University of Pardubice

Faculty of Electrical Engineering and Informatics

Trajectory Tracking of Differential Drive Mobile Robot by Model Predictive Control

Rahul Sharma K.

Doctoral Thesis

2017

I hereby declare that I have written the Doctoral paper on my own.
All the literary sources and the information used are listed in the bibliography.
I was familiar with the fact that rights and obligations arising from the Act No. 121/2000 Coll., the Copyright Act, apply to my thesis, especially the fact that the University of Pardubice has the right to enter into a license agreement for use of the paper as a school work pursuant to § 60, Section 1 of the Copyright Act, and the fact that should this thesis be used by me or should a license be granted for the use to another entity, the University of Pardubice is authorized to claim a reasonable fee to cover the costs incurred during the making of the paper, up to the real amount thereof.

I agree with the reference-only disclosure of my thesis in the University Library.

In Pardubice on March 31, 2017

Rahul Sharma K.

ACKNOWLEDGEMENT

A research endeavor is akin to a long and challenging journey. Without the assistance, motivation and support provided by others along the way, the task would be immensely harder, if not impossible.

First and foremost, I would like to express my sincere gratitude towards my supervisors, František Dušek and Daniel Honc, whose guidance, patience and support during this time were fundamental to complete this work. Special thanks goes to Paul Charles Hooper for his important help in the English language proof reading the thesis. I also would like to thank my colleagues at FEI for providing me a friendly and cooperative environment at work.

I would like to thank to the staff at international office at University of Pardubice, for their constant support.

Many thanks to all my friends in Pardubice, who helped me in many ways.

I gratefully acknowledge Svaagata Erasmus Mundus Programme for the financial support of the scholarship that made this project a reality. I am very thankful to the whole Svaagata team, especially Anne, Martin Fárek and Balu, for offering me the scholarship option and helped me throughout the project. I also would like to thank the other funding sources: FEI and SGS projects.

I am also would like to express my appreciation and gratitude to Czech Republic people for their warm hospitality and kindness. I feel blessed to have pursued my PhD studies in the Czech Republic – the land where the term „robot“ originated (Karel Čapek).

Last, but not least, I would like to offer my heartfelt thanks to my mother and family for all the love, encouragement, support and patience that they have shown me over the years. I would be nothing without them. Thanks to all the people who care about me, and that I care about.

ABSTRACT

In the age of self-driving cars, a great deal of attention is being paid to research on the motion control problems of the mobile robot / autonomous vehicle. The trajectory tracking of the mobile robot is one of the motion control problems, which refers to tracking the robot through a time parameterized reference trajectory. This is generally achieved by a kinematic controller and a dynamics controller. Kinematic MPC, assuming a "perfect velocity tracking" dynamics controller on low level, has been successfully applied in academia as well as industry. A dynamics controller (commonly PID) at dynamics level produces the required forces for motion control. However, advanced controllers are still not applied to the dynamics of the mobile robot. This is mainly because of the difficulties in modelling the dynamics of the mobile robot.

The thesis proposes modelling of dynamics of a differential drive robot, and application of MPC in both the kinematics and dynamics parts. The thesis is divided in to three parts: kinematic modelling and predictive control, dynamics modelling and control, and Kino-Dynamics control. Firstly, the non-linear kinematic equations were linearized into two different models and nonlinear MPCs were applied with these models. The responses were compared with state-of-the-art controllers. Secondly, the mathematical model of dynamics of mobile robot was derived from first principles. The tangential and angular velocities were controlled by generating motor voltages by Linear MPC and the response was compared to PID controllers. Thirdly, the kinematic controllers and dynamics controllers were cascaded and a comparative study has been conducted with respect to different control structures. It has been noticed that, MPC of the dynamics part, can not only generate an optimal control action, but also can influence the kinematic part and decrease the overall tracking errors.

The NMPC-LMPC control structure has several advantages for a trajectory-tracking problem: it can generate optimal control actions by considering system constraints, increase the overall stability, decrease the overall tracking errors etc. The thesis concludes with the comparative analysis of control structures for trajectory tracking problem. Suggestions for future research work are also presented.

Keywords

Model Predictive Control, trajectory tracking, mobile robot, modelling, optimization

NÁZEV

Řízení pohybu mobilního robota pomocí prediktivního regulátoru

ANOTACE

V práci je navrženo dvouúrovňové řízení pohybu mobilního dvoukolového robotu zajišťující sledování známé trajektorie. V obou úrovních je použit prediktivní regulátor s uvažováním omezení. Pro návrh řízení ve vyšší úrovni vycházející z kinematického modelu (závislost polohy a orientace robotu na jeho aktuální tangenciální a úhlové rychlosti) byly vytvořeny dva nelineární modely chyby sledování známé trajektorie. Pro návrh řízení v nižší úrovni a pro možnost simulačního ověření celého řízení robotu byl vytvořen metodou matematicko-fyzikální analýzy lineární dynamický model robotu popisující závislost jeho tangenciální a úhlové rychlosti na napětích elektromotorů poháněných obě kola.

Simulačně byly porovnány průběhy řízení pro různé struktury řízení i regulátory. Pozornost byla věnována zejména vlivu zanedbání dynamiky robotu a přínosu prediktivních regulátorů oproti standardně používaným řešením jak v kinematické tak i dynamické úrovni. Výsledky simulací ukazují, že prediktivní regulátor dynamické části, kromě respektování zadaných omezení, také ovlivňuje kinematickou část a zvyšuje celkovou kvalitu regulace.

KLÍČOVÁ SLOVA

Prediktivní řízení, řízení pohybu mobilního robota, modelování, optimalizace

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

LIST OF SYMBOLS

$\{\cdot,\ldots,\cdot\}$	A set or sequence
$\mathbb{R}$	Set of real numbers
$\mathbb{Z}$	Set of natural numbers
$\mathbb{R}^n$	Set of n-dimensional vector with real numbers
$\mathbb{R}^{n\times m}$	Set of n by m matrix with real numbers
n_x	Number of states, $n_x \in \mathbb{Z}$
n_y	Number of outputs, $n_y \in \mathbb{Z}$
n_u	Number of inputs, $n_u \in \mathbb{Z}$
$\boldsymbol{u}$	Input vector, $\boldsymbol{u} \in \mathbb{R}^{n_u}$
$\boldsymbol{y}$	Output vector, $\boldsymbol{y} \in \mathbb{R}^{n_y}$
$\boldsymbol{x}$	State vector, $\boldsymbol{x} \in \mathbb{R}^{n_x}$
$\boldsymbol{w}$	Set-point vector
T_s	Sampling time
t_k	Time at $t = kT_s$
N	Prediction/control horizon length
$\boldsymbol{u}(x)$	Vector depends upon parameter x
$\boldsymbol{u}_N$	Sequence of input vector $\boldsymbol{u}_N = [\boldsymbol{u}(\boldsymbol{t}) \ldots . \boldsymbol{u}(\boldsymbol{t}+N)]^T$
$\boldsymbol{I}$	Identity matrix
$\mathbf{0}$	Zero matrix
$\boldsymbol{x}_k$	State vector at time, $t = t_k$
$\Delta\boldsymbol{u}$	Deviation variable (not to be confused with u(k)-u(k-1) -which is difference variable as in difference equation)
$\boldsymbol{x}^T, \boldsymbol{A}^T$	Vector or matrix transform
$\boldsymbol{A}^{-1}$	Matrix inverse

Variables

x_B, y_B, θ_B	Real/simulated coordinates and orientation
x_r, y_r, θ_r	Reference coordinates and orientation
v_B, ω_B	Real/simulated tangential and angular velocities
v_r, ω_r	Reference tangential and angular velocities
$v_{B,r}, \omega_{B,r}$	Reference tangential and angular velocities for low-level controller

ω_L, ω_R	Right and left motor speeds
i_L, i_R	Right and left motor currents
u_L, u_R	Right and left motor control voltage signals
$\boldsymbol{U}_o$	Source voltage
$\boldsymbol{y}_{fo}$,$\boldsymbol{y}_{fr}$	Forced and free output responses
$\boldsymbol{x}_{fo}$,$\boldsymbol{x}_{fr}$	Forced and free response state variables
N_u,N_y	Control and output prediction horizon
$\boldsymbol{R}, \boldsymbol{Q}, \boldsymbol{Q}_N$	Weighting matrices of criteria for MPC
$\boldsymbol{u}_{ff}, \boldsymbol{u}_{fb}$	Feedforward and feedback control input
$\boldsymbol{u}_K$	Kinematic controller manipulated variables
$\boldsymbol{x}_K$	Kinematic controller state variables
$\boldsymbol{y}_K$ $\boldsymbol{w}_K$	Kinematic controller outputs and set-points
$\boldsymbol{A}_K, \boldsymbol{B}_K, \boldsymbol{C}_K$	Kinematic controller state space matrices
N_K	Prediction horizon for kinematic controller
$\boldsymbol{R}_K, \boldsymbol{Q}_K, \boldsymbol{Q}_{K_N}$	Weighting matrices of criteria for kinematic MPC
$\boldsymbol{u}_D$	Dynamics controller manipulated variables
$\boldsymbol{x}_D$	Dynamics controller state variables
$\boldsymbol{y}_D$ $\boldsymbol{w}_D$	Dynamics controller outputs and set-points
$\boldsymbol{A}_D, \boldsymbol{B}_D, \boldsymbol{C}_D$	Dynamics controller state space matrices
N_D	Prediction horizon for dynamics controller
$\boldsymbol{R}_D, \boldsymbol{Q}_D, \boldsymbol{Q}_{D_N}$	Weighting matrices of criteria for dynamics MPC

LIST OF ABBREVIATIONS

DC	Direct Current
EMF	Electro-Motive Force
GPS	Global Positioning System
HIL	Hardware-in-the-Loop
IC	Initial Condition
KC	Kanayama Controller
LMPC	Linear Model Predictive Control
LTI	Linear Time Invariant
LTV	Linear Time Variant
MBD	Model-Based Design
MIMO	Multi Input Multi Output system
MPC	Model Predictive Control
NMPC	Non-Linear Model Predictive Control
$NMPC_1$	NMPC with successive linear model
$NMPC_2$	NMPC with error based model
ODE	Ordinary Differential Equation
PC	Personal Computer
PID	Proportional - Integral - Derivative
PWM	Pulse Width Modulation
RHC	Receding Horizon Control
SC	Samson's Controller
SSE	Sum Squared Error
TITO	Two Input Two Output system
USB	Universal Serial Bus
WLAN	Wireless Local Area network
WMR	Wheeled Mobile Robot

1. INTRODUCTION

1.1 Introduction and state of the art

The past few decades have witnessed an increased research effort in the area of motion control of autonomous vehicles. In the age of self-driving cars, the importance of the study of motion control of the autonomous systems is ever increasing. Robust motion control algorithms are fundamental to the autonomous operation of mobile robot. Motion control refers to "how to control the robot to make some particular motion- either time bound or not". There are basically three types of motion control problems: trajectory tracking, path following and point stabilization. Point stabilization (parking) refers to stabilization of the robot into a predefined position and orientation. Path following refers to move a robot along a path in a time independent manner. The trajectory tracking problem is similar to path following problem, but in a predefined time. A typical motion control problem is trajectory tracking, which is concerned with the design of control laws, that force a vehicle to reach and follow a time parameterized reference (i.e., a geometric path with an associated timing law). The degree of difficulty involved in solving this problem is highly dependent on the configuration of the vehicle and quality of position information.

Wheeled mobile robots (WMR) are widely used in many applications mainly because of the high loading capacity, less complexity, ease of control etc. Design of WMR depends on application with which they are applied (Cook 2011; Niku 2001). The number of wheels, configuration, type of steering, motors etc. depends on various design considerations. Mobile robots use several wheel configurations, such as differentially driven, car-type, omni-directional, and synchro drive (Siegwart et al. 2011). The most common wheel configuration used in mobile robot designs is the differential drive. In a differential drive, the movement is based on two separately driven wheels on either side of the robot and one or more castor wheels which provide the stability of the robot. The steering is achieved by a relative rate of rotation of the wheels and hence no additional steering mechanism is needed. Differential drive vehicles have the added advantage that they can turn in place.

Differential drive mobile robots have many potential applications, but motion planning is difficult as they are subject to rolling constraints that limit the possible directions of motion, i.e. they cannot move sideways directly, but must move forwards or backwards in order to turn. Hence, complicated manoeuvres may be required to move from one configuration to another nearby one; even in the absence of obstacles. Such constraints, that limit the possible directions

of motion are called non-holonomic (Triggs 1993). The difficulties of non-holonomic system are that, the non-holonomic system does not satisfy the Brockett theorem (Brockett 1983), so there does not exist smooth time-invariant state feedback control such that can makes the system asymptotically stable. However, it has been proven that, the asymptotic stabilization can be obtained using time-varying, discontinuous or hybrid control laws.

The tracking control problem is classified as a kinematic control problem or dynamic control problem based on the system description – by kinematic model or dynamic model. The structure of the kinematic and dynamic models of WMR are analyzed and classified in (Campion et al. 1996). At the beginning, the research effort was focused only on the kinematic model, assuming that there is perfect velocity tracking (Kolmanovsky & McClamroch 1995). The main objective was to find suitable velocity control inputs, which stabilize the kinematic closed loop control.

The dynamics of WMR are nonlinear and involve non-holonomic constraints, which cause difficulty in their modeling and analysis. There are two common formulations for mobile robot dynamics in the literature; one based on Lagrangian mechanics and the other on Newton-Euler mechanics (refer (Hatab & Dhaouadi 2013) and the references with in).

The problem with control of dynamics part are precomputation of velocities by a pair of PID controllers. Furthermore, there are problems with constraints and noises too. At present, the PID controller is still widely used in motor control of mobile robot. However, its ability to cope with some complex process properties such as nonlinearities, and time-varying parameters are known to be very poor. The reference trajectory is not tracked directly, but by two controllers – high level controller for generating velocity control inputs (kinematics controller) and low level controller for generating motor torque (dynamics controller). The usage of more sophisticated controllers can solve these problems, which require dynamic model of the mobile robot.

Controlling non-holonomic systems as they follow a reference path is a well-known problem that has been studied by many authors. The control problem is solved by considering its first-order kinematic model. The obtained model can later be upgraded to include the dynamic properties. Usually, the reference trajectory is obtained by using a reference robot; therefore, all the kinematic constraints are implicitly considered in the reference trajectory. The control inputs are mostly obtained by a combination of feedforward inputs which are, calculated from the reference trajectory, and feedback control law, as in (Wenjie Dong et al. 2000; Sarkar

et al. 1994; De Luca & Oriolo 1995; Oriolo et al. 2002). Lyapunov stable time-varying state-tracking control laws were pioneered by (Kanayama et al. 1990; Samson & Ait-Abderrahim 1991; Fierro & Lewis 1995) , where the system's equations are linearized with respect to the reference trajectory; and by defining the desired parameters of the characteristic polynomial, the controller parameters are calculated. A sliding mode control law is proposed for asymptotically stabilizing the mobile robot to a desired trajectory (Yang & Kim 1999). Dynamic feedback linearization is presented in (Oriolo et al. 2002), for both trajectory tracking and point stabilization problem.

All the above-discussed techniques consider only kinematic model and assumes a "perfect velocity tracking" controller (to generate actual velocity control inputs) is present at lower level. However, dynamics of mobile robot cannot be neglected, especially when high performance is required. A dynamical extension that makes possible the integration of a kinematic controller and a torque controller is presented in (Fierro & Lewis 1995). Various other non-linear techniques are also proposed in literature, e.g. sliding model controller (Yang & Kim 1999), adaptive controller (Kim et al. n.d.; Fukao et al. 2000), robust adaptive controller (Pourboghrat & Karlsson 2002) together with various artificial intelligence techniques (Fierro & Lewis 1998; Das & Kar 2006; Hu & Yang 2001) considering system disturbances and unknown parameters.

The Model Predictive Control (MPC) (also known as Receding Horizon Control (RHC)) has been an important research area for decades. MPC is also seems to be very promising in the field of mobile robotic trajectory tracking, because the reference trajectory is known beforehand. It is designed to handle complex, constrained, multivariable control problems. It is an online optimization tool, which will generate optimal control actions required at every time instance by minimizing an objective function based on predictions (Camacho & Bordons 2004) and by respecting constraints. With the increase in computational power, the MPC is not only limited to slow dynamics processes, where dynamical optimization is easily possible, but also there are new applications for faster systems. Most of the MPC technologies are based on linear dynamic models and therefore referred to as a linear model predictive controller (LMPC). However, many processes are sufficiently nonlinear which hinder the successful application of LMPC. This has led to the development of nonlinear model predictive controllers (NMPC) in which nonlinear models are used for prediction and optimization. The main problem with NMPC is that, the nonlinear program has to be solved online at every sampling time to generate

control action, which is a computationally heavy task. There are various NMPC formulations in the literature (refer (Henson 1998) and the references with in).

In case of trajectory tracking of mobile robots, MPC techniques produce promising results as shown by (Maurovic et al. 2011; Chen et al. 2009; Kunhe, F., J. Gomes 2005; Lages & Vasconcelos Alves 2006; Kuhne, Felipe, Walter Fetter Lages 2004). In (Klančar & Škrjanc 2007), a tracking error based predictive control is presented. A mobile robot trajectory tracking problem with linear and nonlinear state-space MPC can be seen in (Kunhe, F., J. Gomes 2005). A survey of motion control problems of Wheeled Mobile Robots (WMRs) using MPC can be found in (Kanjanawanishkul 2012).

1.2 Problem formulation and objectives

Trajectory tracking of mobile robots requires usage of nonlinear kinematic differential equations. This requires, either linearizing the model and applying the LPMC or using nonlinear equations in prediction and/or in optimization and applying the NMPC. Various LMPC techniques applied to the trajectory-tracking problem (that can be seen in the literature) but usage of NMPC is quite rare, mainly because of the computation time requirement. One solution could be, linearize the nonlinear model into a LTV model and use it for prediction and optimization or use a combination of LTV model and non-linear model.

In most of the trajectory tracking problems, the solutions in literature considers only the kinematics and the dynamics of the mobile robot is neglected. This is mainly because of the difficulty in modelling and identifying the nonlinearities associated with dynamics of the mobile robot. However, the kinematic trajectory tracking with cascade structure (high-level kinematic controller – low-level dynamics controller) has successfully been applied by many researchers. In a typical trajectory tracking problem, the trajectory is tracked by generating reference tangential and angular velocities (kinematic controller), assuming a perfect velocity tracker is available and this velocity is tracked by generating motor torques using a pair of PID controllers. If the dynamics of the mobile robot can be modelled and identified with considerable precision, the mobile robot motion control problems can be solved with adequate accuracy - considering non- holonomic constraints and other soft constraints (e.g. energy, comfort (acceleration, jerk, slipping, skidding etc.)). A straight forward control of motor power (control actions) to track a robot into a trajectory (outputs) has obvious advantages when compared to controlling with a high level (kinematic) controller (usually implemented in PC) and low level (dynamics) controller (usually PID implemented in mobile robot controller).

Dynamic constraints (soft and hard) can also be easily integrated with kinematic (non-holonomic) constraints, and control action can be computed. This integrated kinematic-dynamics model can also leverage the possibility of different feedback configurations (e.g. mobile robot positioning systems (Borenstein et al. 1997)).

MPC is a straightforward choice for the trajectory-tracking problem because of the following reasons,

a. Flexibility in the formulation of MPC elements (prediction, cost function and constraints)
b. In the trajectory tracking problem, the future set-point are known
c. Ease of constraint handling (physical constraints of mobile robot and soft constraints) capability
d. Capability of dealing with non-linear (kinematic model is non-linear) and multi variable control system

To summarize the objectives,

a. Design linear and non-linear state space model predictive controller with a criteria penalizing the control effort, control error and terminal state error.
b. Derive the linear time varying model and reference variables from the basic kinematic equations. Conduct open-loop model verification experiments.
c. Design the kinematic MPC and state-of-the-art controllers for trajectory tracking of mobile robot and to compare the performances.
d. Derive mathematical model of dynamics of mobile robot considering motor dynamics and chassis dynamics. Verify the model by simulation and open loop experiments.
e. Design Dynamics controllers for velocity tracking of mobile robot and compare the performances.
f. Study the performance of different control structures for Kino-Dynamics controllers for trajectory tracking problem. Discuss the advantages of different control structures.

1.3 Assumptions

The following assumptions are considered in the thesis,

a) An ideal mobile robot respecting non-holonomic constraints (rolling without slipping)
b) A perfect localization system is present, i.e. robot's position and orientation are known at every time instant
c) An ideal differential drive robot powered independently by two motors and a castor wheel to support the robot

d) Non-linearities of gears (saturation, backlash and friction) are not considered in the dynamic model
e) There is no time delay between measurement-control action calculation, actuation-measurement etc.

The other assumptions are mentioned in respective chapters.

1.4 Thesis outline

This chapter introduces the main idea of the thesis and outlines the motivation and objectives of the research. The rest of the thesis is organized in the following manner:

Chapter 2 provides the control strategies for trajectory tracking of the mobile robot. This includes trajectory generation, kinematics and dynamics of the mobile robot

Chapter 3 explains basics of predictive controller and design of state space linear and non-linear MPC.

Chapter 4 examines the non-linear kinematic model of non-holonomic robot. Two models are derived based on reference co-ordinate frame – world coordinates and local coordinates system of the robot. Open-loop model verification experiments are also provided.

Chapter 5 presents the kinematic control of non-holonomic mobile robot with two non-linear MPCs and a comparison with respect to state of the art trajectory tracking controllers.

Chapter 6 explores in detail the mathematical modelling of differential drive robot. A state space linear model is derived and open-loop model verification experiments are conducted.

Chapter 7 describes the dynamics velocity tracking controller design. A linear MPC and PID controllers are proposed and a comparative study also provided.

Chapter 8 illustrates Kino-Dynamics controller design for trajectory tracking problems. Design of various control structures are explained and a comparative analysis based on the performance of control structures is described.

Chapter 9 consists of the overall conclusions of the research and proposes future remarks.

2. CONTROL STRATEGY FOR TRAJECTORY TRACKING OF MOBILE ROBOT

Trajectory tracking of mobile robot refers to track the mobile robot in desired positional coordinates and orientation in a time parameterized way. The reference positional coordinates are usually the inputs which are way points (or goals) to be tracked at particular time instance. The trajectory planner interpolates the way points into smooth trajectory (robot pose – coordinates in x,y axis and orientation) with equidistant time samples. The ideal feasible tangential and angular velocities at particular time instant is also generated. The next stage is to generate the velocities control inputs depending on current and reference robot poses and velocities. Motor control voltages are generated with respect to the required velocity commands. Figure 1 shows a general block diagram of trajectory tracking which is subdivided into various levels.

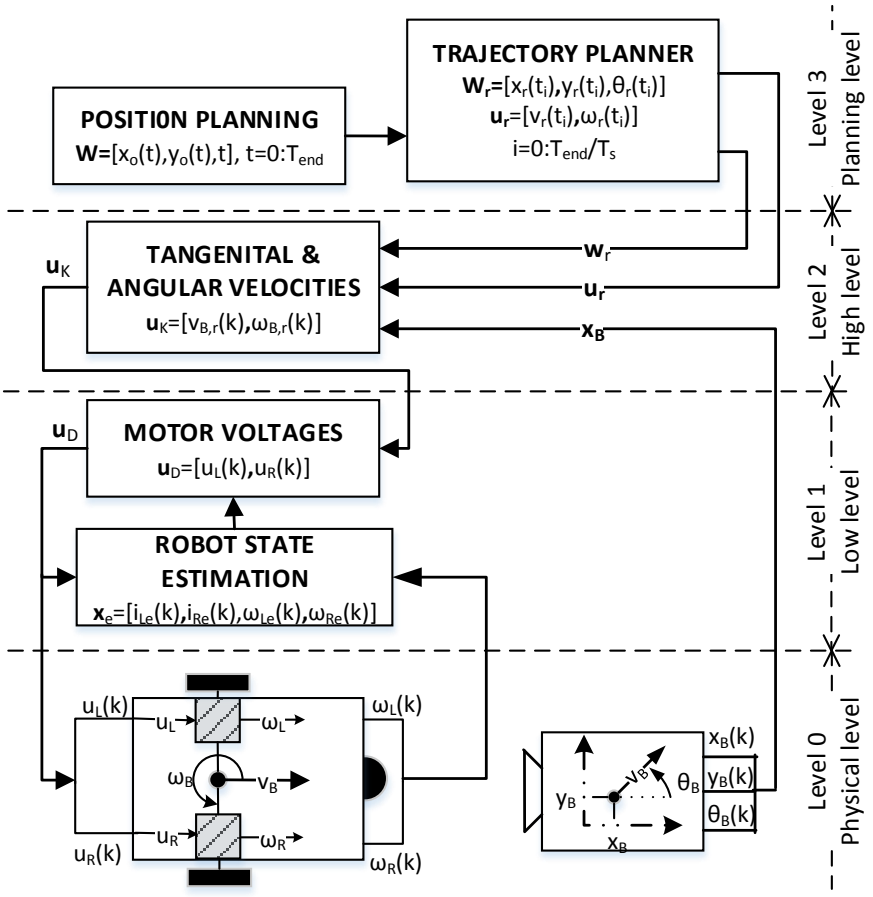

Figure 1 General block diagram of trajectory tracking

Level 3 (planning level): The reference way points (x_o, y_o, t) are interpolated into smooth trajectory $\boldsymbol{w}_r = (x_r, y_r, \theta_r)$ with a sampling time of T_s. The tangential and angular velocities $\boldsymbol{u}_r = (v_r, \omega_r)$ of reference robot travelling through the smooth trajectory are generated from the first order kinematic equations.

Level 2 (high level): The command velocities $\boldsymbol{u}_K = (v_{B,r}, \omega_{B,r})$ for the real robot are calculated from the reference pose and current pose of the robot with respect to the reference robot.

Level 1 (low level): The motor voltages $\boldsymbol{u}_D = (u_R, u_L)$ are generated according to the command velocities. The motor speeds (ω_R, ω_L) are measured by encoder and the motor state variables (motor currents i_R, i_L) are estimated.

Level 0 (physical level): The robot motors are independently driven by the control voltages calculated by the previous level. The robot position and orientation $\boldsymbol{x}_B = (x_B, y_B, \theta_B)$ in Cartesian coordinates are measured by an overhead camera.

2.1 Trajectory Planner (Planning level)

The functions of the trajectory planner are smooth trajectory generation and reference velocities generation. The feasible smooth trajectory is generated by path planning algorithms or by interpolation methods. As path planning is out of scope of this contribution, a simple interpolation method is preferred. The way points (or data points) $(\boldsymbol{x}_o, \boldsymbol{y}_o, \boldsymbol{t})$ are interpolated into $N_r = int(\frac{T_{end}}{T_s})$ data points as $\{x_r(t_k), y_r(t_k), t_r(t_k)\}$ and $t_k = kT_s, \forall\ k = 0 : N_r$

The reference velocities are velocities and orientation of a reference robot traveling through the smooth trajectory. The first order (continuous time) kinematic equation is given by,

$$\begin{aligned} \dot{x}_r &= v_r \cos\theta_r \\ \dot{y}_r &= v_r \sin\theta_r \\ \dot{\theta}_r &= \omega_r \end{aligned} \tag{2.1}$$

The tangential velocity at every time instant is approximated by,

$$v_r(t_k) = \sqrt{\frac{x_r(t_k) - x_r(t_{k-1})}{T_s} + \frac{y_r(t_k) - y_r(t_{k-1})}{T_s}} \tag{2.2}$$

The orientation at every time instance is,

$$\theta_r(t_k) = arctan2\left(\frac{y_r(t_k) - y_r(t_{k-1})}{T_s}, \frac{x_r(t_k) - x_r(t_{k-1})}{T_s}\right) \tag{2.3}$$

Angular velocity is obtained as,

$$\omega_r(t_k) = \frac{\theta_r(t_k) - \theta_r(t_{k-1})}{T_s} \tag{2.4}$$

In order to generate feasible reference trajectories, the constraints have to be taken into account during trajectory generation. Let the constraints on tangential and angular velocities are v_{max} and ω_{max} respectively. It is not possible to generate maximum tangential and angular velocities at the same time as curvature radius cannot be preserved. A velocity scaling will preserve the curvature radius corresponding to tangential velocity v and angular velocity ω. The actual reference velocities are computed by defining (De Luca et al. 2001),

$$\sigma(k) = \max\left\{\frac{|v(k)|}{v_{max}}, \frac{|\omega(k)|}{\omega_{max}}, 1\right\}$$

and letting,

$$\begin{aligned} v_r(k) &= sign(v(k))v_{max}, \omega_r(k) = \frac{\omega(k)}{\sigma(k)} && if\ \sigma(k) = \frac{|v(k)|}{v_{max}} \\ v_r(k) &= \frac{v(k)}{\sigma(k)}, \omega_r(k) = sign(\omega(k))\omega_{max} && if\ \sigma(k) = \frac{|\omega(k)|}{\omega_{max}} \\ v_r(k) &= v(k), \omega_r(k) = \omega(k) && if\ \sigma(k) = 1 \end{aligned}$$

The parameters of the reference robot are,

$$[\boldsymbol{x}_r, \boldsymbol{u}_r] = [x_r(t_k)\ y_r(t_k)\ \theta_r(t_k)\ v_r(t_k)\omega_r(t_k)]$$

The calculated robots reference velocities will drive the robot through the desired trajectory only if there are no disturbances and no initial state errors. However, in real world scenario, this is not always the case, and this brings us to design a kinematic controller, which can cope with these uncertainties.

2.2 Kinematics (High level)

According to Wikipedia, Kinematics is "the branch of mechanics that deals with pure motion, without reference to the masses or forces involved in it". Only the geometry of motion is considered at the kinematics level – robot pose and velocities. The kinematics of non-holonomic mobile robot is described as in eq (2.1). The Kinematics level is responsible for generating the required tangential and angular velocities to track the robot through a desired trajectory. Usually a kinematic controller generates the required velocities by considering, the current and reference pose of the WMR and the reference velocities. Chapter 4 is dedicated to kinematic modelling and chapter 5 for kinematic controllers. Even though, the described kinematics is common to all non-holonomic mobile robots, only a differential drive mobile robot is considered at the dynamics and Kino-Dynamics level.

2.3 Dynamics (Low level)

According to Wikipedia, dynamics is “the branch of mechanics concerned with the study of forces and torques and their effect on motion”. A differential drive robot with independently controlled motors is considered. The required forces and torques (for particular motion) are generated by controlling the motor voltages. A dynamics controller is often required to generate the motor voltages depending upon the current and reference velocities and the state variables. A state space model describes the dynamics of the motor with state variables having physical meaning, e.g. motor currents, motor speeds etc. These state variables have to be estimated from the measured variable. The dynamics modelling is described in chapter 5 and dynamics controllers in chapter 6.

2.4 Mobile robot and localization system (Physical level)

At physical level, it is assumed that an ideal differential drive robot and a localization is present. A differential drive mobile robot with two wheels independently controlled by motors and a caster wheel is considered. The motor speeds are measured by wheel encoders attached to the wheel. An onboard microcontroller with wireless communication module has the functions: get the command motor voltages or velocities, measure the motor speeds, and send the measured motor speeds.

The localization system consists of an overhead camera and necessary image processing algorithms. Markers are attached to the robot to localize the robot, and with the help of suitable image processing techniques, the robot is localized. A coordinate frame is fixed and the robot’s position and orientation are measured with respect to the coordinate axis. Tangential and angular velocities are also calculated from this information.

3. MODEL PREDICTIVE CONTROL

Model predictive control is based on a philosophy which reflects human behavior whereby the control actions which are brought about by our thought processes, lead to the best predicted outcome, over some limited horizon optimizing certain criteria (Rossiter 2003). E.g. while driving a car through a rough terrain, the required acceleration and steering (control actions) will be decided by taking into account several factors: the way (set-points); braking, acceleration and steering (control actions); obstacles, fuel efficiency, comfort, jerking (constraints) and various others. The key advantage of MPC when compared to classical control theory is the flexibility in the formulation – prediction model, criteria, optimization, constraints etc.

In general, for practical implementation, the MPC methods for linear or nonlinear systems are developed by assuming that the plant under control is described by a discrete-time representation. At each sampling time, the model predictive controller generates an optimal control sequence by optimizing a cost function. The first control action of this sequence is applied to the system. The optimization problem is solved again at the next sampling time, using the updated process measurements and a shifted horizon. The cost function formulation depends on the control requirements.

Let a continuous time system, linear or nonlinear be in the form of,

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \qquad \boldsymbol{x}(t = t_0) = \boldsymbol{x}_0 \\ \boldsymbol{y}(\boldsymbol{t}) &= \boldsymbol{g}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \qquad t \geq t_0 \end{aligned} \tag{3.1}$$

A prediction model, which predicts the system output $\boldsymbol{y}(t)$ for a finite horizon t_N, is the key to MPC approach. If the system is linear, the output can be decomposed into the sum of two wave-forms - **free response and the forced response** (see figure 2).

- Free response, $\boldsymbol{y}_{fr}$, represents the outputs of the system when all system inputs $\boldsymbol{u}_0(t)$ will vary according to the known waveform (e.g. will be constant from the time t_0) with the initial conditions as current state.
- Forced response, $\boldsymbol{y}_{fo}$ represents the portion of actual output of the system which varies from the actual free response output when changing the input variables ($\Delta\boldsymbol{u}(t)$, the difference from the known course $\boldsymbol{u}_0(t)$), under zero initial conditions.

$$\underbrace{\boldsymbol{y}(\boldsymbol{x}(t), \boldsymbol{u}(t), t \geq t_0)}_{system\ output} = \underbrace{\boldsymbol{y}_{fr}\big(\boldsymbol{x}_{fr}(t), \boldsymbol{u}_0(t), t \geq t_0\big)}_{free\ response} + \underbrace{\boldsymbol{y}_{fo}\big(\boldsymbol{x}_{fo}(t), \Delta\boldsymbol{u}(t), t \geq t_0\big)}_{forced\ response} \tag{3.2}$$
$$\boldsymbol{x}_{fr}(t_0) = x(t_0) \qquad \boldsymbol{x}_{fo}(t_0) = 0 \qquad \Delta\boldsymbol{u}(t) = \boldsymbol{u}(t) - \boldsymbol{u}_0(t)$$

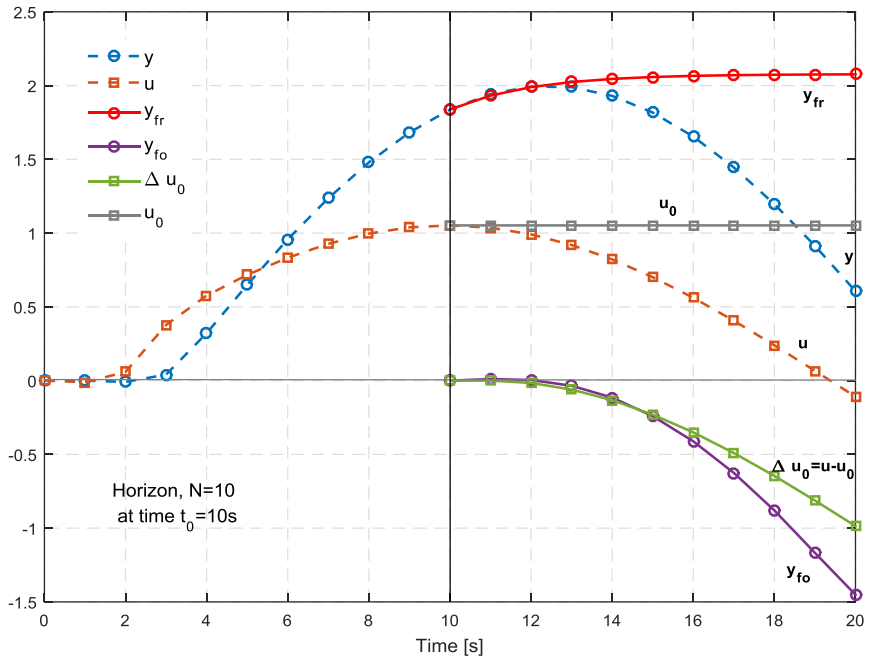

Figure 2 Prediction – decomposition into free and force response

If we know the system eq (3.1) and the initial state vector, the free response $\boldsymbol{y}_{fr}$ in eq (3.2) can be determined from the process (LTI/LTV) model – with the information of the past state variables and inputs. The forced response $\boldsymbol{y}_{fo}$ depends on the process model and unknown input variable $\Delta\boldsymbol{u}$. Hence, the objective is to calculate the course of the variable $\Delta\boldsymbol{u}$, in such a way that the sum of free and forced responses will have the desired course.

For control design purposes, we consider the discrete time representation. The prediction horizon N (number of samples), is the time interval for which the control inputs are calculated, solving a constraint optimal control problem for the current state of control. There are two main approaches when considering the horizon – input horizon N_u (the horizon at which control input is considered) and output horizon N_y (the horizon at which the predicted output is considered). For the sake of simplicity, the input horizon is chosen to be the same as the output horizon i.e. $N_u = N_y = N$ in the following formulation.

3.1 Linear MPC – output prediction with linear model

Let the plant model, eq (3.1), be linearized into a discrete time LTI state space model with a sampling rate of T_s s,

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \\ \boldsymbol{y}_k &= \boldsymbol{C}\boldsymbol{x}_k + \boldsymbol{D}\boldsymbol{u}_k \end{aligned} \tag{3.3}$$

where,

$$\boldsymbol{x} \in \mathbb{R}^{n_x} \quad \boldsymbol{u} \in \mathbb{R}^{n_u} \quad \boldsymbol{y} \in \mathbb{R}^{n_y}$$
$$\boldsymbol{A} \in \mathbb{R}^{n_x \times n_x} \quad \boldsymbol{B} \in \mathbb{R}^{n_x \times n_u} \quad \boldsymbol{C} \in \mathbb{R}^{n_y \times n_x} \quad \boldsymbol{D} \in \mathbb{R}^{n_y \times n_u}$$

For n_x state variables, n_u control variables and n_y output variables. The matrix $\boldsymbol{D}$ allows direct coupling between inputs and outputs and is absent in most of the applications ($\boldsymbol{D} = \boldsymbol{0}$).

Prediction model

The future state variables and outputs are recursively calculated from the linear model. For e.g. at sampling time, $t = (k+1)T_s$,

$$\begin{aligned} \boldsymbol{x}_{k+2} &= \boldsymbol{A}(\boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k) + \boldsymbol{B}\boldsymbol{u}_{k+1} \\ &= \boldsymbol{A}^{\boldsymbol{2}}\boldsymbol{x}_k + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{B}\boldsymbol{u}_{k+1} \end{aligned}$$

$$\boldsymbol{y}_{k+1} = \boldsymbol{C}(\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{u}_k) + \boldsymbol{D}\boldsymbol{u}_{k+1}$$

At sampling time, $t = (k+2)T_s$,

$$\begin{aligned} \boldsymbol{x}_{k+3} &= \boldsymbol{A}\left(\boldsymbol{A}^{\boldsymbol{2}}\boldsymbol{x}_k + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{B}\boldsymbol{u}_{k+1}\right) + \boldsymbol{B}\boldsymbol{u}_{k+2} \\ &= \boldsymbol{A}^{\boldsymbol{3}}\boldsymbol{x}_k + \boldsymbol{A}^{\boldsymbol{2}}\boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_{k+1} + \boldsymbol{B}\boldsymbol{u}_{k+2} \end{aligned}$$

$$\boldsymbol{y}_{k+2} = \boldsymbol{C}\left(\boldsymbol{A}^{\boldsymbol{2}}\boldsymbol{x}_k + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_k + \boldsymbol{B}\boldsymbol{u}_{k+1}\right) + \boldsymbol{D}\boldsymbol{u}_{k+1}$$

Representing in general form of the predicted state vector and output vectors for a horizon length N, at sampling time, t_k as,

$$\begin{aligned} \boldsymbol{x}_{k+N+1} &= \boldsymbol{S}_{xx}\boldsymbol{x}_k + \boldsymbol{S}_{xu}\boldsymbol{u}_N \\ \boldsymbol{y}_N &= \boldsymbol{S}_{yx}\boldsymbol{x}_k + \boldsymbol{S}_{\boldsymbol{yu}}\boldsymbol{u}_N \end{aligned} \tag{3.4}$$

where, $\boldsymbol{u}_N = \begin{bmatrix} u_k \\ \vdots \\ u_{k+N} \end{bmatrix}$; $\boldsymbol{y}_N = \begin{bmatrix} y_k \\ \vdots \\ y_{k+N} \end{bmatrix}$

$$\boldsymbol{S}_{xx} = \boldsymbol{A}^{N+1} \qquad \in \mathbb{R}^{n_x \times n_x}$$

$$\boldsymbol{S}_{xu} = \begin{bmatrix} \boldsymbol{A}^N\boldsymbol{B} & \boldsymbol{A}^{N-1}\boldsymbol{B} & \cdots & \boldsymbol{B} \end{bmatrix} \qquad \in \mathbb{R}^{n_x \times (N+1)*n_u}$$

$$\boldsymbol{S}_{yx} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C}\boldsymbol{A} \\ \boldsymbol{C}\boldsymbol{A}^{\boldsymbol{2}} \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}^N \end{bmatrix} \qquad \in \mathbb{R}^{(N+1)*n_y \times n_x}$$

$$\boldsymbol{S}_{yu} = \begin{bmatrix} \boldsymbol{D} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{C}\boldsymbol{B} & \boldsymbol{D} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{C}\boldsymbol{A}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{B} & \boldsymbol{D} & \cdots & \boldsymbol{0} \\ \boldsymbol{C}\boldsymbol{A}^{\boldsymbol{2}}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{B} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{C}\boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}^{N-2}\boldsymbol{B} & \boldsymbol{C}\boldsymbol{A}^{N-3}\boldsymbol{B} & \cdots & \boldsymbol{D} \end{bmatrix} \in \mathbb{R}^{(N+1)*n_x \times (N+1)*n_u}$$

The output, $\boldsymbol{y}_N$, is decomposed into the sum of free and forced responses. The free response depends on the current state and constant current input and forced response depends on zero initial condition and deviation control input as,

$$\boldsymbol{y}_N = \boldsymbol{y}_{fr,N}(\boldsymbol{x}_0, \boldsymbol{u}_{N,0}) + \boldsymbol{y}_{fo,N}(\boldsymbol{0}, \Delta\boldsymbol{u}_N)$$

where, $\boldsymbol{x}_0$ is a vector of state variables at t_k

$\boldsymbol{u}_{N,0}$ is a column vector of control input at time $(t_k + iT_s)\ \forall\ i = [0,1 \ldots N]$

$\Delta\boldsymbol{u}_N = \boldsymbol{u}_N - \boldsymbol{u}_{N,0}$

The free response is given by,

$$\begin{aligned} \boldsymbol{x}_{fr}(k+N+1) &= \boldsymbol{S}_{xx}\boldsymbol{x}_0 + \boldsymbol{S}_{xu}\boldsymbol{u}_{N,0} \\ \boldsymbol{y}_{fr,N} &= \boldsymbol{S}_{yx}\boldsymbol{x}_0 + \boldsymbol{S}_{yu}\boldsymbol{u}_{N,0} \end{aligned} \tag{3.5}$$

and the force response is derived as,

$$\begin{aligned} \boldsymbol{x}_{fo}(k+N+1) &= \boldsymbol{S}_{xu}\Delta\boldsymbol{u}_N \\ \boldsymbol{y}_{fo,N} &= \boldsymbol{S}_{yu}\Delta\boldsymbol{u}_N \end{aligned} \tag{3.6}$$

3.2 Non-Linear MPC – output prediction with non-linear model

Linear MPC is the most commonly used predictive controller because of the easiness of using the linear model in prediction and optimization. If a fairly accurate linear model of process is available, it's better to apply LMPC because of low computational requirements. However, most of the systems are non-linear and a non-linear feedback control of the system is inevitable. LMPC can be easily extended to NMPC by using a non-linear model. The simplest way to employ a non-linear model is by linearizing the system at many time instances (LTV model). Another possibility is to use the non-linear model as it is, with an ordinary differential equation (ODE) solver.

Let the discrete time state space LTV model of the continuous time model, eq (3.1) be written as,

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{A}_k\boldsymbol{x}_k + \boldsymbol{C}_k\boldsymbol{u}_k \\ \boldsymbol{y}_k &= \boldsymbol{C}_k\boldsymbol{x}_k + \boldsymbol{D}_k\boldsymbol{u}_k \end{aligned} \tag{3.7}$$

where $\boldsymbol{A}_k, \boldsymbol{B}_k, \boldsymbol{C}_k, \boldsymbol{D}_k$ are the time varying matrices at time $t = t_k$.

Prediction model

The predicted state variables and outputs at horizon *N* are derived as follows:

$$\begin{aligned} \boldsymbol{x}_{k+N+1} &= \boldsymbol{S}_{xx,k}\boldsymbol{x}_k + \boldsymbol{S}_{xu,k}\boldsymbol{u}_N \\ \boldsymbol{y}_N &= \boldsymbol{S}_{yx,k}\boldsymbol{x}_k + \boldsymbol{S}_{yu,k}\boldsymbol{u}_N \end{aligned}$$

The free response, either is obtained by the solution of nonlinear equation, eq (3.1) or can be derived from the LTV model as given by,

$$\begin{aligned} \boldsymbol{x}_{fr}(k+N+1) &= \boldsymbol{S}_{xx,k}\boldsymbol{x}_0 + \boldsymbol{S}_{xu,k}\boldsymbol{u}_{N,0} \\ \boldsymbol{y}_{fr,N} &= \boldsymbol{S}_{yx,k}\boldsymbol{x}_0 + \boldsymbol{S}_{yu,k}\boldsymbol{u}_{N,0} \end{aligned} \tag{3.8}$$

and the forced response is given by,

$$\begin{aligned} \boldsymbol{x}_{fo}(k+N+1) &= \boldsymbol{S}_{xu,k}\Delta\boldsymbol{u}_N \\ \boldsymbol{y}_{fo,N} &= \boldsymbol{S}_{yu,k}\Delta\boldsymbol{u}_N \end{aligned} \tag{3.9}$$

where,

$$\boldsymbol{S}_{xx,k} = \boldsymbol{A}_{k+N}\,\boldsymbol{A}_{k+N-1}\ldots\boldsymbol{A}_{k+1}\boldsymbol{A}_k$$

$$\boldsymbol{S}_{xu,k} = [\boldsymbol{A}_{k+N}\boldsymbol{A}_{k+N-1}\ldots\boldsymbol{A}_{k+1}\boldsymbol{B}_k \quad \boldsymbol{A}_{k+N}\boldsymbol{A}_{k+N-1}\ldots\boldsymbol{A}_{k+2}\boldsymbol{B}_{k+1} \quad \cdots \quad \boldsymbol{A}_{k+N}\boldsymbol{B}_{k+N-1} \quad \boldsymbol{B}_{k+N}]$$

$$\boldsymbol{S}_{yx,k} = \begin{bmatrix} \boldsymbol{C}_k \\ \boldsymbol{C}_{k+1}\boldsymbol{A}_k \\ \boldsymbol{C}_{k+2}\boldsymbol{A}_{k+1}\boldsymbol{A}_k \\ \vdots \\ \boldsymbol{C}_{k+N}\boldsymbol{A}_{k+N-1}\ldots\boldsymbol{A}_{k+1}\boldsymbol{A}_k \end{bmatrix}$$

$$\boldsymbol{S}_{yu,k} = \begin{bmatrix} \boldsymbol{D}_k & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{C}_{k+1}\boldsymbol{B}_k & \boldsymbol{D}_{k+1} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{C}_{k+2}\boldsymbol{A}_{k+1}\boldsymbol{B}_k & \boldsymbol{C}_{k+2}\boldsymbol{B}_{k+1} & \boldsymbol{D}_{k+2} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{C}_{k+N}\prod_{i=k+1}^{k+N-1}\boldsymbol{A}_{k+i}\,\boldsymbol{B}_k & \boldsymbol{C}_{k+N}\prod_{i=k+2}^{k+N-2}\boldsymbol{A}_{k+i}\,\boldsymbol{B}_{k+1} & \boldsymbol{C}_{k+N}\prod_{i=k+3}^{k+N-3}\boldsymbol{A}_{k+i}\,\boldsymbol{B}_{k+2} & \cdots & \boldsymbol{C}_{k+N}\boldsymbol{B}_{k+N-1} & \boldsymbol{D}_{k+N} \end{bmatrix}$$

3.3 Optimization

The main task of optimization is to achieve the best possible approximation of the smallest deviation of the system output, from the desired output waveform; by the smallest possible change in the system input, during future time interval (i.e. horizon). The mathematical formulation of this goal is an optimization task – choice of input to minimize a cost function, which expresses the "rate of fulfillment" of the goal.

The MPC allows a lot of flexibility in the choice of cost function. A general cost function consists of three parts: costs to penalize the control error during the horizon, costs to penalize the control effort during horizon, and a terminal cost to ensure stability of the control at the terminal state. The cost function is defined as follows:

$$\begin{aligned} J\big(N, \boldsymbol{x}_0, \boldsymbol{u}_{N,0}\big) &= \Delta\boldsymbol{x}^T(N)\boldsymbol{Q}_N\Delta\boldsymbol{x}(N) + \boldsymbol{e}_N{}^T\boldsymbol{Q}\boldsymbol{e}_N + \Delta\boldsymbol{u}_N{}^T\boldsymbol{R}\Delta\boldsymbol{u}_N \\ \boldsymbol{e}_N &= \boldsymbol{w}_N - \boldsymbol{y}_N \\ \Delta\boldsymbol{u}_N &= \boldsymbol{u}_N - \boldsymbol{u}_{N,0} \\ \boldsymbol{u}_{min,N} &< \boldsymbol{u}_N < \boldsymbol{u}_{max,N} \end{aligned} \tag{3.10}$$

where $\boldsymbol{Q}_N$ is the weighting matrix for the terminal state error, $\boldsymbol{Q}$ is the weighting matrix for control error and $\boldsymbol{R}$ is the weighting matrix for control effort.

$$\boldsymbol{w}_N = \begin{bmatrix} \boldsymbol{w}(t_k) \\ \vdots \\ \boldsymbol{w}(t_k + iT_s) \\ \vdots \\ \boldsymbol{w}(t_k + NT_s) \end{bmatrix} \in \mathbb{R}^{(N+1)n_y \times 1} \qquad \boldsymbol{y}_N = \begin{bmatrix} \boldsymbol{y}(t_k) \\ \vdots \\ \boldsymbol{y}(t_k + iT_s) \\ \vdots \\ \boldsymbol{y}(t_k + NT_s) \end{bmatrix} \in \mathbb{R}^{(N+1)n_y \times 1}$$

$$\boldsymbol{u}_N = \begin{bmatrix} \boldsymbol{u}(t_k) \\ \vdots \\ \boldsymbol{u}(t_k + iT_s) \\ \vdots \\ \boldsymbol{u}(t_k + NT_s) \end{bmatrix} \in \mathbb{R}^{(N+1)n_u \times 1} \qquad \boldsymbol{u}_{N,0} = \begin{bmatrix} u_{fr}(t_k) \\ \vdots \\ \boldsymbol{u}_{fr}(t_k + iT_s) \\ \vdots \\ \boldsymbol{u}_{fr}(t_k + NT_s) \end{bmatrix} \in \mathbb{R}^{(N+1)n_u \times 1}$$

and the deviation of the final state is given by,

$$\Delta \boldsymbol{x}(N) = \boldsymbol{x}_w(t_k + (N+1)T_s) - \boldsymbol{x}(t_k + (N+1)T_s) \quad \in \mathbb{R}^{n_x}$$

where, $\boldsymbol{x}_w$ is the desired terminal state, i.e. the steady state corresponding to the desired value at the end of horizon (valid only for square system).

Rewriting the criteria, eq (3.10) in terms of free and force response, we get,

$$J(N, \boldsymbol{x}_0, \boldsymbol{u}_{N,0}) = [\boldsymbol{x}_w(N) - \boldsymbol{x}_{fr}(N) - \boldsymbol{x}_{fo}(N)]^T \boldsymbol{Q}_N [\boldsymbol{x}_w(N) - \boldsymbol{x}_{fr}(N) - \boldsymbol{x}_{fo}(N)] + \\ + (\boldsymbol{w}_N - \boldsymbol{y}_{fr,N} - \boldsymbol{y}_{fo,N})^T \boldsymbol{Q} (\boldsymbol{w}_N - \boldsymbol{y}_{fr,N} - \boldsymbol{y}_{fo,N}) + \Delta \boldsymbol{u}_N{}^T \boldsymbol{R} \Delta \boldsymbol{u}_N$$

Substituting eq (3.6) and eq (3.9), then,

$$\begin{aligned} J(N, \boldsymbol{x}_0, \boldsymbol{u}_{N,0}) &= [\boldsymbol{x}_w(N) - \boldsymbol{x}_{fr}(N) - \boldsymbol{S}_{xu} \Delta \boldsymbol{u}_N]^T \boldsymbol{Q}_N [\boldsymbol{x}_w(N) - \boldsymbol{x}_{fr}(N) - \boldsymbol{S}_{xu} \Delta \boldsymbol{u}_N] + \\ &\quad + (\boldsymbol{w}_N - \boldsymbol{y}_{fr,N} - \boldsymbol{S}_{yu} \Delta \boldsymbol{u}_N)^T \boldsymbol{Q} (\boldsymbol{w}_N - \boldsymbol{y}_{fr,N} - \boldsymbol{S}_{yu} \Delta \boldsymbol{u}_N) + \Delta \boldsymbol{u}_N{}^T \boldsymbol{R} \Delta \boldsymbol{u}_N \\ &\quad \underbrace{= [\boldsymbol{x}_w(N) - \boldsymbol{x}_{fr}(N)]^T \boldsymbol{Q}_N [\boldsymbol{x}_w(N) - \boldsymbol{x}_{fr}(N)] + (\boldsymbol{w}_N - \boldsymbol{y}_{fr,N})^T \boldsymbol{Q} (\boldsymbol{w}_N - \boldsymbol{y}_{fr,N})}_{c} - \\ &\quad + \Delta \boldsymbol{u}_N{}^T \underbrace{\{\boldsymbol{S}_{xu}{}^T \boldsymbol{Q}_N [\boldsymbol{x}_w(N) - \boldsymbol{x}_{fr}(N)] + \boldsymbol{S}_{yu}{}^T \boldsymbol{Q} (\boldsymbol{w}_N - \boldsymbol{y}_{fr,N})\}}_{-\boldsymbol{m}} - \\ &\quad \underbrace{- \{[\boldsymbol{x}_w(N) - \boldsymbol{x}_{fr}(N)]^T \boldsymbol{Q}_N \boldsymbol{S}_{xu} + (\boldsymbol{w}_N - \boldsymbol{y}_{fr,N})^T \boldsymbol{Q} \boldsymbol{S}_{yu}\}}_{-\boldsymbol{m}^T} \Delta \boldsymbol{u}_N + \\ &\quad + \Delta \boldsymbol{u}_N{}^T \underbrace{[\boldsymbol{S}_{xu}{}^T \boldsymbol{Q}_N \boldsymbol{S}_{xu} + \boldsymbol{S}_{yu}{}^T \boldsymbol{Q} \boldsymbol{S}_{yu} + \boldsymbol{R}]}_{\boldsymbol{M}} \Delta \boldsymbol{u}_N \\ &= \Delta \boldsymbol{u}_N{}^T \boldsymbol{M} \Delta \boldsymbol{u}_N + \Delta \boldsymbol{u}_N{}^T \boldsymbol{m} + \boldsymbol{m}^T \Delta \boldsymbol{u}_N + \boldsymbol{c} \end{aligned} \tag{3.11}$$

where,

$$\boldsymbol{m} = -\boldsymbol{S}_{xu}{}^T \boldsymbol{Q}_N [\boldsymbol{x}_w - \boldsymbol{S}_{xx} \boldsymbol{x}_0 - \boldsymbol{S}_{xu} \boldsymbol{u}_{N,0}] + \boldsymbol{S}_{yu}{}^T \boldsymbol{Q} (\boldsymbol{w}_N - \boldsymbol{S}_{yx} \boldsymbol{x}_0 - \boldsymbol{S}_{yu} \boldsymbol{u}_{N,0})$$

$$\boldsymbol{M} = \boldsymbol{S}_{xu}{}^T \boldsymbol{Q}_N \boldsymbol{S}_{xu} + \boldsymbol{S}_{yu}{}^T \boldsymbol{Q} \boldsymbol{S}_{yu} + \boldsymbol{R}$$

In case of unconstraint control, an analytical solution is derived by differentiating with respect to $\Delta \boldsymbol{u}$. Assuming the weighting matrices ($\boldsymbol{Q}, \boldsymbol{Q}_N, \boldsymbol{R}$) to be positive semi definite, the control law is then in the form of,

$$\Delta \boldsymbol{u} = -\boldsymbol{M}^{-1}\boldsymbol{m} \tag{3.12}$$

In case of constraint control, the criteria, eq (3.11), is written in the form of a quadratic programming problem.

$$\begin{gathered}\min_{\Delta \boldsymbol{u}_N} J\left(N, \boldsymbol{x}_0, \boldsymbol{u}_{N,0}\right) = \Delta \boldsymbol{u}_N{}^T \boldsymbol{M} \Delta \boldsymbol{u}_N + \Delta \boldsymbol{u}_N{}^T \boldsymbol{m} + \boldsymbol{m}^T \Delta \boldsymbol{u}_N + \boldsymbol{c} \\ \boldsymbol{u}_{min} - \boldsymbol{u}_{N,0} < \Delta \boldsymbol{u}_N < \boldsymbol{u}_{max} - \boldsymbol{u}_{N,0}\end{gathered} \tag{3.13}$$

The optimal control action is the solution of the quadratic programing problem, obtained by minimizing the criteria.

$$\min_{\Delta \boldsymbol{u}} J = \frac{1}{2}\Delta \boldsymbol{u}^T M \Delta \boldsymbol{u} + \boldsymbol{m}^T \Delta \boldsymbol{u} \text{ such that } \boldsymbol{A_o} \Delta \boldsymbol{u} \leq \boldsymbol{b_0}$$

where the matrix $\boldsymbol{A_o}$ and the vector $\boldsymbol{b_0}$ are constraint matrices of control input,

$$\begin{gathered}\boldsymbol{u}_{min} \leq \boldsymbol{u}_N \leq \boldsymbol{u}_{max} \\ \boldsymbol{u}_{min} \leq \Delta \boldsymbol{u}_N + \boldsymbol{u}_{N,0} \leq \boldsymbol{u}_{max} \\ \boldsymbol{u}_{min} - \boldsymbol{u}_{N,0} \leq \Delta \boldsymbol{u}_N \leq \boldsymbol{u}_{max} - \boldsymbol{u}_{N,0}\end{gathered}$$

Representing the constraints for future control inputs for horizon *N*, is derived as,

$$\underbrace{\begin{bmatrix} \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{I} & \cdots & \boldsymbol{I} \\ -\boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ -\boldsymbol{I} & \cdots & -\boldsymbol{I} \end{bmatrix}}_{\boldsymbol{A_o}} \Delta \boldsymbol{u} \leq \underbrace{\begin{bmatrix} \boldsymbol{u}_{max} - \boldsymbol{u}_0 \\ \vdots \\ \boldsymbol{u}_{max} - \boldsymbol{u}_0 \\ -\boldsymbol{u}_{min} + \boldsymbol{u}_0 \\ \vdots \\ -\boldsymbol{u}_{min} + \boldsymbol{u}_0 \end{bmatrix}}_{\boldsymbol{b_0}} \tag{3.14}$$

where $\boldsymbol{u}_0$ is the last control action, $\boldsymbol{u}_0 = \boldsymbol{u}(k-1)$.

In MATLAB, the solution to the quadratic programing problem by the function `quadprog`.

```
Δu = quadprog(M,m,Ao,bo)
```

4. KINEMATICS OF WHEELED MOBILE ROBOT

Let the pose of the WMR in Cartesian coordinate with an angle θ_B, be measured clock wise from the x-axis,

$$\boldsymbol{x}_B = \begin{bmatrix} x_B \\ y_B \\ \theta_B \end{bmatrix}$$

The basic kinematic equations of the differential drive mobile robot are given by,

$$\begin{aligned} \dot{x}_B &= v_B \cos\theta_B \\ \dot{y}_B &= v_B \sin\theta_B \\ \dot{\theta}_B &= \omega_B \end{aligned}$$

where v_B and ω_B, are the tangential velocity and angular velocity respectively. This can be represented in matrix format,

$$\dot{\boldsymbol{x}}_B = \begin{bmatrix} \dot{x}_B \\ \dot{y}_B \\ \dot{\theta}_B \end{bmatrix} = \begin{bmatrix} \cos\theta_B & 0 \\ \sin\theta_B & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_B \\ \omega_B \end{bmatrix} \tag{4.1}$$

Trajectory of a mobile robot refers to the locus of all the points (x_B, y_B) in the Cartesian coordinate. In a trajectory-tracking approach to a mobile robot motion control problem, the reference trajectory must be known beforehand. A feasible trajectory considering, the velocity and acceleration limits, non-holonomic and holonomic constraints, and an obstacle free trajectory should be generated (by trajectory planner module). The reference trajectory Cartesian coordinates $(x_r\ y_r)$, orientation θ_r, and velocities $(v_r\ \omega_r)$ fulfil the same kinematic equations, eq (4.1), as,

$$\dot{\boldsymbol{x}}_r = \begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} \tag{4.2}$$

In trajectory tracking control of the WMR, the aim is to minimize the difference between a reference trajectory state vector and a current state vector of the mobile robot (i.e. tracking deviation).

$$\boldsymbol{x}_r - \boldsymbol{x}_B = \begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix} - \begin{bmatrix} x_B \\ y_B \\ \theta_B \end{bmatrix}$$

There are two major approaches on how to express the tracking deviation, which are further linearized to get an approximate linear model. The starting point of both the models is the basic kinematic equations eq (4.1), but the main difference lies in the choice of the co-ordinate frame of the mobile robot and reference trajectory.

4.1 Successive linearized kinematic model (M1)

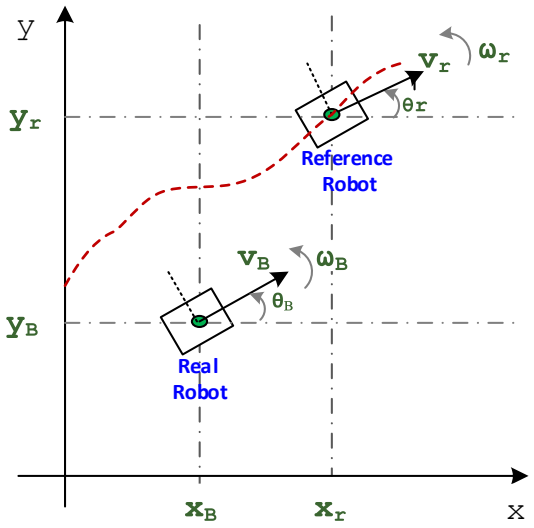

Figure 3 Coordinate System of Real Robot and Reference Robot

A linear model can be derived from the non-linear model, eq (4.1), by successively linearizing around the trajectory of the reference robot (see figure 3). A reference robot can be considered as a robot with reference (desired) parameters of the robot, which follows a reference trajectory. The kinematic equations eq (4.1) can be represented as a simple model,

$$\dot{\boldsymbol{x}}_B = f(\boldsymbol{x}_B, \boldsymbol{u}_B) \tag{4.3}$$

where state variables $\boldsymbol{x}_B = [x_B, y_B, \theta_B]^T$ and control inputs $\boldsymbol{u}_B = [v_B, \omega_B]^T$

Let the reference robot be following a reference trajectory (x_r, y_r) with an orientation of θ_r. The kinematic equations are same as that of the real mobile robot.

$$\dot{\boldsymbol{x}}_r = f(\boldsymbol{x}_r, \boldsymbol{u}_r) \tag{4.4}$$

The reference parameters are $[x_r\ y_r\ \theta_r\ v_r\ \omega_r]$. The tangential velocity, orientation angle and angular velocity of the reference robot can be calculated from eq (2.1-2.3). Applying the Taylor series approximation to eq (4.2), around the time varying reference points $(\boldsymbol{x_r}, \boldsymbol{u_r})$ we can derive,

$$\dot{\boldsymbol{x}}_B = f(\boldsymbol{x}_r, \boldsymbol{u}_r) + \left.\frac{\partial f(\boldsymbol{x}_B, \boldsymbol{u}_B)}{\partial \boldsymbol{x}_B}\right|_{\substack{\boldsymbol{x}_B = \boldsymbol{x}_r \\ \boldsymbol{u}_B = \boldsymbol{u}_r}} (\boldsymbol{x}_B - \boldsymbol{x}_r) + \left.\frac{\partial f(\boldsymbol{x}_B, \boldsymbol{u}_B)}{\partial \boldsymbol{u}_B}\right|_{\substack{\boldsymbol{x}_B = \boldsymbol{x}_r \\ \boldsymbol{u}_B = \boldsymbol{u}_r}} (\boldsymbol{u}_B - \boldsymbol{u}_r)$$

$$\dot{\boldsymbol{x}}_B = f(\boldsymbol{x}_r, \boldsymbol{u}_r) + \widetilde{\boldsymbol{A}}_S(\boldsymbol{x}_r, \boldsymbol{u}_r).\underbrace{(\boldsymbol{x}_B - \boldsymbol{x}_r)}_{\Delta \boldsymbol{x}} + \widetilde{\boldsymbol{B}}_S(\boldsymbol{x}_r, \boldsymbol{u}_r).\underbrace{(\boldsymbol{u}_B - \boldsymbol{u}_r)}_{\Delta \boldsymbol{u}} \tag{4.5}$$

Subtracting eq (4.5) from eq (4.4) gives,

$$\Delta \dot{\boldsymbol{x}} = \widetilde{\boldsymbol{A}}_S(\boldsymbol{x}_r, \boldsymbol{u}_r).\Delta \boldsymbol{x} + \widetilde{\boldsymbol{B}}_S(\boldsymbol{x}_r, \boldsymbol{u}_r).\Delta \boldsymbol{u} \tag{4.6}$$

where $\Delta \boldsymbol{x}$ is the error vector of state variables and $\Delta \boldsymbol{u}$ is the error vector of control variables with respect to the reference robot. The approximation of $\Delta \dot{\boldsymbol{x}}$ in eq (4.6), by the forward differences, gives the following discrete-time linear time-variant state-space model:

$$\Delta \boldsymbol{x}(k+1) = \boldsymbol{A}_S(k)\Delta \boldsymbol{x}(k) + \boldsymbol{B}_S(k)\Delta \boldsymbol{u}(k) \tag{4.7}$$

$$\boldsymbol{A}_s = \begin{bmatrix} 1 & 0 & -v_r \sin\theta_r(k) T_s \\ 0 & 1 & v_r \cos\theta_r(k)\, T_s \\ 0 & 0 & 1 \end{bmatrix};\ \boldsymbol{B}_s = \begin{bmatrix} \cos\theta_r(k)\, T_s & 0 \\ \sin\theta_r(k) T_s & 0 \\ 0 & T_s \end{bmatrix}$$

$$\Delta \boldsymbol{x} = \begin{bmatrix} x_B(k) - x_r(k) \\ y_B(k) - y_r(k) \\ \theta_B(k) - \theta_r(k) \end{bmatrix};\ \Delta \boldsymbol{u} = \begin{bmatrix} v_B(k) - v_r(k) \\ \omega_B(k) - \omega_r(k) \end{bmatrix}$$

where T_s is the sampling period and $\Delta \boldsymbol{x}$ is deviation state vector which represents the error with respect to the reference robot and $\Delta \boldsymbol{u}$ is associated with the control input. The reference values, v_r, θ_r, ω_r are the reference tangential velocity, orientation angle and angular velocity.

4.2 Tracking error based linear model (M2)

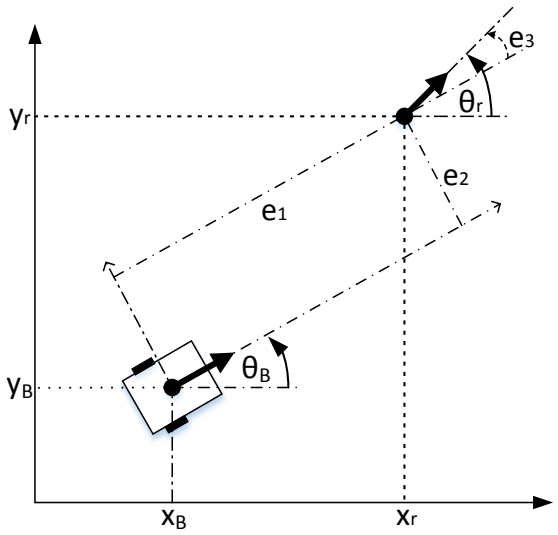

Figure 4 Tracking error based linear model: Coordinate System of Real Robot and Reference Robot

Another way of modeling is to consider the difference, in the local coordinate system of the mobile robot, see figure 4. This differences in local coordinate system is called as "tracking error" represented as,

$$\boldsymbol{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\theta_B & \sin\theta_B & 0 \\ -\sin\theta_B & \cos\theta_B & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x_B \\ y_r - y_B \\ \theta_r - \theta_B \end{bmatrix} = \boldsymbol{T}_{\boldsymbol{x}}(\boldsymbol{x}_r - \boldsymbol{x}_B) \tag{4.8}$$

where, $\boldsymbol{T}_{\boldsymbol{x}}$ is the coordinate transformation matrix. Now the aim of the trajectory tracking controller is to $\lim_{t\to\infty} \boldsymbol{e}(t) = \boldsymbol{0}$. Differentiating eq (4.8) by considering eq (4.1) and eq (4.2),

$$
\begin{aligned}
\dot{e}_1 &= (\dot{x}_r - \dot{x}_B)\cos\theta_B + (\dot{y}_r - \dot{y}_B)\sin\theta_B - (x_r - x_B)\sin\theta_B\,\dot{\theta}_B + (y_r - y_B)\cos\theta_B\,\dot{\theta}_B \\
&= e_2\omega_B + \dot{x}_r\cos\theta_B + \dot{y}_r\sin\theta_B - v\,cos^2\theta_B - v\,sin^2\theta_B \\
&= e_2\omega_B - v_B + \dot{x}_r\cos\theta_B + \dot{y}_r\sin\theta_B \\
&= e_2\omega_B - v_B + \dot{x}_r\cos(\theta_r - \theta_e) + \dot{y}_r\sin(\theta_r - \theta_e) \\
&= e_2\omega_B - v_B + \dot{x}_r(\cos\theta_r\cos\theta_e + \sin\theta_r\sin\theta_e) + \dot{y}_r(\sin\theta_r\cos\theta_e - \cos\theta_r\sin\theta_e) \\
&= e_2\omega_B - v_B + (\dot{x}_r\cos\theta_r + \dot{y}_r\sin\theta_r)\cos\theta_e + (\dot{x}_r\sin\theta_r - \dot{y}_r\cos\theta_r)\sin\theta_e \\
&= e_2\omega_B - v_B + v_r\cos\theta_e \\
&= e_2\omega_B - v_B + v_r\cos e_3
\end{aligned}
$$

$$
\begin{aligned}
\dot{e}_2 &= -(\dot{x}_r - \dot{x}_B)\sin\theta_B + (\dot{y}_r - \dot{y}_B)\cos\theta_B - (x_r - x_B)\cos\theta_B\,\dot{\theta}_B - (y_r - y_B)\sin\theta_B\,\dot{\theta}_B \\
&= -e_1\omega_B - \dot{x}_B\sin\theta_B + \dot{y}_B\cos\theta_B - \dot{x}_r\sin\theta_B + \dot{y}_r\cos\theta_B \\
&= -e_1\omega_B - \dot{x}_r\sin(\theta_r - \theta_e) + \dot{y}_r\cos(\theta_r - \theta_e) \\
&= -e_1\omega_B - \dot{x}_r\sin(\sin\theta_r\cos\theta_e - \cos\theta_r\sin\theta_e) + \dot{y}_r(\cos\theta_r\cos\theta_e + \sin\theta_r\sin\theta_e) \\
&= -e_1\omega_B - (\dot{x}_r\cos\theta_r + \dot{y}_r\sin\theta_r)\sin\theta_e + (\dot{y}_r\cos\theta_r - \dot{x}_r\sin\theta_r)\cos\theta_e \\
&= -e_1\omega_B + v_r\sin\theta_e \\
&= -e_1\omega_B + v_r\sin e_3
\end{aligned}
$$

Rearranging in matrix format,

$$
\dot{\boldsymbol{e}} = \begin{bmatrix}\dot{e}_1\\ \dot{e}_2\\ \dot{e}_3\end{bmatrix} = \begin{bmatrix} e_2\omega_B - v_B + v_r\cos e_3 \\ -e_1\omega_B + v_r\sin e_3 \\ \omega_r - \omega_B \end{bmatrix} \tag{4.9}
$$

In order to get a linear model, eq (4.9) is linearized around the equilibrium point ($\boldsymbol{e} = \mathbf{0}$), and the operating points as $[v_B\ \omega_B] = [v_r\ \omega_r]$ and by the approximation $\sin\theta \approx \theta$ (at small angles – θ is error variable and the aim is to minimize the errors), we arrive at,

$$
\begin{bmatrix}\dot{e}_1\\ \dot{e}_2\\ \dot{e}_3\end{bmatrix} = \begin{bmatrix} 0 & \omega_r & 0 \\ -\omega_r & 0 & v_r \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix}e_1\\ e_2\\ e_3\end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} v_r\cos e_3 - v_B \\ \omega_r - \omega_B \end{bmatrix}
$$

The continuous time state space model after linearization and approximation is given by,

$$
\begin{bmatrix}\dot{e}_1\\ \dot{e}_2\\ \dot{e}_3\end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \omega_r & 0 \\ -\omega_r & 0 & v_r \\ 0 & 0 & 0 \end{bmatrix}}_{\widetilde{\boldsymbol{A}}_E}\begin{bmatrix}e_1\\ e_2\\ e_3\end{bmatrix} + \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}_{\widetilde{\boldsymbol{B}}_E}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{4.10}
$$

Separating control inputs as feedforward and feedback inputs,

$$
\boldsymbol{u}_{fb} = \begin{bmatrix}u_1\\ u_2\end{bmatrix} = \begin{bmatrix}v_{fb}\\ \omega_{fb}\end{bmatrix} = \underbrace{\begin{bmatrix} v_r\cos e_3 \\ \omega_r \end{bmatrix}}_{\boldsymbol{u}_{ff}} - \underbrace{\begin{bmatrix} v_B \\ \omega_B \end{bmatrix}}_{\boldsymbol{u}_B} \tag{4.11}
$$

eq (4.11) can be seen as the transformation of $\boldsymbol{u}_r$ into the local coordinate system of the robot:

$$
\boldsymbol{u}_{fb} = \boldsymbol{T}_{\boldsymbol{u}}(\boldsymbol{u}_r) - \boldsymbol{u}_B
$$

where, $\boldsymbol{T}_{\boldsymbol{u}}$ is the transformation function**.** After discretizing with a sample time of T_s, then the discrete time LTV state space model is given by,

$$
\boldsymbol{e}(k+1) = \boldsymbol{A}_E(k)\boldsymbol{e}(k) + \boldsymbol{B}_E(k)\boldsymbol{u}_{fb}(k) \tag{4.12}
$$

where, $\boldsymbol{A}_E$ and $\boldsymbol{B}_E$ are discretized version of matrices $\widetilde{\boldsymbol{A}}_E$ and $\widetilde{\boldsymbol{B}}_E$ as given by,

$$\boldsymbol{A}_E(k) = \begin{bmatrix} 1 & \omega_r(k)T_s & 0 \\ -v_r(k)T_s & 1 & v_r(k)T_s \\ 0 & 0 & 1 \end{bmatrix};\ \boldsymbol{B}_E(k) = \begin{bmatrix} T_s & 0 \\ 0 & 0 \\ 0 & T_s \end{bmatrix}$$

4.3 Model verification – comparison of linear model vs non-linear model

The linearized models, eq (4.7) and eq (4.12), are verified by considering the error between the discrete approximations (linear model) and the continuous time nonlinear kinematic equations, eq (4.1). The real robot and the virtual reference robot (reference trajectory tracked in ideal conditions) are simulated with different sets of control inputs and initial conditions. The linear model is simulated with deviations in control input and initial conditions.

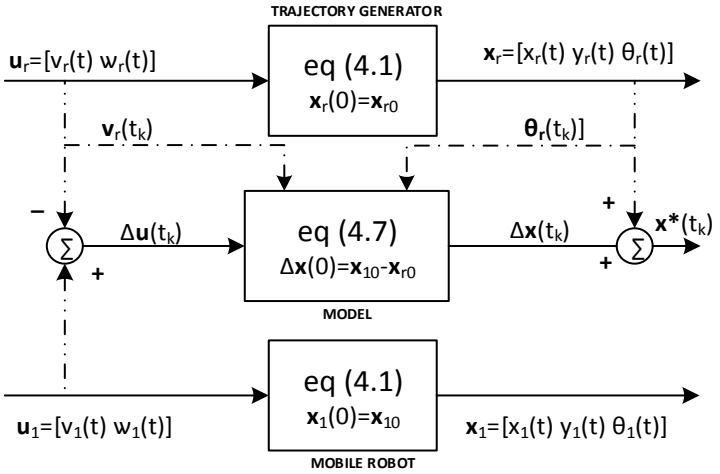

Figure 5 Comparison scheme for successive linear model

The smooth continuous time reference trajectory (x_r, y_r, θ_r) is generated with eq (4.1) based on the initial condition $\boldsymbol{x}_{r0}$ and with input $\boldsymbol{u}_r$. Let the real robot be driven with initial condition $\boldsymbol{x}_{10}$ and input $\boldsymbol{u}_1$.

The same trajectory is calculated by the discrete time linear model (**M1**), eq (4.7), with zero initial condition. The approximated trajectory is then $\boldsymbol{x}^*(t) = \boldsymbol{x}_r(t) + \Delta\boldsymbol{x}(t)$ and the input is the deviation variable, $\Delta\boldsymbol{u}(t) = \boldsymbol{u}_r(t) - \boldsymbol{u}_1(t)$. The block diagram in figure 5 explains the simulation scheme in detail.

The open loop simulation of error tracking model (**M2**) consists of a feed forward control input part. The approximated trajectory is calculated with the discrete time LTV model, eq (4.12), with initial condition $\boldsymbol{e_0}$ (which is coordinate transformed initial condition $\Delta\boldsymbol{x}(0)$). The input to the model $\boldsymbol{u}_{fb}$ is the difference between feedforward input and original input as in eq (4.11). The error state is transformed back to global coordinates and combined with

feedforward outputs (reference variables) to get the approximated trajectory, $\boldsymbol{x}^*$. See figure 6 for more details.

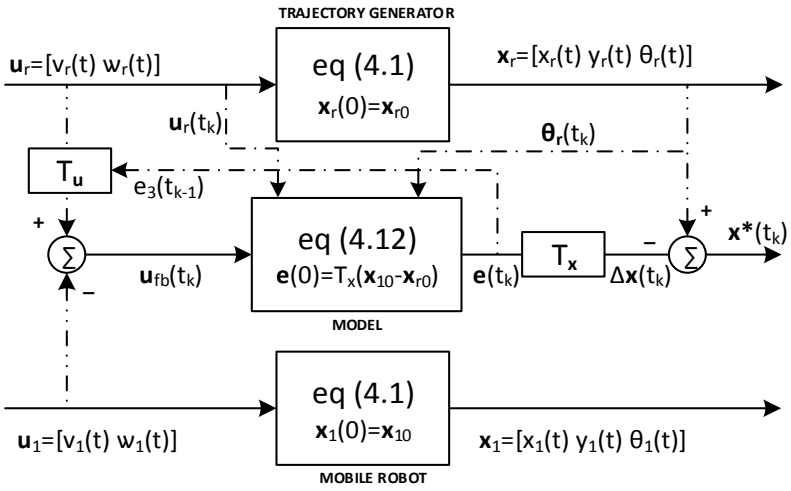

Figure 6 Comparison scheme for tracking error linear model

The input (velocities to move the robot in the reference trajectory) to the reference generator is,

$$\boldsymbol{u}_r = \begin{bmatrix} v_r \\ \omega_r \end{bmatrix} = \begin{bmatrix} 1 + \sin\left(\frac{\pi}{15}t\right) \\ -\sin\left(\frac{\pi}{15}t\right) \end{bmatrix} \quad 0 \le t \le 15\ s \tag{4.13}$$

The same initial condition for the trajectory generator is used for all the simulation schemes with a sampling time, $T_s = 0.1s$ for the discrete time linear models (**M1** and **M2**). Three inputs to the mobile robot are used in the simulation – with same linear velocities but different angular velocities (zeros, time varying, constant) as,

$$\left.\begin{aligned} \mathbf{IN1}: &= \boldsymbol{u}_r + \begin{bmatrix} \sin\left(\frac{2\pi}{7.5}t_k\right) \\ 0 \end{bmatrix} \\ \mathbf{IN2}: &= \boldsymbol{u}_r + \begin{bmatrix} \sin\left(\frac{2\pi}{7.5}t_k\right) \\ 0.2\ \sin\left(\frac{2\pi}{3}t_k\right) \end{bmatrix} \\ \mathbf{IN3}: &= \boldsymbol{u}_r + \begin{bmatrix} \sin\left(\frac{2\pi}{7.5}t_k\right) \\ -0.1 \end{bmatrix} \end{aligned}\right\}\ t_k = 0 \le T_s \le 15$$

Figures 7 -10 show the simulation results. Table 1 lists the simulation parameters – Initial Conditions (IC), input and Sum Square Error (SSE). The SSE is calculated by,

$$SSE_{xy\theta} = |\boldsymbol{x}_1 - \boldsymbol{x}^*|^2$$

The simulation scheme **S1-S3** shows the trajectories, approximation errors and inputs after applying the three inputs with same initial conditions for both real and reference robot. There is no significant difference in SSEs between both the models. The simulation scheme **S4-S7**

shows the trajectories when the initial condition (starting location of robot and first point to be tracked) is different. The linear model **M2** more closely follows the real robot trajectory than the model **M1** and the approximation error is comparatively smaller. The problem with successive linearization model, **M1**, is that it does not consider the initial tracking error, and as a result of this, a large approximation error will be accumulated over the time of tracking.

Table I Model verification – Input, Initial condition and SSE

		S1	S2	S3	S4	S5	S6	S7
		fig. 7	fig. 8	fig. 9	fig. 10	fig. 10	fig. 10	fig. 10
IC	$\mathbf{x}_{r0}$	[0 0 -pi/4]						
	$\mathbf{x}_{10}$	[0 0 -pi/4]	[0 0 -pi/4]	[0 0 -pi/4]	[0.5 0.5 -pi/4]	[0 0 -pi/6]	[0.5 0.5 -pi/6]	[0.5 0.5 -pi/6]
Input	$\mathbf{u}_r$	eq (4.13)						
	$\mathbf{u}_1$	**IN1**	**IN2**	**IN3**	**IN1**	**IN1**	**IN1**	**IN2**
SSE	**M1**	0.0902	13.8958	408.7267	0.0919	106.8508	106.8502	166.6250
	M2	2.6181	8.9507	499.3891	2.0085	0.9642	1.4009	4.7890
Remarks		IC: Same	IC: Same	IC: Same	IC: different location	IC: different orientation	IC: different orientation & location	IC: different orientation & location

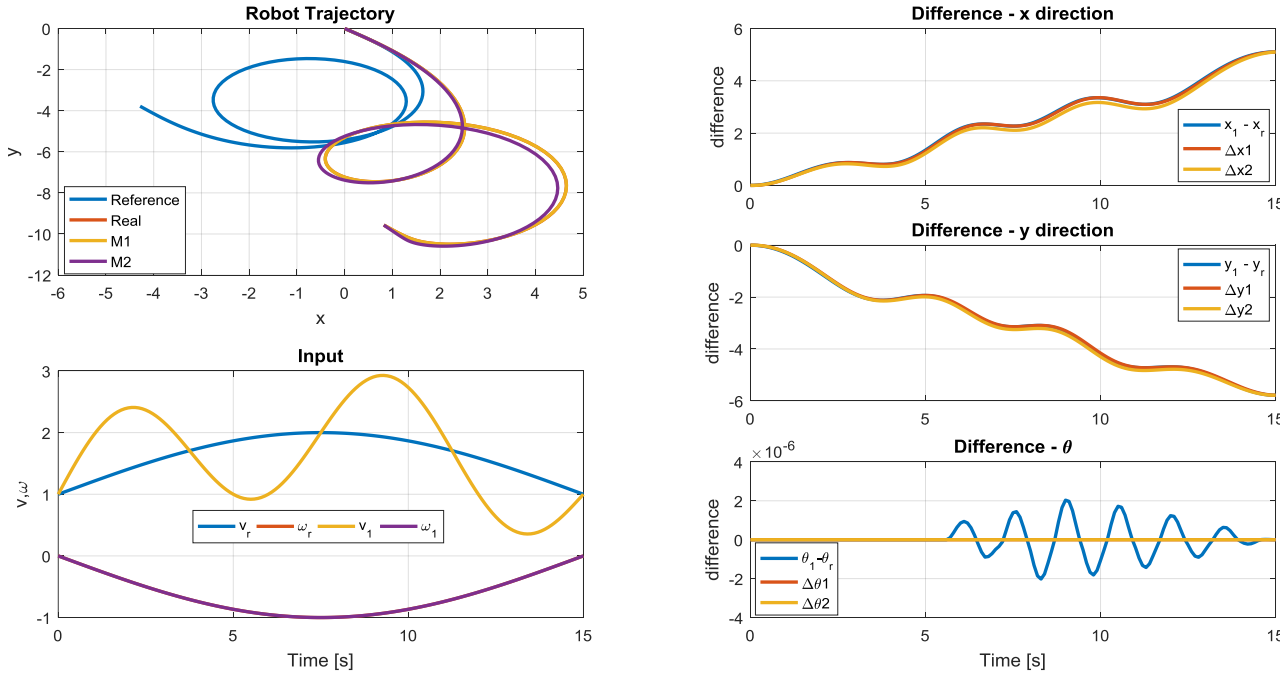

Figure 7 Model verification simulation scheme S1

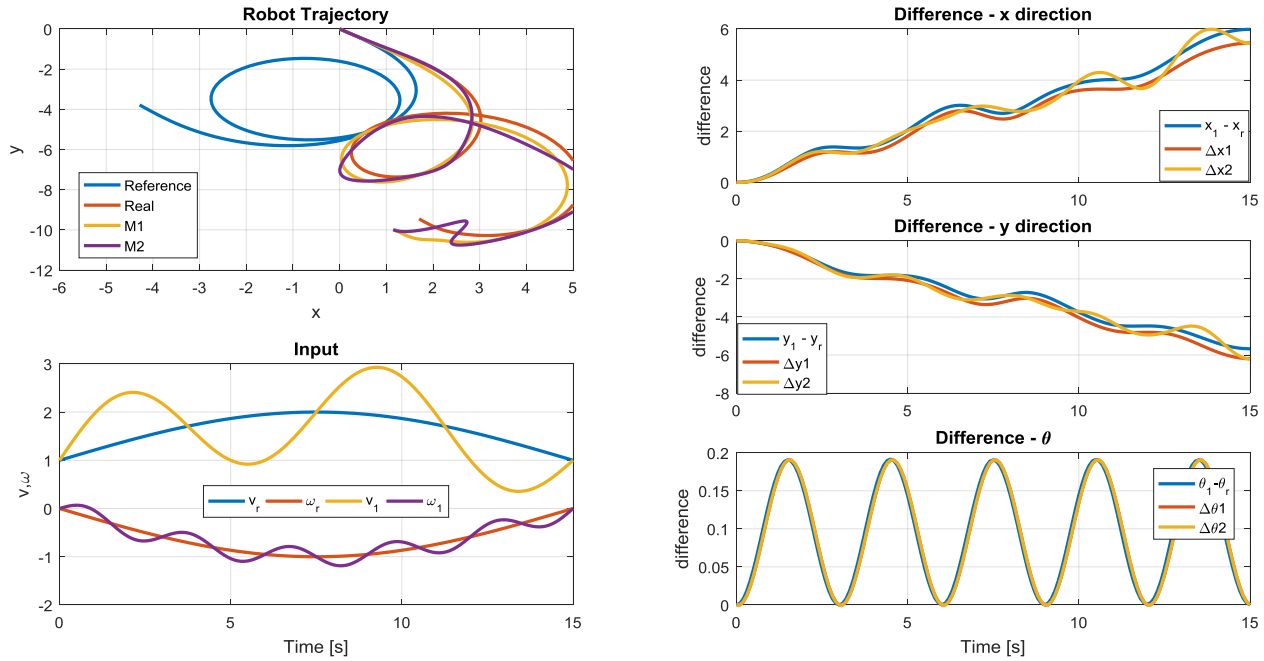

Figure 8 Model verification simulation scheme S2

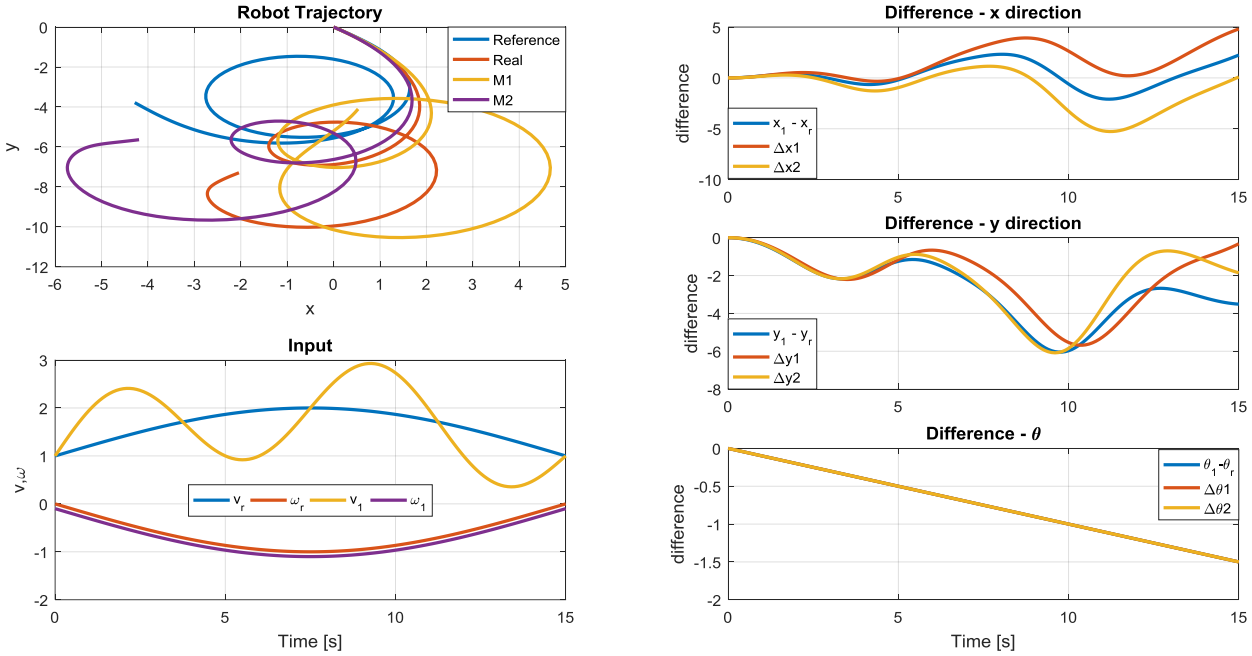

Figure 9 Model verification simulation scheme S3

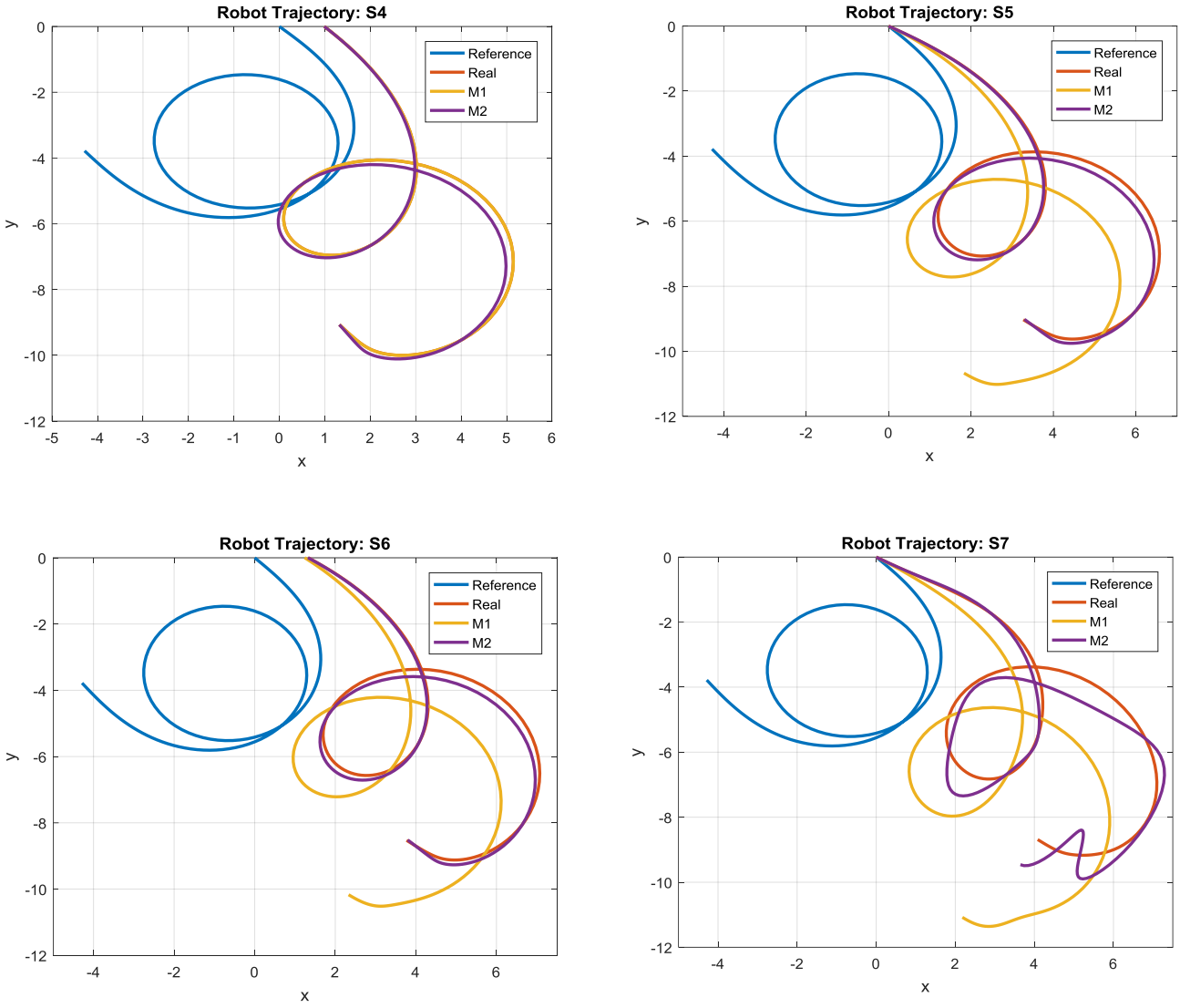

Figure 10 Model verification simulation scheme S4-S7

5. TRAJECTORY TRACKING OF WMR – KINEMATIC CONTROLLER

The kinematic modelling of WMR was discussed in Chapter 4. Two LTV models were derived based on the choice of coordinate frame. This chapter presents the trajectory tracking of the mobile robot by considering the kinematics. A perfect velocity-tracking controller (which is able to track the desired velocities perfectly) is assumed at the lower (dynamics) level. Non-linear MPCs based on the two models are designed with constraints on inputs and compared with state-of-the-art (Kanayama and Samson) controllers.

The trajectory tracking of WMR in a reference trajectory is achieved by generating tangential and angular velocities by discontinuous discrete feedback. At every time instant, control inputs are generated based on the tracking error (difference between real and reference points). The basic non-linear model of WMR based on kinematics, as discussed in Section 4 is,

$$\begin{aligned}\dot{x}_B &= v_B \cos\theta_B \\ \dot{y}_B &= v_B \sin\theta_B \\ \dot{\theta}_B &= \omega_B\end{aligned}$$

The successive linear model is derived by successively linearizing around the reference trajectory points with respect to world coordinates. The resultant LTV model has states, inputs and outputs as deviation variables from the reference trajectory. Figure 11 shows the control scheme. The discrete state space model by successive linearization, as given by eq (4.7) is,

$$\Delta\boldsymbol{x}(k+1) = \boldsymbol{A}_s(v_r(k), \theta_r(k))\Delta\boldsymbol{x}(k) + \boldsymbol{B}_s(\theta_r(k))\Delta\boldsymbol{u}(k)$$

where,

$$\boldsymbol{A}_s = \begin{bmatrix} 1 & 0 & -v_r \sin\theta_r(k) T_s \\ 0 & 1 & v_r \cos\theta_r(k)\, T_s \\ 0 & 0 & 1 \end{bmatrix};\ \boldsymbol{B}_s = \begin{bmatrix} \cos\theta_r(k)\, T_s & 0 \\ \sin\theta_r(k) T_s & 0 \\ 0 & T_s \end{bmatrix}$$

$$\Delta\boldsymbol{x} = \begin{bmatrix} x_B(k) - x_r(k) \\ y_B(k) - y_r(k) \\ \theta_B(k) - \theta_r(k) \end{bmatrix};\ \Delta\boldsymbol{u} = \begin{bmatrix} v_B(k) - v_r(k) \\ \omega_B(k) - \omega_r(k) \end{bmatrix}$$

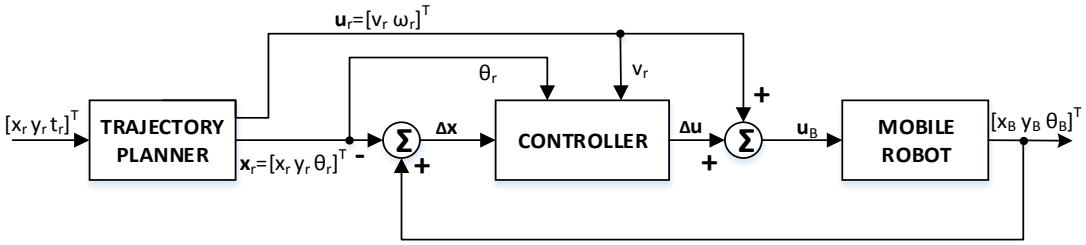

Figure 11 General block diagram of trajectory tracking kinematic controller with successive linear model

The reference parameters $[v_r, \omega_r, \theta_r]$ are generated by the trajectory generating module as discussed in section 2.1.

$$\left.\begin{aligned} v_r(t_k) &= \sqrt{\frac{x_r(t_k)-x_r(t_{k-1})}{T_s} + \frac{y_r(t_k)-y_r(t_{k-1})}{T_s}} \\ \theta_r(t_k) &= arctan2\left(\frac{y_r(t_k)-y_r(t_{k-1})}{T_s}, \frac{x_r(t_k)-x_r(t_{k-1})}{T_s}\right) \\ \omega_r(t_k) &= \frac{\theta_r(t_k)-\theta_r(t_{k-1})}{T_s} \end{aligned}\right\}$$

In error-based modeling, the error is considered in the local coordinate system of the mobile robot. The world co-ordinate pose error is transformed into the mobile robot coordinate frame, eq (4.8), as,

$$\boldsymbol{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta_B & \sin\theta_B & 0 \\ -\sin\theta_B & \cos\theta_B & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{T}_x} \begin{bmatrix} x_r - x_B \\ y_r - y_B \\ \theta_r - \theta_B \end{bmatrix}$$

Discretizing with a sample time of T_s, then the discrete time LTV state space model, eq (4.12), is given by,

$$\boldsymbol{e}(k+1) = \boldsymbol{A}_E\big(\omega_r(k), v_r(k)\big)\boldsymbol{e}(k) + \boldsymbol{B}_E\boldsymbol{u}_{fb}(k)$$

where,

$$\boldsymbol{A}_E(k) = \begin{bmatrix} 1 & T_s\omega_r(k) & 0 \\ -T_s\omega_r(k) & 1 & T_s v_r(k) \\ 0 & 0 & 1 \end{bmatrix}; \boldsymbol{B}_E(k) = \begin{bmatrix} T_s & 0 \\ 0 & 0 \\ 0 & T_s \end{bmatrix}$$

and the real control input is sum of the error-based model feedback control input and feedforward input as,

$$\boldsymbol{u}_{fb} = \underbrace{\begin{bmatrix} v_r \cos e_3 \\ \omega_r \end{bmatrix}}_{\boldsymbol{u}_{ff}} - \underbrace{\begin{bmatrix} v_B \\ \omega_B \end{bmatrix}}_{\boldsymbol{u}_B}$$

Figure 12 shows the general control scheme of the kinematic controller, which uses error-based model.

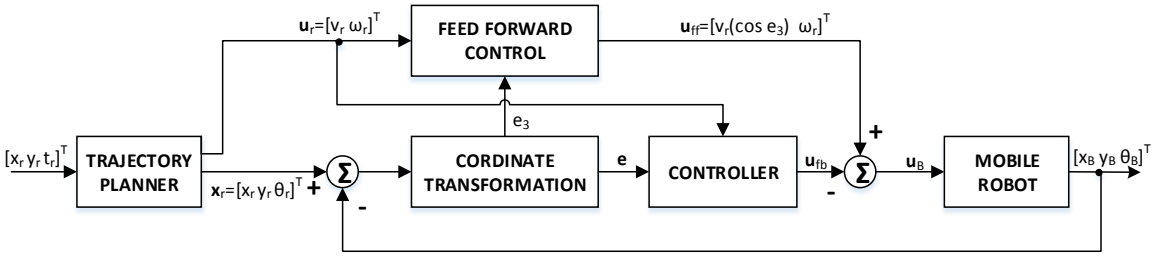

Figure 12 General block diagram of trajectory tracking kinematic controller with error-based linear model

5.1 Trajectory tracking by NMPC

The kinematics of non-holonomic robot is non-linear which requires a non-linear controller to track the trajectory. NMPC has several advantages over other controllers, mainly because of the possibility to use the knowledge of future set-points (trajectory way points).

The formulation of NMPC of trajectory tracking follows the same formulation as discussed in Section 3, with slight changes. Since the state variables correspond to the output of system, the term output, $\boldsymbol{y}_K$, in the formulation, is omitted. The kinematic LTV model is a MIMO system with 2 inputs and 3 outputs. The linear discrete time state space model, consists of only the state equation represented as,

$$\overline{\boldsymbol{x}}_{K_{k+1}} = \boldsymbol{A}_{K_k}\overline{\boldsymbol{x}}_{K_k} + \boldsymbol{B}_{K_k}\overline{\boldsymbol{u}}_{K_k} \tag{5.1}$$

where, $\overline{\boldsymbol{x}}_{K_k}$ is the state variable, and matrices $\boldsymbol{A}_{K_k}$, $\boldsymbol{B}_{K_k}$ are the same as that in eq (4.7) for the successive linear model and eq (4.12) for the error-based model.

5.1.1 Prediction model from LTV model

The prediction equations are same as that discussed in Section 3, but with time varying matrices.

$$\overline{\boldsymbol{X}}_{K_N} = \boldsymbol{S}_{K_{xx,k}}\overline{\boldsymbol{x}}_{K_k} + \boldsymbol{S}_{K_{xu,k}}\overline{\boldsymbol{u}}_{K_N} \tag{5.2}$$

where,

$$\overline{\boldsymbol{u}}_{K_N} = \begin{bmatrix} \overline{\boldsymbol{u}}_{K_k} \\ \vdots \\ \overline{\boldsymbol{u}}_{K_{k+N-1}} \end{bmatrix} \in \mathbb{R}^N;\ \overline{\boldsymbol{X}}_{K_N} = \begin{bmatrix} \overline{\boldsymbol{x}}_{K_{k+1}} \\ \vdots \\ \overline{\boldsymbol{x}}_{K_{k+N}} \end{bmatrix} \in \mathbb{R}^{Nn_x}$$

Decomposing eq (5.2) into free $\overline{\boldsymbol{X}}_{K_{fr,N}}$ and forced responses $\overline{\boldsymbol{X}}_{K_{fo,N}}$,

$$\overline{\boldsymbol{X}}_{K_{fr,N}} = \boldsymbol{S}_{K_{xx,k}}\overline{\boldsymbol{x}}_{K_k} + \boldsymbol{S}_{K_{xu,k}}\overline{\boldsymbol{u}}_{K_{N,0}} \tag{5.3}$$

$$\overline{\boldsymbol{X}}_{K_{fo,N}} = \boldsymbol{S}_{K_{xu,k}}\Delta\boldsymbol{u}_{K_N} \tag{5.4}$$

where,

$$\Delta\boldsymbol{u}_{K_N} = \overline{\boldsymbol{u}}_{K_N} - \overline{\boldsymbol{u}}_{K_{N,0}} \quad ; \quad \overline{\boldsymbol{u}}_{K_{N,0}} = \begin{bmatrix} \overline{\boldsymbol{u}}_{K_{fr,k}} \\ \vdots \\ \overline{\boldsymbol{u}}_{K_{fr,k+N-1}} \end{bmatrix} \in \mathbb{R}^N$$

and the time varying prediction matrices are,

$$\boldsymbol{S}_{K_{xx,k}} = \begin{bmatrix} \boldsymbol{A}_{K_k} \\ \boldsymbol{A}_{K_{k+1}}\boldsymbol{A}_{K_k} \\ \boldsymbol{A}_{K_{k+2}}\boldsymbol{A}_{K_{k+1}}\boldsymbol{A}_{K_k} \\ \vdots \\ \boldsymbol{A}_{K_{k+N}}\boldsymbol{A}_{K_{k+N-1}}\dots\boldsymbol{A}_{K_{k+1}}\boldsymbol{A}_{K_k} \end{bmatrix}$$

$$\boldsymbol{S}_{K_{xu,k}} = \begin{bmatrix} \boldsymbol{B}_{K_k} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{A}_{K_{k+1}}\boldsymbol{B}_{K_k} & \boldsymbol{B}_{K_{k+1}} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{A}_{K_{k+2}}\boldsymbol{A}_{K_{k+1}}\boldsymbol{B}_{K_k} & \boldsymbol{A}_{K_{k+1}}\boldsymbol{B}_{K_{k+1}} & \boldsymbol{B}_{K_{k+2}} & \cdots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \prod_{i=k+N}^{k+N-1} \boldsymbol{A}_{K_i}\boldsymbol{B}_{K_k} & \prod_{i=k+N}^{k+N-2} \boldsymbol{A}_{K_i}\boldsymbol{B}_{K_{k+1}} & \prod_{i=k+N}^{k+N-3} \boldsymbol{A}_{K_i}\boldsymbol{B}_{K_{k+2}} & \cdots & \boldsymbol{A}_{K_{k+N}}\boldsymbol{B}_{K_{k+N-1}} & \boldsymbol{B}_{K_{\boldsymbol{k+N}}} \end{bmatrix}$$

5.1.2 Cost function

The aim of kinematic NMPCs is to bring the state variables closer to zero by optimizing a cost function. The state variables for both the error based and successive linear model are deviation variables but computed with respect to different coordinates. The MPC allows a lot of flexibility in the choice of cost function. A general cost function consists of three parts: costs to penalize the control error during the horizon, costs to penalize the control signal during horizon and the terminal cost to ensure stability of the control at the terminal state. In the case of the trajectory-tracking problem, a separate terminal cost in the criteria formulation is omitted (included in the control error cost instead), as the output of system directly corresponds to the state variables.

$$\begin{gathered} J\left(N_K, \overline{\boldsymbol{x}}_{K_0}, \overline{\boldsymbol{u}}_{K_{N,0}}\right) = \Delta\overline{\boldsymbol{X}}_{K_N}{}^T \boldsymbol{Q}_K \Delta\overline{\boldsymbol{X}}_N + \Delta\boldsymbol{u}_{K_N}{}^T \boldsymbol{R}_K \Delta\boldsymbol{u}_{K_N} \\ \Delta\boldsymbol{u}_{K_N} = \overline{\boldsymbol{u}}_{K_N} - \overline{\boldsymbol{u}}_{K_{N,0}} \\ \boldsymbol{u}_{K_{min,N}} < \boldsymbol{u}_{K_N} < \boldsymbol{u}_{K_{max,N}} \end{gathered} \tag{5.5}$$

where $\Delta\overline{\boldsymbol{X}}_{K_N}$ is the state deviation of future state variables $\overline{\boldsymbol{X}}_{K_N}$ from desired state variable $\overline{\boldsymbol{X}}_{N_{N,r}}$, i.e. $\Delta\overline{\boldsymbol{X}}_{K_N} = \overline{\boldsymbol{X}}_{K_N} - \overline{\boldsymbol{X}}_{N_{N,r}}$. Since the aim is to bring the state variables to zero (origin), the deviation state variables become, $\Delta\overline{\boldsymbol{X}}_N = \overline{\boldsymbol{X}}_N$. The weighting matrix $\boldsymbol{Q}_K$ is positive semi definite ($\boldsymbol{Q}_K \succcurlyeq \boldsymbol{0}$) and matrix $\boldsymbol{R}_K$ is positive definite ($\boldsymbol{R}_K \succ \boldsymbol{0}$).

$$\left.\begin{aligned} \boldsymbol{Q}_K &= diag\left(\boldsymbol{Q}_{K_i}\right) \\ \boldsymbol{R}_K &= diag\left(\boldsymbol{R}_{K_i}\right) \end{aligned}\right\} \forall\, i = 1\ to\ N_K$$

The criteria consist of a cost for control effort and a cost for state variable deviation. The aim of the trajectory tracking controller is to generate optimal control actions which brings the state variables to zero over a finite time horizon N_K. The output of the system directly corresponds to the state variables. The last diagonal element in the matrix $\boldsymbol{Q}_K$, i.e. $\boldsymbol{Q}_{K_N}$ can be seen as terminal state cost and can be tuned to achieve terminal state stability.

Rewriting the criteria, eq (5.5) in terms of free and force responses by substituting eq (5.3) and eq (5.4),

$$
\begin{aligned}
J\left(N_K, \bar{\boldsymbol{x}}_{K_0}, \bar{\boldsymbol{u}}_{K_{N,0}}\right) &= \left[\bar{\boldsymbol{X}}_{K_{fr,N}} + \bar{\boldsymbol{X}}_{K_{fo,N}}\right]^T \boldsymbol{Q}_K \left[\bar{\boldsymbol{X}}_{K_{fr,N}} + \bar{\boldsymbol{X}}_{K_{fo,N}}\right] + \Delta \boldsymbol{u}_{K_N}{}^T \boldsymbol{R}_K \Delta \boldsymbol{u}_{K_N} \\
&= \left[\bar{\boldsymbol{X}}_{K_{fr,N}} + \boldsymbol{S}_{K_{xu,k}} \Delta \boldsymbol{u}_{K_N}\right]^T \boldsymbol{Q}_K \left[\bar{\boldsymbol{X}}_{K_{fr,N}} + \boldsymbol{S}_{K_{xu,k}} \Delta \boldsymbol{u}_{K_N}\right] + \Delta \boldsymbol{u}_{K_N}{}^T \boldsymbol{R}_K \Delta \boldsymbol{u}_{K_N} \\
&= \bar{\boldsymbol{X}}_{K_{fr,N}}{}^T \boldsymbol{Q}_K \bar{\boldsymbol{X}}_{K_{fr,N}} + \bar{\boldsymbol{X}}_{K_{fr,N}}{}^T \boldsymbol{Q}_K \Delta \boldsymbol{u}_{K_N}{}^T \boldsymbol{S}_{K_{xu,k}}{}^T + \Delta \boldsymbol{u}_{K_N}{}^T \boldsymbol{S}_{K_{xu,k}}{}^T \boldsymbol{Q}_K \bar{\boldsymbol{X}}_{fr,N} + \\
&\quad + \Delta \boldsymbol{u}_{K_N}{}^T \boldsymbol{S}_{K_{xu,k}}{}^T \boldsymbol{Q}_K \boldsymbol{S}_{K_{xu,k}} \Delta \boldsymbol{u}_{K_N} + \Delta \boldsymbol{u}_{K_N}{}^T \boldsymbol{R}_K \Delta \boldsymbol{u}_{K_N} \\
&= \underbrace{\bar{\boldsymbol{X}}_{K_{fr,N}}{}^T \boldsymbol{Q}_K \bar{\boldsymbol{X}}_{K_{fr,N}}}_{\boldsymbol{c}} + \Delta \boldsymbol{u}_{K_N}{}^T \underbrace{\boldsymbol{S}_{K_{xu,k}}{}^T \boldsymbol{Q}_K \bar{\boldsymbol{X}}_{K_{fr,N}}}_{\boldsymbol{m}} + \underbrace{\bar{\boldsymbol{X}}_{fr,N}{}^T \boldsymbol{Q}_K \boldsymbol{S}_{K_{xu,k}}}_{\boldsymbol{m}^T} \Delta \boldsymbol{u}_{K_N} + \\
&\quad + \Delta \boldsymbol{u}_{K_N}{}^T \underbrace{\left(\boldsymbol{S}_{K_{xu,k}}{}^T \boldsymbol{Q}_K \boldsymbol{S}_{K_{xu,k}} + \boldsymbol{R}_K\right)}_{\boldsymbol{M}} \Delta \boldsymbol{u}_{K_N} \\
&= \Delta \boldsymbol{u}_{K_N}{}^T \boldsymbol{M} \Delta \boldsymbol{u}_{K_N} + \Delta \boldsymbol{u}_{K_N}{}^T \boldsymbol{m} + \boldsymbol{m}^T \Delta \boldsymbol{u}_{K_N} + \boldsymbol{c}
\end{aligned}
$$

where,

$$
\begin{aligned}
\boldsymbol{m} &= \boldsymbol{S}_{K_{xu,k}}{}^T \boldsymbol{Q}_K \left(\boldsymbol{S}_{K_{xx,k}} \bar{\boldsymbol{x}}_{K_k} + \boldsymbol{S}_{K_{xu,k}} \bar{\boldsymbol{u}}_{K_{N,0}}\right) \\
\boldsymbol{M} &= \boldsymbol{S}_{K_{xu,k}}{}^T \boldsymbol{Q}_K \boldsymbol{S}_{K_{xu,k}} + \boldsymbol{R}_K
\end{aligned}
$$

In case of the unconstraint control, the analytical solution is as follows,

$$\Delta \boldsymbol{u}_K = -\boldsymbol{M}^{-1} \boldsymbol{m}$$

In case of the constraint control, the optimal control action is the solution of the quadratic programing problem, obtained by minimizing the following criteria.

$$\min_{\Delta \boldsymbol{u}_K} J = \Delta \boldsymbol{u}_K{}^T M \Delta \boldsymbol{u}_K + \boldsymbol{2m}^T \Delta \boldsymbol{u}_K \text{ such that } \boldsymbol{A_o} \Delta \boldsymbol{u}_K \leq \boldsymbol{b_0} \tag{5.6}$$

5.1.3 Constrains on manipulated variable

Considering the control input constraints of the successive linear model for a finite horizon N_K,

$$
\begin{aligned}
\boldsymbol{u}_{K_{min}} &\leq \boldsymbol{u}_{K_N} \leq \boldsymbol{u}_{K_{max}} \\
\boldsymbol{u}_{K_{min}} &\leq \Delta \boldsymbol{u}_{K_N} + \bar{\boldsymbol{u}}_{K_{N,0}} + \boldsymbol{u}_{K_r} \leq \boldsymbol{u}_{K_{max}} \\
\boldsymbol{u}_{K_{min}} - \bar{\boldsymbol{u}}_{K_{N,0}} - \boldsymbol{u}_{K_{N,r}} &\leq \Delta \boldsymbol{u}_{K_N} \leq \boldsymbol{u}_{K_{max}} - \bar{\boldsymbol{u}}_{K_{N,0}} - \boldsymbol{u}_{K_{N,r}}
\end{aligned} \tag{5.7}
$$

where $\bar{\boldsymbol{u}}_0$ is the last control action, $\bar{\boldsymbol{u}}_{K_0} = \bar{\boldsymbol{u}}_K(k-1)$ and $\boldsymbol{u}_{K_{N,r}}$ is a vector of reference inputs for the horizon. Deriving the inequality constraints for a horizon N_K,

$$
\underbrace{\begin{bmatrix} \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{I} & \cdots & \boldsymbol{I} \\ -\boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ -\boldsymbol{I} & \cdots & -\boldsymbol{I} \end{bmatrix}}_{A_o} \Delta \boldsymbol{u}_K \leq \underbrace{\begin{bmatrix} \boldsymbol{u}_{K_{max}} - \bar{\boldsymbol{u}}_{K_0} - \boldsymbol{u}_{K_{k,r}} \\ \vdots \\ \boldsymbol{u}_{K_{max}} - \bar{\boldsymbol{u}}_{K_0} - \boldsymbol{u}_{K_{k+N,r}} \\ -\boldsymbol{u}_{K_{min}} + \bar{\boldsymbol{u}}_{K_0} + \boldsymbol{u}_{K_{k,r}} \\ \vdots \\ -\boldsymbol{u}_{K_{min}} + \bar{\boldsymbol{u}}_{K_0} + \boldsymbol{u}_{K_{k+N,r}} \end{bmatrix}}_{b_0}
$$

and for the error-based model,

$$\begin{gathered} \boldsymbol{u}_{K_{min}} \leq \boldsymbol{u}_{K_N} \leq \boldsymbol{u}_{K_{max}} \\ \boldsymbol{u}_{K_{min}} \leq \boldsymbol{u}_{K_{ff,N}} - (\Delta \boldsymbol{u}_{K_N} + \bar{\boldsymbol{u}}_{K_{N,0}}) \leq \boldsymbol{u}_{K_{max}} \\ \boldsymbol{u}_{K_{min}} + \bar{\boldsymbol{u}}_{K_{N,0}} - \boldsymbol{u}_{K_{ff,N}} \leq -\Delta \boldsymbol{u}_{K_N} \leq \boldsymbol{u}_{K_{max}} + \bar{\boldsymbol{u}}_{K_{N,0}} - \boldsymbol{u}_{K_{ff,N}} \end{gathered} \tag{5.8}$$

$$\underbrace{\begin{bmatrix} -\boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ -\boldsymbol{I} & \cdots & -\boldsymbol{I} \\ \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{I} & \cdots & \boldsymbol{I} \end{bmatrix}}_{A_o} \Delta \boldsymbol{u}_K \leq \underbrace{\begin{bmatrix} \boldsymbol{u}_{K_{max}} + \bar{\boldsymbol{u}}_{K_0} - \boldsymbol{u}_{K_{ff,k}} \\ \vdots \\ \boldsymbol{u}_{K_{max}} + \bar{\boldsymbol{u}}_{K_0} - \boldsymbol{u}_{K_{ff,k+N}} \\ -\boldsymbol{u}_{K_{min}} - \bar{\boldsymbol{u}}_{K_0} + \boldsymbol{u}_{K_{ff,k}} \\ \vdots \\ -\boldsymbol{u}_{K_{min}} - \bar{\boldsymbol{u}}_{K_0} + \boldsymbol{u}_{K_{ff,k+N}} \end{bmatrix}}_{b_0}$$

Figure 13 shows the control scheme of NMPC based on the successive linear ($NMPC_1$) on LTV model. The control scheme of the error-based models ($NMPC_2$) is illustrated in figure 14.

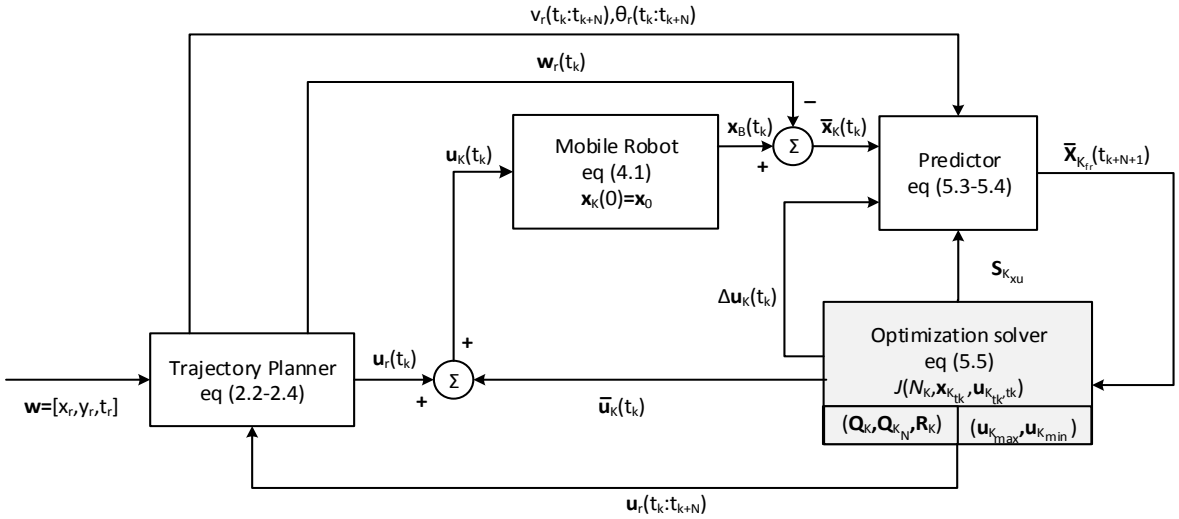

Figure 13 Control scheme of trajectory tracking $NMPC_1$ with LTV model

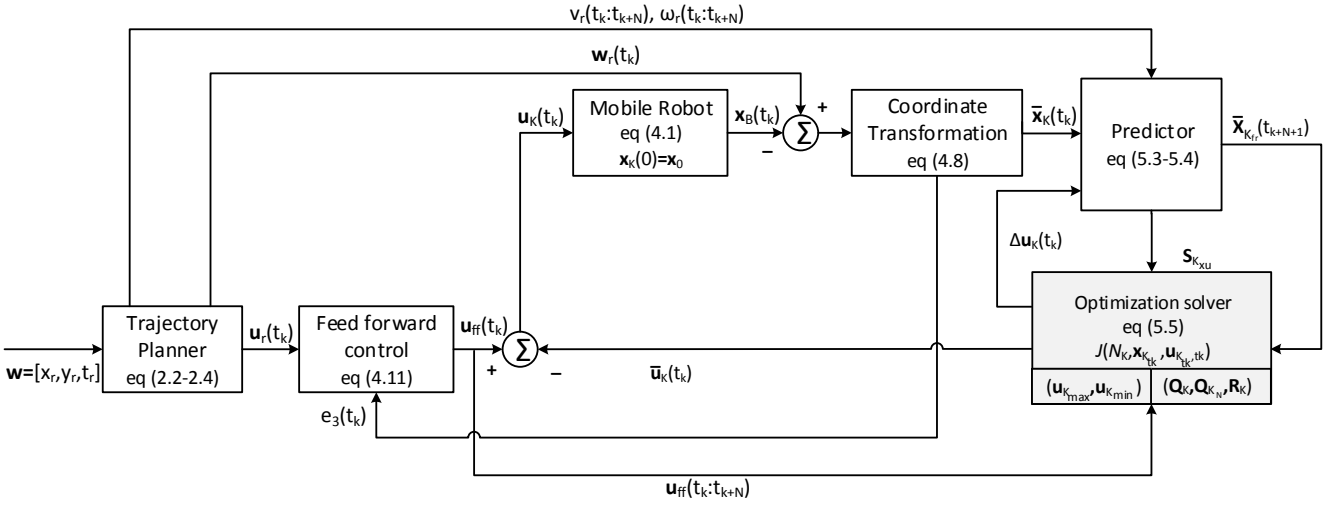

Figure 14 Control scheme of trajectory tracking $NMPC_2$ with LTV model

5.2 Trajectory tracking by state tracking control

5.2.1 Linear state tracking control design (Kanayama controller)

The proposed predictive controller is compared to the state-of-the-art state tracking controllers whose design can be found in (Samson & Ait-Abderrahim 1991; De Luca et al. 2001; Kanayama et al. 1990). The state tracking controller can be designed with a linear feed-back gain as,

$$\bar{\boldsymbol{u}}_{K_k} = -\boldsymbol{K}_s(k)\boldsymbol{e}(k) \tag{5.9}$$

where $\boldsymbol{K}_s$ is the time varying feedback gain matrix in the form of,

$$\boldsymbol{K}_s(k) = \begin{bmatrix} -k_1(k) & 0 & 0 \\ 0 & -sign(v_r)k_2(k) & -k_3(k) \end{bmatrix} \in \mathbb{R}^{n_u \times n_x} \tag{5.10}$$

The controller gains k_1, k_2 and k_3 are determined by comparing (pole placement method) the closed loop characteristic polynomial with a desired closed loop characteristic polynomial in the form of,

$$(\lambda + 2\zeta a)(\lambda^2 + 2\zeta a\lambda + a^2)$$

which has constant eigenvalues (one negative real at $-2\zeta a$ and a complex pair with natural frequency $a > 0$ and damping co-efficient $\zeta > 0$). The controller gains can be then be chosen as,

$$k_1(k) = k_3(k) = 2\zeta a$$

$$k_2(k) = \frac{a^2 - \omega_r(k)^2}{v_r(k)}$$

The gain k_2 will go to infinity as $v_r(k) \to \infty$. In order to avoid this, gain scheduling can be designed by letting $a = a(k) = \sqrt{\omega_r(k)^2 + bv_r(k)^2}$, substituting,

$$k_1(k) = k_3(k) = 2\zeta\sqrt{\omega_r(k)^2 + bv_r(k)^2}\ ;\ k_2(k) = b|v_r(k)|$$

where the factor $b > 0$ can be seen as an additional degree of freedom.

Even the controller gains are chosen in such a way that the closed loop poles are at the left half of the s-plane, while the controller is still non-linear and time varying. Therefore, asymptotic stability of tracking error is not guaranteed. The control scheme is the same as that in figure 12.

5.2.2 Nonlinear state tracking control design (Samson controller)

Considering the nonlinear feedback control law (Samson & Ait-Abderrahim 1991) as,

$$\boldsymbol{K}_s(k) = \begin{bmatrix} -k_1(k) & 0 & 0 \\ 0 & -\bar{k}_2 v_r(k) \frac{\sin(e_3)}{e_3} & -k_3(k) \end{bmatrix} \in \mathbb{R}^{n_u \times n_x} \tag{5.11}$$

The controller gains $k_1, \bar{k}_2$ and k_3 are determined by the same method as in the linear control design.

$$k_1(k) = k_3(k) = 2\zeta\sqrt{\omega_r(k)^2 + b v_r(k)^2} \;\; ; \;\; \bar{k}_2 = b$$

The main difference between the linear and nonlinear state tracking controller is that, global asymptotic stability can be proved in the case of nonlinear controller by the Lyapunov analysis. See (Samson & Ait-Abderrahim 1991) for the proof.

5.3 Simulation results

The inputs are time parameterized reference points, which are interpolated to generate smooth trajectory points by linear interpolation or spline interpolation by Matlab function `interp1`. The linear interpolation generates sharp turns, which results in very high reference velocities (which may not be practically unrealizable). The trajectory planner generates the reference parameters – orientation, tangential and angular velocities. Simulation experiments, with a continuous time model eq (4.1) for a real robot, were performed. Total simulation time was 30s with a sampling time of 100ms. Four different controllers were tested – NMPC with successive linear model ($NMPC_1$), NMPC with error tracking model ($NMPC_2$), Kanayama feedback controller (KC) and Samson feedback controller (SC).

Trajectory tracking $NMPC_1$ of the mobile robot was simulated: by using the model in the form of eq (4.7), predicting the future states with LTV model eq (5.3-5.4), optimizing the cost function in the form of eq (5.5), and defining the constraints in the form of eq (5.7). The optimized control actions for horizon N_K were calculated and the first control action was applied to the system. Only NMPC with LTV model was considered, as the results obtained with non-linear model, eq (4.1), were the same. $NMPC_2$ uses error based model, eq (4.12), prediction model, eq (5.3-5.4) and constraint definition, eq (5.8). The control actions for state feedback tracking controllers KC and SC were calculated by eq (5.10) and eq (5.11) respectively. The input constraints were considered as,

$$-1\, m/s \leq v_B \leq 1\, m/s \quad ; \quad -1\, rad/s \leq \omega_B \leq 1\, rad/s \tag{5.12}$$

Twelve different simulation experiments (**S1-S12**) have been performed with different controllers, initial conditions, interpolation methods, tuning parameters and constraint condition. A different initial condition refers to the pose of the robot, which is different from the reference pose. Table II shows the simulation results of trajectory tracking with different controllers. The simulation results with parameters listed in the table are shown in figure 15-18. Not all the figures of the simulation experiments are shown, as some of the results with different controllers are not significantly different. The results are comparable with the sum of squared error (SSE):

$$SSE_{xy} = |x_B - x_r|^2 + |y_B - y_r|^2 \; ; \qquad SSE_{\theta} = |\theta_B - \theta_r|^2$$

Three sets of experiments were conducted – linearly interpolated trajectories, constraint control and different initial conditions. In all the cases, $NMPC_1$ showed more SSEs when compared to other controllers. The SSE of control responses of linearly interpolated trajectories were almost same, even though $NMPC_1$ is outperformed by all the other controllers. The controllers were able to generate target velocities with respect to the reference velocities. In case of constraint control, state tracking controllers were able to track the robot closer to the reference trajectory. Constraints in the form of eq (5.12) were considered. When the initial conditions were different, $NMPC_2$ performed better than all the other controllers.

The trajectory tracking abilities of $NMCP_1$, KC and SC were comparable in all the experiments without much difference in SSE's. It can be noted that, with the same tuning parameters, the $NMPC_2$ trajectory-tracking controller was able to track the robot closer to the reference trajectory. This is the main advantage of $NMCP_2$ over other controllers, wherein case of other controllers, the tuning parameters have to be tuned separately for different scenarios. This shows that NMPC with error-based model is more suitable for trajectory tracking irrespective of the reference trajectory scenario.

Table II Simulation experiments – Interpolation method, controller, tuning parameters and SSE

Simulation Experiment	**Controller**[1]	**Tuning parameters**				**Constraints** $u_{K_{max}} = -u_{K_{min}}$	**Interpolation Method**	**Initial Condition**	**Control Quality** [SSE_{xy} ,SSE_{θ}]
		N	**R**	**Q**	**Q_N**				
S1	$NMPC_1$	5	10^{-1}***I**	**I**	**0**	--	Linear	--	[0.0059 2.6030]
S2					**I**	[1 1]	Spline		[0.2668 2.8050]
S3			10^{-2}***I**		10^{3}***I**	--		[0.1 -0.1 0]	[0.5322 89.375]
S4	$NMPC_2$	5	10^{-1}***I**	**I**	**I**	--	Linear	--	[0.0192 0.0004]
S5						[1 1]	Spline		[0.0343 2.3770]
S6						--		[0.1 -0.1 0]	[0.1736 57.371]
		b		ζ					
S7	KC	100		0.7		--	Linear	--	[0.0014 0.0067]
S8		50				[1 1]	Spline		[0.0101 1.4151]
S9						--		[0.1 -0.1 0]	[0.2837 67.770]
S10	SC	100		0.7		--	Linear	--	[0.5470 0.0918]
S11		50				[1 1]	Spline		[0.0108 1.4166]
S12						--		[0.1 -0.1 0]	[0.3187 74.651]

[1] $NMPC_1$ – with successive linear model, $NMPC_2$– with error based model, KC – Kanayama controller, SC – Samson controller

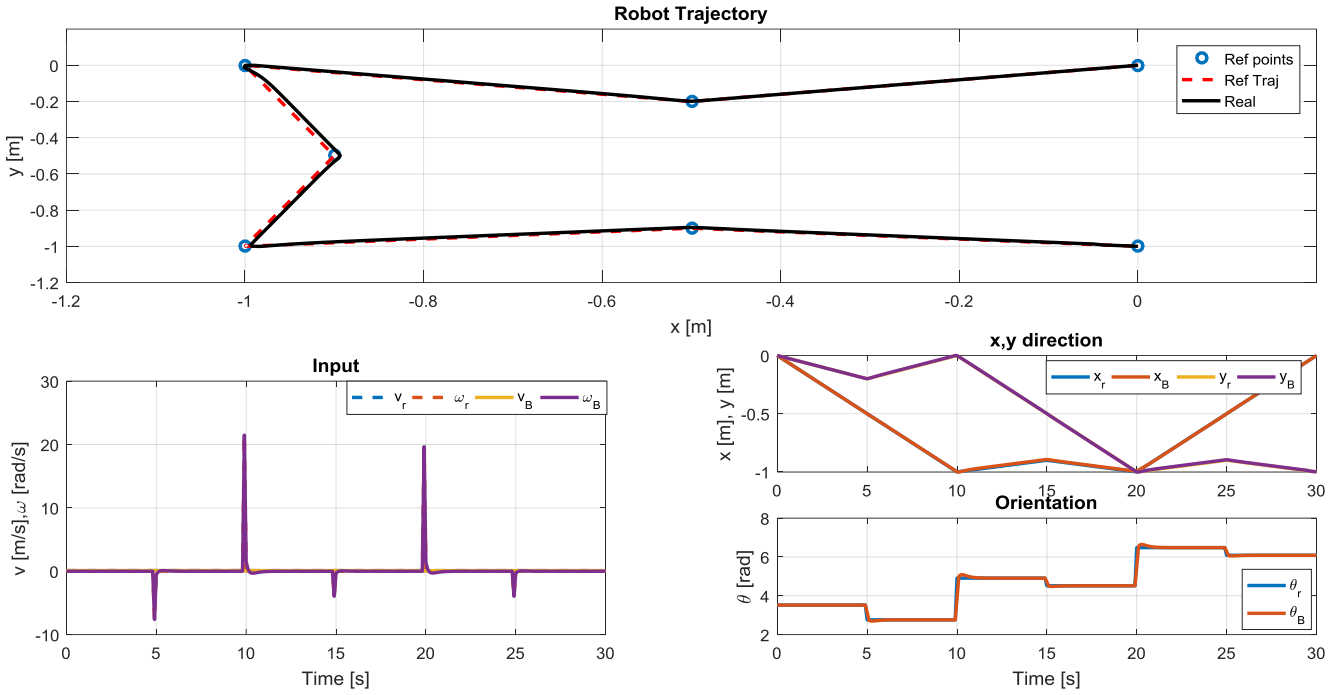

Figure 15 Simulation experiment S1: Linear interpolation, unconstraint NMPC with successive linear model– reference trajectory, reference inputs, tracked trajectory, control actions

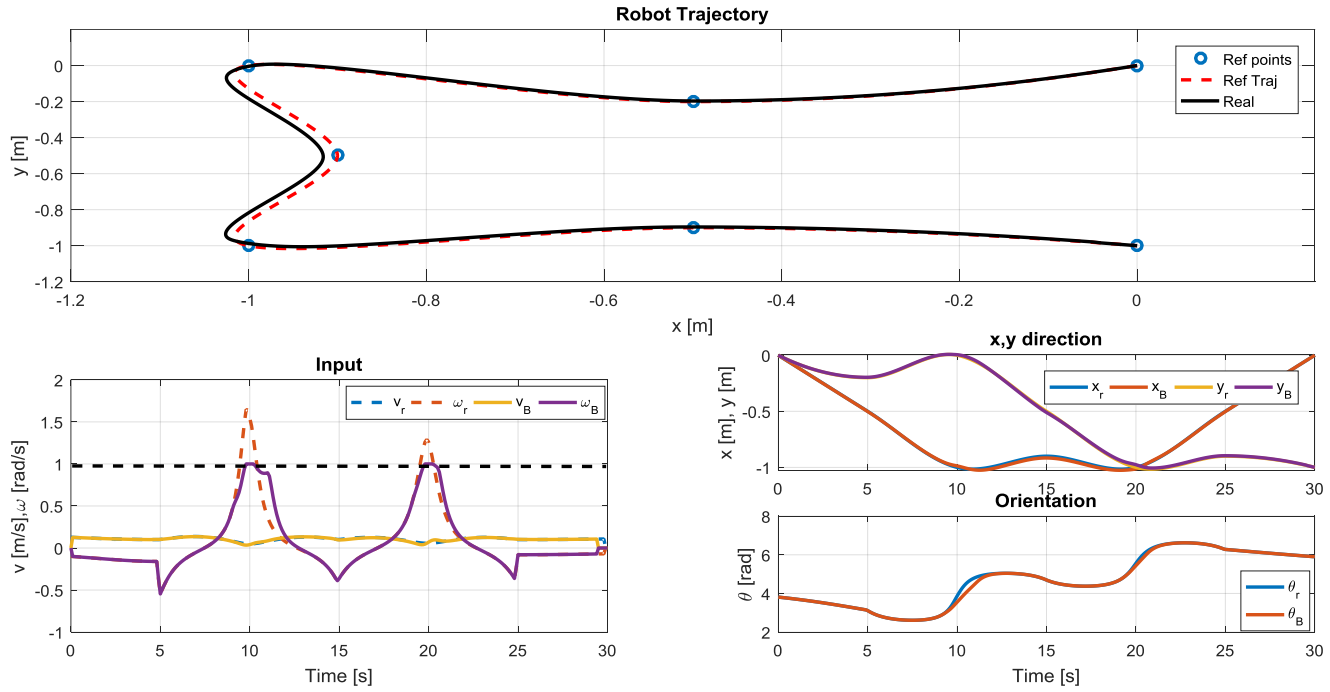

Figure 16 Simulation experiment S5: Spline interpolation, constraint NMPC with error tracking model – reference trajectory, reference inputs, tracked trajectory, control actions

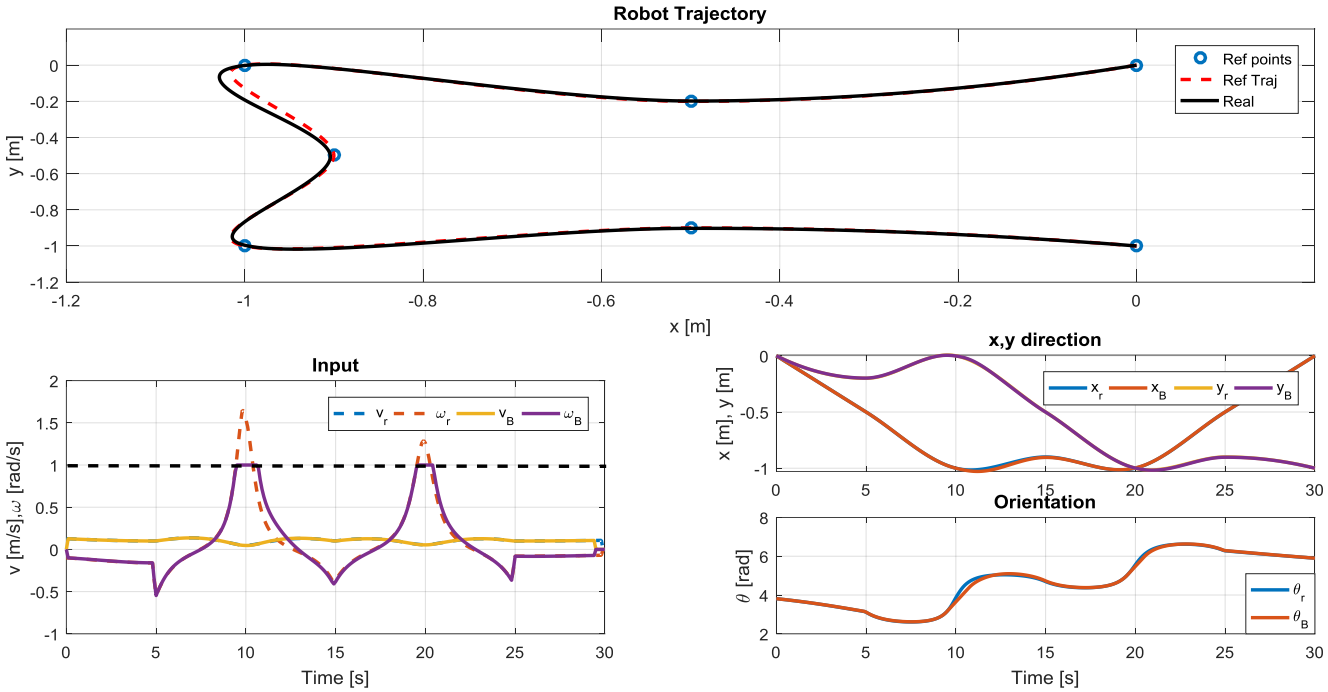

Figure 17 Simulation experiment S8: Spline interpolation, constraint Kanayama controller – reference trajectory, reference inputs, tracked trajectory, control actions

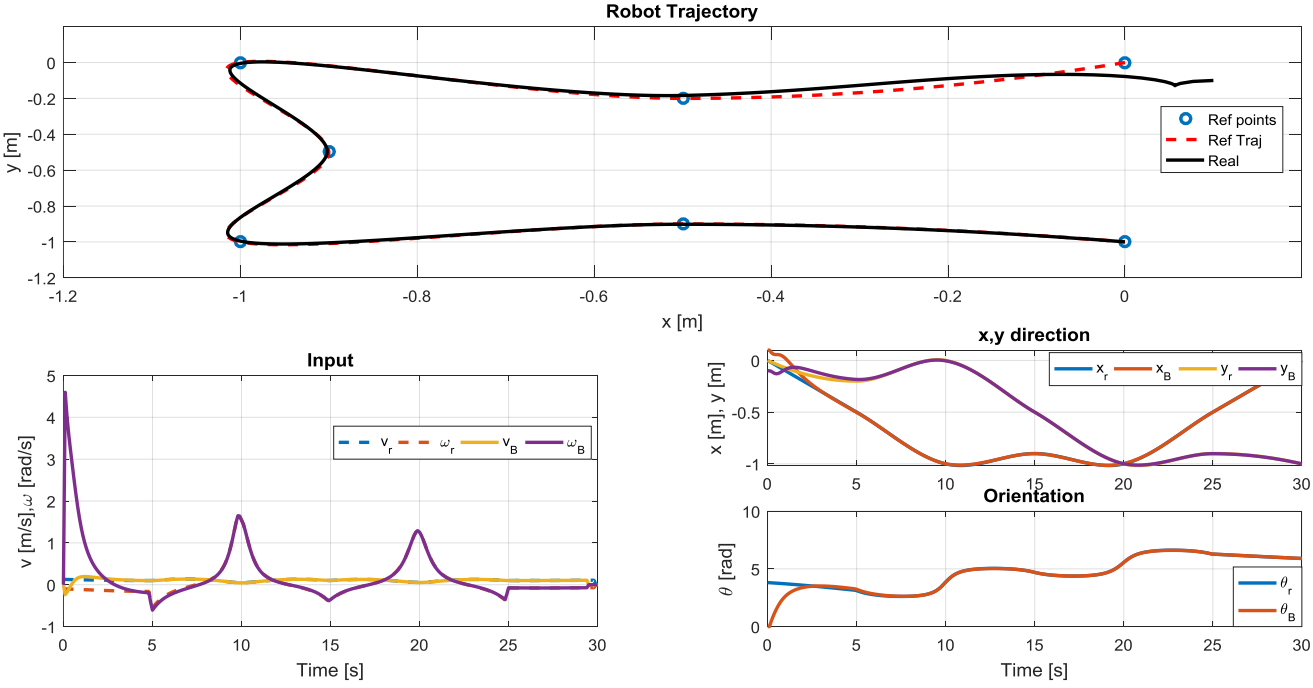

Figure 18 Simulation experiment S12: Spline interpolation, unconstraint Samson controller with different initial condition – reference trajectory, reference inputs, tracked trajectory, control actions

Figure 19 shows the initial tracking of the WMR in the case where the robot's initial pose is different from the reference trajectory – since in this example, the robot is oriented in the opposite direction. $NMPC_2$ converges faster to the reference trajectory compared to all other controllers, followed by state tracking controllers. It is also interesting to note that the $NMPC_1$ initially drives in the opposite direction to the reference orientation and eventually converges with high initial tracking errors.

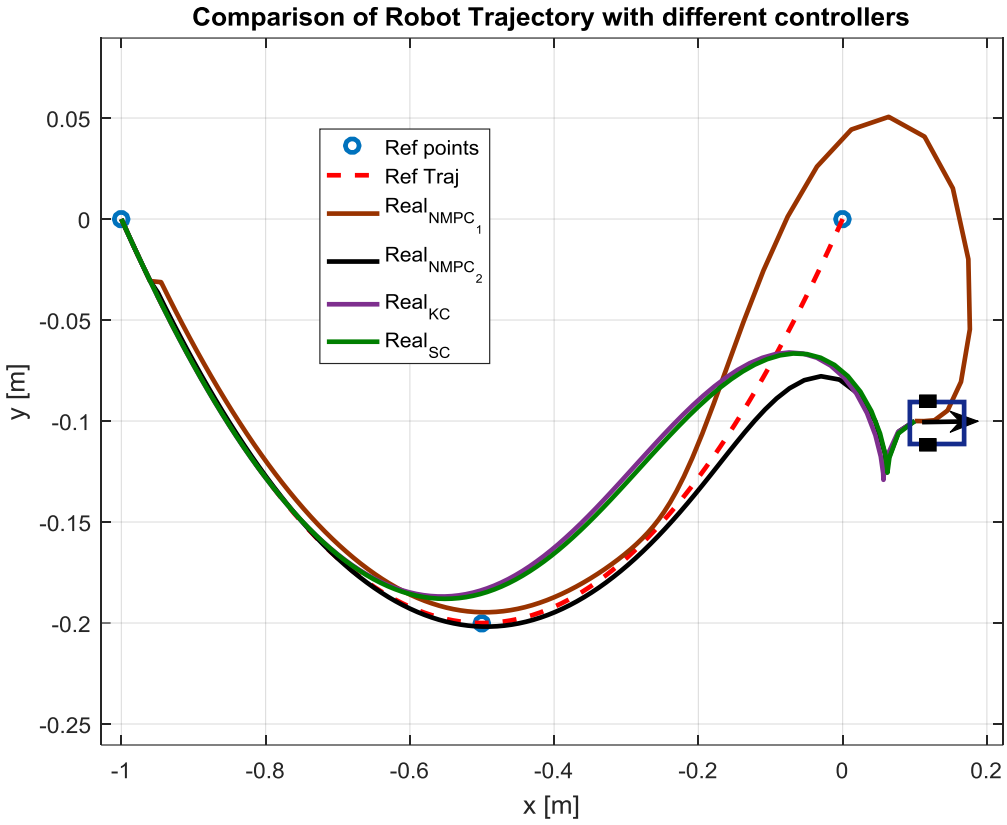

Figure 19 Comparison of kinematic trajectory tracking controllers with different initial conditions

The significance of simulation experiments with linear interpolation is questionable. The linear interpolation generates very high velocities (which is practically unrealizable with most of the robots) especially in sharp corners. The tracking of linearly interpolated trajectory by controllers shows the ability of controllers to work in different operating regions. In other words, simulations with linear interpolation are employed to check whether, the robot is able to track the trajectory at very high and low step changes.

6. DYNAMICS MODELLING OF DIFFERENTIAL DRIVE WMR

Dynamics is the study of forces and torques and their effect on motion, as opposed to kinematics, which is the study of motion without considering the causes of motion. Dynamics modelling is essential in the case of WMR, if an accurate control of robot is required. There are mainly two modelling methods in literature - Lagrangian approach and Newton-Euler approach. In the Lagrangian approach, a multibody robot is treated as a single system and the forces are expressed in terms of energies – kinetic and potential energies. In the Newton-Euler approach, each body/link is considered separately, a free body diagram is drawn, and balance of forces and torques acting on each element is considered. The Lagrangian approach is best suited for study of dynamic properties and analysis of a control scheme, whereas Newton-Euler is suitable for implementation of the control scheme. The Newton-Euler based modelling approach is considered in the following text.

6.1 Mathematical modelling of dynamics of WMR

The differential drive mobile robot is assumed to have two wheels connected with permanent magnet DC motors, powered from a common voltage source and are independently controlled. The motors are connected to the driving wheels through a gearbox with the same gear ratio. An ideal gearbox (nonlinearities are neglected) is considered, which reduces the linear speed and boosts the torque. The chassis is firmly supported by a castor wheel with no influence on chassis motion (resistance force on motion is neglected).

The mathematical model of the robot, consists of three relatively independent parts: the dynamics of the permanent magnet DC motor, chassis dynamics (dependency between translational and rotational velocities of the chassis reference point on moments acting to driving wheels), and kinematics (influence of motor speed to translational and rotational velocities). A chassis reference point is the point in the robot at which kinematics of the robot is considered; usually it is placed at the center of the axis joining the wheels. In the following formulation, this chassis reference point can be placed anywhere in the axis joining the wheels, depending on the center of gravity of the robot.

6.1.1 DC motor dynamics

An equivalent circuit of an ideal permanent magnet DC motor is shown in figure 20. It consists of resistance R, inductance L and magnetic field M. The commutator is not considered. Each motor is independently controlled by its own supply voltages U_L, U_R taken from a common

voltage source U_0 through control signals u_L, u_R. The rotor generated back EMF, which is in reverse polarity and is proportional to rotor angular velocity. The torque of motor is proportional to the current i.

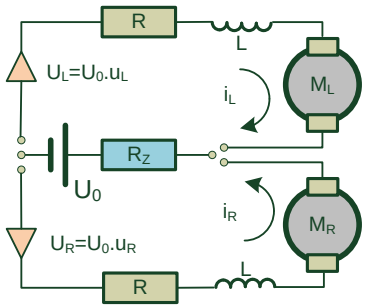

Figure 20 DC Motor Wiring

The Dynamics of the permanent magnet DC motor can be derived from balancing of voltages (Kirchhoff's law) and balancing of moments. From Kirchhoff's voltage law, we can derive,

$$Ri_L + R_z(i_L + i_R) + L\frac{di_L}{dt} = u_L U_0 - K\omega_L \tag{6.1}$$

$$Ri_R + R_z(i_L + i_R) + L\frac{di_R}{dt} = u_R U_0 - K\omega_R \tag{6.2}$$

where,

R	motor winding resistance [Ω]
R_z	source resistance [Ω]
L	motor winding inductance $[H]$
K	back EMF constant $[kg.m^2.s^{-2}.A^{-1}]$
U_o	source voltage $[V]$
u_R, u_L	control voltages of right and left wheels $[-]$
i	current $[A]$
ω_R, ω_L	right and left motor angular velocities $[rad.s^{-1}]$

By considering the balance of moments – moment of inertia M_s, rotational resistance proportional to rotational speed (mechanical losses) M_o and load torque M_x caused by magnetic field which is proportional to current.

$$M_s + M_o + M_x = M_M$$

Writing in terms of right and left wheel drives,

$$J\frac{d\omega_L}{dt} + k_r\omega_L + M_L = Ki_L \tag{6.3}$$

$$J\frac{d\omega_R}{dt} + k_r\omega_R + M_R = Ki_R \tag{6.4}$$

where,

J	moment of inertia of the robot $[kg.m^2]$
k_r	coefficient of rotational resistance $[kg.m^2.s^{-1}]$
M_L, M_R	load torques on left and right wheels $[kg.m^2.s^{-2}]$

6.1.2 Chassis dynamics

Chassis dynamics is defined with a vector of tangential velocity v_B acting on a chassis reference point and angular velocity ω_B (constant at all chassis points). The chassis reference point B is the point of the intersection of the axis joining the wheels and center of gravity normal projection – see figure 21. Point T is the general center of gravity – usually it is placed at the center of the axis joining the wheels.

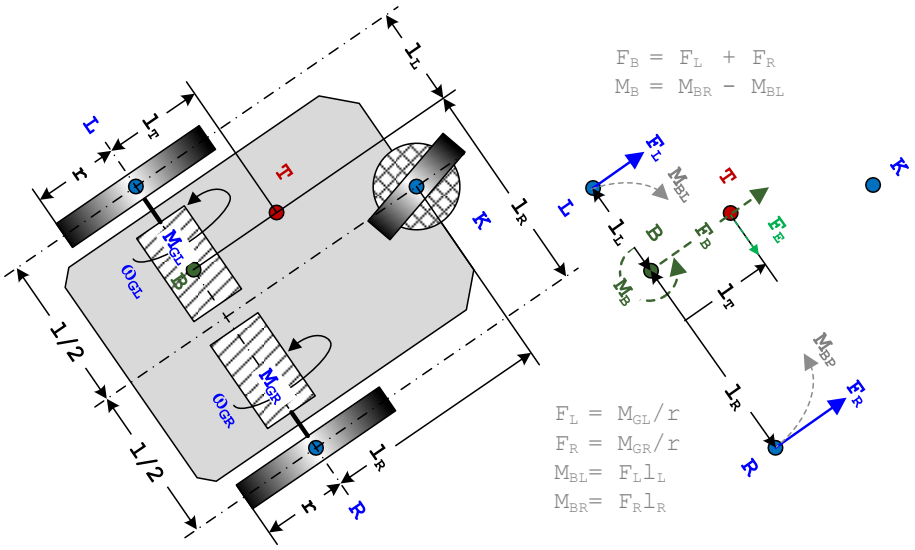

Figure 21 Chassis Scheme and Forces

The two forces acting on the wheels, F_L and F_R, can be replaced with a single force F_B and torsion torque M_B acing at reference point B. The chassis parameters are wheel radius r, total mass m and moment of inertia J_T with respect to the center of gravity, located at a distance l_T, l_R, l_L as shown in figure 21.

By considering the balance of forces (forces on drive wheels) F_L, F_R, inertial force F_s and resistive force F_o which is proportional to tangential velocity of robot. The balance of force influencing linear motion is,

$$F_L + F_R + F_s + F_o = 0$$

$$\frac{M_{GL}}{r} + \frac{M_{GR}}{r} - k_v v_B - m\frac{dv_B}{dt} = 0 \tag{6.5}$$

where,

m	mass of robot $[kg]$
k_v	resistive coefficient against linear motion $[kg.s^{-1}]$
M_{GL}, M_{GR}	torque on left and right drive $[kg.m^2.s^{-2}]$
v_B	tangential velocity $[m.s^{-1}]$
r	radius of wheels $[m]$

By considering balance of torques, with torque generated by drives M_{BL}, M_{BR}, chassis momentum M_T, torque due to Euler force M_E and torque due to resistive force M_o,

$$M_{BL} + M_{BR} + M_o + M_T + M_E = 0$$

$$-\frac{M_{GL}}{r} i_L + \frac{M_{GR}}{r} i_R - k_\omega \omega_B - J_T \frac{d\omega_B}{dt} - {l_T}^2 m \frac{d\omega_B}{dt} = 0 \tag{6.6}$$

where,

l_R, l_L	distance to right and left wheel from point B $[m]$
l_T	distance to center of gravity from point B $[m]$
k_ω	resistive coefficient against rotational motion $[kg.m^2.s^{-1}]$
J_T	moment of inertia with respect to rotational axis at center of gravity $[kg.m^2]$
ω_B	angular velocity $[s^{-1}]$

By applying the parallel axis (Huygens–Steiner) theorem, the moment of inertia J_B with respect to the rotational axis at reference point B can be derived as,

$$J_B = J_T + m{l_T}^2$$

6.1.3 Relationship between rotational speed of the motor and chassis movement

The equations governing behavior of motors (currents and motor speeds) and the behavior of chassis (translational and rotational movement) are only connected through torques of motors. Let the motors be connected to the chassis through an ideal gear box with gear ratio p_G. The nonlinearities (saturation, backlash, friction, dead zone) of the gear box are neglected.

The gear box reduces the angular velocities of motors (ω_L, ω_R) to output angular velocities $(\omega_{GL}, \omega_{GR})$ with respect to the gear ratio. Similarly, the torques of motors (M_L, M_R) are increased to output torques (M_{GL}, M_{GR}).

$$\omega_{GL} = \frac{\omega_L}{p_G} \qquad \omega_{GR} = \frac{\omega_R}{p_G}$$

$$M_{GL} = M_L p_G \qquad M_{GR} = M_R p_G$$

Let both the wheels have same radius r, and their peripheral speeds v_L, v_R depend on output angular velocities of the gear box according to the following relation,

$$v_L = r\omega_{GL} = r\frac{\omega_L}{p_G}$$

$$v_R = r\omega_{GR} = r\frac{\omega_R}{p_G}$$

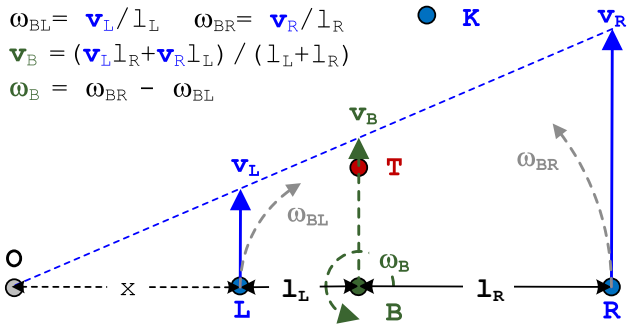

Figure 22 Linear and Angular Velocity Recalculation

The tangential velocity v_B and angular velocity ω_B at chassis reference point B can be recalculated from the peripheral velocities v_L, v_R. Assuming both the motors have same axis of rotation and the peripheral velocities will hence always be parallel. From the theorems of similar triangles, depicted in figure 22, the tangential velocity and angular velocity at point B are given by,

$$v_B = \frac{v_L l_R + v_R l_L}{l_L + l_R} = \frac{r}{p_G(l_L + l_R)}(l_L\omega_R + l_R\omega_L) \tag{6.7}$$

$$\omega_B = \frac{v_B}{x + l_L} = \frac{v_R - v_L}{l_L + l_R} = \frac{r}{p_G(l_L + l_R)}(-\omega_L + \omega_R) \tag{6.8}$$

From the tangential and angular velocities, the position in Cartesian coordinates (x_B, y_B) and orientation of the robot θ_B with respect to x-axis can be calculated from the basic kinematic nonlinear equation.

$$\begin{aligned} \dot{x}_B &= v_B \cos\theta_B \\ \dot{y}_B &= v_B \sin\theta_B \\ \dot{\theta}_B &= \omega_B \end{aligned} \tag{6.9}$$

6.2 Combined state space model

The dynamic parts consist of four differential equations, eq (6.1-6.4) describing the behavior of motors, two differential equations, eq (6.5-6.6) describing the chassis dynamics and two algebraic equations, eq (6.7-6.8) which relates tangential and angular velocities of chassis to the peripheral speed of motors. From this, eight state variables describing the current state of

left and right motors (currents, motor speeds and load torques) and chassis parameters (tangential and angular velocities) can be derived.

The differential equations eq (6.5-6.6) and algebraic equations eq (6.7-6.8) are rewritten by introducing the "reduced" wheel radius r_G and total moment of inertia J_B as,

$$r_G = \frac{r}{p_G} \qquad J_B = J_T + m{l_T}^2$$

$$\frac{p_G}{r} M_L + \frac{p_G}{r} M_R - k_v v_B - m\frac{dv_B}{dt} = 0$$

$$M_L + M_R - r_G k_v v_B - r_G m\frac{dv_B}{dt} = 0 \tag{6.10}$$

$$-i_L \frac{p_G}{r} M_L + i_R \frac{p_G}{r} M_R - k_\omega \omega_B - (J_T + m{l_T}^2)\frac{d\omega_B}{dt} = 0$$

$$-i_L M_L + i_R M_R - r_G k_\omega \omega_B - r_G J_B \frac{d\omega_B}{dt} = 0 \tag{6.11}$$

Rewriting the algebraic equations,

$$v_B = \frac{r_G}{(l_L + l_R)}(l_R \omega_L + l_L \omega_R) \tag{6.12}$$

$$\omega_B = \frac{r_G}{(l_L + l_R)}(-\omega_L + \omega_R) \tag{6.13}$$

These six differential equations, eq (6.1-6.4,6.10-6.11), and two algebraic eq (6.12-6.13) containing eight state variables represent a mathematical description of the dynamic behavior of ideally differentially steered mobile robots, with losses linearly dependent on the revolutions or speed. The control signals, u_L and u_R, that control the supply voltages of the motors are the input variables.

The calculation of steady-state values for constant engine power voltages is given below. A calculation of steady state is useful both for the checking of derived equations and for the experimental determination of the values of the unknown parameters. The steady state in matrix representation is,

$$\begin{bmatrix} R+R_z & R_z & K & 0 & 0 & 0 & 0 & 0 \\ R_z & R+R_z & 0 & K & 0 & 0 & 0 & 0 \\ K & 0 & -k_r & 0 & -1 & 0 & 0 & 0 \\ 0 & K & 0 & -k_r & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -r_G k_v & 0 \\ 0 & 0 & 0 & 0 & -l_L & l_R & 0 & -r_G k_\omega \\ 0 & 0 & l_R & l_L & 0 & 0 & -\frac{l_R + l_L}{r_G} & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & -\frac{l_R + l_L}{r_G} \end{bmatrix} \begin{bmatrix} i_L \\ i_R \\ \omega_L \\ \omega_R \\ M_L \\ M_R \\ v_B \\ \omega_B \end{bmatrix} = \begin{bmatrix} U_0 u_L \\ U_0 u_R \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6.14}$$

The eq (6.1-6.4,6.10-6.11) can be reduced to a state-space model with four state variables by exclusion of the dependent state variables $(M_L, M_R, \omega_L, \omega_R)$. The six differential equations can be reduced to four by substituting eq (6.12-6.13) into eq (6.10-6.11), by eliminating the load torques M_L and M_R, and by substituting eq (6.1-6.4) to eq (6.10-6.11) to arrive at four differential equations eq (6.15-6.18). Introducing the following parameters, for simplification,

$$\left.\begin{aligned} a_L &= k_r + \frac{k_v l_R {r_G}^2}{l_L + l_R} & a_R &= k_r + \frac{k_v l_L {r_G}^2}{l_L + l_R} \\ b_L &= J + \frac{m l_R {r_G}^2}{l_L + l_R} & b_R &= J + \frac{m l_L {r_G}^2}{l_L + l_R} \\ c_L &= k_r l_L + \frac{k_\omega {r_G}^2}{l_L + l_R} & c_R &= k_r l_R + \frac{k_\omega {r_G}^2}{l_L + l_R} \\ d_L &= J l_L + \frac{J_B {r_G}^2}{l_L + l_R} & d_R &= J l_R + \frac{J_B {r_G}^2}{l_L + l_R} \end{aligned}\right\}$$

The reduced linear equations of the model consist of the following equations,

$$\frac{di_L}{dt} = \frac{u_L U_0 - K\omega_L - (R+R_z) i_L - R_z i_R}{L} \tag{6.15}$$

$$\frac{di_R}{dt} = \frac{u_R U_0 - K\omega_R - (R+R_z) i_R - R_z i_L}{L} \tag{6.16}$$

$$\frac{d\omega_L}{dt} = \frac{1}{b_L d_R + b_R d_L}\left(d_R[K(i_L + i_R) - a_L\omega_L - a_R\omega_R] - b_R[K(-l_L i_L + l_R i_R) + c_L\omega_L - c_R\omega_R]\right) \tag{6.17}$$

$$\frac{d\omega_R}{dt} = \frac{1}{b_L d_R + b_R d_L}\left(d_L[K(i_L + i_R) - a_L\omega_L - a_R\omega_R] + b_L[K(-l_L i_L + l_R i_R) + c_L\omega_L - c_R\omega_R]\right) \tag{6.18}$$

and the output equations are given by the algebraic equations eq (6.12-6.13). The dynamics of differential drive WMR can be represented in the general state space model as,

$$\begin{aligned} \dot{\boldsymbol{x}}_D &= \widetilde{\boldsymbol{A}}_D \boldsymbol{x}_D + \widetilde{\boldsymbol{B}}_D \boldsymbol{u}_D \\ \boldsymbol{y}_D &= \widetilde{\boldsymbol{C}}_D \boldsymbol{x}_D \end{aligned} \tag{6.19}$$

$$\boldsymbol{x}_D = \begin{bmatrix} i_L \\ i_R \\ \omega_L \\ \omega_R \end{bmatrix} \quad \boldsymbol{u}_D = \begin{bmatrix} u_L \\ u_R \end{bmatrix} \quad \boldsymbol{y}_D = \begin{bmatrix} v_B \\ \omega_B \end{bmatrix}$$

with system matrices as,

$$\widetilde{\boldsymbol{A}}_D = \begin{bmatrix} -\frac{R+R_z}{L} & -\frac{R_z}{L} & -\frac{K}{L} & 0 \\ -\frac{R_z}{L} & -\frac{R+R_z}{L} & 0 & -\frac{K}{L} \\ \frac{K(d_R + b_R l_L)}{b_L d_R + b_R d_L} & \frac{K(d_R - b_R l_R)}{b_L d_R + b_R d_L} & -\frac{d_R a_L + b_R c_L}{b_L d_R + b_R d_L} & -\frac{d_R a_R - b_R c_R}{b_L d_R + b_R d_L} \\ \frac{K(d_L - b_L l_L)}{b_L d_R + b_R d_L} & \frac{K(d_R + b_L l_R)}{b_L d_R + b_R d_L} & -\frac{d_L a_L - b_L c_L}{b_L d_R + b_R d_L} & -\frac{d_L a_R + b_L c_R}{b_L d_R + b_R d_L} \end{bmatrix}$$

$$\widetilde{\boldsymbol{B}}_D = \begin{bmatrix} \frac{U_0}{L} & 0 \\ 0 & \frac{U_0}{L} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad \widetilde{\boldsymbol{C}}_D = \begin{bmatrix} 0 & 0 & \frac{l_R r_G}{l_L + l_R} & \frac{l_L r_G}{l_L + l_R} \\ 0 & 0 & -\frac{r_G}{l_L + l_R} & \frac{r_G}{l_L + l_R} \end{bmatrix}$$

6.3 Parameter identification and estimation

Typical chassis parameters are given in Table III and typical motor parameters in Table IV. The motor parameters can be found in the data sheet or can be identified by experiments, see (Cong et al. 2010; Saab & Kaed-Bey n.d.; Völlmecke 2013). The chassis geometric parameters can be directly measured from the robot. The other chassis parameters (moment of inertias and coefficient of resistances) have to be identified either experimentally or approximated with typical values. These values are chosen so that they roughly correspond to the values of a real robot. Assuming the robot is symmetrically shaped, the center of gravity is located at the line of symmetry and hence the chassis reference point B, is at the center of axis joining the wheels.

Table III Chassis Parameters

Notation	Value	Unit	Description
l_L	0.080	m	distance of the left wheel from point B
l_P	0.080	m	distance of the right wheel from point B
l_T	0.050	m	distance of center of gravity to the axis joining the wheels
l_K	0.100	m	distance of caster wheel to the axis joining the wheels
r	0.035	m	radius of driving wheel
M	1.250	kg	total weight of the robot
k_v	0.100	$kg.s^{-1}$	coefficient of the resistance against robot linear motion
J_T	0.550	$kg.m^2$	moment of inertia of robot with respect to center of gravity
k_ω	1.350	$kg.m^2.s^{-1}$	coefficient of the resistance against robot's rotation

Table IV DC Motors Parameters

Notation	**Value**	**Unit**	**Description**
R	2.000	Ω	motor winding resistance
L	0.015	H	motor inductance
K	0.150	$kg.m^2.s^{-2}.A^{-1}$	back EMF constant
R_Z	0.012	Ω	source resistance
U_0	8.000	V	source voltage
J	0.015	$kg.m^2$	total moment of inertia of rotor and gearbox
k_r	0.002	$kg.m^2.s^{-1}$	coefficient of the resistance against rotation of rotor and gearbox
p_G	25	--	gearbox transmission ratio

6.4 Dynamics model verification by open loop simulation

The linear state-space model, eq (6.19), is discretized with a sampling time of, $T_S = 0.01s$. The dynamics model output (tangential and angular velocities) are inputs into the nonlinear kinematic part eq (6.9) to derive the position and orientation of the robot – see figure 23. The initial location of mobile robot is assumed at the origin with an orientation in the direction of the x axis.

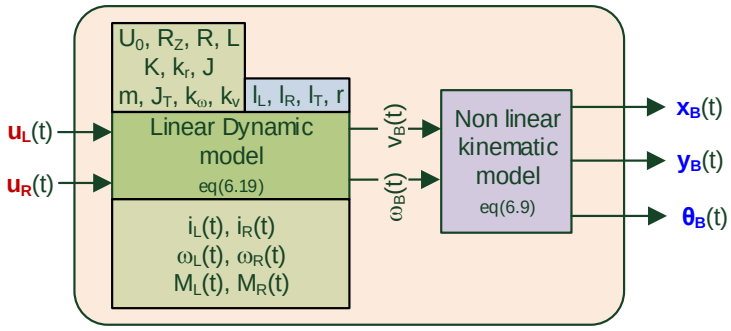

Figure 23 Scheme of dynamics model for open loop verification

Figures 24-26 show the response of the three simulation experiments.

i. *Only translational motion* (figure 24) – both the motors are powered with the same control voltage (8V), as a result no steering ($\omega_B = 0$) is occurred, robot moves in a straight line (practically not possible because of friction, slipping, motor nonlinearities etc.)

ii. *Only rotational motion* (figure 25) – only right motor is powered (8V) as a result the robots rotates in place.

iii. *With both translational and rotational motion* (figure 26) – the motors are driven with a rectangular wave control voltage with amplitude of 8V.

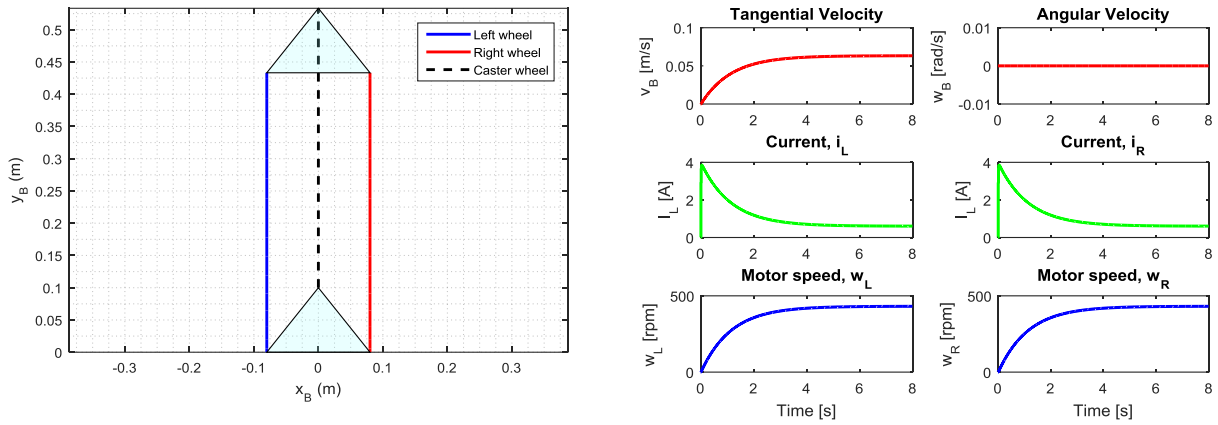

Figure 24 Dynamics model verification – robot with only translational motion

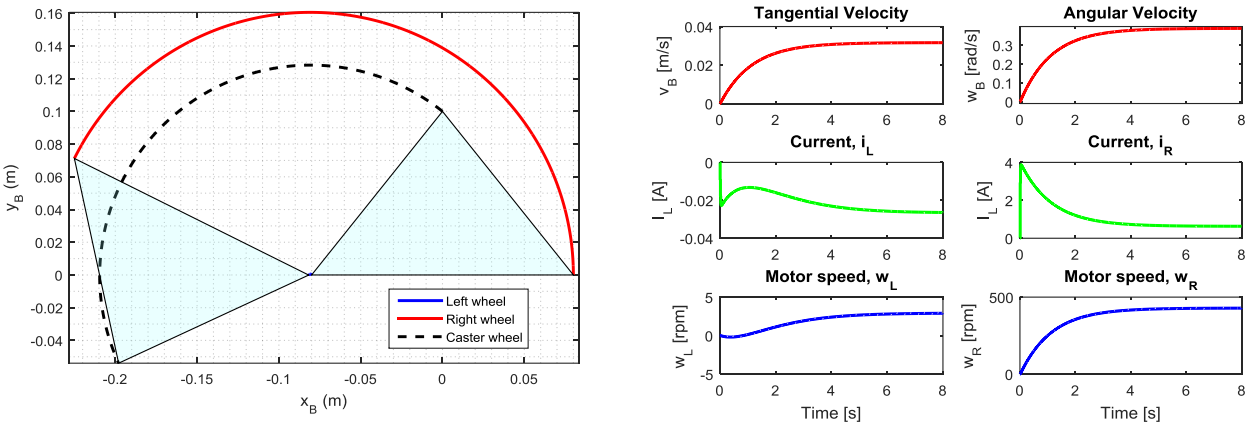

Figure 25 Dynamics model verification – robot with only rotational motion

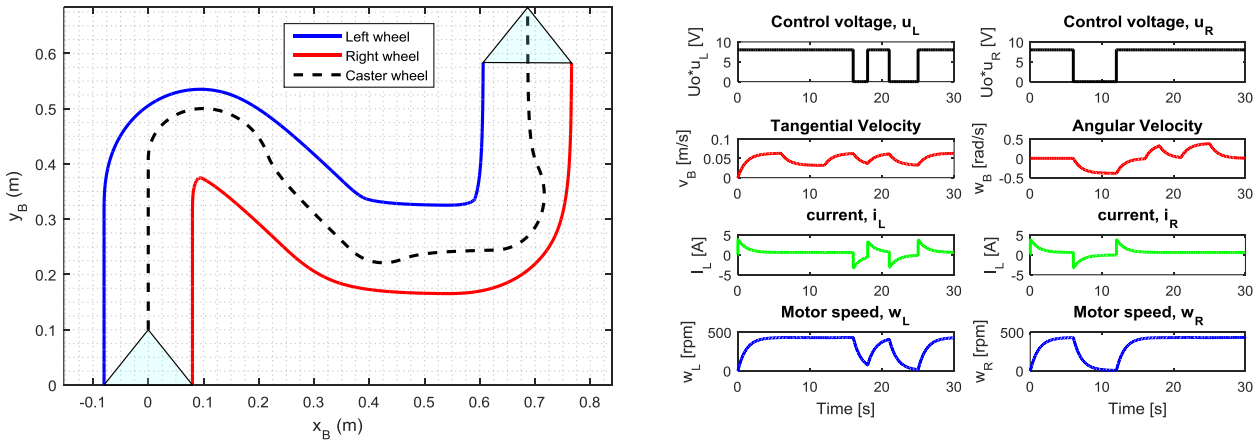

Figure 26 Dynamics model verification – robot with both translational and rotational motion

An illustrative example of dynamic behavior when both the motors are driven by signals with same amplitude is given in figure 27. The left motor is driven by signals with 10s time period and 50% duty ratio, and the right motor with same duty ratio but with a time period of 20s. The saturation of different state variables which are the constraints of the real system can be inferred from the figure. It is not possible to excite above that level. e.g., it is important to consider these saturation levels for generating feasible trajectories for trajectory tracking control of robot.

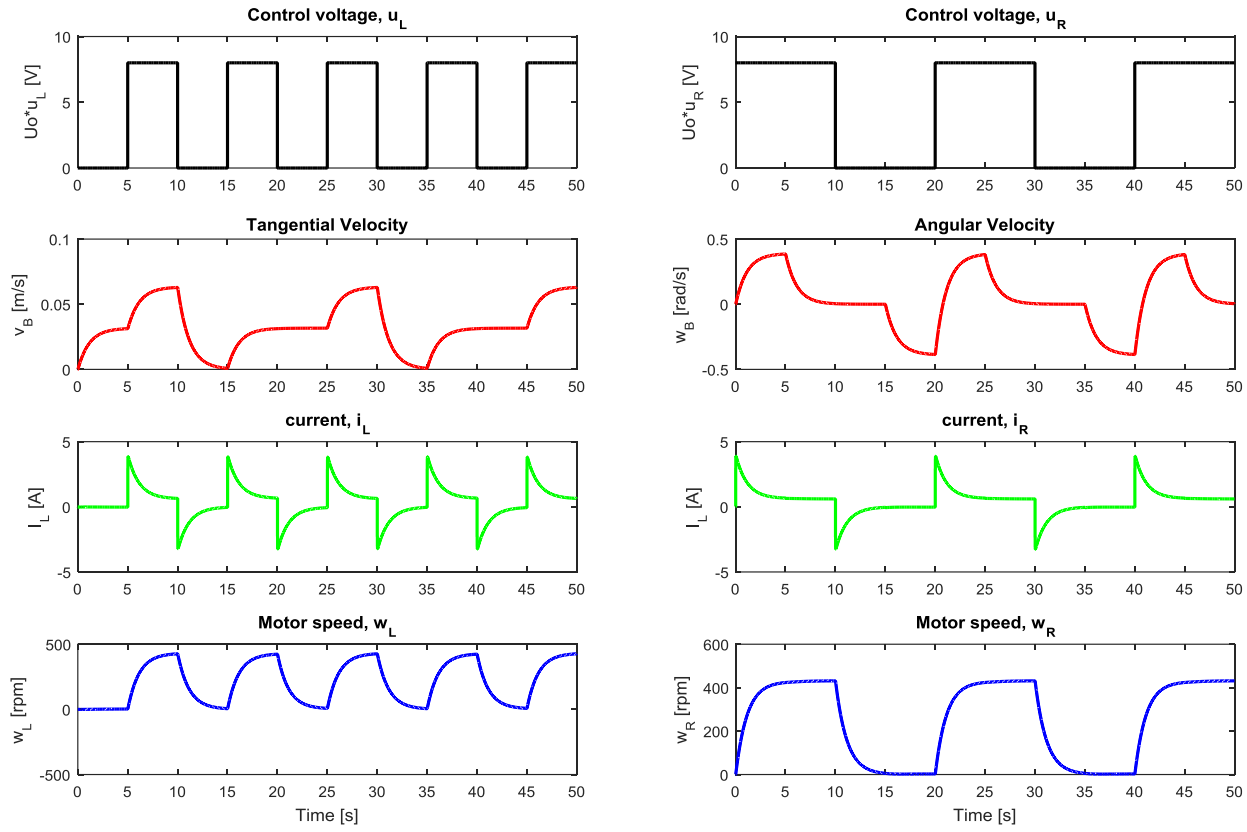

Figure 27 Dynamic behavior of robot – currents and motor speeds

7. VELOCITY TRACKING OF WMR – DYNAMICS CONTROLLER

In chapter 6, we discussed the mathematical modelling of the dynamics of a WMR considering the motor dynamics and chassis dynamics. A linear state space model of mobile robot dynamics has been derived with motor currents and motor speeds as state variables, motor control variables as manipulated variables, and linear and tangential velocities as outputs.

This chapter presents the mobile robot dynamics (velocity tracking) controller. The aim of dynamics controller is to generate control voltages (u_L, u_R) to the motors for controlling the tangential and angular velocities (v_B, ω_B). A general block diagram of the state space controller is shown in figure 28. The set points are the reference tangential and angular velocities $(v_{B,r}, \omega_{B,r})$. Three controllers are studied: discrete PID, Linear MPC and static feedforward control.

The dynamics of WMR is a TITO system with four state variables (see section 6.2). Discretizing eq (6.19) with a sampling time of T_s, the discrete linear state space model of dynamics of mobile robot is,

$$\begin{aligned} \boldsymbol{x}_D(k+1) &= \boldsymbol{A}_D\boldsymbol{x}_D(k) + \boldsymbol{B}_D\boldsymbol{u}_D(k) \\ \boldsymbol{y}_D(k) &= \boldsymbol{C}_D\boldsymbol{x}_D(k) \end{aligned} \tag{7.1}$$

$$\boldsymbol{x}_D(k) = \begin{bmatrix} i_L(k) \\ i_R(k) \\ \omega_L(k) \\ \omega_R(k) \end{bmatrix} \quad \boldsymbol{u}_D(k) = \begin{bmatrix} u_L(k) \\ u_R(k) \end{bmatrix} \quad \boldsymbol{y}_D(k) = \begin{bmatrix} v_B(k) \\ \omega_B(k) \end{bmatrix}$$

where $\boldsymbol{A}_D, \boldsymbol{B}_D, \boldsymbol{C}_D$ are discretized version of matrices in eq (6.19). The only measurable variable is the motor speeds (ω_L, ω_R). The state variables and output have to be estimated with state estimation.

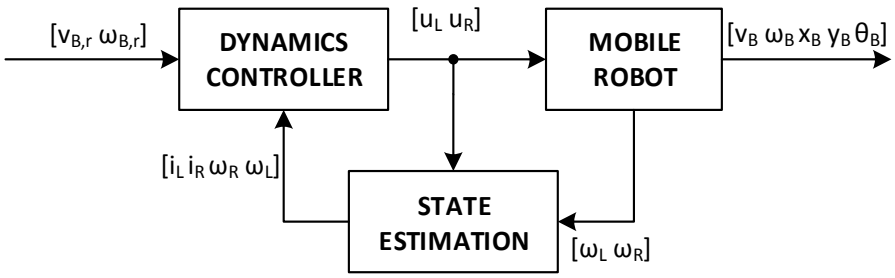

Figure 28 General block diagram of dynamics velocity controller

7.1 Discrete-time PID control

The dynamics model is a multivariable system, where direct controlling of the velocities is not easy. The multivariable system can be decoupled in terms of motor speeds and can be controlled easily by PID control or by a steady state gain. Tangential velocity and angular velocity equations, eq (6.12-6.13) are rewritten to derive motor speeds as,

$$\begin{bmatrix} v_{B,r} \\ \omega_{B,r} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{l_L r_G}{(l_L + l_R)} & \frac{l_R r_G}{(l_L + l_R)} \\ -\frac{r_G}{(l_L + l_R)} & \frac{r_G}{(l_L + l_R)} \end{bmatrix}}_{\dot{\boldsymbol{z}}} \begin{bmatrix} \omega_{L,r} \\ \omega_{R,r} \end{bmatrix}$$

$$\begin{bmatrix} \omega_{L,r} \\ \omega_{R,r} \end{bmatrix} = \boldsymbol{Z}^{-1} \begin{bmatrix} v_{B,r} \\ \omega_{B,r} \end{bmatrix} \tag{7.2}$$

Two PID controllers for controlling the reference motor speeds achieve the target velocity tracking. The control inputs are motor control voltages as can be seen in figure 29.

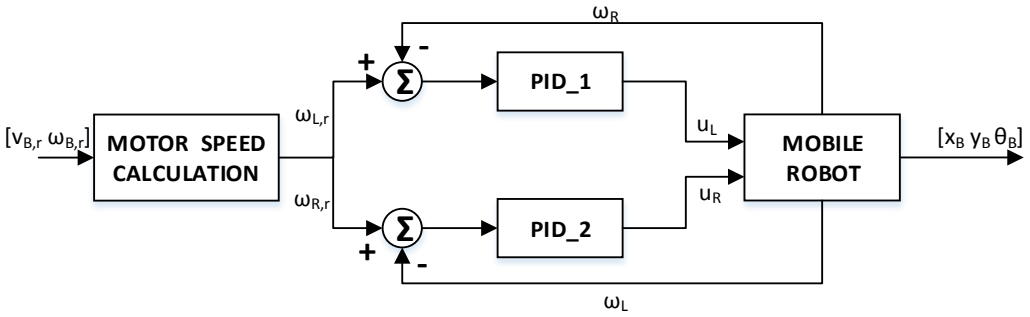

Figure 29 Block diagram of PID wheel speed controller

The standard parallel form of a continuous time PID (Proportional-Integral-Derivative) controller is given by,

$$u(t) = K_P e(t) + K_I \int_0^\tau e(\tau) d\tau + K_D \dot{e}(t)$$

where u is the control input and e is the control error, $e(t) = w(t) - y(t)$

There are several forms of discrete PID controller in literature, depending on the discretizing method. The discrete PID difference equation by Simson's method of numerical integration with a sampling time of T_s, is,

$$u(k) = u(k-1) + q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) \tag{7.3}$$

where,

$$q_0 = \left(K_P + \frac{K_I}{2} T_s + \frac{K_D}{T_s} \right)$$

$$q_1 = \left(-K_P + \frac{K_I}{2}T_s + \frac{K_D}{T_s}\right)$$

$$q_2 = \frac{K_D}{T_s}$$

7.2 Mobile robot dynamics Linear MPC (LMPC)

The formulation of LMPC is the same as that discussed in section 3.1. Considering the discrete time dynamics model as in eq (7.1), the prediction model is,

$$\boldsymbol{x}_{D\,k+N+1} = \boldsymbol{S}_{D\,xx}\boldsymbol{x}_{D\,k} + \boldsymbol{S}_{D\,xu}\boldsymbol{u}_{D\,N}$$
$$\boldsymbol{y}_{D\,N} = \boldsymbol{S}_{D\,yx}\boldsymbol{x}_{D\,k} + \boldsymbol{S}_{D\,yu}\boldsymbol{u}_{D\,N}$$

where,

$$\boldsymbol{u}_{D\,N} = \begin{bmatrix} u_L(k) \\ u_R(k) \\ \vdots \\ u_L(k+N) \\ u_R(k+N) \end{bmatrix};\ \boldsymbol{y}_{D\,N} = \begin{bmatrix} v_B(k) \\ \omega_B(k) \\ \vdots \\ v_B(k+N) \\ \omega_B(k+N) \end{bmatrix}$$

Decomposing into free and force responses, the free response is given by,

$$\boldsymbol{x}_{D\,fr}(N+1) = \boldsymbol{S}_{D\,xx}\boldsymbol{x}_{D\,0} + \boldsymbol{S}_{D\,xu}\boldsymbol{u}_{D\,N,0}$$
$$\boldsymbol{y}_{D\,fr,N} = \boldsymbol{S}_{D\,yx}\boldsymbol{x}_{D\,0} + \boldsymbol{S}_{D\,yu}\boldsymbol{u}_{D\,N,0} \tag{7.4}$$

and the forced response is,

$$\boldsymbol{x}_{D\,fo}(N+1) = \boldsymbol{S}_{D\,xu}\Delta\boldsymbol{u}_{D\,N}$$
$$\boldsymbol{y}_{D\,fo,N} = \boldsymbol{S}_{D\,yu}\Delta\boldsymbol{u}_{D\,N}$$

where, $\boldsymbol{u}_{D\,N} = \begin{bmatrix} \boldsymbol{u}_{D\,k} \\ \vdots \\ \boldsymbol{u}_{D\,k+N} \end{bmatrix};\ \boldsymbol{u}_{D\,N,0} = \begin{bmatrix} \boldsymbol{u}_{D\,fr,k} \\ \vdots \\ \boldsymbol{u}_{D\,fr,k+N} \end{bmatrix};\ \Delta\boldsymbol{u}_{D\,N} = \boldsymbol{u}_{D\,N} - \boldsymbol{u}_{D\,N,0}$

$$\boldsymbol{S}_{D\,xx} = \boldsymbol{A}_D{}^{N+1} \qquad \in \mathbb{R}^{n_x \times n_x}$$

$$\boldsymbol{S}_{D\,xu} = \begin{bmatrix} \boldsymbol{A}_D{}^{N}\boldsymbol{B}_D & \boldsymbol{A}_D{}^{N-1}\boldsymbol{B}_D & \cdots & \boldsymbol{B}_D \end{bmatrix} \qquad \in \mathbb{R}^{n_x \times (N+1)*n_u}$$

$$\boldsymbol{S}_{D\,yx} = \begin{bmatrix} \boldsymbol{C}_D \\ \boldsymbol{C}_D\boldsymbol{A}_D \\ \boldsymbol{C}_D\boldsymbol{A}_D{}^{2} \\ \vdots \\ \boldsymbol{C}_D\boldsymbol{A}_D{}^{N} \end{bmatrix} \qquad \in \mathbb{R}^{(N+1)*n_y \times n_x}$$

$$\boldsymbol{S}_{D\,yu} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{C}_D\boldsymbol{B}_D & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{C}_D\boldsymbol{A}_D\boldsymbol{B}_D & \boldsymbol{C}_D\boldsymbol{B}_D & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{C}_D\boldsymbol{A}_D{}^{2}\boldsymbol{B}_D & \boldsymbol{C}_D\boldsymbol{A}_D\boldsymbol{B}_D & \boldsymbol{C}_D\boldsymbol{B}_D & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{C}_D\boldsymbol{A}_D{}^{N-1}\boldsymbol{B}_D & \boldsymbol{C}_D\boldsymbol{A}_D{}^{N-2}\boldsymbol{B}_D & \boldsymbol{C}_D\boldsymbol{A}_D{}^{N-3}\boldsymbol{B}_D & \cdots & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{(N+1)*n_x \times (N+1)*n_u}$$

Cost function

$$J\left(N, \boldsymbol{x}_{D_0}, \boldsymbol{u}_{D_{N,0}}\right) = \Delta \boldsymbol{x}_D{}^T(N)\boldsymbol{Q}_{D_N}\Delta \boldsymbol{x}_D(N) + \boldsymbol{e}_N{}^T\boldsymbol{Q}_D\boldsymbol{e}_N + \Delta \boldsymbol{u}_{D_N}{}^T\boldsymbol{R}_D\Delta \boldsymbol{u}_{D_N}$$
$$\Delta \boldsymbol{x}_D(N) = \boldsymbol{x}_{D_w}(k+N+1) - \boldsymbol{x}_D(k+N+1)$$
$$\boldsymbol{e}_N = \boldsymbol{w}_{D_N} - \boldsymbol{y}_{D_N}$$
$$\Delta \boldsymbol{u}_{D_N} = \boldsymbol{u}_{D_N} - \boldsymbol{u}_{D_{N,0}}$$
$$\boldsymbol{u}_{D_{min,N}} < \boldsymbol{u}_{D_N} < \boldsymbol{u}_{D_{max,N}}$$

The cost function is same as that discussed in section 3.3 with penalization on terminal state error, control error and control effort; with weighting matrices $\boldsymbol{Q}_{D_N}, \boldsymbol{Q}_D, \boldsymbol{R}_D$ respectively. The desired terminal state $\boldsymbol{x}_{D_w}$ is derived from steady state output $\boldsymbol{w}_D$ as,

$$\boldsymbol{x}_{D_w}(k+N+1) = [(\boldsymbol{I} - \boldsymbol{A}_D)^{-1}\boldsymbol{B}_D][\boldsymbol{C}_D(\boldsymbol{I} - \boldsymbol{A}_D)^{-1}\boldsymbol{B}_D]^{-1}\boldsymbol{w}_D(k+N+1)$$

The optimal control action is the solution of a quadratic programing problem by minimizing the criteria.

$$\min_{\Delta \boldsymbol{u}} J = \Delta \boldsymbol{u}_D{}^T M \Delta \boldsymbol{u}_D + \boldsymbol{2m}^T \Delta \boldsymbol{u}_D \text{ Such that } \boldsymbol{A_o}\Delta \boldsymbol{u}_D \leq \boldsymbol{b_0} \tag{7.5}$$

where the matrix $\boldsymbol{A_o}$ and the vector $\boldsymbol{b_0}$ are the constraint matrix of control input,

$$\boldsymbol{u}_{D_{min}} \leq \boldsymbol{u}_{D_N} \leq \boldsymbol{u}_{D_{max}}$$
$$\boldsymbol{u}_{D_{min}} \leq \Delta \boldsymbol{u}_{D_N} + \boldsymbol{u}_{D_{N,0}} \leq \boldsymbol{u}_{D_{max}}$$
$$\boldsymbol{u}_{D_{min}} - \boldsymbol{u}_{D_{N,0}} \leq \Delta \boldsymbol{u}_{D_N} \leq \boldsymbol{u}_{D_{max}} - \boldsymbol{u}_{D_{N,0}}$$

7.3 Static feedforward control (with steady state gain)

The standard state space model in discrete time is given by,

$$\boldsymbol{x}_{D_{k+1}} = \boldsymbol{A}_D\boldsymbol{x}_{D_k} + \boldsymbol{B}_D\boldsymbol{u}_{D_k}$$

At steady state, $\boldsymbol{x}_{D_{k+1}} = \boldsymbol{x}_{D_k}$

$$(\boldsymbol{I} - \boldsymbol{A}_D)\boldsymbol{x}_{D_k} = \boldsymbol{B}_D\boldsymbol{u}_{D_k}$$
$$\boldsymbol{x}_{D_k} = (\boldsymbol{I} - \boldsymbol{A}_D)^{-1}\boldsymbol{B}_D\boldsymbol{u}_{D_k}$$

Assuming at steady state, system output $\boldsymbol{y}_{D_k}$ is equal to set-point $\boldsymbol{w}_{D_k}$,

$$\boldsymbol{y}_{D_k} = \boldsymbol{w}_{D_k} = \boldsymbol{C}_D(\boldsymbol{I} - \boldsymbol{A}_D)^{-1}\boldsymbol{B}_D\boldsymbol{u}_{D_k}$$

$$\boldsymbol{u}_{D_k} = \underbrace{[\boldsymbol{C}_D(\boldsymbol{I} - \boldsymbol{A}_D)^{-1}\boldsymbol{B}_D]^{-1}}_{\boldsymbol{Z}_{ss}}\boldsymbol{w}_{D_k} \tag{7.6}$$

where $\boldsymbol{Z}_{ss}$ is the steady state gain.

7.4 State estimation / observer

Since the only measurable variable is motor speeds, the remaining state variables (currents) have to be estimated with state estimation. A general block diagram is depicted in figure 30. The dynamics model is rewritten into an estimation model incorporating the measured variable as,

$$\begin{aligned} \boldsymbol{x}_e(k+1) &= \boldsymbol{A}_e\boldsymbol{x}_e(k) + \boldsymbol{B}_e\boldsymbol{u}_D(k) \\ \boldsymbol{y}_e(k) &= \boldsymbol{C}_e\boldsymbol{x}_e(k) \end{aligned} \tag{7.7}$$

where $\boldsymbol{x}_e$ and $\boldsymbol{y}_e$ are the estimated state variables (currents and motor speeds) and outputs (velocities), and the matrices $\boldsymbol{A}_e$ and $\boldsymbol{B}_e$ are same as that of dynamics model eq (7.1) and,

$$\boldsymbol{C}_e = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Assuming that the system is observable, then an estimation feedback gain $\boldsymbol{K}$ can be designed which ensures the estimated state $\boldsymbol{x}_e$ will asymptotically approach the actual state $\boldsymbol{x}_D$.

The estimated state $\boldsymbol{x}_e(k+1)$ is then described as,

$$\begin{aligned} \boldsymbol{x}_e(k+1) &= \boldsymbol{A}_e\boldsymbol{x}_e(k) + \boldsymbol{B}_e\boldsymbol{u}_D(k) + \boldsymbol{K}[\boldsymbol{y}_D(k) - \boldsymbol{C}_e\boldsymbol{x}_e(\boldsymbol{k})] \\ &= (\boldsymbol{A}_e - \boldsymbol{K}\boldsymbol{C}_e)\boldsymbol{x}_e(k) + \boldsymbol{B}_e\boldsymbol{u}_D(k) + \boldsymbol{K}\boldsymbol{y}_D(k) \end{aligned} \tag{7.8}$$

where the observer gain $\boldsymbol{K}$**,** can be determined as a solution of the dual task of an infinite horizon LQ control problem. In Matlab, by minimizing the cost function,

$$\min_{\boldsymbol{K}} J = \sum_{i=0}^{\infty}[\Delta\boldsymbol{x}_D{}^T(i)\boldsymbol{Q}_e\Delta\boldsymbol{x}_D(i) + (\boldsymbol{y}_D(i) - \boldsymbol{y}_e(i))^T\boldsymbol{R}_e(\boldsymbol{y}_D(i) - \boldsymbol{y}_e(i))] \tag{7.9}$$

where $\Delta\boldsymbol{x}_D = \boldsymbol{x}_D(k) - \boldsymbol{x}_e(k)$ and $\boldsymbol{Q}_e$ and $\boldsymbol{R}_e$ are positive semi definite matrices.

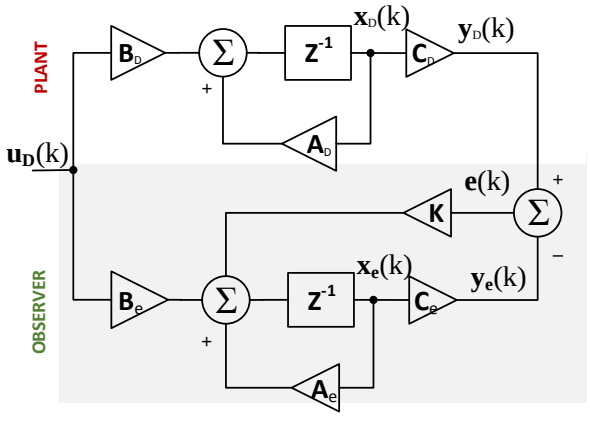

Figure 30 General block diagram of state observer

7.5 Simulation results

The motor and chassis parameters are chosen as listed in Table III-IV. Assuming the wheel speeds are measurable (by wheel encoders), the system outputs (tangential and angular velocities) are calculated from these values with a sampling time of 10ms. All the state variables are estimated as in eq (7.8). The observer gain is calculated in Matlab, $\mathbf{K} = \mathtt{dlqr}(\mathbf{A}_e^T, \mathbf{C}_e^T, \mathbf{Q}_e, \mathbf{R}_e)$ with matrices $\boldsymbol{Q}_e$ and $\boldsymbol{R}_e$ as identity matrices. Figure 31 shows the simulation results of state estimation with zero initial condition for estimated state variables and system states with constant values (both motors are powered with 8V i.e. only linear motion). The dashed line represents system states and the bold line represents estimated state variable. The wheel speeds are calculated from reference tangential and angular velocities by eq (7.2).

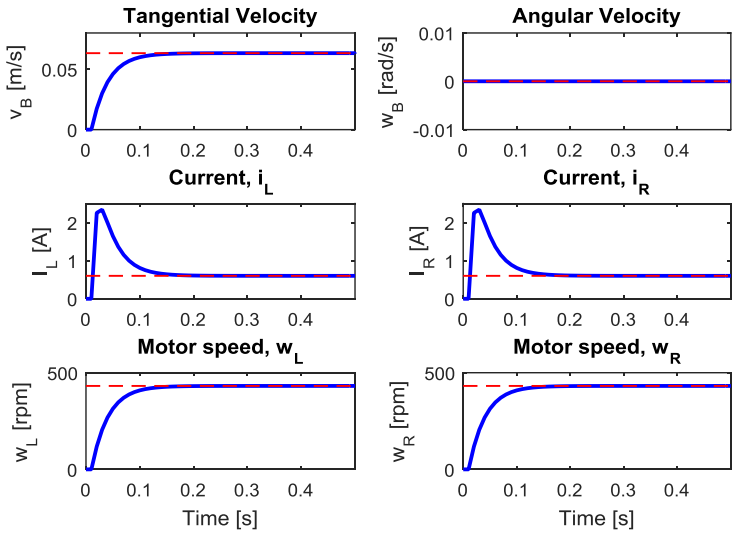

Figure 31 Dynamics state estimation– State variables and outputs

The reference velocities were tracked by two discrete PID control in the form of eq (7.3). The reference wheel speeds were calculated by eq (7.2). The controller parameters of the mobile robot dynamics control with PID controllers were $K_P = [0.5\ 0.5], K_I = [4.5\ 4.5]$.

LMPC was performed by calculating the predicted state variables and outputs as in eq (7.4), using the optimization function as in eq (7.5), the optimized control sequence for a horizon length of N_D was calculated, only the first control action was applied. Constraints for input were considered with $-8\,V \leq U_0 u_L \leq 8\,V; -8\,V \leq U_0 u_R \leq 8\,V$. The controller parameters were $N_D = 5, \boldsymbol{Q}_D = diag(1,1), \boldsymbol{R}_D = diag(0.1,0.1), \boldsymbol{Q}_{D_N} = [0.1\ 0.1\ 0.001\ 0.001]$.

The sampling time for the PID controllers was 10ms and that of LMPC was 100ms. In the first simulation experiment, the set-points were a series of step changes in velocities. Figure 32 shows the comparison between PID and MPC control responses. There weren't significant differences in the control response, though MPC showed slightly better performance.

In the second simulation experiment, the reference velocities for a trajectory were generated by an open loop simulation as discussed in section 6.4. Figure 33 shows the system responses of the two control approaches. Constraints were applied to the input for MPC. Figure 34 shows the control actions, motor currents and motor speeds of MPC.

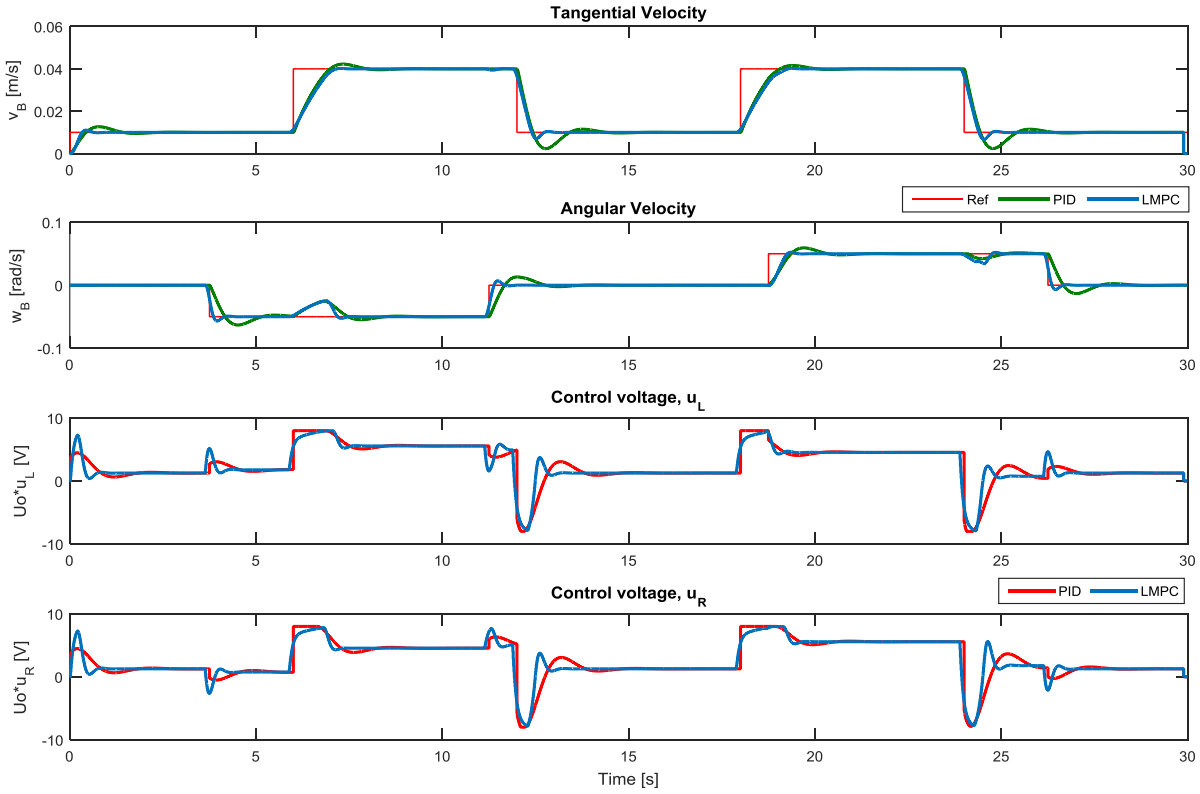

Figure 32 PID vs LMPC - Step Response

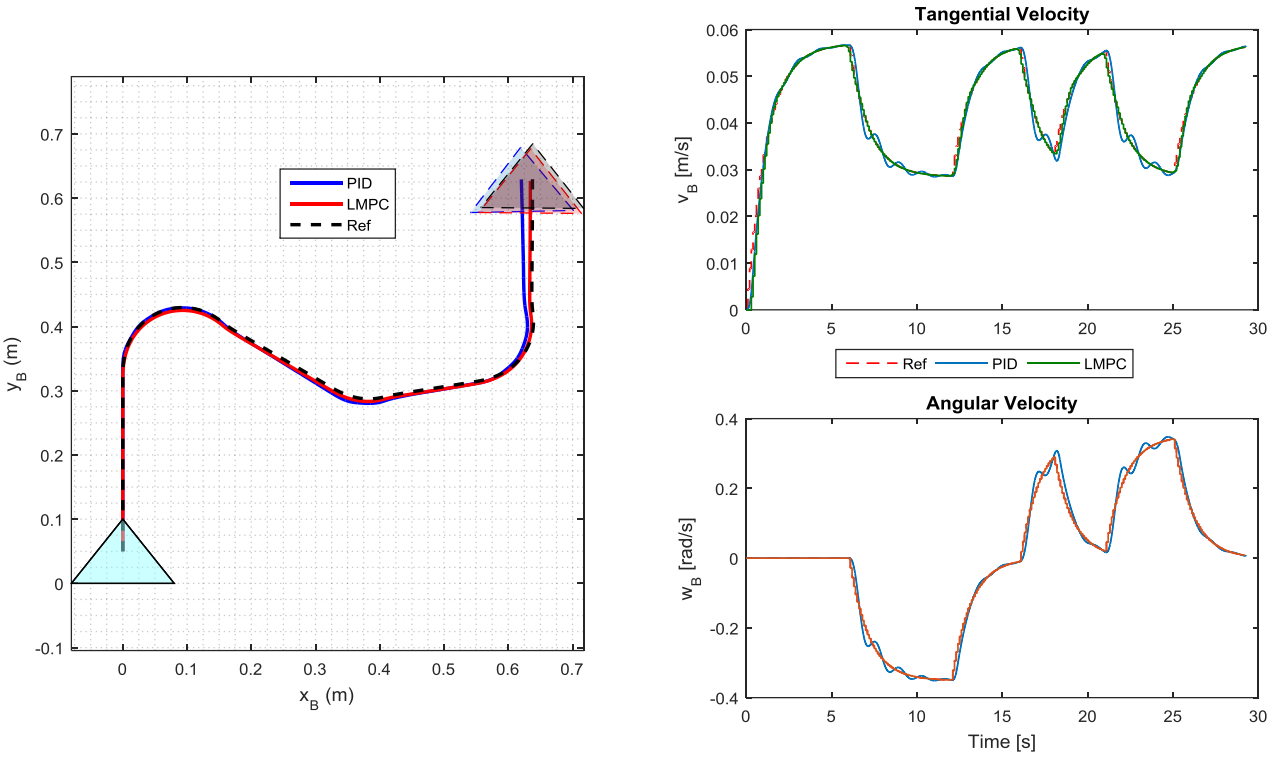

Figure 33 PID vs LMPC control responses (solid line – simulated, dotted line – reference)

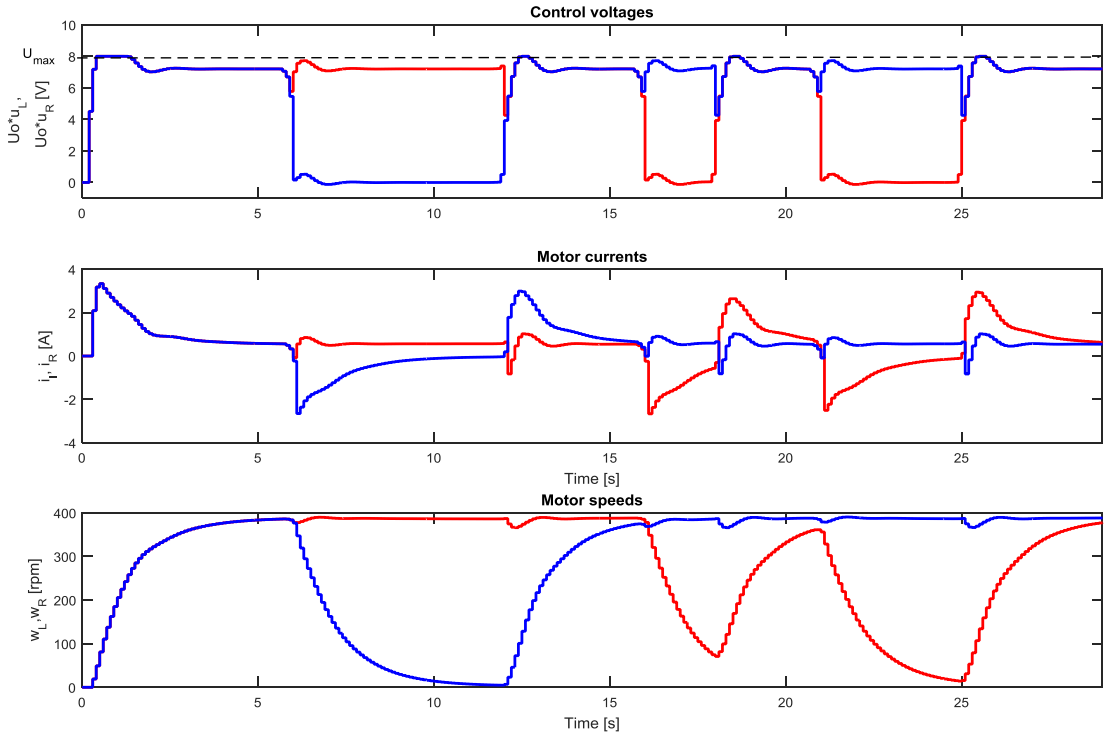

Figure 34 LMPC dynamics control – state variables and control actions (Red -left motor, Blue -right motor)

The main advantage of MPC over PID control is the possibility to consider constraints (real physical constraints) when generating the required control action with respect to current and future states. As well-known problem of MPC implementation - high computational requirement, which makes it harder to work with faster systems. The designed LMPC for dynamics of mobile robot with a sampling time of 100ms (much higher when compared to PID -10ms) requires less than 4.5ms average CPU (for Intel core i5-4310M, 2.70GHz CPU) time for every control step, which clearly shows the possibility of MPC in real application. Furthermore, the computation time can be considerably reduced by coding into C code (automatic code generation in SIMULINK) and an offline code will help to work with real systems.

8. TRAJECTORY TRACKING OF WMR – KINO-DYNAMICS CONTROLLER

In section 5, we discussed trajectory tracking by kinematic controllers by assuming a perfect velocity tracking low-level controller is present, which will generate the desired velocities. This is followed by an assumption that the dynamics of the robot is very fast, and the low-level controller is able to generate motor voltages to make the robot drive with required velocities. However, in practice, this is not always the case, especially when the robot's dynamics is slow and there are physical saturation limits. This brings us to consider dynamics of the robot as well. In section 6, we discussed modelling of the WMR considering motor and chassis dynamics. Dynamics controllers were designed in section 7, considering the robot's parameters and constraints.

To begin with, let us consider a situation when the robot's dynamics is slow and what happens when we assume 'perfect velocity tracking' ability of low-level controller. A feedforward controller is replaced by a low-level controller to see the effect i.e. by considering the dynamics of the WMR. Figure 35 shows the control scheme with static feedforward control as a low-level controller. As discussed in section 7.3, the control actions (motor voltages) by feed forward control, eq (7.6) is given by,

$$\boldsymbol{u}_{D\,k} = \boldsymbol{Z}_{ss}{}^{-1}\boldsymbol{w}_{D\,k}$$

where $\boldsymbol{Z}_{ss}$ is the steady state gain and,

$$\boldsymbol{u}_{D\,k} = \begin{bmatrix} u_L(k) \\ u_R(k) \end{bmatrix}; \boldsymbol{w}_{D\,k} = \begin{bmatrix} v_{B,r}(k) \\ \omega_{B,r}(k) \end{bmatrix}$$

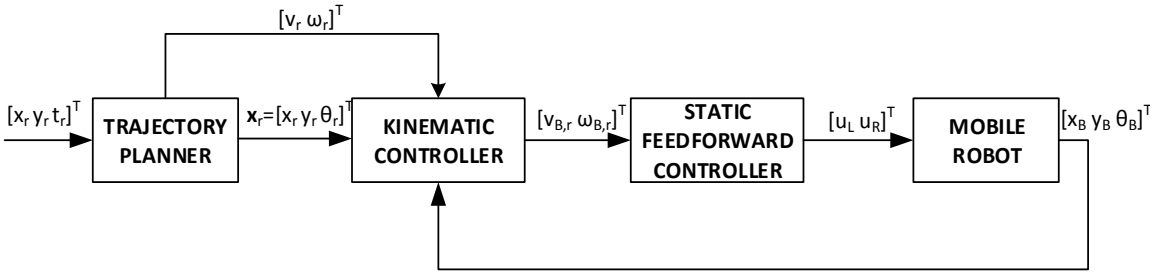

Figure 35 Block diagram of static feedforward dynamics controller

Figure 36 shows the trajectory tracking response of the mobile robot with $NMPC_2$ as kinematic controller (parameters as in section 5, simulation experiment **S4**) without considering constraints on velocities and motor voltages. As can be noted in the figure, large tracking errors

are visible even without considering the constraints on a low level and with the same initial conditions. This shows the importance of feedback controllers at the low level.

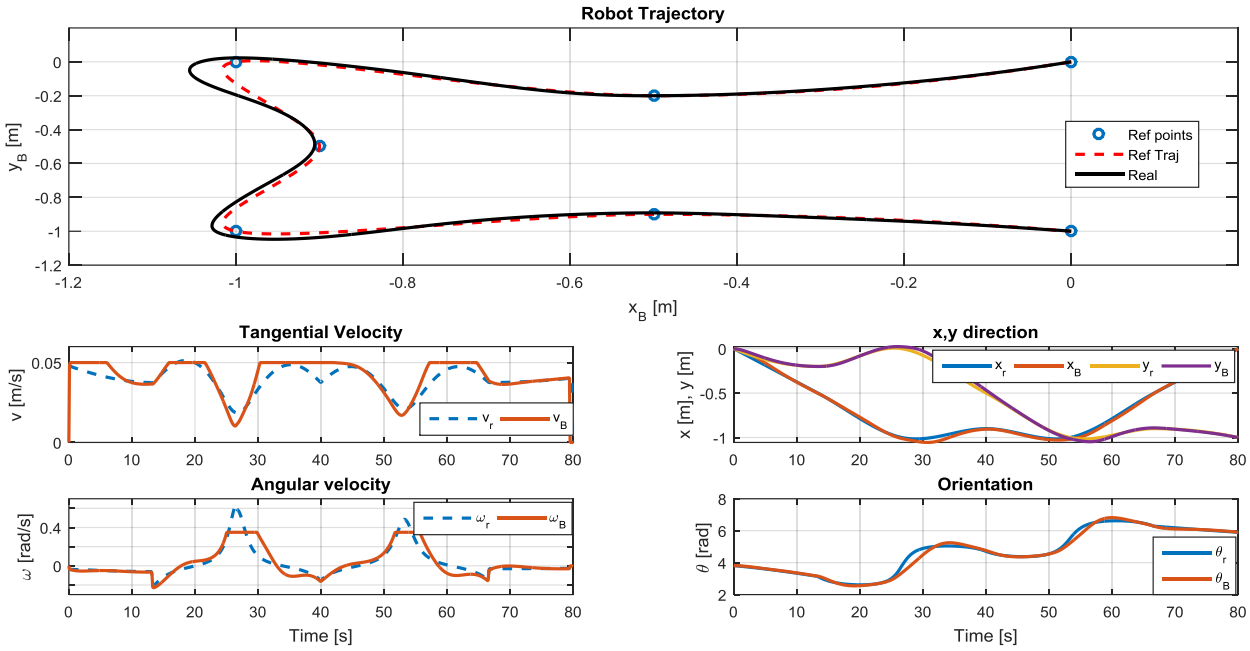

Figure 36 Trajectory tracking response with static feedforward control as low-level controller without considering constraints.

8.1 Trajectory tracking with Kino-Dynamics controller

The trajectory planner generates the reference velocities, position and orientation with the information on time parameterized reference waypoints. The trajectory tracking is done at two levels – kinematic control and dynamics control. A general control structure is shown in figure 37. The high level consists of a kinematic controller with feedback of position and orientation (overhead camera). The outputs of the kinematic controller are reference velocities $(v_{r,B}, \omega_{r,B})$ which are the inputs to the low-level controller. In dynamics control, the motor and chassis dynamics and constraints of the mobile robot are considered. The measurable variables for the dynamics part are motor speeds (measured by encoder). The output of the dynamics controller are motor voltages with respect to current and reference velocities. The constraints on velocities were considered as,

$$-0.05\, m/s \leq v_B \leq 0.05\, m/s \quad ; \quad -0.35\, rad/s \leq \omega_B \leq 0.35\, rad/s \tag{8.1}$$

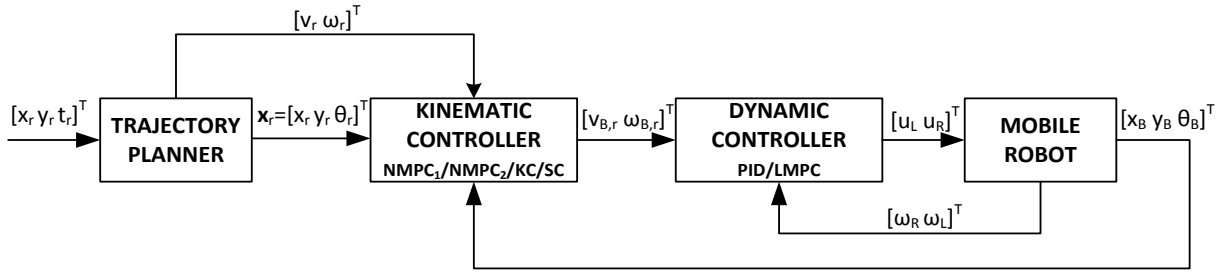

Figure 37 General control structure of Kino-Dynamics controller

8.1.1 Trajectory tracking with low level discrete PID controller

A feedforward controller is replaced with discrete PID controller, eq (7.3), at low level, whose design is discussed in section 7.1.

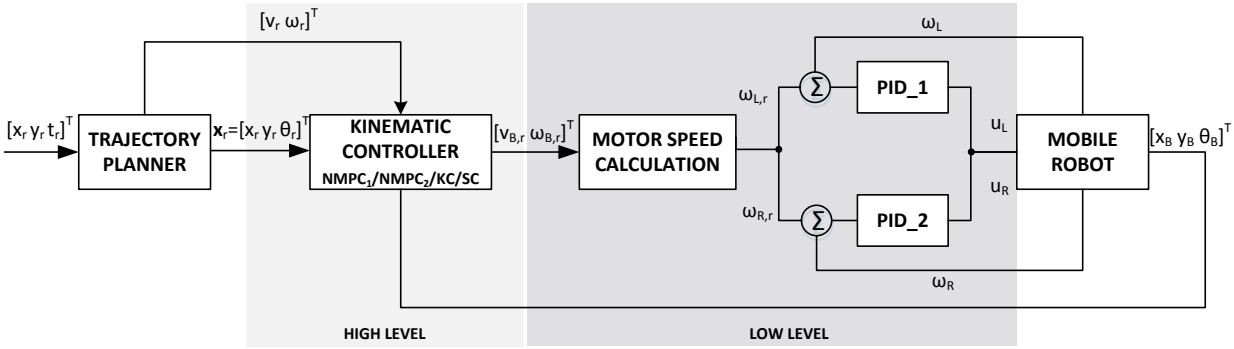

Figure 38 Block diagram of Kino-Dynamics controller with PID as low-level controller

The low-level PID controllers track the reference motor speeds which are calculated from the reference velocities, eq (7.2) - see figure 38. The measured wheel speeds are fed back to the control loop. The PID controllers are sampled at much high frequency (Sampling time 10ms) than the kinematic controller (100ms). The parameters of PID controllers are the same as discussed in section 7. Figure 39 shows the simulation results with Kanayama Controller (KC) as the high-level controller and PID controller as the low-level controller. Constraints were considered for the low-level controller, $-8\,V \leq U_0 u_L \leq 8\,V$; $-8\,V \leq U_0 u_R \leq 8\,V$.

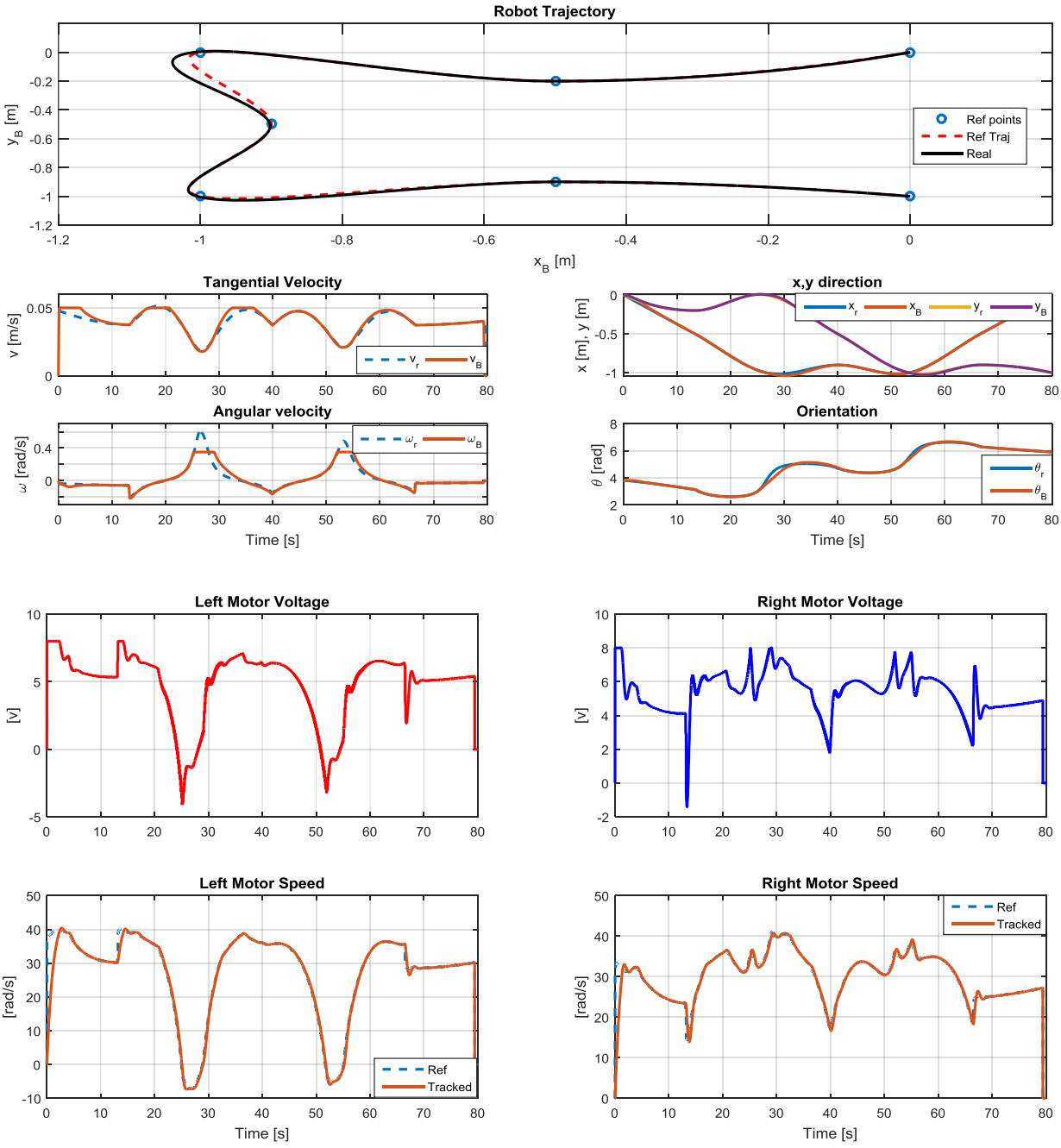

Figure 39 KC-PID trajectory tracking with constraints – Reference vs tracked trajectory, speeds and velocities, control voltages

8.1.2 Trajectory tracking with low level LMPC

LMPC is considered as a low-level controller where the set points are obtained from the output of the kinematic controller. The main advantage of Kino-Dynamics LMPC, in contrast to other classical controllers, is the flexibility in usage of the information about future set-points. In the case of NMPC as a kinematic controller, the velocities control inputs $\boldsymbol{w}_D$ (output of kinematic controller, $\boldsymbol{y}_K$) are vector of velocities calculated from optimal control actions $\boldsymbol{u}_K(i)$, $\forall\, i = \{1,2,\dots,N_K\}$. This future set-point information can be used either as future set-points and/or can be used to calculate the terminal state for low-level controller.

The design of LMPC is discussed in section 7.2. The dynamics model consists of currents and motor speeds as state variables. The state variables are estimated, as discussed in section 7.4, from the measured motor speeds. The objective function of LMPC consists of costs to control effort, control error and terminal state error. Figure 40 shows the general block diagram of MPC as a low-level controller.

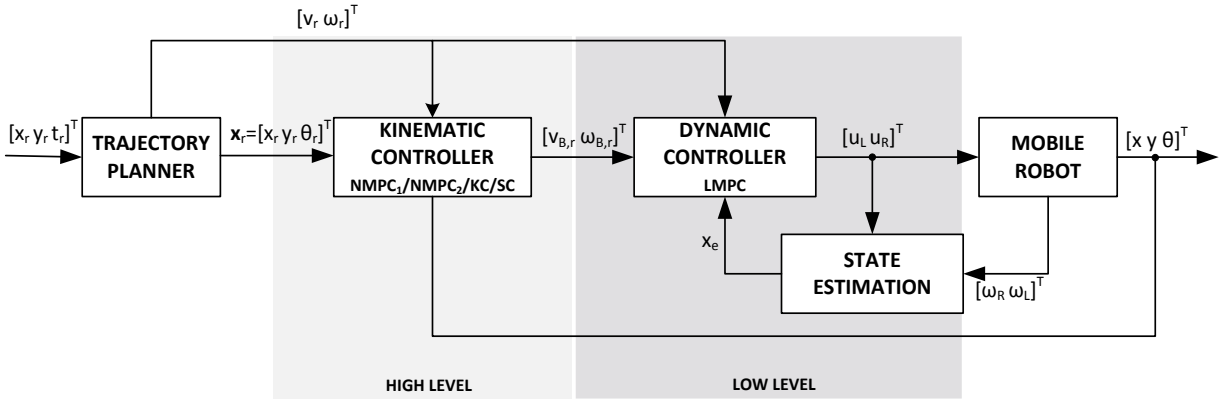

Figure 40 Block diagram of Kino-Dynamics controller with LMPC as low-level controller

The simulation experiment of trajectory tracking of WMR with a Samson controller, eq (5.11) as the kinematic controller and LMPC as the low-level dynamics controller was performed. The set-points for LMPC controller were reference velocities $(v_{B,r}, \omega_{B,r})$ which were kept constant during the entire horizon. There are several options for choice of terminal state calculation – e.g. use the reference velocity from the kinematic controller or use the reference velocity from the trajectory planner. In the simulation experiments, reference velocity from the trajectory planner was chosen and the reason for this choice is explained in the forthcoming section.

The controller parameters of low level LMPC were $N_D = 5, \boldsymbol{Q}_D = diag(1,1), \boldsymbol{R}_D = diag(0.1,0.1), \boldsymbol{Q}_N = [0.1\ 0.1\ 0.001\ 0.001]$ and sampling time of 100ms. Velocity constraints as in

eq (8.1) were considered. Figure 41 shows the result of the SC-LMPC trajectory tracking simulation experiment - reference and tracked trajectory and velocities, estimated currents, control voltages, and the measured motor speeds. It can be noted that the performance of the controller was better than with PID controller as a low-level controller.

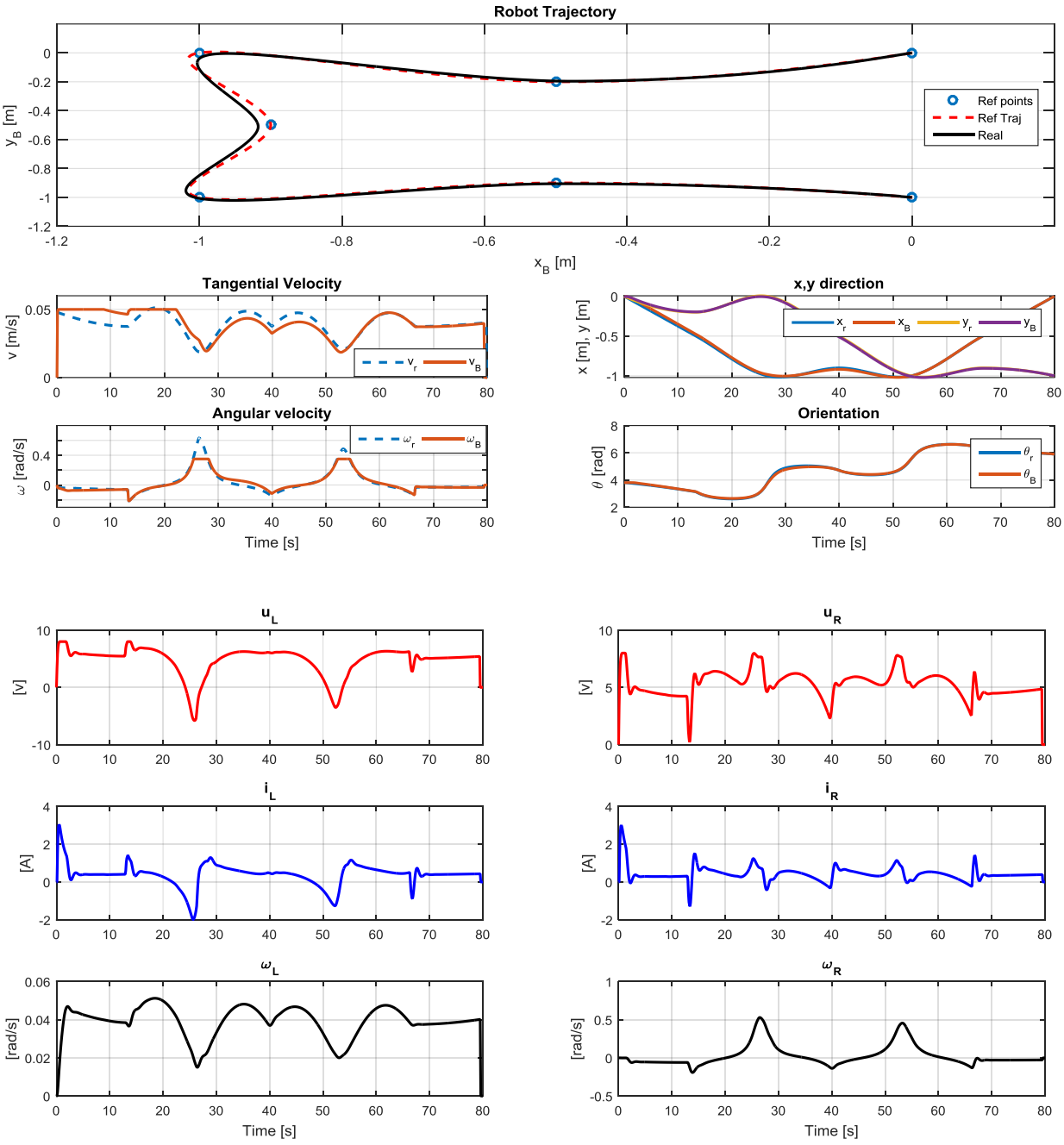

Figure 41 SC-LMPC trajectory tracking with velocity constraints – Reference vs tracked trajectory, speeds and velocities, control voltages

8.2 Comparative analysis of control structures

The effectiveness of the control structures were analyzed by various simulation experiments. The high-level kinematic controllers were $NMPC_1$, $NMPC_2$, Kanayama Controller and Samson Controller as discussed in section 5. The low-level kinematic controllers were PID and LMPC as discussed in section 7. Table V shows the SSE's of the simulation experiments conducted with different control structures. Figures corresponding to particular simulation experiments are also listed. The initial pose of the mobile robot and reference robot were considered as the same (i.e. same initial condition). Comparison has also been made with a "perfect velocity tracker" and a feed forward control. The "perfect velocity tracker" refers to the assumption that the kinematic controller will generate the required tangential and angular velocities, assuming that there is a perfect velocity tracking control at low level on the mobile robot. Feedforward control refers to the condition when the kinematic controller generates the velocities, and what happens when there isn't a perfect velocity tracker i.e. dynamics of robot is taken care of.

Table V Comparison of SSE's of trajectory tracking with constraints

Simulation Exp	Figure	Kinematic Controller	Dynamics Controller[2]	SSE [SSE_{xy} SSE_{θ}]
S1	--	$NMPC_1$	-none-	[1.0101 6.3563]
S2	--		Feedforward	[7.1278 30.854]
S3	--		PID	[2.7950 13.274]
S4	--		LMPC	[0.1334 0.8368]
S5	--	$NMPC_2$	-none-	[0.1690 5.9869]
S6	Fig 36		Feedforward	[0.5182 16.401]
S7	--		PID	[0.2225 7.1509]
S8	Fig 42		LMPC	[0.1264 0.6414]
S9	--	KC	-none-	[0.0604 6.0890]
S10	--		Feedforward	[0.6867 21.274]
S11	Fig 39		PID	[0.0995 7.5049]
S12	--		LMPC	[0.1819 1.1997]
S13	--	SC	-none-	[0.0606 6.0924]
S14	--		Feedforward	[0.6862 21.273]
S15	--		PID	[0.0997 7.5084]
S16	Fig 41		LMPC	[0.1819 1.1996]

[2] 'none' refers to the assumption of perfect velocity tracking, i.e. without considering dynamics of robot

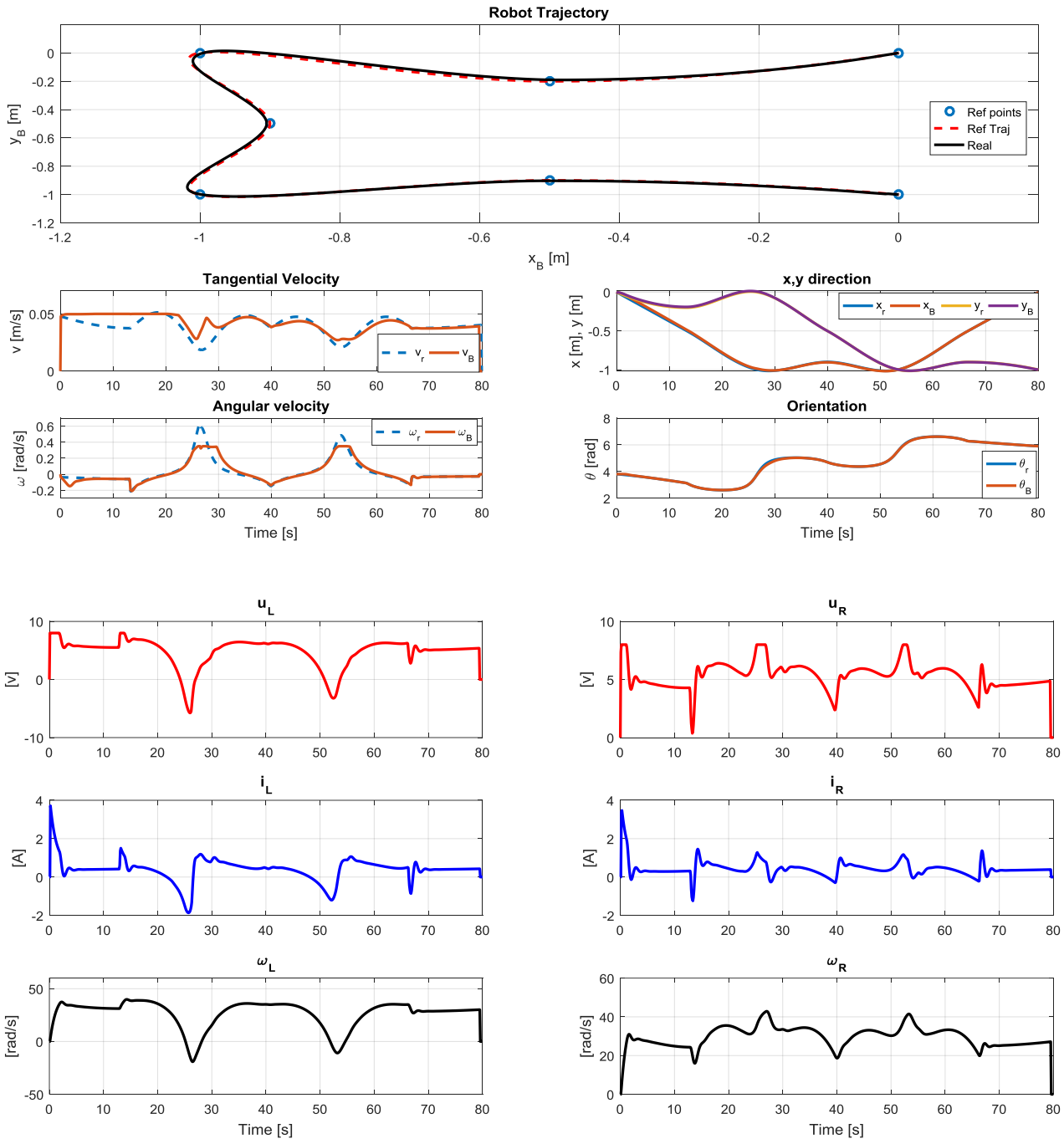

Figure 42 NMPC$_2$-LMPC trajectory tracking with constraints - Reference vs tracked trajectory, control voltages, estimated currents and measured speed

8.3 Discussion

Perfect velocity tracking: The dynamics of the robot was not considered (w/o dynamics controller) and the controllers generated control velocities. All the control structures produced low SSE other than with $NMPC_1$, which produced significantly high tracking errors. The state tracking controllers, KC and SC, were slightly better than $NMPC_1$.

Feedforward control: Dynamics of the mobile robot, as discussed in section 6, were considered. The motor voltages were calculated by the static feedforward gain as in eq (7.6). It can be noted that very high tracking errors are generated when dynamics of robot is taken into account. This shows the significance of feedback control on low level considering dynamics of the robot. Again, all controllers other than $NMPC_1$ generated comparatively the same tracking errors, even though $NMPC_2$ was slightly better.

Low level PID control: The motor voltages are generated according to the reference motor speeds and current speeds of motor. The advantage of feedback control on low level is visible by comparing the SSE's with that of feedforward control. In the same way as the previous two cases, $NMPC_2$ produced low tracking errors.

Low level LMPC: The trend of tracking errors of all the three previous cases were obvious. However, very interesting results can be noted in the case of LMPC on low level. The SSE's were significantly lower for all the controllers when compared with feedforward or PID control, see figure 42. The tracking errors of $NMPC_1$ were very high, when compared with other controllers in all the three previous cases, but the control structure $NMPC_1$-LMPC were able to track the robot closer to the reference trajectory than state tracking controllers. The $NMPC_2$-LMPC control combination was the best among all the other combination. The advantages of MPC on the dynamics level can be clearly seen in the figure 43, when compared to the PID control, feedforward control and perfect velocity tracker (only kinematic control).

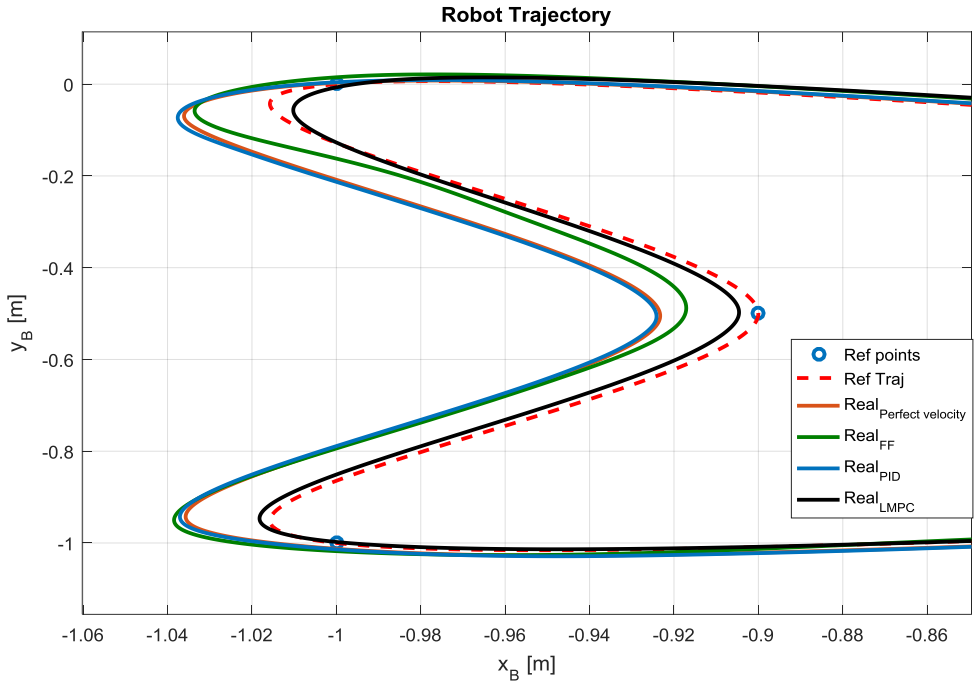

Figure 43 Comparison of trajectory tracking –$NMPC_2$ as high-level controller

It is interesting to note that, the Kino-Dynamics controller performance was better than the "perfect velocity tracker". This is achieved by the choice of set-points on low level LMPC controller. The set-point for the N_D horizon is kept constant, where the terminal state is considered to be the same as that of the original reference velocity. Consequently, the terminal state penalization will force the robot to converge to the reference trajectory. By contrast, the MPC on low level control will not only track the reference velocity from the kinematic controller, but also will reduce the overall tracking error caused by kinematic control. This was achieved by proper choice of MPC structure.

9. CONCLUSION

The trajectory-tracking problem of the mobile robot is a well-researched area. However, most of the research has concentrated on the kinematics of mobile robot, assuming an ideal dynamics controller ("perfect velocity tracker") is present at the lower level. This is followed by the assumption that dynamics of the mobile robot is faster, and a dynamics controller is able to generate the required (tangential and angular) velocities at every time instant. But, in practice this is not an ideal solution especially when the dynamics is slower and highly non-linear.

The thesis investigates the possibilities of using MPC in trajectory tracking problem for the mobile robot at both kinematic and dynamics level. Linear and Non-linear state space MPC were derived by decomposing the free and forced responses. Criteria consisting of penalization of control error, control effort and terminal state deviation were considered for calculating the optimal control actions. The trajectory-tracking problem was studied by separating the thesis into three parts.

Firstly, the kinematics of a non-holonomic mobile robot was considered assuming a perfect velocity tracker is present at the lower level. The basic non-linear kinematic equations were linearized into two LTV models based on the choice of coordinate frames. The successive linear model and error-based model were derived, with respect to the world coordinate system and local coordinate system of the mobile robot respectively. Non-linear MPCs were applied considering these models to the kinematic trajectory-tracking problem, with a cost function consisting of penalization of control error, control effort and terminal state deviation. The responses were compared with state-of-the-art control techniques (Kanayama and Samson controllers) and the simulation results showed that the NMPC with an error-based model was able to track the mobile robot with comparatively less tracking error when compared with the other three controllers.

Secondly, the mathematical model of dynamics of the mobile robot was derived and linear MPC was applied to track the desired velocities. A linear state space model was derived with state variables as motor speeds and currents, control inputs as motor control voltages, and the outputs as velocities. Linear MPC and discrete PID controllers were designed and simulated for the velocity-tracking problem. The simulation results showed that efficiency of LMPC for dynamics control when compared to PID control. This was achieved by considering constraints, and optimizing the criteria with weights on control effort, control error and terminal state deviation.

Finally, the trajectory tracking problem was solved by different Kino-Dynamics control structures. The simulation results show the advantage of MPC on dynamics part in contrast to the assumption of "perfect velocity tracker". It has been noticed that, MPC on the dynamics part can, not only generate the optimal control action for the dynamics part, but also can affect the kinematic part and decrease the overall tracking errors. The NMPC-LMPC control structure have several advantages for the trajectory tracking problem: it can generate optimal control actions by considering system constraints, increase the overall stability, decrease the overall tracking errors etc.

To sum up, the thesis proposes a simple solution for the trajectory-tracking problem by leveraging various advantages of MPC. Though the thesis mainly concentrated on simulation analysis, various implementation issues and suggestions are also covered to some extent. The simulation results demonstrate the advantages of Kino-Dynamics MPC over various state-of-the-art approaches. However, several implementation issues are open for future research.

9.1 Future directions

Although we consider that the objectives of this thesis have been accomplished, there are plenty of improvements that could be done in order to achieve better results. Here, only the main points are summarized.

MPC – Even though the non-linear MPC with LTV and the non-linear model proposed in the thesis, was able to achieve the desired results, there are various designs of NMPCs available in the literature. Explicit MPC lessens the computation burden when dealing with faster systems when comes to real world implementation.

Kinematics – The derived kinematics models for non-holonomic robot only deals with the trajectory-tracking problem and doesn't consider the point stabilization problem. There are various other models in literature (e.g. (Xie & Fierro 2008)) which consider both the motion control problems simultaneously.

Dynamics – The dynamics model was derived without considering various other non-linearities (e.g. motor saturation, backlash, dead zone etc.). The moment of inertia of the robot was assumed as constant, however it varies with the load on mobile robot. For better performances, non-linear model is more suitable and the parameters have to be identified by offline-online methods. Furthermore, advanced state estimation methods like the Moving Horizon estimation, Extended Kalman Filter, particle filter etc. can also be employed for model uncertainties and, together with non-linear MPC would steer the robot through the desired direction.

Implementation – There are several implementation issues which can be expected, especially the problem with time delays. A Model-Based Design (MBD) is much more suited and SIMULINK allows lot of possibilities for developing MBD, e.g. time delay simulation, real time testing, automatic code generation, Hardware-in-the-Loop (HIL) testing etc.

Localization – For laboratory experiments, an overhead camera together with image processing algorithms can localize the robot. As part of the thesis work, localization of mobile robot with an overhead camera and Raspberry Pi has been developed. However, this is not included in the thesis, as the main objective was to do simulation analysis. In case of real world implementation, a GPS sensor with sensor fusion techniques is recommended.

REFERENCES

Borenstein, J. et al., 1997. Mobile robot positioning: Sensors and techniques. *Journal of Robotic Systems*, 14(4), pp.231–249.

Brockett, R.W., 1983. Asymptotic stability and feedback stabilization. *Differential Geometric Control Theory*, pp.181–191.

Camacho, E.F. & Bordons, C. (Carlos), 2004. *Model predictive control*, Springer.

Campion, G., Bastin, G. & D'Andréa-Novel, B., 1996. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 12(1), pp.47–62.

Chen, H. et al., 2009. Moving Horizon H-infinity Tracking Control of Wheeled Mobile Robots With Actuator Saturation. *Ieee Transactions on Control Systems Technology*, 17(2), pp.449–457.

Cong, S., Li, G. & Feng, X., 2010. Parameters Identification of Nonlinear DC Motor Model Using Compound Evolution Algorithms. *Lecture Notes in Engineering and Computer Science*, 2183(1), pp.15–20.

Cook, G., 2011. *Mobile Robots: Navigation, Control and Remote Sensing*,

Das, T. & Kar, I.N., 2006. Design and implementation of an adaptive fuzzy logic-based controller for wheeled mobile robots. *IEEE Transactions on Control Systems Technology*, 14(3), pp.501–510.

Fierro, R. & Lewis, F.L., 1995. Control of a nonholonomic mobile robot: backstepping kinematics into dynamics. In *Proceedings of the 34th IEEE Conference on Decision and Control (CDC)*. pp. 3805–3810.

Fierro, R. & Lewis, F.L., 1998. Control of a nonholonomic mobile robot using neural networks. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 9(4), pp.589–600.

Fukao, T., Nakagawa, H. & Adachi, N., 2000. Adaptive tracking control of a nonholonomic mobile robot. *IEEE Transactions on Robotics and Automation*, 16(5), pp.609–615.

Hatab, A.A. & Dhaouadi, R., 2013. Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework. *Advances in Robotics & Automation*, 2(2).

Henson, M.A., 1998. Nonlinear model predictive control: current status and future directions. *Computers & Chemical Engineering*, 23(2), pp.187–202.

Hu, T. & Yang, S.X., 2001. An efficient neural controller for a nonholonomic mobile robot. In *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA*. pp. 461–466.

Kanayama, Y. et al., 1990. A stable tracking control method for an autonomous mobile robot. *Proceedings., IEEE International Conference on Robotics and Automation*, 30(5),

pp.384–389.

Kanjanawanishkul, K., 2012. Motion control of a wheeled mobile robot using model predictive control: A survey. *KKU Research Journal*, 17(5), pp.811–837.

Kim, M.S., Shin, J.H. & Lee, J.J., 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)*. IEEE, pp. 1816–1821.

Klančar, G. & Škrjanc, I., 2007. Tracking-error model-based predictive control for mobile robots in real time. *Robotics and Autonomous Systems*, 55(6), pp.460–469.

Kolmanovsky, I. & McClamroch, N.H., 1995. Developments in Nonholonomic Control Problems. *IEEE Control Systems*, 15(6), pp.20–36.

Kuhne, Felipe, Walter Fetter Lages, J.M.G. da S.J., 2004. Model predictive control of a mobile robot using linearization. In *Proceedings of mechatronics and robotics*. pp. 525–530.

Kunhe, F., J. Gomes, W.F., 2005. Mobile robot trajectory tracking using model predictive control. In *II IEEE Latin-american Robotics Symposium*.

Lages, W.F. & Vasconcelos Alves, J.A., 2006. REAL-TIME CONTROL OF A MOBILE ROBOT USING LINEARIZED MODEL PREDICTIVE CONTROL. *IFAC Proceedings Volumes*, 39(16), pp.968–973.

De Luca, A. & Oriolo, G., 1995. Modeling and control of nonholonomic mechanical systems. *Kinematics and Dynamics of Multi-Body Systems*, pp.277–342.

De Luca, A., Oriolo, G. & Vendittelli, M., 2001. Control of Wheeled Mobile Robots: An Experimental Overview. In Springer Berlin Heidelberg, pp. 181–226.

Maurovic, I., Baotic, M. & Petrovic, I., 2011. Explicit Model Predictive Control for trajectory tracking with mobile robots. In *2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. pp. 712–717.

Niku, S.B., Saeed B., 2001. *Introduction to robotics analysis, systems, applications*, Prentice Hall.

Oriolo, G., Luca, A. De & Vendittelli, M., 2002. WMR control via dynamic feedback linearization: design, implementation, and experimental validation. *IEEE Transactions on Control Systems Technology*, 10(6), pp.835–852.

Pourboghrat, F. & Karlsson, M.P., 2002. Adaptive control of dynamic mobile robots with nonholonomic constraints. *Computers and Electrical Engineering*, 28(4), pp.241–253.

Rossiter, J.A., 2003. *Model-based predictive control : a practical approach*, CRC Press.

Saab, S.S. & Kaed-Bey, R.A., Parameter identification of a DC motor: an experimental approach. In *ICECS 2001. 8th IEEE International Conference on Electronics, Circuits and Systems (Cat. No.01EX483)*. IEEE, pp. 981–984.

Samson, C. & Ait-Abderrahim, K., 1991. Feedback control of a nonholonomic wheeled cart in

Cartesian space. In *Proc. IEEE International Conf. on Robotics and Automation*. pp. 1136–1141.

Sarkar, N., Yun, X. & Kumar, V., 1994. Control of Mechanical Systems With Rolling Constraints: Application to Dynamic Control of Mobile Robots. *The International Journal of Robotics Research*, 13(1), pp.55–69.

Siegwart, R., Nourbakhsh, I.R. & Scaramuzza, D., 2011. *Introduction to autonomous mobile robots*,

Triggs, B., 1993. Motion Planning for Nonholonomic Vehicles: An Introduction.

Völlmecke, D.-I.I., 2013. Parameter identification of dc motors. *IMC Berlin*.

Wenjie Dong et al., 2000. Tracking control of uncertain dynamic nonholonomic system and its application to wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 16(6), pp.870–874.

Xie, F.X.F. & Fierro, R., 2008. First-state contractive model predictive control of nonholonomic mobile robots. *2008 American Control Conference*, (ii), pp.3494–3499.

Yang, J.-M. & Kim, J.-H., 1999. Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 15(3), pp.578–587.

LIST OF AUTHOR'S PUBLICATIONS

Rahul Sharma K., Frantisek Dusek and Daniel Honc, 2017. " Comparitive study of predictive controllers for trajectory tracking of non-holonomic mobile robot." In *2017 21st International Conference on Process Control (PC)*. IEEE. (Submitted)

Frantisek Dusek, Daniel Honc and **Rahul Sharma K.**, 2017. "Modelling of ball and plate system based on first principle model and optimal control." In *2017 21st International Conference on Process Control (PC)*. IEEE. (Submitted)

František Dušek, Daniel Honc and **Rahul Sharma K.**, 2017. " Optimal Control With Disturbance Estimation." . In *Proceedings – 31st European Conference On Modelling And Simulation, ECMS 2017*. (Accepted)

Rahul Sharma K., Daniel Honc and František Dušek, 2016. "Predictive Control of Differential Drive Mobile Robot Considering Dynamics and Kinematics." . In *Proceedings - 30th European Conference On Modelling And Simulation ECMS 2016*. Nottingham: European Council Modelling & Simulation, 2016. pp. 354-360.

Rahul Sharma K., Daniel Honc, František Dušek and Gireesh Kumar T., 2016. "Frontier Based Multi Robot Area Exploration Using Prioritized Routing." In *Proceedings - 30th European Conference On Modelling And Simulation Ecms 2016*. Nottingham: European Council Modelling & Simulation, 2016. pp. 25-30.

František Dušek, Daniel Honc, **Rahul Sharma K.** and Libor Havlíček, 2016. "Inverted Pendulum Optimal Control Based on First Principle Model." In *Automation Control Theory Perspectives In Intelligent Systems*. Berlin: Springer-Verlag Berlin, 2016. pp. 63-74.

Daniel Honc, **Rahul Sharma K.**, Anuj Abraham, František Dušek and Natarajan Pappa, 2016. "Teaching and Practicing Model Predictive Control." In *IFAC Papersonline*. Amsterdam: Elsevier Science BV, 2016. pp. 34-39.

František Dušek, Daniel Honc and **Rahul Sharma K.**, 2015. "Dynamic Behaviour of Differentially Steered Mobile Robot." *Transactions on Electrical Engineering*, 2015, vol. 4, no. 2, pp. 51-58.

František Dušek, Daniel Honc and **Rahul Sharma K.**, 2015. "A Comparative Study of State-Space Controllers with Offset-Free Reference Tracking." In *2015 20th International Conference on Process Control (PC)*, pp. 176–80.

Anuj Abraham, Natarajan Pappa, Daniel Honc and **Rahul Sharma K.**, 2015. "A Hybrid Method for Determination of Effective Poles Using Clustering Dominant Pole Algorithm." *International Journal of Electrical and Computer Engineering*, World Academy of Science, Engineering and Technology, vol.2, no. 3, pp. 102-106.

Rahul Sharma K., Daniel Honc and František Dušek, 2015. "Model Predictive Control of Trajectory Tracking of Differentially Steered Mobile Robot." In *Intelligent Data Analysis and Applications: Proceedings of the Second Euro-China Conference on Intelligent Data Analysis and Applications. ECC 2015*. Berlin: Springer, 2015. pp. 85-95.

Rahul Sharma K., Daniel Honc and František Dušek, 2015. "Sensor Fusion for Prediction of Orientation and Position from Obstacle Using Multiple IR Sensors an Approach Based on Kalman Filter." In *Proceedings of 19th International Conference on Applied Electronics 2014*. Plzeň: Západočeská univerzita, 2014. pp. 263-266.

Rahul Sharma K., Daniel Honc and František Dušek, 2014. "Sensor Fusion: An Application to Localization and Obstacle Avoidance in Robotics Using Multiple IR Sensors." In *Nostradamus 2014: Prediction, Modeling and Analysis of Complex Systems*. New York: Springer, 2014. pp. 385-392.

Rahul Sharma K., 2014."Design and development of GUI for mapping of obstacle by sensor fusion techniques." In *XXXVIII. Seminar ASR 2014 Instruments and Control*. Ostrava: Vysoká škola báňská-Technická univerzita Ostrava, 2014. pp. 88-94.

Daniel Honc, František Dušek and **Rahul Sharma K.**, 2014. "Gunt RT 010 Experimental Unit Modelling and Predictive Control Application." In *Nostradamus 2014: Prediction, Modeling and Analysis of Complex Systems*. New York: Springer, 2014. pp. 175-184.