Universitat Autònoma de Barcelona

Modelling Trailer Parking

Alfredo Hernández David Masip Martí Municoy
Jan-Hendrik Niemann

20th December 2020

Abstract

The interesting aspect of car-trailer system is that the motion control while moving forward is stable, while reversing the motion is unstable. In this report the system is studied to analyse its stability and how to control it. Based on Classical Control Theory a first model is constructed using linear controller design (programmed in MATLAB). A second model is optimisation-based control (programmed in C). Last but not least, a model is constructed using steering-based control (programmed in Python). It is shown that two of the methods are viable options for control with their own advantages and disadvantages.

Contents

Abstract **2**

1 The Problem **6**
1.1 Introduction 6
1.2 Modelling the Kinematics 6
1.3 Controller Design 9

2 Mathematical Systems and Control Theory **10**
2.1 Controllability 11
2.2 Pole Placement Theorem 14
2.3 Controller Design 17

3 Implementation using MATLAB **20**
3.1 Programming the Controller 20
3.2 Controlling the Trailer on Different Curves 23
3.3 Issues with the Model 26

4 Alternative Controlling Methods **28**
4.1 Optimisation-based control 28
4.2 Steering-based control 29
4.3 Comparison of the two methods 32
4.4 Further improvements 32

5 Conclusions **34**

References **36**

1 The Problem

1.1 Introduction

Backing up a car can sometimes be a stressful experience. When you have something attached to your car, it gets even more nerve-racking. The problem is that the motion control while moving forward is stable, while reversing the motion is unstable. We want to analyse and solve this problem using mathematical and physical concepts.

To tackle this problem, we need to create a model of the system using differential equations and then design a controller for the system, that is, a mathematical model that can modify dynamically the behaviour of the system to follow a control or reference trajectory.

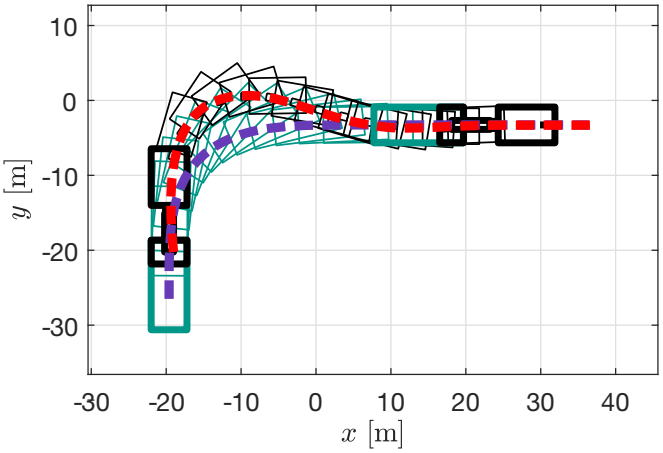


Figure 1: Traversed paths by the car (red line) and the trailer (purple line)

In short, using figure 1 as a reference, we would like to predict and control the trajectory purple dashed line (trajectory of the trailer) whilst only having direct access to the physical system that controls the red line (trajectory of the car).

1.2 Modelling the Kinematics

Before thinking on the model, we need to make some initial assumptions to simplify the problem. These are the initial assumptions we thought they will ease the modelling work:

- The system has a unique pivoting point, the hitch. This is completely necessary because it is the responsible of yielding this unstable equilibrium point when driving backwards.
- We assume a flat, nonslippery ground.
- The velocity of the car will be constant when going backwards. Given a steering angle, the car follows a circular path.

- As we want to work with a geometric model, we need the effect of the mass distribution to be negligible, so we assume a trailer with a uniformly distributed mass.
- Finally, we will work with a trailer that only has a single axle for the wheels.

Considering these assumptions, we can focus on the model. We chose a geometric model which only looks at the rate at which the ϕ angle changes along time. So our main goal is to predict the ϕ angle along the trajectory of the car.

Figure 2 shows the model of the vehicle-trailer system, and the parameters of the model are presented in table 1 below.

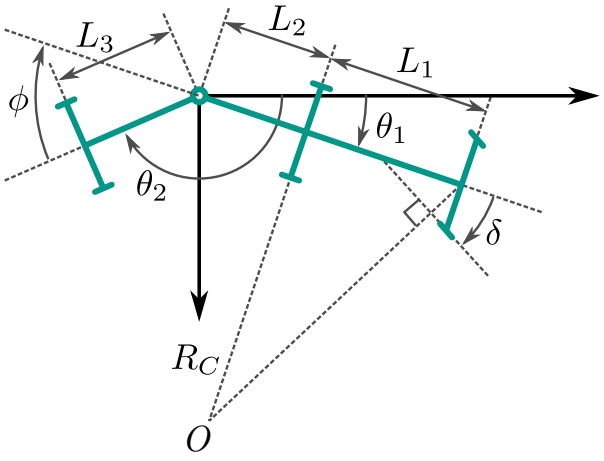


Figure 2: Geometric model of the car-trailer system

System parameters	Description
θ_1	The angle the car is travelling at with respect to a global coordinate system.
θ_2	The angle the trailer is travelling at with respect to a global coordinate system.
ϕ	The angle between the car and the trailer.
δ	The steering angle of the vehicle's wheels.
V	Reversing speed of the car.
$V_{trailer}$	Reversing speed of the trailer.
R_C	Distance from the centre of rotation O to the rear axle of the car.
L_1	Wheelbase of the car.
L_2	Overhang from the rear axle of the car to the hitch point.
L_3	Distance from the trailer axle to the hitch.

Table 1: Physical parameters of the system described by our model

If we know ϕ, we will be able to predict the trajectory of the trailer. The angle ϕ is defined as

$$\phi = \pi - (\theta_2 - \theta_1), \tag{1}$$

where θ_2 and θ_1 are the angles belonging to the car and to the trailer, expressed according to one reference axle.

Taking the derivative of the angle ϕ, we obtain

$$\dot{\phi} = -\dot{\theta}_2 + \dot{\theta}_1, \tag{2}$$

Using the distance to the centre of rotation O from the cars' rear axis, an expression for the angular velocity of the car is obtained:

$$\dot{\theta}_1 = -\frac{V}{R_C}. \tag{3}$$

This is, of course, because given a δ, the car follows a uniform circular path. This is what is usually known as Ackerman kinematics (see [1]), which state that, for a given δ, every point of the car follows a concentric circular path. The centre of these circles can be computed using trigonometry.

Using Ackerman kinematics, we can compute the angular velocity as the ratio of the longitudinal speed and the radius of the path. We can compute the radius of the path using trigonometry, and we obtain:

$$R_C = \frac{\tan\delta}{L_1}$$

Joining the two last expressions we can obtain the following expression:

$$\dot{\theta}_1 = -\frac{V}{L_1}\tan\delta. \tag{3 bis}$$

To predict ϕ, we also need to know how θ_2 varies along time. For this reason, we need to know which is the longitudinal speed and the normal speed at the point that joins the car and the trailer. Of course, the longitudinal speed is V, and, as the car is rotating at angular velocity $\dot{\theta}_1$, the normal speed of that point is $L_2 \cdot \dot{\theta}_1$. We can compute the longitudinal speed of the trailer by projecting in the longitudinal axis of the trailer the normal and longitudinal speed of the car in the point than joins them:

$$V_{trailer} = -L_2\dot{\theta}_1 \sin\phi + V\cos\phi. \tag{4}$$

We can also compute the normal speed by projecting the longitudinal and normal speeds in the normal direction of the trailer. This normal speed will be $L_3\dot{\theta}_2$. Using this, $\dot{\theta}_2$ can be isolated:

$$\dot{\theta}_2 = \frac{L_2\dot{\theta}_1\cos\phi + V\sin\phi}{L_3}. \tag{5}$$

Replacing (3 bis) and (5) into (2), we get the variation of ϕ along time. It is expressed in terms of the velocity of the car, the distance between the hitch and the trailer axle, the distance between the rear car axle and its hitch, the distance between both wheel axles for the car and the angle of the car's front wheels:

$$\dot{\phi} = -\frac{V}{L_3}\sin\phi - \frac{V}{L_1}\tan\delta\Big(1 + \frac{L_2}{L_3}\cos\phi\Big). \tag{6}$$

Of course, this system assumes that $V > 0$ when the car is going backwards. As all the geometric arguments are valid, if we take $V < 0$, we will obtain the dynamics of the trailer when the car is going forward.

Here we observe that if we set $\delta = 0$, then the differential equation becomes

$$\dot{\phi} = -\frac{V}{L_3}\sin\phi. \tag{6 bis}$$

In this case, $\phi = 0$ is a critical point of the differential equation. To analyse its stability, we have to discuss the sign of:

$$\frac{\mathrm{d}}{\mathrm{d}\phi}\Big(-\frac{V}{L_3}\sin\phi\Big) = -\frac{V}{L_3}\cos\phi \tag{7}$$

at the critical point. At $\phi = 0$, the previous expression becomes

$$-\frac{V}{L_3}. \tag{7 bis}$$

This shows that when $V > 0$ (car going backwards) this critical point is unstable, and when $V < 0$ the point is stable.

In the simulation, we will also take into account the phenomenon of jackknifing. A trailer jackknifes when its ϕ angle grows up to the point where the trailer contacts the car. Of course, by all means we want to avoid this situation. As a general fact accepted in the bibliography, (see [2]) when ϕ surpasses a critical ϕ_{crit}, the trailer cannot be controlled and the jackknifing phenomenon occurs.

We have already derived a model for the kinematics of the trailer with respect to the movement of the car. In order to produce a proper simulation of the whole system, we need to model the kinematics of the car as well. The model that we have used is based on Ackerman kinematics. The result of the model is that every point of the car moves through a circular path of centre O (see Figure 2).

1.3 Controller Design

We think that our problem is mainly a control theory based problem. The aim is to design a controller for the system to drive the state ϕ to the wished state by changing the steering angle δ. We would like to consider a linear controller which applies to the nonlinear model. The following section gives a mathematical introduction on control theory and afterwards the controller is presented.

2 Mathematical Systems and Control Theory

The following section deals with the topic of mathematical control theory. A short introduction is given, followed by a discussion of selected terms and definitions needed for this work. Among other things the term *controllability* and the *pole placement theorem* will be presented. We will only see the relevant proofs. All omitted proofs can be found in [3, 4, 5].[1]

Classical techniques for analysis and design of control systems have been used for more than five decades in a vast variety of industrial applications. These methods are based on transfer function models of the plant to be controlled. However, when it comes to more evolved tasks with e.g. more input and output variables other techniques are used. These techniques are based on state space models of the plant. One major advantage of state space models is for example that single-input single-output systems are equally treated as multi-input multi-output systems.

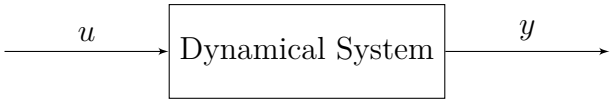


Figure 3: Block diagram of a dynamical system

For this work we consider a dynamical system which can be represented by a block diagram as seen in figure 3. There is an input $u(t)$ which represents controls, noises and disturbances and an output $y(t)$ representing the measurements. We assume that the behaviour of this "black box" can be approximated by the mathematical model

$$\begin{aligned} \dot{x}(t) &= f(t, x(t), u(t)) \\ \dot{y}(t) &= h(x(t), u(t)) \end{aligned} \tag{8}$$

were x is the vector of internal variables. When modelling a physical system, x is not necessary related to the "real" physical variables.

We consider that system (8) is a linear system of the form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ \dot{y}(t) &= Cx(t) + Du(t) \end{aligned}$$

which is called *linear time-invariant control system* (LTI). The matrices are called *system matrix* A, *input matrix* B, *output matrix* C and *feed-through matrix* D. The class of LTI systems is a very simple class. Nevertheless, lots of processes can be described using this class by a "sufficiently good" linearisation. Nowadays linear models are still used to model "real world" applications. It can be used to approximate the system in a neighbourhood of certain points. This model is obtained by linearisation which is feasible at points (x^*, u^*) such that $f(x^*, u^*) = 0$. These points are equilibrium points.

[1] If you are familiar with this topic you can skip this section and continue with § 2.3

2.1 Controllability

This section will introduce one important property of linear systems that determine whether or not given control objectives can be achieved. Simply said, a system is said to be controllable if it is possible to find a control input that takes the system from any initial state to any final state in any given time interval. We will see necessary and sufficient conditions for controllability.

Consider a system in state space realisation

$$\dot{x}(t) = Ax(t) + Bu(t), \qquad x(t_0) = x^0 \tag{9}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $(t_0, x^0) \in \mathbb{R} \times \mathbb{R}^n$, and $u(\cdot) \in U := \mathcal{L}^1_{loc}$. System (9) will shortly be represented as (A, B). The space of input functions is defined as

$$\begin{aligned}\mathcal{L}^1_{loc} := \{f : \mathbb{R} \to \mathbb{R}^m \mid f_i \quad &\text{measureable} \\ &\text{and} \int_K \mid f_i(t) \mid dt < \infty, \quad \forall K \subseteq \mathbb{R} \quad \text{compact}, \quad i = 1, ..., m\}.\end{aligned}$$

Alternatively, the space of input functions U can be defined as $U := \mathcal{PC}(\mathbb{R} \to \mathbb{R}^m)$ which are all piece-wise continuous functions. An important property of the input space is that U is closed under certain cencatenation, i.e. for any $u_1, u_2 \in U$ and $b \in \mathbb{R}$ the concatenation

$$u_1 \quad \&_b \quad u_2 := \left(t \mapsto \begin{cases} u_1(t), & t \le b \\ u_2(t), & t > b \end{cases} \right)$$

is in U.

The unique global solution of the initial value problem (9) is

$$t \mapsto x(t; t_0, x^0, u) = e^{A(t-t_0)}x^0 + \int_{t_0}^{t} e^{A(t-s)}Bu(s)ds.$$

The set of pairs (x, u) which solve (9) for some (t_0, x^0) is called the *behaviour* of (9):

$$\mathcal{B}_{(A,B)} := \{(x, u) \in X \times U \mid \exists (t_0, x^0) \in \mathbb{R} \times \mathbb{R}^n, \quad x(\cdot) = x(\cdot; t_0, x^0, u)\}$$

Here, X is a suitable space of functions such that

$$\{x(\cdot; t_0, x^0, u) \mid (t_0, x^0) \in \mathbb{R} \times \mathbb{R}^n, \quad u \in U\} \subseteq X.$$

First, we will define the terms *reachable*, *controllable* and *null-controllable*.

Definition 1. *A system (A, B) is called*

- Reachable *at time $T > 0$ if for all $x^1 \in \mathbb{R}^n$ there exists $(x, u) \in \mathcal{B}_{(A,B)}$ such that $x(0) = 0$ and $x(T) = x^1$,*

- Controllable *at time T if for all $x^0, x^1 \in \mathbb{R}^n$ there exists $(x,u) \in \mathcal{B}_{(A,B)}$ such that $x(0) = x^0$ and $x(T) = x^1$,*
- Null-controllable *at time T if for all $x^0 \in \mathbb{R}^n$ there exists $(x,u) \in \mathcal{B}_{(A,B)}$ such that $x(0) = x^0$ and $x(T) = 0$.*

Simply said: *Reachable* means that we can steer the system from 0 to any final state in any time Z. *Controllable* mean that it is possible to steer the system from any initial state to any final state in any given time T and hence *null-controllable* means that the final state is 0. One easily checks that $x^0 \in \mathbb{R}^n$ is null-controllable at T if and only if there exists a $u \in U$ such that $e^{AT}x^0 + \int_0^T e^{A(T-s)}Bu(s)ds = 0$. This is equivalent to: $-e^{AT}x^0$ is reachable at time T. Next, we define the *reachable set* and the *controllable set.*

Definition 2. *The* reachable set *from $x^0 \in \mathbb{R}^n$ at time $T > 0$ is defined as*

$$\mathcal{R}_{x^0}(T) := \{x^1 \in \mathbb{R}^n \mid \exists (x,u) \in \mathcal{B}_{(A,B)} : x(0) = x^0, x(T) = x^1\}.$$

The controllable set *to $x^1 \in \mathbb{R}^n$ at time T is defined as*

$$\mathcal{C}_{x^1}(T) := \{x^0 \in \mathbb{R}^n \mid \exists (x,u) \in \mathcal{B}_{(A,B)} : x(0) = x^0, x(T) = x^1\}.$$

Having this definition we can make the following proposition.

Proposition 1. *For (A,B) we have $\mathcal{R}_0(T) = e^{AT}\mathcal{C}_0(T)$ and, in particular, (A,B) is reachable at time T if and only if (A,B) is null controllable at time T.*

From Proposition 1 we can deduce the following.

Proposition 2. *The system (A,B) is reachable at $T > 0$ if and only if (A,B) is null-controllable at $T > 0$ if and only if (A,B) is controllable at $T > 0$.*

Proof. By Proposition 1 it only remains to prove that if (A,B) is reachable at T then (A,B) is controllable at T.
Let $x^0, x^1 \in \mathbb{R}^n$ be reachable at time T. Then there exists $(x_1,u_1) \in \mathcal{B}_{(A,B)}$ such that $x_1(0) = 0$ and $x_1(T) = x^0$ and there exists $(x_2,u_2) \in \mathcal{B}_{(A,B)}$ such that $x_2(0) = 0$ and $x_2(T) = x^1 - x_1(2T)$. Since $(x_1(\cdot + T), u_1(\cdot + T)) \in \mathcal{B}_{(A,B)}$, which is known as *shift-invariance* of the behaviour, we find

$$(x(\cdot), u(\cdot)) := (x_1(\cdot + T) + x_2(\cdot), u_1(\cdot + T) + u_2(\cdot)) \in \mathcal{B}_{(A,B)}$$

with

$$\begin{aligned} x(0) &= x_1(T) + x_2(0) = x^0 \\ x(T) &= x_1(2T) + x^1 - x_1(2T) = x^1. \end{aligned}$$

□

In general it is hard to check if for any $x^0, x^1 \in \mathbb{R}^n$ there exists an input u that steers x^0 to x^1. To check this easily the following helps.

Theorem 1. *For all $T > 0$ it holds*

$$\mathcal{R}_0(T) = \text{im}[B, AB, ..., A^{n-1}B] =: \text{im}\, K(A, B) \in \mathbb{R}^{n \times nm}.$$

The matrix K is known as the *Kalman*-matrix. To proof this theorem we need the following lemma.

Lemma 1. *For any $\eta \in \mathbb{R}^n$ and system (A, B) the following conditions are equivalent:*

1. *η is orthogonal to the reachable set $\mathcal{R}_0(T)$*
2. *For all $t \in [0, T]$ it holds $\eta^T e^{At} B = 0$*
3. *For all $k \in \mathbb{N}_0$ it holds $\eta^T A^k B = 0$*
4. *$\eta^T [B, AB, ..., A^{n-1}B] = 0$.*

Proof. First we show that 1 implies 2.
Let $x^1 \in \mathcal{R}_0(T)$ Then $x^1 = \int_0^T e^{A(T-s)} B u(s) ds$ for some input $u \in U$. Since $\eta^T x^1 = 0$ it follows that for all inputs $u \in U$ we have

$$\eta^T \int_0^T e^{A(T-s)} B u(s) ds = 0.$$

For $u(\cdot) = B^T e^{A^T(T-s)} \eta \in U$ we find

$$\int_0^T \eta^T e^{A(T-s)} B B^T e^{A^T(T-s)} \eta ds = 0$$

which is equivalent to

$$\int_0^T || B^T e^{A^T(T-s)} \eta ||^2 ds \geq 0.$$

Therefore, for all $s \in [o, T]$ we have $B^T e^{A^T(T-s)} \eta = 0$.
Now we will show that 2 implies 1.
Let $x^1 \in \mathcal{R}_0(T)$. Then there exists $u \in U$ so that

$$\eta^T x^1 = \eta^T \int_0^T e^{A(t-s)} B u(s) ds = 0$$

which implies that η is orthogonal to $\mathcal{R}_0(T)$.
Next we will show that 2 implies 3.
For $t = 0$ we have $\eta^T B = 0$. Therefore, for all $t \in [0, T]$ we have

$$\begin{aligned} 0 &= (\frac{d}{dt})^k \eta^T e^{At} B \\ &= \eta^T A^k e^{At} B \quad \xrightarrow{t=0} \quad \eta^T A^k B = 0 \end{aligned}$$

which shows 3.
Looking at $e^{At} = \sum_{k=0}^{\infty} \frac{t^k A^k}{k!}$ we see that 3 also implies 2.
That 3 implies 4 is obvious and the opposite direction can be shown by using the *Cayley-Hamilton* theorem. □

Now we can easily proof Theorem 1.

Proof. By Lemma 1 we have $\eta \in \mathcal{R}_0(T)^\perp$ which is equivalent to η is perpendicular to the image of $(K(A,B))$. This implies $\mathcal{R}_0(T) = \mathrm{im}(K(A,B))$. □

Now we can state how to check controllability easily.

Corollary 1. *For a system (A,B) the following statements are equivalent:*

1. *There exists $T > 0$ such that (A,B) is controllable at time T.*
2. $\mathrm{im}(K(A,B)) = \mathbb{R}^n$
3. $\mathrm{rank}(K(A,B)) = n$
4. *For all $T > 0$ the system (A,B) is controllable at time T. We say that (A,B) is controllable.*

Remark 1. *Controllability is invariant under change of coordinates. Define $z(t) := T^{-1}x(t)$ for an invertible matrix $T \in \mathbb{R}^{n\times n}$. Then*

$$\begin{aligned} T^{-1}TT^{-1}\dot{x}(t) &= T^{-1}ATT^{-1}x(t) + T^{-1}Bu(t) \\ \dot{z}(t) &= \tilde{A}z(t) + \tilde{B}u(t) \end{aligned}$$

Now we compute the Kalman-matrix.

$$\begin{aligned} K(\tilde{A},\tilde{B}) &= K(T^{-1}AT, T^{-1}B) \\ &= [T^{-1}B, T^{-1}ATT^{-1}B, ..., (T^{-1}AT)^{n-1}T^{-1}B] \\ &= T^{-1}[B, AB, ..., A^{n-1}B] \\ &= T^{-1}K(A,B) \end{aligned}$$

This shows that the image of $K(A,B)$ equals the image of $K(\tilde{A},\tilde{B})$.

Now we have a powerful tool to check whether or not a system is controllable. Indeed we will use Corollary 1 to show the controllability of our problem. Next, we will give a tool to control a system.

2.2 Pole Placement Theorem

As we may know from the theory of differential equations stability of a system can be analysed by considering the eigenvalues of the matrix A. We might ask us what eigenvalues and controllability have in common. In this section we will see how we can make a system (A,B) become stable if it is controllable. Therefore we introduce the *pole placement theorem*. An important question is whether for any given choice of eigenvalues there exists a feedback gain F that achieves the desired closed-loop eigenvalues, also know as poles. We will see that this is indeed the case if the system is controllable.

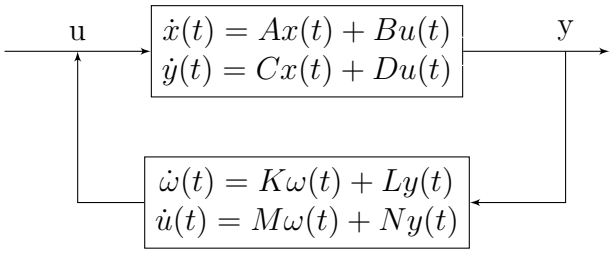


Figure 4: Block diagram of a feedback loop

In the following section we will discuss how to achieve stabilisation through state feedback if possible. A very powerful theorem will be the *pole placement theorem*.

We want to determine a feedback controller as represented in figure 4 such that the closed-loop system

$$\begin{pmatrix} \dot{x}(t) \\ \dot{\omega}(t) \end{pmatrix} = \begin{pmatrix} A + BNC & BM \\ LC & K \end{pmatrix} \begin{pmatrix} x(t) \\ \omega(t) \end{pmatrix}$$

is internally stable. A system (A, B) is said to be *internally stable* if $\sigma(A) \subseteq \mathbb{C}_-$, i.e. if all eigenvalues of A have negative real part. Such a feedback controller is called *stabilising controller*.

Lemma 2. *Let $\lambda \in \mathbb{C}$ be an uncontrollable eigenvalue of (A, B), i.e. $\operatorname{rank}[A-\lambda I, B] < n$. Then for all matrices $F \in \mathbb{R}^{m\times n}$ we have $\lambda \in \sigma(A+BF)$, where $\sigma(\cdot)$ denotes the spectrum of a matrix.*

Theorem 2 (Pole Placement)**.** *Let (A, B) be a system as in (9). The the system is controllable if and only if for all monic, i.e. the leading coefficient is 1, polynomial $p(s) \in \mathbb{R}[s]$ with $\deg(p(s)) = n$ there exists $F \in \mathbb{R}^{m\times n}$ such that $p(s) = \det((A+BF)-sI)$.*

To proof the pole placement theorem we need the following theorem and lemma.

Theorem 3. *The following two maps $\Gamma : \Sigma^C_{n,1} \to \Sigma^C_{n,1}$ are canonical forms:*

$$\textit{i)}\ \Gamma(A, b) = \left(\begin{pmatrix} 0 & & & a_1 \\ 1 & \ddots & & \vdots \\ & \ddots & 0 & a_{n-1} \\ & & 1 & a_n \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \right)$$

$$\textit{ii)}\ \Gamma(A, b) = \left(\begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ a_1 & \dots & a_{n-1} & a_n \end{pmatrix}, \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \right)$$

where in both cases the constants $a_1, ..., a_n$ are determined by the characteristic polynomial

$$det(sI - A) = s^n - [a_n s^{n-1} + ... + a_2 s + a_1] \in \mathbb{R}[s].$$

Lemma 3. *Let the system (A, B) be controllable. Then there exist $u_0, ..., u_{n-1} \in \mathbb{R}^m$ such that $x_1, ..., x_n$ defined by $x_0 := 0$, $x_{k+1} = Ax_k + Bu_k$ for $k = 0, ..., n-1$ are linearly independent.*

Proof Theorem 2. First we will show the if-part by seeking a contradiction. Assume that (A, B) is not controllable. Then there exists an uncontrollable eigenvalue λ. By Lemma 2 we have that $\lambda \in \sigma(A + BF)$ for all matrices $F \in \mathbb{R}^{m\times n}$. If $p(s) \in \mathbb{R}[s]$ is such that $p(\lambda) \neq 0$, then $p(s) \neq \det((A + BF) - sI)$ for all $F \in \mathbb{R}^{m\times n}$, which is a contradiction.

Now we will show the only-if-part. Let $m = 1$. By Theorem 3 and Remark 1 we may assume that (A, B) is in controller canonical form, i.e.

$$A = \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ -a_n & \dots & -a_2 & -a_1 \end{pmatrix} \qquad B = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

Let $F = [f_n, ..., f_1] \in \mathbb{R}^{1\times n}$, then the determinant of $(A + BF) - sI$ is given as

$$\det((A + BF) - sI) = s^n + (a_1 - f_1)s^{n-1} + ... + (a_n - f_n).$$

If $p(s) = s^n + p_1 s^{n-1} + ... + p_n \in \mathbb{R}[s]$ is given, we may choose $f_k := a_k - p_k$ for $k = 1, ..., n$. Now choose $m > 1$, $u_0, ..., u_{n-1}$ and $x_1, ..., x_n$ as in Lemma 3 and let $u_n \in \mathbb{R}^m$ be arbitrary and $b := Bu_0$. Set $F_0 := [u_1, ..., u_n][x_1, ..., x_n]^{-1} \in \mathbb{R}^{m\times n}$. Then $F_0[x_1, ..., x_n] = [u_1, ..., u_n]$ and

$$x_{k+1} = Ax_k + Bu_k = (A + BF_0)x_k$$

for $k = 1, ..., n-1$. This implies

$$x_k = (A + BF_0)^{k-1}x_1 = (A + BF_0)^{k-1}b$$

for $k = 1, ..., n$. We now see that $[b, (A + BF_0)b, ..., (A + BF_0)^{n-1}b]$ is invertible which implies that $((A + BF_0), b)$ is controllable. This system can be seen as a single input system. We now use the case $m = 1$. There exists $f \in \mathbb{R}^{1\times n}$ such that

$$\det((A + BF_0 + bf) - sI) = p(s)$$

where

$$BF_0 + bf = BF_0 + Bu_0 f = B(F_0 + u_0 f) =: BF$$

This finishes the proof. □

We see that if the system is controllable we can place the eigenvalues everywhere we like and therefore are able to stabilise our system. Next we will apply this theory to our problem.

2.3 Controller Design

Next we will present a controller to achieve that car and the trailer become a stable system. The first step is to linearise the model about $\phi = 0$ and $\delta = 0$ using a Taylor series:

$$\Delta\phi = \phi - \phi_0.$$

This linearisation is needed to apply the theory presented in § 2.2. The desired form of the system is given as in equation (8). Here, D is 0 since we do not have a feed-through in our system. For our system we get the following expression

$$\Delta\dot{\phi} = \frac{V}{L_3}\Delta\phi - \frac{V}{L_1}\Big(1 + \frac{L_2}{L_3}\Big)\Delta\delta. \tag{10}$$

It can easily be seen that $A = \frac{V}{L_3}$, $B = -\frac{V}{L_1}(1 + \frac{L_2}{L_3})$, $x = \Delta\phi$ and $u = \Delta\delta$.

The system is given in standard form and therefore it is controllable which means our problem is solvable. This fact can also be proven over the rank-criterion. The Kalman-matrix is $K(A, B) = B$. Since B is a non-zero scalar the Kalman-matrix has full rank and therefore the system is controllable. We will use full-feedback, i.e. the pole placement theorem 2, to stabilise the system. In figure 5 we can see how this controller looks like and how it can be modelled.

We know how the system behaves when there is no actual controller. We introduce a closed-loop pole to control the system. We assume that we have a full-state feedback system, meaning that all state variables are known to the controller at all times.

For simplicity, let's assume the reference is zero, $r = 0$. The input is then

$$u = -Kx$$

The state-space equations for the closed-loop feedback system are, therefore,

$$\begin{aligned} \dot{x} &= Ax + B(-Kx) = (A - BK)x \\ y &= Cx \end{aligned} \tag{11}$$

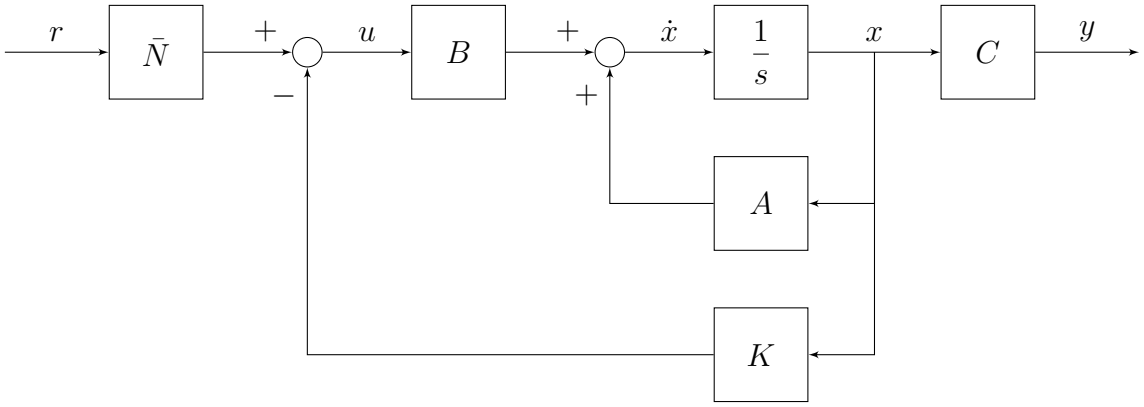


Figure 5: Block diagram of the state space model

The stability and time-domain performance of the closed-loop feedback system are determined primarily by the location of the eigenvalues of the matrix $(A - BK)$, which are equal to the closed-loop poles. Since the matrices A and BK are both 1×1, there will be one pole for the system. By choosing an appropriate state-feedback gain matrix K, we can place these closed-loop poles anywhere we would like because the system is controllable. The pole placement theorem provides the theory that we can compute such a matrix K.

In the block diagram 5 one may see that we use a factor $\bar{N}$ in the calculation of the time response. This scale factor $\bar{N}$ is used to scale the reference input to make it equal to Kx in steady-state, ensuring that Kx will be equal to the desired output, i.e. that we do not get a non-zero steady state error. This $\bar{N}$ is computed in general as $\bar{N} = N_u + KN_x$ where $N_x = N(1:n)$ and $N_u = N(1+n)$ using MATLAB notation and n denotes the dimension of A. The vector N is

$$N = \begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

where $[0, ..., 0, 1]$ is a vector of size $n + 1$, shortly represented as $[0, 1]$.

3 Implementation using MATLAB

Before starting the actual implementation, we need realistic physical values for the different parameters involved in the problem.

For the physical parameters of the car, we will use the specifications of a Mercedes-Benz C-Class S203 (as seen in figure 6): $L_1 = 2.715\,\text{m}$, $L_2 = 1.169\,\text{m}$. We will assume the distance from the trailer axle to the hitch is $L_3 = 1.2\,\text{m}$.

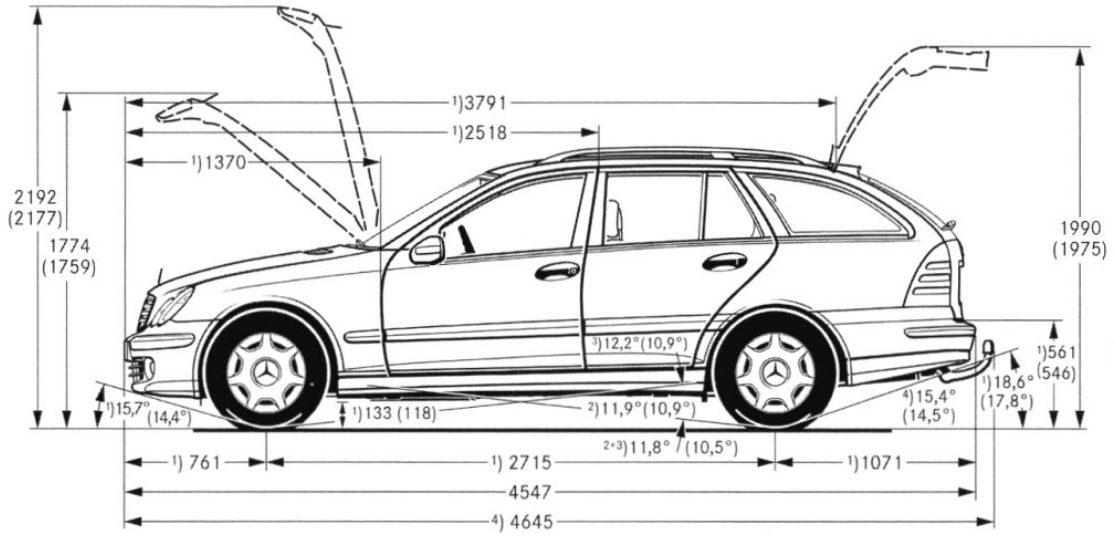


Figure 6: Side view diagram of a Mercedes-Benz CLC S203 AMG

3.1 Programming the Controller

The first need we need to do is to define some of the physical parameters of the system defined in table 1, this is shown in snippet 1 below:

Snippet 1: Physical parameters for the model

```
%% Parameters
% Reference angle
r = 0.25;
% Initial condition in degree
phi_0 = r + 0.1;
% Velocity in meters per second
V = 3.0;
% Example for Mercedes-Benz CLC S203 AMG in meters
L1 = 2.715;
L2 = 1.169;
L3 = 1.2;
```

As discussed in § 2, the eigenvalues of the system matrix, A, (equal to the poles of the transfer function) determine stability. In our system (10) we see that A is positive; thus

the system is unstable. The parameters to model the controller are defined in snippet 2 below:

Snippet 2: Basic parameters to model the controller

```
%% Model
A = V/L3;
B = V/L1*(1+L2/L3);
C = 1;
D = 0;
```

One of the first things we want to do is analyse whether the open-loop system (without any control) is stable. To observe what happens to this unstable system when there is a non-zero initial condition, we define the state space using `sys = ss(A,B,C,0)` and then we simulate the output time response $y(t)$ of dynamic system with input $u(t)$ using the function `lsim()`. This implementation can be seen in snippet 3 below:

Snippet 3: Method to calculate open-loop response to non-zero initial condition

```
%% Calculate open-loop response to non-zero initial condition

% Time in seconds
t = 0:0.01:10;
% Input
u = zeros(size(t))*r;
% Create state space object
sys = ss(A,B,C,0);

% Calculate open-loop response to non-zero initial condition
[y,t,x] = lsim(sys,u,t,phi_0);

% Plot the response
figure(1)
  plot(t,y)
  hold on
  plot(t,u)
  str_num = num2str(phi_0);
  str = strcat('Open-Loop␣Response␣to␣Non-Zero␣Initial␣Condition␣
     \phi_0=', str_num);
  title(str)
  xlabel('Time␣(sec)')
  ylabel('\phi␣(rad)')
  legend('Open-loop␣response', 'Reference')
  hold off
```

In figure 7 below we can see the uncontrolled behaviour of the system when there are no closed-loop poles (i.e., open-loop response). It's remarkable how quickly the jackknifing phenomenon occurs.

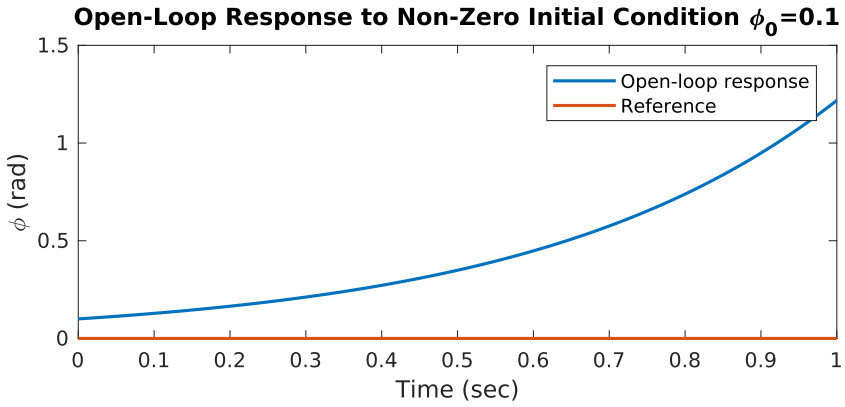


Figure 7: Open-Loop response to non-zero initial condition driving backwards with a trailer

It is also worth noting that just as we expected from (7 bis), when $V < 0$ (using `V = -3.0;` for instance) the system is stable, as seen in figure 8 below. In the figure, it can be clearly seen how when the car moves forward, the angle ϕ is corrected in a matter of seconds and then trailer flawlessly follows the car.

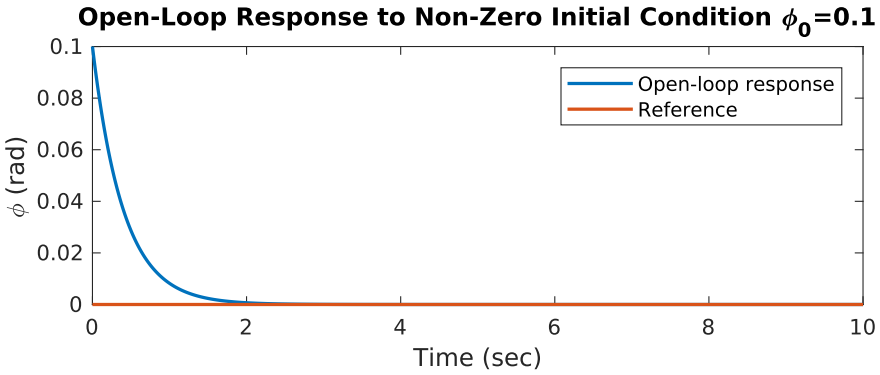


Figure 8: Open-Loop response to non-zero initial condition driving forwards with a trailer

Now that we know how the system behaves when there is no actual controller, let's see what happens when we introduce a closed-loop pole to control the system. We will work with several reference trajectories (discussed in § 3.2) to see how the controller can handle different scenarios.

The major difference between this method and the previous one is that we now build the controller for the system using a pole placement approach and assuming we have a full-state feedback system, meaning that all state variables are known to the controller at all times.

We use the function `place()` to find the state-feedback gain, K, which will provide the desired closed-loop poles. Then we define the state space using `sys_cl = ss(A-B*K,B,C,0);` and simulate the output time response $y(t)$ of dynamic system with input $u(t)$ using the function `lsim()`, as seen in snippet 4.

One may see that we use a $\bar{N}$ factor in the calculation of the time response. We use the function `rscale(sys,K)` to compute the desired $\bar{N}$ like presented in § 2.3.

Snippet 4: Method to calculate closed-loop response to non-zero initial condition

```
%% Calculate closed-loop response to non-zero initial condition

% Time in seconds
t = 0:0.01:60;
% Inputs
u = curve1(t, 0.35); % Standard curve
% u = curve2(t, 0.4); % U turn
% u = curve3(t, 0.65); % Sharp curve
% u = curve4(t, 0.4); % S shape

% Apply pole placement
K = place(A,B,-0.73);
% State space
sys_cl = ss(A-B*K,B,C,0);
% Calculate scaling factor
Nbar = rscale(sys,K);

% Calculate closed-loop response to non-zero initial condition
[y_cl,t,x_cl] = lsim(sys_cl,Nbar*u,t,phi_0);

% Plot the response
figure(2)
  plot(t,y_cl)
  hold on
  plot(t,u)
  str = strcat('Closed-Loop␣Response␣to␣Non-Zero␣Initial␣Condition␣
     \phi_0=', str_num);
  title(str)
  xlabel('Time␣(sec)')
  ylabel('\phi␣(rad)')
  legend('Closed-loop␣response', 'Reference')
  hold off
```

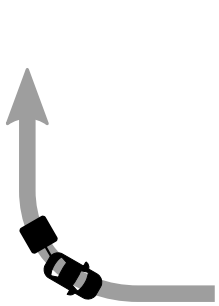
This implementation of a controller is the one previously illustrated in figure 5.

3.2 Controlling the Trailer on Different Curves

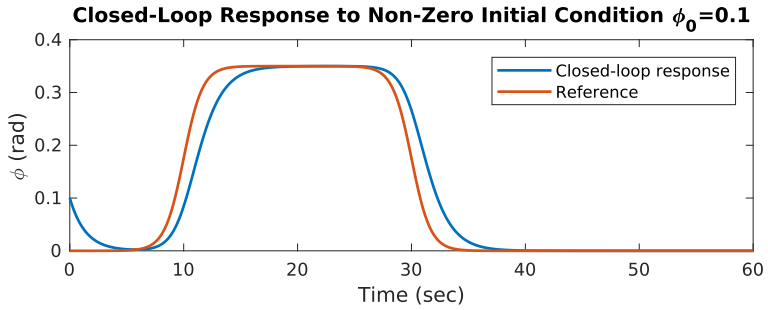
In this part we will see how the controller performs in terms of reaction time and fidelity with different trajectories. In particular we will work with (a) a short sharp curve, (b) a U turn, and (c) an S-shaped curve.

Standard Curve

In figure 9a we can see the shape of the trajectory followed by the car-trailer system, and in figure 9b we can see how the response of the controller to the input trajectory (defined in snippet 5).



(a) Diagram of the curve



(b) Closed-loop response of the controller

Figure 9: Diagram and control of the car-trailer system in a short, sharp curve

Snippet 5: Function used to simulate a standard curve

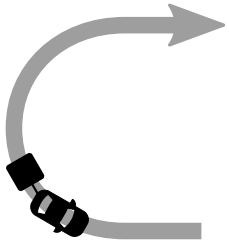
```
function [ x ] = curve1( t, height)
    steep = 2;
    n = length(t);

    x_1 = logistic(height, steep, linspace(-6,6,floor(n/3)), 0);
    x_2 = flip(x_1);

    x = [ x_1 height x_2 zeros(1,floor(n/3)) ];
end
```

This curve, along with a straight line, are the basis of basically all possible trajectories a car-trailer system can make whilst parking.

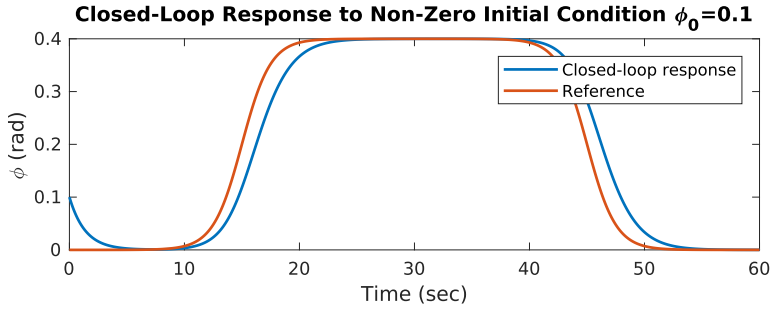
U-Turn Curve

A *U* turn simply consists on steering the wheel a longer time to the standard curve pr

In figure10a we can see the shape of the trajectory followed by the car-trailer system, and in figure 10b we can see how the response of the controller to the input trajectory (defined in snippet 6).



(a) Diagram of the curve



(b) Closed-loop response of the controller

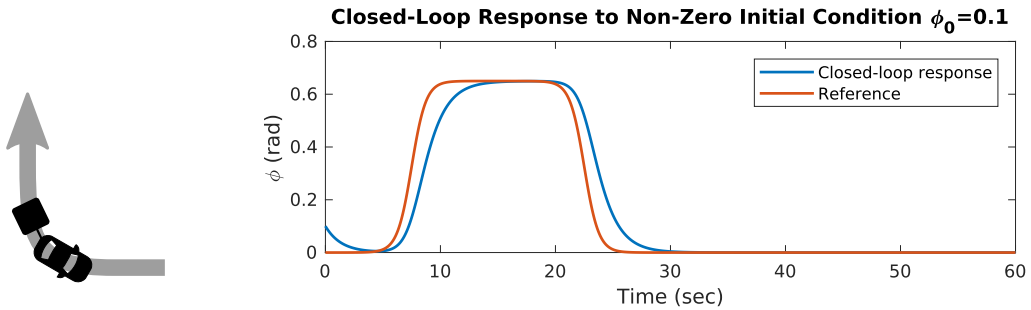
Figure 10: Diagram and control of the car-trailer system in a *U* turn

Snippet 6: Function used to simulate a U turn

```
function [ x ] = curve2( t, height)
  steep = 2;
  n = length(t);

  x_1 = logistic(height, steep, linspace(-6,6,floor(n/2)), 0);
  x_2 = flip(x_1);

  x = [x_1 height x_2];
end
```

Short, Sharp Curve

In figure 11a we can see the shape of the trajectory followed by the car-trailer system, and in figure 11b we can see how the response of the controller to the input trajectory (defined in snippet 7).



(a) Diagram of the curve

(b) Closed-loop response of the controller

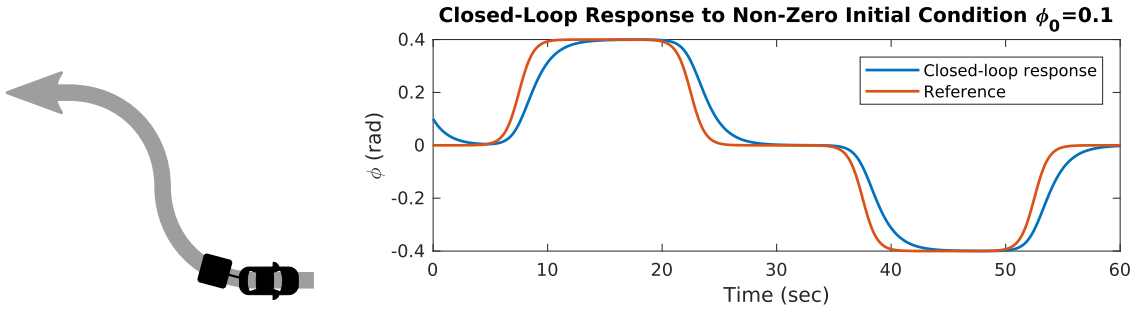
Figure 11: Diagram and control of the car-trailer system in a short, sharp curve

Snippet 7: Function used to simulate a short sharp curve

```
function [ x ] = curve1( t, height)
  steep = 2;
  n = length(t);

  x_1 = logistic(height, steep, linspace(-6,6,floor(n/4)), 0);
  x_2 = flip(x_1);

  x = [ x_1 height x_2 zeros(1,floor(n/2)) ];
end
```

S-Shaped Curve

In figure 12a we can see the shape of the trajectory followed by the car-trailer system, and in figure 12b we can see how the response of the controller to the input trajectory (defined in snippet 8).



(a) Diagram of the curve

(b) Closed-loop response of the controller

Figure 12: Diagram and control of the car-trailer system in a S-shaped curve

Snippet 8: Function used to simulate an S-shaped curve

```
function [ x ] = curve3( t, height )
  steep = 2;
  n = length(t);

  x_1 = logistic(height, steep, linspace(-6,6,floor(n/4)), 0);
  x_2 = flip(x_1);
  x_3 = [ x_1 x_2 ];
  x_4 = -x_3;

  x = [ x_3 0 x_4 ];
end
```

3.3 Issues with the Model

Main problem about this model is that it is a continuous model which cannot be implemented on a chip without modification. One needs to discretise it. There are several options to do so, each having advantages and disadvantages. One could think about creating a continuous model with continuous model and afterwards doing the discretisation. Another approach would be just to discretise the model and create a discrete version of the controller. These two approaches are not equivalent. Obviously, the first one is more powerful because it will provide insights into the system behaviour for all times t whereas the second one only yield information for the sampling instances kT. The advantage of this method is that it is easier to apply.

Another problem is that the model is obtained by linearisation about the operation point 0°. One improvement would be to use a gain-scheduling controller which is also linear. This group of controller is often applied to nonlinear models. It selects the gain according to some strategy. This strategy e.g. can be time-varying or depended on the state of the system. In our case we would prefer a state dependent scheduler since our system behaves linearly in small regions for different operating points: $[-40°, 40°]$. For implementation we would need to linearise the model for all these operating points with

a sufficient small step size. However, to show feasibility and solvablity of the problem it is sufficient to linearise about one operation point.

4 Alternative Controlling Methods

We have used alternative methods that do not use control theory in order to make the trailer follow the path that we want. These methods focus on different heuristics to determine which δ we have to choose in order to go through the desired path. The first method thinks on a function to optimise in order to be as close as possible to the desired path and as close as possible to the critical point $\phi = 0$. The second one aims to replicate the strategy of expert trailer drivers that have a lot of experience in our problem.

4.1 Optimisation-based control

Using the model derived in § 1.2, we aim to move our trailer in a straight path. Of course, our straight path will start with an angle $\phi \neq 0$, which tries to resemble a situation where we are trying to go backwards but the trailer starts to bend due to some imperfection of the ground/wheels.

We will assume, without loss of generality, that the centre of mass of our car starts at the origin. As we want the car to go straight, we will make the car go backwards in the x axis, and the car will start aligned with this axis. On the other hand, the trailer will not start aligned with the x axis (if not, the solution would be trivial, setting $\delta = 0$ the whole time would solve the problem). Our aim is to go through the x axis during a quantity of time T. However, what does going through the x axis mean? We will set a tolerance Δy, and if at time T the trailer is at a distance from the x axis lower than Δy, we will take it a success. In mathematical terms, we have to decide a function $\delta(t)$, for $0 \leq t \leq T$, such that $|y(T)| < \Delta y$ and $|\phi| < \phi_{crit}$, where ϕ_{crit} is the critical jackknifing angle.

To find the *steering function* $\delta(t)$, we will use a brute force method. We will split the interval $[0, T]$ into n interval times of length T/n. At the beginning of each interval, we will choose which is the δ that puts the trailer in the best possible configuration at the end of the interval. We will simulate the trajectory of the whole system in order to decide how good is a choice of δ. In addition, we have to determine when is a configuration better than another one. To do this, we have chosen a cost function. The cost function that we have chosen depends on the angle ϕ and the distance from the x axis, $|y|$. We have taken:

$$\text{cost}_\alpha(y, \phi) = y^2 + \alpha\phi^2 \tag{12}$$

The cost function defined above depends on a parameter, α. The higher α is, the more we are interested in keeping the trailer aligned. The lower α is, the more we are interested in keeping the car on the path regardless of the trailer. Then, α is a parameter that needs to be tuned in every different geometrical configuration.

Now that we have our cost function defined, our method to find the steering function $\delta(t)$ works as follows: for each of the time intervals that we have divided our time into, at the beginning of the interval we choose the δ that minimises cost_α at the end of the interval. The way to find the optimal δ is by means of brute force: We search in a grid of angles in the interval $[-\delta_{max}, \delta_{max}]$ and number of elements n_δ. With this method, the output function $\delta(t)$ is a piecewise function that at every time interval has a constant value.

One of the main drawbacks of the method is that, given a time T, the method depends on three parameters that we do not know which value they should have a priori: n, n_δ and α. One could argue that, the higher the n, the more chances that the method can find the solution. However, this is far from the truth. The main problem is that method tries to optimise the configuration of the system at the next time step: however, this does not in principle guarantee that this configuration will be optimal after that time step. For this reason, having a very large n can be considered very elitist, thus not allowing us to find the proper steering function. These kind of elitist searches are usually known as greedy algorithms.

We have implemented the previous method in C, using the analytical expressions for the circular paths of the points in the car and using a Runge–Kutta–Fellberg 78 method to integrate the differential equation for ϕ. We have used sensible data for the length and width of each of the elements of the system. Our aim would be to keep the system going straight for a given amount of time T. When T is large enough ($T \approx 10$ seconds), the system is very hard to be controlled in the whole interval. We have tried many values of α, n and n_δ, but they all fail when given a sufficiently large amount of time T.

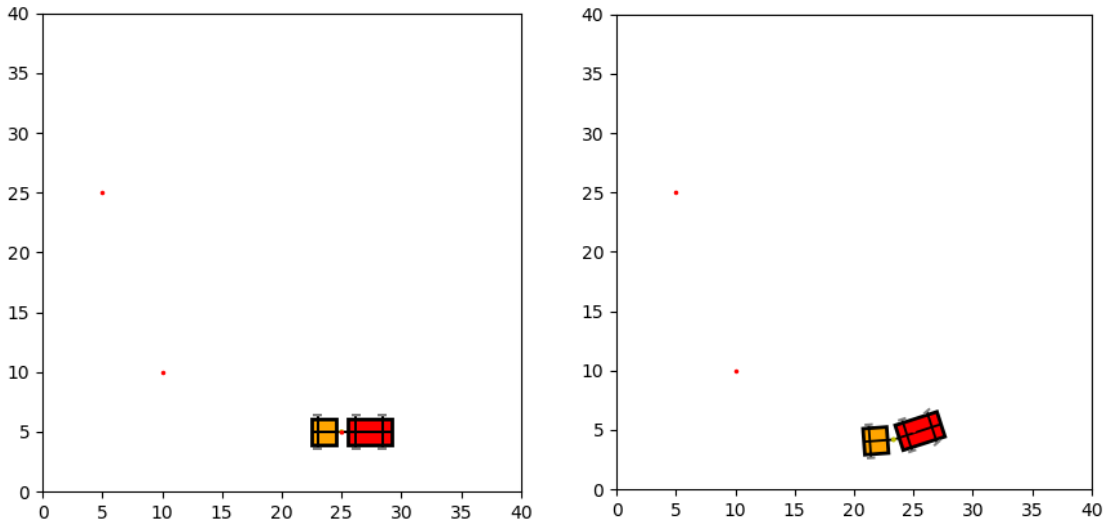
We think that this method does not work due to the fact that the method only focuses on optimising the next time step. This is a main drawback, as not only it forgets about time steps that follow the next one, but it may also be that optimising the next time step is bad for the long run. In terms of optimisation, we can say that we are constantly being elitist, and this is usually not the best strategy when we want to optimise complicated functions. For this reason, the next method is a little bit more exploratory, that is, it does not always look at the best solution in its neighbourhood, but in the long run it performs better.

4.2 Steering-based control

In this section we describe the method to get the trailer from one point to another by using steering strategies that reassemble the ones used by experienced drivers.

The method is made up of iteratively-repeated steering steps that we have defined as a *manoeuvre*. A *manoeuvre* consists in two steps: the first one, to bend the trailer into the right direction, and the second one, to move the car towards the desired end.

We will call the first part of the *manoeuvre trailer facing*. The way to perform *trailer facing* is the following. Assume that if you had to go from the beginning point to the final point *without* a trailer you would have to take $\delta > 0$ to go upwards. In this case, the trailer would quickly face the opposite direction, it would go downwards. In this way, in order get the trailer facing to the right direction, we need to start by taking $\delta < 0$. This is counter-intuitive: we are moving in the wrong direction. We move into the wrong direction in order to face the trailer into the right direction. The problem of starting to move into the right direction is that the trailer faces the wrong direction and then we would have to correct it afterwards. This later correction would not allow us to achieve the final point.



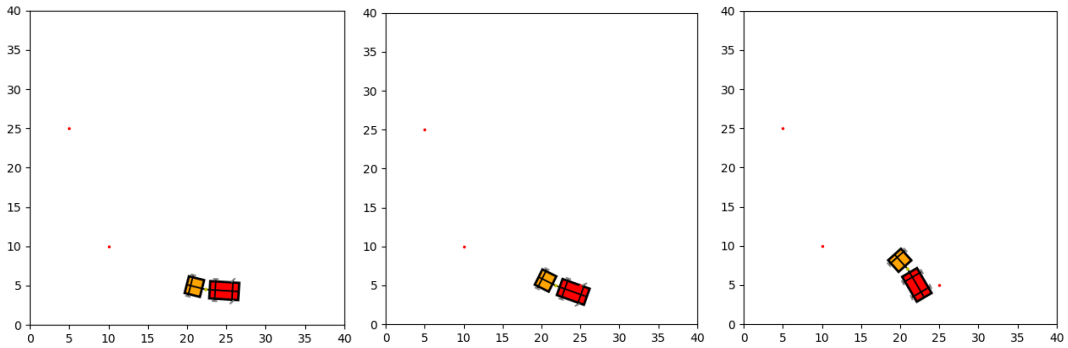
(a) Initial situation (b) Situation after the first trailer facing

Figure 13: Trailer facing

In the picture above, we can see an example of the *trailer facing* working. The initial conditions are the ones shown in 13a, and our aim is to arrive to red point at (10, 10). If we were not using a trailer, we would steer the car such that it went towards higher y. However, as we can see in 13b we are doing the exact opposite: we are going to even lower y. Once we have done this, the trailer is properly faced: when we go to the red point, the trailer will not jackknife as easily as if we had gone to the red point when we were in 13a. This is because, when going to the red point, the trailer will bend to the opposite direction after the trailer facing, thus having to cross through the point $\phi = 0$.

However, we still have to decide what having a proper facing means. We will set a ϕ_{max} at which we end the trailer facing. This ϕ_{max} will not be static. If we are far away from the end point, we will set a high ϕ_{max}, as we want to bend more. If we are close to the point, we will set a lower ϕ_{max}, as we do not need to bend that much. In all cases, when $\phi = \phi_{max}$ we will end the trailer facing and start the car facing.

Once we have the trailer facing into the right direction, we start the second part of the manoeuvre: car facing. In this part, we act as if we had no trailer and we choose the δ such that the car goes to the end point. We do this until the trailer is faced into the wrong direction, and then we go back to the first part of the *manoeuvre* again.



(a) Faced trailer (b) Trailer with $\phi \approx 0$ (c) Trailer wrongly faced

Figure 14: Trailer facing

In the picture shown above we can see how the trailer goes from being properly faced to a facing that if goes on it can cause a jackknife. In this picture we can see how the car gets closer to the red point (y grows), and the price to pay is the misalignment of the trailer. Once it is enough misaligned, we will start with the trailer facing.

So far, we have just told about choosing $\delta > 0$ or $\delta < 0$, but we haven't specified which δ to use. In order to make the model more realistic, $\delta(t)$ will be a continuous function. That is, we are allowed a maximum variation of angle $\Delta\delta$ every time step Δt. That is, the maximum steering speed of the driver is $\Delta\delta/\Delta t$. For this reason, we can't always aim at the optimal δ every time step, as, if we were steering at δ_0 at the last step, we can only aim at the best δ in $(\delta_0 - \Delta\delta, \delta_0 + \Delta\delta)$. How do we choose this best δ? We use the discretised differential equation. Ideally, if we want to make an increase in ϕ of $\Delta\phi$, we have to take the δ satisfying the following equality:

$$\frac{\Delta\phi}{\Delta t} = -\frac{V}{L_3}\sin\phi - \frac{V}{L_1}\tan\delta\left(1 + \frac{L_2}{L_3}\cos\phi\right) \tag{13}$$

Which is given by:

$$\delta = \arctan\frac{-L1\cdot(\Delta\phi\cdot L3 - \sin\phi\cdot\Delta t\cdot V)}{\Delta t\cdot V(L3 + L2\cdot\cos\phi)} \tag{14}$$

However, we cannot always choose this δ, as this δ might not be in the interval $(\delta_0 - \Delta\delta, \delta_0 + \Delta\delta)$. For this reason, if $\delta > \delta_0 + \Delta\delta$, we choose $\delta = \delta_0 + \Delta\delta$. If $\delta < \delta_0 - \Delta\delta$, we choose $\delta = \delta_0 - \Delta\delta$.

Of course, we still need a criteria to decide which is the $\Delta\phi$ value that we want to attain. Different choices can be made, but we have chosen just to focus on the direction, not thinking about the actual value of ϕ, for simplicity. If we want the trailer to go towards negative ϕ values, we will just set $\Delta\phi = -1$, and if we want it to go towards positive values we will choose $\Delta\phi = 1$. The decision of going towards positive and negative values is made upon the trailer facing or car facing criteria.

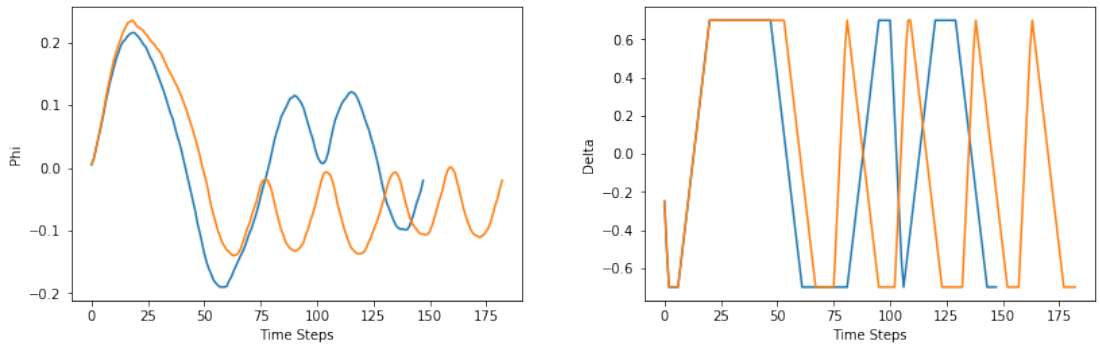
4.3 Comparison of the two methods

As we have said before, the first method works by constantly looking for the optimal solution at the next time step. We have checked that this method does not work when aiming at a sufficiently large amount of time. The main drawback of this method is that the optimal configuration in the next step might not be good in the long run. On the other hand, the steering-based method does not have this problem. Sometimes, it starts bending to the opposite side and getting much worse locally. However, this allows it to get better globally.

4.4 Further improvements

From experience, it is hard to make the trailer go straight, even when the axles are aligned (due to the fact that alignment is not perfect, and the equilibrium point is unstable). To model this, we can introduce some perturbations in the angle of the trailer. After doing an integration step, we will add some uniform noise to the ϕ angle, with expectation value 0. We choose this kind of noise because a priori we do not have any preferred direction of the perturbations. The interval $(-r, r)$ of the uniform random variable that models the noise will have a fixed length $2r$, which has been chosen small (but way bigger than 10^{-16}, which is the *numerical noise* given by the finite decimals of the representation of the double precision numbers). The randomisation just affects to the kinematics, as the method to control the trailer is based on the same criteria. However, every trajectory of the trailer is different, due to the random effects. As, with $r = 0.025$, the trailer finishes successfully the path, this means that our method is robust and could be applied to ground with some slippery.

To illustrate that every simulation is different, and although the method is the same, it provides different steering functions $\delta(t)$, we have plotted in two runs of the method how ϕ and δ vary over time.



(a) Comparison of $\phi(t)$ evolution

(b) Comparison of steering functions $\delta(t)$

Figure 15: Variation of the angles in different runs of the algorithm

We can see that one of the methods (blue) finishes earlier. Although having very different paths, they both terminate successfully. We can see that the initial conditions are similar but at some point they start to differ. After they start to differ, the steering functions change, as they adapt to the current ϕ value. As the steering starts to differ, this makes the path change more and more, ending with very different trajectories.

5 Conclusions

In the previous sections we have three different approaches to implement a controller to aid the driver in the process of parking with a trailer attached to their car:

- State space control (using MATLAB).
- Optimisation-based control (using C).
- Steering-based control (using Python).

Except the optimisation-based control, we showed that all of the models are viable options for control with their own advantages and disadvantages.

We think that as an improvement and further development of our work it would be interesting to test our results. We could consider a simulation or even a real miniature car with a trailer using LEGO MINDSTORMS NXT (or similar) using MATLAB's Simulink or Python's `nxt-python` library.

References

[1] R. N. Jazar. *Vehicle Dynamics: Theory and Application*. Springer-Verlag New York, 2014. ISBN: 978-1-4614-8544-5.

[2] J. Chiu and A. Goswami. The critical hitch angle for jackknife avoidance during slow backing up of vehicle–trailer systems:26, 2014.

[3] E. D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Texts in Applied Mathematics. Springer, New York, 1998. ISBN: 978-1-4612-0577-7.

[4] D. Hinrichsen and A. Pritchard. *Mathematical Systems Theory I: Modelling, State Space Analysis, Stability and Robustness*. Texts in Applied Mathematics. Springer, 2005. ISBN: 978-3-540-26410-1.

[5] H. Trentelman, A. A. Stoorvogel and M. Hautus. *Control Theory for Linear Systems*. Communications and Control Engineering. Springer London, 2001. ISBN: 978-1-4471-0339-4.

[6] J. Nilsson and S. Abraham. *Trailer Parking Assist (TPA)*. Master's Thesis, 2013.