SAPIENZA
UNIVERSITÀ DI ROMA

Enhancing Tractor-Trailer System Stability through Nonlinear Anti-Jackknifing Model Predictive Control

Faculty of Information Engineering, Computer Science and Statistics
Department of Computer, Control, and Management Engineering
Master of Science in Control Engineering

Matteo Facci
ID number 1597454

Advisor
Prof. Giuseppe Oriolo

Co-Advisor
Dr. Tommaso Belvedere

Academic Year 2021/2022

Thesis defended on 30 May 2023
in front of a Board of Examiners composed by:

Prof. Giuseppe Oriolo (chairman)
Prof. Febo Cincotti
Prof. Andrea Cristofaro
Prof. Giorgio Grisetti
Prof. Daniela Iacoviello
Prof. Leonardo Lanari
Prof. Marilena Vendittelli

Enhancing Tractor-Trailer System Stability through Nonlinear Anti-Jackknifing Model Predictive Control
Master's thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: May 15, 2023

Author's email: matteo.facci@outlook.it

Abstract

The field of automotive control is continuously evolving to meet the increasing demands for more efficient, safe, and environmentally friendly vehicles. Tractor-trailer systems, which play a vital role in the transportation of goods, pose a unique control challenge due to their nonlinear dynamics and constraints. One major challenge faced by drivers is the "jackknife" effect, which occurs during reversing maneuvers and can lead to difficult and dangerous situations.

This thesis aims to develop a nonlinear Model Predictive Control (NMPC) technique to address the problem of jackknifing during backward motion in tractor-trailer systems. The proposed control technique uses a mathematical model of the system to make predictions and optimizes a control strategy that keeps the system within safe and efficient bounds. Through simulation and experimental validation, the effectiveness of the proposed control methods in addressing the problem of jackknifing is demonstrated.

The results of this research show the potential of the nonlinear MPC control technique to improve safety and efficiency in reversing tractor-trailers and provide a solution to the jackknifing problem.

Acknowledgments

Ringrazio il Prof. Giuseppe Oriolo, per avermi guidato e supportato nella fase più importante del mio percorso accademico.

Un sentito grazie al Dott. Tommaso Belvedere, correlatore di tesi, per il supporto costante e le dritte indispensabili nella realizzazione di ogni capitolo della mia tesi.

A mamma e papà, al loro costante sostegno ed ai loro insegnamenti senza i quali oggi non sarei ciò che sono. Senza di voi tutto questo non sarebbe stato possibile. Prima o poi tutte le ansie, le preoccupazioni e le esasperazioni che vi ho gentilmente donato dovevano essere ripagate. Spero di averlo fatto nel migliore dei modi rendendovi fieri.

A mio fratello Davide e mia sorella Cecilia, due pezzi della mia vita importanti e fondamentali. Un continuo confronto più o meno animato, ma senza il quale la mia vita sarebbe infinitamente più noiosa. Questo mio traguardo è anche merito vostro.

A Francesca, la persona che mi conosce meglio di quanto io conosca me stesso, che sin dal mio primo anno di università mi sopporta e supporta come nessun altro al mondo potrebbe fare, e senza la quale non sarei riuscito a portare a termine tutto questo. Grazie per tutto il tempo che mi dedichi. Grazie perché ci sei sempre. Grazie per avermi trasmesso la tua forza e la tua caparbietà nei momenti più difficili di questo percorso.

A Roberto, Maria, Sara e Pietro, i quali non mi hanno mai fatto mancare il loro sostegno e affetto incondizionato.

Al piccolo Edoardo, che con la sua semplice presenza e spontaneità è in grado di rendere splendidi anche i giorni più cupi.

Ad Andrea, con il quale ho condiviso tutto il mio percorso, non solo dal punto di vista accademico, ma anche, e soprattutto, dal punto di vista emotivo. Le nottate di studio, le ore al pc, i videogames, i progetti, gli esami, le pause, le ansie, le delusioni, le gioie e i sogni sono una minima parte di ciò che ci accomuna. Fiero di aver trovato in questi anni un grande amico prima che un grande collega.

Ai miei compagni del team Sapienza Corse e al Prof. Giovanni Battista Broggiato, con i quali ho condiviso alcune delle esperienze più entusiasmanti e formative di questo percorso e ai quali sarò eternamente riconoscente. Portare una monoposto formidabile in gara contro tutto (pandemia e fenomeni atmosferici estremi inclusi) e tutti, è un qualcosa che non si può esprimere a parole.

Ai miei amici di sempre, ultimi per menzione, ma non per importanza. Ci siamo conosciuti che eravamo praticamente bambini, e dopo tutto questo tempo, oggi ho la fortuna di festeggiare questo traguardo con voi. Nulla mi rende più orgoglioso di voi, la mia seconda famiglia. Si dice che l'amicizia vera resista al tempo, alla distanza e al silenzio. Voi siete l'esempio concreto che in qualsiasi momento avrò sempre una spalla su cui contare.

Infine ai nonni, a tutta la mia grande famiglia, agli amici, ai miei colleghi di università e a tutti coloro che non ho menzionato, che hanno incrociato la loro vita con la mia lasciandomi qualcosa di buono. Grazie per essere stati presenti, ognuno a suo modo, in questo percorso intenso ed entusiasmante, nel bene e nel male.

Grazie ad ognuno di voi per aver reso questo traguardo davvero speciale.

Contents

Chapter 1

Introduction

1.1 Motivation

The problem of jackknifing in tractor-trailer systems is a major challenge that affects a wide range of industries, such as agriculture, logistics, and transportation. Tractor-trailer systems are an essential component of these industries, as they are used for a variety of tasks such as transporting goods and materials, plowing fields, planting crops, and harvesting crops. With the advancement of technology, the field of automotive control has become increasingly vital in the development of more efficient, safe, and environmentally friendly vehicles. Therefore, the control of tractor-trailer systems has gained significant attention in recent years, with researchers focusing on developing effective control methods that can prevent the jackknife effect and improve the safety and efficiency of reversing these vehicles.

The "jackknife" effect is a term used to describe when the angle between a truck and its trailer, known as the "hitch" angle, becomes uncontrollable during reverse maneuvers. This can happen even at relatively low speeds and can lead to the trailer folding in on the driving vehicle. When the hitch angle reaches a certain point, the vehicle and the trailer can fold together around the hitch point like a jackknife, hence the name. This can make reversing the vehicle difficult and even dangerous, causing problems such as contact between the tractor and the trailer, wasted time, and in worst-case scenarios, injury or damage.

Reversing a tractor-trailer system, particularly when a second trailer is added, becomes more difficult. This task becomes even harder when reversing on a circular trajectory or on trajectories lacking constant curvature, such as polynomial trajectories, which are required for backward parking and avoiding obstacles.

This is a challenging task as a tractor-trailer vehicle has a complicated mechatronic system with strong coupling and high nonlinearity. Thus, preventing the jackknife effect requires appropriate control, and in particular, a closed-loop execution, which is a difficult task that uses nonlinear control theories. This is important for maintaining safety and preventing material damage.

To improve transportation, the development of effective control systems for tractor-trailer vehicles has the potential to bring significant advancements in other industries that rely on these vehicles. Having a proper control system in place can lead to major improvements in the efficiency and safety of these industries as well.

Therefore, this thesis aims to address the problem of jackknifing in tractor-trailer systems through the development and implementation of nonlinear Model Predictive Control (NMPC) techniques. NMPC employs a mathematical model of the system to make projections about the system's behavior. Based on these projections, it optimizes a control strategy to ensure the system operates within safe and efficient bounds. This control technique is particularly well suited to the control of tractor-trailer systems because it can handle the nonlinear dynamics and constraints of these systems, which are important factors in preventing jackknifing. Additionally, it can take into account multiple inputs and outputs of the system, which is crucial for this kind of model. Furthermore, NMPC has the capability to optimize the control strategy for a given time horizon, which can be set to anticipate and prevent jackknifing.

The research proposed in this thesis includes the development and implementation of nonlinear MPC control techniques for the aforementioned kinematic model, to improve safety and efficiency during reversing maneuvers by preventing the jackknife effect. Through the use of simulation and experimental validation, this research demonstrates the effectiveness of the proposed control methods in addressing the problem of jackknifing in tractor-trailer systems.

Furthermore, the proposed control methods have the potential to not only improve the safety and efficiency of reversing tractor-trailer systems but also contribute to the overall sustainability of transportation and other industries. The reduction of jackknifing incidents can lead to fewer accidents, less vehicle downtime, and reduced maintenance costs. Additionally, by increasing the efficiency of reversing tractor-trailer systems, this research has the potential to decrease fuel consumption and emissions, ultimately reducing the environmental impact of these vehicles.

In conclusion, this thesis aimed to address the problem of jackknifing in tractor-trailer systems through the development and implementation of nonlinear Model Predictive Control techniques. The research was conducted through simulation and demonstrated the effectiveness of the proposed control methods in addressing the problem of jackknifing and improving the safety and efficiency of reversing tractor-trailer systems.

Figure 1.1. Tractor-trailer in side jackknifing maneuver.

Figure 1.2. Tractor-trailer in jackknifing situation, causing road accident.

1.2 Literature review

The control of tractor-trailer vehicles has been widely studied in the literature, with different approaches being proposed to tackle this problem.

For example [25] focuses on the control aspect of autonomous vehicles and their ability to maneuver with agility. The author discusses the importance of agile maneuvering in transportation and the challenges faced, including the complexity of control systems and real-time decision-making. The article explores various control strategies, including model-based control, adaptive control, and reinforcement learning, and provides a detailed analysis of their advantages and limitations.

Generally, the two main categories of approaches are based on either a Cartesian space reference motion or a configuration space reference motion. The first category corresponds to output tracking, while the second category corresponds to state tracking.

In the first category of output tracking, various feedback control schemes have been proposed to drive the tractor-trailer vehicle along a specified path or trajectory. In [21], a feedback control scheme is proposed to drive a general one-trailer system along backward trajectories, using the control inputs of a virtual vehicle moving forward along the same trajectory. In [16], a two-level trajectory tracking controller is introduced for a zero-hooked one-trailer system. In [13], the trajectory tracking problem is considered for a general n-trailer vehicle, where the last trailer must track a linear or circular trajectory. More recently, [26] addresses the problem by defining the last trailer as a virtual tractor, and [22] proposes a curvature-based method for both forward and backward path following in the presence of sideslip.

In the second category of state tracking, the reference state path or trajectory corresponds to specific Cartesian motions, such as linear or circular motions. This includes works such as [31], where a zero-hooked one-trailer system is controlled via input-state linearization and time scale transformation, and [32] and [4], which tackle the general one-trailer system using linear and Lyapunov-based designs, respectively. In [9] and [2], path tracking is considered for the general two-trailers system.

Another subgroup of state-tracking approaches deals with generic reference paths or trajectories in configuration space. In [29], a low-level hitch angle controller is designed to simplify the general one-trailer system model and the associated control problem. In [24], a Linear Quadratic (LQ) controller is used to drive a general two-trailer system along a backward path generated by composing motion primitives. Additionally, Model Predictive Control has also been used in this area, as seen in works such as [20], [17], which only consider forward motions, and [23], which focuses on the case of the general two-trailers system.

For the problem of jackknifing and instability in the backward motion of a tractor-trailer system, several control strategies have been proposed to address this problem, including the use of Model Predictive Control (MPC) and nonlinear MPC (NMPC).

In general, these studies have shown that MPC and nonlinear MPC are effective control strategies for preventing instability issues and jackknifing in tractor-trailer systems in reverse motion. However, there is still ongoing research in this area, and more studies are needed to fully understand the effectiveness of these control strategies.

MPC is a control strategy that uses a model of the system to predict its future behaviour and optimize control inputs to achieve a desired performance while taking into account constraints on the control inputs and system states. To address the problem of jackknifing and instability in the backward motion of a tractor-trailer system, several MPC-based control strategies have been proposed in the literature.

For example, in [36] the authors proposed an MPC strategy for controlling the yaw angle and lateral position of a trailer during backward motion, taking into account constraints on the steering angle of the trailer wheels and the maximum braking force that can be applied. Additionally, this work proposed an online optimization algorithm that can handle real-time constraints and it has been implemented and tested successfully on a real-world prototype.

In [28], the authors proposed an MPC strategy for controlling the yaw angle of a tractor-trailer system in reverse motion and implemented it in real-time using a digital signal processor (DSP). The proposed control strategy was able to prevent instability issues and jackknife by predicting the future behaviour of the system and optimizing the control actions.

NMPC is a variant of MPC that can handle nonlinear systems. To address the problem of jackknifing and instability in the backward motion of a tractor-trailer system, several NMPC-based control strategies have been proposed in the literature.

For example, in [30], the authors proposed an NMPC strategy for controlling the yaw angle and lateral position of a trailer during backward motion, taking into account nonlinear dynamics and time-varying road surface and wind gust disturbances. The authors reported a significant improvement in the performance of the trailer yaw angle and lateral position control using this approach.

In [33], the authors proposed a nonlinear MPC strategy for controlling the yaw angle of a tractor-trailer system in reverse motion and tested it in simulation. The proposed control strategy was able to prevent instability issues and jackknife by predicting the future behaviour of the system and optimizing the control actions.

Additionally, [11] proposed a robust NMPC approach for the backward motion control of a tractor-trailer system, considering the effects of tire-road friction uncertainties. This work proposed a robust nonlinear model predictive controller that can handle the tire-road friction uncertainty by considering the estimated tire-road friction as a disturbance in the prediction model.

Another recent work [35] proposed a robust nonlinear MPC approach for the backward motion control of tractor-trailer systems, taking into account the effects of tire-road friction uncertainties.

Similarly, [38] proposed an anti-jackknifing control strategy based on robust nonlinear MPC for tractor-trailer systems, also handling tire-road friction uncertainties and wind disturbances.

Other works like [19] and [37] also proposed similar control strategies using nonlinear MPC.

Another important approach in the literature is based on the use of an intrinsically stable Model Predictive Control (IS-MPC) approach, in [8]. This method can guarantee the intrinsic stability of the closed-loop system without the need for a Lyapunov function, and it was shown to be effective through simulation and real-world experiments. The authors make the MPC intrinsically stable in the paper by incorporating anti-jackknifing control constraints into the MPC optimization

problem. This is achieved by modeling the trailer dynamics and considering the anti-jackknifing constraints as soft constraints in the optimization problem. The optimization problem is then solved using a quadratic programming algorithm, which provides a control input that satisfies the anti-jackknifing constraints while minimizing the cost function. The MPC approach is validated through simulations, demonstrating its ability to maintain stability while effectively avoiding jackknifing scenarios.

The present work can be considered a natural extension of the latter.

The control of tractor-trailer vehicles has been a hot topic in the literature, as the challenge of preventing jackknifing and instability during backward motion has been widely researched. A range of control strategies have been proposed to tackle this issue, with a focus on MPC and NMPC. These strategies vary in their approach, including the use of constraints on control inputs and system states, the incorporation of nonlinear dynamics and time-varying disturbances, and utilizing stable MPC and robust nonlinear MPC to handle tire-road friction uncertainties.

Though most studies concentrate on the control of the yaw angle and traction, other studies delve into the control of lateral and longitudinal motion. Recently, robust MPC and adaptive MPC have been introduced as potential solutions, offering improved adaptability and robustness to different operating conditions and uncertainties.

It is important to note that most of these studies have only been validated in simulation, calling for further validation in real-world scenarios.

In conclusion, controlling tractor-trailer vehicles has been thoroughly explored, with a plethora of techniques and approaches put forward to solve the jackknifing and instability problem. These approaches can be categorized into two groups based on the reference motion used, and provide a comprehensive understanding of the challenges and solutions in controlling tractor-trailer vehicles.

Chapter 2

Modeling of wheeled vehicles

The aim of this chapter is to highlight the various models that are commonly utilized for the motion planning and feedback control of wheeled vehicles during low-speed maneuvers. The chapter begins with a concise explanation of nonholonomic systems, followed by the presentation of the kinematic vehicle model in Section 2.2.

Modeling of wheeled vehicles has been a longstanding practice and the choice of modeling techniques varies based on the specific application. A vehicle model can either be dynamic, which is derived from force balances, or kinematic, which is derived from velocity constraints known as nonholonomic constraints.

In the case of wheeled vehicles, these constraints arise from the assumption of wheels rolling without slipping. During low-speed movements on dry road conditions, a kinematic model is frequently sufficient to describe the vehicle's behaviour, while during high-speed maneuvers a dynamic model is necessary to capture the effects. This chapter focuses on the basics of modeling the kinematic properties of the vehicle, which is a popular approach in motion planning and control.

The reason for using kinematic models with lower state dimensions, instead of advanced models with higher fidelity, is because the latter would result in a high dimensional state-space, which may not provide real-time performance in practical applications. Nevertheless, advanced models are often preferred for simulation purposes.

2.1 Nonhonolomic systems

A kinematic model of a wheeled vehicle is established based on the premise that the wheels roll without slipping. This leads to the conclusion that there are certain velocity directions in which the vehicle cannot move. The system that is subject to such velocity restrictions is referred to as nonholonomic. The configuration of the vehicle is represented by the vector $q \in \mathbb{R}^n$ and is referred to as the configuration space $\mathcal{C}$ [1], n-dimesional, which is assumed to be a smooth manifold of all possible configurations.

The constraints on the velocities of the system are expressed in the form of

[1] A space where each neighbourhood of a point is homeomorphic to a neighbourhood of $\mathbb{R}^n$, in this case. A correct representation of $\mathcal{C}$ coincides with a torus, a surface of revolution generated by revolving a circle in three-dimensional space, defined as a manifold that is not a Euclidean space.

$$a(q, \dot{q}) = 0, \tag{2.1}$$

for example, a car cannot move sideways. These are defined as kinematic constraints.

In some cases, the velocity constraints can be explicitly integrated, resulting in geometrical constraints on the configuration of the vehicle, referred to as holonomic constraints and expressed as

$$h(q) = 0. \tag{2.2}$$

In contrast, when a velocity constraint cannot be explicitly integrated, it is considered a nonholonomic constraint. Different holonomic and nonholonomic constraints arise naturally due to physical limitations of the system, such as configuration inequality constraints $h(q) \leq 0$ and velocity inequality constraint $h(\dot{q}) \leq 0$. These are constraints on the configuration q and the velocity $\dot{q}$ of the system, and for example, they can be demonstrated by limited steering angle and steering angle rate for a car. The focus of this chapter is on nonholonomic systems with linear velocity constraints, which are represented by

$$a_i^T(q)\dot{q} = 0 \quad i = 1, \ldots, k < n, \tag{2.3}$$

where $a_i(q) \in R^{n\times 1}$, therefore typically linear in $\dot{q}$, and the vector fields of the system must be orthogonal to each $a_i(q)$. These kinematic constraints can be rearranged, in order to obtain the so-called Pfaffian form:

$$\begin{pmatrix} a_1^T(q) \\ a_2^T(q) \\ \vdots \\ a_k^T(q) \end{pmatrix} \dot{q} = A^T(q)\dot{q} = 0, \tag{2.4}$$

with $A \in \mathbb{R}^{k\times n}$.

By transforming the constraints in (2.3) to explicit form $\dot{q} = f(q, u)$, defined $\{g_1(q), g_2(q), \ldots, g_m(q)\}$ a basis of $\mathcal{N}(A^T(q))$ with $m = n - k > 0$, and assigning each $g_j(q) \in R^n$ with a control signal $u_j \in \mathbb{R}^m$ (a scalar coefficient), a control-affine driftless system is established with the equation

$$\dot{q} = \sum_{j=1}^{m} g_j(q)u_j \iff \dot{q} \in \mathcal{N}(A^T(q)). \tag{2.5}$$

Nonholonomic systems are typically underactuated, meaning the number of control signals is less than the dimension of the configuration space ($m < n$). The nonholonomic system can be modeled as a nonlinear system $\dot{x} = f(x, u)$, where the state vector x is a subset of the configuration q, or additional properties are also considered. The degree of freedom for the system remains unchanged. Models of nonholonomic systems that are relevant to this thesis are derived and discussed in detail in the next sections.

2.2 Kinematic tractor-trailer model

The kinematic bicycle model is widely utilized in the field of motion planning and control and is particularly applicable to low-speed vehicle maneuvering. The model, as illustrated in Figure 2.1, depicts a vehicle operating on a flat surface with front-wheel Ackerman steering, where the inner front wheel turns more than the outer front wheel during cornering. The kinematic bicycle model is derived based on the assumption that both front and rear wheels are rolling without slipping.

However, the use of this model requires certain assumptions in addition, in order to simplify and make it feasible for analysis. These assumptions include:

- Rigid bodies: both the tractor and trailer are modeled as rigid bodies, disregarding any potential deformation under load or impact.
- Point mass: the trailer is modeled as a point mass, ignoring any dimensions in any direction.
- Flat and smooth road: as mentioned earlier, the road is assumed to be flat and smooth without any bumps or obstacles.
- Constant velocity: the linear velocity of the tractor is assumed to be constant, with any changes occurring instantly.
- Constant steering velocity: the steering velocity is assumed to be constant, with any changes occurring instantly.
- Ignored actuators: the dynamics of the actuators responsible for wheel movement or trailer angle adjustment are not considered.
- No external forces: only the kinematics of the system are considered, neglecting external forces such as friction, tire-road interaction, and wind.
- No sensor noise: the presence of measurement noise from sensors is not accounted for in the model.

It should be noted that these assumptions simplify the model, but do not accurately reflect real-world systems. To improve the accuracy for real-world scenarios, more advanced models that incorporate these factors may need to be developed.

The tractor-trailer system under consideration in this study is the one with off-axle hitching, as depicted in Figure 2.1. The system configuration can be represented by the generalized coordinates

$$q = \begin{pmatrix} x \\ y \\ \theta \\ \psi \\ \phi \end{pmatrix}, \tag{2.6}$$

and is referred to as the configuration space $\mathcal{C} = \mathbb{R}^2 \times SO(2) \times SO(2) \times SO(2)$, where $SO(2)$ is the special orthogonal group of 2D rotations.

These coordinates are chosen for their simplicity, as the assumption of collapsing the two wheels on each axle into a single wheel located at the midpoint of the axle

can be justified by the presence of a differential on the driving wheels and a steering system with the aforementioned Ackermann geometry in most vehicles of this type. The Cartesian coordinates of the contact point between the tractor's rear wheel and the ground referred to as (x, y), represent the center of the tractor's rear wheel. θ represents the tractor heading, while ψ is defined as the difference between the trailer heading and the tractor heading, also known as the hitch angle. Finally, ϕ represents the steering angle of the front wheel with respect to the vehicle orientation.

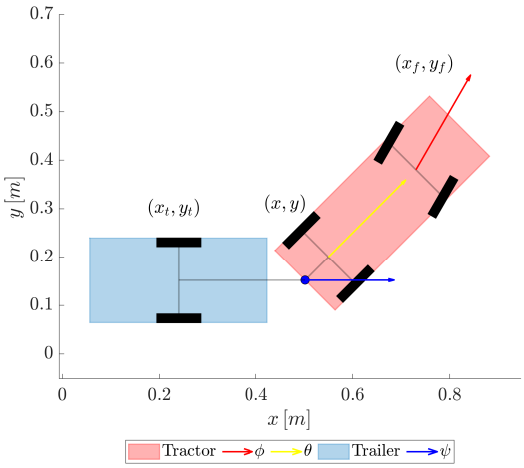

Figure 2.1. Tractor-trailer kinematic model.

Given that the vehicle moves on a flat plane and is traveling slowly enough to avoid wheel slip, its motion is subject to three pure rolling without slipping (nonholonomic) constraints, one for each wheel of the bicycle model plus the trailer. These constraints have been expressed mathematically as follows.

It has been determined that the speed of the center of the tractor's front wheel is null in the direction normal to the straight line oriented as the wheel itself, as expressed by the equation:

$$\dot{x_f} \sin(\theta + \phi) - \dot{y_f} \cos(\theta + \phi) = 0, \tag{2.7}$$

where (x_f, y_f) are the Cartesian coordinates of the center of the tractor front wheel. Equivalently, given L, the distance between the front and the rear axle of the tractor, the previous equation can be expressed as:

$$\dot{x} \sin(\theta + \phi) - \dot{y} \cos(\theta + \phi) - \dot{\theta} L \cos\phi = 0. \tag{2.8}$$

Similarly, it has been established that the speed of the center of the tractor's rear wheel is null in the direction normal to the sagittal axis of the tractor, as described by the equation:

$$\dot{x} \sin\theta - \dot{y} \cos\theta = 0. \tag{2.9}$$

Finally, it has been concluded that the speed of the center of the trailer wheel is null in the direction normal to the sagittal axis of the trailer, as expressed by the equation:

$$\dot{x_t} \sin\theta_1 - \dot{y_t} \cos\theta_1 = 0, \tag{2.10}$$

where (x_t, y_t) are the Cartesian coordinates of the center of the trailer wheel, and θ_1 is the absolute orientation of the trailer. Or equivalently, given L_1, the distance between the rear axle of the tractor and the hitch joint, and L_2, the distance between the hitch joint and the trailer axle, the latter can be defined as:

$$\dot{x} \sin\theta_1 - \dot{y} \cos\theta_1 + \dot{\theta}(L_1 \cos\psi + L_2) + \dot{\psi} L_2 = 0. \tag{2.11}$$

These three constraints ensure that the absence of slippages is guaranteed by the previously made assumptions.

The constraints, represented in Pfaffian form, are linear with respect to the generalized velocities and can be reorganized as

$$A^T(q)\dot{q} = \begin{pmatrix} \sin\theta & -\cos\theta & 0 & 0 & 0 \\ \sin(\theta+\phi) & -\cos(\theta+\phi) & -L\cos\phi & 0 & 0 \\ \sin\theta_1 & -\cos\theta_1 & L_1\cos\psi + L_2 & L_2 & 0 \end{pmatrix} \dot{q} = 0. \tag{2.12}$$

This means that the generalized velocities are limited to the kernel of $A^T(q)$. Since $A^T(q)$ has a rank of $k = 3$, the dimension of the $ker(A^T(q))$ is $n - k = 2$, and all possible generalized velocities can be represented as linear combinations of one of its bases.

For example, the following are such bases:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \\ \frac{\tan\phi}{L} \\ -\frac{\tan\phi}{L}\left(\frac{L_1}{L_2}\cos\psi + 1\right) - \frac{\sin\psi}{L_2} \\ 0 \end{pmatrix} u_1 + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} u_2. \tag{2.13}$$

By defining u_1 as the traction speed of the tractor's rear wheel ($u_1 = v$) and u_2 as the steering velocity ($u_1 = \omega$), the behaviour of the vehicle can be described through the following system of equations:

$$\begin{aligned} \dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \frac{v\tan\phi}{L} \\ \dot{\psi} &= -\frac{v\tan\phi}{L}\left(\frac{L_1}{L_2}\cos\psi + 1\right) - \frac{v\sin\psi}{L_2} \\ \dot{\phi} &= \omega. \end{aligned} \tag{2.14}$$

These equations represent the motion of the vehicle in terms of its position and orientation, as well as its traction speed and steering velocity.

2.2.1 Double-trailer model

The process of deriving the kinematic model of a tractor-trailer involves determining the relationships between the various parameters that govern the motion of the system. With this understanding, it is now possible to use the same procedure to derive the kinematic model of a tractor with two trailers. This involves considering the additional trailer and accounting for its effect on the overall motion of the system. By taking into account the various parameters such as the position and velocity of each component, it is possible to create a mathematical model that accurately describes the motion of the tractor with double trailer.

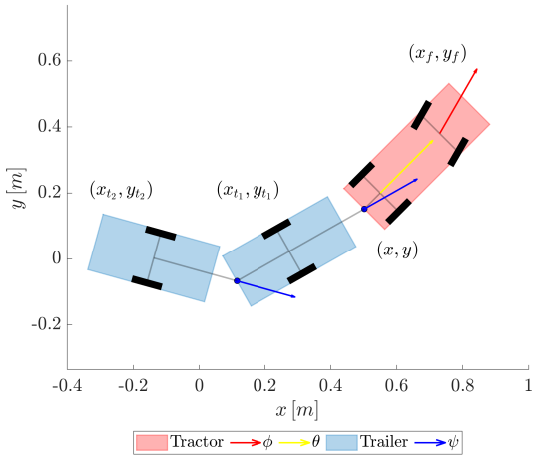

Figure 2.2. Double-trailer kinematic model.

Given that the vehicle moves on a flat plane and is traveling slowly enough to avoid wheel slip, its motion is subject to four pure rolling without slipping (nonholonomic) constraints, one for each wheel of the bicycle model plus the two trailers. These constraints have been expressed mathematically as follows:

$$\begin{aligned} &\dot{x_f} \sin(\theta + \phi) - \dot{y_f} \cos(\theta + \phi) = 0 \\ &\dot{x} \sin\theta - \dot{y} \cos\theta = 0 \\ &\dot{x_{t_1}} \sin(\theta + \psi_1) - \dot{y_{t_1}} \cos(\theta + \psi_1) = 0 \\ &\dot{x_{t_2}} \sin(\theta + \psi_1 + \psi_2) - \dot{y_{t_2}} \cos(\theta + \psi_1 + \psi_2) = 0. \end{aligned} \tag{2.15}$$

The new fourth constraint, with respect to the single-trailer model, refers in fact to the second trailer. (x_{t_2}, y_{t_2}) are the Cartesian coordinates of the center of the second trailer wheel, and ψ_2 is defined as the difference between the second trailer heading and ψ_1, the hitch angle of the first trailer.

Again it may be useful to express the whole set of constraints as an expression of (x, y). In this regard, the following equations hold:

$$
\begin{aligned}
x_f &= x + L\cos\theta \\
y_f &= y + L\sin\theta \\
x_{t_1} &= x - L_1\cos\theta - L_2\cos(\theta+\psi_1) \\
y_{t_1} &= y - L_1\sin\theta - L_2\sin(\theta+\psi_1) \\
x_{t_2} &= x - L_1\cos\theta - (L_2+L_3)\cos(\theta+\psi_1) - L_4\cos(\theta+\psi_1+\psi_2) \\
y_{t_2} &= y - L_1\sin\theta - (L_2+L_3)\sin(\theta+\psi_1) - L_4\sin(\theta+\psi_1+\psi_2)
\end{aligned}
\quad , \tag{2.16}
$$

given the new L_3, the distance between the rear axle of the first trailer and its hitch joint, and L_4, the distance between the second hitch joint and the second trailer axle.

The pure rolling constraints (2.15) can be re-expressed as:

$$
\begin{aligned}
&\dot{x}\sin(\theta+\phi) - \dot{y}\cos(\theta+\phi) - \dot{\theta}L\cos\phi = 0 \\
&\dot{x}\sin\theta - \dot{y}\cos\theta = 0 \\
&\dot{x}\sin(\theta+\psi_1) - \dot{y}\cos(\theta+\psi_1) + \dot{\theta}(L_1\cos\psi_1 + L_2) + \dot{\psi_1}L_2 = 0 \\
&\dot{x}\sin(\theta+\psi_1+\psi_2) - \dot{y}\cos(\theta+\psi_1+\psi_2) + \quad \dots \\
&\dots \quad + (\dot{\theta}+\dot{\psi_1}+\dot{\psi_2})L_4 + (\dot{\theta}+\dot{\psi_1})(L_2+L_3) + \dot{\theta}(L_1\cos\psi_1 + L_2) + \dot{\psi_1}L_2 = 0.
\end{aligned}
\tag{2.17}
$$

These equations are in Pfaffian form, i.e. linear in the generalized velocities $A^T(q)\dot{q} = 0$, where the matrix $A(q)$ is composed by:

$$
A(q) = \begin{pmatrix}
\sin\theta & \sin(\theta+\phi) & \sin(\theta+\psi_1) & \sin(\theta+\psi_1+\psi_2) \\
-\cos\theta & -\cos(\theta+\phi) & -\cos(\theta+\psi_1) & -\cos(\theta+\psi_1+\psi_2) \\
0 & -L\cos\phi & L_1\cos\psi_1 + L_2 & L_1\cos(\psi_1+\psi_2) + L_2 + L_3 + L_4 \\
0 & 0 & L_2 & L_2 + L_3 + L_4 \\
0 & 0 & 0 & L_4 \\
0 & 0 & 0 & 0
\end{pmatrix}. \tag{2.18}
$$

This means that the generalized velocities are limited to the kernel of $A^T(q)$. Since $A^T(q)$ has a rank of $k = 4$, the dimension of the $ker(A^T(q))$ is $n - k = 2$, and all possible generalized velocities can be represented as linear combinations of one of its bases.

As done previously ,by defining u_1 as the traction speed of the tractor's rear wheel ($u_1 = v$) and u_2 as the steering velocity ($u_1 = \omega$), the behaviour of the double-trailer vehicle can be described through the following system of equations:

$$
\begin{aligned}
\dot{x} &= v\cos\theta \\
\dot{y} &= v\sin\theta \\
\dot{\theta} &= \frac{v\tan\phi}{L} \\
\dot{\psi}_1 &= -\frac{v\tan\phi}{L}\left(\frac{L_1}{L_2}\cos\psi + 1\right) - \frac{v\sin\psi}{L_2} \\
\dot{\psi}_2 &= \frac{v\tan\phi}{LL_4}\left(\frac{(L_2+L_3)\cos\psi_1\cos\psi_2}{L_2} + \frac{L_4\cos\psi_1}{L_2} - \cos\left(\psi_1+\psi_2\right)\right) + \quad \dots \\
&\dots \quad + \frac{v\sin\psi_1}{L_2}\left(\frac{(L_2+L_3)\cos\psi_2}{L_4} + 1\right) - \frac{v\sin\left(\psi_1+\psi_2\right)}{L_4} \\
\dot{\phi} &= \omega.
\end{aligned}
\tag{2.19}
$$

The next sections concerning the formulation of the problem and the implementation of the proposed approach will only deal with the case of the single trailer for simplicity and compactness.

Therefore, any subsequent consideration made for the vehicle with a single trailer can also be automatically extended to the case of the double trailer with the necessary adaptations.

Chapter 3

Problem formulation

As extensively covered in Chapter 1, because of control instability, backward motions are treated notably in literature.

Theoretically, the velocity vector components throughout the connections in the articulated system should be consistent as the system is physically connected. This consistency can be seen in Figure 3.1, where the limits on the tractor's steering angle (ϕ) and the trailer's hitch angle (ψ) are upheld. However, Figure 3.2 demonstrates a case of inconsistency leading to jackknifing, which violates the bounds on the ψ angle and can result in damage to the link or sideslipping of the trailer.

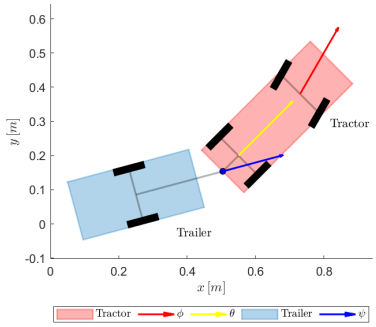

Figure 3.1. Tractor-trailer in backward motion, respecting bounds on the angles ϕ and ψ.

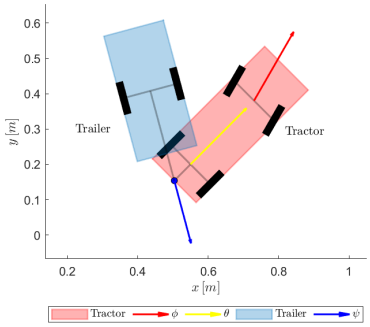

Figure 3.2. Tractor-trailer in a jackknifing situation, violating bounds on the angle ψ.

To address this issue, a control technique is presented in the chapter. The purpose of the feedback controller is to minimize control error and ensure that the controlled vehicle follows the computed motion plan, even in the presence of disturbances and initial control errors. The feedback control problem is referred to as a trajectory tracking control problem, with the reference obtained from the motion planner.

It is assumed that the vehicle is modeled as a time-invariant nonlinear system and that all vehicle states are perfectly measured [1] and the system is controllable around

[1] In practice, vehicle state is obtained through onboard sensors that collect data, which is then processed using filtering techniques to produce a state estimate $\hat{x}(t)$, providing a precise representation of the vehicle's current state for safe navigation.

the nominal trajectory. Additionally, the motion plan is assumed to be feasible and not in conflict with surrounding obstacles, satisfying the system dynamics. The theory presented in the chapter is in continuous-time first, but it has a discrete-time counterpart for implementation purposes presented as well.

3.1 Trajectory tracking control

The function of the feedback controller is to maintain the stability of the vehicle as it follows a trajectory determined by a motion planner. To achieve this, a feedback control law is employed. The control task involves the creation of a closed-loop system between the vehicle and the trajectory tracking controller, ensuring that the vehicle executes the predetermined nominal trajectory $(x^r(t), u^r(t))$ with minimal tracking error, represented by $\bar{x}(t) = x(t) - x^r(t)$.

The reference trajectory is comprised of nominal vehicle states and controls $(x^r(t), u^r(t))$ that are parametrized over time. The objective of the trajectory tracking controller is to keep the vehicle close to the nominal state trajectory. This nominal trajectory, which is determined in advance, is independent of the vehicle's current state $(x(t))$.

It is important to note that point stabilization is a subcategory of trajectory tracking in which the nominal trajectory is limited to a single point.

In control systems, it is often desirable for a system to be regulated to follow a reference trajectory. One commonly utilized approach for achieving this is to change the coordinates so that the new coordinate system is based on the error between the system and the reference trajectory, also known as error coordinates or error transformations.

The error signal, obtained by subtracting the reference trajectory from the nonlinear model, represents the deviation of the system from the reference. This error signal can then be utilized to design a control law that drives the system toward the reference.

The utilization of error coordinates simplifies the control design process. For instance, if the error signal is small, linear control techniques can be employed to design a controller for the system as the linearization of the nonlinear model around the reference trajectory leads to a linear system in the error coordinates. Furthermore, in the error coordinates, the steady-state error for a step input is zero, ensuring that the system tracks the reference trajectory exactly.

If the nonlinear model is not linearized and normal coordinates are utilized without performing error transformations, the control design process becomes more complicated. The nonlinearities in the system have to be taken into account, leading to steady-state errors and making it harder to design a control law that regulates the system accurately to the reference trajectory. In such a case, the control law has to compensate for both the nonlinearities of the system and its deviation from the reference trajectory, presenting a challenging design process. The steady-state error for a step input will not be zero, resulting in the system not tracking the reference exactly.

In conclusion, the search for stability constraints to address nonlinearities, tracking errors, and other arising issues is the aim of this thesis. The utilization of error

coordinates simplifies the control design process and results in zero steady-state error, making it easier to regulate a nonlinear system to a reference trajectory. On the other hand, not linearizing the nonlinear model and working in normal coordinates without error transformations will result in a more complex control

3.1.1 Background on optimization problems

The characterization of an optimization problem is composed of three essential components. The first component is the objective function, represented by $\Phi(\mathbf{w})$, which needs to be either minimized or maximized. The second component involves decision variables, $\mathbf{w}$, that can be chosen by the decision-maker. Finally, constraints that must be respected, either in the form of equality constraints (e.g. $\mathbf{g}_1(\mathbf{w}) = 0$) or inequality constraints (e.g. $\mathbf{g}_2(\mathbf{w}) \leq 0$), make up the third component.

The standard form of a mathematical formulation in optimization problems is known as a Nonlinear Programming Problem (NLP). This standard form is represented as follows:

$$\begin{aligned} \min_{\mathbf{w}} \Phi(\mathbf{w}) \quad & \text{Objective function} \\ \text{s.t. } \mathbf{g}_1(\mathbf{w}) \leq 0, \quad & \text{Inequality constraints} \\ \mathbf{g}_2(\mathbf{w}) = 0. \quad & \text{Equality constraints} \end{aligned} \tag{3.1}$$

It is commonly assumed that the functions $\phi(\cdot), \mathbf{g}_1(\cdot)$, and $\mathbf{g}_2(\cdot)$ are differentiable. There are various special cases of NLP, such as Linear Programming (LP) when the functions are affine, i.e. these functions can be expressed as linear combinations of the elements of $\mathbf{w}$, Quadratic Programming (QP) when the constraints are affine and the objective function is a linear-quadratic function, etc.

The solution to the optimization problem involves finding the value of the decision variable, $\mathbf{w}$, that minimizes the objective function. The value of $\mathbf{w}$ that satisfies this condition is represented by $\mathbf{w}^* = \arg\min_{\mathbf{w}} \Phi(\mathbf{w})$. By direct substitution, the corresponding value of the objective function can be obtained and represented as $\Phi(\mathbf{w}^*) := \Phi(\mathbf{w})|_{\mathbf{w}^*} := \min_{\mathbf{w}} \Phi(\mathbf{w})$.

3.1.2 Model Predictive Control

Model Predictive Control or Receding/Moving Horizon Control, is a control strategy that involves formulating a dynamic optimization problem and solving it online at each time step to compute the control action for the current time step. In MPC, the controller predicts the future behaviour of the system based on a mathematical model of the system and optimization criteria specified by the user, and uses this prediction to compute a control action that drives the system towards a desired goal.

Nonlinear MPC is a variant of MPC that is used to control systems with nonlinear dynamics. These systems may include nonlinearities such as saturation, dead-zone, hysteresis, and backlash, among others. Nonlinear MPC algorithms are designed to handle such nonlinearities and achieve good performance in the presence of these types of system behaviours.

To implement nonlinear MPC, the controller must have a model of the system dynamics that accurately captures the nonlinearities present in the system. This

model is used to predict the future behaviour of the system and compute the control action. In addition to the model, the controller also requires information about the current state of the system and the desired goal. Based on this information, the controller formulates an optimization problem that minimizes a cost function over a finite horizon of future time steps, subject to constraints on the control action and the system dynamics. The optimization problem is solved online at each time step to compute the control action for the current time step, and the process is repeated at each subsequent time step.

Continuous-time nonlinear MPC problem

Nonlinear MPC has been successfully applied to a wide range of control problems, including process control, robotics, and aircraft control. It has the advantage of being able to handle complex system dynamics and constraints, and of being able to explicitly incorporate optimization criteria specified by the user. However, nonlinear MPC can be computationally intensive and may require the use of specialized software and hardware to solve the optimization problem in real-time.

The use of online optimization techniques to solve an Optimal Control Problem (OCP) at a specified sampling rate has been deemed one of the most powerful tools for designing feedback controllers. Hard-to-account physically imposed constraints on the states and controls during control design have been a challenge in previously presented control strategies (Section 1.2). This is particularly evident during moderate driving conditions, where the nominal trajectory may be feasible. However, if the motion planner relies on a simplified vehicle model, it may not result in a feasible motion plan, especially during slippery road conditions or emergency maneuvers, which require a higher-fidelity vehicle model.

To address this issue, MPC has emerged as a general control design methodology that can be highly effective. Like a finite horizon Linear Quadratic (LQ) control problem, MPC utilizes a model of the vehicle to plan its trajectory in a receding horizon fashion, with the ability to take into account the physical limitations of the vehicle using a nonlinear model.

MPC works by computing the control signal through a continuous resolution of the receding horizon OCP, using the most up-to-date state information, at every time step Δt. The MPC control law is then derived by solving the following continuous-time nonlinear MPC problem, with the first portion of the control signal being applied to the vehicle to provide feedback properties:

$$\begin{aligned} \underset{u(\cdot)}{\text{minimize}} \quad & \Phi_T(\bar{x}(t+T)) + \int_t^{t+T} (||\bar{x}(\tau)||_Q^2 + ||\bar{u}(\tau)||_R^2) d\tau \\ \text{subject to} \quad & \dot{x}(\tau) = f(x(\tau), u(\tau)), \\ & u(\tau) \in \mathbb{U}, \quad x(\tau) \in \mathbb{X}, \\ & x(t) = x_0, \end{aligned} \tag{3.2}$$

where T is the prediction horizon, $\Phi_T(\bar{x}(t+T)) \geq 0$ is the terminal cost which could not be present, $Q > 0$ and $R > 0$ are weighting matrices used to evaluate the cost index.

The issue presented in equation (3.2) pertains to a nonlinear optimal control problem, which has proven to be a challenging task to solve with global optimality. Nevertheless, advancements in technology have produced efficient software capable of providing approximate solutions in real-time for the NMPC controller. The implementation and usage of this software will be expanded upon in subsequent sections.

Discrete-time nonlinear MPC problem

Discrete-time Model Predictive Control, as shown in Figure 3.3, is a control strategy that aims to control systems by making predictions and implementing optimized control actions based on those predictions. It operates in discrete time intervals, called decision instants, and works recursively, repeating the same process at each decision instant.

The MPC strategy starts by measuring the current state of the system, represented by $x(k)$, at decision instant k. Based on this measurement, a prediction is made of the future states of the system over a predefined horizon, N, using the equation $x(k+1) = f(x(k), u(k))$. The prediction is then used to compute the optimal sequence of control actions over the prediction horizon, represented by $u^*(x(k)) := (u^*(k), u^*(k+1), \dots u^*(k+N-1))$.

After the optimal sequence of control actions has been computed, the first control action in the sequence, $u^*(k)$, is applied during the current sampling period $[k, k+1]$. At the next decision instant, the same process is repeated.

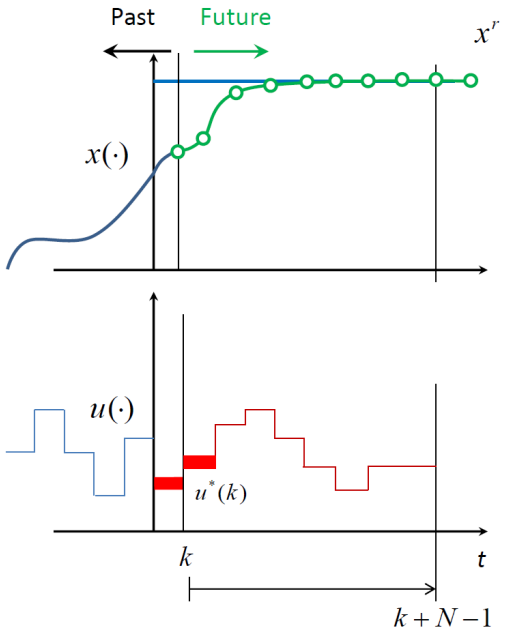

Figure 3.3. Model Predictive Control strategy.

In summary, the MPC strategy consists of three steps: prediction, online optimization, and receding horizon implementation. The prediction step involves making predictions of the future states of the system based on the current state and control

actions. The online optimization step involves computing the optimal sequence of control actions over the prediction horizon. The receding horizon implementation step involves applying the first control action in the optimized sequence during the current sampling period and repeating the process at the next decision instant.

Therefore, this technique is based on the optimization of a cost function, which evaluates the running costs along the entire prediction horizon.

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & J_N(x,u) = \sum_{k=0}^{N-1} \mathcal{L}(x(k),u(k)) \\ \text{subject to} \quad & x(k+1) = f(x(k),u(k)), \\ & u(k) \in \mathbb{U}, \quad \forall k \in [0, N-1], \\ & x(k) \in \mathbb{X}, \quad \forall k \in [0, N], \\ & x(0) = x_0. \end{aligned} \tag{3.3}$$

The running costs are characterized by the control objective, which is represented by the equation $\mathcal{L}(x,u) = ||x - x^r||_Q^2 + ||u - u^r||_R^2$. This equation calculates the difference between the current state and the desired state, and the current control input and the desired control input.

The cost function, $J_N(x,u)$, is a summation of the running costs over the entire prediction horizon.

The Optimal Control Problem (OCP) is defined as finding a control sequence that minimizes the cost function. It also includes constraints on the control input, $u(k)$, and the state, $x(k)$, which must be within the defined set of feasible inputs and states, $\mathbb{U}$ and $\mathbb{X}$, respectively. The OCP also requires that the initial state, x_0, be defined.

3.1.3 Computational aspects of nonlinear MPC

Nonlinear Model Predictive Control is a widely used control strategy in the industry, despite its computational challenges. This method requires the online solution of a nonlinear OCP, which is a more complex and computationally expensive task than solving a linear MPC problem. In the case of linear MPC, the solution of the optimal control problem can be expressed as the solution of a convex quadratic program and can be efficiently solved online.

NMPC, on the other hand, involves solving a nonlinear program, which does not have the same computational advantages as linear MPC. However, it has been observed that the nonlinear program associated with NMPC has a special structure that can be exploited to achieve a real-time feasible solution. In this case, it is assumed that the control and prediction horizon are the same, and no final region constraint is present, to simplify the discussion.

Therefore, the Optimal Control Problem (OCP) is a crucial component in the Nonlinear Model Predictive Control (NMPC) online optimization problem. The first step in solving this problem involves transforming the OCP into a Nonlinear Programming Problem (NLP). NLP is a commonly used formulation in numerical optimization and is well represented by equation (3.1) in the general form of an optimization problem.

There are several NLP optimization solvers available, each with its strengths and weaknesses. For instance, one of the popular solvers is IPOPT, which is utilized in this particular study. Another well-known NLP optimization solver is `fmincon`.

It is important to note that the choice of NLP optimization algorithm depends on the specific requirements and constraints of the problem being solved. However, regardless of the algorithm selected, the transformation of the OCP into an NLP represents the crucial first step in the NMPC optimization problem.

The simultaneous approach is a common method used to solve the NMPC optimization problem. In this approach, the system dynamics at each sampling point are incorporated into the optimization problem as nonlinear constraints. This means that at every sampling point, a new equality constraint must be satisfied. The use of nonlinear constraints allows for a more flexible and accurate representation of the system dynamics, making it a popular approach for solving the NMPC optimization problem.

This approach coincides with the multiple shooting technique. The shooting methods are in fact used in the transformation of an OCP into an NLP. The multiple shooting method is used in this thesis as a further development of the single shooting method. For completeness, both methods will be illustrated, clarifying the reason for this choice.

Shooting methods

In the single shooting method, the OCP can be described as the NLP (3.1) by defining as problem decision variables $\mathbf{w} = [u_0, \dots, u_{N-1}]$ and setting $x(\cdot)$ as a function of $\mathbf{w}$, x_0, and t_k:

$$\begin{aligned} x(\cdot) &= F(\mathbf{w}, x_0, t_k) \\ x_0 &= F(\mathbf{w}, x_0, t_0), \end{aligned} \tag{3.4}$$

then solving the NLP discretized only in control u:

$$\begin{aligned} &\min_{\mathbf{w}} \Phi(F(\mathbf{w}, x_0, t_k), \mathbf{w}) \\ &\text{s.t. } \mathbf{g}_1(F(\mathbf{w}, x_0, t_k), \mathbf{w}) \leq 0, \end{aligned} \tag{3.5}$$

therefore single shooting involves finding the minimum value of a function that satisfies certain constraints. Inequality constraints, such as map margins and control limits, are included in the optimization problem. Equality constraints, on the other hand, are not used in this method.

However, single shooting has a major drawback: nonlinearity propagation. The integrator function, which is central to this optimization method, tends to become highly nonlinear when dealing with large values of N. This nonlinearity can result in inaccurate solutions and make the method unsuitable for nonlinear and/or unstable systems. When optimizing over a long prediction horizon, the nonlinearity of the integrator function can become even more pronounced, leading to significant errors in the results.

In conclusion, single shooting is a useful method for solving optimization problems with inequality constraints, but it should be used with caution in cases where the system is nonlinear and/or unstable, and when optimizing over a long prediction

horizon. If the system is nonlinear and/or unstable, it is recommended to consider other optimization methods that may be better suited to the problem at hand.

Therefore the multiple shooting method has been used to transform the OCP into an NLP.

As introduced previously, the key idea is to break down the system integration into shorter time intervals, i.e. use the system model as a state constraint at each optimization step.

The technique of multiple shooting involves a process of lifting, which entails modifying a function that has multiple variables in order to reduce its level of nonlinearity.

Compared to single shooting, multiple shooting is considered to be a more effective method due to its ability to convert a problem into a higher dimension, which has been found to improve convergence.

Additionally, multiple shooting has the option to initialize the state trajectory with a known guess.

One downside, however, is that the problem that is solved using this method is often larger, although this drawback is typically offset by the fact that it is also more sparsely populated.

In this case, also x become decision variables in the optimization problem as well as u, obtaining $\mathbf{w} = [u_0, \dots u_{N-1}, x_0, \dots, x_N]$, and additional path constraints are applied at each optimization step, i.e. $x(k+1) - f(x(k), u(k)) = 0$.

Therefore the NLP can be expressed as:

$$\begin{aligned} &\min_{\mathbf{w}} \Phi(\mathbf{w}) \\ \text{s.t. } &\mathbf{g}_1(\mathbf{w}) = \begin{pmatrix} g_1(x_0, u_0) \\ \vdots \\ g_1(x_{N-1}, u_{N-1}) \\ g_1(x_N) \end{pmatrix} \leq 0 \\ &\mathbf{g}_2(\mathbf{w}) = \begin{pmatrix} \bar{x_0} - x_0 \\ f(x_0, u_0) - x_1 \\ \vdots \\ f(x_{N-1}, u_{N-1}) - x_N \end{pmatrix} = 0, \end{aligned} \tag{3.6}$$

so multiple shooting involves finding the minimum value of a function that satisfies certain constraints. Inequality constraints, such as map margins and control limits, are included in the optimization problem. Equality constraints, present in this case, involve the system dynamics.

Multiple shooting is a technique used in Model Predictive Control (MPC) to improve the computational efficiency of the optimization problem that must be solved at each time step. In a single shooting technique, the entire optimization problem for the entire horizon is solved at once. This can be computationally expensive for long horizons or systems with high dimensional state space.

In contrast, multiple shooting divides the horizon into several smaller intervals and solves a separate optimization problem for each interval. This can be much more computationally efficient, as the size of each optimization problem is smaller.

However, it also increases the number of optimization variables, and thus, it increases the complexity of the optimization problem.

The main advantage of the multiple shooting technique is that it reduces the computational burden of solving the optimization problem. This can be especially beneficial for systems with long horizons or high dimensional state space. Additionally, multiple shooting can be useful for handling certain types of constraints, such as nonlinear or time-varying constraints, which may be difficult to handle with a single shooting technique.

On the other hand, multiple shooting can introduce additional complexity to the optimization problem, which may require more advanced solvers or require the use of more computational resources. Additionally, there is a risk of introducing additional local minima which may not give the optimal solution.

In summary, multiple shooting is more computationally efficient for longer horizons, large dimension state space, and also better in dealing with certain types of constraints. On the other hand, it increases complexity and can introduce additional local minima which may not give the optimal solution.

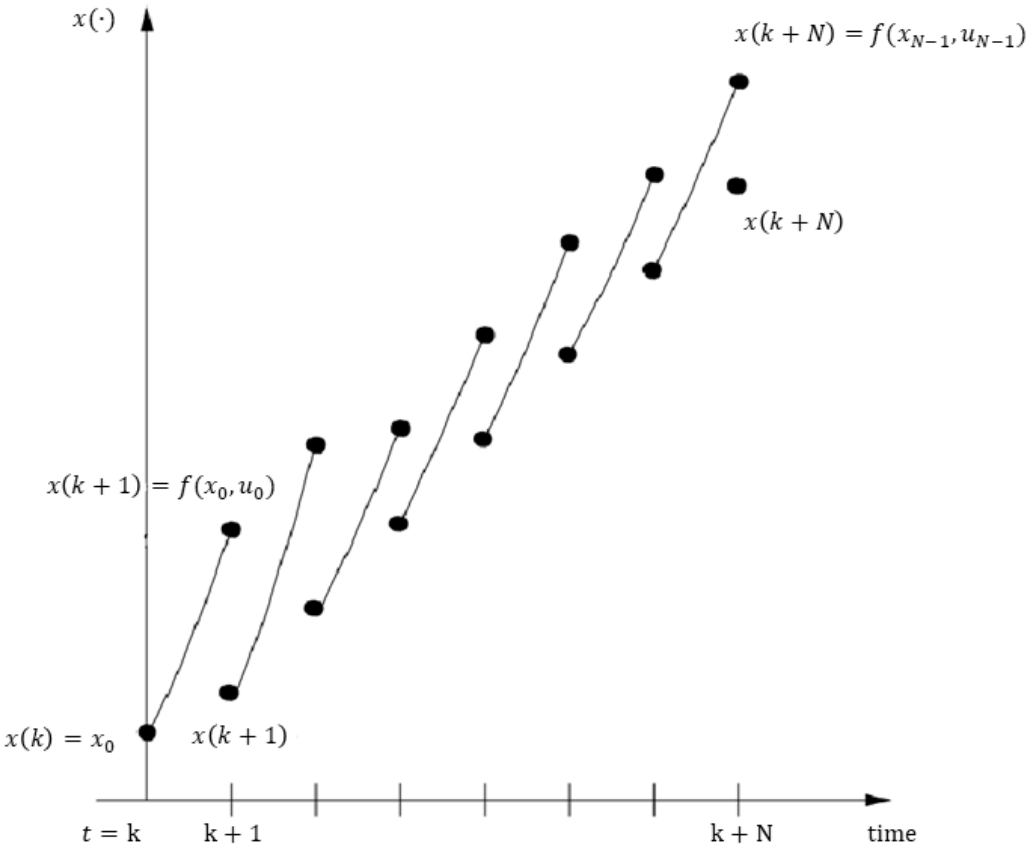

Figure 3.4. Multiple shooting approach.

3.1.4 Stable reverse motion with nonlinear MPC

The implementation of a nonlinear Model Predictive Control system for controlling a tractor-trailer kinematic model has recently been made. However, simply taking precautions to account for the nonlinearities of the system proved insufficient in overcoming the problem of instability during backward motion tracking, which can lead to dangerous jackknifing incidents. To address this challenge, a new approach is proposed that involves adding a corrective constraint to the system.

The proposed approach is to add a corrective constraint to prevent the system from becoming unstable. This is done by generating a stable auxiliary trajectory for the system in question by an inner trajectory tracking control via output error feedback through input-output linearization from which to derive the aforementioned anti-jackknifing constraint. This process is explained in more detail in the next sections.

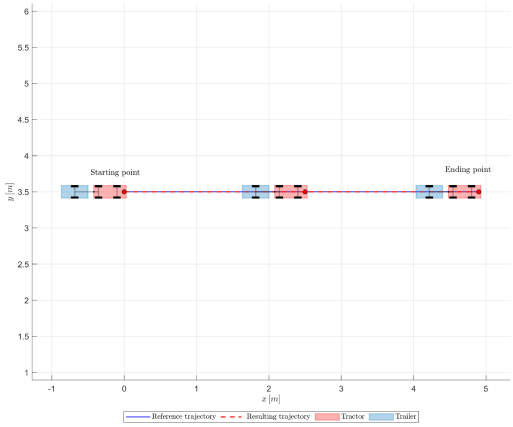

Figure 3.5. The nonlinear MPC used for tracking in forward motion is internally stable.

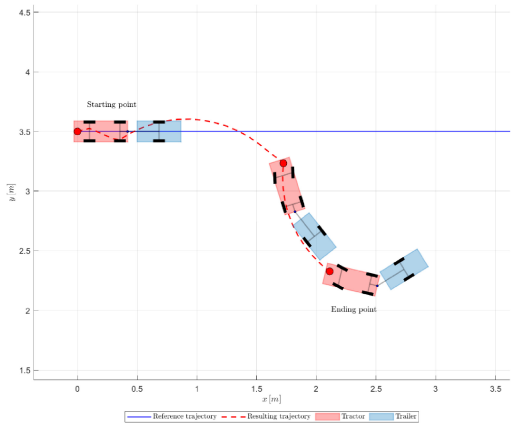

Figure 3.6. The nonlinear MPC used for tracking in backward motion is unstable, leading to jackknifing.

Trajectory tracking via output error feedback linearization

The concept of trajectory tracking through output error feedback and input-output linearization involves the development of a feedback action that is derived from the Cartesian error in the output. This feedback action is then used to accurately track a reference trajectory by ensuring that the velocity along the trajectory is closely followed. The basic idea behind this approach is to use a combination of feedback and feedforward control techniques to achieve the desired trajectory tracking performance.

To understand this concept more clearly, consider a simple block scheme that synthesizes the idea (Figure 3.7). In this scheme, the output Cartesian error is used as the input for the feedback control action. This error is then transformed into an appropriate control signal that can be used to drive the system along the desired trajectory. The velocity along the reference trajectory is also taken into account in this scheme through the use of a feedforward term. The feedforward term helps to ensure that the system is able to follow the reference trajectory with high accuracy and in real-time.

In this way, the system can achieve global exponential convergence to the desired trajectory, including square corners, by leveraging the invertibility of the map between available inputs and some derivative of the output, making the system linear. The method is dependent on the output error, the derivative of the reference trajectory, and the state variables for both the output trajectory reconstruction and input transformation.

In summary, the use of output error feedback linearization in trajectory tracking allows for a high level of accuracy and stability in tracking a reference trajectory. By combining feedback and feedforward control techniques, the system can be made to follow the desired trajectory with a high degree of precision. This approach leads to a highly effective control strategy for robotics and automation systems, enabling them to smoothly follow complex, arbitrary paths.

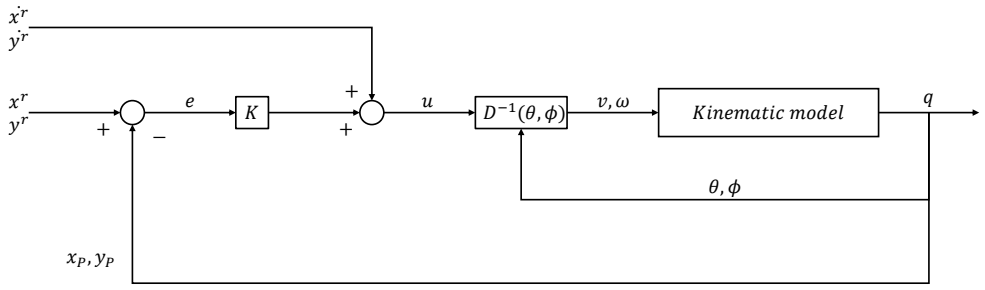

Figure 3.7. Output error feedback linearization control scheme.

Analyzing in more detail the procedure used: a Cartesian reference trajectory $(x^r(t), y^r(t))$ is a path assigned to be followed by the vehicle. The goal from a control perspective is to track this trajectory, which is treated as an output. To achieve this, a control system design approach called input-output linearization is utilized. This is achieved through a static state feedback input transformation.

The problem at hand involves finding an output whose dynamics can be linearized through static state feedback. Ideally, one would like to track the reference trajectory using the vehicle's representative point (x, y) (in (2.14) this point coincides with the middle point of the tractor's rear axle). However, this ideal scenario cannot be accomplished through static feedback alone because the decoupling matrix (D in Figure 3.7) becomes singular.

In fact, by choosing (x, y) as controlled output for the tracking problem, the driftless nonholonomic system cannot be transformed into a linear controllable one, because of its time derivative, derived directly from the kinematic model:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 \\ \sin\theta & 0 \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix}, \tag{3.7}$$

where its decoupling matrix is clearly singular.

As a solution, a different output point P with coordinates (x_P, y_P), which is a point on the tractor as well, placed at distance d from the tractor's front axle along the steering direction (Figure 3.8), is chosen to track the reference trajectory. This alternative approach allows for a workaround to the limitations posed by static feedback.

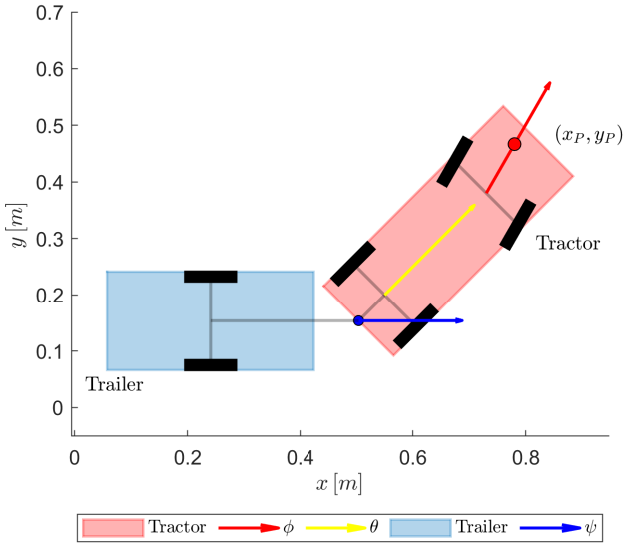

Figure 3.8. Tractor-trailer kinematic model controlled from a new point P placed at distance d from the tractor's front axle along the steering direction.

The (x_P, y_P) coordinates are defined as:

$$\begin{aligned} x_P &= x + L\cos\theta + d\cos(\theta + \phi) \\ y_P &= y + L\sin\theta + d\sin(\theta + \phi), \end{aligned} \tag{3.8}$$

whose derivatives result to be:

$$\begin{pmatrix} \dot{x_P} \\ \dot{y_P} \end{pmatrix} = \begin{pmatrix} \cos\theta - \frac{\tan\phi}{L}(L\sin\theta + d\sin(\theta + \phi)) & -d\sin(\theta + \phi) \\ \sin\theta + \frac{\tan\phi}{L}(L\cos\theta + d\cos(\theta + \phi)) & d\cos(\theta + \phi) \end{pmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix}. \tag{3.9}$$

Using these latter in place of the original $(\dot{x}, \dot{y})$ in the control-affine driftless model (2.13), therefore modifying the bases g_1 and g_2:

$$\begin{pmatrix} \dot{x_P} \\ \dot{y_P} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \cos\theta - \frac{\tan\phi}{L}(L\sin\theta + d\sin(\theta+\phi)) \\ \sin\theta + \frac{\tan\phi}{L}(L\cos\theta + d\cos(\theta+\phi)) \\ \frac{\tan\phi}{L} \\ -\frac{\tan\phi}{L}\left(\frac{L_1}{L_2}\cos\psi + 1\right) - \frac{\sin\psi}{L_2} \\ 0 \end{pmatrix} u_1 + \begin{pmatrix} -d\sin(\theta+\phi) \\ d\cos(\theta+\phi) \\ 0 \\ 0 \\ 1 \end{pmatrix} u_2, \tag{3.10}$$

and again, by defining u_1 as the traction speed of the tractor's rear wheel ($u_1 = v$) and u_2 as the steering velocity ($u_1 = \omega$), the behaviour of the vehicle controlled from the point P can be described through the following system of equations:

$$\begin{aligned} \dot{x_P} &= v\cos\theta - \frac{v\tan\phi}{L}(L\sin\theta + d\sin(\theta+\phi)) - \omega d\sin(\theta+\phi) \\ \dot{y_P} &= v\sin\theta + \frac{v\tan\phi}{L}(L\cos\theta + d\cos(\theta+\phi)) + \omega d\cos(\theta+\phi) \\ \dot{\theta} &= \frac{v\tan\phi}{L} \\ \dot{\psi} &= -\frac{v\tan\phi}{L}\left(\frac{L_1}{L_2}\cos\psi + 1\right) - \frac{v\sin\psi}{L_2} \\ \dot{\phi} &= \omega. \end{aligned} \tag{3.11}$$

From now on, in particular for the simulations, this model is considered the new reference model also for the NMPC.

However, when matrix $G(q_P)$ made up of g_1 g_2 in (3.10) has full column rank, m equations can always be transformed via feedback into simple integrators (input-output linearization and decoupling).

In fact, taken the previously defined decoupling matrix:

$$D(\theta,\phi) = \begin{pmatrix} \cos\theta - \frac{\tan\phi}{L}(L\sin\theta + d\sin(\theta+\phi)) & -d\sin(\theta+\phi) \\ \sin\theta + \frac{\tan\phi}{L}(L\cos\theta + d\cos(\theta+\phi)) & d\cos(\theta+\phi) \end{pmatrix}, \tag{3.12}$$

since $\det(D) = \frac{d}{\cos\phi}$ (so the decoupling matrix is invertible if d is nonzero), using the globally defined state feedback:

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = D^{-1}(\theta,\phi)\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}, \tag{3.13}$$

the input-output linearized model is equivalent to

$$\begin{aligned} \dot{x_P} &= u_1 \\ \dot{y_P} &= u_2 \\ \dot{\theta} &= \frac{\sin\phi}{L}\left(\cos(\theta+\phi)u_1 + \sin(\theta+\phi)u_2\right) \\ \dot{\psi} &= -\frac{1}{LL_2}\left(L_1\sin\phi\cos\psi + L_2\sin\phi + L\cos\phi\sin\psi\right)\left(\cos(\theta+\phi)u_1 + \sin(\theta+\phi)u_2\right) \\ \dot{\phi} &= -\left(\frac{\cos(\theta+\phi)\sin\phi}{L} + \frac{\sin(\theta+\phi)}{d}\right)u_1 - \left(\frac{\sin(\theta+\phi)\sin\phi}{L} - \frac{\cos(\theta+\phi)}{d}\right)u_2. \end{aligned} \tag{3.14}$$

The analyzed system can be separated into two components: a linear subsystem and the so-called zero dynamics. The linear subsystem consists of two separate channels, each with one integrator, and it is solely responsible for the input-output behaviour of the system. This part of the system has a relative degree of one, indicating that it requires only one derivative of the input to determine the output.

The other component, the nonlinear zero dynamics, does not have any effect on the output of the system. This implies that the zero dynamics can theoretically be unbounded without impacting the tracking performance of the system. However, in practice, if the zero dynamics were to diverge, particularly if the hitch angle were to diverge, it would result in jackknifing and unsuccessful output tracking. Therefore, it is essential to consider the behaviour of the zero dynamics to ensure the successful tracking of the system.

As a consequence, a linear feedback controller for $u = (u_1, u_2)^T$ makes the point P track any reference trajectory, even with a discontinuous tangent to the path (e.g., a square without stopping at corners).

This linear feedback is defined as:

$$u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} \dot{x}^r + k_x(x^r - x_p) \\ \dot{y}^r + k_y(y^r - y_p) \end{pmatrix}, \tag{3.15}$$

with $k_x > 0$, $k_y > 0$.

This is sufficient for the tracking error to converge exponentially to zero for any initial condition.

When it comes to control a tractor-trailer vehicle in reverse motion ($v < 0$), the stability of the zero dynamics plays a crucial role. If a system has positive eigenvalues, it indicates that the origin is unstable for the zero dynamics, making the system non-minimum phase [8]. This means that the evolution of the internal variables is not bounded when tracking in backward motion, leading to the divergence of variables such as θ, ψ, and ϕ as the vehicle moves. Notably, this instability is more pronounced at higher speeds, for shorter vehicles, or when the distance from the point P to the wheels is small.

On the other hand, if the tracking is in forward motion ($v > 0$), the eigenvalue analysis shows that the origin of the zero dynamics is asymptotically stable, allowing θ, ψ and ϕ to converge to zero.

Therefore, the divergence of angular variables, especially ψ, can result in the jackknife phenomenon, where the vehicle becomes unstable and difficult to control. In essence, the instability of the zero dynamics is the root cause of the jackknife phenomenon when tracking in backward motion, making it a critical factor in the control of the vehicle. The simulations in Figure 3.5 and Figure 3.6 serve to summarize and illustrate the concepts discussed.

In the case in question, beyond the insight into the zero dynamics instability of the model mentioned above, in the algorithm used in this thesis, trajectory tracking via output error feedback linearization is used only for forward motion trajectory tracking. Thus the instabilities due to the steering, hitch, and heading angles are avoided as they will always converge, as established in the analysis made above. The way this control technique is used as mentioned at the beginning of the section is essentially for the creation of a stable auxiliary trajectory (Figure 3.9) for the

system in question and for deriving stability constraints for zero dynamics to add to the nonlinear MPC introduced previously.

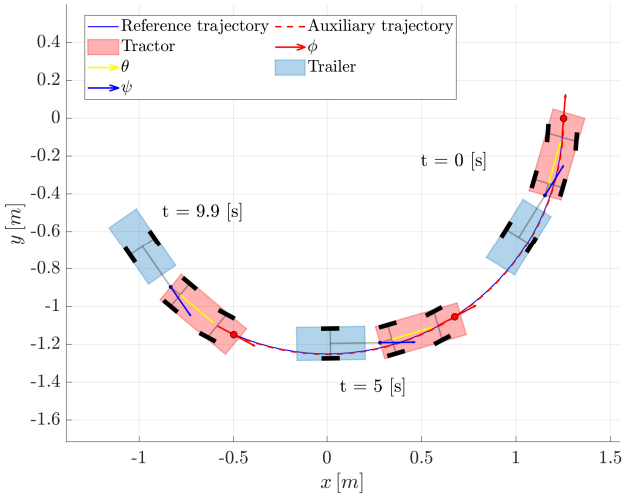

Figure 3.9. Stable and feasible auxiliary trajectory obtained via output error feedback and I-O linearization on a circular reference trajectory.

Basic stabilizing terminal constraint

It is possible to express a reverse motion trajectory as a forward motion trajectory by inverting the time from the end to the start, and arriving at the starting configuration at a given time. This is achieved practically by reflecting the trajectory about the midpoint of the original trajectory.

Basic idea: given a reference output trajectory to be covered in reverse motion from the instant $t = k$ to the instant $t = k + \tilde{N}$, with $\tilde{N} > N$, this can be seen as the same trajectory covered in reverse, in forward motion, from the final point to starting point considering a time frame reversed symmetrically with respect to the original case in reverse motion (as a sort of "rewind" function).

This is because, from the previous analysis, it is evident that a trajectory traveled in forward motion does not present problems of instability of the zero dynamics, therefore it is possible to exploit this fact to derive a stable and feasible auxiliary trajectory for the system while keeping the heading, hitch and steering angles bounded, a trajectory which instead of being covered in reverse motion from the initial point is covered in forward motion from the final point.

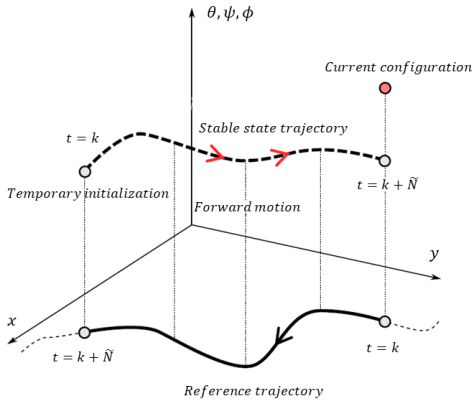

Figure 3.10. The generation of a stable state trajectory began with a temporary initialization and was achieved by purely tracking the reversed output trajectory in a forward motion (Source: [8]).

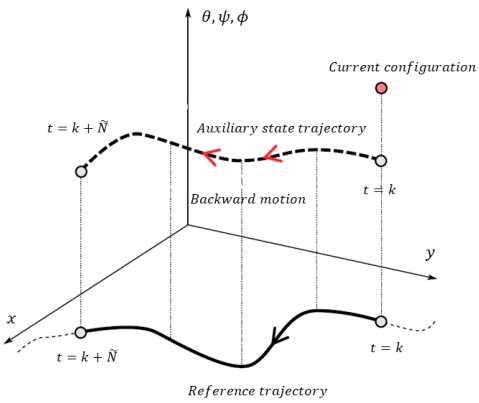

Figure 3.11. The stable trajectory resulted from the previous tracking was reversed to generate the auxiliary state trajectory (Source: [8]).

Therefore, to do this, it can make sense to express a reverse motion trajectory as a forward motion trajectory by inverting the time from the end to the start, considering as a new initial configuration the final reference state initialized temporarily in $t = k$ and arriving at the starting configuration (the final state if travelled in forward motion) at $t = k + \tilde{N}$.

In practice, once the stable state trajectory in forward motion has been obtained (Figure 3.10), simply by virtually inverting the initial and final states and instants (remember the "rewind" effect), to obtain a stable state trajectory in reverse motion it is sufficient to re-invert the forward trajectory previously obtained in order to have the original initial and final states and instants again (Figure 3.11).

What has been done is that, when the time is reversed from the end to the start, the trajectory is essentially reflected about the point $(t, q) = (k + \tilde{N}/2, (q_0 + q_{\tilde{N}})/2)$, which is the midpoint of the original trajectory. This produces a new trajectory that starts ideally at the original final time instant $t = k + \tilde{N}$ and ends at the original $t = k$.

If the original trajectory has symmetrical properties, such as constant velocity or acceleration, then starting the new trajectory at the final time can still represent the same motion as the original trajectory, but in forward motion. However, if the original trajectory has non-symmetrical properties, such as varying acceleration, then starting the new trajectory at the final time may not produce an accurate representation of the original motion.

In summary, it can make sense to start the new trajectory at $t = k + \tilde{N}$ and arrive at the starting configuration at $t = k$, but the resulting trajectory may not match the original reverse motion trajectory exactly, particularly if the motion is non-symmetrical.

But since in this case the auxiliary trajectory does not serve as a new reference

trajectory to follow but simply to extrapolate new stability constraints on the θ, ψ, and ϕ angles and therefore on the configuration of the system, at the instant $t = k + N$, to be controlled in particular on the zero dynamics, an exact matching in terms of acceleration and velocity between the trajectory in forward motion and in reverse motion can be omitted, as the focus is on deriving stable and feasible constraints that can be used to control the system.

Finally, the stabilizing constraints on the angles θ, ψ, and ϕ, given the new auxiliary trajectory resulting from the trajectory tracking via output error feedback linearization, is derived by taking the values of these angles at the instant $t = k+N$ and using them as equality constraints on the state variables in (3.6).

The general expression of this constraint to be added to the aforementioned nonlinear problem would be like the following:

$$\begin{aligned} \theta(k+N) - \theta_f &= 0 \\ \psi(k+N) - \psi_f &= 0 \\ \phi(k+N) - \phi_f &= 0, \end{aligned} \tag{3.16}$$

where θ_f, ψ_f and ϕ_f are the final values for the angles θ_{aux}, ψ_{aux} and ϕ_{aux} respectively at the instant $t = k + N$ obtained with the previous procedure from the auxiliary trajectory.

Currently, a terminal constraint is being applied to the unstable variables θ, ψ and ϕ. It is being imposed that these three angles, at the instant $t = k + N$, match the corresponding angles of the auxiliary trajectory.

At this point, defining as ξ_{k+N} the error between the whole state at time $t = k+N$ and the corresponding state values on the auxiliary trajectory:

$$\xi_{k+N} = \begin{pmatrix} x_P(k+N) - x_{Pf} \\ y_P(k+N) - y_{Pf} \\ \theta(k+N) - \theta_f \\ \psi(k+N) - \psi_f \\ \phi(k+N) - \phi_f \end{pmatrix}, \tag{3.17}$$

and defining a constant activation matrix, denoted as $T_u \in \mathbb{R}^{3\times5}$, where $T_u = (0_{3\times2} \quad I_{3\times3})$, the constraint mentioned earlier can be restated as:

$$T_u\, \xi_{k+N} = 0. \tag{3.18}$$

This is a 3-dimensional linear constraint for the NMPC and, in particular, thanks to the matrix T_u, in this way it is possible to operate on the unstable variables of the state.

This new statement to express the constraint is made ad hoc. In fact, a constraint with this structure can be further exploited, as explained in more detail below.

Advanced stabilizing terminal constraint

The stabilizing constraint in the basic version is a crucial element that ensures the stability of the nonlinear Model Predictive Control. This constraint involves the imposition of predetermined angles that are known to the system. These angles are obtained by following an auxiliary trajectory using a trajectory tracking procedure

via output error feedback and input-output linearization, which have been described above.

Assuming that the output error feedback via input-output linearization allows the system to converge to the reference trajectory, a feasible and stable trajectory can be obtained for the angles θ, ψ, and ϕ. However, to achieve this, the variables x_P and y_P also play an important role in the dynamics of the system. In fact, in this case, these variables are used as inputs for the input-output linearization.

It is important to note that the stability constraint in its basic version does not consider the role of x_P and y_P variables in the system's dynamics. The basic constraint only considers the values of the auxiliary trajectory and imposes their equality for the final state. However, by taking into account the role of x_P and y_P variables, the constraint can be further developed to obtain more degrees of freedom. This can lead to better performance and results on the final behaviour of the nonlinear MPC.

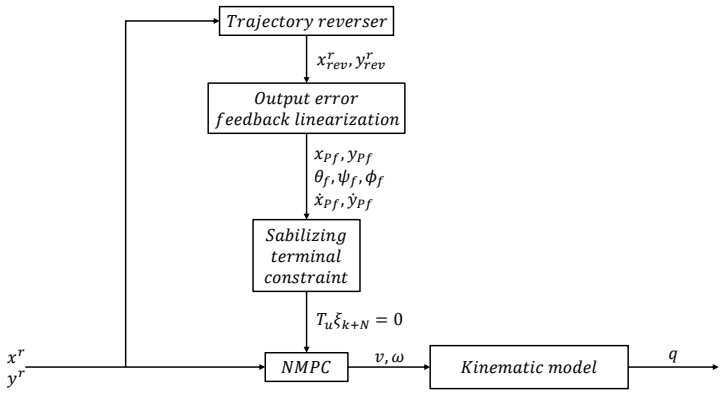

Figure 3.12. Trajectory tracking control scheme with stabilizing terminal constraint.

To formulate this new advanced constraint, the reasoning carried out starts with linearization. This is a technique used to approximate the behaviour of a nonlinear system by a linear system around a given trajectory.

Consider the nonlinear system described by the state equation 3.11, $\dot{q} = f(q, u)$, where q is the state vector, u is the input vector, and $f(q, u)$ is the vector-valued function that describes the dynamics of the system.

Let q_{aux} be the auxiliary trajectory obtained before, around which it is meant to linearize the system. It is possible to define the error coordinates as in the case of the basic stabilizing condition, by setting $\xi = q - q_{aux}$. The idea is to express the dynamics of the system in terms of the error coordinates and then linearize the resulting equation around the auxiliary trajectory.

To do this, the Taylor series expansion to approximate the function $f(q, u)$ around the auxiliary trajectory is used. This gives:

$$f(q, u) = f(q_{aux}, u) + \nabla f(q_{aux}, u)(q - q_{aux}) + \mathcal{O}(||q - q_{aux}||^2), \tag{3.19}$$

where $\nabla f(q_{aux}, u)$ is the Jacobian matrix of $f(q, u)$ evaluated at the auxiliary trajectory.

Substituting this into the original equation, the previous expression becomes:

$$\dot{q} = f(q_{aux}, u) + \nabla f(q_{aux}, u)(q - q_{aux}) + \mathcal{O}(||q - q_{aux}||^2). \tag{3.20}$$

Now, it is possible to express the dynamics of the system in terms of the error coordinates:

$$\dot{\xi} = \dot{q} - \dot{q}_{aux} = f(q_{aux}, u) + \nabla f(q_{aux}, u)(q - q_{aux}) - \dot{q}_{aux}. \tag{3.21}$$

This equation can be simplified by assuming that $\dot{q}_{aux}$ is equal to $f(q_{aux}, u)$, which is a reasonable assumption if the auxiliary trajectory is a steady-state trajectory. This leads to:

$$\dot{\xi} = \nabla f(q_{aux}, u)(q - q_{aux}). \tag{3.22}$$

Finally, linearizing this equation around the auxiliary trajectory by assuming that the error coordinates are small, gives us:

$$\dot{\xi} = \nabla f(q_{aux}, u)(q - q_{aux}) \approx \nabla f(q_{aux}, u)\xi. \tag{3.23}$$

This linear equation describes the behaviour of the system in terms of the error coordinates ξ.

In summary, the process of linearizing the nonlinear system around the auxiliary trajectory involves expressing the dynamics of the system in terms of error coordinates, using Taylor series expansion to approximate the nonlinear function, and then linearizing the resulting equation around the given trajectory. The resulting linear equation describes the behaviour of the system in terms of the error coordinates.

In this way, it is possible to express the state matrix A of the system in terms of the error coordinates by setting $A = \nabla f(q_{aux}, u)$ and assuming that the second-order terms in the Taylor series expansion of $f(q, u)$ are negligible. This results in:

$$\dot{\xi} = A\xi. \tag{3.24}$$

To compute the Jacobian matrix $\nabla f(q_{aux}, u)$, it is needed to compute the partial derivatives of $f(q, u)$ with respect to the state variables q and the input variables 3.15, the ones of the output error feedback linearization, and then to evaluate the matrix $A \in \mathbb{R}^{5\times5}$ obtained with the auxiliary trajectory q_{aux}.

Therefore the structure of the state matrix A, evaluated with the auxiliary trajectory q_{aux} and the input 3.15 is:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix}, \tag{3.25}$$

where

$$
\begin{aligned}
a_{11} &= -k_x \\
a_{12} &= 0 \\
a_{13} &= 0 \\
a_{14} &= 0 \\
a_{15} &= 0 \\
a_{21} &= 0 \\
a_{22} &= -k_y \\
a_{23} &= 0 \\
a_{24} &= 0 \\
a_{25} &= 0 \\
a_{31} &= -\frac{k_x\, c_{\phi_{aux}+\theta_{aux}}\, s_{\phi_{aux}}}{L} \\
a_{32} &= -\frac{k_y\, s_{\phi_{aux}+\theta_{aux}}\, s_{\phi_{aux}}}{L} \\
a_{33} &= \frac{s_{\phi_{aux}}\,(\dot{y}_{P_{aux}}\, c_{\phi_{aux}+\theta_{aux}} - \dot{x}_{P_{aux}}\, s_{\phi_{aux}+\theta_{aux}})}{L} \\
a_{34} &= 0 \\
a_{35} &= \frac{\dot{x}_{P_{aux}}\, c_{2\phi_{aux}+\theta_{aux}} + \dot{y}_{P_{aux}}\, s_{2\phi_{aux}+\theta_{aux}}}{L} \\
a_{41} &= \frac{k_x\, c_{\phi_{aux}+\theta_{aux}}\,(L_2\, s_{\phi_{aux}} + L\, c_{\phi_{aux}}\, s_{\psi_{aux}} + L_1\, c_{\psi_{aux}}\, s_{\phi_{aux}})}{L\, L_2} \\
a_{42} &= \frac{k_y\, s_{\phi_{aux}+\theta_{aux}}\,(L_2\, s_{\phi_{aux}} + L\, c_{\phi_{aux}}\, s_{\psi_{aux}} + L_1\, c_{\psi_{aux}}\, s_{\phi_{aux}})}{L\, L_2} \\
a_{43} &= -\frac{(\dot{y}_{P_{aux}}\, c_{\phi_{aux}+\theta_{aux}} - \dot{x}_{P_{aux}}\, s_{\phi_{aux}+\theta_{aux}})\,(L_2\, s_{\phi_{aux}} + L\, c_{\phi_{aux}}\, s_{\psi_{aux}} + L_1\, c_{\psi_{aux}}\, s_{\phi_{aux}})}{L\, L_2} \\
a_{44} &= -\frac{(L\, c_{\phi_{aux}}\, c_{\psi_{aux}} - L_1\, s_{\phi_{aux}}\, s_{\psi_{aux}})\,(\dot{x}_{P_{aux}}\, c_{\phi_{aux}+\theta_{aux}} + \dot{y}_{P_{aux}}\, s_{\phi_{aux}+\theta_{aux}})}{L\, L_2} \\
a_{45} &= -\frac{(\dot{x}_{P_{aux}}\, c_{\phi_{aux}+\theta_{aux}} + \dot{y}_{P_{aux}}\, s_{\phi_{aux}+\theta_{aux}})\,(L_2\, c_{\phi_{aux}} + L_1\, c_{\phi_{aux}}\, c_{\psi_{aux}} - L\, s_{\phi_{aux}}\, s_{\psi_{aux}})}{L\, L_2} + \quad \dots \\
&\dots \quad - \frac{(\dot{y}_{P_{aux}}\, c_{\phi_{aux}+\theta_{aux}} - \dot{x}_{P_{aux}}\, s_{\phi_{aux}+\theta_{aux}})\,(L_2\, s_{\phi_{aux}} + L\, c_{\phi_{aux}}\, s_{\psi_{aux}} + L_1\, c_{\psi_{aux}}\, s_{\phi_{aux}})}{L\, L_2} \\
a_{51} &= k_x \left(\frac{s_{\phi_{aux}+\theta_{aux}}}{d} + \frac{c_{\phi_{aux}+\theta_{aux}}\, s_{\phi_{aux}}}{L} \right) \\
a_{52} &= -k_y \left(\frac{c_{\phi_{aux}+\theta_{aux}}}{d} - \frac{s_{\phi_{aux}+\theta_{aux}}\, s_{\phi_{aux}}}{L} \right) \\
a_{53} &= -\dot{x}_{P_{aux}} \left(\frac{c_{\phi_{aux}+\theta_{aux}}}{d} - \frac{s_{\phi_{aux}+\theta_{aux}}\, s_{\phi_{aux}}}{L} \right) - \dot{y}_{P_{aux}} \left(\frac{s_{\phi_{aux}+\theta_{aux}}}{d} + \frac{c_{\phi_{aux}+\theta_{aux}}\, s_{\phi_{aux}}}{L} \right) \\
a_{54} &= 0 \\
a_{55} &= -\dot{x}_{P_{aux}} \left(\frac{c_{2\phi_{aux}+\theta_{aux}}}{L} + \frac{c_{\phi_{aux}+\theta_{aux}}}{d} \right) - \dot{y}_{P_{aux}} \left(\frac{s_{2\phi_{aux}+\theta_{aux}}}{L} + \frac{s_{\phi_{aux}+\theta_{aux}}}{d} \right),
\end{aligned}
\tag{3.26}
$$

setting for compactness $\cos\alpha = c_\alpha$ and $\sin\alpha = s_\alpha$.

The next step in order to reach the so-called advanced stabilizing constraint consists of decoupling the state matrix.

In this context, the stability constraint applies solely to unstable states, which are determined by the values of the error dynamics $\xi = (\xi_1\ \xi_2\ \xi_3\ \xi_4\ \xi_5)^T$. Specifically, the states of θ, ψ, and ϕ are included in the unstable error dynamics, with the diverging components being ξ_3, ξ_4 and ξ_5. Importantly, this system does not have any eigenvalues with zero real parts.

In a linear system of the form of 3.24, the behaviour of the system is determined by the eigenvalues of A.

Eigenvalues are values λ such that when A is multiplied by a corresponding eigenvector ν, the result is a scalar multiple of ν, i.e., $A\nu = \lambda\nu$. The sign of the eigenvalue determines whether the corresponding eigenvector points towards or away from the origin. If the real part of an eigenvalue is negative, the corresponding eigenvector points toward the origin and the system is stable. If the real part is positive, the corresponding eigenvector points away from the origin and the system is unstable.

If the eigenvalues of the new state matrix can be arranged such that the stable eigenvalues are on one side and the unstable eigenvalues are on the other side, then the system can be decoupled into stable and unstable subsystems.

This can be done by finding the transformation matrix T such that the new state matrix is in a Jordan canonical form. Jordan canonical form is a matrix form that simplifies the eigenvalues of the matrix, making it easier to identify stable and unstable modes.

Therefore, the goal of decoupling the dynamics matrix of the state is to find a change of coordinates that transforms the system into a set of subsystems, each of which has a simpler and more manageable behaviour. This is done by finding a transformation matrix T such that $\eta = T\xi$, where η is a new set of coordinates. The result is the following:

$$\begin{aligned} \begin{pmatrix} \dot{\eta_s} \\ \dot{\eta_u} \end{pmatrix} &= \begin{pmatrix} \Lambda_s & 0 \\ 0 & \Lambda_u \end{pmatrix} \begin{pmatrix} \eta_s \\ \eta_u \end{pmatrix} \\ \Lambda &= TAT^{-1} \\ T &= U^{-1}. \end{aligned} \tag{3.27}$$

This transformation relies on the matrix U that contains the right eigenvectors of the system and the diagonal matrix $\Lambda = diag(\Lambda_s, \Lambda_u)$ that contains the eigenvalues. Furthermore, the eigenvalues are split into two categories: Λ_s and Λ_u. Λ_s has a negative real part, while Λ_u has a positive real part. Both are in diagonal form.

Once the system has been decoupled into stable and unstable subsystems, the stable subsystem can be controlled independently of the unstable subsystem. This makes it easier to design controllers that stabilize the system and keep it from becoming unstable.

As regards the advanced stabilizing constraint, given the considerations made above, this is based on the previous relation $\eta = T\xi$. In fact, setting $\eta = 0$ in the transformed coordinate system means that the error between the current state q and the auxiliary trajectory q_{aux}, i.e., ξ, is equal to zero. This implies that the system

state q is exactly equal to the auxiliary trajectory q_{aux}, and the system is following the desired trajectory.

In other words, setting $\eta = 0$ implies that the error between the current state q and the auxiliary trajectory q_{aux} has been completely eliminated using the transformation matrix T. The diagonalization of the matrix A into a stable subsystem and an unstable subsystem has enabled to analyze and control the system more efficiently, and setting $\eta = 0$ means that the system is in the desired state.

It is worth noting that since the system matrix $A = A(t)$ is time-dependent, the eigenvalues, eigenvectors, and transformation matrix T all change with time.

In order to obtain a block-diagonal system as in (3.27), the matrix T has the following structure:

$$T = U^{-1} = \begin{pmatrix} I_{2\times 2} & 0_{2\times 3} \\ & T_u \end{pmatrix}, \tag{3.28}$$

where $T_u \in \mathbb{R}^{3\times 5}$ is, of course, different from the matrix introduced for the basic version of the stabilizing constraint: it is a time-varying version, in order to operate on the unstable variables θ, ψ, and ϕ, with more degrees of freedom and different values, thanks to its dependence also to the x_P and y_P coordinates, as stated before.

Since it is a terminal constraint, defining ξ_{k+N} the error between the whole state at time $t = k + N$ and the corresponding state values on the auxiliary trajectory as in (3.17), the new advanced stabilizing terminal constraint can be expressed as in (3.18), but with a time-dependent T_u:

$$T_u \, \xi_{k+N} = 0. \tag{3.29}$$

Obtained this new terminal constraint, for sake of completeness, the algorithm used is shown in the block scheme in Figure 3.12.

The behaviour of the two terminal constraints is explored in more detail in Chapter 5.

Chapter 4

Implementation of the proposed approach

In this chapter of the thesis, the implementation of a nonlinear Model Predictive Control (MPC) algorithm using MATLAB and CasADi [1] is discussed.

MPC is a control strategy that predicts future states of a system by using a model of the system and optimizes control inputs over a finite prediction horizon to achieve a desired performance. Nonlinear models in MPC provide greater flexibility in modeling the system's dynamics, but also bring greater challenges in terms of computational tractability and stability.

MATLAB, a widely used numerical computing environment and programming language, and CasADi, an open-source toolbox for numerical optimization, are used to implement the MPC algorithm. CasADi can generate code for the nonlinear optimization problem, while MATLAB is used to simulate the system and the control algorithm. The simulation section provides an overview of the nonlinear MPC algorithm and its mathematical formulation, details of the nonlinear system used as an example, and the design of the MPC controller, including the selection of cost functions and constraints. The simulation results are then presented and discussed, evaluating the performance of the nonlinear MPC algorithm.

CasADi is widely used in industry and research to solve nonlinear optimization problems due to its efficiency and convenience in formulating and solving optimization problems in the MATLAB environment and its robust solver. It also allows for code generation and running the code in other languages like C and Python for increased computational speed. CasADi has a general scope of numerical optimization and specifically facilitates the solution of Nonlinear Programs (NLPs). It facilitates but does not actually solve the NLPs, but solver plugins can be added post-installation. CasADi is a free and open-source tool that can be used commercially, and it has been in development since 2009. CasADi can handle four standard problems, including Quadratic Programs (QPs), Nonlinear Programs (NLPs), Root finding problems, and Initial-value problems in ODE/DAE.

4.1 Design choices

4.1.1 Initial condition

One important aspect of using MPC for trajectory tracking is the initialization of the model. The initial state of the model is used to predict the future states of the system.

To guarantee the reliability of the result and the optimality of the trajectory, the experiments carried out foresee an initial position not too far from the objective (however, the possibility of considering a high initial distance from the objective is not excluded), first to allow the optimizer to converge much more quickly, reducing the number of computations (with consequent reduction of the probability of error), thus ensuring the feasibility of the calculated trajectory, reaching the goal or arriving asymptotically very close to it.

Another reason is that if the initial position of the model is not on the desired trajectory, the predicted trajectory may deviate significantly from the desired trajectory. This can lead to unexpected behaviour and instability in the system. For example, the controller may generate inputs that are too aggressive or result in overshooting the desired trajectory, leading to oscillations or even instability.

Therefore, it is important to initialize the nonlinear MPC algorithm with an initial state that lies on the desired trajectory to ensure that the predicted trajectory remains close to the desired trajectory and the system behaves as expected.

Therefore, given as starting instant t_0 (typically set as $t_0 = 0\,\mathrm{s}$), the starting configuration q_0 is aligned properly to the correspondent point on the reference trajectory, therefore starting with null error on the Cartesian coordinates (x, y, θ) neglecting the ψ and ϕ angles initially:

$$q_0 = \begin{pmatrix} x^r(t_0) \\ y^r(t_0) \\ \mathrm{ATAN2}(\dot{y}^r(t_0), \dot{x}^r(t_0)) \\ 0 \\ 0 \end{pmatrix}. \tag{4.1}$$

However, the case in which the initial error both in position and in orientation is greater, therefore an initial configuration which does not lie on the reference trajectory, has also been evaluated. This is to compare the behaviour of the system with the different terminal stabilizing constraints discussed in 3.1.4.

4.1.2 Model integration method

The integration method is a fundamental technique used in the odometry-based localization process. By assuming small enough time intervals, the integration method considers the linear and angular velocities, v_k and ω_k, to be constant during the k-th interval. With this information and the initial configuration q_k of the system, the next configuration can be computed. In other words, the integration method helps determine the model's position and orientation at each sampling interval as it moves along an arc of a circle or a line segment if there is no angular velocity ($\omega_k = 0$).

The most common integration method is the Euler method. It is a first-order numerical procedure for solving ordinary differential equations (ODEs) with a given initial value. It is considered a simple and straightforward method to approximate the solution to an ODE, and it is relatively easy to implement. However, it is also relatively inaccurate, as it is a first-order method that relies on a small step size to reduce the error in the approximation.

The Runge-Kutta method, specifically the Runge-Kutta 4-th order method, is a higher-order numerical procedure for solving ODEs. It is considered to be a more accurate method than the Euler method, as it uses a more complex algorithm to approximate the solution to an ODE. This method provides a more accurate approximation of the solution in a given time step, and thus, it requires a smaller time step than the Euler method.

The reason that Runge-Kutta 4-th order method is superior to the Euler method is that it uses a more accurate formula for approximating the solution. The Euler method only uses the derivative of the function at the starting point to estimate the solution at the next step, while the Runge-Kutta method uses multiple estimates at different points along the step. This allows the Runge-Kutta method to make a more accurate estimate of the solution, thus reducing the error and the required step size for an accurate solution.

Another reason that Runge-Kutta method is more preferred is because its error can be calculated with the estimation of local truncation error that allow for adjusting the step size to achieve a desired accuracy. In contrast, Euler method does not have an error estimation mechanism.

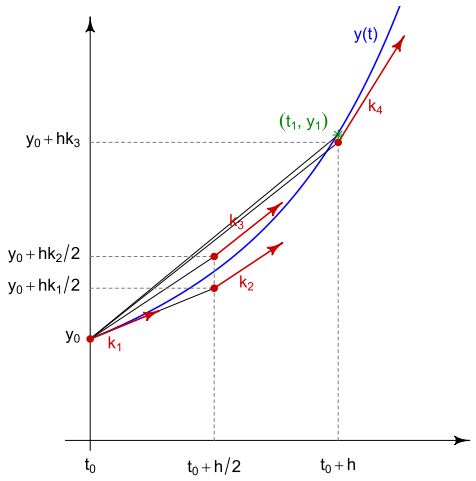

Figure 4.1. Runge-Kutta slopes (Source: [15]).

Overall, Runge-Kutta 4-th order method is more accurate and efficient than the Euler method when solving ODEs, but it is also more computationally expensive in some cases. Nevertheless this latter is the chosen integration method for this thesis.

It is commonly expressed generally as:

$$\begin{aligned}
\dot{x} &= f(x,u) \\
k_1 &= f\left(x_{n-1}, u\right) \\
k_2 &= f\left(x_{n-1} + \frac{h}{2}k_1, u\right) \\
k_3 &= f\left(x_{n-1} + \frac{h}{2}k_2, u\right) \\
k_4 &= f\left(x_{n-1} + hk_3, u\right) \\
x_n &= x_{n-1} + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right),
\end{aligned} \tag{4.2}$$

with h sampling step.

Definetely, the structure of the model is one of the main requirements. The simulator uses this model to make predictions on the future behaviour of the system, predicting all subsequent states. In this way, the optimal control input can be determined and applied accordingly. However, in order to use the mathematical model (3.14) in the simulator, it is necessary to obtain the discretized equivalent, that is to convert the continuous time model into a sampled data model with a sampling step T, this is the reason behind the choice of the Runge-Kutta integration method.

4.1.3 Physical parameters and constraints

To confirm the validity of the proposed methodology, a series of numerical simulations are carried out on a tractor-trailer vehicle that shares the same kinematic properties as a physical prototype, that is the Carson Unimog U300. This remarkable 1:12 scale model is a commercial radio-controlled replica of the renowned Mercedes Unimog U300 truck. In line with this, the parameters $L = 0.255\,\text{m}$, $L_1 = 0.068\,\text{m}$, and $L_2 = 0.262\,\text{m}$ (in the case of double-trailer $L_3 = 0.153\,\text{m}$ and $L_4 = 0.211\,\text{m}$) were established for the simulations. The control scheme in Section 3.1.4 employed a distance of $d = 0.1\,\text{m}$, and $k_x = k_y = 1$.

As for the physical parameters of the tractor-trailer to which the solver refers, these are also the lower and upper bounds of the control variables, therefore of the linear and steering velocities.

As regards the lower bound of linear velocity, in the present work the forward motion is of course less discussed with respect to the reverse motion, in the light of the considerations made previously. However, the complete case with forward and reverse motion together is considered for the choice of the ideal setup.

Given that the overall length of the prototype is approximately 60 cm while, in comparison, a standard tractor-trailer truck, with an average length of 22 m in the US, would correspond to a speed of approximately 40 km/h, the ratio $\frac{speed}{length} = 0.5$ circa, therefore:

$$u_{min} = \begin{bmatrix} -0.5\,\text{m/s} \quad \text{or} \quad 0 \\ -1.5\,\text{rad/s} \end{bmatrix}$$
$$u_{max} = \begin{bmatrix} 0.5\,\text{m/s} \quad \text{or} \quad 0 \\ 1.5\,\text{rad/s} \end{bmatrix}, \tag{4.3}$$

the 0 value for the linear velocity input is for cases where the vehicle can only move in forward motion or only in reverse motion.

For what concerns the constraints on the state, in the tractor-trailer system, the steering angle and the hitch angle cannot exceed a certain value for physical limitation:

$$|\psi(k)| \leq \frac{\pi}{4}\,\text{rad}$$
$$|\phi(k)| \leq \frac{\pi}{12}\,\text{rad}. \tag{4.4}$$

4.1.4 Control horizon, prediction horizon and sampling time

Another important aspect to consider is the time horizon considered, that affects the total duration of the experiment and the time that the user sets for the resolution of the problem. This implies that as the length of the horizon increases, the total elapsed time of the entire process also increases. In the case in question and, in general, for all models of wheeled vehicles (models that consider the orientation), the optimization of this parameter significantly affects the results obtained, as the lengths of the time horizon can influence how vehicles travel. It is therefore advisable to keep the parameter as small as possible for computational reasons, but large enough for the feasibility of the optimization problem and the achievement of the goal. In summary, a trade-off for the time horizon is set for each simulation among those feasible, which may vary according to the simulation setup considered.

In this case N represents the entire prediction horizon,it should be sufficiently large for the transient to be over in T, moreover, it should be clearly larger than the control horizon of the MPC, which in this case is equivalent to $N - 1$.

The choice of sampling time T is crucial as well in the process, to make simulation results more or less realistic. The goal is to respect the system dynamics used and make the vehicle's trajectory as regular as possible and physically acceptable. Considering the characteristics of the model and the technical specifications of the vehicle, an acceptable solution, below which there is a risk of compromising the feasibility of the trajectory and its regularity, is to set $T = 0.2\,\text{s}$ or $T = 0.1\,\text{s}$.

In this case, a low sampling time allows to simulate also the model in terms of physical implementation. However, a sampling time that is too low would lengthen the simulation times too much, unnecessarily tiring the architecture used to evaluate the adopted strategy.

4.1.5 The IPOPT solver

IPOPT (Interior Point OPTimizer) is an open-source solver used in CasADi for solving nonlinear optimization problems, including nonlinear Model Predictive

Control (NMPC) problems. It uses an interior-point algorithm that exploits the problem's sparsity structure to find a solution efficiently.

In CasADi, the IPOPT solver is used to solve the nonlinear optimization problem that arises in NMPC. The problem is formulated as a nonlinear program (NLP) with nonlinear constraints and a cost function, and then passed to the IPOPT solver. The solver uses a primal-dual interior-point method to find the optimal solution by iteratively solving a sequence of linearized problems.

To use the IPOPT solver in CasADi, first, the user must define the dynamic model of the system, the constraints, and the cost function. The problem is then formulated as an NLP using the CasADi syntax. The user can specify the solver options and set the solver to IPOPT. Finally, the NLP problem is solved using the IPOPT solver.

IPOPT is a highly efficient solver that is especially useful for large-scale problems with sparse matrices, which is common in NMPC problems. It also has the advantage of being able to handle nonlinear constraints and providing reliable and accurate solutions.

Overall, the IPOPT solver is a powerful tool for solving nonlinear MPC problems in MATLAB using CasADi, and its efficient algorithms and sparsity exploitation make it an excellent choice for large-scale problems.

4.2 Nonlinear anti-jackknifing MPC

As stated in the previous sections, the MPC framework is subject both to state and inputs constraints and to the terminal constraint (3.16).

Given all the considerations made so far, it is possible to express the generic discrete-time NMPC described in (3.3) in such a way as to solve the problem in question.

As a cost function, for this particular case, given that the reference trajectory implicitly also imposes a travel speed thanks to the timing law that characterizes the path to travel, the terms in the cost function can be defined as follows:

$$J_N(q(k), u(k)) = \sum_{k=0}^{N-1} ||q(k) - q^r(k)||_Q^2 + ||u(k)||_R^2 + ||u(k) - u(k-1)||_P^2, \qquad (4.5)$$

where $q(k)$ is the state of the system, $q^r(k)$ is the reference trajectory, which is the desired behaviour or trajectory of the system and $u(k)$ is the input to the system. Q is a positive semi-definite matrix that penalizes deviations of the system state from the reference trajectory. The term $||q(k) - q^r(k)||_Q^2$ represents the deviation of the state from the reference trajectory, weighted by the matrix Q. R is a positive semi-definite matrix that penalizes the magnitude of the input. The term $||u(k)||_R^2$ represents the magnitude of the input, weighted by the matrix R. P is a positive semi-definite matrix that penalizes the rate of change of the input. The term $||u(k) - u(k-1)||_P^2$ represents the rate of change of the input between two adjacent time steps, weighted by the matrix P.

The goal of using this cost function is typically to find an input trajectory that minimizes the deviation of the system state from the reference trajectory, while also

minimizing the magnitude of the input and ensuring that the rate of change of the input does not exceed some specified limit. The third term in the cost function, involving the matrix P, enforces this rate constraint by penalizing rapid changes in the input. The matrix P determines the trade-off between minimizing the input rate and achieving the desired system behaviour.

When $k = 0$, the cost function includes the term $||u(0) - u(-1)||_P^2$, which involves the control input $u(-1)$. However, since it is not defined at the initial time step, it cannot directly be included in the optimization problem.

To address this issue, it is possible to introduce an additional initialization to represent the "previous" control input at time $k = -1$, precisely the known u_{-1}.

Note that this is an issue of the cost only at time $k = 0$, which depends on $u(0)$ and $u(-1)$. The remaining iterations represent the cost over the remaining time steps $k = 1, \dots, N-1$, which depend on the control inputs $u(1), \dots, u(N-1)$. In this way it is possible to find the optimal control input sequence $u^*(0), \dots, u^*(N-1)$ that minimizes the cost function $J_N(q(k), u(k))$.

Therefore, in the cost function, the squared norm of the error on the state variables is considered. But more precisely, since the aim of this work is to track the output reference trajectory (x^r, y^r), the cost function can be further simplified by setting to 0 the weights correspondent to the other variables in the Q matrix.

The following is the problem to be solved:

$$
\begin{aligned}
\underset{u}{\text{minimize}} \quad & J_N(q(k), u(k)) = \sum_{k=0}^{N-1} ||q(k) - q^r(k)||_Q^2 + ||u(k)||_R^2 + ||u(k) - u(k-1)||_P^2 \\
\text{subject to} \quad & q(k+1) = f(q(k), u(k)), \\
& u_{min} \leq u(k) \leq u_{max}, \quad \forall k \in [0, N-1], \\
& |\psi(k)| \leq \psi_{max}, \quad \forall k \in [0, N], \\
& |\phi(k)| \leq \phi_{max}, \quad \forall k \in [0, N], \\
& T_u \, \xi_{k+N} = 0, \\
& u(-1) = u_{-1}, \\
& q(0) = q_0,
\end{aligned}
\tag{4.6}
$$

which considers also the terminal stabilizing constraints (basic or advanced) and the multiple shooting constraints.

The cost function, $J_N(q(k), u(k))$, is a summation of the running costs over the entire prediction horizon.

For what concerns the matrices $Q \geq 0$, $R \geq 0$ and $P \geq 0$, the optimal set of weights can be obtained from manual empirical (trial-and-error) parameter search, in fact, this type of model parameters are referred to as tuning parameters because there is no analytical formula available to calculate an appropriate value for each of them.

After a substantial initial phase of testing and tuning to obtain the correct functioning of the nonlinear MPC with the problem in question, an acceptable set of parameters is found to be the following

$$
\begin{aligned}
Q &= \begin{pmatrix} 1000 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
R &= \begin{pmatrix} 10 & 0 \\ 0 & 100 \end{pmatrix} \\
P &= \begin{pmatrix} 1000 & 0 \\ 0 & 10 \end{pmatrix}.
\end{aligned}
\tag{4.7}
$$

In addition to the implementation just shown, also a feedforward with specific setpoints is added to the input in order to improve the performance of the tracking system.

Feedforward control is a technique used to improve the performance of a closed-loop control system. It involves adding a pre-calculated term to the control input, based on a predicted setpoint, to reduce the error between the desired output and the actual output. In the context of nonlinear MPC for trajectory tracking, a feedforward term can be used to improve the tracking performance by adding a term to the control input that compensates for any change in the setpoint.

The feedforward term is calculated based on the auxiliary trajectory input variables obtained in (3.13), and is added to the control input at each time step. By incorporating a feedforward term, the nonlinear MPC controller can more accurately track the reference trajectory, reducing the tracking error and improving the overall performance of the system.

The block diagram describing the final structure of the nonlinear anti-jackknifing MPC in question is shown in the following Figure 4.2.

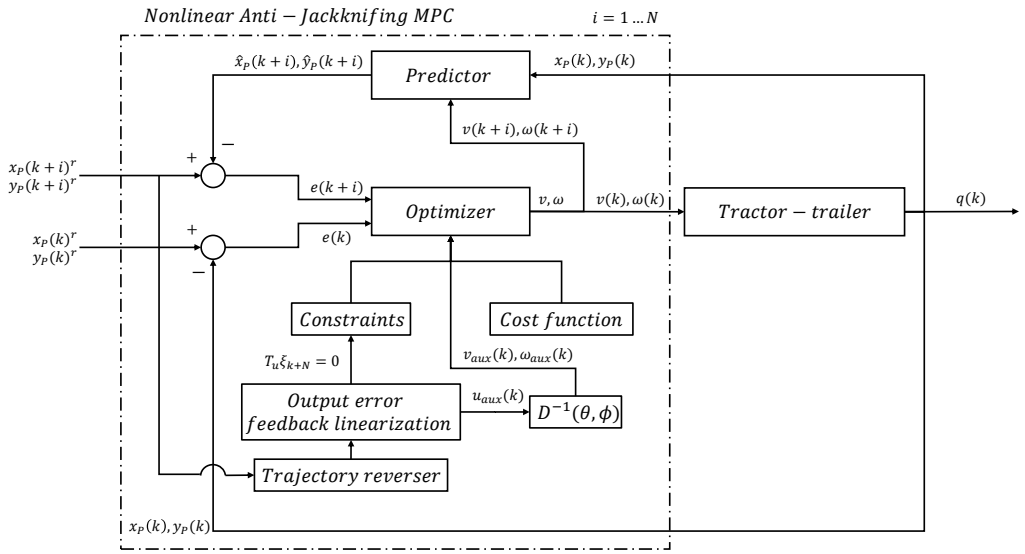

Figure 4.2. Nonlinear anti-jackknifing MPC for trajectory tracking control scheme.

Chapter 5

Trajectory tracking control performance

The behaviour of the tractor-trailer vehicle is investigated in the present study through numerical simulations carried out in MATLAB R2023a. The simulations have been performed on a laptop equipped with a single CPU, an Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz, and 8GB of RAM, and the results and timings of each simulation have been recorded and reported.

It is important to note that the time taken for simulations may vary depending on the hardware used. In this study, the simulations have been conducted on a laptop with a mid-range CPU and memory, and it is reasonable to expect that more powerful hardware would yield better performance. Nonetheless, the results obtained from the simulations offer valuable insights into the behaviour of the tractor-trailer vehicle.

To provide a comprehensive understanding of the simulations and their implications, the following sections discuss the methodology employed, the parameters considered, and the results obtained. The aim is to offer a detailed account of the simulation process and its outcomes, as well as a critical assessment of the approach taken.

Various metrics can be used to evaluate the performance of a control system, and one of the most important metrics for trajectory tracking is settling time. Settling time is defined as the time it takes for a system to settle within a certain range of the desired trajectory after the trajectory has started, and it is a measure of the system's stability and its ability to track the desired trajectory over time.

A shorter settling time indicates a more stable system, which means that the system can track the desired trajectory more quickly and with less oscillation or overshoot. However, it is important to note that a shorter settling time does not always mean better performance, as it can come at the expense of increased overshoot or oscillation. Therefore, it is important to consider other metrics such as the root mean square error (RMSE) and maximum error to evaluate the overall performance of the system.

The root mean square error (RMSE) is a measure of the average tracking error over time. It is calculated as the square root of the mean squared error. The RMSE should be as small as possible, while the maximum error is the largest

deviation between the actual trajectory and the desired trajectory. Of course, also the maximum error should be as small as possible.

The smoother and more precise behaviour to reach the reference, which is an indication of a better tracking performance for the system, can be evaluated using these metrics. For instance, the RMSE can be used to compare the average tracking error over time for different systems or setups. Even if two systems have the same settling time, the system with the smoother and more precise behaviour will have a lower RMSE, indicating a better tracking performance. This means that the system is able to track the desired trajectory more accurately and with less deviation compared to the other system.

Similarly, the maximum error metric can be used to compare the worst-case deviation from the desired trajectory for different setups. The setup with the smoother and more precise behaviour will have a smaller maximum error compared to the other setup, indicating that it has a better ability to control the system's response and keep it within the desired range.

In conclusion, settling time is a critical metric for evaluating the performance of a trajectory tracking system. However, it is important to consider other metrics such as RMSE and maximum error to evaluate the overall tracking performance of the system. The smoother and more precise behaviour to reach the reference is an indication of better tracking performance and can be quantified using these metrics.

5.1 Stabilizing terminal constraints

Section 3.1.4 describes two types of stabilizing terminal constraints that have been introduced. These constraints aim to stabilize the angles θ, ψ, and ϕ of the system.

The basic version of the constraint is derived by taking the values of these angles from the auxiliary trajectory at the instant $t = k + N$ and using them as equality constraints on the state variables. This is done after trajectory tracking via output error feedback linearization.

On the other hand, the advanced version of the constraint considers the role of x_P and y_P variables in the system's dynamics, which the basic version does not. This constraint involves all state variables used in the output error feedback linearization, as explained previously.

Both constraints can improve the performance and results of the nonlinear MPC. However, the goal of this section is to determine which constraint can provide greater robustness and ensure the best completion of the objective.

5.1.1 Misaligned starting poses

The initial simulation of the system involves starting on a reference trajectory but with a significant orientation error in the vehicle's pose concerning the reference. In both scenarios, a part of a circular trajectory is used as a reference, where the initial poses are misaligned, but tangent and aligned to the reference trajectory. However, the entire system, including the trailer, is not perfectly aligned, resulting in a non-zero position error. This error is analyzed to evaluate the system's performance.

To conduct the simulation, a setup is created with a sampling time of 0.1 seconds, a control horizon that allows the completion of the task in the best possible way, and

a total simulation time of 25 seconds. The starting configuration for the following two simulations is given by the vector q_0, which has five components representing the position and orientation of the vehicle and trailer.

The basic stabilizing terminal constraint is used in the first simulation and the related results are analyzed. Next, the advanced stabilizing terminal constraint is simulated, and its results are examined. Finally, a complete analysis and comparison of the two simulations is being performed to evaluate the effectiveness of the different stabilizing terminal constraints.

In both cases, the starting configuration is:

$$q_0 = \begin{pmatrix} 1.25\,\mathrm{m} \\ 0\,\mathrm{m} \\ \frac{\pi}{2}\,\mathrm{rad} \\ 0\,\mathrm{rad} \\ 0\,\mathrm{rad} \end{pmatrix}. \tag{5.1}$$

Basic stabilizing terminal constraint

As mentioned above, the setup of the simulation times using the basic stabilizing terminal constraint is the following:

Table 5.1. Misaligned starting poses simulation setup (basic stabilizing terminal constraint).

Sampling time T	0.1 s
Control horizon	6 s
Total simulation time	25 s

- Average time for each MPC iteration: 48.10 ms.
- Average time for each solver operation: 43.69 ms.
- Total time for output error feedback linearization: 478.02 ms.
- Settling time: 11.60 s.
- Position RMSE: 0.22 m.

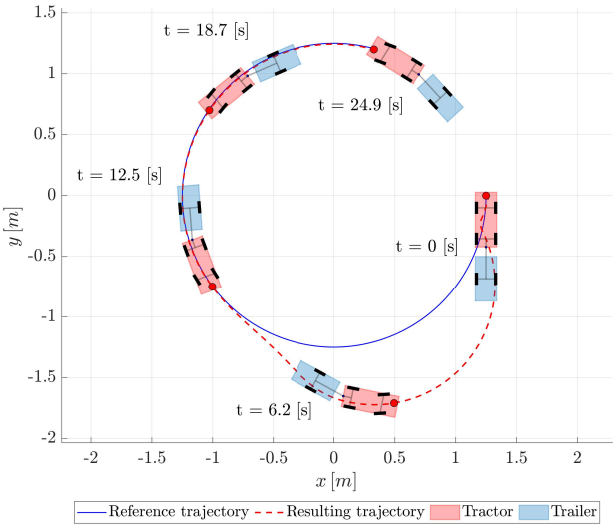

Figure 5.1. Circular trajectory in backward motion using the basic stabilizing terminal constraint.

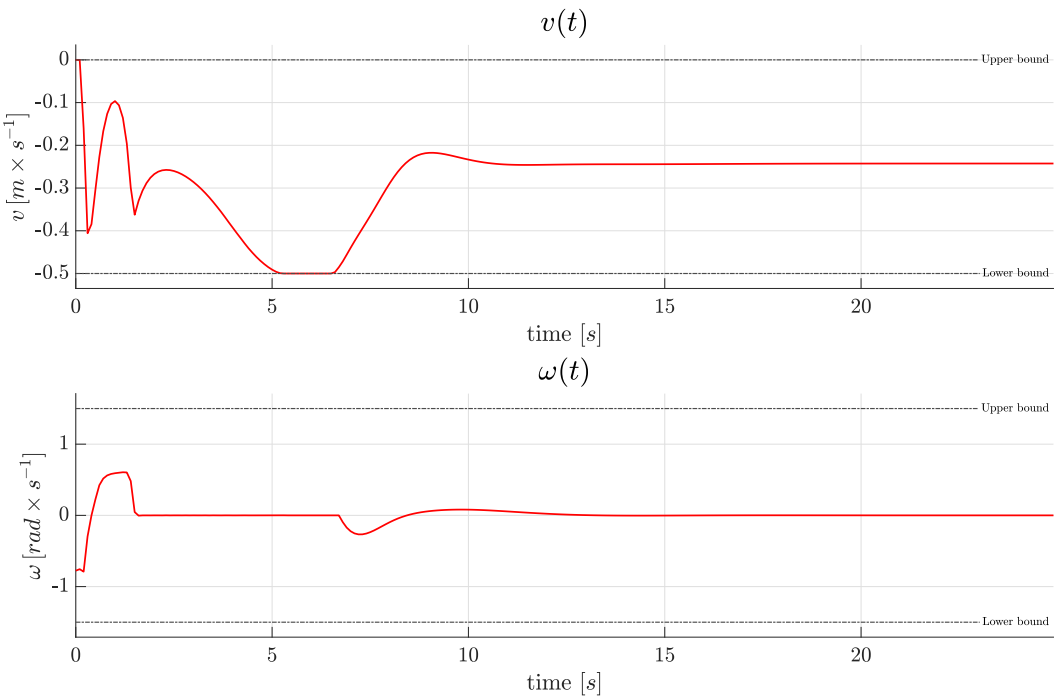

Figure 5.2. Control inputs performing circular trajectory in backward motion using the basic stabilizing terminal constraint.

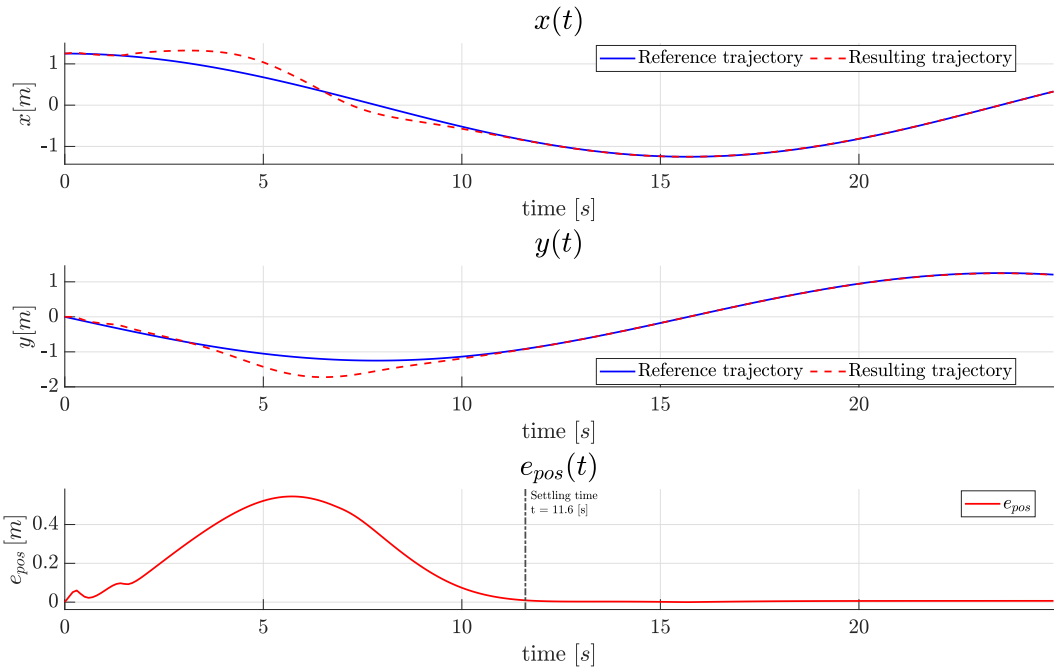

Figure 5.3. Cartesian errors performing circular trajectory in backward motion using the basic stabilizing terminal constraint.

Advanced stabilizing terminal constraint

The setup of the simulation times using the advanced stabilizing terminal constraint is the following:

Table 5.2. Misaligned starting poses simulation setup (advanced stabilizing terminal constraint).

Sampling time T	0.1 s
Control horizon	3 s
Total simulation time	25 s

- Average time for each MPC iteration: 35.40 ms.
- Average time for each solver operation: 30.88 ms.
- Total time for output error feedback linearization: 435.86 ms.
- Settling time: 12.00 s.
- Position RMSE: 0.30 m.

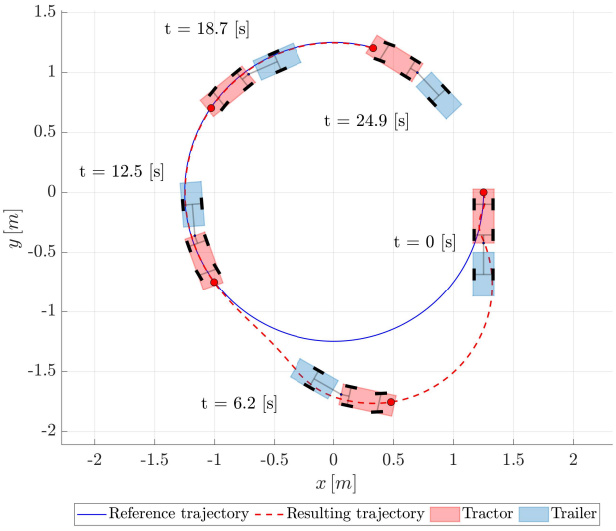

Figure 5.4. Circular trajectory in backward motion using the advanced stabilizing terminal constraint.

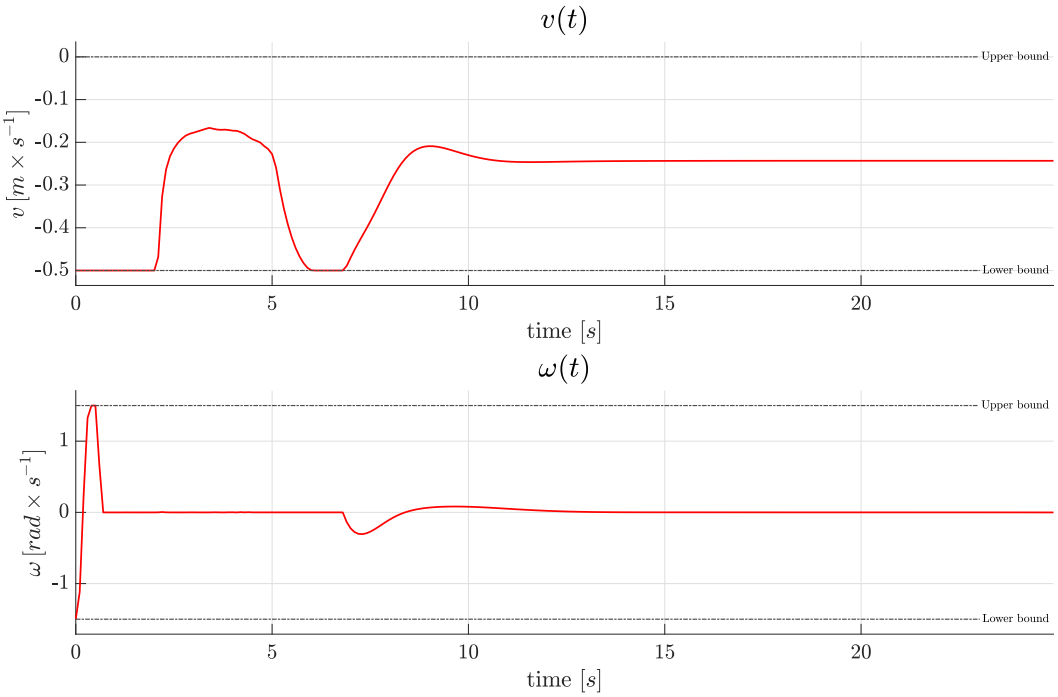

Figure 5.5. Control inputs performing circular trajectory in backward motion using the advanced stabilizing terminal constraint.

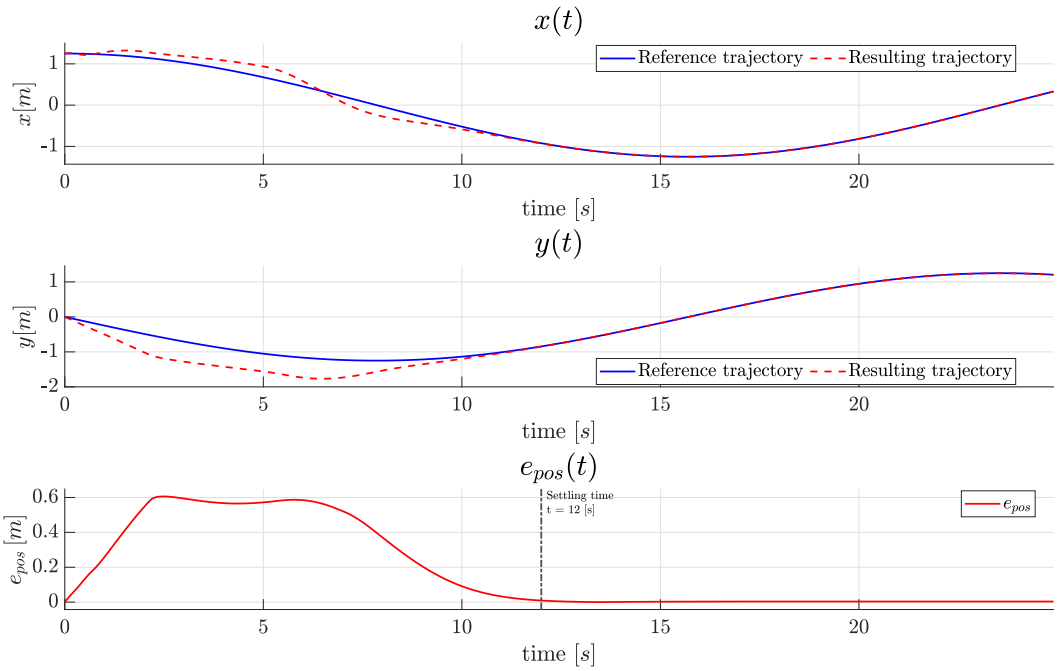

Figure 5.6. Cartesian errors performing circular trajectory in backward motion using the advanced stabilizing terminal constraint.

Analysis of the results

A circular trajectory is tracked for the tractor-trailer system using nonlinear MPC in these simulations. The initial pose of the system is misaligned with respect to the reference trajectory, and two different stabilizing terminal constraints are tested to determine which one yields better performance.

In the basic stabilizing terminal constraint simulation, the control horizon is set to 6 seconds, and the average time for each MPC iteration is 48.10 ms. The average time for each solver operation is 43.69 ms. The total time for output error feedback linearization is 478.02 ms. The system takes 11.60 seconds to settle, and the position root mean square error (RMSE) is 0.22 m. The control inputs and the trajectory's Cartesian errors are plotted, indicating that the system can track the reference trajectory.

In the advanced stabilizing terminal constraint simulation, the control horizon is set to 3 seconds, and the average time for each MPC iteration is 35.40 ms. The average time for each solver operation is 30.88 ms. The total time for output error feedback linearization is 435.86 ms. The system takes 12.00 seconds to settle, and the position RMSE is 0.30 m. The control inputs and the trajectory's Cartesian errors are plotted, indicating that the system can track the reference trajectory as well.

Comparing the two simulations, it is possible to observe that the advanced stabilizing terminal constraint performs better than the basic stabilizing terminal constraint with a lower control horizon. The advanced stabilizing terminal constraint has a similar position RMSE and settles at almost the same time as the basic stabilizing terminal constraint. Also, the control inputs' amplitude in the advanced

stabilizing terminal constraint is smaller than those of the basic stabilizing terminal constraint, which indicates that the advanced stabilizing terminal constraint leads to smoother control inputs.

Therefore, it is possible to conclude that the advanced stabilizing terminal constraint performs better than the basic stabilizing terminal constraint in the trajectory tracking of the tractor-trailer system under a misaligned starting pose.

5.1.2 Position and orientation starting errors combined

The simulation that follows involves a scenario where the vehicle starts out of the reference trajectory with significant errors in both orientation and position with respect to the reference. To be more specific, the initial poses in both scenarios are misaligned with the reference trajectory and have initial position errors, but are oriented towards it. A part of a lemniscate trajectory is used as a reference for both scenarios.

To carry out this simulation, certain setup parameters are chosen. The sampling time T is set to 0.1 seconds, the control horizon is set in order to complete the task in the best possible way, and the total simulation time is set to 35 seconds.

The simulation proceeds by first presenting the results of the basic stabilizing terminal constraint, followed by the results of the advanced stabilizing terminal constraint, and finally a complete analysis and comparison of the two simulations.

The starting configuration for both cases is:

$$q_0 = \begin{pmatrix} 0\,\mathrm{m} \\ -1\,\mathrm{m} \\ \frac{5}{4}\pi\,\mathrm{rad} \\ 0\,\mathrm{rad} \\ 0\,\mathrm{rad} \end{pmatrix}. \tag{5.2}$$

Basic stabilizing terminal constraint

The simulation setup is:

Table 5.3. Position and orientation starting errors simulation setup (basic stabilizing terminal constraint).

Sampling time T	0.1 s
Control horizon	6 s
Total simulation time	35 s

- Average time for each MPC iteration: 38.23 ms.
- Average time for each solver operation: 33.44 ms.
- Total time for output error feedback linearization: 664.93 ms.
- Settling time: 30.90 s.
- Position RMSE: 0.27 m.

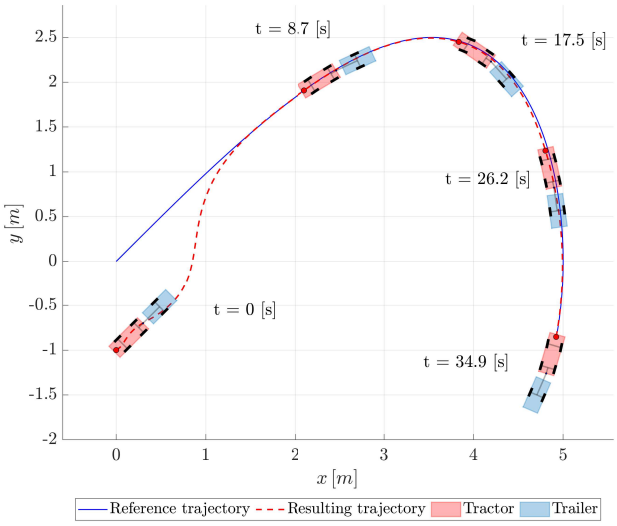

Figure 5.7. Lemniscate trajectory in backward motion using the basic stabilizing terminal constraint.

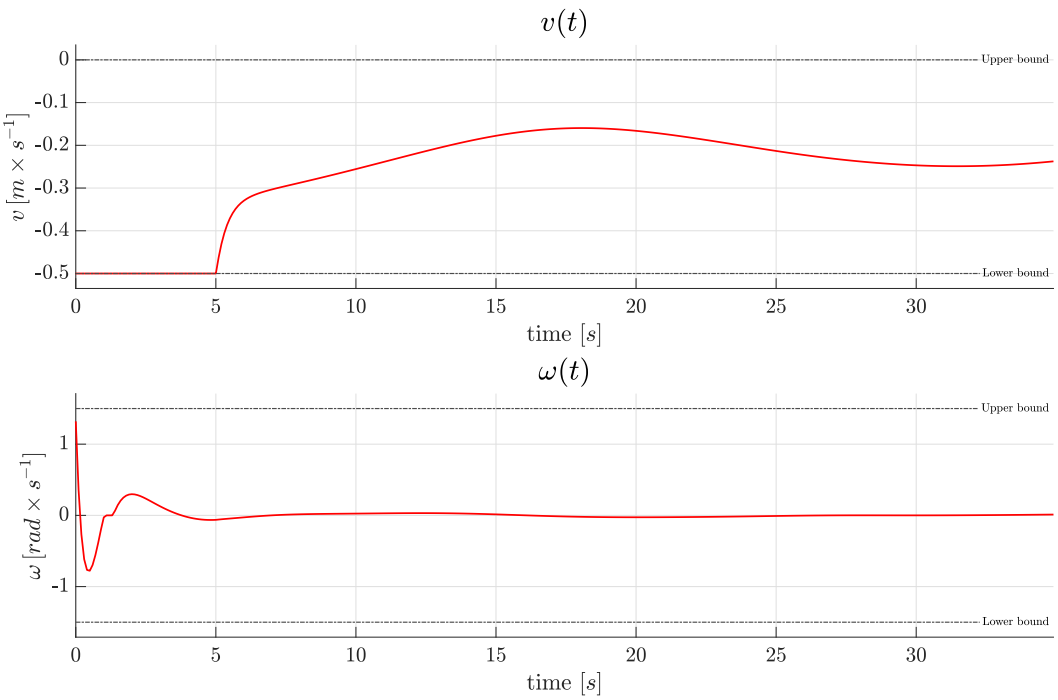

Figure 5.8. Control inputs performing lemniscate trajectory in backward motion using the basic stabilizing terminal constraint.

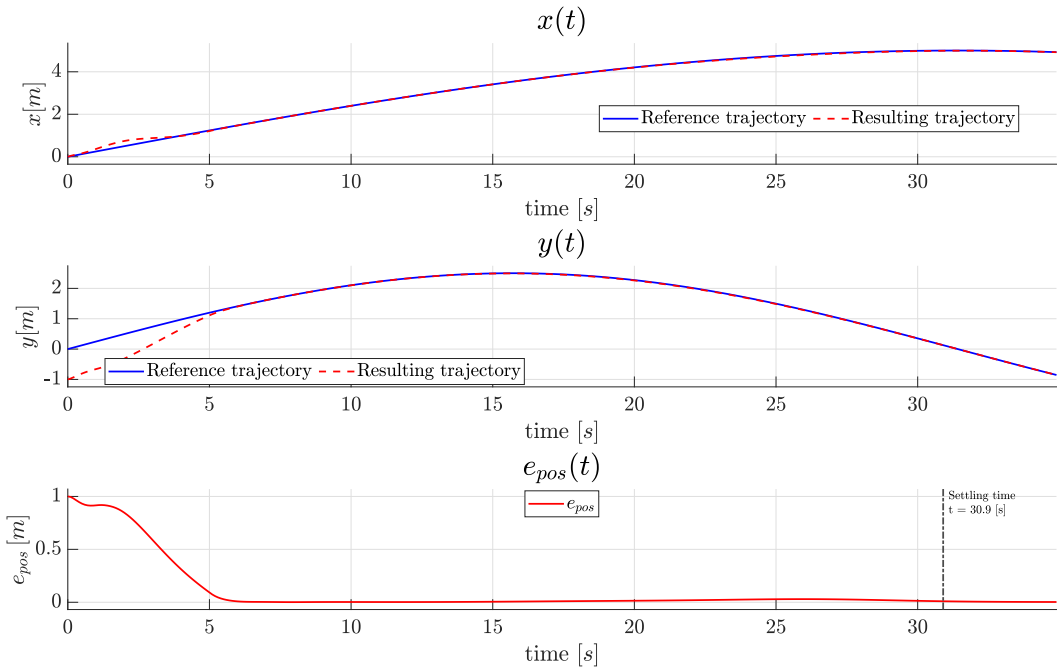

Figure 5.9. Cartesian errors performing lemniscate trajectory in backward motion using the basic stabilizing terminal constraint.

Advanced stabilizing terminal constraint

The simulation setup is:

Table 5.4. Position and orientation starting errors simulation setup (advanced stabilizing terminal constraint).

Sampling time T	0.1 s
Control horizon	3 s
Total simulation time	35 s

- Average time for each MPC iteration: 25.58 ms.
- Average time for each solver operation: 21.24 ms.
- Total time for output error feedback linearization: 598.85 ms.
- Settling time: 6.50 s.
- Position RMSE: 0.27 m.

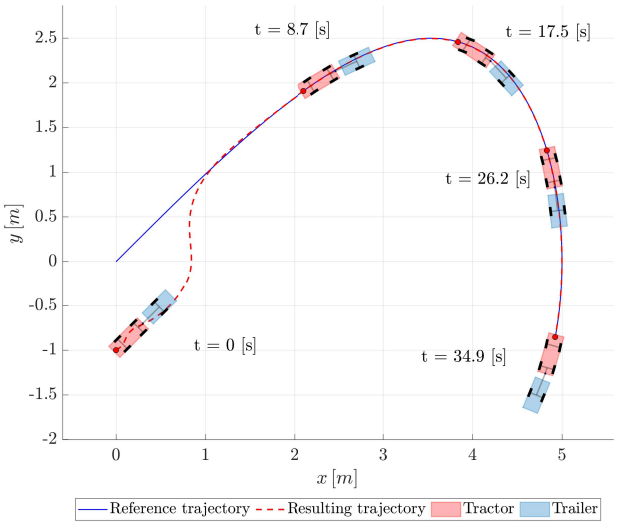

Figure 5.10. Lemniscate trajectory in backward motion using the advanced stabilizing terminal constraint.

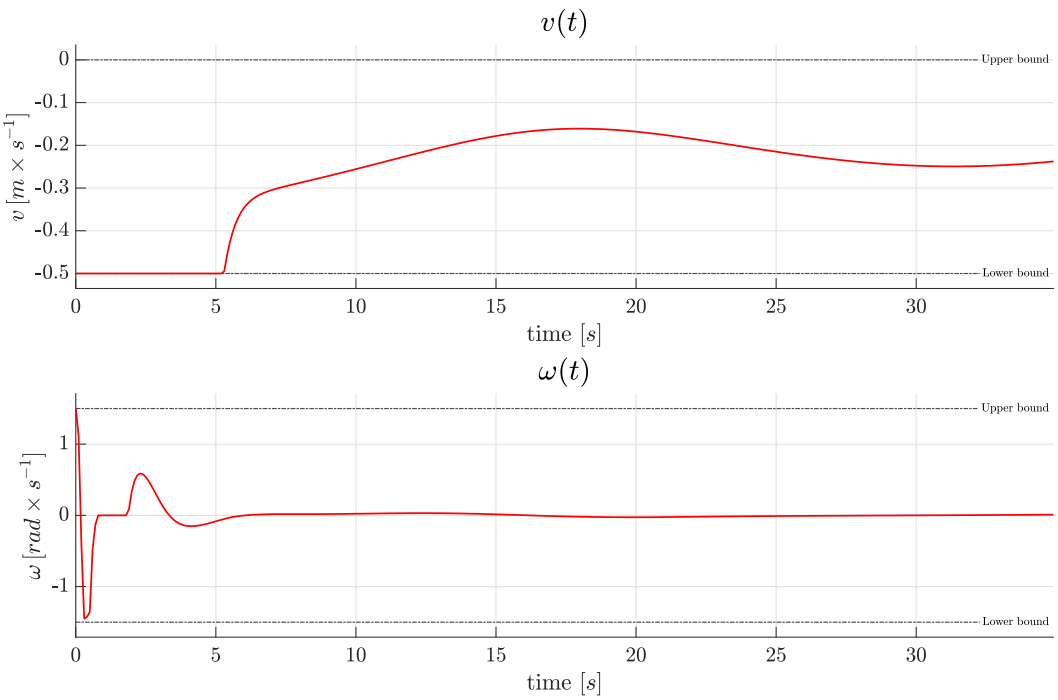

Figure 5.11. Control inputs performing lemniscate trajectory in backward motion using the advanced stabilizing terminal constraint.

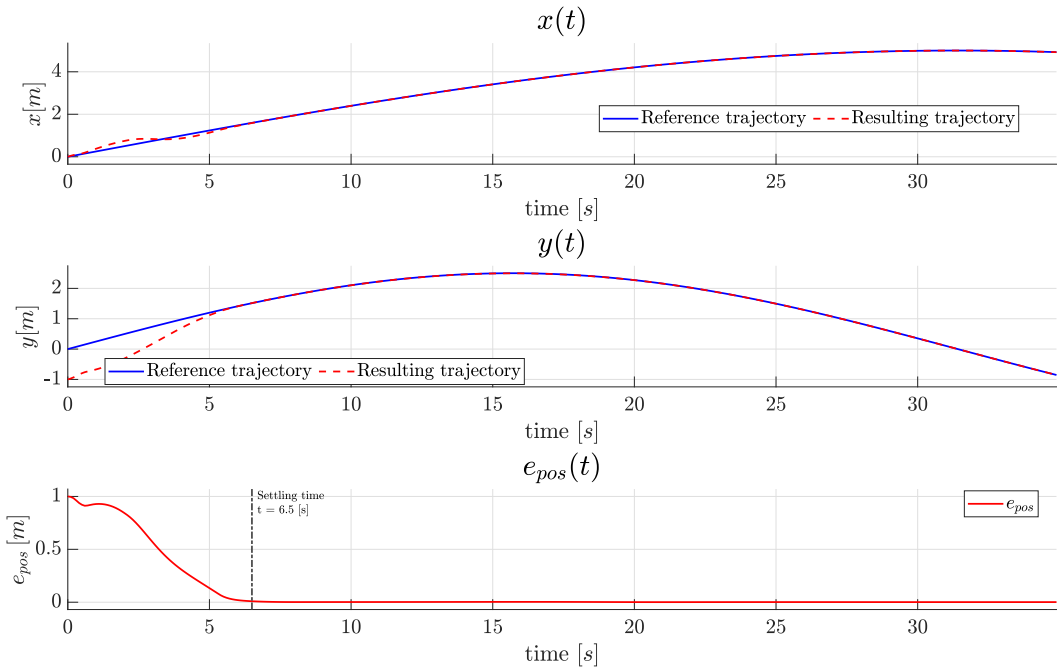

Figure 5.12. Cartesian errors performing lemniscate trajectory in backward motion using the advanced stabilizing terminal constraint.

Analysis of the results

The simulations presented above are about the trajectory tracking of the tractor-trailer system starting with significant errors in both orientation and position with respect to the reference trajectory. In both scenarios, a part of a lemniscate trajectory is used as a reference.

The first simulation uses the basic stabilizing terminal constraint, and the second one uses the advanced stabilizing terminal constraint. The basic stabilizing terminal constraint requires a longer control horizon to complete the task, while the advanced stabilizing terminal constraint needs a shorter control horizon.

Regarding the results of the simulation with the basic stabilizing terminal constraint, the average time for each MPC iteration is 38.23 ms, and for each solver operation, it is 33.44 ms. The total time for output error feedback linearization is 664.93 ms. The settling time is 30.90 s, and the position RMSE is 0.27 m.

On the other hand, the simulation with the advanced stabilizing terminal constraint presents a lower average time for each MPC iteration and solver operation, which are 25.58 ms and 21.24 ms, respectively. The total time for output error feedback linearization is 598.85 ms. The settling time is significantly shorter than the previous simulation, being only 6.50 s. The position RMSE is the same as the previous one, with a value of 0.27 m.

Comparing both simulations, it can be observed that the advanced stabilizing terminal constraint outperforms the basic stabilizing terminal constraint. The former achieves a shorter settling time and requires less computational time per iteration and per solver operation. These benefits come with a shorter control horizon, which is a big plus for the advanced stabilizing terminal constraint. Overall, the advanced

stabilizing terminal constraint provides a better solution for the trajectory tracking problem.

5.1.3 Shorter control horizons

This section presents a new simulation approach that compares the two types of stabilizing terminal constraints. The performance of both cases is analyzed on circular and lemniscate trajectories to determine the most efficient setup for executing the nonlinear MPC. To achieve faster execution times, the stabilization part regarding the output error feedback linearization is carried out offline, while the control horizon is reduced to 1 s, this allows that the execution times of the entire algorithm are significantly reduced, minimizing also the effort of the architecture used in this test phase.

Such an important speeding up of the MPC execution times, gives the possibility to further lower the sampling time (the lower the sampling time, the higher the calculation time required by the solver), so as to have a trade-off of simulation times extremely low.

In this case, the simulation involves a tractor-trailer system that begins from an initial position that is already on the reference trajectory. This initial position is either perfectly aligned with the reference trajectory or is extremely close to it. This is done to ensure that the vehicle velocity vector is pointing towards the reference trajectory and the system can quickly converge with it without aggressive corrections.

Basic stabilizing terminal constraint on a circular trajectory

Simulation setup:

Table 5.5. Circular trajectory tracking setup.

Sampling time T	0.05 s
Control horizon	1 s
Total simulation time	32 s

Starting configuration:

$$q_0 = \begin{pmatrix} 1.75\,\text{m} \\ 0\,\text{m} \\ \frac{\pi}{4}\,\text{rad} \\ 0\,\text{rad} \\ 0\,\text{rad} \end{pmatrix}. \tag{5.3}$$

- Average time for each MPC iteration: 23.01 ms.
- Average time for each solver operation: 19.51 ms.
- Total time for output error feedback linearization: 919.59 ms.
- Settling time: `NaN`.

- Position RMSE: 0.09 m.

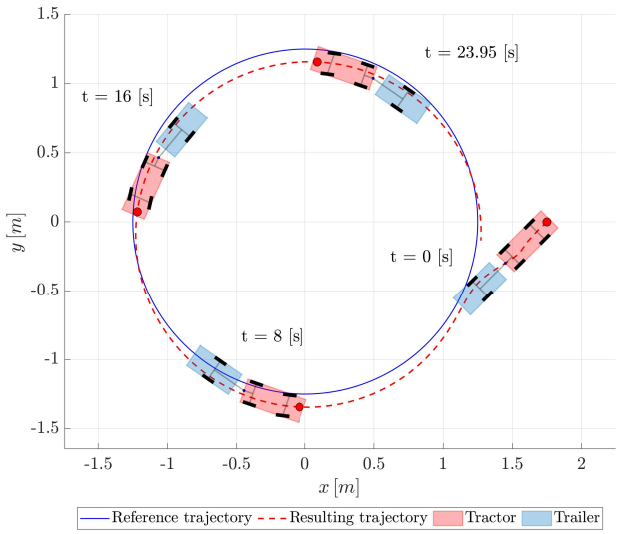

Figure 5.13. Circular trajectory in backward motion using the basic stabilizing terminal constraint.

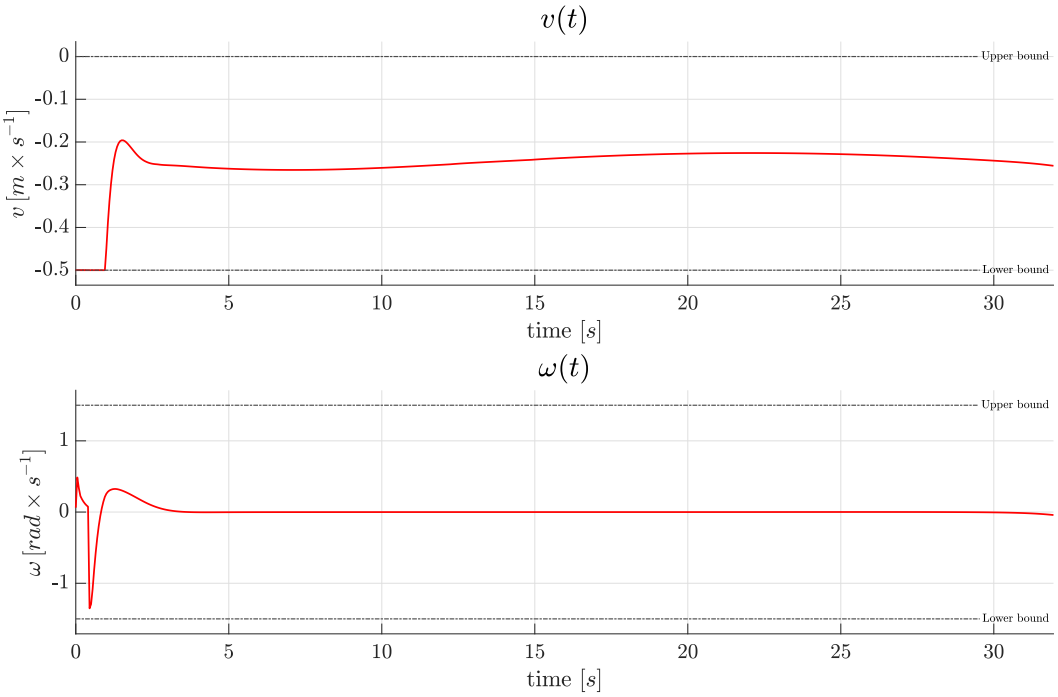

Figure 5.14. Control inputs performing circular trajectory in backward motion using the basic stabilizing terminal constraint.

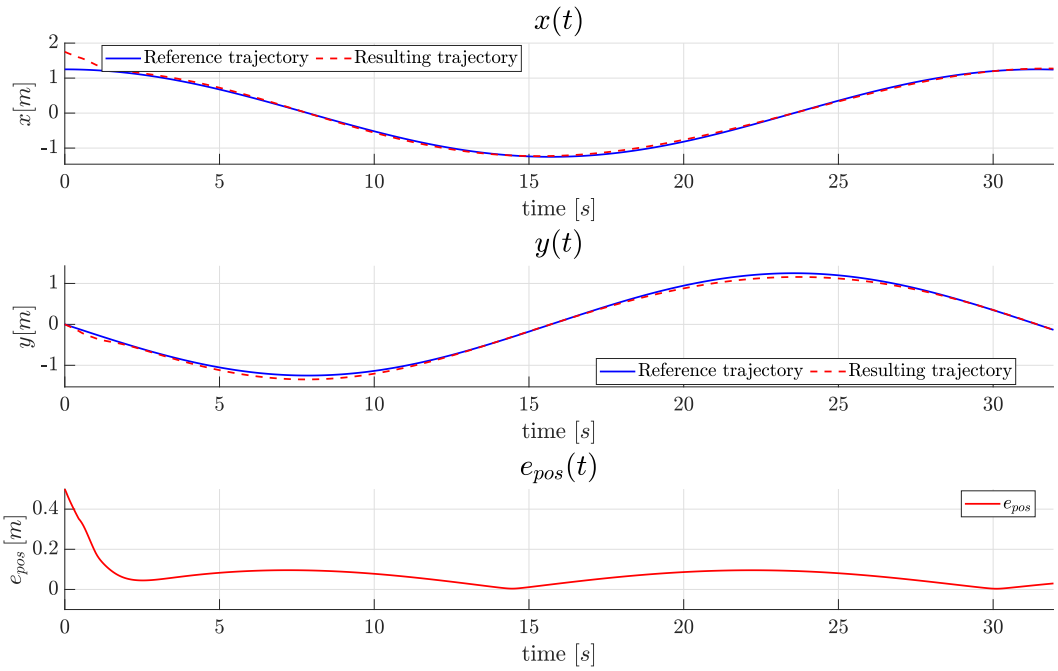

Figure 5.15. Cartesian errors performing circular trajectory in backward motion using the basic stabilizing terminal constraint.

Advanced stabilizing terminal constraint on a circular trajectory

The simulation setup is the same as the simulation regarding the basic version, and also the starting configuration is the same:

$$q_0 = \begin{pmatrix} 1.75\,\text{m} \\ 0\,\text{m} \\ \frac{\pi}{4}\,\text{rad} \\ 0\,\text{rad} \\ 0\,\text{rad} \end{pmatrix}. \tag{5.4}$$

- Average time for each MPC iteration: 20.74 ms.
- Average time for each solver operation: 17.03 ms.
- Total time for output error feedback linearization: 908.21 ms.
- Settling time: 9.90 s.
- Position RMSE: 0.07 m.

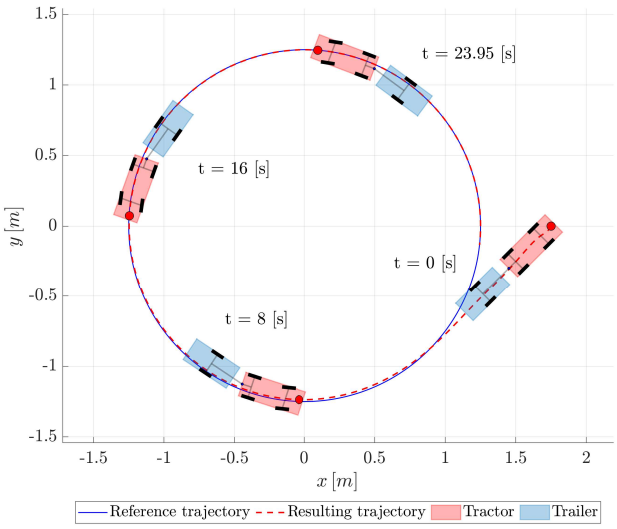

Figure 5.16. Circular trajectory in backward motion using the advanced stabilizing terminal constraint.

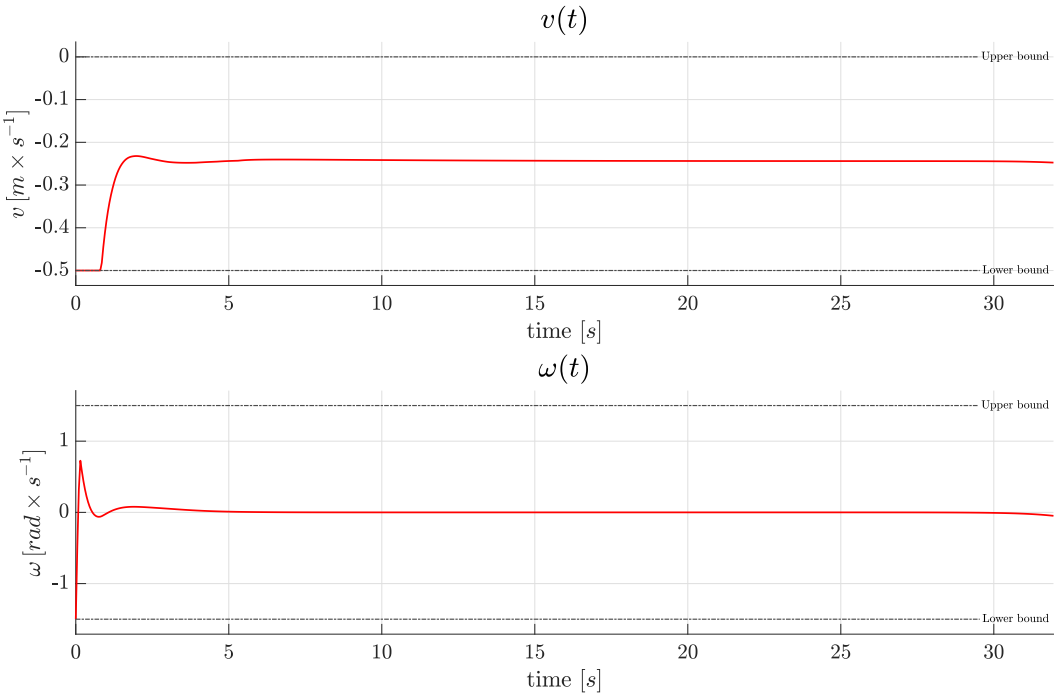

Figure 5.17. Control inputs performing circular trajectory in backward motion using the advanced stabilizing terminal constraint.

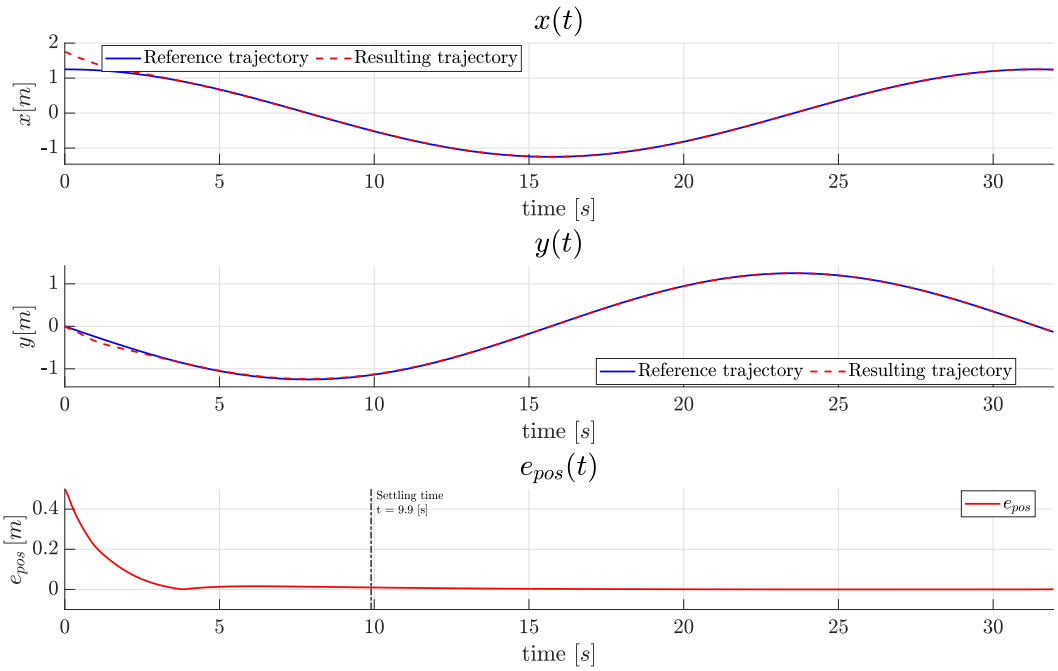

Figure 5.18. Cartesian errors performing circular trajectory in backward motion using the advanced stabilizing terminal constraint.

Basic stabilizing terminal constraint on a lemniscate trajectory

Table 5.6. Lemniscate trajectory tracking setup.

Sampling time T	0.05 s
Control horizon	1 s
Total simulation time	110 s

Starting configuration:

$$q_0 = \begin{pmatrix} 0\,\text{m} \\ 0\,\text{m} \\ \frac{5}{4}\pi\,\text{rad} \\ 0\,\text{rad} \\ 0\,\text{rad} \end{pmatrix}. \tag{5.5}$$

- Average time for each MPC iteration: 22.05 ms.
- Average time for each solver operation: 18.35 ms.
- Total time for output error feedback linearization: 2595.79 ms.
- Settling time: 98.20 s.
- Position RMSE: 0.01 m.

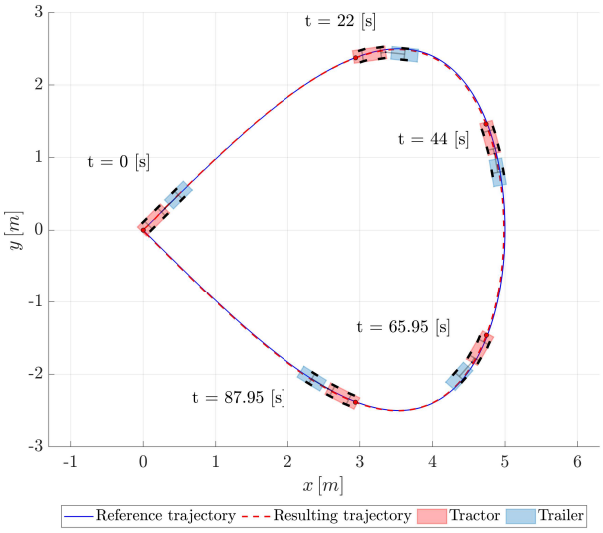

Figure 5.19. Lemniscate trajectory in backward motion using the basic stabilizing terminal constraint.

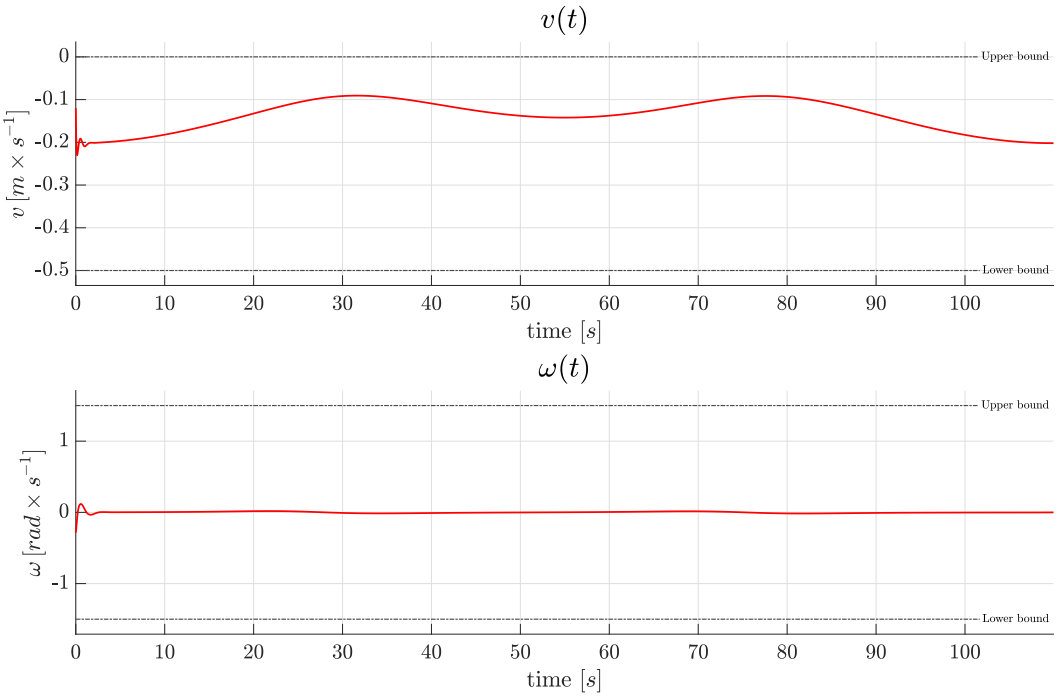

Figure 5.20. Control inputs performing lemniscate trajectory in backward motion using the basic stabilizing terminal constraint.

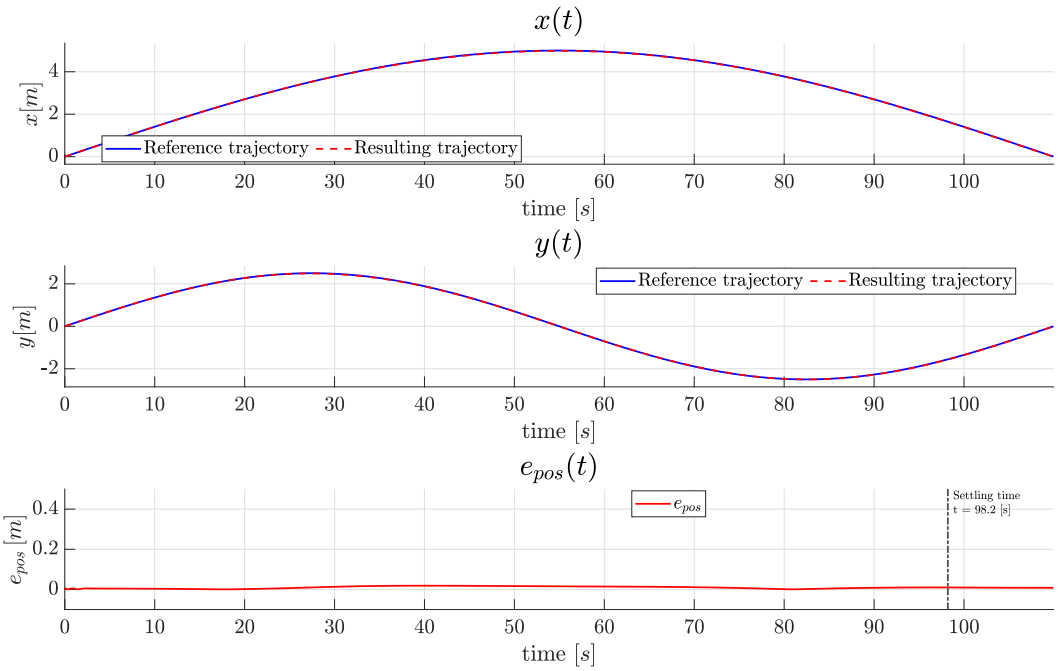

Figure 5.21. Cartesian errors performing lemniscate trajectory in backward motion using the basic stabilizing terminal constraint.

Advanced stabilizing terminal constraint on a lemniscate trajectory

The simulation setup is the same as the previous case, and also the starting configuration is the same:

$$q_0 = \begin{pmatrix} 0\,\text{m} \\ 0\,\text{m} \\ \frac{5}{4}\pi\,\text{rad} \\ 0\,\text{rad} \\ 0\,\text{rad} \end{pmatrix}. \tag{5.6}$$

- Average time for each MPC iteration: 22.27 ms.
- Average time for each solver operation: 18.59 ms.
- Total time for output error feedback linearization: 2619.17 ms.
- Settling time: 2.55 s.
- Position RMSE: 0.00 m.

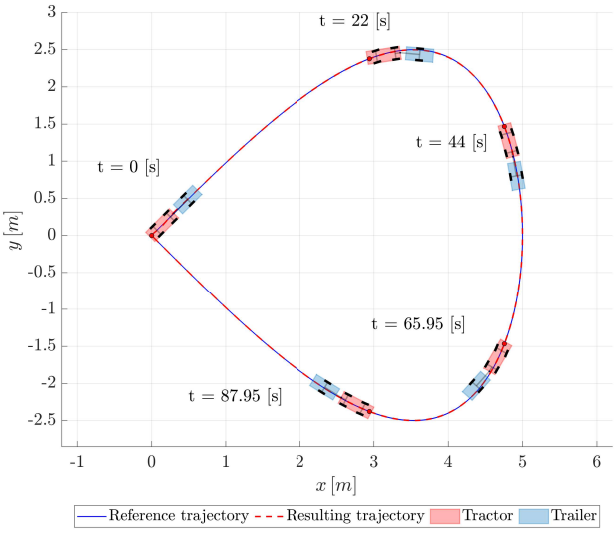

Figure 5.22. Lemniscate trajectory in backward motion using the advanced stabilizing terminal constraint.

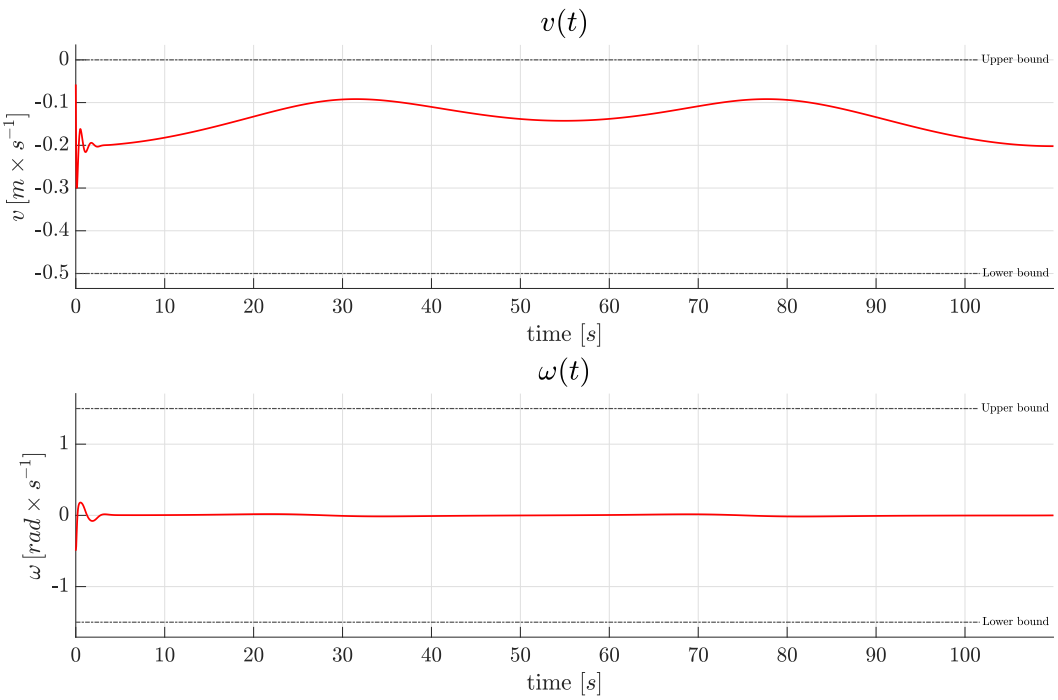

Figure 5.23. Control inputs performing lemniscate trajectory in backward motion using the advanced stabilizing terminal constraint.

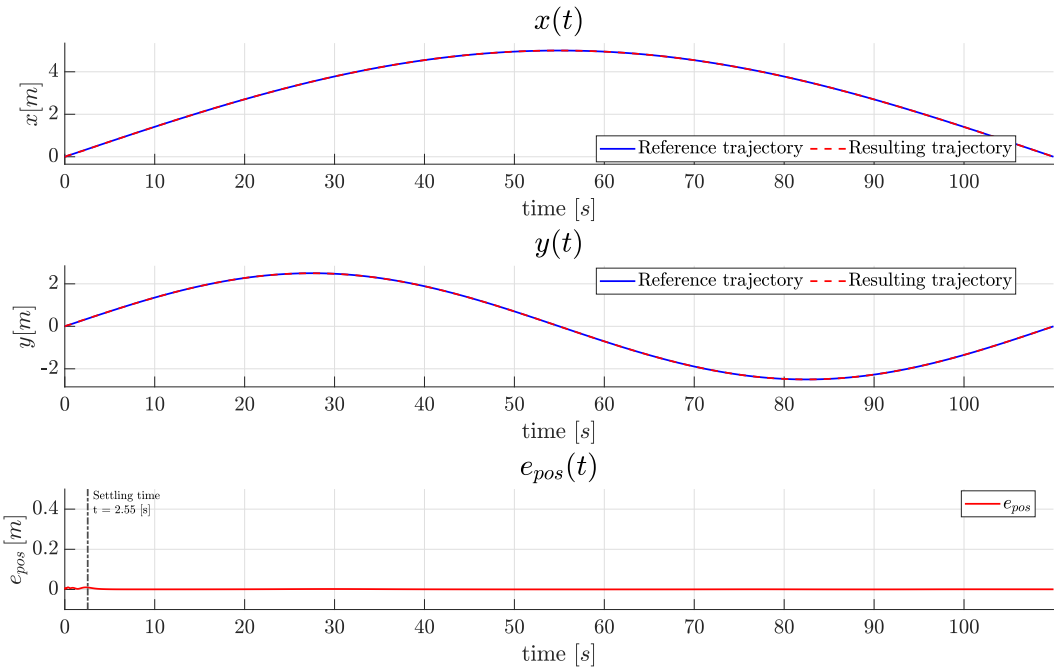

Figure 5.24. Cartesian errors performing lemniscate trajectory in backward motion using the advanced stabilizing terminal constraint.

Analysis of the results

The given results describe the performance of the nonlinear MPC algorithm with the two types of stabilizing terminal constraints: basic and advanced. The algorithm is tested to track circular and lemniscate trajectories.

For the circular trajectory with the basic stabilizing terminal constraint, the average time for each MPC iteration is 23.01 ms, and the average time for each solver operation is 19.51 ms. The total time for output error feedback linearization is 919.59 ms, and the position RMSE is 0.09 m. However, the settling time is `NaN`, indicating that the system did not converge towards the reference trajectory in a finite time.

For the circular trajectory with the advanced stabilizing terminal constraint, the average time for each MPC iteration is 20.74 ms and the average time for each solver operation is 17.03 ms. The total time for output error feedback linearization is 908.21 ms, and the position RMSE is 0.07 m. The settling time is 9.90 s, which indicated that the system reached the reference trajectory in a finite time.

Comparing the two simulations, it is evident that the advanced stabilizing terminal constraint results in a lower RMSE and settling time, indicating a more accurate tracking of the reference trajectory. The average time for each MPC iteration and solver operation is also lower for the advanced constraint, which implies a faster execution time.

In the third simulation, a lemniscate trajectory is tracked. The average time for each MPC iteration is 22.05 ms, and the average time for each solver operation is 18.35 ms. The total time for output error feedback linearization is 2595.79 ms. The settling time is 98.20 seconds, and the position RMSE is 0.01 m.

In the fourth simulation, the same lemniscate trajectory is used with the same starting configuration as the third simulation. The advanced stabilizing terminal constraint is used in this case. The control horizon and the total simulation time are the same as the previous simulation.

The average time for each MPC iteration is 22.27 ms, and the average time for each solver operation is 18.59 ms. The total time for output error feedback linearization is 2619.17 ms. The settling time is 2.55 s, and the position RMSE is 0.00 m.

Comparing the last two simulations, it can be observed that the advanced stabilizing terminal constraint yields better results, achieving a shorter settling time and a smaller position RMSE.

Based on the results, it can be concluded that the advanced stabilizing terminal constraint performs better than the basic one, with lower settling times and similar or even better position RMSE. Additionally, the execution times are comparable between the two types of stabilizing terminal constraints. Hence, the advanced stabilizing terminal constraint is more suitable for every setup of the tractor-trailer system and every trajectory with respect to the basic one.

5.2 Complete trajectories

This section of the study involves performing simulations on complete trajectories in order to assess the effectiveness of the control logic and the robustness of the nonlinear Model Predictive Control in maintaining system stability over a prolonged period of time. The objective is to determine if the nonlinear MPC can successfully cover an entire trajectory while keeping the system stable.

Also in this case the simulation involves a vehicle that begins from an initial position that is already on the reference trajectory. This initial position is either perfectly aligned with the reference trajectory or is extremely close to it. This is done to ensure that the vehicle is pointing towards the reference trajectory and can quickly converge with it. In particular, the initial conditions in terms of simulation setup and initial configuration are the same as the reference paper [8], with the same trajectories to track (apart from the rectangular trajectory), in order to understand whether the work in question is actually able to perform the same type of prolonged task under the same conditions.

The aim of the simulation is to evaluate the ability of the nonlinear MPC to maintain the stability of the system over the entire length of the trajectory.

5.2.1 Linear trajectory

The simulation involves a vehicle whose initial position and orientation are almost perfectly aligned with a reference trajectory. In this simulation, an advanced stabilizing terminal constraint is utilized, and the reference trajectory used is linear.

The setup for the simulation includes a sampling time of 0.1 seconds, a control horizon of 5 second, and a total simulation time of 20 seconds.

Therefore:

Table 5.7. Linear trajectory tracking setup.

Sampling time T	0.1 s
Control horizon	5 s
Total simulation time	20 s

following the reference trajectory given by the equations:

$$\begin{aligned} x^r(t) &= 6 - 0.3t \\ y^r(t) &= 0, \end{aligned} \tag{5.7}$$

starting with:

$$q_0 = \begin{pmatrix} 6\,\text{m} \\ 0.01\,\text{m} \\ 0\,\text{rad} \\ 0\,\text{rad} \\ 0\,\text{rad} \end{pmatrix}. \tag{5.8}$$

- Average time for each MPC iteration: 27.74 ms.
- Average time for each solver operation: 23.11 ms.
- Total time for output error feedback linearization: 494.27 ms.
- Settling time: 3.60 s.
- Position RMSE: 0.00 m.
- Peak Cartesian error: 0.01 m.

The simulation achieves good results, with an average time for each MPC iteration of 27.74 milliseconds and an average time for each solver operation of 23.11 milliseconds. The total time for output error feedback linearization is 494.27 milliseconds, and the settling time is 3.60 seconds.

Furthermore, the position RMSE is 0.00 meters, indicating that the position of the system closely matched the reference trajectory. The peak Cartesian error is 0.01 meters, indicating that the maximum deviation between the actual trajectory and the reference trajectory is very small.

In conclusion, the simulation successfully demonstrates that the tractor-trailer system can accurately track a linear reference trajectory, with the system quickly settling to the reference trajectory and closely matching it throughout the simulation.

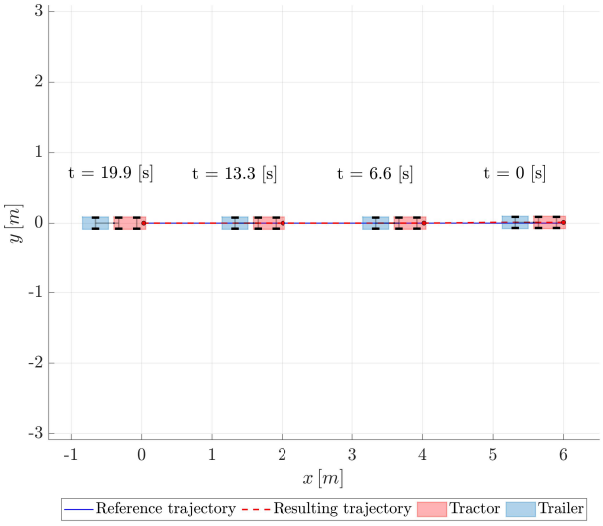

Figure 5.25. Linear trajectory in backward motion.

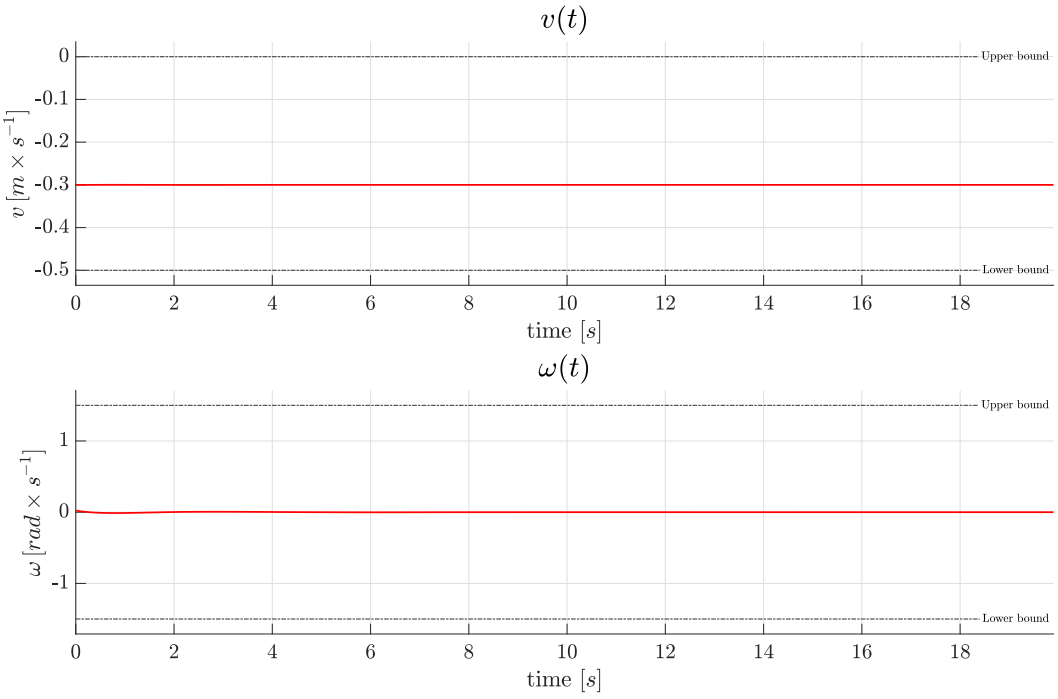

Figure 5.26. Control inputs performing linear trajectory in backward motion.

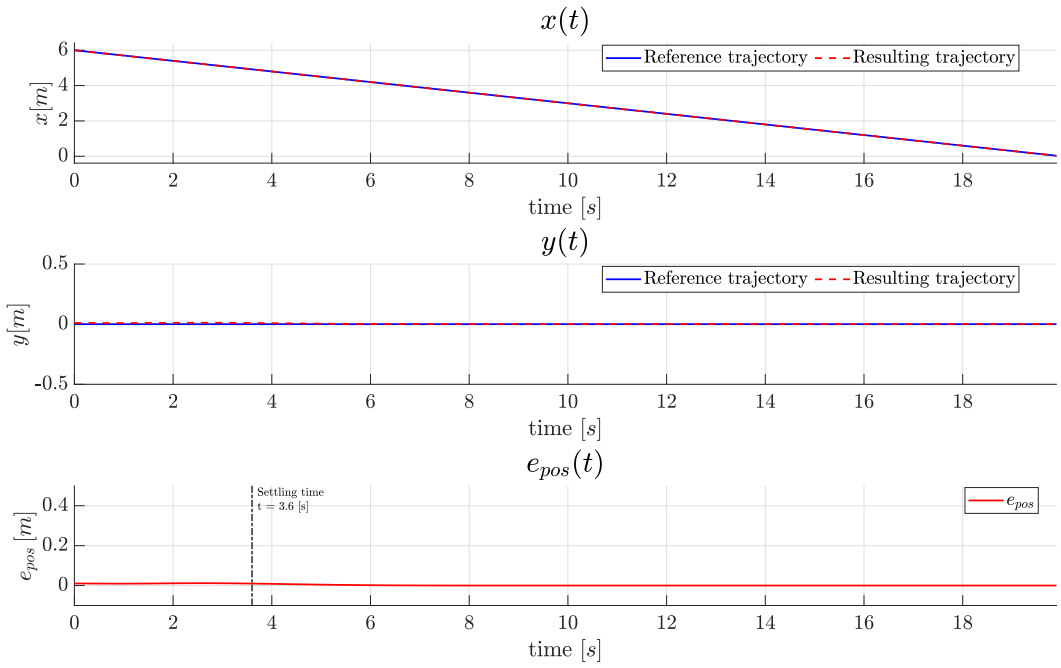

Figure 5.27. Cartesian errors performing linear trajectory in backward motion.

5.2.2 Circular trajectory

The simulation involves setting up the initial position and orientation of a vehicle extremely close to a circular reference trajectory. The aim is to ensure that the vehicle is facing towards the reference trajectory and can quickly converge with it. During the simulation, the advanced stabilizing terminal constraint is utilized.

To carry out the simulation, a specific setup is employed. The sampling time T is set at 0.1 s, and the control horizon is set at 5 s. The total simulation time is 130 s.

Therefore, the simulation setup is:

Table 5.8. Circular trajectory tracking setup.

Sampling time T	0.1 s
Control horizon	5 s
Total simulation time	130 s

following the reference trajectory given by the equations:

$$\begin{aligned} x^r(t) &= 5\,\cos\left(-\frac{1}{20}t\right) \\ y^r(t) &= 5\,\sin\left(-\frac{1}{20}t\right) + 5, \end{aligned} \tag{5.9}$$

with starting condition:

$$q_0 = \begin{pmatrix} 4.99\,\text{m} \\ 5\,\text{m} \\ 1.5\,\text{rad} \\ 0.03\,\text{rad} \\ 0.05\,\text{rad} \end{pmatrix}. \tag{5.10}$$

- Average time for each MPC iteration: 46.40 ms.
- Average time for each solver operation: 39.34 ms.
- Total time for output error feedback linearization: 3402.08 ms.
- Settling time: 7.80 s.
- Position RMSE: 0.01 m.
- Peak Cartesian error: 0.08 m.

The simulation results show that the average time for each MPC iteration is 46.40 ms, and the average time for each solver operation is 39.34 ms. The total time for output error feedback linearization is 3402.08 ms. The settling time for the system is 7.80 s. The position RMSE is 0.01 m, and the peak Cartesian error is 0.08 m.

The settling time indicates that the system converges to the reference trajectory within 7.80 s after the start of the simulation. The position RMSE shows that the position of the vehicle during the simulation is very close to the reference trajectory, with an average deviation of only 0.01 m. The peak Cartesian error shows that the maximum deviation of the vehicle from the reference trajectory is 0.08 m, indicating that the system is accurately tracking the reference trajectory.

The average time for each MPC iteration and each solver operation is relatively low, indicating that the control algorithm is efficient.

In conclusion, the simulation shows that the tractor-trailer system can accurately track a circular reference trajectory with a small deviation from the reference trajectory.

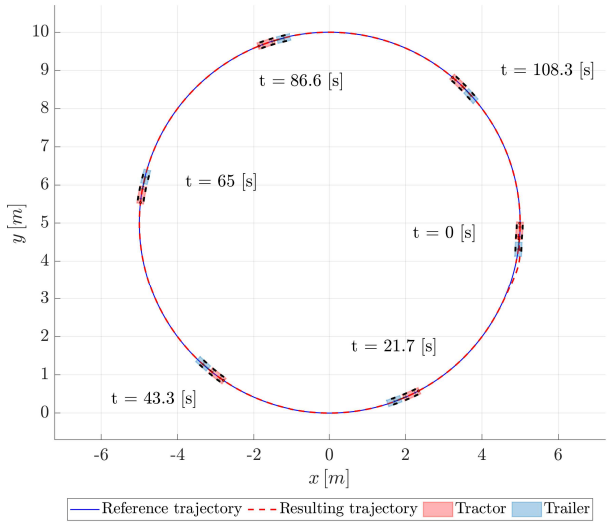

Figure 5.28. Circular trajectory in backward motion.

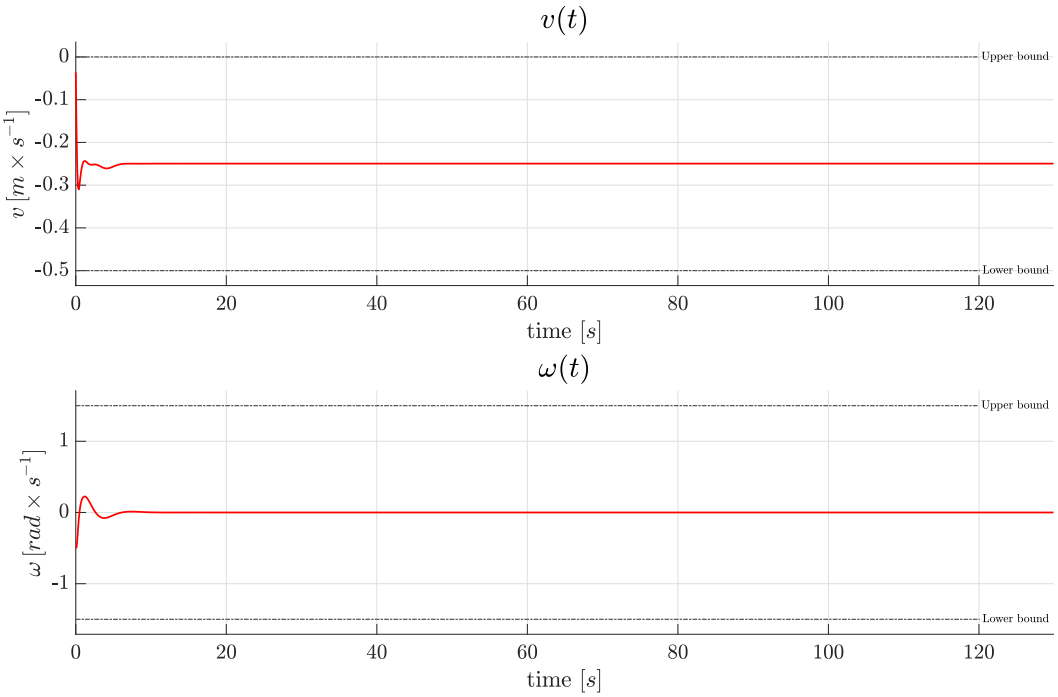

Figure 5.29. Control inputs performing circular trajectory in backward motion.

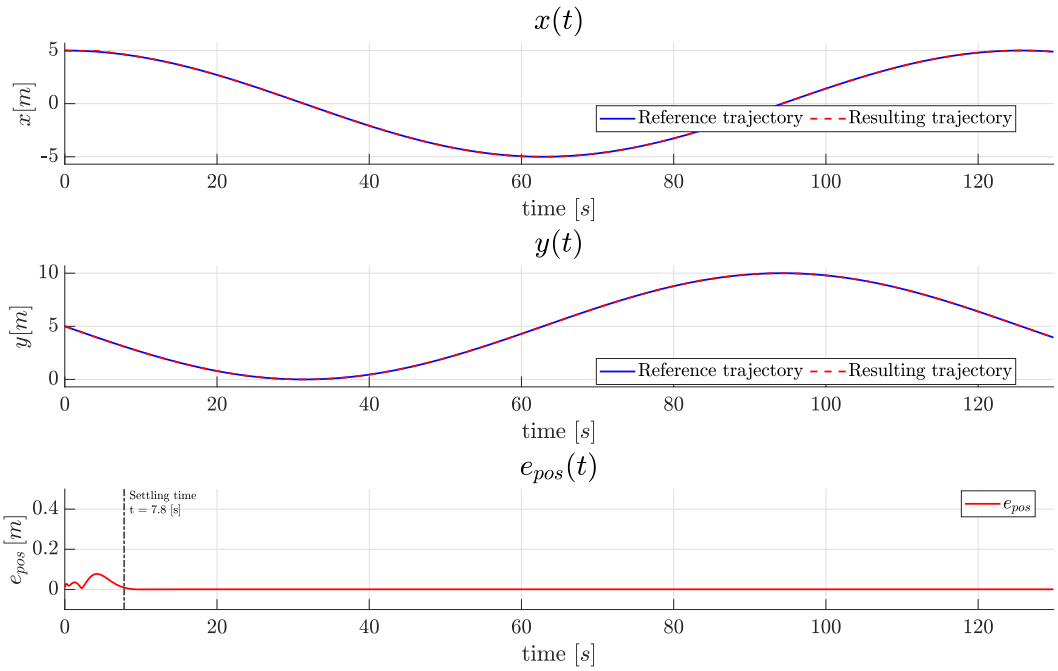

Figure 5.30. Cartesian errors performing circular trajectory in backward motion.

5.2.3 Lemniscate trajectory

The simulation that follows entails starting with the initial position and orientation of the vehicle in an almost perfect alignment with the reference trajectory, which is a lemniscate. To ensure stability during the simulation, the advanced stabilizing terminal constraint is employed. In order to track the lemniscate trajectory, the setup of the simulation times is as follows: the sampling time is set to 0.1 seconds, the control horizon is set to 5 seconds, and the total simulation time is set to 220 seconds.

Therefore, the simulation setup is:

Table 5.9. Lemniscate trajectory tracking setup.

Sampling time T	0.1 s
Control horizon	5 s
Total simulation time	220 s

following the reference trajectory given by the equations:

$$\begin{aligned} x^r(t) &= 5 \sin\left(\frac{1}{35}t\right) \\ y^r(t) &= 5 \sin\left(\frac{1}{35}t\right) \cos\left(\frac{1}{35}t\right), \end{aligned} \tag{5.11}$$

with starting configuration:

$$q_0 = \begin{pmatrix} 0\,\text{m} \\ 0.05\,\text{m} \\ 3.92\,\text{rad} \\ -0.09\,\text{rad} \\ 0\,\text{rad} \end{pmatrix}. \tag{5.12}$$

- Average time for each MPC iteration: 36.82 ms.
- Average time for each solver operation: 31.42 ms.
- Total time for output error feedback linearization: 3119.22 ms.
- Settling time: 8.10 s.
- Position RMSE: 0.01 m.
- Peak Cartesian error: 0.06 m.

The results of the simulation indicated that the average time for each MPC iteration is 36.82 milliseconds, while the average time for each solver operation is 31.42 milliseconds. The total time for output error feedback linearization is 3119.22 milliseconds. The simulation settles within 8.10 seconds and the position RMSE is 0.01 meters. Additionally, the peak Cartesian error is 0.06 meters.

These results indicate that the tractor-trailer system is able to successfully track the lemniscate trajectory with high accuracy. The settling time is relatively short, and the position RMSE is quite small, indicating that the system is able to closely follow the reference trajectory. The peak Cartesian error is also relatively small, indicating that the system does not deviate significantly from the reference trajectory at any point during the simulation.

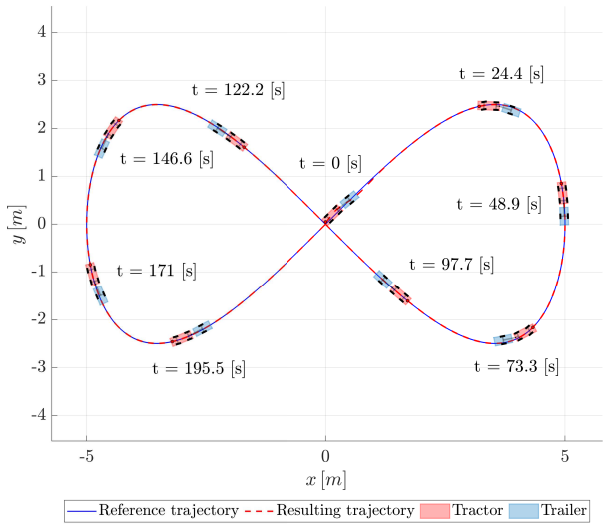

Figure 5.31. Lemniscate trajectory in backward motion.

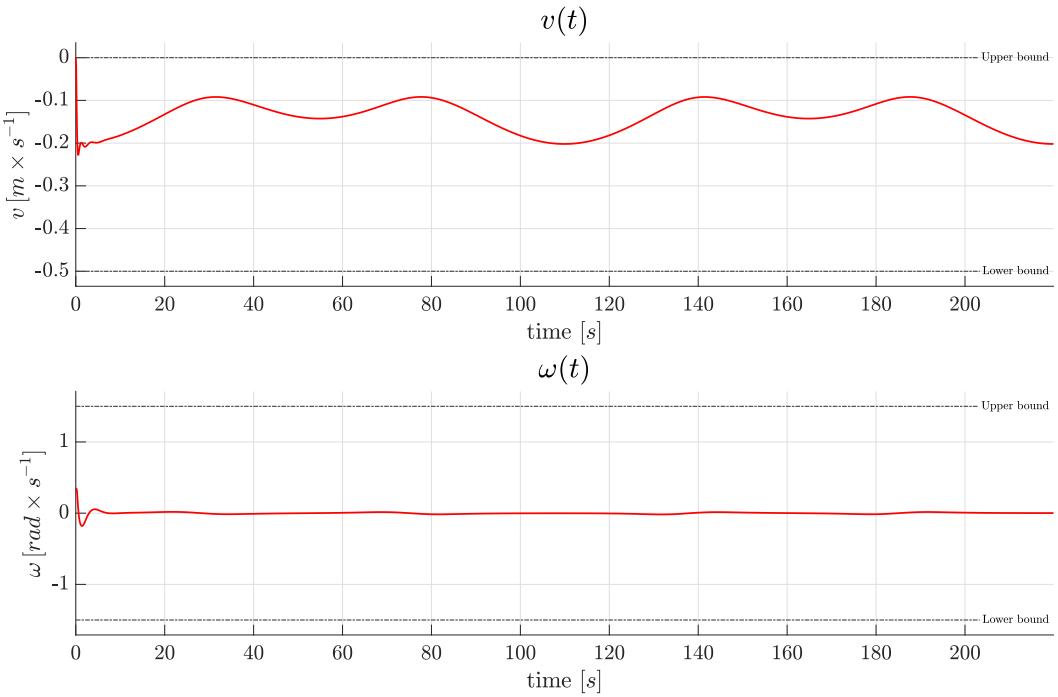

Figure 5.32. Control inputs performing lemniscate trajectory in backward motion.

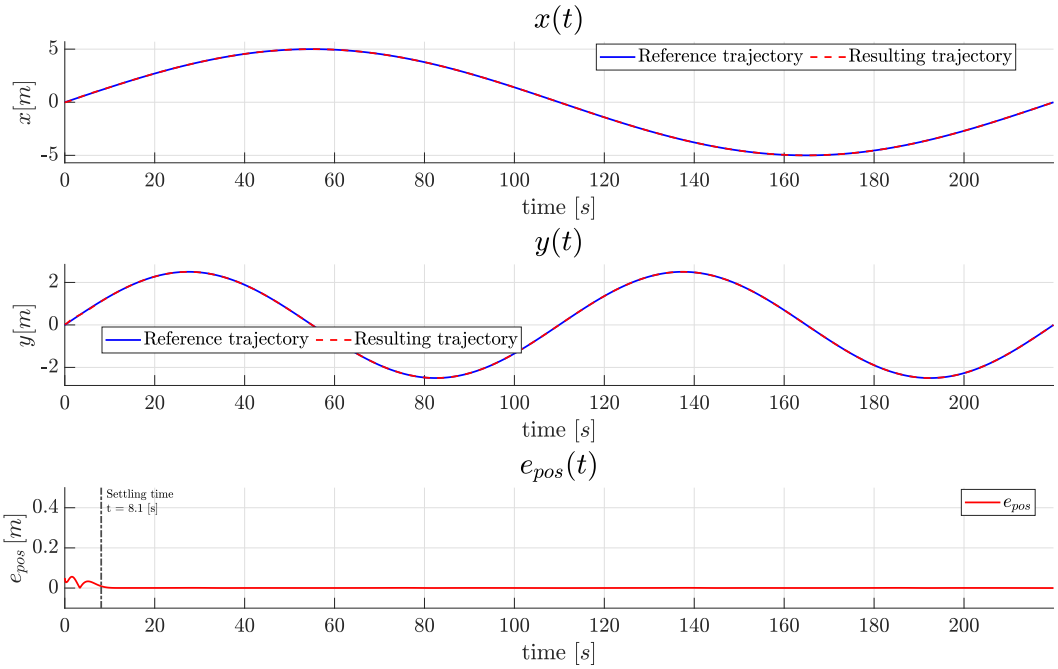

Figure 5.33. Cartesian errors performing lemniscate trajectory in backward motion.

5.2.4 Rectangular trajectory

The simulation that is set up is one in which the tractor-trailer travels along a rectangular trajectory. This particular trajectory has proven to be tricky and

difficult to navigate accurately. To overcome this difficulty and to ensure faster execution, there is a relaxation of the simulation parameters, in particular, a slightly longer sampling time and an abundant control horizon in order to guarantee the convergence to the reference even in case of more complicated paths.

The simulation is designed with the time step set at 0.2 seconds. The control horizon is set to 15 seconds. This parameter defines the length of time over which the control input to the system is optimized. The total simulation time is set to 85 seconds, which is long enough to allow the system to travel a complete lap of the rectangle.

Table 5.10. Rectangular trajectory tracking setup.

Sampling time T	0.2 s
Control horizon	15 s
Total simulation time	85 s

The reference trajectory is a piecewise function given by the concatenation of the following equations, the fixed values vary according to the side of the rectangle in which the vehicle is transiting:

$$\begin{aligned} x^r(t) &= x_{i-1} + (x_i - x_{i-1}) \left(\frac{t - t_{i-1}}{t_i - t_{i-1}} \right) \\ y^r(t) &= y_{i-1} + (y_i - y_{i-1}) \left(\frac{t - t_{i-1}}{t_i - t_{i-1}} \right), \end{aligned} \tag{5.13}$$

where $i-1$ and i refer to the initial and final coordinates of the vertices of a side of the rectangle and vary according to the direction of travel.

By rearranging these equations the rectangular trajectory is set up in order to have a rectangle whose sides are of 6 m and 4 m respectively, and a constant velocity with an absolute value of 0.25 m/s

The initial configuration of the system is defined by the vector q_0, which specifies the initial position and orientation of the tractor-trailer. The vector q_0 is given by:

$$q_0 = \begin{pmatrix} 0\,\text{m} \\ 0\,\text{m} \\ \pi\,\text{rad} \\ 0\,\text{rad} \\ 0\,\text{rad} \end{pmatrix}. \tag{5.14}$$

This vector specifies that the tractor-trailer is initially located at the origin aligned with the corresponding side of the rectangle. So, the trailer is also initially in a straight configuration with no turning.

- Average time for each MPC iteration: 64.59 ms.
- Average time for each solver operation: 59.23 ms.
- Total time for output error feedback linearization: 634.84 ms.

- Settling time: `NaN`.
- Position RMSE: 0.07 m.

During the simulation, the MPC controller runs with an average time of 64.59 ms per iteration, and the solver operation takes an average time of 59.23 ms. The total time for output error feedback linearization is 634.84 ms.

However, it should be noted that the position error in this simulation does not stabilize. The position error increases at each corner of the rectangular trajectory, due to the tractor-trailer structure with very stringent steering and hitch angle constraints, which do not allow perfect tracking of the corners of the rectangle, and as a result, the settling time is undefined. The position root-mean-square error (RMSE) is 0.07 meters, which indicates that the tractor-trailer is still able to follow the desired trajectory with a high degree of accuracy.

In conclusion, the simulation analysis shows that the tractor-trailer system is able to track the rectangular trajectory accurately with a position RMSE of 0.07 m. The MPC controller is able to run with an average time of 64.59 ms per iteration, which suggests that the system's control inputs are optimized efficiently.

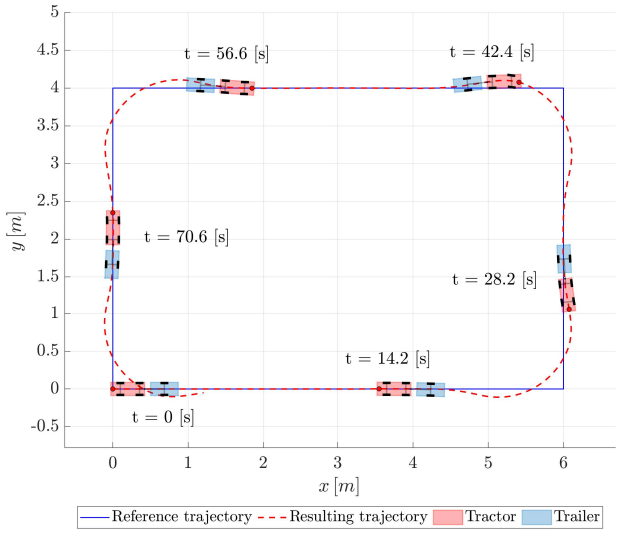

Figure 5.34. Rectangular trajectory in backward motion.

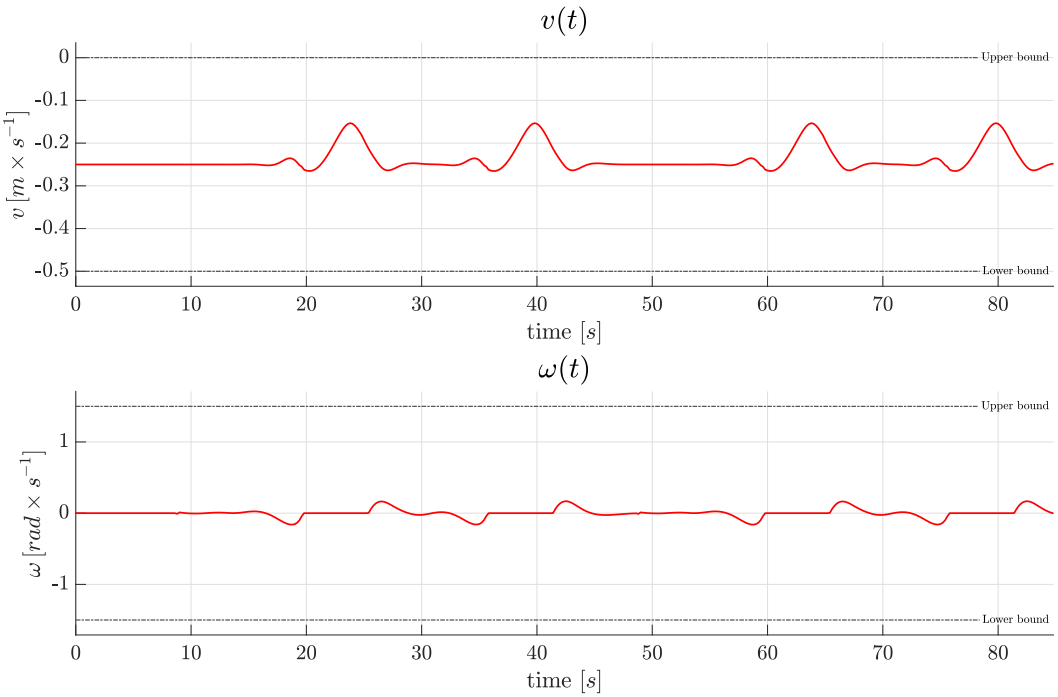

Figure 5.35. Control inputs performing rectangular trajectory in backward motion.

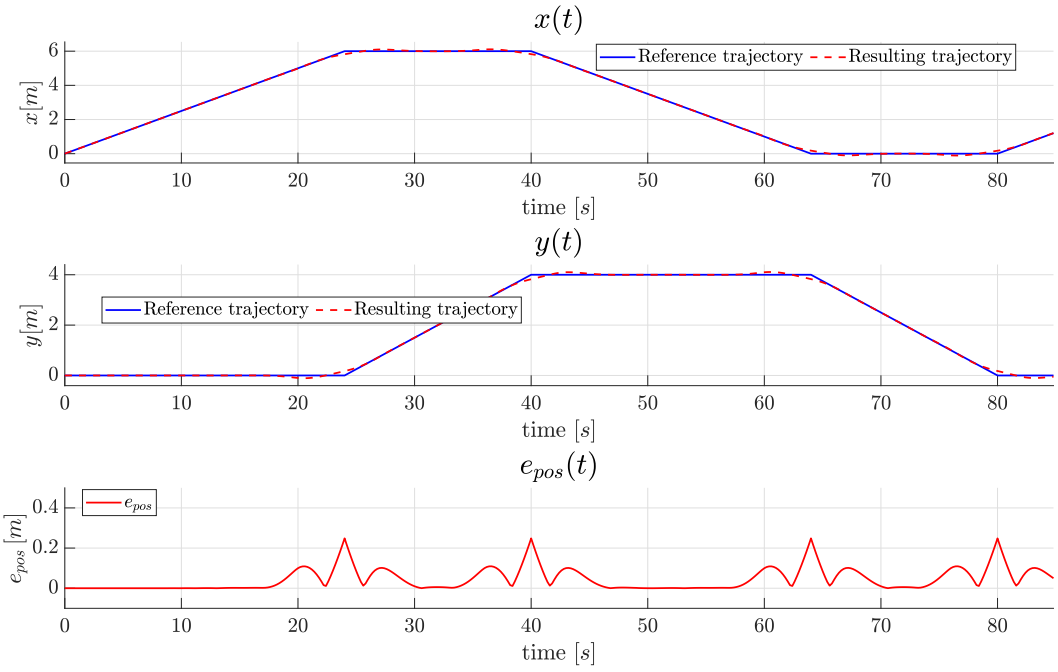

Figure 5.36. Cartesian errors performing rectangular trajectory in backward motion.

5.3 NMPC vs. IS-MPC

This work is proposed as a natural extension or a further development of the article [8], as just mentioned in 1.2.

Briefly, the article describes a control strategy for tractor-trailer vehicles that aims to prevent jackknifing with a proposed control strategy based on linear Model Predictive Control that is intrinsically stable (IS-MPC), i.e. using a strategy based on stabilizing constraints.

The authors validate the performance of the proposed controller through simulations and experiments on a prototype tractor-trailer vehicle. The results demonstrate that the intrinsically stable MPC controller effectively prevents jackknifing and improves the overall stability and maneuverability of the vehicle.

Since previous full trajectory simulations have shown that the proposed approach, based on nonlinear MPC (NMPC), can achieve the same performance as the linearized version, if not better in terms of convergence and execution times, the next step consists in evaluating the robustness of the two approaches in comparison, in order to understand if the use of the new approach leads to significant advantages.

Given that both methods guarantee excellent tracking on the trajectories analyzed above, in this case the objective is to evaluate the behaviour with the same setup, on a relatively simple trajectory (a linear trajectory), with more or less unfavorable initial configurations.

The setup of the simulation times is as follows: the sampling time is set to 0.1 seconds, the control horizon is set to 5 seconds, and the total simulation time is set to 20 seconds.

Therefore:

Table 5.11. Comparison trajectory tracking setup.

Sampling time T	0.1 s
Control horizon	5 s
Total simulation time	20 s

5.3.1 Position errors with slight misalignment

In this first simulation comparing the two MPCs, the initial configuration is with a position error of 0.5 meters, and a heading angle such that the velocity vector points slightly towards the reference trajectory.

Therefore, the vector q_0 is given by:

$$q_0 = \begin{pmatrix} 6\,\text{m} \\ 0.5\,\text{m} \\ \frac{\pi}{8}\,\text{rad} \\ 0\,\text{rad} \\ 0\,\text{rad} \end{pmatrix}. \tag{5.15}$$

IS-MPC

- Average time for each MPC iteration: 65.19 ms.
- Settling time: 6.30 s.
- Peak Cartesian error: 0.5 m.

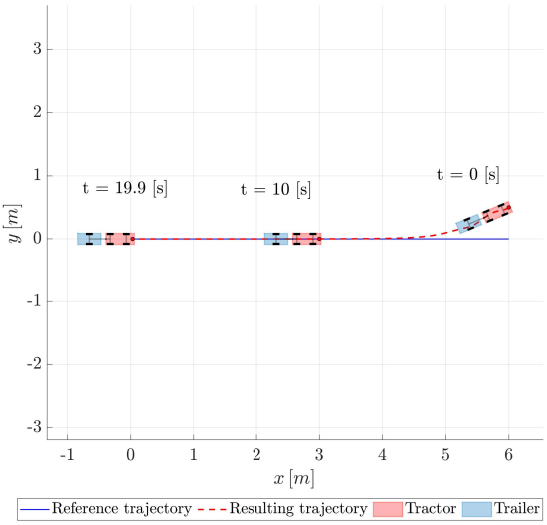

Figure 5.37. IS-MPC performing linear trajectory in backward motion.

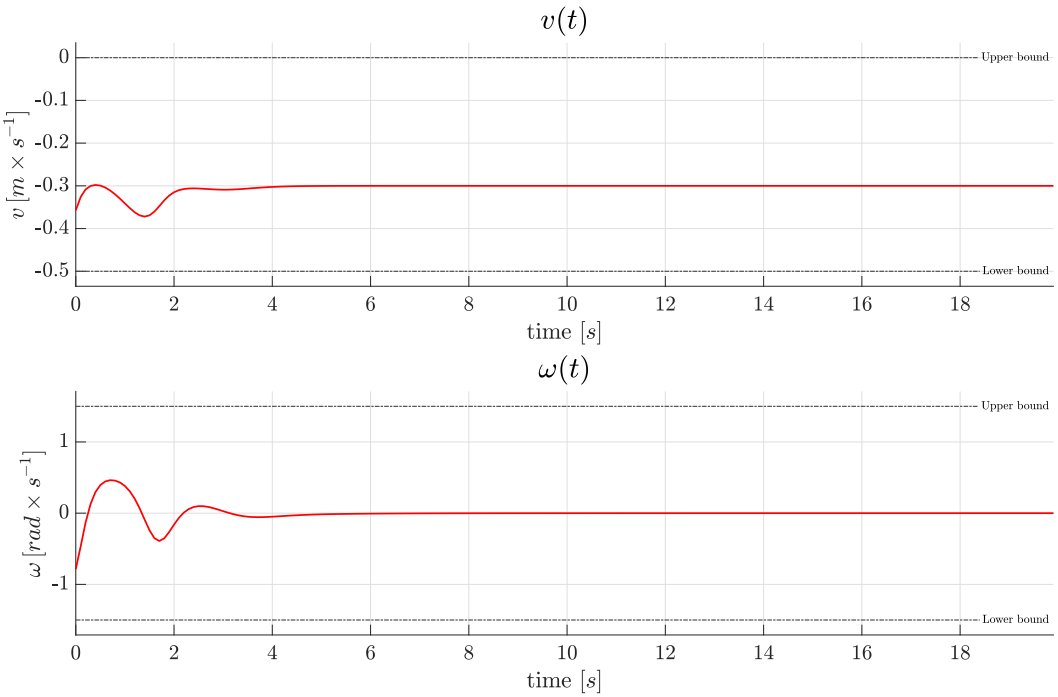

Figure 5.38. IS-MPC control inputs performing linear trajectory in backward motion.

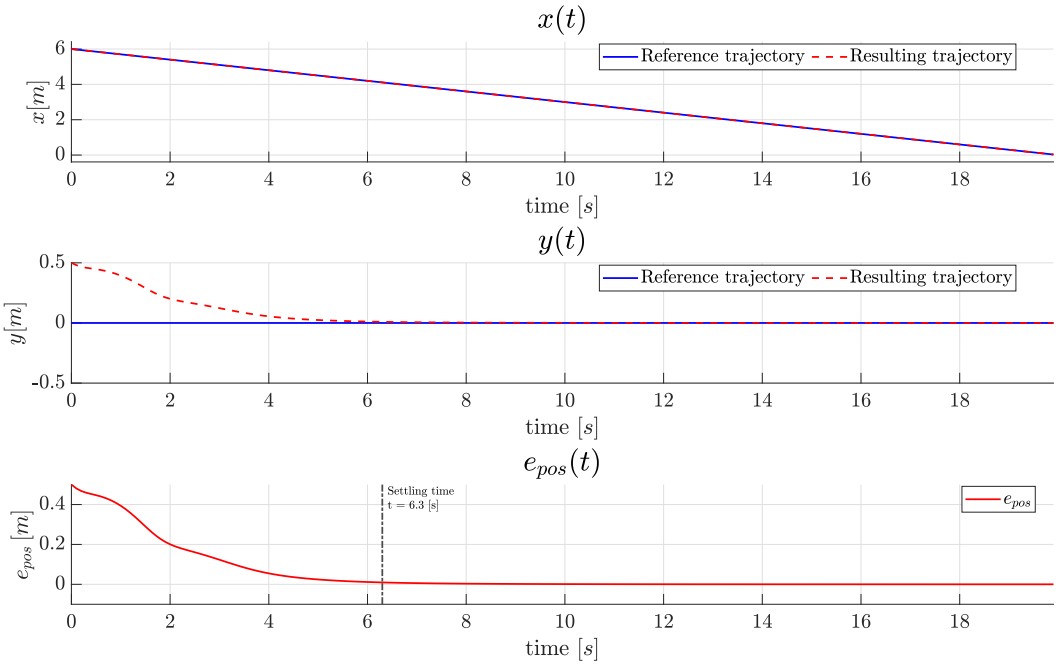

Figure 5.39. IS-MPC Cartesian errors performing linear trajectory in backward motion.

NMPC

- Average time for each MPC iteration: 32.33 ms.

- Settling time: 4.40 s.
- Peak Cartesian error: 0.5 m.

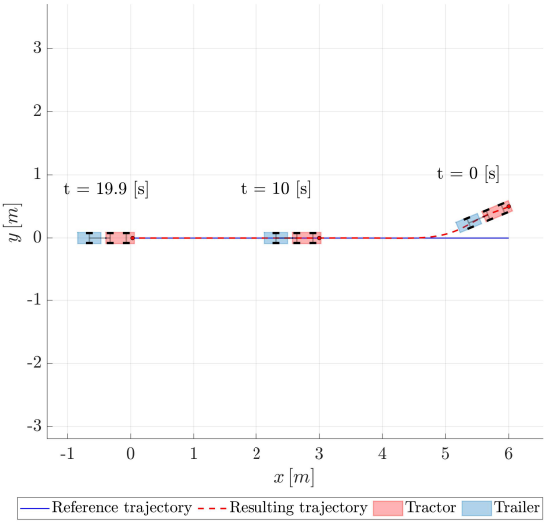

Figure 5.40. NMPC performing linear trajectory in backward motion.

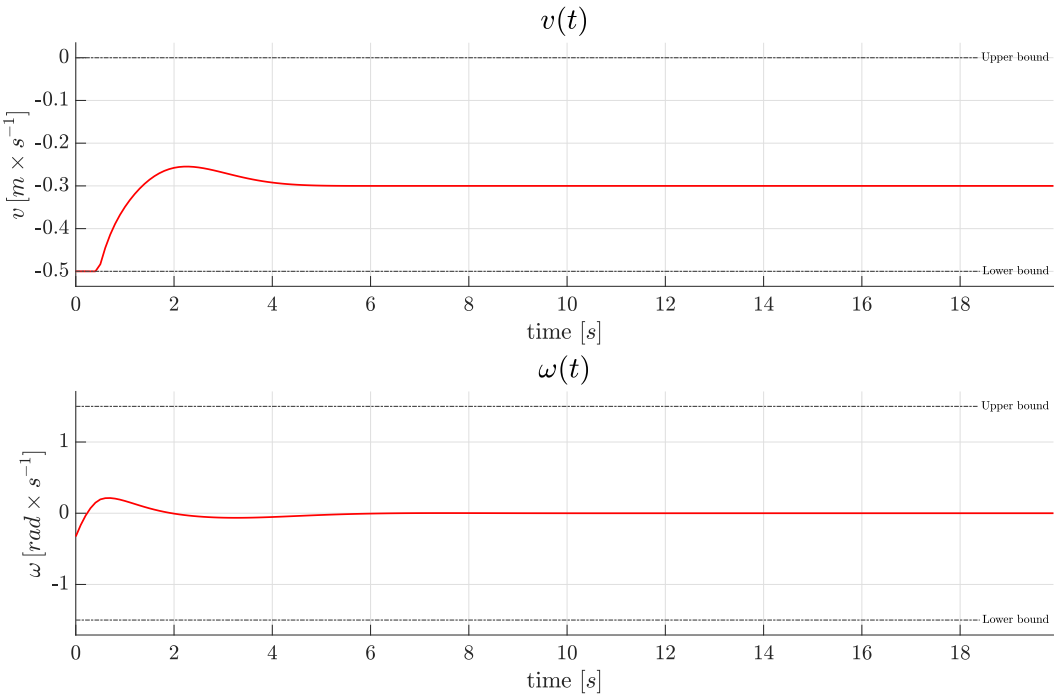

Figure 5.41. NMPC control inputs performing linear trajectory in backward motion.

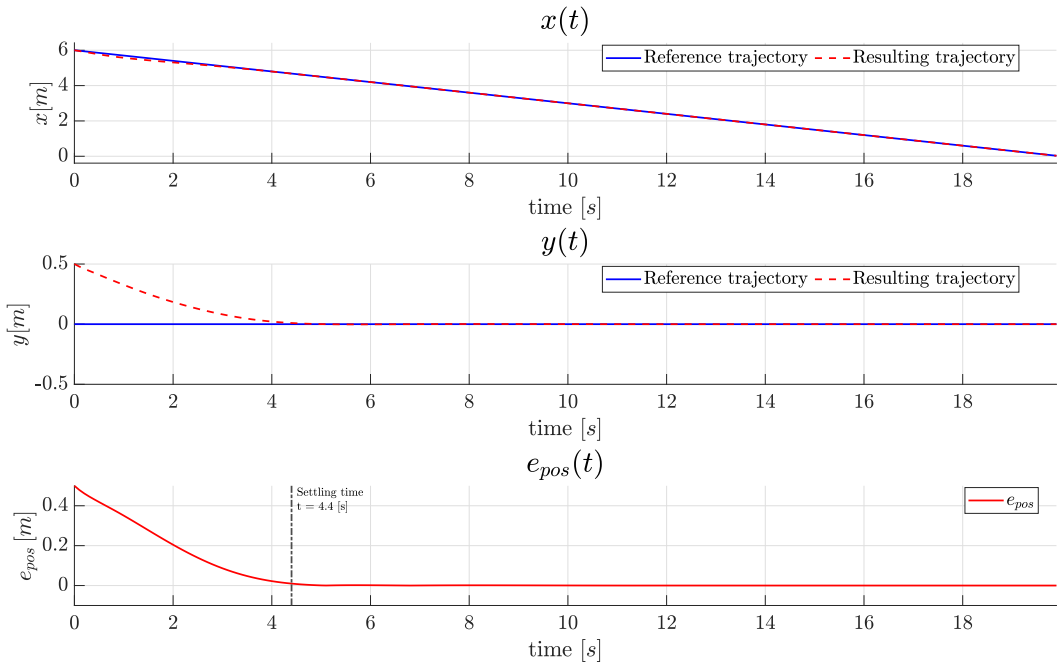

Figure 5.42. NMPC Cartesian errors performing linear trajectory in backward motion.

Analysis of the results

For this initial configuration, both MPCs perform well in terms of tracking the reference trajectory with a peak Cartesian error of 0.5 meters. The settling time for the NMPC is shorter, at 4.40 seconds, compared to the IS-MPC, which has a settling time of 6.30 seconds. The average time for each MPC iteration is also faster for the NMPC, at 32.33 ms, compared to the IS-MPC, which has an average time of 65.19 ms.

Overall, the NMPC performs slightly better than the IS-MPC in terms of settling time and average time for each iteration. However, both controllers perform well in terms of tracking the reference trajectory, with small peak Cartesian errors.

5.3.2 Position errors with large misalignment

In this second simulation comparing the two MPCs, the initial configuration is with a position error of 0.5 meters, and a heading angle such that the velocity vector does not point towards the reference trajectory.

So, the vector q_0 is given by:

$$q_0 = \begin{pmatrix} 6\,\text{m} \\ 0.5\,\text{m} \\ -\frac{\pi}{4}\,\text{rad} \\ 0\,\text{rad} \\ 0\,\text{rad} \end{pmatrix}. \tag{5.16}$$

IS-MPC

- Average time for each MPC iteration: 334.32 ms.
- Settling time: `NaN`.
- Peak Cartesian error: 0.67 m.

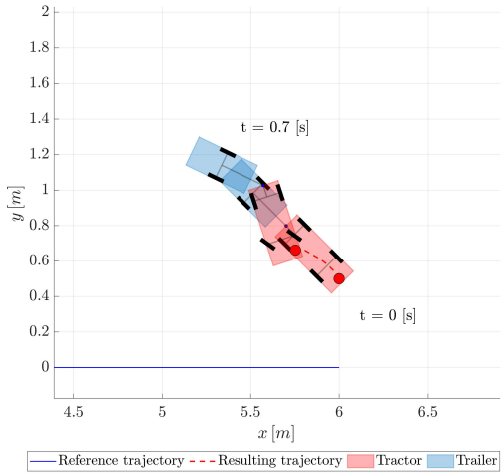

Figure 5.43. IS-MPC performing linear trajectory in backward motion.

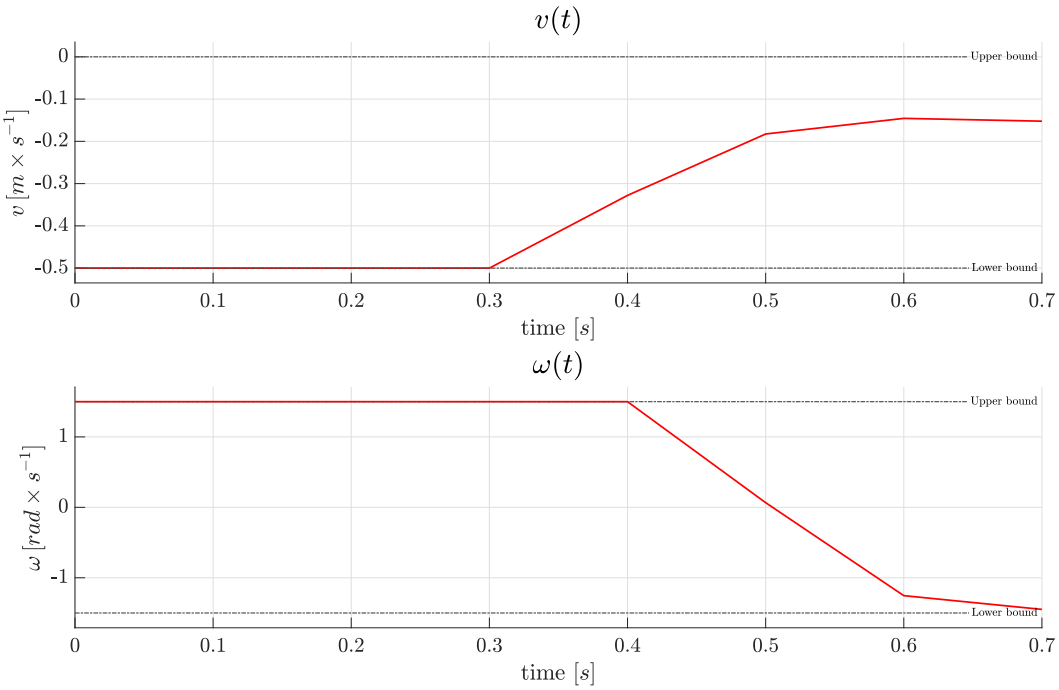

Figure 5.44. IS-MPC control inputs performing linear trajectory in backward motion.

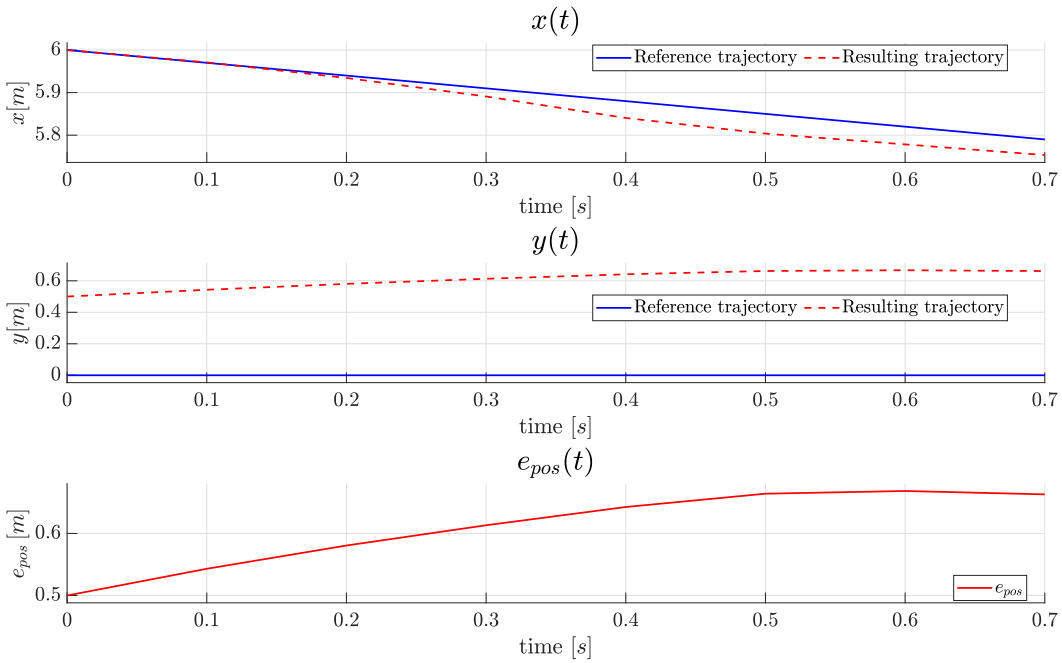

Figure 5.45. IS-MPC Cartesian errors performing linear trajectory in backward motion.

NMPC

- Average time for each MPC iteration: 44.12 ms.
- Settling time: 11.00 s.
- Peak Cartesian error: 1.42 m.

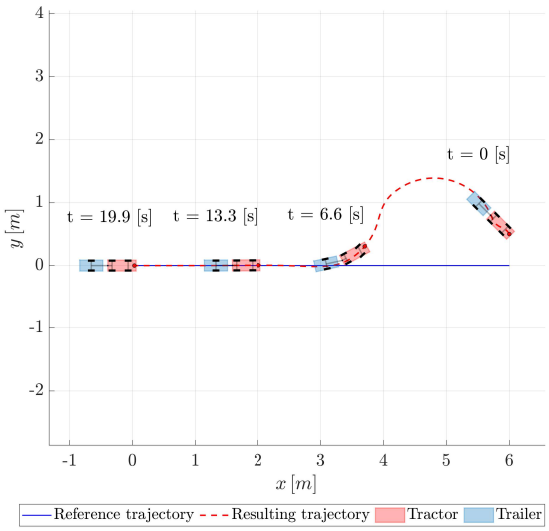

Figure 5.46. NMPC performing linear trajectory in backward motion.

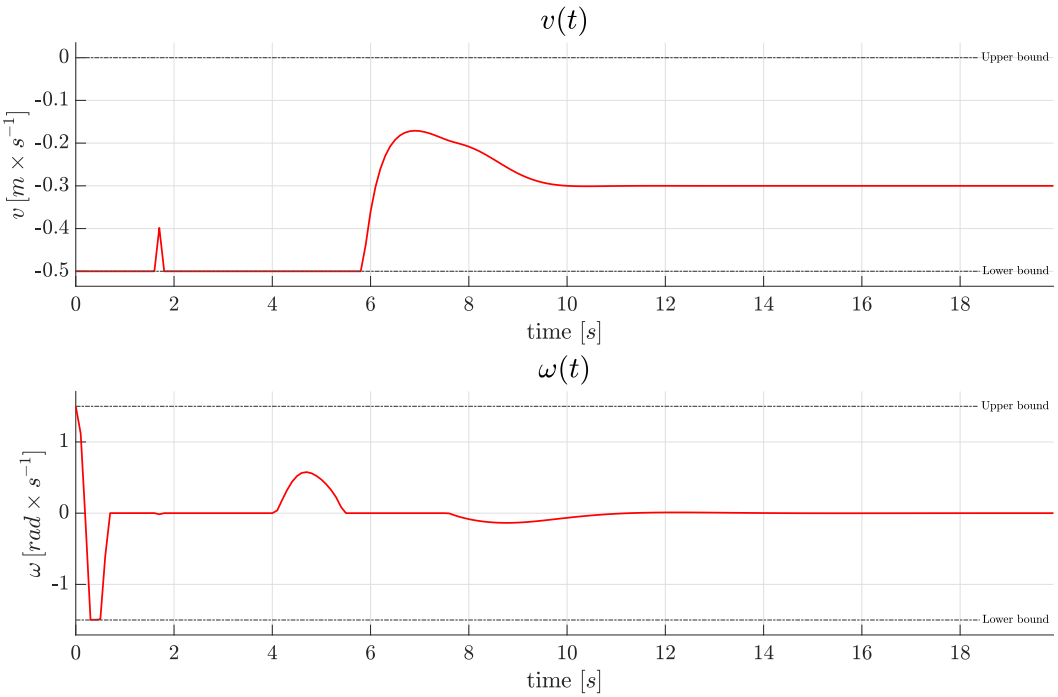

Figure 5.47. NMPC control inputs performing linear trajectory in backward motion.

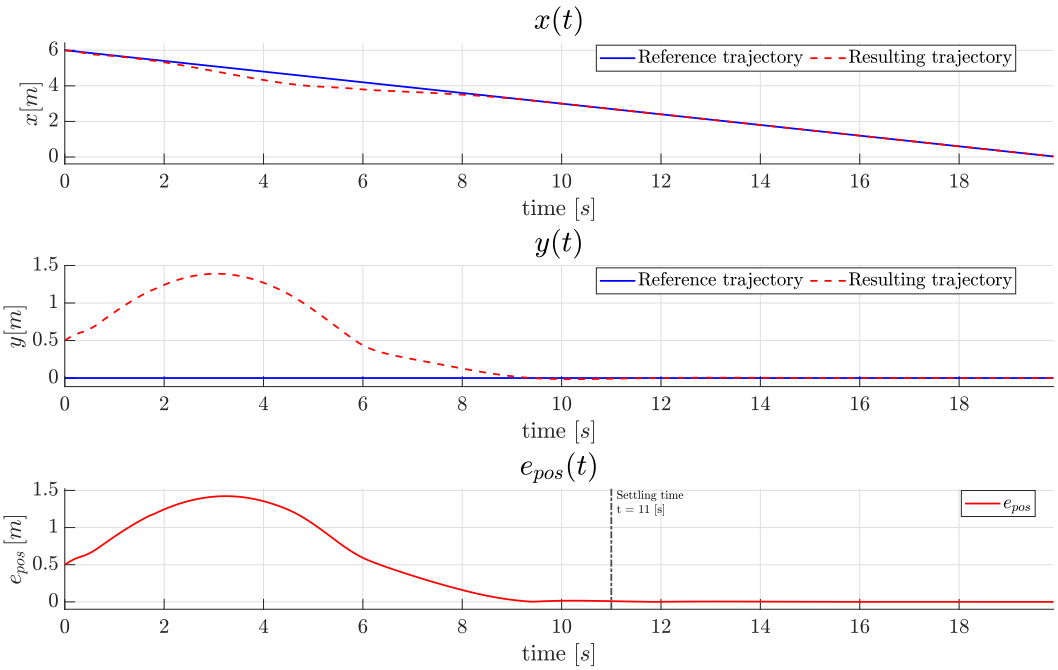

Figure 5.48. NMPC Cartesian errors performing linear trajectory in backward motion.

For completeness, a new simulation is carried out, increasing the misalignment in a much more evident way.

In the case of the IS-MPC, the previous simulation already highlighted the impossibility of tracking in more unfavorable conditions, with a larger misalignment,

therefore it is superfluous to further increase the misalignment.

However, it is possible to increase the misalignment in the case of the NMPC. Therefore a new q_0 is set:

$$q_0 = \begin{pmatrix} 6\,\mathrm{m} \\ 0.5\,\mathrm{m} \\ -\frac{\pi}{2}\,\mathrm{rad} \\ 0\,\mathrm{rad} \\ 0\,\mathrm{rad} \end{pmatrix}. \tag{5.17}$$

- Average time for each MPC iteration: 66.50 ms.
- Settling time: 14.20 s.
- Peak Cartesian error: 2.29 m.

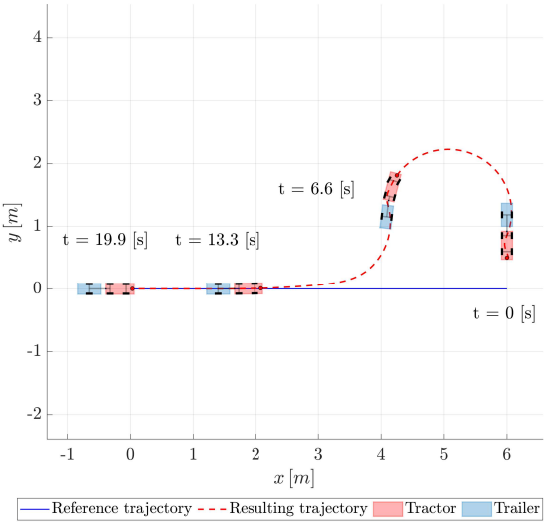

Figure 5.49. NMPC performing linear trajectory in backward motion.

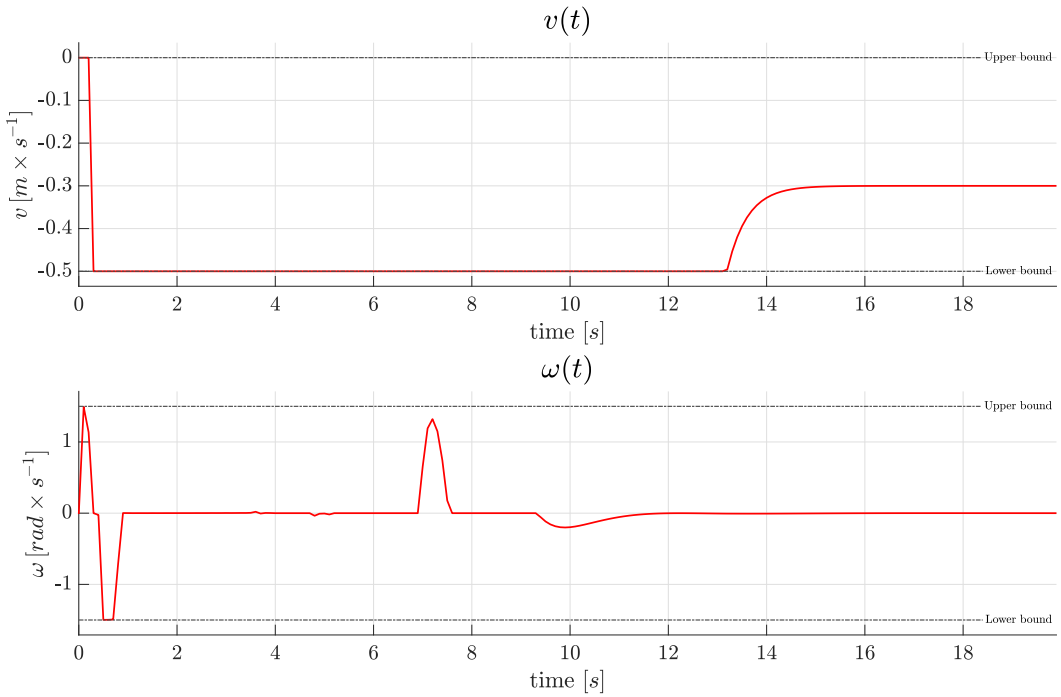

Figure 5.50. NMPC control inputs performing linear trajectory in backward motion.

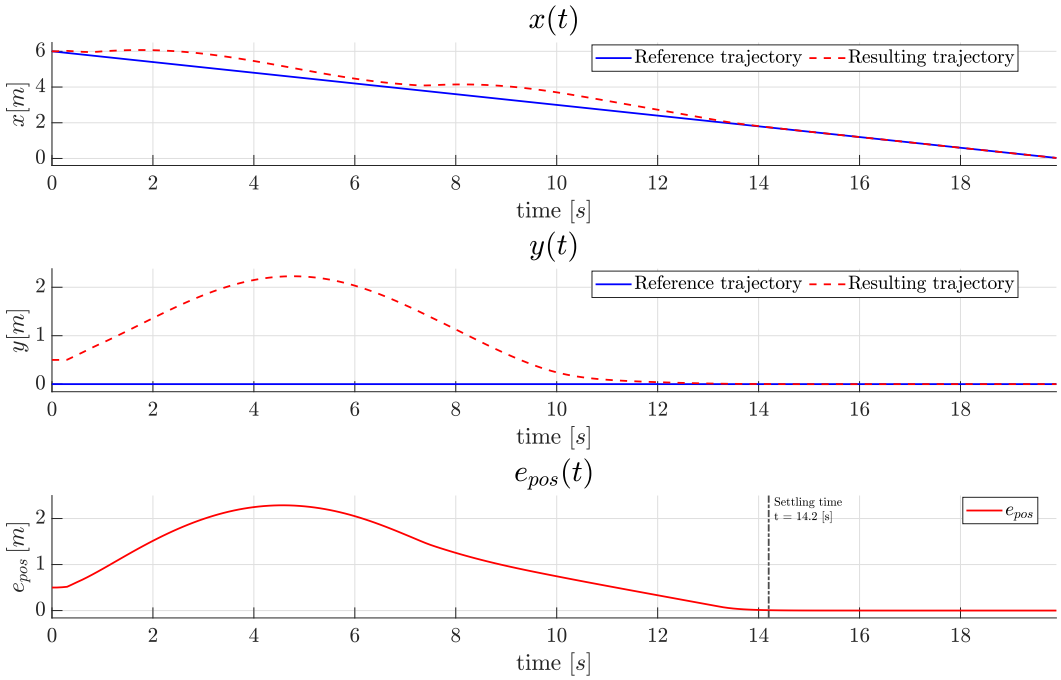

Figure 5.51. NMPC Cartesian errors performing linear trajectory in backward motion.

Analysis of the results

The analysis of the results immediately shows that the IS-MPC is less robust than the NMPC since the jackknife phenomenon occurs just a few instants after starting, This is the reason why the relative simulation is blocked after only 0.7 s. While instead the NMPC manages in any situation, even the most extreme, to complete the task successfully and in satisfactory execution times. In fact, NMPC has an average time of 44.12 ms for each MPC iteration, and the settling time is 11.00 s. The peak Cartesian error is 1.42 m.

With the larger misalignment, the NMPC algorithm can still track the reference trajectory with a peak Cartesian error of 2.29 m, an average time for each MPC iteration of 66.50 ms, and a settling time of 14.20 s. This result further confirms the robustness of the NMPC algorithm.

5.4 Behaviour of the double-trailer vehicle

In this part of the study, simulations are being conducted on complete trajectories traveled by a double-trailer vehicle. The goal of these simulations is to evaluate the effectiveness of the control logic and the robustness of the nonlinear Model Predictive Control in maintaining system stability over a prolonged time with this type of vehicle model. The objective is to determine if the nonlinear MPC can successfully cover an entire trajectory while keeping the system stable.

Similar to the previous simulations, the vehicle starts from an initial position that is already on the reference trajectory. This initial position is either perfectly aligned with the reference trajectory to ensure that the vehicle is pointing towards it and can quickly converge with it. This simulation aims to assess the ability of the nonlinear MPC to maintain the stability of the system throughout the entire length of the trajectory. However, this setup presents a greater challenge than the previous one, as there is an additional unstable variable to be stabilized due to the presence of the additional trailer.

The reference trajectory is a lemniscate. To ensure stability during the simulation, the advanced stabilizing terminal constraint is employed and the output error feedback is performed offline. In order to track the lemniscate trajectory, the setup of the simulation times is as follows: the sampling time is set to 0.1 seconds, the control horizon is set to 5 seconds, and the total simulation time is set to 220 seconds.

Therefore, the simulation setup is:

Table 5.12. Lemniscate trajectory tracking setup with a double-trailer vehicle.

Sampling time T	0.1 s
Control horizon	5 s
Total simulation time	220 s

following the reference trajectory given by the equations:

$$\begin{aligned} x^r(t) &= 5 \sin\left(\frac{1}{35}t\right) \\ y^r(t) &= 5 \sin\left(\frac{1}{35}t\right) \cos\left(\frac{1}{35}t\right), \end{aligned} \tag{5.18}$$

with starting configuration:

$$q_0 = \begin{pmatrix} 0\,\text{m} \\ 0\,\text{m} \\ \frac{5}{4}\pi\,\text{rad} \\ 0\,\text{rad} \\ 0\,\text{rad} \end{pmatrix}. \tag{5.19}$$

- Average time for each MPC iteration: 69.10 ms.
- Average time for each solver operation: 64.00 ms.
- Total time for output error feedback linearization: 3349.03 ms.
- Settling time: 10.00 s.
- Position RMSE: 0.00 m.

Based on the provided results, it can be inferred that the simulation using the nonlinear MPC is successful in maintaining system stability over the entire trajectory. The average time for each MPC iteration is found to be 87.88 ms, which is a reasonable time for real-time control applications. Similarly, the average time for each solver operation is 82.41 ms, indicating efficient performance of the solver.

The total time for output error feedback linearization is 2172.46 ms, which is a relatively short time for this type of computation, considering that is performed offline before the algorithm execution. The settling time of the system is 8.00 s, which suggests that the system reached a steady state quickly and effectively. Additionally, the position RMSE is found to be 0.00 m, which indicates that the MPC is successful in accurately tracking the reference trajectory.

Overall, these results suggest that the nonlinear MPC is effective in maintaining stability and accurately tracking a reference trajectory for a double-trailer vehicle over a prolonged time.

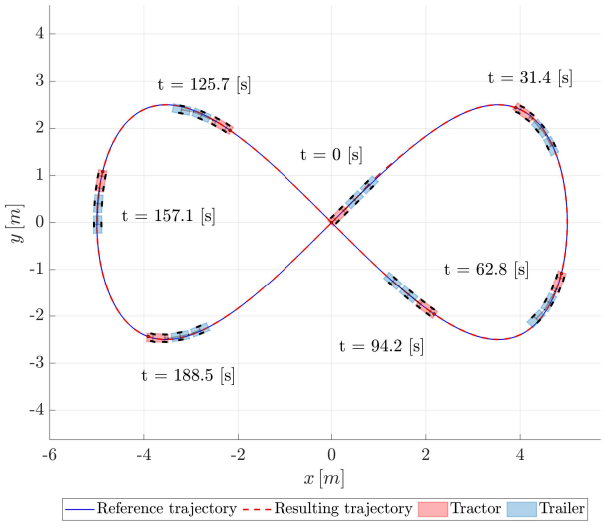

Figure 5.52. Lemniscate trajectory in backward motion with a double-trailer vehicle.

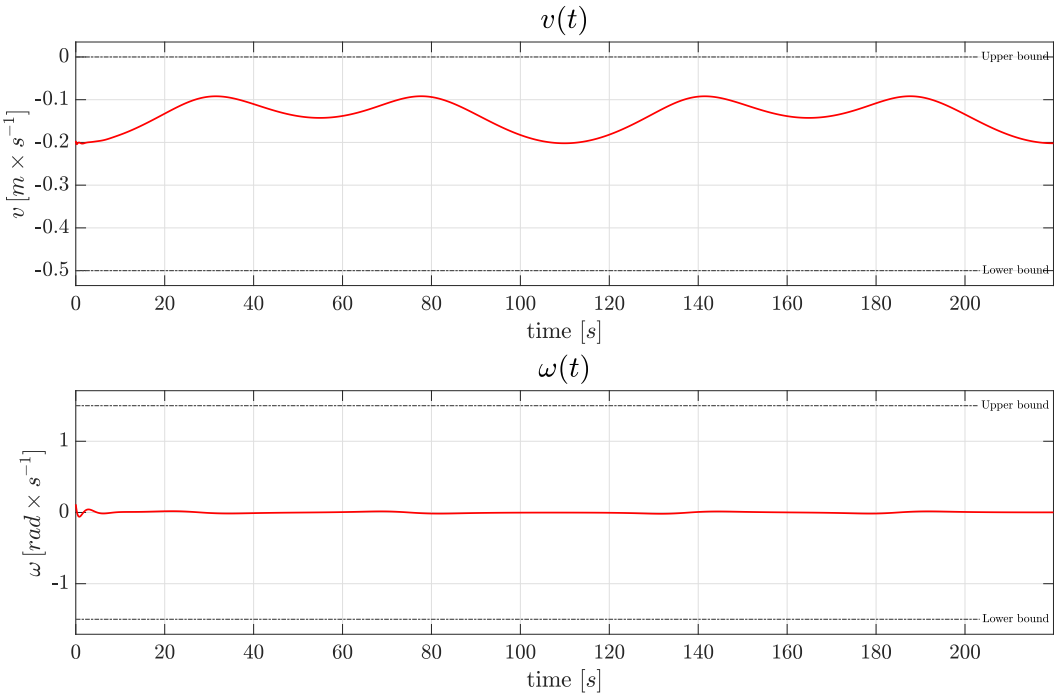

Figure 5.53. Control inputs performing lemniscate trajectory in backward motion with a double-trailer vehicle.

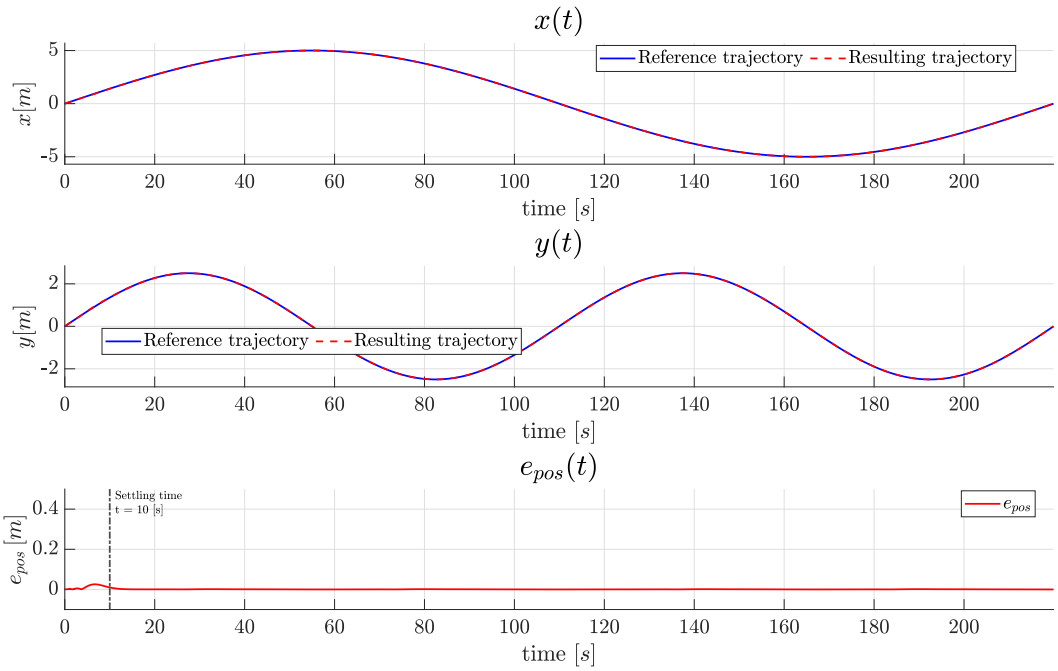

Figure 5.54. Cartesian errors performing lemniscate trajectory in backward motion with a double-trailer vehicle.

5.5 Insights from simulations

The first given results describe the performance of the nonlinear MPC algorithm with two types of stabilizing terminal constraints: basic and advanced. The simulations have been conducted for the tractor-trailer system's trajectory tracking, starting with significant errors in orientation and position with respect to the reference trajectory. The results have been compared for two different scenarios: misaligned starting poses and shorter control horizons.

For the misaligned starting poses scenario, both stabilizing terminal constraints perform well in terms of position RMSE and settling time. However, the advanced stabilizing terminal constraint leads to smoother control inputs and requires less computational time per iteration and per solver operation, making it the better choice for this scenario.

For the position and orientation starting errors combined scenario, the advanced stabilizing terminal constraint outperforms the basic stabilizing terminal constraint. It achieves a shorter settling time and requires less computational time per iteration and per solver operation, all with a shorter control horizon. Therefore, the advanced stabilizing terminal constraint is also the better choice for this scenario.

For the shorter control horizons scenario, both stabilizing terminal constraints perform well in terms of position RMSE (an exception is the case of the basic stabilizing terminal constraint on the circular trajectory, which never stabilizes), but the advanced stabilizing terminal constraint achieves a shorter settling time and requires less computational time per iteration and per solver operation. Therefore, the advanced stabilizing terminal constraint is again the better choice for this

scenario.

In summary, the advanced stabilizing terminal constraint is the better choice for trajectory tracking of the tractor-trailer system under different scenarios, as it leads to smoother control inputs, requires less computational time, and achieves a shorter settling time, therefore the advanced stabilizing terminal constraint is considered for all subsequent simulations.

Following, four simulations have been conducted to evaluate the performance of the tractor-trailer system with the advanced stabilizing terminal constraint. The system's ability to track a linear, circular, lemniscate, and rectangular reference trajectory has been assessed, and the results have been analyzed.

The simulation results show that the system can accurately track the reference trajectories, with small deviations from the reference trajectory. The position root-mean-square error (RMSE) for all trajectories was relatively small, indicating that the tractor-trailer system can closely follow the reference trajectory. The settling time for all trajectories was also reasonable, with the system converging to the reference trajectory within a few seconds.

The average time for each MPC iteration and solver operation is relatively low for all trajectories, suggesting that the control algorithm is efficient. However, it should be noted that the rectangular trajectory has larger position errors at the corners due to the stringent steering and hitch angle constraints.

Given the satisfactory results, a final comparison with [8] has been carried out.

The comparison between NMPC and IS-MPC shows that the former is more robust. The IS-MPC experiences jackknifing soon after the start of the simulation, which results in the simulation being blocked within a short time. On the other hand, the NMPC algorithm can complete the task successfully and efficiently, even in extreme situations.

The NMPC algorithm has an average MPC iteration time of less than the IS-MPC algorithm. Moreover, the NMPC algorithm has a longer settling time and a larger peak Cartesian error than the IS-MPC algorithm in normal conditions.

Furthermore, the NMPC algorithm can still track the reference trajectory with a larger misalignment, but the algorithm is still efficient, both under normal conditions and unfavorable conditions, proving its larger basin of attraction and confirming its robustness.

Further proof is given by the extension of the proposed algorithm to the case of the double-trailer. The simulation using the nonlinear MPC for the double-trailer vehicle is successful in maintaining stability throughout the trajectory. The algorithm shows efficient performance with a reasonable average time for each MPC iteration and solver operation. The system reaches a steady state quickly with a reduced settling time. The position RMSE indicates that the MPC accurately tracks the reference trajectory. Therefore, the nonlinear MPC is an effective approach for controlling also a double-trailer vehicle and maintaining stability while accurately tracking the reference trajectory.

In conclusion, the results of these simulations demonstrate that the tractor-trailer system with the advanced stabilizing terminal constraint can accurately track different types of reference trajectories with small or large deviations from the reference trajectory and a reasonable settling time. The control algorithm is efficient, and the system's performance is suitable for real-world applications.

Chapter 6

Output trajectory planning

In a real-world scenario, a mobile robot (in this instance, the tractor-trailer system) is presented with the daunting task of navigating through a complex and dynamic environment filled with numerous obstacles, both stationary and moving. This environment can range from indoor spaces such as factories and warehouses to outdoor environments such as construction sites and agricultural fields.

The process of enabling the robot to navigate the environment without colliding with any of the obstacles is known as motion planning. It is the task of designing a feasible path from the robot's starting point to its final destination while considering the presence of obstacles along the way.

Motion planning involves analyzing the robot's kinematics and dynamics, along with the properties of the environment, to determine a path that satisfies certain constraints such as safety, efficiency, and optimality. The motion planner takes into account the robot's physical limitations to ensure that the robot can follow the path generated by the planner.

To accomplish this objective, it is crucial to have prior knowledge of the obstacles in the environment. This information can be acquired through various means such as sensor data from the robot, maps of the environment, or manual annotations of the obstacles. Once the obstacle information is obtained, it is possible to solve the motion planning problem offline, before the robot starts to move. This enables the robot to navigate the environment with greater efficiency and safety, reducing the risk of collisions and damage to the robot and its surroundings.

6.1 RRT motion planning: safe robot navigation in complex environments

In the canonical motion planning problem, the robot operates in a workspace that can be represented as a mathematical space known as $\mathcal{W} = \mathbb{R}^N$, where $N = 2$. The robot is considered to be free-flying in its configuration space $\mathcal{C}$, which means it is not constrained by any kinematic limitations.

The obstacles in this workspace are viewed as immovable, rigid objects. Therefore, the challenge is to determine a path from the starting configuration, q_s, to the final configuration, q_g, while ensuring that the path remains within the free configuration space, $\mathcal{C}_{\text{free}}$, and avoids the set of all obstacles, $\mathcal{C}_{\text{obs}}$.

One commonly used approach for tackling this problem is the probabilistic method known as Rapidly-exploring Random Tree (RRT). This method falls under the category of sampling-based approaches, where random samples are taken from the configuration space $\mathcal{C}$. The RRT algorithm creates a tree-like structure that explores the configuration space, connecting sampled points to build a reference trajectory.

RRT is a widely-used algorithm that serves the dual purpose of constructing a graph (Figure 6.1) and determining a path. The constructed graph represents the configuration space of the problem, and the path is a sequence of connected nodes in the graph that corresponds to a motion from the initial to the goal configuration. It is worth noting that the path obtained by RRT may not be optimal in terms of distance or any other metric of interest.

RRT* is an improved version of RRT that aims to produce an optimal or near-optimal path. This algorithm is an evolution of the original RRT algorithm and utilizes the same principles but with a modification in the way the tree is constructed. The key difference between RRT* and RRT is that RRT* uses a cost function that takes into account the distance between the nodes in the tree and the new nodes being added to the tree. The algorithm connects the new node to its nearest neighbour in the tree, but only if doing so improves the overall cost of the path. This process iteratively builds a tree, favoring the nodes that lead to a shorter path.

Therefore, unlike RRT, which focuses on exploring the configuration space and constructing a feasible path, RRT* considers both feasibility and optimality. This algorithm is particularly useful in motion planning problems where the objective is to find the shortest path between two points in a given space. RRT* has been proven to produce optimal paths, meaning that as the number of iterations increases, the path generated by RRT* approaches the optimal path.

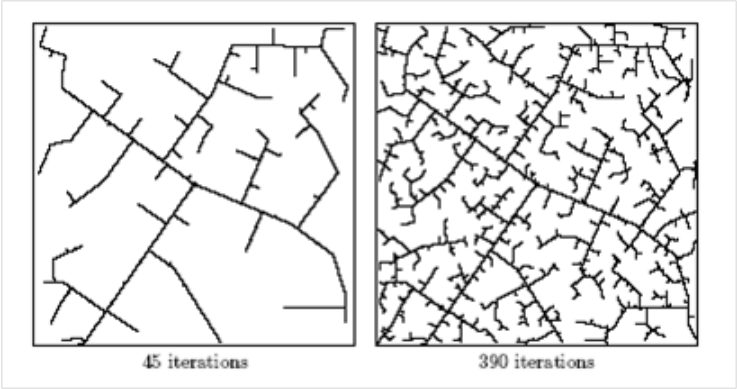

Figure 6.1. A visualization of an RRT graph after 45 and 390 iterations (Source: [14]).

6.1.1 RRT

The RRT algorithm is a way to find a path from a starting point to a goal point by randomly generating points and connecting them to the nearest existing point. When a new point is added, it must be checked to make sure it is not inside an obstacle, and the path to its closest neighbour must also avoid obstacles. The algorithm ends when a point is generated within the goal region, or when a limit is reached.

The way the points are randomly generated and connected can be customized. One way is to calculate the shortest distance between the new point and the closest edge and add a new point at the intersection. Another way is to add a chain of points between the new point and the closest existing point.

The graphs produced by RRT are not very efficient because they follow the two legs of a triangle, instead of navigating across the hypotenuse between two points, which is not the shortest path. To address this, RRT* was developed.

RRT is a fast algorithm compared to other path planning algorithms, but finding the closest neighbour can become computationally expensive as more points are added.

Algorithm 1 RRT algorithm

1: Root the tree T_s at q_s.
2: **while** goal state is not reached **do**
3: Generate q_{rand} in $\mathcal{C}$ with uniform probability distribution.
4: Search tree for the nearest configuration q_{near}.
5: Choose q_{new} at distance ϵ from q_{near} in the direction of q_{rand}.
6: Check for collision for the segment from q_{near} to q_{new}.
7: If no collision, add q_{new} to T_s.
8: Once the goal state is reached, construct the path from the start state to the goal state by following the parent nodes from the goal node back to the start node.

6.1.2 RRT*

RRT* is an improved version of RRT, a path planning algorithm used in robotics. RRT* finds the shortest possible path to the goal, as the number of nodes increases. It is achieved by adding two features to the RRT algorithm.

Algorithm 2 RRT* algorithm

1: Root the tree T_s at q_s.
2: **while** goal state is not reached **do**
3: Generate q_{rand} in $\mathcal{C}$ with uniform probability distribution.
4: Search tree for the nearest configuration q_{near}.
5: Choose q_{new} at distance ϵ from q_{near} in the direction of q_{rand}.
6: Check for collision for the segment from q_{near} to q_{new}.
7: If no collision, find all nodes q_{near_i} in the tree within a radius r of q_{new}.
8: Choose the parent q_{min} for q_{new} that results in the minimum cost-to-come along the path from q_s to q_{new}.
9: Add q_{new} to the tree and connect it to q_{min}.
10: Rewire the tree by checking if any existing nodes can be reached with a lower cost from q_{new}. If so, update their costs and parent nodes accordingly.
11: Once the goal state is reached, construct the path from the start state to the goal state by following the parent nodes from the goal node back to the start node.

The first feature records the distance each node has traveled relative to its parent node. This is called the "cost" of the node. After the closest node is found, a group of nodes within a fixed radius from the new node are examined. If a node with a lower cost than the closest node is found, it replaces the closest node. This addition creates a fan-shaped tree structure that is different from the cubic structure of RRT.

The second feature is called the rewiring of the tree. After connecting a node to its cheapest neighbour, the neighbours are re-examined. If rewiring a neighbour to the newly added node will make their cost decrease, it is rewired to the newly added node. This addition creates smoother paths around obstacles.

RRT* creates very straight paths, and its graphs are different from those of RRT. It is useful for finding optimal paths in a dense field of obstacles. However, RRT* takes longer to complete a single path on average than the default RRT, due to the considerable number of obstacle avoidance checks required.

In this work, the RRT* algorithm is used to generate a safe path in the Cartesian space where there are obstacles between the starting configuration and the target one. The obstacles (whose coordinates are known a priori) are randomly generated and a security expansion is applied to each of them to make the path even safer (Figure 6.2).

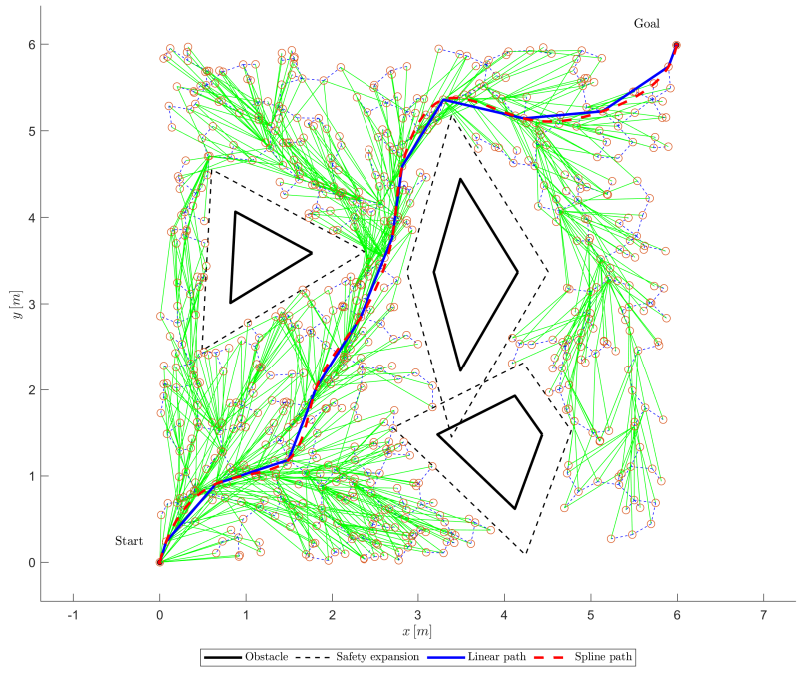

Figure 6.2. Safe linear and spline paths generated with RRT* in the Cartesian space.

The RRT* algorithm can be done in the Cartesian space, but the implementation and performance may vary depending on the specific problem being solved. The Cartesian space is a natural choice when the robot operates in a 2D or 3D environment,

where obstacles are represented as geometric shapes such as rectangles, circles, or polygons. In this case, the configuration space is reduced to the Cartesian space, and the algorithm can sample random points within this space.

One of the advantages of using the Cartesian space is that it simplifies the representation of the environment, making it easier to model and simulate. The geometric shapes can be easily described using equations or coordinates, and collision detection can be done efficiently by checking for overlap between the robot and the obstacles.

Therefore, when dealing with geometric fixed obstacles with a safety expansion, RRT* can be advantageous in the Cartesian space for several reasons. Firstly, the Cartesian space allows for efficient random sampling of points within the free configuration space, which is necessary for the RRT* algorithm. This is because the Cartesian space is easy to parameterize and generate random samples within its bounds. Secondly, collision detection can be done efficiently using geometric algorithms. This allows for faster computation of the cost function that is used in the RRT* algorithm to connect new nodes to the tree. Thirdly, RRT* in Cartesian space is capable of finding both feasible and optimal paths in the free configuration space while avoiding obstacles. This is because RRT* is designed to balance between feasibility and optimality, resulting in paths that are both collision-free and as close to optimal as possible. Fourthly, the Cartesian space provides a natural representation of the environment, making it easier to visualize the robot's motion and the generated paths. Finally, the Cartesian space is a common representation used in many applications, making RRT* in Cartesian space a versatile algorithm.

RRT* can make paths that aren't the shortest or most direct because it uses an ϵ value to add more nodes, which slows down the algorithm. To make the resulting paths faster and less tricky, a possible solution is given by the following method: the algorithm checks for collisions between each node against all the other nodes in the path. If there's no collision between a node and another non-adjacent node, it is possible to connect those nodes with a straight path and remove any nodes in between.

In other words, if there are no collisions between two extreme waypoints, the intermediate waypoints can be removed, and the two extreme points can be directly connected to form a more efficient path.

The sub-optimal path generated in this case is formed by a broken line, however, given the waypoints of the broken line, it is possible to use other types of interpolation to increase the feasibility of the route.

However, given the success obtained in the tracking simulation of a rectangular trajectory (Section 5.2.4), it is possible to assume that even in the case of a trajectory with a broken line, the nonlinear MPC is fully capable of carrying out the trajectory tracking to the tractor-trailer system.

The intermediate step from the path resulting from the RRT* algorithm and the trajectory tracking problem is precisely the conversion of the path into a feasible reference trajectory for the system in question.

To create a reference trajectory for a system to travel with a given velocity, a timing law must be added to a generated path. The desired velocity profile should first be determined in order to define the speed at which the system should travel along the path at each point in time (in the present case, a desirable constant velocity

with absolute value $v = 0.2\,\mathrm{m/s}$ is chosen).

Once the desired velocity profile has been established, the time it will take for the system to travel along the path at the desired velocity can be calculated using the distance between two points on the path and the velocity at that point ($time = \frac{distance}{velocity}$).

To generate a timing law, begin by assuming that the system starts at the beginning of the path at time $t = 0$. Calculate the time it will take for the system to travel to the next point on the path based on the desired velocity at that point, and add this time to the previous time to get the total time it will take to travel from the beginning of the path to the current point. Repeat this process for each point on the path to generate a timing law that maps each point on the path to a specific time at which the system should reach that point to follow the desired velocity profile.

With the timing law generated, the motion of the system along the path can be controlled by monitoring its position along the path and adjusting its velocity in real-time based on the timing law. This ensures that the system follows the desired velocity profile as it travels along the path.

Overall, adding a timing law to a generated path is an important step in creating a reference trajectory for a system to travel with a given velocity. By following the steps outlined above, a precise and accurate timing law can be generated to ensure that the system follows the desired velocity profile along the path.

At this point the main algorithm, starting from the RRT*, is able to generate the best reference trajectory according to the characteristics desired for the trajectory tracking problem in question. For the next simulations, the initial and final Cartesian positions are fixed (opposite diagonal positions of the map edges), while the obstacle positions are random (obstacles can overlap to form larger obstacles of different shapes) and the path type generated is a broken line for the first simulation and a smooth trajectory for the second one.

Two methods for interpolating data points to obtain a smooth curve are Pchip (Piecewise Cubic Hermite Interpolating Polynomial) and spline trajectories. Pchip is a specific type of spline method that uses cubic Hermite polynomials, which are functions that are defined by their values and slopes at the endpoints of each interval. The slopes are chosen to ensure monotonicity, meaning that the curve never turns back on itself between two data points. This makes Pchip a good choice when dealing with data that has a physical meaning and cannot change direction, such as position or velocity.

Spline trajectories, on the other hand, use piecewise polynomial functions to interpolate the data points. These functions are typically cubic or higher order, and they are chosen to minimize the curvature or other smoothness criteria. Spline trajectories can be more flexible than Pchip and can handle data with arbitrary shapes and features, but they may not guarantee monotonicity.

Therefore, due to these considerations, an approach using Pchip interpolation is preferable in this particular case.

Subsequently, the behaviour of the vehicle in the presence of obstacles is presented using the algorithm just described, the following plots show the relative results of the RRT* algorithm and the state and input variables. Both the simulation with a broken line trajectory and a Pchip spline has been performed. In both cases, as it is evident,

the tracking is perfect, a further demonstration that, given a feasible trajectory as a reference, the nonlinear MPC presented in this work allows to control the nonlinearities of the tractor-trailer model without problems, preventing jackknifing situations, and completing the required tasks successfully.

Broken line reference trajectory

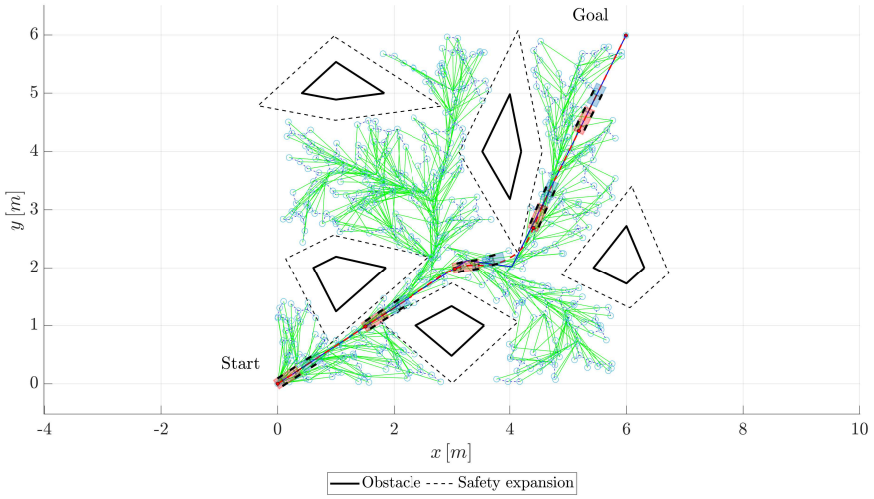

Figure 6.3. Broken line path resulting from RRT*.

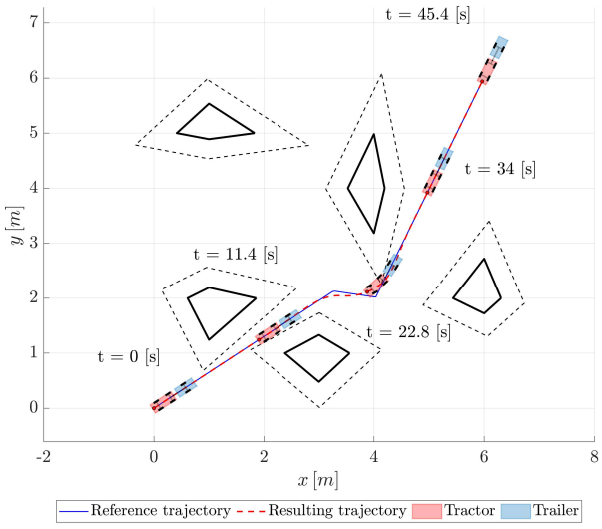

Figure 6.4. Broken line trajectory in backward motion.

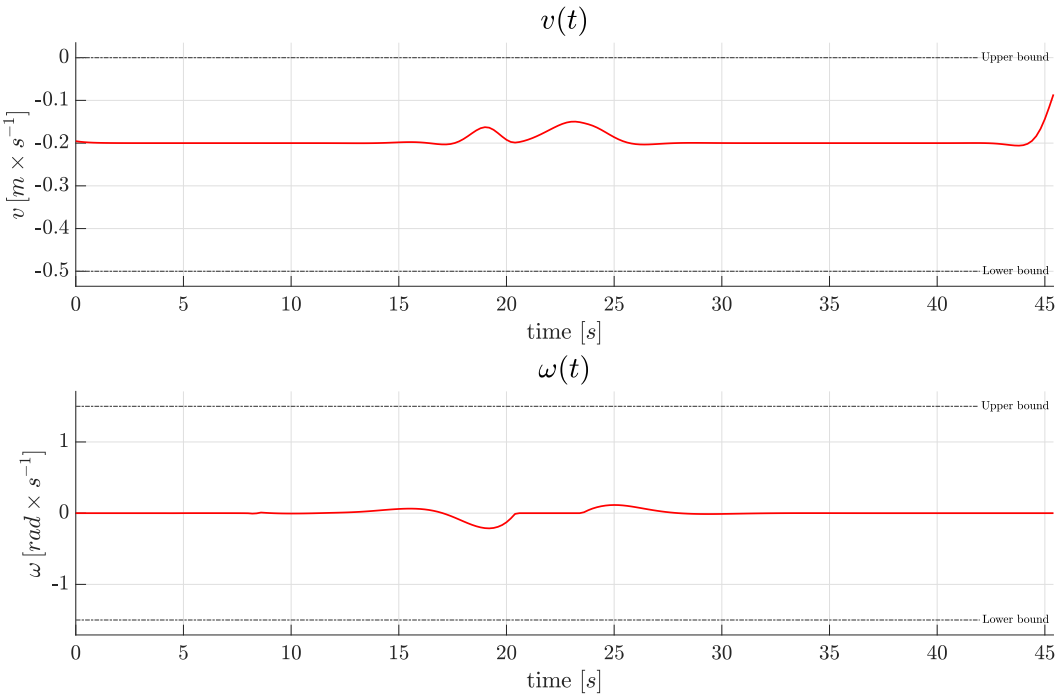

Figure 6.5. Control inputs performing broken line trajectory in backward motion.

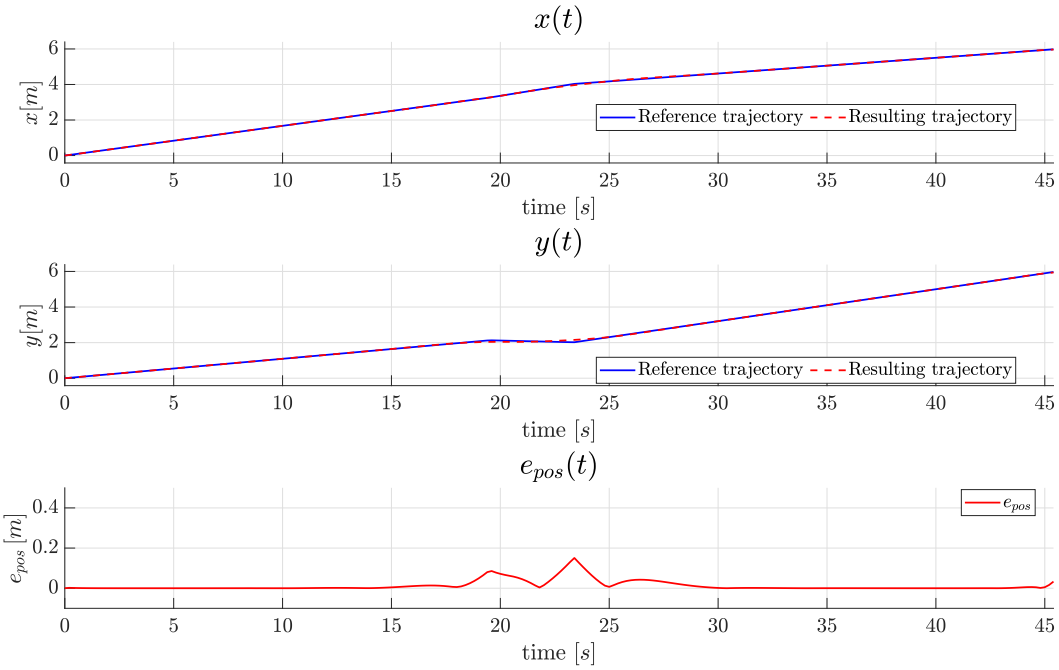

Figure 6.6. Cartesian errors performing broken line trajectory in backward motion.

Spline reference trajectory

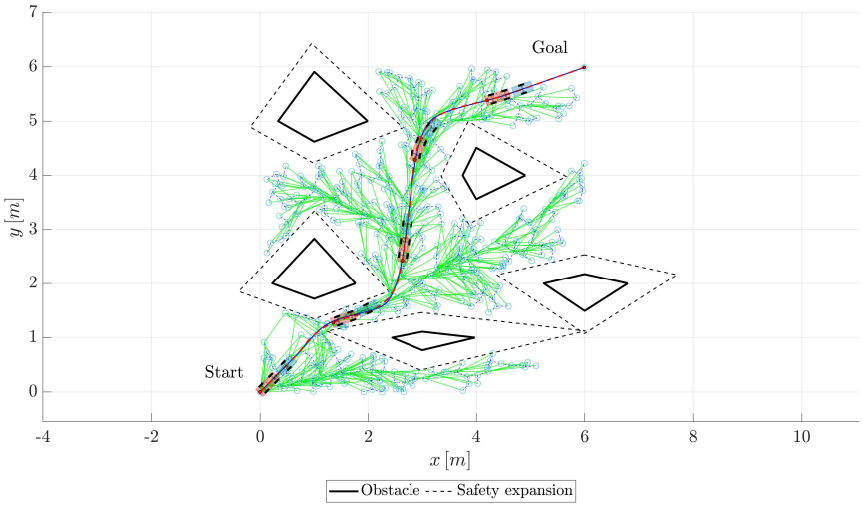

Figure 6.7. Spline path resulting from RRT*.

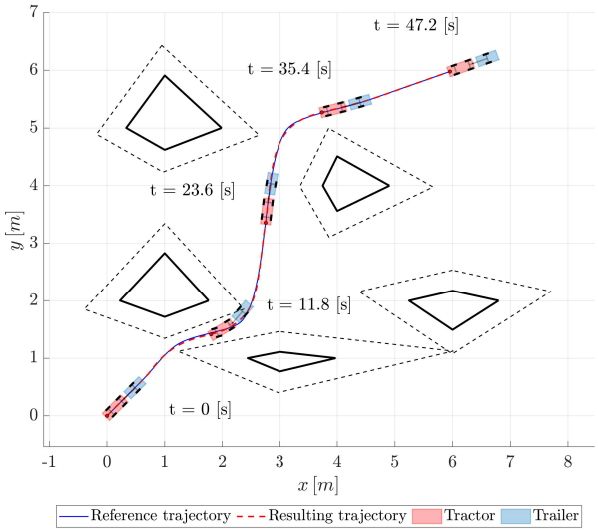

Figure 6.8. Spline trajectory in backward motion.

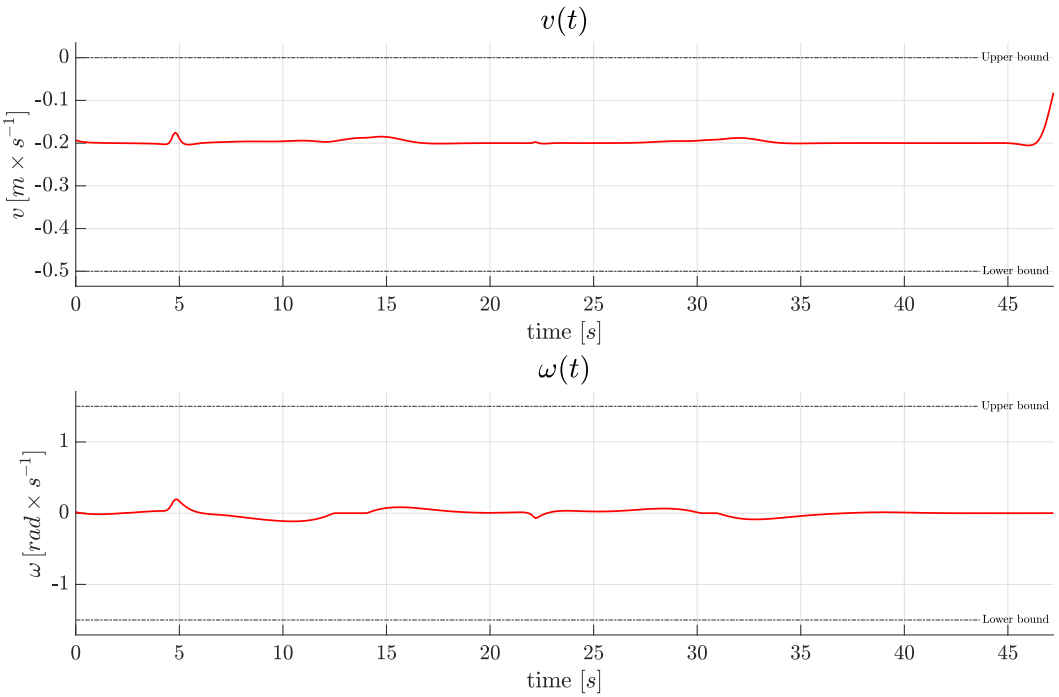

Figure 6.9. Control inputs performing spline trajectory in backward motion.

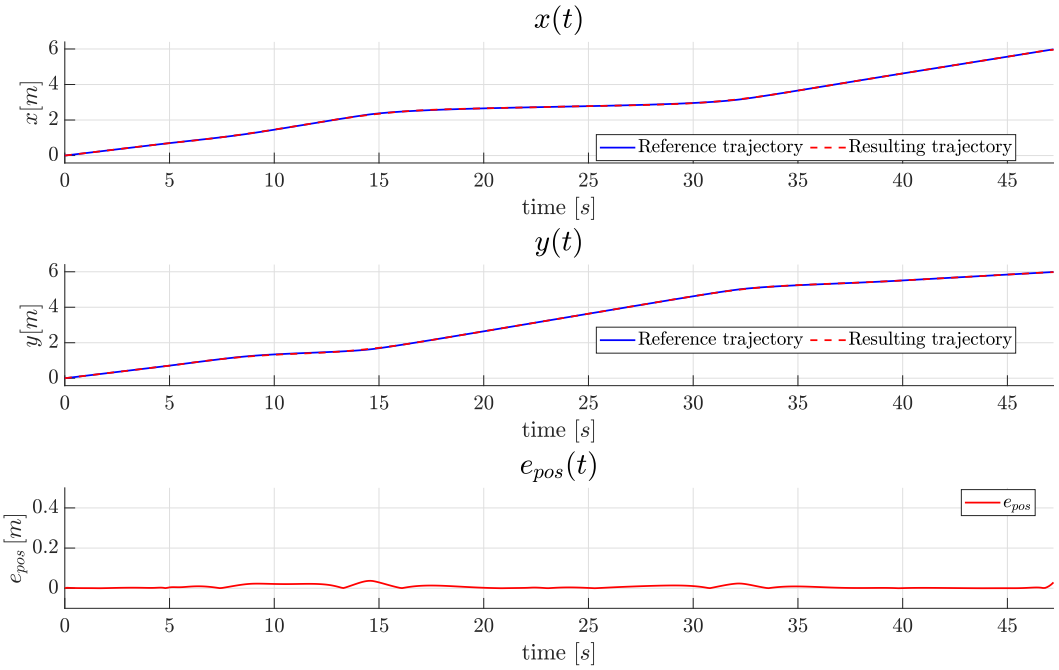

Figure 6.10. Cartesian errors performing spline trajectory in backward motion.

Chapter 7

Conclusions and future work

The thesis work presented a novel method for controlling a tractor-trailer system during reversing maneuvers. The approach has been developed and simulated in detail.

Chapter 2 provided a brief introduction to nonholonomic systems followed by the presentation of the kinematic model of the tractor-trailer system.

Chapter 3 offered a complete analysis and formulation of the problem to be addressed. This set the stage for the practical implementation in Chapter 4 and the simulation phase in Chapter 5 and Chapter 6.

The simulations conducted in this study have provided valuable insights into the performance of the tractor-trailer system controlled by nonlinear MPC with stabilizing terminal constraints. The advanced stabilizing terminal constraint was found to be the better choice for trajectory tracking, as it resulted in smoother control inputs, shorter settling times, and required less computational time.

The simulations also demonstrated that the tractor-trailer system could accurately track different types of reference trajectories, also with large deviations from the reference trajectory and a reasonable settling time. The control algorithm was efficient, and the system's performance was suitable for real-world applications.

Furthermore, the comparison with a previous study showed that the proposed nonlinear MPC algorithm with stabilizing terminal constraint was more robust, efficient, and had a larger basin of attraction, making it suitable for unfavourable situations. Additionally, the extension of the proposed algorithm to a double-trailer system demonstrated its effectiveness and ability to maintain stability while accurately tracking the reference trajectory.

The results indicate also that controlling a nonlinear system without considering its nonlinearities can lead to steady-state errors and make designing an accurate control law difficult. However, using a stability constraint has been shown to effectively handle the nonlinearities and instability issues, leading to almost perfect tracking of the reference trajectory.

In light of these findings, it can be concluded that the instability problem in reversing a tractor-trailer kinematic model has been effectively solved through the implementation of the nonlinear MPC controller that prevents jackknifing. This approach leverages real-time optimization and control techniques to precisely predict and adjust the trailer's trajectory, leading to improved stability and safety during

the reversing maneuver. The nonlinear nature of the MPC controller enables it to handle complex and dynamic system behaviors, further enhancing its effectiveness in addressing the instability issue.

In addition to its performance benefits, the use of a nonlinear MPC controller has the advantage of being adaptive to changing conditions and uncertainties in the system, making it a robust solution to the instability problem. By taking into account constraints and optimization objectives related to stability and safety, the controller can effectively prevent jackknifing and other forms of instability while still allowing for efficient and effective maneuvering. This is especially important in scenarios where the trailer is operating in challenging or unpredictable environments, such as tight spaces or uneven terrain.

Overall, the implementation of a nonlinear MPC controller for reversing a tractor-trailer kinematic model provides a valuable tool in addressing the persistent problem of instability and significantly improves the stability and safety of the system and it demonstrates the potential for continued advancements in the control design process.

In the future, it would be beneficial to validate the simulation results with the experimental prototype, from which all the physical parameters for the development of this work have been taken, as this would provide a more accurate assessment of the proposed algorithm's performance.

Additionally, an online obstacle avoidance strategy could be implemented to further enhance the system's capabilities in dynamic environments. Another possible direction for future work would be to explore the use of different primitives for the path generation in RRT* algorithm and interpolation methods for the trajectory's feasibility. One possible approach could be the use of Bezier curves with feasible curvature radii, which could provide a more efficient and feasible trajectory planning solution.

Further development of the RRT* algorithm could also be pursued, to improve its performance and expand its applicability in different scenarios.

These potential developments could enhance the effectiveness and versatility of the proposed algorithm, paving the way for its application in a wide range of robotic systems and applications.

Bibliography

[1] J.A.E. Andersson et al. *CasADi*. https://web.casadi.org/. 2023.

[2] C Altafini, A Speranzon, and B Wahlberg. "A feedback control scheme for reversing a truck and trailer vehicle". In: *IEEE Trans. Robot. Automat.* 17.6 (Dec. 2001), pp. 915–922.

[3] Joel A E Andersson et al. "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

[4] A. Astolfi, P. Bolzern, and A. Locatelli. "Path-tracking of a tractor-trailer vehicle along rectilinear and circular paths: A Lyapunov-based approach". In: *IEEE Trans. Robot. Automat.* 20.1 (Feb. 2004), pp. 154–160.

[5] Timothy Atkinson. *Robotic Path Planning: RRT and RRT**. https://theclassytim.medium.com/robotic-path-planning-rrt-and-rrt-212319121378. 2023.

[6] Manuel Beglini. "Backing up a tractor-trailer vehicle via intrinsically stable Model Predictive Control". Sapienza Università di Roma, 2019.

[7] Manuel Beglini, Leonardo Lanari, and Giuseppe Oriolo. "Anti-Jackknifing Control of Tractor-Trailer Vehicles via Intrinsically Stable MPC". In: *2020 IEEE International Conference on Robotics and Automation (ICRA) 31 May - 31 August, 2020. Paris, France*. IEEE, 2020.

[8] Manuel Beglini et al. "An Intrinsically Stable MPC Approach for Anti-Jackknifing Control of Tractor-Trailer Vehicles". In: *IEEE/ASME Transactions on Mechatronics* (2022), pp. 1–12. DOI: 10.1109/TMECH.2022.3154295.

[9] P Bolzern et al. "Pathtracking for articulated vehicles with off-axle hitching". In: *IEEE Trans. Control Syst. Technol.* 6.4 (July 1998), pp. 515–523.

[10] Rossella Bonuomo. "Motion planning and control of a tractor-trailer vehicle in backward maneuvers". Sapienza Università di Roma, 2022.

[11] Y. Chen et al. "Robust nonlinear model predictive control for the backward motion of tractor-trailer systems". In: *IEEE Transactions on Control Systems Technology* 28.5 (2020), pp. 1878–1888.

[12] Jimmy Chiu and Ambarish Goswami. "The critical hitch angle for jackknife avoidance during slow backing up of vehicle–trailer systems". In: *Vehicle System Dynamics* 52.7 (2014), pp. 992–1015. DOI: 10.1080/00423114.2014.909944.

[13] W. Chung et al. "Backward-motion control of a mobile robot with n passive off-hooked trailers". In: *J. Mech. Sci. Technol.* 25.11 (2011), pp. 2895–2905.

[14] Wikipedia contributors. *Rapidly-exploring random tree*. `https://en.wikipedia.org/wiki/Rapidly-exploring_random_tree`. 2023.

[15] Wikipedia contributors. *Runge-Kutta methods*. `https://en.wikipedia.org/wiki/Runge-Kutta_methods`. 2023.

[16] A Gonzalez-Cantos and A Ollero. "Backing-up maneuvers of autonomous tractor-trailer vehicles using the qualitative theory of nonlinear dynamical systems". In: *Int. J. Robot. Res.* 28.1 (2009), pp. 49–65.

[17] H Guo et al. "Simultaneous trajectory planning and tracking using an MPC method for cyber-physical systems: A case study of obstacle avoidance for an intelligent vehicle". In: *IEEE Trans. Ind. Informat.* 14.9 (Sept. 2018), pp. 4273–4283.

[18] Olov Holmer. "Motion Planning for a Reversing Full-Scale Truck and Trailer System". Linköping University, 2016.

[19] X. Ji, G. Lian, and Z. Ren. "Nonlinear model predictive control for anti-jackknifing control of tractor-trailer systems". In: *IEEE Transactions on Control Systems Technology* 28.3 (2020), pp. 1409–1420.

[20] E Kayacan, H Ramon, and W Saeys. "Robust trajectory tracking error model-based predictive control for unmanned ground vehicles". In: *IEEE/ASME Trans. Mechatronics* 21.2 (Apr. 2016), pp. 806–814.

[21] F. Lamiraux and J.P. Laumond. "A practical approach to feedback control for a mobile robot with trailer". In: *Proc. IEEE Int. Conf. Robot. Automat.* 1998, pp. 3291–3295.

[22] Z. Leng and M. A. Minor. "Curvature-based ground vehicle control of trailer path following considering sideslip and limited steering actuation". In: *IEEE Trans. Intell. Transp. Syst.* 18.2 (Feb. 2017), pp. 332–348.

[23] O Ljungqvist, D Axehill, and H Pettersson. "On sensing-aware model predictive path-following control for a reversing general 2-trailer with a car-like tractor". In: *Proc. IEEE Int. Conf. Robot. Automat.* 2020, pp. 8813–8819.

[24] O Ljungqvist et al. "A path planning and path-following control framework for a general 2-trailer with a car-like tractor". In: *J. Field Robot.* 36.8 (2019), pp. 1345–1377.

[25] Maciej Marcin Michałek. "Agile maneuvering with intelligent articulated vehicles: a control perspective". In: *IFAC-PapersOnLine* 52.8 (2019). 10th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2019, pp. 458–473. ISSN: 2405-8963. DOI: `https://doi.org/10.1016/j.ifacol.2019.08.092`.

[26] J. Morales et al. "Steering the last trailer as a virtual tractor for reversing vehicles with passive on-and off-axle hitches". In: *IEEE Trans. Ind. Electron.* 60.12 (Dec. 2013), pp. 5729–5736.

[27] Lars Nielsen and Uwe Kiencke. *Automotive Control Systems for Engine, Driveline, and Vehicle*. 2005.

[28] S. J. Park, J. H. Seo, and D. S. Lee. "Real-time control of tractor-semitrailer systems using model predictive control". In: *IEEE Transactions on Control Systems Technology* 17.5 (2009), pp. 1249–1257.

[29] C Pradalier and K Usher. "A simple and efficient control scheme to reverse a tractor-trailer system on a trajectory". In: *Proc. IEEE Int. Conf. Robot. Automat.* 2007, pp. 2208–2214.

[30] J. Qu, H. Li, and Y. Chen. "Nonlinear model predictive control for backward motion of tractor-trailer systems". In: *IEEE Transactions on Intelligent Transportation Systems* 20.2 (2019), p. 518.

[31] M. Sampei et al. "Arbitrary path tracking control of articulated vehicles using nonlinear control theory". In: *IEEE Trans. Control System Technol.* 3.1 (Mar. 1995), pp. 125–131.

[32] R. M. De Santis. "Path-tracking for articulated vehicles via exact and Jacobian linearization". In: *Proc. IFAC Intell. Auton. Veh.* 1998, pp. 159–164.

[33] Y. Song, C. Wang, and L. Hanzo. "Nonlinear Model Predictive Control of Tractor-Trailer System in Reverse Motion". In: *IEEE Transactions on Intelligent Transportation Systems* 20.11 (2019), pp. 3940–3949.

[34] Karl Worthmann et al. "Model Predictive Control of Nonholonomic Mobile Robots Without Stabilizing Constraints and Costs". In: *IEEE Transactions on Control Systems Technology* 24.4 (2016), pp. 1394–1406. DOI: 10.1109/TCST.2015.2488589.

[35] W. Wu et al. "Robust nonlinear model predictive control for backward motion of tractor-trailer systems with tire-road friction uncertainties". In: *IEEE Transactions on Control Systems Technology* 29.2 (2021), pp. 732–741.

[36] X. Xu, Y. Wang, and B. Chen. "Model predictive control for trailer's yaw angle and lateral position during backward motion". In: *IEEE Transactions on Intelligent Transportation Systems* 17.1 (2016), pp. 59–67.

[37] Y. Zhang, G. Lian, and Z. Ren. "Nonlinear model predictive control for anti-jackknifing control of tractor-trailer systems with wind disturbances". In: *IEEE Transactions on Control Systems Technology* 28.6 (2020), pp. 2746–2757.

[38] Y. Zhu, G. Lian, and Z. Ren. "Anti-jackknifing control strategy based on robust nonlinear model predictive control for tractor-trailer systems". In: *IEEE Transactions on Control Systems Technology* 29.4 (2021), pp. 1628–1638.

[39] Endrit Ziba. "Online Trajectory Re-Planning and Anti-Jackkning Control of a Two-Trailer System with a Car-like Tractor". Sapienza Università di Roma, 2021.