The rover measures its position by merging the data from IMU and encoder starting from an initial null pose, re-setted at each power on, and increasing it during movement based on the sensor readings. Since these values are used for position feedback, and the reference trajectory is generated in the global reference system defined by the RTS, it is necessary that the initial odometry value coincide with the pose of the first trajectory waypoint. For doing this, we call via MATLAB the rosservice *set_pose*, which initialises the initial position end orientation of the UGV at the values received from the RTS. The service call is implemented in the bash file *set_pose.sh* reported in Listing 5.3, that is written whenever it is necessary to initialise the pose:

Listing 5.3: Set pose rosservice bash script.

```
1  rosservice call /set_pose "pose:
2  header:
3  seq: 0
4  stamp:
5  secs: 0
6  nsecs: 0
7  frame_id: odom
8  pose:
9  pose:
10 position: {x: X_rts, y: Y_rts, z: 0}
11 orientation: {x: 0, y: 0, z: Zq_rts, w: Wq_rts}
```

```
12          covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
                0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,0.0, 0.0, 0.0, 0.0, 0.0,
                0.0, 0.0]"
```

where X_{rts} and Y_{rts} represent the robot real position directly taken from the tracking system, and Zq_{rts} and Wq_{rts} are the first and last coefficient of the Husky quaternion orientation always taken from the RTS and suitably modified to match, as shown above, the orientations of the two reference systems.

- ***MATLAB-Linux integration*:** Despite the ROS integration with MATLAB 2019a granted by the Robotics System Toolbox, it is not possible to use the MATLAB console to run a Linux script that executes ROS commands, like the one reported in Listing 5.3. In particular, scripts that function flawlessly when called from the Linux shell failed to execute when called using the common ![*scriptpath*] MATLAB syntax. The cause of this issue was concluded in MATLAB itself, that changes the Linux environment variables whenever it is running, altering therefore the correct functioning of any script. To overcome such problem, it was decided to modify the affected scripts to include a series of export instances to reset the environment variables to the desired values. The listing of a generic script of that kind is provided in Listing 5.4; where the correct values of the environment variables can be determined by running the *env* command within the Linux shell.

Listing 5.4: Environmental variables export example.

```
1   #!/bin/bash
2   export LD_LIBRARY_PATH=...
3   export ROS_ETC_DIR=...
4   export CMAKE_PREFIX_PATH=...
5   export ROS_ROOT=...
6   export ROS_MASTER_URI=...
7   export ROS_VERSION=...
8   export ROS_PYTHON_VERSION=...
9   export PYTHONPATH=...
10  export ROS_PACKAGE_PATH=...
11  export ROSLISP_PACKAGE_DIRECTORIES=...
12  export PATH=...
13  export PKG_CONFIG_PATH=...
14  export ROS_DISTRO=...
15  #!... Script Body ...
```

5.2 PID Controller Results

Once the structure of the simulation program has been modified, as mentioned in Section 5.1, obtaining the block diagram representation shown in Figure 5.5, it was possible to perform the experimental tests whose main results are shown in the following. For each of the reference trajectories described in Section 4.2, several tests were carried out at incremental speeds, starting from the nominal value of 0.25 m/s up to the most demanding one of 0.75 m/s to test the rover and controller limits. Each result is therefore compared

with the relative one deriving from the numerical simulations to draw the final considerations. The PID controller tuning, whose values are shown in Table 5.2, is identical for each test and was performed on the results deriving from the optimisation carried out on the discrete model relating to the complex trajectory performed at the nominal speed. The graphical representation was reserved only for those cases necessary to draw the conclusions on the performance of the control law, while the numerical results, together with their counterparts deriving from the numerical simulations are listed in them entirety in Table 5.3.

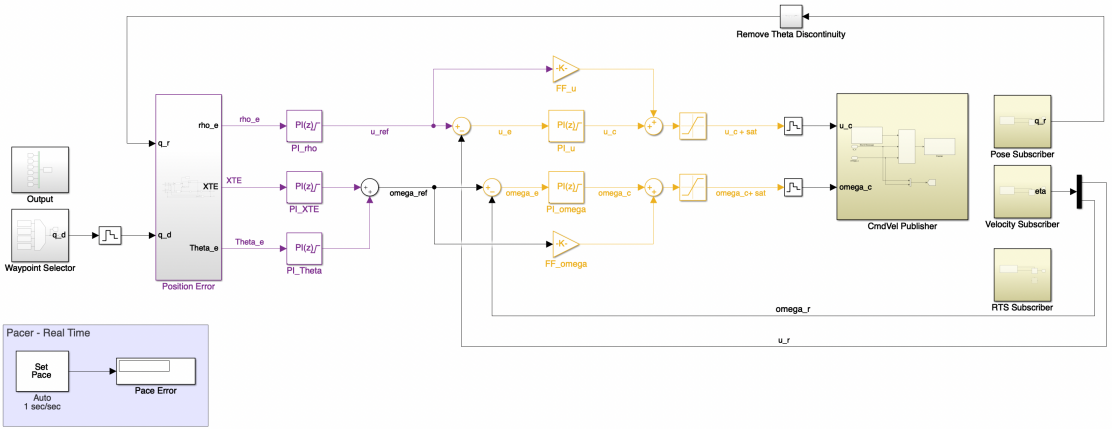
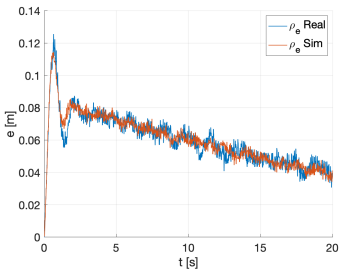


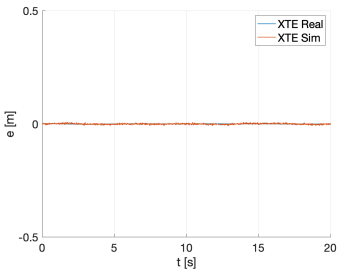
Figure 5.5: PID experimental model block diagram.

Table 5.2: PID gains used in the experimental tests.

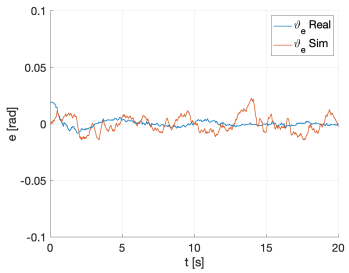
Controller	Branch	Gain	Value	Gain	Value
Position	Linear	K_{P_ρ}	2.8360	K_{I_ρ}	0.1146
		$K_{P_{XTE}}$	5.9828	$K_{I_{XTE}}$	1.9708
	Angular	K_{P_ϑ}	4.5073	K_{I_ϑ}	5.2760×10^{-4}
Velocity	Linear	K_{P_v}	0.0372	K_{I_v}	0.4769
	Angular	K_{P_ω}	0.6498	K_{I_ω}	0.5481
Feed Forward	Both	FF_ρ	1.00	FF_ω	0.00



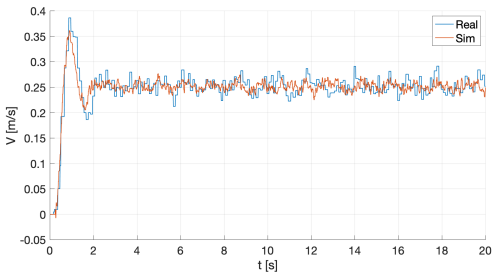
(a) ρ error evolution.



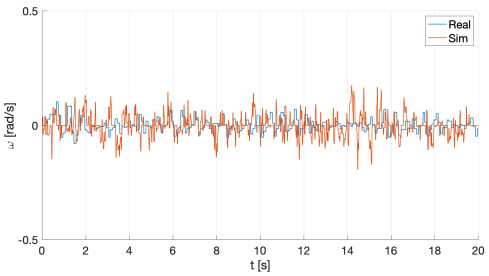
(b) XTE evolution.



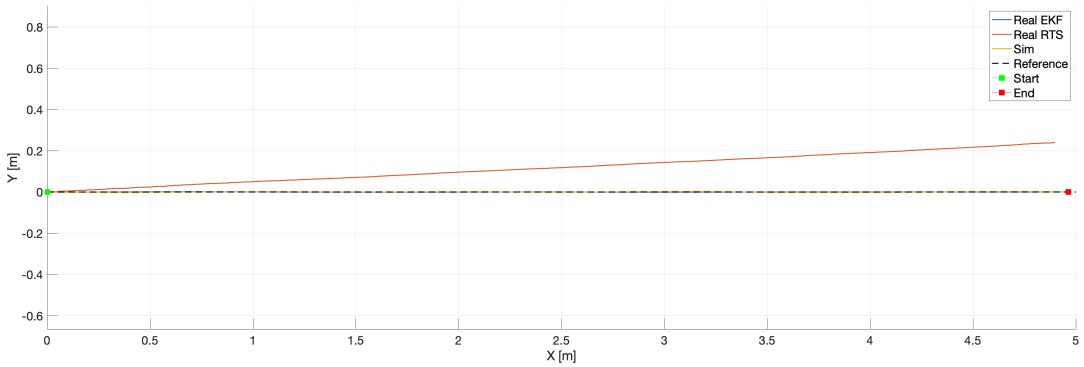
(c) ϑ error evolution.



(d) Linear velocity evolution.

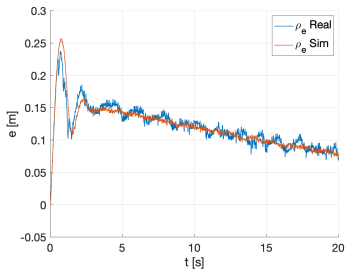


(e) Angular velocity evolution.

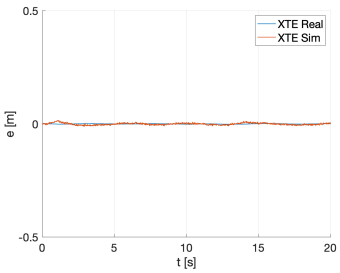


(f) Trajectory comparison.

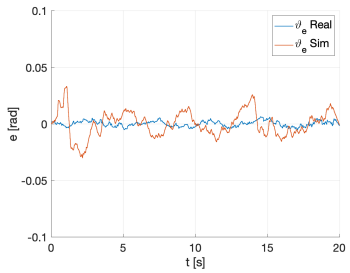
Figure 5.6: PID experimental tracking results of a linear trajectory executed at 0.25 m/s.



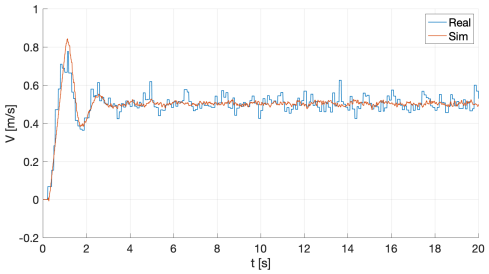
(a) ρ error evolution.



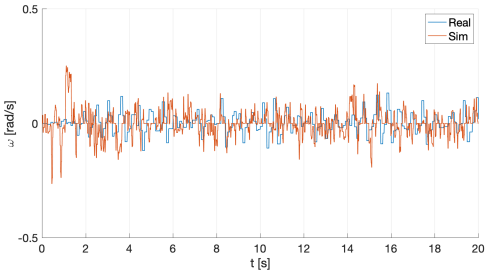
(b) XTE evolution.



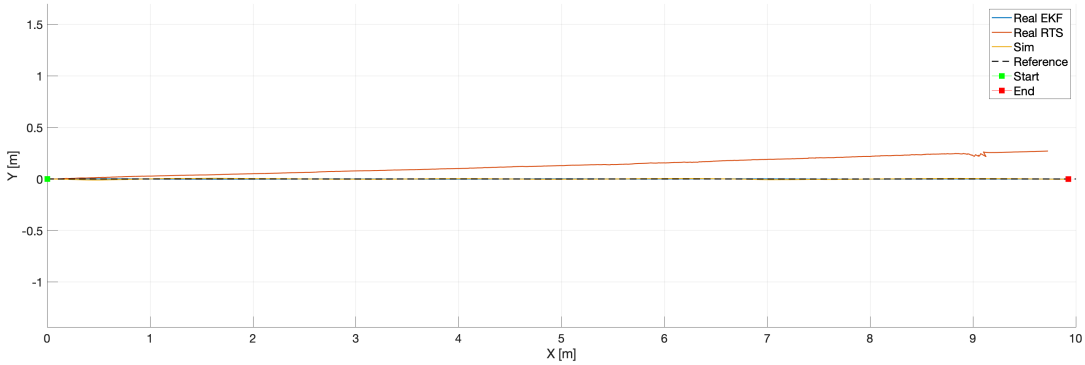
(c) ϑ error evolution.



(d) Linear velocity evolution.

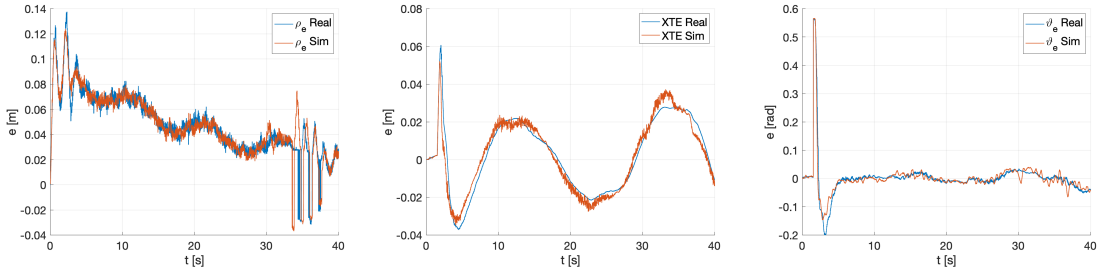


(e) Angular velocity evolution.



(f) Travelled trajectory comparison.

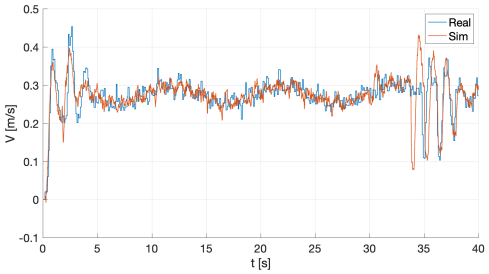
Figure 5.7: PID experimental tracking results of a linear trajectory executed at 0.50 m/s.



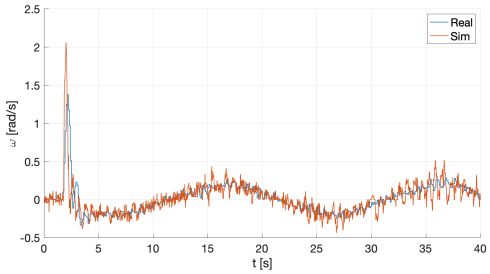
(a) ρ error evolution.

(b) XTE evolution.

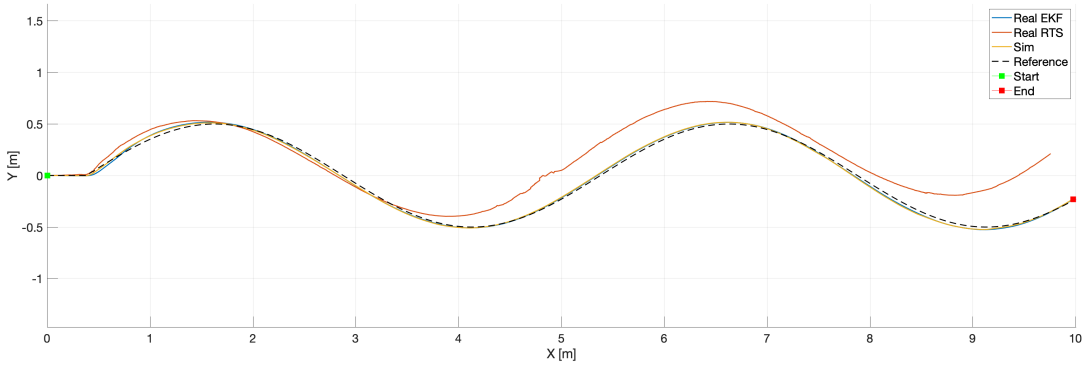
(c) ϑ error evolution.



(d) Linear velocity evolution.

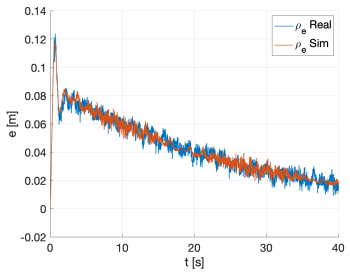


(e) Angular velocity evolution.

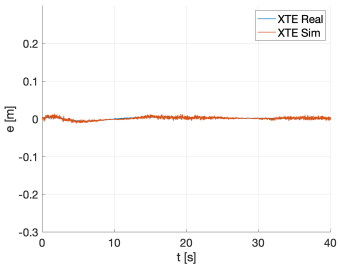


(f) Travelled trajectory comparison.

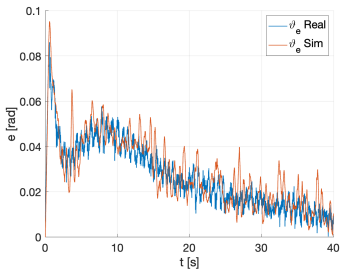
Figure 5.8: PID experimental tracking results of a sinusoidal trajectory executed at 0.25 m/s.



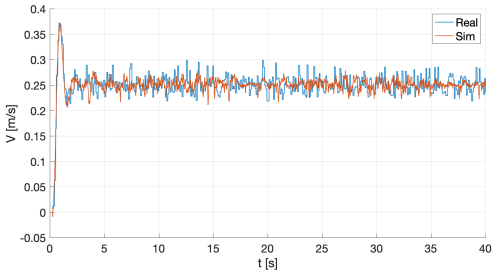
(a) ρ error evolution.



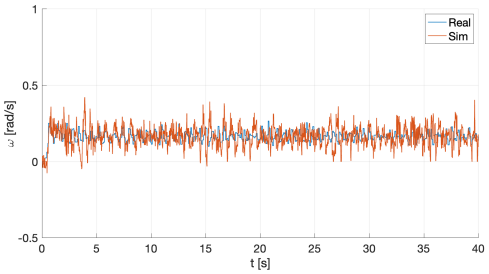
(b) XTE evolution.



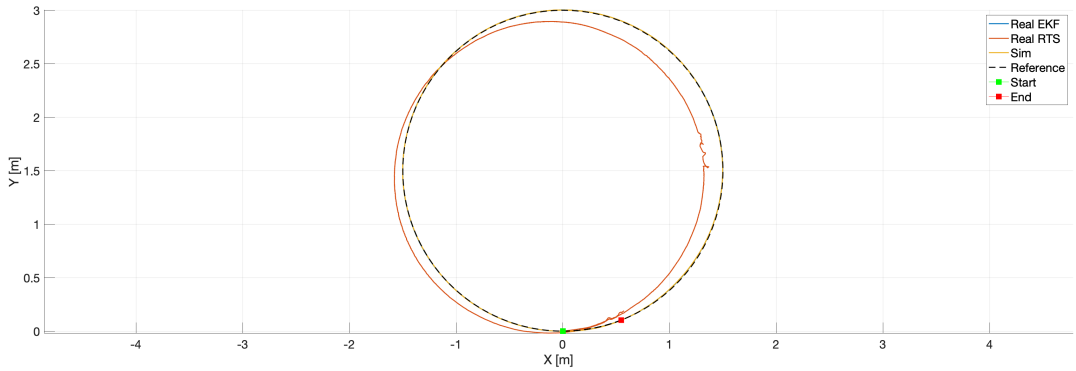
(c) ϑ error evolution.



(d) Linear velocity evolution.

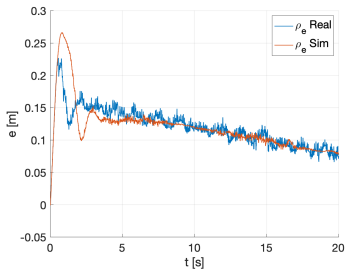


(e) Angular velocity evolution.

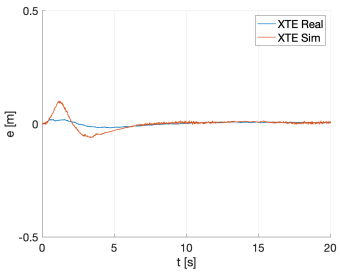


(f) Travelled trajectory comparison.

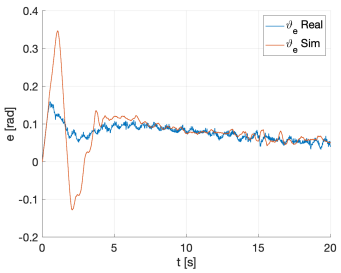
Figure 5.9: PID experimental tracking results of a circumference executed at 0.25 m/s.



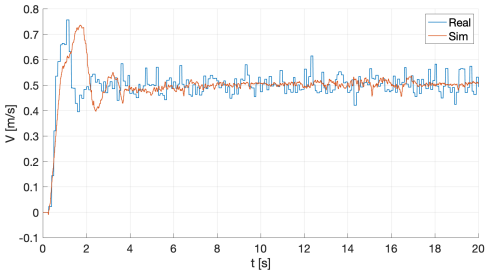
(a) ρ error evolution.



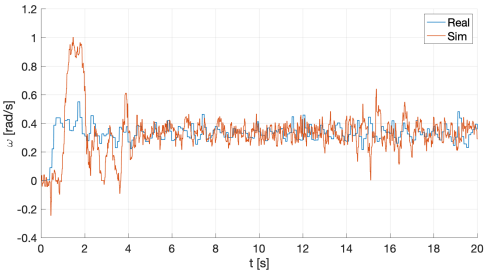
(b) XTE evolution.



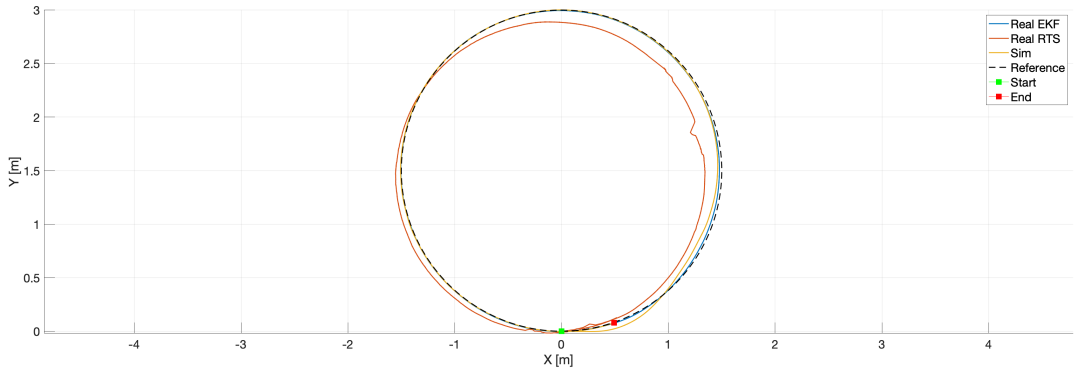
(c) ϑ error evolution.



(d) Linear velocity evolution.

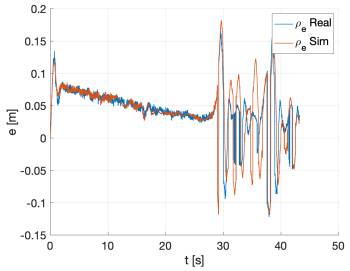


(e) Angular velocity evolution.

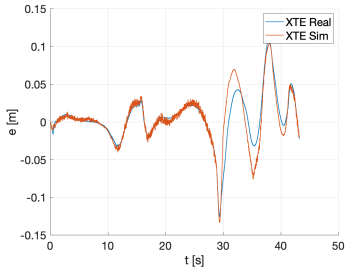


(f) Travelled trajectory comparison.

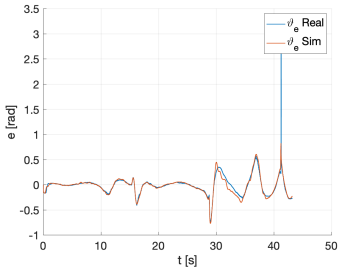
Figure 5.10: PID experimental tracking results of a circumference executed at 0.50 m/s.



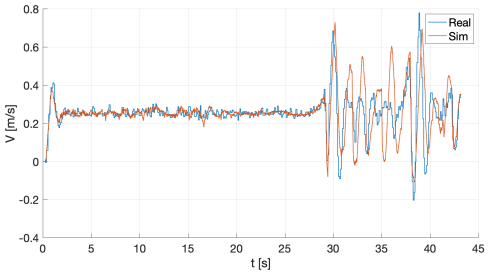
(a) ρ error evolution.



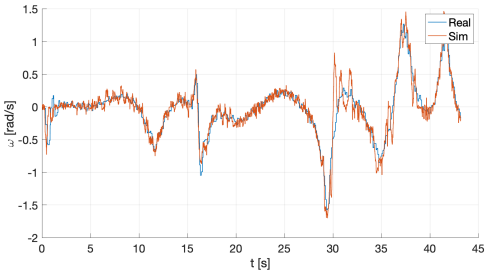
(b) XTE evolution.



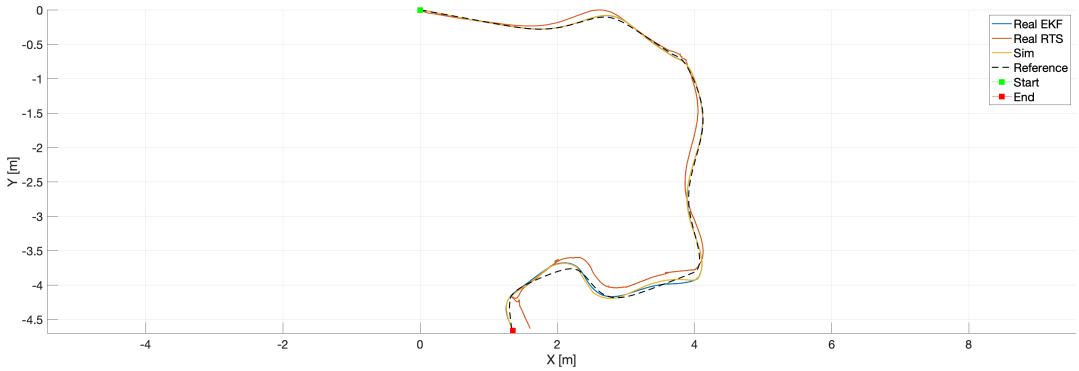
(c) ϑ error evolution.



(d) Linear velocity evolution.

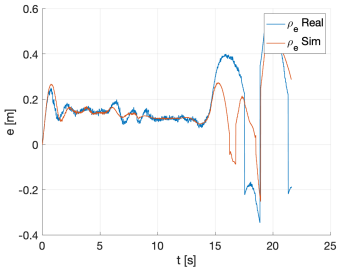


(e) Angular velocity evolution.

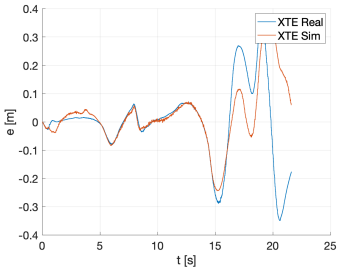


(f) Travelled trajectory comparison.

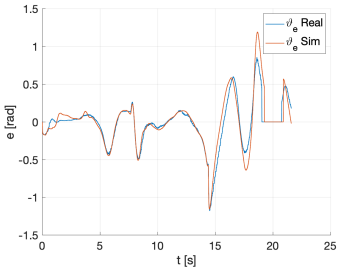
Figure 5.11: PID experimental tracking results of a complex trajectory executed at 0.25 m/s.



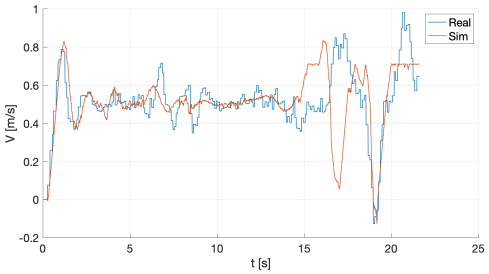
(a) ρ error evolution.



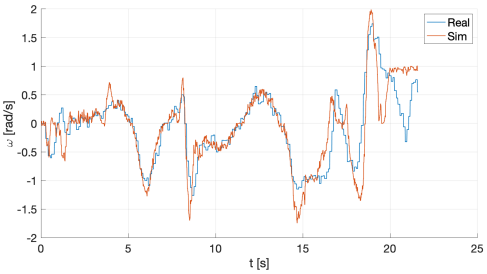
(b) XTE evolution.



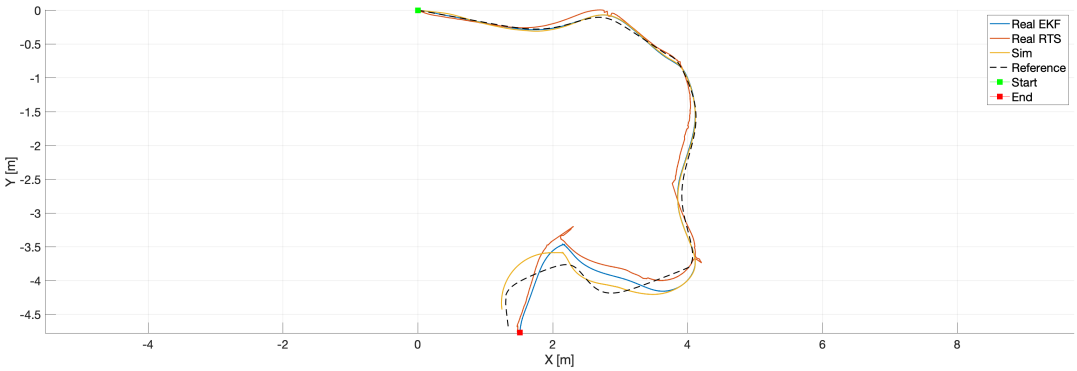
(c) ϑ error evolution.



(d) Linear velocity evolution.



(e) Angular velocity evolution.



(f) Travelled trajectory comparison.

Figure 5.12: PID experimental tracking results of a complex trajectory executed at 0.50 m/s.

Table 5.3: PID trajectory tracking experimental results.

Reference Velocity	***Simulation***			***Experimental***		
	d_{ratio}	d_{haus} [m]	d_{rms} [m]	d_{ratio}	d_{haus} [m]	d_{rms} [m]
Straight line						
0.25 m/s	0.992	0.038	0.004	0.993	0.037	0.005
0.50 m/s	0.992	0.078	0.008	0.993	0.074	0.008
0.75 m/s	0.991	0.130	0.016	0.991	0.125	0.014
Circumference						
0.25 m/s	0.999	0.013	0.006	0.999	0.015	0.007
0.50 m/s	0.991	0.061	0.023	0.991	0.029	0.012
0.75 m/s	0.990	0.286	0.096	0.910	0.200	0.096
Sinusoidal trajectory						
0.25 m/s	0.997	0.044	0.016	1.004	0.036	0.019
0.50 m/s	0.993	0.081	0.013	0.992	0.084	0.013
0.75 m/s	1.001	0.150	0.071	0.993	0.155	0.038
Complex trajectory						
0.25 m/s	1.033	0.105	0.034	1.042	0.109	0.032
0.50 m/s	1.029	0.298	0.103	1.038	0.348	0.118

After a first look at the graphical and numerical results, it can be stated that the PID controller was able, regardless of the goodness of the results, to complete all the simulations carried out, that is, to perform all the proposed trajectories under different conditions; thus validating the design of the control algorithm. Analysing the performance instead, it can be seen how the increase in speed leads to less satisfactory results arriving in some cases, see Figures 5.12 and 5.23, to be almost unacceptable. This loss of performance is undoubtedly due to a not very robust design of the controller, which, moving away from the nominal operating condition, struggles to track the reference trajectory. However, the limited number of tests leaves room for possible improvements. Further optimisation through the jDE algorithm, carried out on different trajectories and also considering the feed-forward gains, could lead to an improvement in the results.

An interesting consideration can be made on the curves representing the Husky ground truth measured by the motion tracking system. These, representing the most accurate measurement possible of the rover's pose, differ from those measured by the on-board EKF even in the case of linear trajectories in which the angular velocity is practically zero. A possible reason may be the presence of bias in the modelling of the sensors. However, since this is not the case, the cause has been identified in a mechanical damage of one wheel encoder. Since the control system only refers to the feedback generated by the EKF, this error cannot be corrected.

5.3 Sliding Mode Controller Results

By applying the same procedure to the SMC architecture, whose final block diagram representation and control parameters are defined respectively in Figure 5.13 and Table 5.4, it was possible to obtain the following results. The graphical representation was reserved only for those cases necessary to draw the conclusions on the performance of the control law, while the numerical results, together with their counterparts deriving from the numerical simulations are listed in them entirety in Table 5.5.

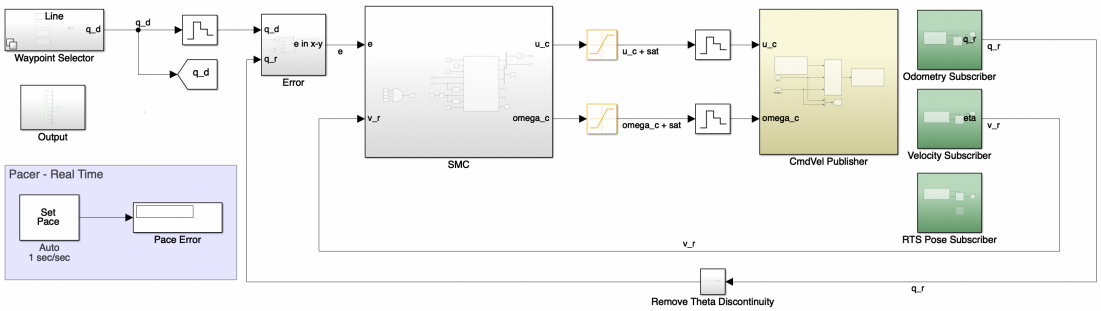


Figure 5.13: SMC experimental model block diagram.

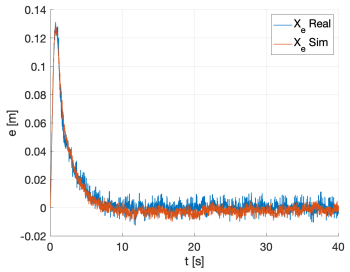
Table 5.4: SMC gains used in the experimental tests.

Gains and Parameters	Values	Gains and Parameters	Values
$\boldsymbol{Q_1}$	1.5756	$\boldsymbol{P_1}$	0.0103
$\boldsymbol{Q_2}$	0.4759	$\boldsymbol{P_2}$	0.4009
$\boldsymbol{k}$	0.5662	$\boldsymbol{\varepsilon}$	0.1

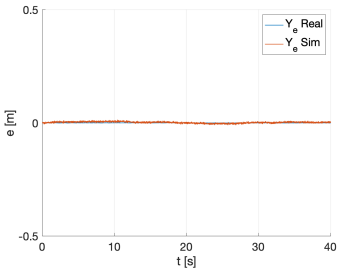
After a first look at the graphical and numerical results, it can be stated that also the SMC controller was able to perform all the proposed trajectories under different conditions; thus validating the design of the control algorithm. Furthermore, the better performances of this class of more advanced controllers are immediately notable. In essence, the smoother control action leads to fewer peaks and oscillations in the velocity profiles and the tracking errors are quickly driven towards zero achieving an optimal tracking of the reference trajectory at low speed. Moreover, the increase in the reference speed has an inferior influence on the loss of performance. This can be seen from the circumference performed at 0.75 m/s, see Figure 5.23, which represents one of the worst cases of application, whose response is this time widely acceptable.

Regarding the comparison between experimental and simulated results, the close similarity of the low-speed responses, see Figure 5.19, demonstrates the reliability of the mathematical model. As this increases, the fidelity of the model begins to decline. This is due to the non-linearities that characterise the model, some of which have not been considered in the mathematical formulation. As the speed at which these problems begin

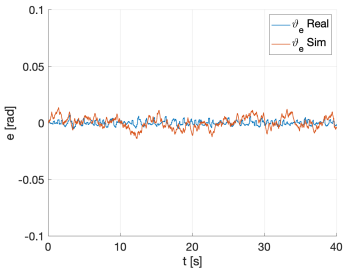
to occur is close to the maximum operative of the mobile robot, it is possible to consider it an acceptable result.



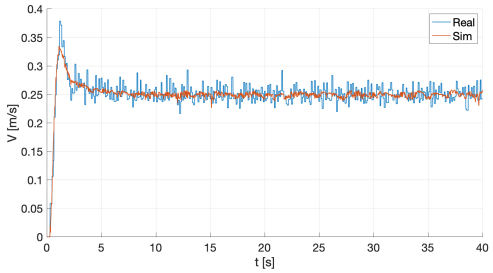
(a) X error evolution.



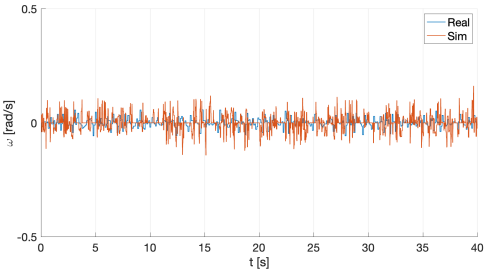
(b) Y error evolution.



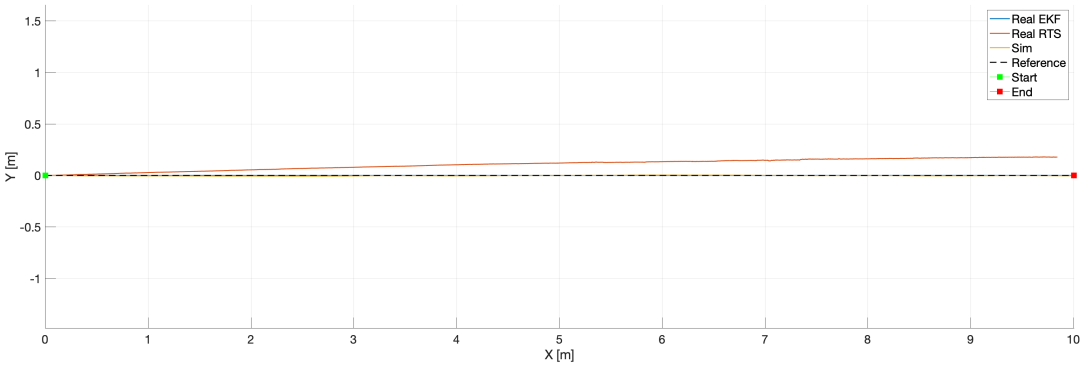
(c) ϑ error evolution.



(d) Linear velocity evolution.

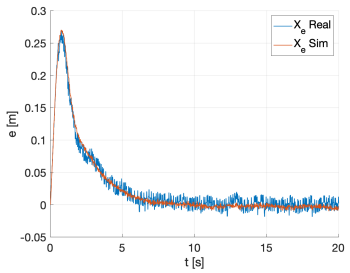


(e) Angular velocity evolution.

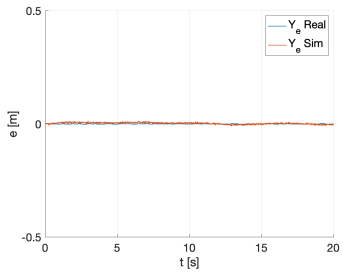


(f) Travelled trajectory comparison.

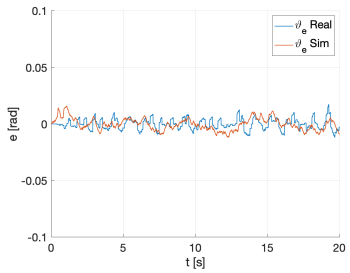
Figure 5.14: SMC experimental tracking results of a linear trajectory executed at 0.25 m/s.



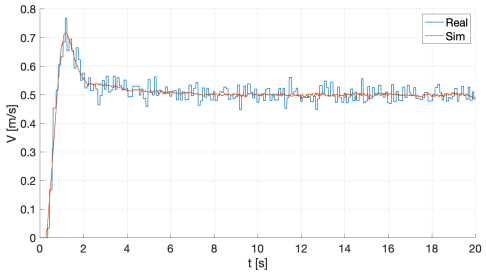
(a) X error evolution.



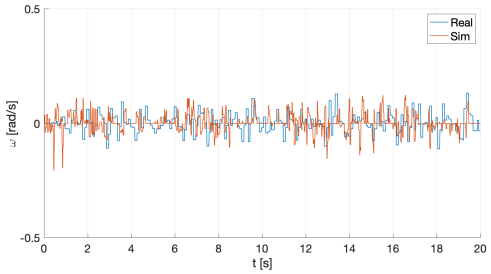
(b) Y error evolution.



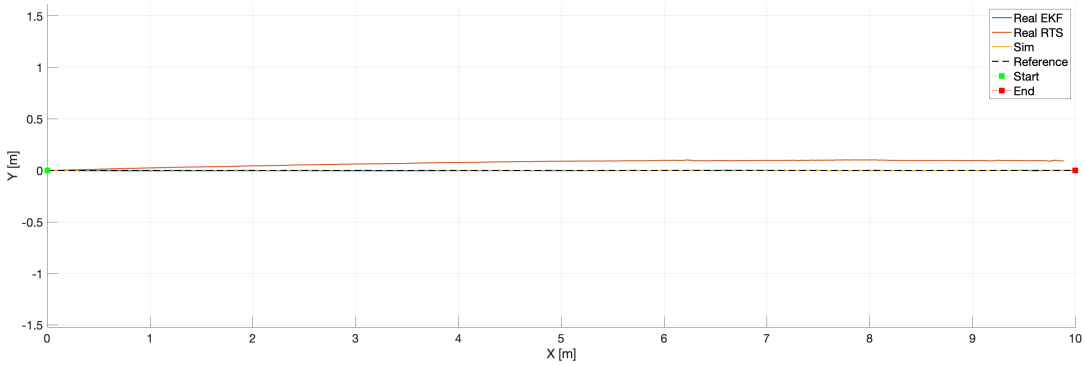
(c) ϑ error evolution.



(d) Linear velocity evolution.

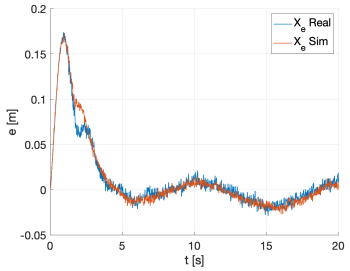


(e) Angular velocity evolution.

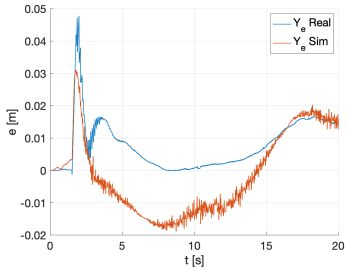


(f) Travelled trajectory comparison.

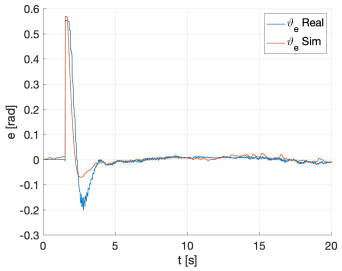
Figure 5.15: SMC experimental tracking results of a linear trajectory executed at 0.50 m/s.



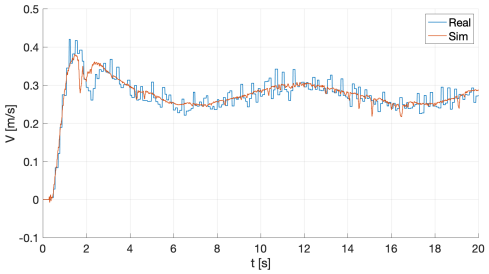
(a) X error evolution.



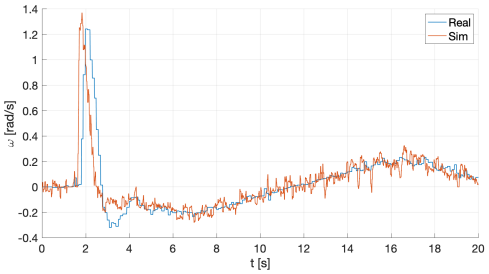
(b) Y error evolution.



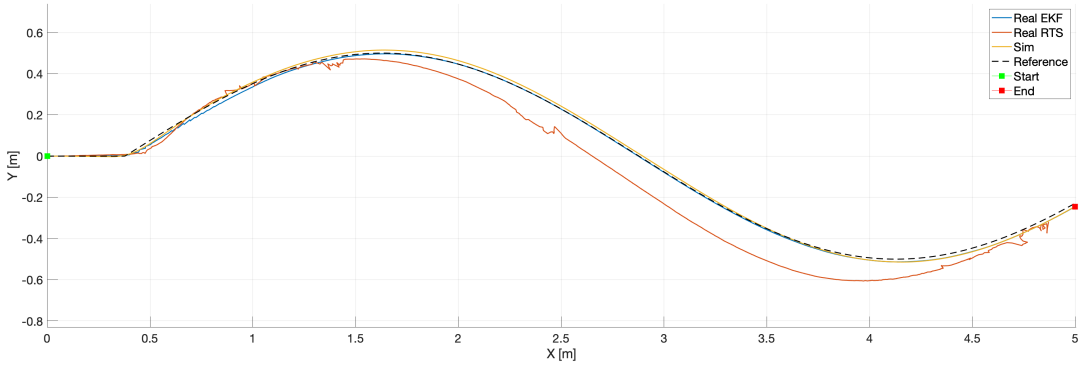
(c) ϑ error evolution.



(d) Linear velocity evolution.

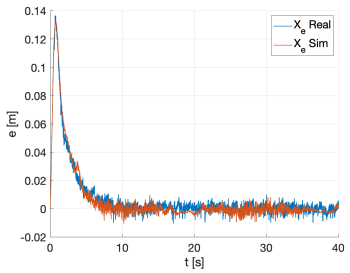


(e) Angular velocity evolution.

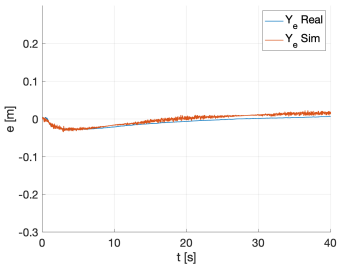


(f) Travelled trajectory comparison.

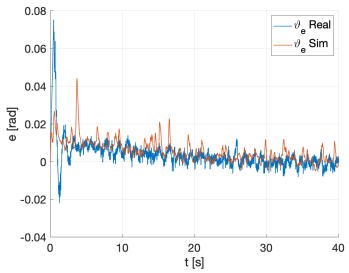
Figure 5.16: SMC experimental tracking results of a sinusoidal trajectory executed at 0.25 m/s.



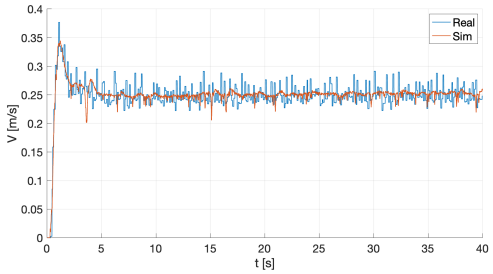
(a) X error evolution.



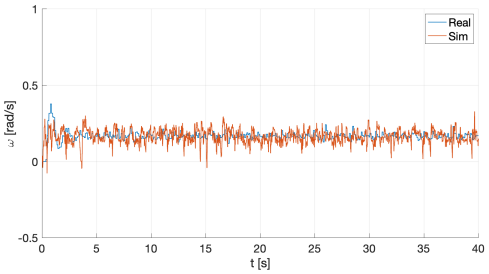
(b) Y error evolution.



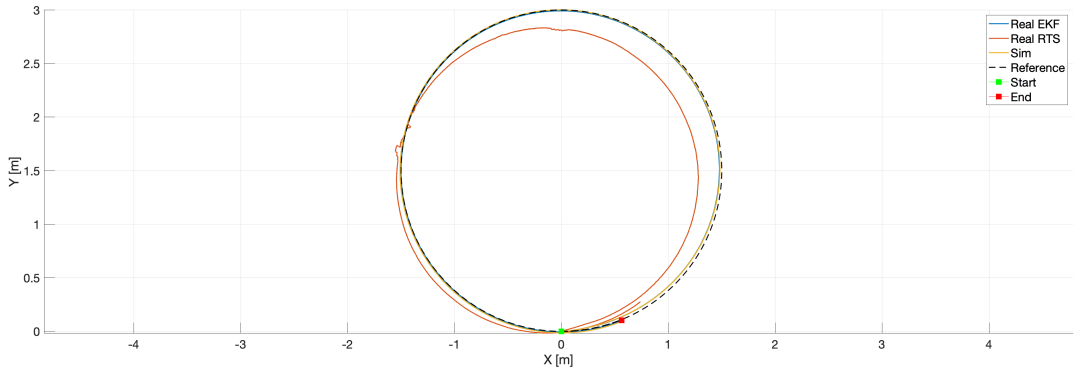
(c) ϑ error evolution.



(d) Linear velocity evolution.

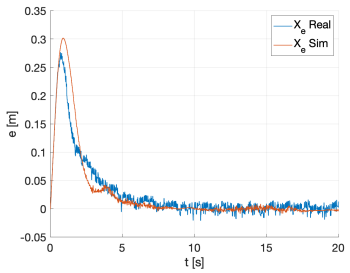


(e) Angular velocity evolution.

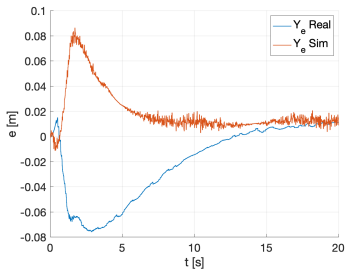


(f) Travelled trajectory comparison.

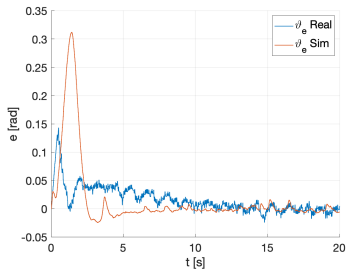
Figure 5.17: SMC experimental tracking results of a circumference executed at 0.25 m/s.



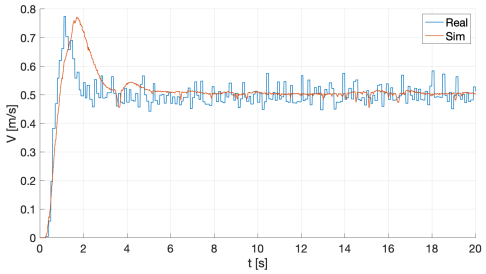
(a) X error evolution.



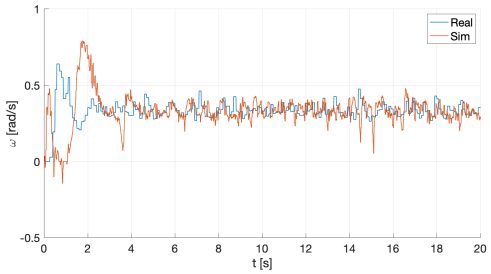
(b) Y error evolution.



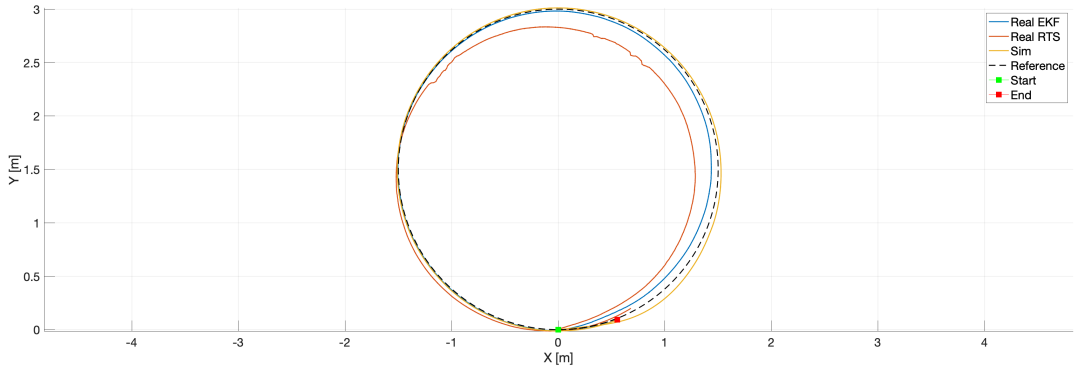
(c) ϑ error evolution.



(d) Linear velocity evolution.

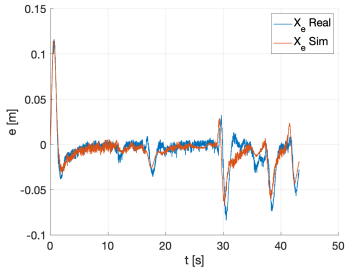


(e) Angular velocity evolution.

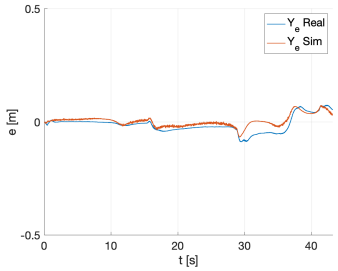


(f) Travelled trajectory comparison.

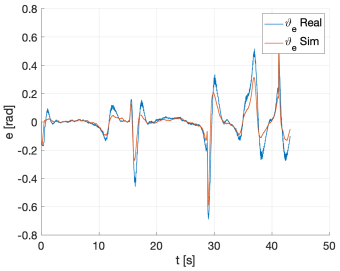
Figure 5.18: SMC experimental tracking results of a circumference executed at 0.50 m/s.



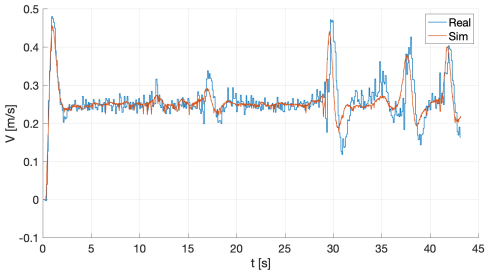
(a) X error evolution.



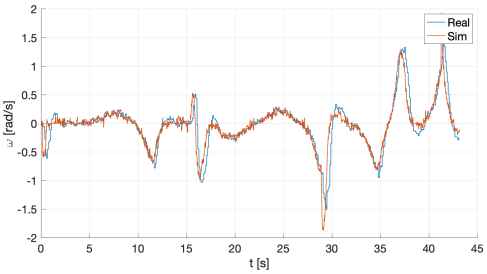
(b) Y error evolution.



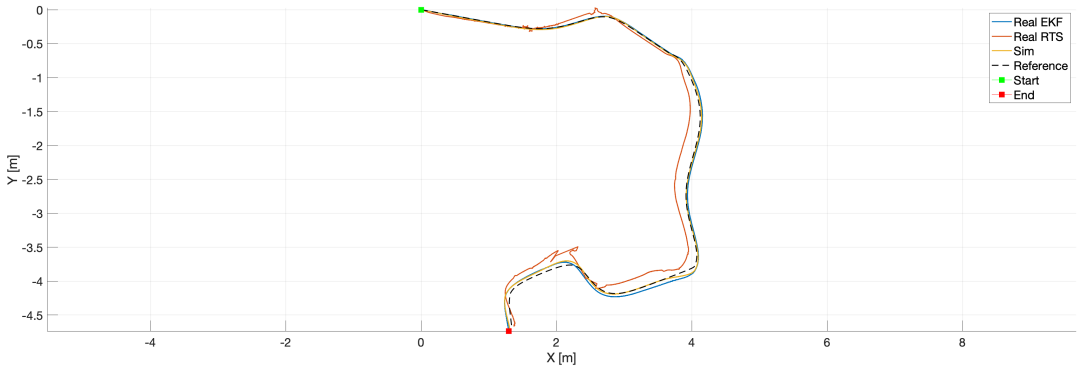
(c) ϑ error evolution.



(d) Linear velocity evolution.

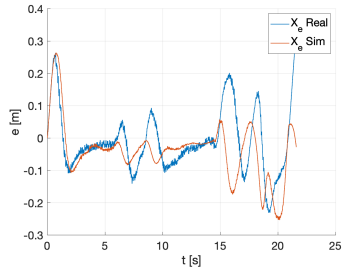


(e) Angular velocity evolution.

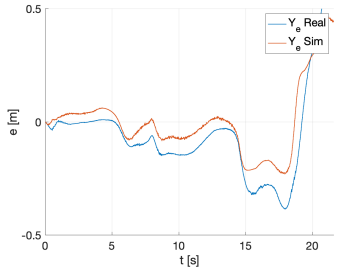


(f) Travelled trajectory comparison.

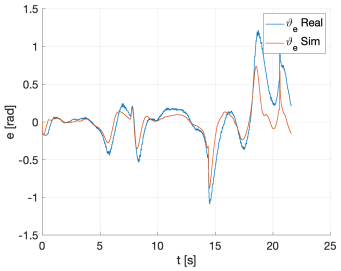
Figure 5.19: SMC experimental tracking results of a complex trajectory executed at 0.25 m/s.



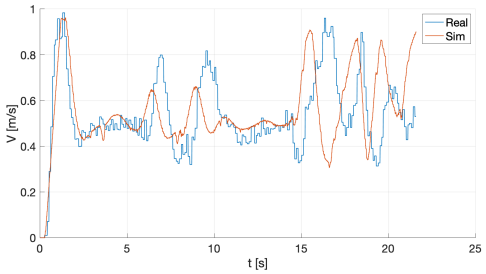
(a) X error evolution.



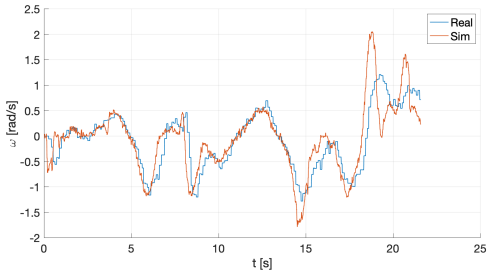
(b) Y error evolution.



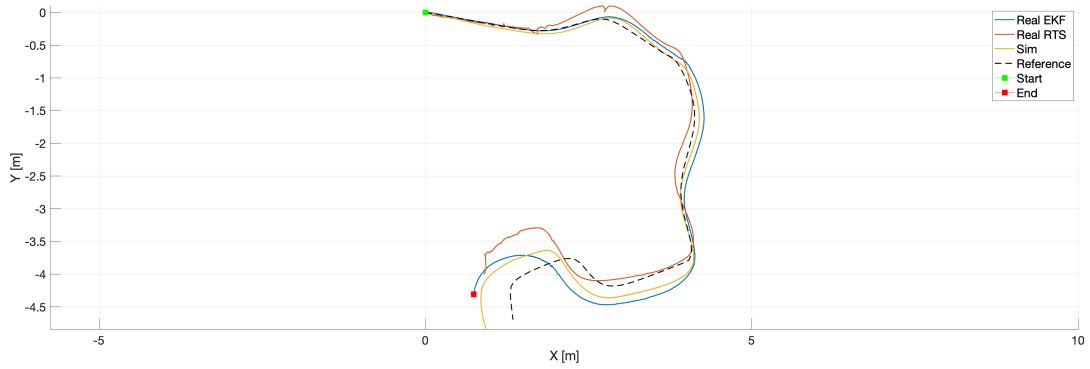
(c) ϑ error evolution.



(d) Linear velocity evolution.



(e) Angular velocity evolution.



(f) Travelled trajectory comparison.

Figure 5.20: SMC experimental tracking results of a complex trajectory executed at 0.50 m/s.

Table 5.5: SMC trajectory tracking experimental results.

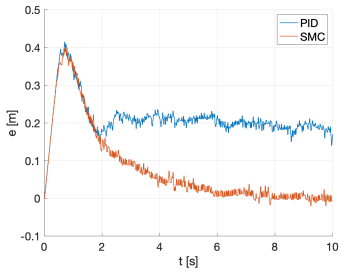
Reference Velocity	***Simulation***			***Experimental***		
	d_{ratio}	d_{haus} [m]	d_{rms} [m]	d_{ratio}	d_{haus} [m]	d_{rms} [m]
Straight line						
0.25 m/s	1.001	0.008	0.005	1.0	0.015	0.004
0.50 m/s	1.0	0.014	0.006	1.002	0.026	0.008
0.75 m/s	1.0	0.044	0.017	1.010	0.027	0.008
Circumference						
0.25 m/s	0.998	0.030	0.014	0.994	0.031	0.014
0.50 m/s	1.013	0.069	0.025	0.985	0.077	0.037
0.75 m/s	1.065	0.414	0.261	0.913	0.203	0.115
Sinusoidal trajectory						
0.25 m/s	1.004	0.018	0.012	1.003	0.020	0.011
0.50 m/s	1.002	0.017	0.012	1.003	0.023	0.009
0.75 m/s	1.003	0.146	0.061	1.006	0.071	0.038
Complex trajectory						
0.25 m/s	1.022	0.070	0.035	1.027	0.087	0.036
0.50 m/s	1.104	0.458	0.238	1.089	0.715	0.211

5.4 Controllers Comparison

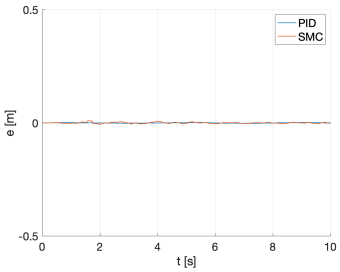
In this last section, the two trajectory tracking controllers are compared over the reference trajectories. The graphical representation was reserved only for the most notable cases, while the numerical results represented by the cost function defined in eq. 4.18, are listed in them entirety in Table 5.6. In this table are also represented the percentage changes of the fitness function results in function of reference speed Δ and between the different control structures ε. It is worth noting that for all experimental results, the controllers' performances were assessed not considering the bending energy ratio because, due to system noise, its value was not correctly calculated.

Even if both controllers have proven their ability to track a trajectory within the specified speed range, as demonstrated by the variations of the fitness function the SMC architecture is superior from every point of view when compared to the PID. Starting from robustness to speed variations, the general deterioration following an increase in the reference velocity is always lower than the PID and by analysing the linear and sinusoidal trajectory the variation between the two limit cases is less than 30% producing a satisfactory result. Regarding the circumference, even if the test at 0.75 m/s causes a worsening of the results by more than 60%, it must be considered that it represents one of the limit cases of application and if compared to the one obtained through the PID, see

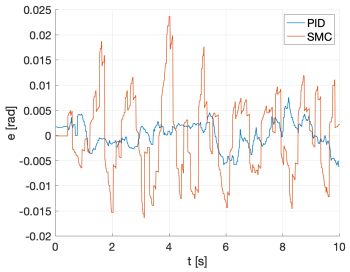
Figure 5.23, it shows the superiority of the control law as it is able to adapt the physical limits of Husky UGV to the tracking of the trajectory, positioning itself on a circular path with a lower radius defined by the reaching of the drive torque limits.



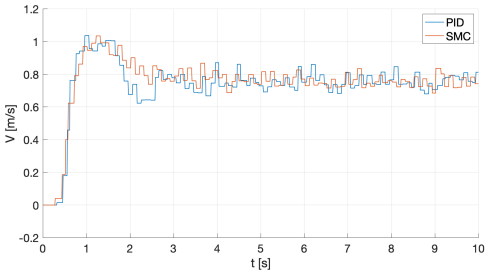
(a) X error evolution.



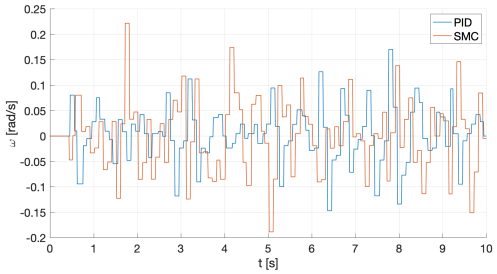
(b) Y error evolution.



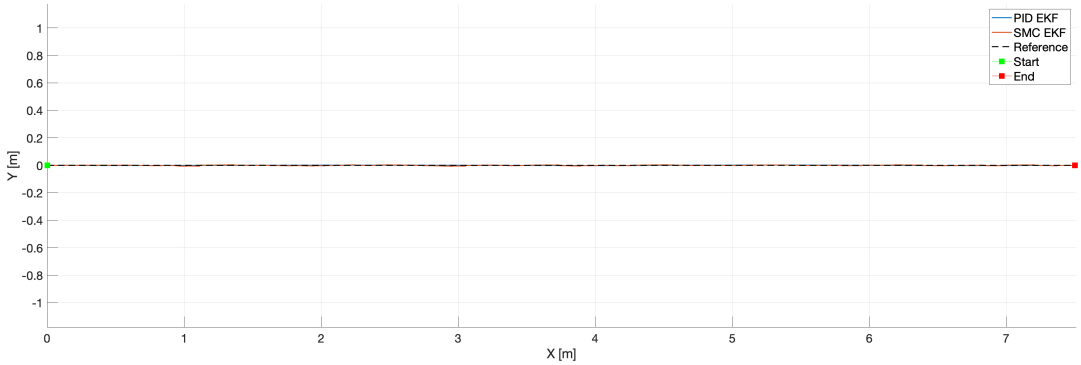
(c) ϑ error evolution.



(d) Linear velocity evolution.

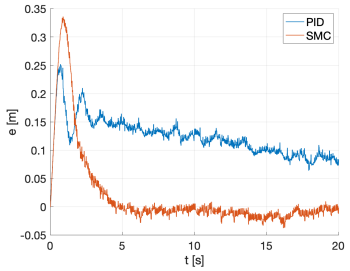


(e) Angular velocity evolution.

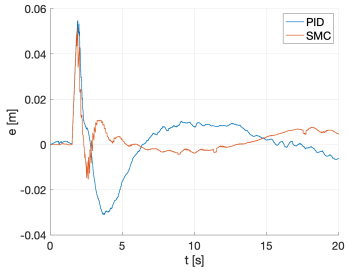


(f) Travelled trajectory comparison.

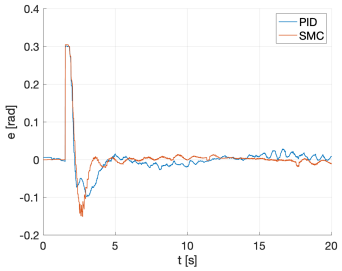
Figure 5.21: PID and SMC experimental tracking results comparison for a linear trajectory executed at 0.75 m/s.



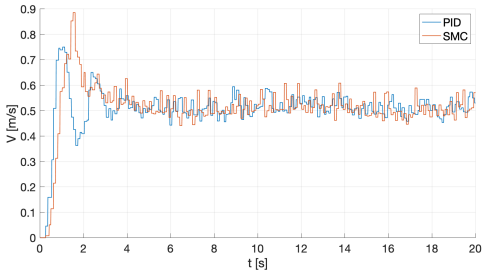
(a) X error evolution.



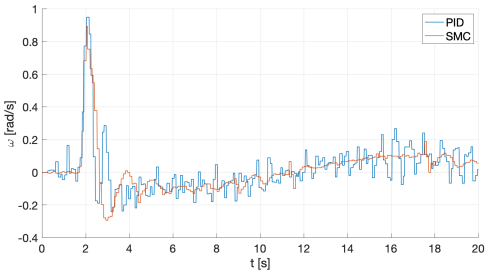
(b) Y error evolution.



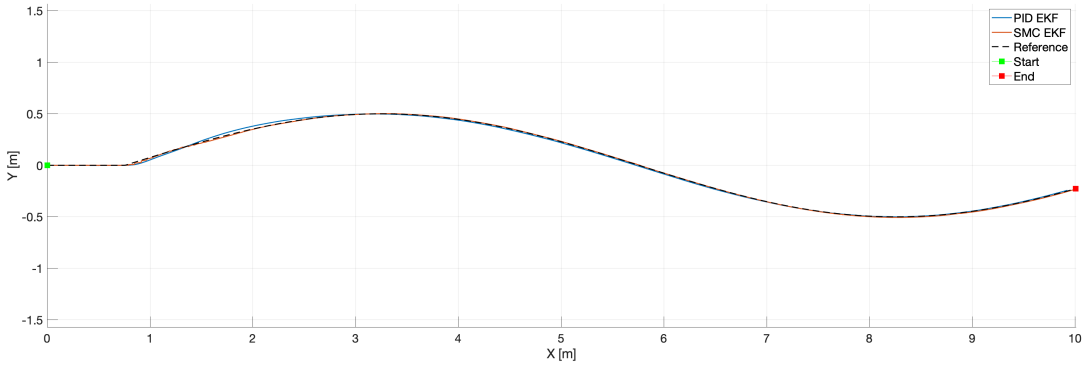
(c) ϑ error evolution.



(d) Linear velocity evolution.

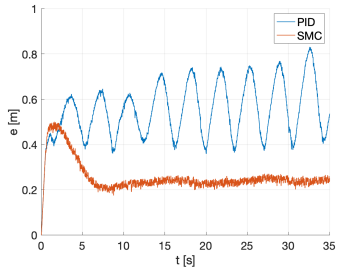


(e) Angular velocity evolution.

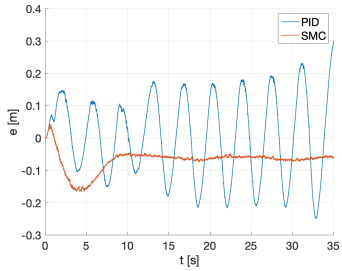


(f) Travelled trajectory comparison.

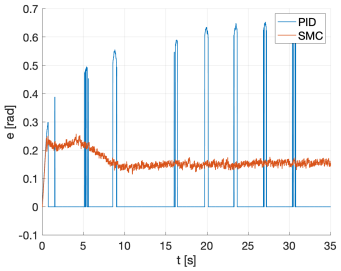
Figure 5.22: PID and SMC experimental tracking results comparison for a sinusoidal trajectory executed at 0.5 m/s.



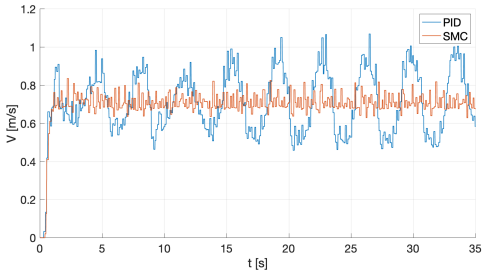
(a) X error evolution.



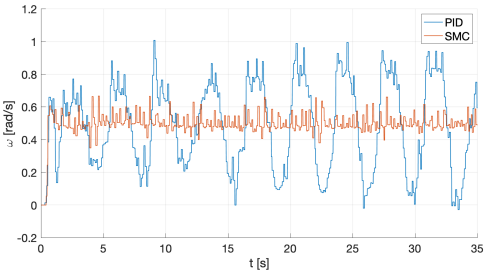
(b) Y error evolution.



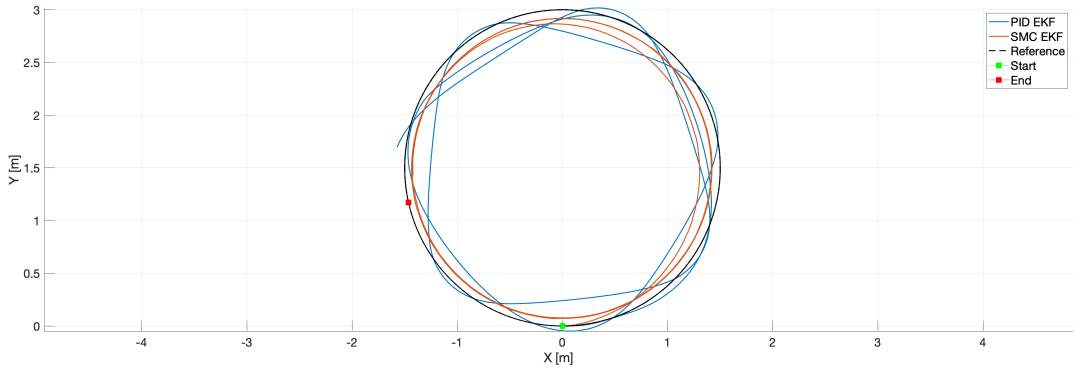
(c) ϑ error evolution.



(d) Linear velocity evolution.

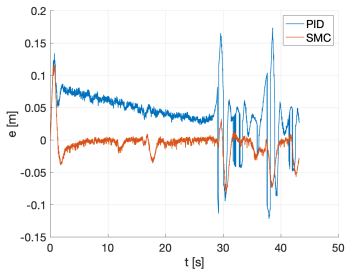


(e) Angular velocity evolution.

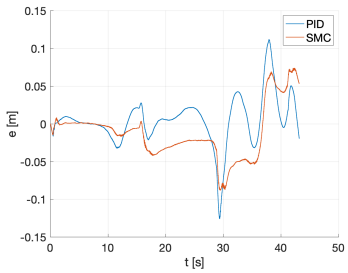


(f) Travelled trajectory comparison.

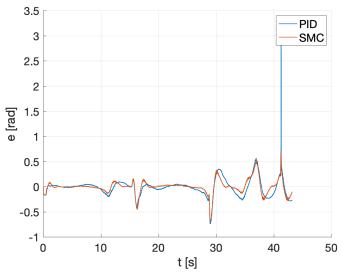
Figure 5.23: PID and SMC experimental tracking results comparison for a circumference executed at 0.75 m/s.



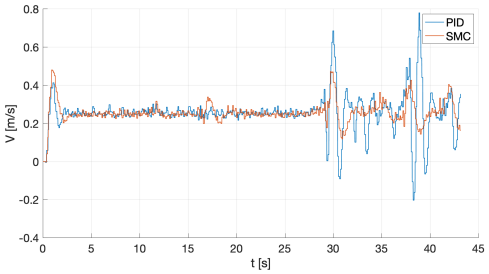
(a) X error evolution.



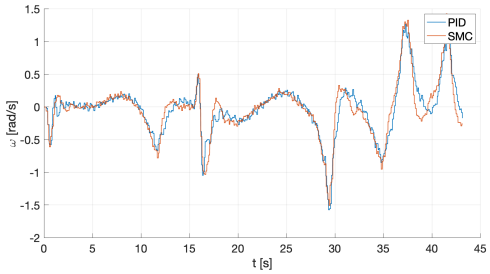
(b) Y error evolution.



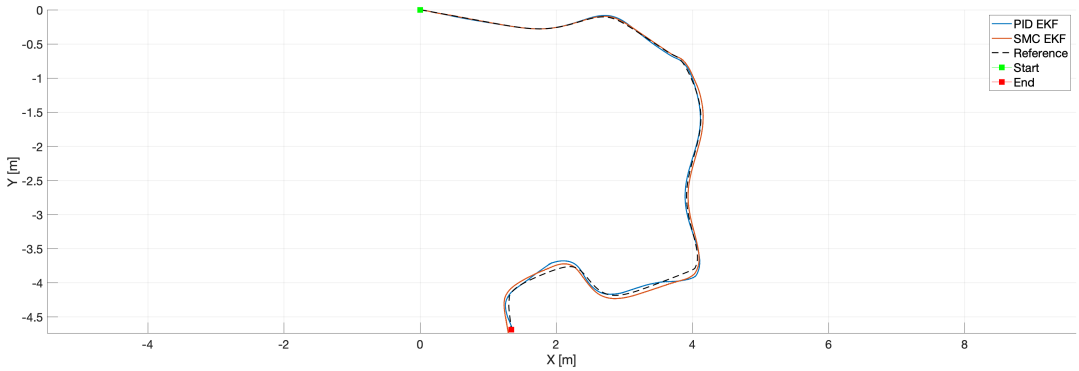
(c) ϑ error evolution.



(d) Linear velocity evolution.



(e) Angular velocity evolution.



(f) Travelled trajectory comparison.

Figure 5.24: PID and SMC experimental tracking results comparison for a complex trajectory executed at 0.25 m/s.

On the other hand, by analysing the extreme case, that is the complex trajectory performed at 0.5 m/s, it should be specified that during the SMC test a problem that arose on the stop condition caused the early end of the simulation and the following result of the maximum deviation from the trajectory, and therefore of the cost function, does not represent the actual performance of the controller. The result was however reported for the sake of completeness, but cannot be considered to draw reliable conclusions, which have to be derived from direct analysis of the graphs shown in Figures 5.12 and 5.20. In particular, it is noted that the tracking responses are almost similar but both far away

from what is the reference trajectory. In this case, the wheel actuators are operating at their limit torque condition, saturating the system. It can therefore be concluded that this condition exceeds the physical limits of the Clearpath Husky which is unable to accurately track the proposed trajectory at a constant speed greater than 0.25 m/s in the final part characterised by tight turns.

Table 5.6: PID and SMC experimental trajectories cost functions comparison.

Reference Velocity	***PID***		***SMC***		ε
	f_{opt}	Δ	f_{opt}	Δ	
Straight line					
0.25 m/s	10.9384	−−	10.6583	−−	−0.0256
0.50 m/s	11.8707	0.0852	11.1932	0.0502	−0.0571
0.75 m/s	13.1504	0.1078	11.3537	0.0143	−0.1366
Δ_{max}	0.2022		0.0653		
Circumference					
0.25 m/s	10.8903	−−	10.8211	−−	−0.0064
0.50 m/s	11.5370	0.0594	14.6738	0.3560	0.2719
0.75 m/s	21.6967	0.8806	18.0112	0.2274	−0.1699
Δ_{max}	0.9923		0.6645		
Sinusoidal trajectory					
0.25 m/s	12.4882	−−	11.3997	−−	−0.0872
0.50 m/s	12.5138	0.0020	11.4222	0.0020	−0.0872
0.75 m/s	16.0373	0.2816	14.9295	0.3071	−0.0691
Δ_{max}	0.2842		0.2847		
Complex trajectory					
0.25 m/s	15.2852	−−	15.1786	−−	−0.0233
0.50 m/s	27.3486	0.7892	42.6703	1.8112	0.5602
Δ_{max}	0.7892		1.8112		

Finally, by directly comparing the performance of the controllers concerning the same tested trajectory, ε parameter, the superiority of the SMC architecture is due, as mentioned above, to the smoother control action, see Figure 5.24 and to its insensitivity to matched uncertainties.

Chapter 6

Conclusions

The present MSc thesis illustrated the trajectory tracking problem of a 4W-SSMR in a comprehensive way, going from the definition of its mathematical model, to the design and development of two different controllers architectures, finishing with their validation through experimental test on a real robotic mobile platform: the Clearpath Husky. The two implemented controllers, namely a classic PID and a more advanced SMC, have been first introduced from their theoretical point of view and then designed also introducing innovative approaches, that have proven to produce satisfactory results. Numerical simulations and preliminary experimental tests validated their design, establishing the controllers' suitability for WMR trajectory tracking applications. Although these results have demonstrated to be promising, a more careful analysis has shown that the limited number of tests performed on the real platform leaves room for further improvements necessary to obtain a more mature design and more reliable results.

By going more into detail, the heuristic PID tuning approach described in Section 4.3.2 guarantees satisfying tracking performances even though this control technique is limited about its design point; as the achieved performances have been obtained at the expense of system robustness to parametric uncertainties, sensors noise and external disturbances. A trade-off between these two characteristics must be evaluated. On the contrary, the high-frequency variable structure of the SMC control law, proposed in Section 4.4.3, supported by its parameters tuning trough the self-adaptive evolutionary algorithm, demonstrated excellent robustness to matched uncertainness leading at the same time to more satisfying performances; thus enshrining its superiority over the well known but limited PID.

The preliminary tests carried out at BRI Robotics Testing and Training Area fully validated the conclusions drawn over the two proposed controller architectures but, as presumed by the word *'preliminary'*, these do not represent the optimum solution and give way for further improvements. In particular, the adjustment of the control parameters and a better assessment of the existing correlation between the theoretical and physical performance limits of the Husky SSMR which occur at high speeds, so that the operation of the controllers can be faithfully compared in a larger scenario.

The future development of the proposed research should be aimed at making several improvements by following different branches, obviously starting from the aforementioned controllers gain refinement. Once the optimum condition is achieved, and the robustness

problem is completely solved, this gives the possibility to carry out an actual real-time implementation of the controller embedding the control algorithm into physical hardware, testing it via Software in the Loop (SIL) or Hardware in the Loop (HIL) simulation and finally testing it on a WMR. This solution identifies the SMC as the perfect candidate thanks to its assessed better performances, its insensitivity to matched uncertainties and its already implemented method of chattering avoidance. A possible parallel development could be the improvement of the mathematical model of the SSMR considering its three-dimensional dynamics, leading to a higher level of accuracy. Moreover, with the further adoption of the here proposed advanced friction model, the 3D modelling could solve the uncertainties that occur at wheel-ground interaction level as the result of the simplifications we have made. Finally, a more in-depth study on the sensors installed, and the introduction of new ones can lead to a better dead reckoning estimation, for example by reducing the odometry drifting of the encoders and IMUs adopting sensor fusion techniques, and to new abilities such as obstacle avoidance.

Appendix A

jDE Optimization Algorithm

Differential evolution is a simple but powerful EA[1] for performing global optimization tasks first introduced by Storn in 1997 [73] which gained great popularity diverse fields, such as mechanical engineering [116,117], power systems [118], signal processing [119] and communication [120], tanks to its enhanced convergence properties [121], its improved robustness [122] and its good comprehensibility.

DE can be described as a direct parallel search method which relies on a population of NP D-dimensional parameter vectors called *individuals* which encode the candidate solutions:

$$\boldsymbol{\mathcal{X}}_{i,j} = \left\{x_{i,j}^1, \ldots, x_{i,j}^D\right\}, \quad \text{for} \quad i = 1, \ldots, NP \; and \; j = 1, \ldots, NG \tag{A.1}$$

where NG is the number of generations.

The initial population should contain uniformly randomized initial individuals for covering as much as possible the entire search space constrained by the prescribed minimum and maximum parameter bounds:

$$\begin{aligned} \boldsymbol{\mathcal{X}}_{min} &= \left\{x_{min}^1, \ldots, x_{min}^D\right\} \\ \boldsymbol{\mathcal{X}}_{max} &= \left\{x_{max}^1, \ldots, x_{max}^D\right\} \end{aligned} \tag{A.2}$$

for example, the initial value of the jth parameter of the ith individual at generation $NG = 0$ is given by:

$$x_{i,0}^k = x_{min}^k + rand\,(0,1)\left(x_{max}^k - x_{min}^k\right), \quad \text{for} \quad k = 1, \ldots, D \tag{A.3}$$

where $rand\,(0,1)$ represents a uniformly distributed random constant in the range [0, 1]. After the initialization, according to [73] the algorithm generates trial parameter vectors through the succession of three operations: *mutation* and *crossover* for generating new

[1] EAs represent a wide class of stochastic optimization algorithms inspired by biological processes, like survival of the fittest or genetic inheritance, that allow the adaptability of a population of organisms to their surrounding environment. Their biggest advantage over other types of optimization approaches is the only need to know of the objective function to be minimized.

trial candidates, and *selection* for determining which among these will survive into the next generation. At each generation, the fitness of the population is compared with the one of the mutation-derived trial vectors by means of an objective function f_{opt}. The vectors giving the minimum value of the cost function move to the next generation, until a stop criterion is achieved or the final generation is reached. Going into detail of each operation:

1. *Mutation:* This operation is carried out by the DE algorithm after the initialization, and consist in the production of NP mutant vectors $\boldsymbol{\mathcal{V}}_{i,j}$ which target each individual $\boldsymbol{\mathcal{X}}_{i,j}$ in the current population. Within a generic NG generation, for each target vector its associated mutant vector, defined as:

$$\boldsymbol{\mathcal{V}}_{i,NG} = \left\{v^1_{i,NG}, \ldots, v^D_{i,NG}\right\}, \quad \text{for} \;\; i = 1, \ldots, NP \tag{A.4}$$

, can be generated via several mutation strategies, some of the most commonly used are described in [120]. In this thesis, strategy named *DE/best/2* has been chosen due to its ability to improve diversity by producing more trial vectors. This strategy, evaluates the mutant vector introducing two difference vectors as a perturbation:

$$\boldsymbol{\mathcal{V}}_{i,j} = \boldsymbol{\mathcal{X}}_{best,j} + F\left\{\boldsymbol{\mathcal{X}}_{r^i_1,j} - \boldsymbol{\mathcal{X}}_{r^i_2,j}\right\} + F\left\{\boldsymbol{\mathcal{X}}_{r^i_3,j} - \boldsymbol{\mathcal{X}}_{r^i_4,j}\right\} \tag{A.5}$$

for $i = 1, \ldots, NP$ and $j = 1, \ldots, NG$. $\boldsymbol{\mathcal{X}}_{best,j}$ represents the individual with best fitness in the population at jth generation and the indices $r^i_1,\ r^i_2,\ r^i_3,\ r^i_4$ are mutually exclusive integers randomly generated once for each mutant vector within the range $[1,\ NP]$. F is a positive control parameter used for scaling the difference vector.

2. *Crossover:* After mutation, to each pair of target vector and its corresponding mutation is applied the crossover operation. This phase generates a trial vector:

$$\boldsymbol{\mathcal{U}}_{i,j} = \left\{u^1_{i,j}, \ldots, u^D_{i,j}\right\}, \quad \text{for} \;\; i = 1, \ldots, NP \; and \; j = 1, \ldots, NG \tag{A.6}$$

defined by the binomial equation:

$$u^k_{i,j} = \begin{cases} v^k_{i,j}, & \text{if} \;\; rand_k\,(0,\ 1) \leq CR \; or \; k = k_{rand} \\ x^k_{i,j}, & \text{otherwise} \end{cases} \tag{A.7}$$

, for $i = 1, \ldots, NP,\ j = 1, \ldots, NG$ and $k = 1, \ldots, D$. CR represents the crossover rate, a user-specified constant in the range [0, 1] which controls the fraction of parameter taken from mutation; while k_{rand} is a randomly integer taken in the range $[1,\ D]$. This operation defines, through the occurrence or not of a specified condition, which trial vector parameters derive from the corresponding target or mutant vector. The condition $k = k_{rand}$ is introduced only to ensure that the trial vector will differ from its corresponding target vector by at least one parameter.

3. *Selection:* This operation is performed after having reinitialized to a random and acceptable value the trial vector parameters that exceeds the specified bounds and after having evaluated the objective functions of all trial vectors. After that, the

selection process consist, for a generic jth generation in the current ith population, in comparing the fitness function deriving from each trial vector $f(\boldsymbol{\mathcal{U}}_{i,j})$ to the one of the corresponding target vector $f(\boldsymbol{\mathcal{X}}_{i,j})$. If the fitness of the trial vector is better, its value replaces the corresponding one in the target vector, gaining the entry in the population of the next generation:

$$\boldsymbol{\mathcal{X}}_{i,j+1} = \begin{cases} \boldsymbol{\mathcal{U}}_{i,j}, & \text{if } f(\boldsymbol{\mathcal{U}}_{i,j}) \leq f(\boldsymbol{\mathcal{X}}_{i,j}) \\ \boldsymbol{\mathcal{X}}_{i,j}, & \text{otherwise} \end{cases} \tag{A.8}$$

Having selected the NP and NG values, which depend on the size of the problem, it is necessary to identify the remaining control parameters values, namely the mutation scale factor F, and crossover rate CR. Since the value of these parameters highly affects the performance of the optimisation algorithm, choosing suitable values generally requires a time-consuming trial-and-error tuning process. This approach is not appropriate if the optimisation is required in an automated process, like the one which should be applied in this thesis. To overcome this issue, a parameter self-adaptive approach, based on [77] and named jDE, have been proposed. Here, the control parameters F and CR, initialized for each individual at the same value, are encoded into the individuals and are adjusted by introducing two new parameters τ_1 and τ_2. The most suitable individual carries to the next generations the better parameter which, in turn, are more likely to produce offspring capable of propagating these F and CR values to the next generations. In this thesis, a variant of the classic jDE algorithm proposed by Brest is employed, in which if F and CR are not smaller than a randomly generated number, used for the adaptation process, they are evaluated using a gradient descent approach instead of being kept unchanged. So, the factors for the following parent vector can be derived as follows:

$$F_{i,j+1} = \begin{cases} F_l + rand_1 F_u & \text{,if } rand_2 \leq \tau_1 \\ F_{i,j} - F_{i,j}\xi\frac{j-1}{NG} & \text{,if } rand_2 > \tau_1 \wedge F_{i,j+1} \geq F_l \\ F_l & \text{,otherwise} \end{cases} \tag{A.9}$$

$$CR_{i,j+1} = \begin{cases} rand_3 & \text{,if } rand_4 \leq \tau_2 \\ CR_{i,j} - CR_{i,j}\xi\frac{j-1}{NG} & \text{,if } rand_2 > \tau_1 \wedge CR_{i,j+1} \geq CR_{min} \\ CR_{min} & \text{,otherwise} \end{cases} \tag{A.10}$$

where ξ is the constant slope gradient, $rand_n$, with $n = 1,\ldots,4$, are uniform random values in the range $[0,\ 1]$ and F_u and F_l respectively indicate the upper and lower limits of the mutation scale factor F. In such a way, a flexible DE scheme is achieved and no tuning of the control parameters is required.

The choice to use this algorithm has been based on its good convergence performances, its improved robustness and due to the extensive related work already conducted by the hosting University. The listing of the proposed algorithm is reported in Algorithm A.1.

Algorithm A.1 jDE/Best/2 Optimization

Input: NP, NG, τ_1, τ_2, F bounds, CR bounds
Output: $\boldsymbol{\mathcal{X}}_{best}$

Set $j \leftarrow 0$
Randomly initialize a population of NP individuals $\boldsymbol{\mathcal{X}}_{i,j}$
while stopping criteria is NOT satisfied **do**
    procedure PERFORM MUTATION TO GENERATE NP MUTANT VECTORS($\boldsymbol{\mathcal{V}}_{i,j}$)
        for $i \leftarrow 1$ to NP **do**
            Randomly select $r_1^i \neq r_2^i \neq r_3^i \neq r_4^i$
            procedure RANDOMLY GENERATE MUTATION SCALE FACTOR PARAMETERS($F_{i,j+1}$)
                for $n \leftarrow 1$ to 4 **do**
                    Randomly generate $rand_n$ parameter indexes in $[0,\ 1]$
                end for
                if $rand_2 \leq \tau_1$ **then**
                    $F_{i,j+1} \leftarrow F_l + rand_1 F_u$
                else if $rand_2 > \tau_1 \wedge F_{i,j+1} \geq F_l$ **then**
                    $F_{i,j+1} \leftarrow F_{i,j} - F_{i,j}\xi\frac{j-1}{NG}$
                else
                    $F_{i,j+1} \leftarrow F_{min}$
                end if
            end procedure
            $\boldsymbol{\mathcal{V}}_{i,j} \leftarrow \boldsymbol{\mathcal{X}}_{best,j} + F_{i,j}\left\{\boldsymbol{\mathcal{X}}_{r_1^i,j} - \boldsymbol{\mathcal{X}}_{r_2^i,j}\right\} + F_{i,j}\left\{\boldsymbol{\mathcal{X}}_{r_3^i,j} - \boldsymbol{\mathcal{X}}_{r_4^i,j}\right\}$
        end for
    end procedure

    procedure PERFORM CROSSOVER TO GENERATE NP TRIAL VECTORS($\boldsymbol{\mathcal{U}}_{i,j}$)
        for $i \leftarrow 1$ to NP **do**
            procedure RANDOMLY GENERATE CROSSOVER RATE PARAMETERS($CR_{i,j+1}$)
                if $rand_4 \leq \tau_2$ **then**
                    $CR_{i,j+1} \leftarrow rand_3$
                else if $rand_4 > \tau_2 \wedge CR_{i,j+1} \geq CR_{min}$ **then**
                    $CR_{i,j+1} \leftarrow CR_{i,j} - CR_{i,j}\xi\frac{j-1}{NG}$
                else
                    $CR_{i,j+1} \leftarrow CR_{min}$
                end if
            end procedure
            for $k \leftarrow 1$ to D **do**
                if $rand_k\,(0,\ 1) \leq CR_{i,j} \vee k \leftarrow k_{rand}$ **then**
                    $u_{i,j}^k \leftarrow v_{i,j}^k$
                else

▷ *Continued on next page*

Algorithm A.1 – *Continued from previous page*

```
                    u^k_{i,j} ← x^k_{i,j}
                end if
            end for
        end for

    end procedure
    procedure PERFORM SELECTION TO CHOOSE THE NEXT GENERATION NP INDIVIDUALS(X_{i,j+1})
        for i ← 1 to NP do
            Evaluate the objective functions f(U_{i,j}) and f(X_{i,j})
            if f(U_{i,j}) ≤ f(X_{i,j}) then
                X_{i,j+1} ← f(U_{i,j})
                if f(U_{i,j}) < f(X_{i,j}) then
                    X_{best,j} ← f(U_{i,j})
                end if
            else
                X_{i,j+1} ← f(X_{i,j})
            end if
        end for
    end procedure
    j ← j + 1
end while
```

As can be seen from Algorithm A.1, it seems that the proposed self-adaptive algorithm has more control parameters than the three of the classic DE; but, analysing it more carefully, can be seen see that τ_1, τ_2, ξ, F_l, F_u and CR bounds have all fixed values, so the user does not need to adjust those additional parameters. This choice is justified by a study made in [77], in which the algorithm performances have been assessed, through the use of 23 different benchmark functions, changing the control parameters values. The results obtained have not shown any significant difference, thus giving validity to this decision. Considering our application, the chosen fixed control parameters values are shown in the Table A.1.

Table A.1: jDE control parameters.

Parameters	**Values**	**Parameters**	**Values**
$\boldsymbol{\xi}$	0.2	$\boldsymbol{F_l}$	0.1
df	0.001	$\boldsymbol{F_u}$	1.0
$\boldsymbol{\tau_1}$	0.1	$\boldsymbol{CR_{min}}$	0.0
$\boldsymbol{\tau_2}$	0.1	$\boldsymbol{CR_{max}}$	1.0

It is important to note the choice of mutation scale factor lower limit F_l; if the control parameter $F = 0$, the new trial vector will be generated using crossover but no mutation so, to avoid this perspective, its minimum has been limited to 0.1.

Bibliography

[1] M. Teulieres, J. Tilley, L. Bolz, P. M. Ludwig-Dehm, and S. Wägner, "Industrial robotics – insights into the sector's future growth dynamics," 2019.

[2] United State Department of Defense, "Unmanned Systems Roadmap 2007 – 2032," Tech. Rep., 2007.

[3] Radio Corporation Of America, "World Wide Wireless," vol. 2, 1921. [Online]. Available: https://archive.org/details/WorldWideWirelessV2/page/n7

[4] A. Finn and S. Scheding, *Developments and Challenges for Autonomous Unmanned Vehicles*, ser. Intelligent Systems Reference Library, J. Kacprzyk and L. C. Jain, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 3. [Online]. Available: http://link.springer.com/10.1007/978-3-642-10704-7

[5] W. Grey Walter, "An Imitation of Life," 1948.

[6] Nils J. Nilsson, "A MOBILE AUTOMATON:AN APPLICATION OF ARTIFICIAL INTELLIGENCE TECHNIQUES," Jan. 1969.

[7] R. S. Mosher, "Exploring the Potential of a Quadruped," Feb. 1969, p. 690191. [Online]. Available: https://www.sae.org/content/690191/

[8] J.J. KESSIS, "Systemes de perception et de decision pour robots mobiles en univers accidente," 1984.

[9] M. H. RAIBERT, "Dynamically stable legged locomotion," 1983.

[10] N. J. Nilsson, *The quest for artificial intelligence: a history of ideas and achievements*. Cambridge ; New York: Cambridge University Press, 2010, oCLC: ocn396185265.

[11] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, Oct. 1979. [Online]. Available: http://portal.acm.org/citation.cfm?doid=359156.359164

[12] Richard E. Fikes and Nils J. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," 1971.

[13] H. P. Moravec, "Obstacle Avoidance and Navigatio n in the Real World by a Seeing Robot Rover," Sep. 1980.

[14] Rodney Albert Schmidt, "A study of the real-time control of a computer-driven vehicle," 1971.

[15] A.M. THOMPSON, "The navigation system of the JPL robot," 1977.

[16] H. P. Moravec, "The Stanford Cart and the CMU Rover," *Proceedings of the IEEE*, vol. 71, no. 7, pp. 872–884, 1983. [Online]. Available: http://ieeexplore.ieee.org/document/1456952/

[17] Thorpe, C.E, "Vision and Navigation for The Carnegie Mellon Navlab," 1990.

[18] Georges Giralt, Ralph P. Sobek, and Raja Chatila, "A multi-level planning and navigation system for a mobile robot: a first approach to HILARE," 1979.

[19] G. Giralt, R. Chatila, and M. Vaisset, "An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots," in *Autonomous Robot Vehicles*, I. J. Cox and G. T. Wilfong, Eds. New York, NY: Springer New York, 1990, pp. 420–443. [Online]. Available: http://link.springer.com/10.1007/978-1-4613-8997-2_31

[20] S. Sekhavat, F. Lamiraux, J. Laumond, G. Bauzil, and A. Ferrand, "Motion planning and control for Hilare pulling a trailer: experimental issues," in *Proceedings of International Conference on Robotics and Automation*, vol. 4. Albuquerque, NM, USA: IEEE, 1997, pp. 3306–3311. [Online]. Available: http://ieeexplore.ieee.org/document/606793/

[21] N. R. C. (U.S.) and N. R. C. (U.S.), Eds., *Technology development for Army unmanned ground vehicles.* Washington, D.C: National Academies Press, 2002, oCLC: ocm51746797.

[22] R. Siegwart and I. R. Nourbakhsh, *Introduction to autonomous mobile robots*, ser. Intelligent robots and autonomous agents. Cambridge, Mass: MIT Press, 2004.

[23] C. Ilas, "Electronic sensing technologies for autonomous ground vehicles: A review," in *2013 8TH INTERNATIONAL SYMPOSIUM ON ADVANCED TOPICS IN ELECTRICAL ENGINEERING (ATEE).* Bucharest, Romania: IEEE, May 2013, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/6563528/

[24] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard, "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems," in *The 11th International Conference on Advanced Robotics*, 2003, pp. 317–323.

[25] M. Montemerlo, N. Roy, and S. Thrun, "Perspectives on standardization in mobile robot progranuning : the carnegie mellon navigation (carmen) toolkit," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 3. Las Vegas, Nevada, USA: IEEE, 2003, pp. 2436–2441. [Online]. Available: http://ieeexplore.ieee.org/document/1249235/

[26] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob C. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA), 2009*, Kobe, Japan, May 2009.

[27] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 189–208, Dec. 2008. [Online]. Available: http://link.springer.com/10.1007/s11721-008-0014-4

[28] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3. Sendai, Japan: IEEE, 2004, pp. 2149–2154. [Online]. Available: http://ieeexplore.ieee.org/document/1389727/

[29] Y. Khosiawan and I. Nielsen, "A system of UAV application in indoor environment,"

Production & Manufacturing Research, vol. 4, no. 1, pp. 2–22, Jan. 2016. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/21693277.2016.1195304

[30] L. Borzemski, J. Świątek, and Z. Wilimowska, Eds., *Information Systems Architecture and Technology: Proceedings of 39th International Conference on Information Systems Architecture and Technology – ISAT 2018: Part I*, ser. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2019, vol. 852. [Online]. Available: http://link.springer.com/10.1007/978-3-319-99981-4

[31] *Robotic exploration of the solar system.* New York, NY: Springer Berlin Heidelberg, 2016.

[32] S. Chowdhury, A. Emelogu, M. Marufuzzaman, S. G. Nurre, and L. Bian, "Drones for disaster response and relief operations: A continuous approximation model," *International Journal of Production Economics*, vol. 188, pp. 167–184, Jun. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925527317301172

[33] United State Department of Defense, "Joint Roboticcs Program Master Plan FY2004," Tech. Rep., 2004.

[34] K. Koz\lowski and D. Pazderski, "Modeling and control of a 4-wheel skid-steering mobile robot," *International journal of applied mathematics and computer science*, vol. 14, pp. 477–496, 2004.

[35] Jingang Yi, Junjie Zhang, Dezhen Song, and Suhada Jayasuriya, "IMU-based localization and slip estimation for skid-steered mobile robots," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Diego, CA, USA: IEEE, Oct. 2007, pp. 2845–2850. [Online]. Available: http://ieeexplore.ieee.org/document/4399477/

[36] J. Yi, D. Song, J. Zhang, and Z. Goodwin, "Adaptive Trajectory Tracking Control of Skid-Steered Mobile Robots," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. Rome, Italy: IEEE, Apr. 2007, pp. 2605–2610. [Online]. Available: http://ieeexplore.ieee.org/document/4209476/

[37] L. Caracciolo, A. de Luca, and S. Iannitti, "Trajectory tracking control of a four-wheel differentially driven mobile robot," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 4. Detroit, MI, USA: IEEE, 1999, pp. 2632–2638. [Online]. Available: http://ieeexplore.ieee.org/document/773994/

[38] A. Mandow, J. L. Martinez, J. Morales, J. L. Blanco, A. Garcia-Cerezo, and J. Gonzalez, "Experimental kinematics for wheeled skid-steer mobile robots," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Diego, CA, USA: IEEE, Oct. 2007, pp. 1222–1227. [Online]. Available: http://ieeexplore.ieee.org/document/4399139/

[39] J. L. Martínez, A. Mandow, J. Morales, S. Pedraza, and A. García-Cerezo, "Approximating Kinematics for Tracked Mobile Robots," *The International Journal of Robotics Research*, vol. 24, no. 10, pp. 867–878, Oct. 2005. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364905058239

[40] C. C. Wit, B. Siciliano, and G. Bastin, *Theory of Robot Control*. London:

Springer London, 1996, oCLC: 853263033. [Online]. Available: https://doi.org/10.1007/978-1-4471-1501-4

[41] W. Khalil and E. Dombre, *Modeling, identification & control of robots*, ser. Kogan Page Science paper edition. London ; Sterling, VA: Kogan Page Science, 2004, oCLC: ocm56760376.

[42] G. Oriolo, A. De Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: design, implementation, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 6, pp. 835–852, Nov. 2002. [Online]. Available: http://ieeexplore.ieee.org/document/1058053/

[43] W. Yu, O. Chuy, E. G. Collins, and P. Hollis, "Dynamic modeling of a skid-steered wheeled vehicle with experimental verification," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. St. Louis, MO, USA: IEEE, Oct. 2009, pp. 4212–4219. [Online]. Available: http://ieeexplore.ieee.org/document/5354381/

[44] Wei Yu, O. Chuy, E. Collins, and P. Hollis, "Analysis and Experimental Verification for Dynamic Modeling of A Skid-Steered Wheeled Vehicle," *IEEE Transactions on Robotics*, vol. 26, no. 2, pp. 340–353, Apr. 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5427035/

[45] K. Kozlowski and D. Pazderski, "Practical Stabilization of a Skid-steering Mobile Robot - A Kinematic-based Approach," in *2006 IEEE International Conference on Mechatronics*. Budapest: IEEE, Jul. 2006, pp. 519–524. [Online]. Available: https://ieeexplore.ieee.org/document/4018416/

[46] E. Maalouf, M. Saad, and H. Saliah, "A higher level path tracking controller for a four-wheel differentially steered mobile robot," *Robotics and Autonomous Systems*, vol. 54, no. 1, pp. 23–33, Jan. 2006. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889005001454

[47] Anh Tuan Le, D. Rye, and H. Durrant-Whyte, "Estimation of track-soil interactions for autonomous tracked vehicles," in *Proceedings of International Conference on Robotics and Automation*, vol. 2. Albuquerque, NM, USA: IEEE, 1997, pp. 1388–1393. [Online]. Available: http://ieeexplore.ieee.org/document/614331/

[48] S. Moosavian and A. Kalantari, "Experimental slip estimation for exact kinematics modeling and control of a Tracked Mobile Robot," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nice: IEEE, Sep. 2008, pp. 95–100. [Online]. Available: http://ieeexplore.ieee.org/document/4650798/

[49] G. Anousaki and K. Kyriakopoulos, "Simultaneous localization and map building of skid-steered robots," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 79–89, Mar. 2007. [Online]. Available: http://ieeexplore.ieee.org/document/4141036/

[50] ——, "A dead-reckoning scheme for skid-steered vehicles in outdoor environments," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004.* New Orleans, LA, USA: IEEE, 2004, pp. 580–585 Vol.1. [Online]. Available: http://ieeexplore.ieee.org/document/1307211/

[51] Y. Wu, T. Wang, J. Liang, J. Chen, Q. Zhao, X. Yang, and C. Han, "Experimental kinematics modeling estimation for wheeled skid-steering mobile robots," in *2013 IEEE International Conference on Robotics and Biomimetics*

(ROBIO). Shenzhen, China: IEEE, Dec. 2013, pp. 268–273. [Online]. Available: http://ieeexplore.ieee.org/document/6739470/

[52] S. Jeon and W. Jeong, "The Stable Trajectory Tracking Control of a Skid-Steered Mobile Platform with Dynamic Uncertainties," *International Journal of Advanced Robotic Systems*, vol. 12, no. 9, p. 122, Sep. 2015. [Online]. Available: http://journals.sagepub.com/doi/10.5772/61241

[53] Lorenzo Borello and Matteo D. L. Dalla Vedova, "Dry Friction Discontinuous Computational Algorithms," *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 3, no. 8, Feb. 2014.

[54] A. Ansuategui, A. Arruti, L. Susperregi, Y. Yurramendi, E. Jauregi, E. Lazkano, and B. Sierra, "Robot Trajectories Comparison: A Statistical Approach," *The Scientific World Journal*, vol. 2014, pp. 1–13, 2014. [Online]. Available: http://www.hindawi.com/journals/tswj/2014/298462/

[55] M. Buehler, K. Iagnemma, S. Singh, B. Siciliano, O. Khatib, and F. Groen, Eds., *The 2005 DARPA Grand Challenge: The Great Robot Race*, ser. Springer Tracts in Advanced Robotics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, vol. 36. [Online]. Available: http://link.springer.com/10.1007/978-3-540-73429-1

[56] "RoboCup@Home: Scientific Competition and Benchmarking for Domestic Service Robots," *Interaction Studies*, vol. 10, no. 3, pp. 392–426, Dec. 2009. [Online]. Available: https://benjamins.com/catalog/is.10.3.06wis

[57] Gabriele Ermacora, Daniele Sartori, Ling Pei, and Wenxian Yu, "An Evaluation Framework for the Deployment of Mobile Robots Performing Real-World Indoor Autonomous Navigation.pdf."

[58] E. Saggini, E. Zereik, M. Bibuli, G. Bruzzone, M. Caccia, and E. Riccomagno, "Performance Indices for Evaluation and Comparison of Unmanned Marine Vehicles' Guidance Systems," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 12 182–12 187, 2014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1474667016435536

[59] N. D. Munoz Ceballos, J. Alejandro, and N. Londono, "Quantitative Performance Metrics for Mobile Robots Navigation," in *Mobile Robots Navigation*, A. Barrera, Ed. InTech, Mar. 2010. [Online]. Available: http://www.intechopen.com/books/mobile-robots-navigation/quantitative-performance-metrics-for-mobile-robots-navigation

[60] E. Aguirre and A. González, "Fuzzy behaviors for mobile robot navigation: design, coordination and fusion," *International Journal of Approximate Reasoning*, vol. 25, no. 3, pp. 255–289, Nov. 2000. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0888613X00000566

[61] R. C. Dorf and R. H. Bishop, *Modern control systems*, 12th ed. Prentice Hall: Pearson, 2010.

[62] G. Cohen and G. Coon, "Theoretical consideration of retarded control," *Transactions of ASM*, vol. 75, no. 1, pp. 827–834, 1953.

[63] J. G. Ziegler and N. B. Nichols, "Optimum Settings for Automatic Controllers," *Journal of Dynamic Systems, Measurement, and*

Control, vol. 115, no. 2B, pp. 220–222, Jun. 1993. [Online]. Available: https://asmedigitalcollection.asme.org/dynamicsystems/article/115/2B/220/417448/Optimum-Settings-for-Automatic-Controllers

[64] W. Ho, C. Hang, and L. Cao, "Tuning of PID Controllers based on Gain and Phase Margin Specifications," *IFAC Proceedings Volumes*, vol. 26, no. 2, pp. 199–202, Jul. 1993. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1474667017484539

[65] T. Jain and M.J. Nigam, "Optimisation of PD-PI Controller Using Swarm Intelligence," *Journal of Theoretical and Applied Information Technology*, vol. 4, no. 11, pp. 1013–1018, 2008.

[66] K. Åström and T. Hägglund, "Automatic tuning of simple regulators with specifications on phase and amplitude margins," *Automatica*, vol. 20, no. 5, pp. 645–651, Sep. 1984. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0005109884900141

[67] C. Hang, K. Åström, and W. Ho, "Refinements of the Ziegler–Nichols tuning formula," *IEE Proceedings D Control Theory and Applications*, vol. 138, no. 2, p. 111, 1991. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/ip-d.1991.0015

[68] R. S. Ali, A. A. Aldair, and A. K. Almousawi, "Design an Optimal PID Controller using Artificial Bee Colony and Genetic Algorithm for Autonomous Mobile Robot," *International Journal of Computer Applications*, vol. 100, no. 16, pp. 8–16, Aug. 2014. [Online]. Available: http://research.ijcaonline.org/volume100/number16/pxc3898016.pdf

[69] P. de Moura Oliveira, "Modern Heuristics Review for PID Control Systems Optimization: a Teaching Experiment," in *2005 International Conference on Control and Automation*, vol. 2. Budapest, Hungary: IEEE, 2005, pp. 828–833. [Online]. Available: http://ieeexplore.ieee.org/document/1528237/

[70] P. Šuster and A. Jadlovská, "Tracking Trajectory of the Mobile Robot Khepera II Using Approaches of Artificial Intelligence," *Acta Electrotechnica et Informatica*, vol. 11, no. 1, Jan. 2011. [Online]. Available: http://www.aei.tuke.sk/papers/2011/1/06_Suster.pdf

[71] A. Alouache and Q. Wu, "Genetic Algorithms for Trajectory Tracking of Mobile Robot Based on PID Controller," in *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP)*. Cluj-Napoca: IEEE, Sep. 2018, pp. 237–241. [Online]. Available: https://ieeexplore.ieee.org/document/8516587/

[72] C. Kumar and T. Alwarsamy, "Solution of Economic Dispatch Problem using Differential Evolution Algorithm," *International Journal of Soft Computing and Engineering*, vol. 1, no. 6, pp. 236–241, Jan. 2012.

[73] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997. [Online]. Available: http://link.springer.com/10.1023/A:1008202821328

[74] V. Pano and P. Ouyang, "Comparative study of GA, PSO, and DE for tuning

position domain PID controller," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*. Bali, Indonesia: IEEE, Dec. 2014, pp. 1254–1259. [Online]. Available: http://ieeexplore.ieee.org/document/7090505/

[75] Wufan Wang, Xiaming Yuan, and Jihong Zhu, "Automatic PID tuning via differential evolution for quadrotor UAVs trajectory tracking," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. Athens, Greece: IEEE, Dec. 2016, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/document/7849872/

[76] R. Mallipeddi, P. Suganthan, Q. Pan, and M. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679–1696, Mar. 2011. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1568494610001043

[77] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, Dec. 2006. [Online]. Available: https://ieeexplore.ieee.org/document/4016057/

[78] C. Edwards and S. K. Spurgeon, *Sliding mode control: theory and applications*, ser. Systems and control book series ; vol. 7. London: Taylor & Francis, 1998.

[79] M. Asif, M. J. Khan, and N. Cai, "Adaptive sliding mode dynamic controller with integrator in the loop for nonholonomic wheeled mobile robot trajectory tracking," *International Journal of Control*, vol. 87, no. 5, pp. 964–975, May 2014. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/00207179.2013.862597

[80] C.-Y. Chen, T.-H. S. Li, Y.-C. Yeh, and C.-C. Chang, "Design and implementation of an adaptive sliding-mode dynamic controller for wheeled mobile robots," *Mechatronics*, vol. 19, no. 2, pp. 156–166, Mar. 2009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S095741580800144X

[81] J. H. Lee, C. Lin, H. Lim, and J. M. Lee, "Sliding mode control for trajectory tracking of mobile robot in the RFID sensor space," *International Journal of Control, Automation and Systems*, vol. 7, no. 3, pp. 429–435, Jun. 2009. [Online]. Available: http://link.springer.com/10.1007/s12555-009-0312-7

[82] M. Belhocine, M. Hamerlain, and F. Meraoubi, "Variable structure control for a wheeled mobile robot," *Advanced Robotics*, vol. 17, no. 9, pp. 909–924, Jan. 2003. [Online]. Available: https://www.tandfonline.com/doi/full/10.1163/156855303770558688

[83] A. K. Kar, N. K. Dhar, R. Chandola, S. S. F. Nawaz, and N. K. Verma, "Trajectory tracking by automated guided vehicle using GA optimized sliding mode control," in *2016 11th International Conference on Industrial and Information Systems (ICIIS)*. Roorkee, India: IEEE, Dec. 2016, pp. 71–76. [Online]. Available: http://ieeexplore.ieee.org/document/8262910/

[84] Jong-Min Yang and Jong-Hwan Kim, "Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 578–587, Jun. 1999. [Online]. Available: http://ieeexplore.ieee.org/document/768190/

[85] R. Solea and D. Cernega, "Sliding Mode Control for Trajectory Tracking Problem - Performance Evaluation," in *Artificial Neural Networks – ICANN 2009*, C. Alippi, M. Polycarpou, C. Panayiotou, and G. Ellinas, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, vol. 5769, pp. 865–874. [Online]. Available: http://link.springer.com/10.1007/978-3-642-04277-5_87

[86] H. Mehrjerdi and M. Saad, "Dynamic tracking control of mobile robot using exponential sliding mode," in *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society.* Glendale, AZ, USA: IEEE, Nov. 2010, pp. 1517–1521. [Online]. Available: http://ieeexplore.ieee.org/document/5675457/

[87] J. Yang, R. Ma, Y. Zhang, and C. Zhao, "Sliding Mode Control for Trajectory Tracking of Intelligent Vehicle," *Physics Procedia*, vol. 33, pp. 1160–1167, 2012. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1875389212015040

[88] H. B. Pacejka and I. Besselink, *Tire and vehicle dynamics*, 3rd ed., ser. Engineering Automotive engineering. Amsterdam: Elsevier/Butterworth-Heinemann, 2012, oCLC: 796260687.

[89] D. Karnopp, "Computer Simulation of Stick-Slip Friction in Mechanical Dynamic Systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. 1, pp. 100–103, Mar. 1985. [Online]. Available: https://asmedigitalcollection.asme.org/dynamicsystems/article/107/1/100/400601/Computer-Simulation-of-StickSlip-Friction-in

[90] D. D. Quinn, "A New Regularization of Coulomb Friction," *Journal of Vibration and Acoustics*, vol. 126, no. 3, pp. 391–397, Jul. 2004. [Online]. Available: https://asmedigitalcollection.asme.org/vibrationacoustics/article/126/3/391/459627/A-New-Regularization-of-Coulomb-Friction

[91] Clearpath Robotics, "HUSKY UGV: user manual."

[92] Masehian E. and Katebi Y., "Robot Motion Planning in Dynamic Environments with Moving Obstacles and Target." *International Journal of Mechanical Systems Science and Engineering.*, vol. 1, no. 1, pp. 20–25, 2007.

[93] M. Wang and J. N. Liu, "Fuzzy logic-based real-time robot navigation in unknown environment with dead ends," *Robotics and Autonomous Systems*, vol. 56, no. 7, pp. 625–643, Jul. 2008. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889007001467

[94] S. Yang, Tiemin Hu, Xiaobu Yuan, P. Liu, and Max Meng, "A neural network based torque controller for collision-free navigation of mobile robots," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 1. Taipei, Taiwan: IEEE, 2003, pp. 13–18. [Online]. Available: http://ieeexplore.ieee.org/document/1241566/

[95] D. Sartori, D. Zou, and W. Yu, "An efficient approach to near-optimal 3D trajectory design in cluttered environments for multirotor UAVs," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE).* IEEE, 2019, pp. 1016–1022.

[96] G. Walsh, D. Tilbury, S. Sastry, R. Murray, and J. Laumond, "Stabilization of trajectories for systems with nonholonomic constraints," *IEEE Transactions on*

Automatic Control, vol. 39, no. 1, pp. 216–222, Jan. 1994. [Online]. Available: http://ieeexplore.ieee.org/document/273373/

[97] E. Freund and R. Mayr, "Nonlinear path control in automated vehicle guidance," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 1, pp. 49–60, Feb. 1997. [Online]. Available: http://ieeexplore.ieee.org/document/554346/

[98] R. Fierro and F. Lewis, "Control of a nonholonomic mobile robot: backstepping kinematics into dynamics," in *Proceedings of 1995 34th IEEE Conference on Decision and Control*, vol. 4. New Orleans, LA, USA: IEEE, 1995, pp. 3805–3810. [Online]. Available: http://ieeexplore.ieee.org/document/479190/

[99] P. Gallina and A. Gasparetto, "A Technique to Analytically Formulate and to Solve the 2-Dimensional Constrained Trajectory Planning Problem for a Mobile Robot," *Journal of Intelligent and Robotic Systems*, vol. 27, no. 3, pp. 237–262, 2000. [Online]. Available: http://link.springer.com/10.1023/A:1008168615430

[100] V. Munoz, A. Ollero, M. Prado, and A. Simon, "Mobile robot trajectory planning with dynamic and kinematic constraints," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation.* San Diego, CA, USA: IEEE Comput. Soc. Press, 1994, pp. 2802–2807. [Online]. Available: http://ieeexplore.ieee.org/document/350914/

[101] N. Minorsky., "DIRECTIONAL STABILITY OF AUTOMATICALLY STEERED BODIES," *Journal of the American Society for Naval Engineers*, vol. 34, no. 2, pp. 280–309, Mar. 2009. [Online]. Available: http://doi.wiley.com/10.1111/j.1559-3584.1922.tb04958.x

[102] X.-G. Yan, S. K. Spurgeon, and C. Edwards, "Memoryless Static Output Feedback Sliding Mode Control for Nonlinear Systems With Delayed Disturbances," *IEEE Transactions on Automatic Control*, vol. 59, no. 7, pp. 1906–1912, Jul. 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6690208/

[103] J. Hung, W. Gao, and J. Hung, "Variable structure control: a survey," *IEEE Transactions on Industrial Electronics*, vol. 40, no. 1, pp. 2–22, Feb. 1993. [Online]. Available: http://ieeexplore.ieee.org/document/184817/

[104] Utkin, V. I, *Sliding modes and their application in variable structure systems.* Moscow: Mir, 1978.

[105] ——, "Variable structure systems with sliding modes," *IEEE Transactions on Automatic Control*, vol. 22, no. 2, pp. 212–222, Apr. 1977. [Online]. Available: http://ieeexplore.ieee.org/document/1101446/

[106] Utkin, V. I and Young, K. D., "Method for Constructing Discontinuity Planes in Multidimensional Variable Structure Systems," *Automation and Remote Control,*, vol. 39, no. 10, pp. 1466–1470, 1978.

[107] Utkin, V. I, *Sliding modes in control and optimization*, ser. Communication and control engineering series. Berlin ; New York: Springer-Verlag, 1992.

[108] K. David Young and U. Ozguner, "Frequency shaping compensator design for sliding mode," *International Journal of Control*, vol. 57, no. 5, pp. 1005–1019, May 1993. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/00207179308934427

[109] M. Kim, J. Shin, and J. Lee, "Design of a robust adaptive controller for

a mobile robot," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, vol. 3. Takamatsu, Japan: IEEE, 2000, pp. 1816–1821. [Online]. Available: http://ieeexplore.ieee.org/document/895235/

[110] W.-C. Su, S. V. Drakunov, and U. Ozguner, "Constructing discontinuity surfaces for variable structure systems: A Lyapunov approach," *Automatica*, vol. 32, no. 6, pp. 925–928, Jun. 1996. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0005109896000179

[111] Weibing Gao and J. Hung, "Variable structure control of nonlinear systems: a new approach," *IEEE Transactions on Industrial Electronics*, vol. 40, no. 1, pp. 45–55, Feb. 1993. [Online]. Available: http://ieeexplore.ieee.org/document/184820/

[112] A.M. Loh, "Chattering Reduction in Sliding Mode Control: An Improvement for Nonlinear Systems," *WSEAS Transaction on Systems*, vol. 9, no. 3, pp. 2878–2885, 2004.

[113] R. DeCarlo, S. Zak, and G. Matthews, "Variable structure control of nonlinear multivariable systems: a tutorial," *Proceedings of the IEEE*, vol. 76, no. 3, pp. 212–232, Mar. 1988. [Online]. Available: http://ieeexplore.ieee.org/document/4400/

[114] K. Young, V. Utkin, and U. Ozguner, "A control engineer's guide to sliding mode control," *IEEE Transactions on Control Systems Technology*, vol. 7, no. 3, pp. 328–342, May 1999. [Online]. Available: http://ieeexplore.ieee.org/document/761053/

[115] G. Bartolini, A. Ferrara, and E. Usai, "Chattering avoidance by second-order sliding mode control," *IEEE Transactions on Automatic Control*, vol. 43, no. 2, pp. 241–246, Feb. 1998. [Online]. Available: https://ieeexplore.ieee.org/document/661074/

[116] T. Rogalsky, R.W. Derksen, and S. Kocabiyik, "Differential evolution in aerodynamic optimization," in *Proceeding of 46th Annual Conference of Canadian Aeronautics and Space Institute*, Montreal, Quebec, 1999, pp. 29–36.

[117] R. Joshi and A. Sanderson, "Minimal representation multisensor fusion using differential evolution," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 29, no. 1, pp. 63–76, Jan. 1999. [Online]. Available: http://ieeexplore.ieee.org/document/736361/

[118] M. Varadarajan and K. Swarup, "Differential evolution approach for optimal reactive power dispatch," *Applied Soft Computing*, vol. 8, no. 4, pp. 1549–1561, Sep. 2008. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S156849460700172X

[119] R. Storn, "Differential evolution design of an IIR-filter," in *Proceedings of IEEE International Conference on Evolutionary Computation*. Nagoya, Japan: IEEE, 1996, pp. 268–273. [Online]. Available: http://ieeexplore.ieee.org/document/542373/

[120] ——, "On the usage of differential evolution for function optimization," in *Proceedings of North American Fuzzy Information Processing*. Berkeley, CA, USA: IEEE, 1996, pp. 519–523. [Online]. Available: http://ieeexplore.ieee.org/document/534789/

[121] A. Qin, V. Huang, and P. Suganthan, "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization," *IEEE Transactions on*

Evolutionary Computation, vol. 13, no. 2, pp. 398–417, Apr. 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4632146/

[122] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 61–106, Feb. 2010. [Online]. Available: http://link.springer.com/10.1007/s10462-009-9137-2