**Title**

Non-Euclidean Vehicle Motion Models



**Author**

Fork, Thomas

**Publication Date**

2024

Peer reviewed|Thesis/dissertation

# Non-Euclidean Vehicle Motion Models

By

Thomas David Fork

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Francesco Borrelli, Chair
Professor Koushil Sreenath
Professor Anil Aswani

Summer 2024

Non-Euclidean Vehicle Motion Models

Abstract

Non-Euclidean Vehicle Motion Models

by

Thomas David Fork

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Francesco Borrelli, Chair

Numerous control applications involve geometry that is not Euclidean. For instance a car on an undulating off-road surface, a motorcycle on a banked racetrack turn, and a drone following a spatial reference trajectory all involve features of interest that are not Euclidean. Understanding these scenarios from a control point of view requires the development of vehicle models which capture these non-Euclidean elements and any resulting changes in the behaviour of said vehicle.

This dissertation develops geometric motion models to describe rigid-body vehicles relative to a surface or curve. Said models are developed in a general sense for the surface or curve, and the vehicle being considered. Examples are provided of several ground vehicle and aircraft models developed using non-Euclidean motion models. Numerical implementation and use of these motion models is discussed with code examples. Finally, current applications of non-Euclidean motion models are discussed along with potential future research areas.

# Contents

# Preface

The past decades have witnessed tremendous development in the analysis, design, and control of vehicular systems, from ground vehicles and submersibles to air and spacecraft. Despite the considerably different theory and technology involved in each, all involve behaviour that may be viewed as involving non-Euclidean geometry. For instance the motion of a car on a three dimensional surface, the tracking of an aerial maneuver by a drone, or the orbital maneuver of a spacecraft may all be viewed in a non-Euclidean coordinate system based upon a surface, maneuver, or other geometric feature of interest.

This dissertation is based on my past four years of work in this area, first on extending ground vehicle models to general three dimensional roads, and subsequently on path-parametric coordinate systems for aircraft. It is written with the intent that example applications are intuitive and familiar to readers in these fields. However, the geometric developments involved are not limited to these applications, and it is my hope that this will inspire future research and application in other fields as well.

Berkeley, USA
August 2024

Thomas Fork

# Acknowledgements

This work has been supported by a number of key individuals and organizations over the year, to which thanks are owed.

First, to my advisor Francesco Borrelli and his advisor Manfred Morari, who both supported the research developments contained within this dissertation, and without which this dissertation would not have materialized.

Special thanks are owed to the developers and maintainers of the CasADi software package. Their software proved instrumental, even essential, to implementing the vehicle models and control applications used throughout my research. I am doubtful that the numerical implementation of my research would have progressed very far without the tools they have made available, and am grateful for their work.

And finally, I would like to thank my family for their unwavering support throughout my years as a graduate student.

# Chapter 1
# Introduction

Consider the motion of a car on a road, a sled on a hill, and a person on a water-slide. These systems are fundamentally different in nature, but exhibit similar motion behaviour. For example, they may each be approximated as a single rigid body which is tangent to and in contact with a surface as it moves. Without this "motion model" we cannot model any of these systems, even if we know the forces acting upon them. One can of course develop more complicated motion models, such as multi-body physics simulators, but for the purposes of control it is desirable to keep things simple while capturing relevant behaviour, an obviously application-dependent trade-off.

Developing new motion models is the fundamental result of this dissertation. The models considered here did not exist previously or were limited in scope. That is, certain roads, hills, and water-slides could only be studied by high-fidelity simulation engines. These engines are incompatible with many control techniques, especially ones used to analyze the stability, performance, robustness, and safety of systems. The first part of this chapter introduces the motion models developed in this dissertation from a purely conceptual perspective. Previous literature is discussed, particularly its limiting factors and what will be used to avoid these.

The remainder of this chapter provides introductory material on notation, Newtonian mechanics, and several examples of vehicles which may be studied using the motion models developed in this dissertation. It is important to emphasize that these examples are by no means comprehensive in nature, and that many other vehicles may be described by the same motion model with different forces acting upon it. For instance, a water-slide enthusiast and a car might both be modeled as a single rigid body tangent to a surface, but the forces actuating each system are different, and as a result pose dissimilar control problems.

At the end of this chapter, Section 1.6 discusses how non-Euclidean motion models are used to build complete vehicle models. This requires different thinking from Euclidean vehicle dynamics approaches, and the reader is encouraged to not skip it.

Following this chapter, Chapter 2 develops general motion models relative to a surface, introducing mathematics necessary to work with the conceptual motion models of this chapter. Chapter 3 does the same for motion models relative to a curve. Numerical implementation of non-Euclidean motion models is discussed in Chapter 4, with a detailed example of building a vehicle model. Example applications for ground vehicles and aircraft are presented in Chapters 5 and 6. These are enabled by the motion models developed in this dissertation, but reflect only a small portion of the potential for future research and application, which is discussed in Chapter 7

The reader interested in jumping directly to example code of applications of this dissertation may find the following to be useful:

Ground vehicles on surfaces: github.com/thomasfork/Nonplanar-Vehicle-Control/

Aircraft relative to curves: github.com/thomasfork/aircraft_trajectory_optimization

## 1.1 Vehicle Motion Models

This dissertation will refer to a motion model as the kinematic (geometric) assumptions on the appearance and degrees of freedom of a system combined with Newtonian mechanics. For example, Figure 1.1 illustrates a tangent contact motion model on a road surface. Conceptually, this is a rigid body which remains tangent to a surface at a fixed normal offset from it. This body may move around on the surface, but is assumed to be fixed to it, either by hard practical constraints, or that the system is controlled to avoid going airborne. No information is present as to what vehicle this model corresponds to, however by providing information on the forces and moments which actuate it, one may build a model of a car or other ground vehicle upon this motion model. Note that in the case of ground vehicles on a road it can be useful to refer to the motion model as a road model [1], as it is in essence a model of how a vehicle moves over a road. The motion models studied here, not all of which invovle a road, are introduced next.

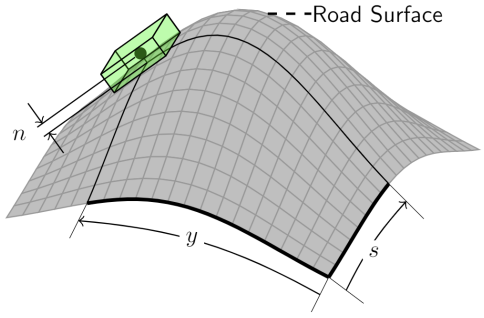



Figure 1.1: Illustration of tangent-contact motion model: A single rigid body tangent to an in contact with a road surface. The body may move across the surface, but may not move normal to it. The body must simultaneously translate and rotate when moving over a curved surface.

### 1.1.1 Motion Models Relative to a Surface

Many ground vehicles exhibit behaviour which is determined in part by a surface: the surface which the vehicle is moving over. This behaviour may be extremely complex, as is the case for off-road vehicles, with deformation of wheel and suspension elements, and the potential for a vehicle to lose contact with the ground. From the perspective of high level planning and control however, these effects are distracting if modeled in full, and are often approximated by keep-out zones, reduced speed zones, or assuming reduced road friction. Note that vehicles such as trains and roller-coasters are the special case of one-dimensional motion over a surface, and may be studied by constraining one degree of freedom of the next motion model. Two simplified motion models relative to a surface are introduced next.

#### 1.1.1.1 Tangent Contact Surface Motion Model

In this motion model, a vehicle is treated as a rigid body tangent to and in contact with a surface. This is trivial when done for a surface that is flat, the most common motion model used for cars [2]. Minor extensions can be made such as constant slope [3], straight motion on a banked road [4] or simple crests and dips [5].

Extending the tangent contact motion model to more general, three-dimensional geometry was first done in [6], with the consideration of a ribbon-surface. In this approach a road is modeled with a three-dimensional curve, with linear (ie. flat) cross-section. While this work was later extended to the case of a curved cross-section [7], it establishes a worrisome trend: each new change to the surface must be derived anew. Furthermore, the general progression has been to start with flat roads and add mathematical treatment to handle more and more general roads. It is worth questioning how long such a procedure can be sustained before the algebraic skills of the authors become insufficient, or if this can ever capture surfaces which do not have roads, such as off-road driving. Furthermore, assumptions on how a vehicle moves are not strictly enforced. For instance all of the previous methods involve approximations which in effect mean a vehicle does not precisely remain tangent to the surface when the ribbon is in torsion [8] or at large angles [7]. Without strictly enforcing motion assumptions it is hardly possible to say if a resulting vehicle model is accurate.

To resolve this, my work employs a novel tactic: instead of trying to "augment" existing motion models with more and more features, a general motion model is developed, of which all existing models are a special case. This is made possible in Chapter 2 by beginning with the perspective that the road surface is a near-arbitrary parametric surface, providing a framework to describe nearly any road surface and apply this motion model to it. Furthermore, the surface parameterization itself provides natural and flexible coordinates to describe vehicle pose relative to the surface. These provide simpler equations of motion than trying to work directly with constrained SE3 kinematics, and may hold special meaning in practice, such as encoding a road with one coordinate.

This motion model is obviously simplistic in nature. Not only have we assumed that the body never leaves the surface, meaning a car never goes airborne, but implicit assumptions exist on the scale of the vehicle. For instance, this model does not capture the motion of a car over a speed bump. Neither does it capture the deformation of a cars suspension. Some of these assumptions may be controlled in practice, for example a car may be slowed down to avoid going airborne, but are ultimately vehicle and application dependent.

#### 1.1.1.2 Unconstrained Surface Motion Model

One of the key observations to develop the tangent contact motion model generally was to use a parametric surface to describe where a vehicle is and capture surface shape effects. This same observation can be used to develop a second motion model: one in which tangent and contact constraints are omitted, but vehicle position and orientation are still described relative to the surface. In this setting, a vehicle may exhibit roll, pitch, and heave motion. This motion model is useful to describe vehicles in which suspension or other body deformation should be approximated with greater fidelity, such as incorporating mass-spring-damper models of a four-wheeled vehicle. This motion model is developed in Section 2.2.

### 1.1.2 Motion Models Relative to a Curve

Certain vehicles, particularly aircraft, exhibit behaviour that has no correspondence to a surface. However, there are many cases where aircraft should follow a spatial curve or trajectory. In the theme of this dissertation: leveraging geometry for control model development, this problem is studied in a similar manner to the surface-relative models. The end result here is a motion model which transforms the coordinate system used to describe the position of an aircraft, as a result of which following a curve is equivalent to regulating one or more coordinates to zero. Whereas the motion of a ground vehicle on a three-dimensional surface is influenced by the surface, the transformation here is purely geometric in nature: aircraft motion will be unchanged, but described in a new coordinate system, one motivated by the goal of encoding control concepts in geometry, such as a reference curve that should be followed. Geometrically, this motion model has the appearance of a moving frame of reference: a 2D Euclidean coordinate system that follows a curve, as shown in Figure 1.2. Notice that in addition to choosing a curve, we have also chosen the orientation of the relative coordinates along this curve, a second degree of freedom.

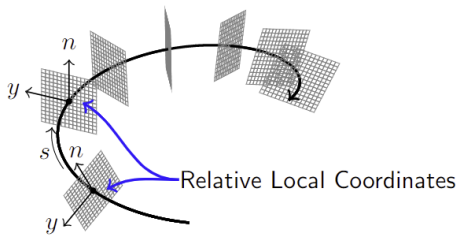



Figure 1.2: Illustration of path-parametric motion model. Adapted from [9].

As is the case for surfaces, prior art exists in developing motion models relative to a curve, but has undesirable model approximations or limitations to specific curves. For example [10] introduces an approach where the centerline must be parameterized by its arc length, and the $y$ coordinate direction in Figure 1.2 is the direction of curvature. However, the direction of curvature may be discontinuous and does not exist for straight lines, in which case vehicle models no longer exist. [11] makes identical arc-length assumptions and builds a geometrically correct motion model based on user-input curvature of the moving frame of reference. These restrictions can be problematic when not using arc-length and curvature parameterizations, such as when fitting splines to known spatial geometry. [12] avoids arc-length assumptions for the special case where the cross-sectional coordinate system in Figure 1.2 is relative to gravity, which is undefined at any point where the curve is parallel to gravity.

Chapter 3 develops this motion model with general choice of the curve and relative coordinates and captures previous approaches as special cases.

### 1.1.3 Multi-Body Systems

Many systems are more complicated in nature than a single body interacting with a surface or other geometric feature. This is self-evident with the inertial effects of engines and other spinning parts of a vehicle, but more complicated systems are common practice. For instance,

the motion of a motorcycle on a three-dimensional surface is never restricted to be tangent to the surface: both the motorcycle and the rider may camber sideways while driving, and the driver may move in other ways as well.

However, a motorcycle may be modeled with the assumption that it cambers about a body-fixed longitudinal axis. This allows the tangent contact model to be used by introducing a fictitious frame of reference in tangent contact with a road surface. Camber motion is present between this frame and the actual motorcycle, but the motion model used may adequately capture vehicle motion for the purposes of control. Special considerations are necessary to capture the linear and angular momentum of the system and derive mechanical laws, however this process will allow us to develop a motorcycle model on a general three-dimensional surface. This is introduced in Section 1.5.3 and implemented in Section 4.5.2.

This same approach can be applied to other multi-body systems, such as spherical robots. However, this dissertation will not study the case of multi-contact systems beyond the motion models introduced so far. For example, truck-trailer systems and legged robots are out of present scope but are also areas of potential future research.

## 1.2 Coordinate-Free Vector Convention

In many engineering treatments of vectors they are assumed to be expressed in a coordinate system. For instance, one might describe the position $x$ of an object in a coordinate system as a function of time ($t$) as:

$$x(t) = [x_1(t), x_2(t), x_3(t)]^\top .$$

This is an intuitive but flawed way to represent a vector. Foremost, this definition of $x$ is tied to some $1-2-3$ coordinate system. Suppose we introduced a different coordinate system $a-b-c$, with

$$x'(t) = [a(t), b(t), c(t)]^\top .$$

From our definition of $x$, how would one deduce or enforce that $x(t) = x'(t)$? One might be tempted to assume that there exists a rotation matrix $R$ such that $x = Rx'$, but this is only the case for certain orthonormal coordinate systems. As non-Euclidean coordinate systems will appear throughout this dissertation, it is necessary to consider vectors in a more abstract sense.

Instead we will define $\boldsymbol{x}$ simply as a vector that represents the position of an object relative to a reference location. We do so without assuming a coordinate system, and assert that multiple coordinate systems should have equivalent descriptions of the same vector, as they should be able to equivalently describe the same object.

Consider for example the common approach of expressing a vector $\boldsymbol{x}$ by scalar coefficients $x_1$ through $x_3$ multiplied by vectors $\boldsymbol{e}_1$ through $\boldsymbol{e}_3$:

$$\boldsymbol{x} = x_1\boldsymbol{e}_1 + x_2\boldsymbol{e}_2 + x_3\boldsymbol{e}_3. \tag{1.1}$$

This is the definition of a particular coordinate system: it defines a way in which to represent a vector $\boldsymbol{x}$ in terms of scalar coefficients $x_1$ through $x_3$. The vectors $\boldsymbol{e}_1$ through $\boldsymbol{e}_2$ are referred to as basis vectors, and are part of the definition of the coordinate system.

The form of coordinate system in equation (1.1) is so common that it is widespread practice to create an ordered set out of the scalar coefficients and refer to that as a vector, such as:

$$\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^\top.$$

Doing so is perilous. For example, how does the ordered set change with respect to time or with respect to a different coordinate system, and how do these change if we originally used a different coordinate system to describe position? Special rules are needed to use vectors defined as ordered-sets in a manner consistent across all coordinate systems, and these are well-known for Euclidean coordinate systems [13].

Extending these rules to non-Euclidean settings with the ordered-set approach is not straightforward. Instead, $\boldsymbol{x}$ and other vectors will be defined first in the coordinate-free sense. In other words, they are assumed to exist just like the vehicles being described, and coordinate systems will be imposed upon them subsequently. Consistency between coordinate systems will appear from equating $\boldsymbol{x}$ expressed in one frame to that in another. This replicates standard Euclidean results while allowing us to derive non-Euclidean results in a straightforwards and logical manner. Suppose for instance we have two coordinate systems $a$ and $b$ which describe $\boldsymbol{x}$ as:

$$\boldsymbol{x} = x_1^a \boldsymbol{e}_1^a + x_2^a \boldsymbol{e}_2^a + x_3^a \boldsymbol{e}_3^a \tag{1.2a}$$

$$\boldsymbol{x} = x_1^b \boldsymbol{e}_1^b + x_2^b \boldsymbol{e}_2^b + x_3^b \boldsymbol{e}_3^b. \tag{1.2b}$$

Here we have added superscripts to the coordinates and vectors to distinguish their frame of reference. Suppose we wish to relate one set of coordinates to the other. This can be done by starting with the obvious relation $\boldsymbol{x} = \boldsymbol{x}$, and expanding each side in a different coordinate system:

$$x_1^a \boldsymbol{e}_1^a + x_2^a \boldsymbol{e}_2^a + x_3^a \boldsymbol{e}_3^a = x_1^b \boldsymbol{e}_1^b + x_2^b \boldsymbol{e}_2^b + x_3^b \boldsymbol{e}_3^b. \tag{1.3}$$

For this to be useful we need to reduce this to numerical operations. Vectors $\boldsymbol{x}$ and basis vectors of both coordinate systems are not themselves numeric. For example, try to express $\boldsymbol{x}$ numerically on a computer without resorting to an ordered-set form.

We can however turn the above expression into a set of scalar expressions by taking its inner product with respect to another vector, for instance, the inner product of (1.3) with respect to $\boldsymbol{e}_1^a$ is:

$$x_1^a \boldsymbol{e}_1^a \cdot \boldsymbol{e}_1^a + x_2^a \boldsymbol{e}_2^a \cdot \boldsymbol{e}_1^a + x_3^a \boldsymbol{e}_3^a \cdot \boldsymbol{e}_1^a = x_1^b \boldsymbol{e}_1^b \cdot \boldsymbol{e}_1^a + x_2^b \boldsymbol{e}_2^b \cdot \boldsymbol{e}_1^a + x_3^b \boldsymbol{e}_3^b \cdot \boldsymbol{e}_1^a.$$

Doing so has made important progress: The inner products between vectors are scalar. Moreover, if both coordinate systems are known, these inner products between basis vectors

are known as well, and the above equation consists of repeated scalar multiplication and addition we can evaluate by hand or on a computer. Taking inner products of (1.3) with respect to all basis vectors of the $a$ coordinate system, we obtain the relationship:

$$\begin{bmatrix} \boldsymbol{e}_1^a \cdot \boldsymbol{e}_1^a & \boldsymbol{e}_2^a \cdot \boldsymbol{e}_1^a & \boldsymbol{e}_3^a \cdot \boldsymbol{e}_1^a \\ \boldsymbol{e}_1^a \cdot \boldsymbol{e}_2^a & \boldsymbol{e}_2^a \cdot \boldsymbol{e}_2^a & \boldsymbol{e}_3^a \cdot \boldsymbol{e}_2^a \\ \boldsymbol{e}_1^a \cdot \boldsymbol{e}_3^a & \boldsymbol{e}_2^a \cdot \boldsymbol{e}_3^a & \boldsymbol{e}_3^a \cdot \boldsymbol{e}_3^a \end{bmatrix} \begin{bmatrix} x_1^a \\ x_2^a \\ x_3^a \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}_1^b \cdot \boldsymbol{e}_1^a & \boldsymbol{e}_2^b \cdot \boldsymbol{e}_1^a & \boldsymbol{e}_3^b \cdot \boldsymbol{e}_1^a \\ \boldsymbol{e}_1^b \cdot \boldsymbol{e}_2^a & \boldsymbol{e}_2^b \cdot \boldsymbol{e}_2^a & \boldsymbol{e}_3^b \cdot \boldsymbol{e}_2^a \\ \boldsymbol{e}_1^b \cdot \boldsymbol{e}_3^a & \boldsymbol{e}_2^b \cdot \boldsymbol{e}_3^a & \boldsymbol{e}_3^b \cdot \boldsymbol{e}_3^a \end{bmatrix} \begin{bmatrix} x_1^b \\ x_2^b \\ x_3^b \end{bmatrix}. \tag{1.4}$$

Here we have organized the result in the usual ordered-set form of a vector and a matrix, with the standard matrix-vector product. Coordinate transformations of vectors between two coordinate systems of the form (1.1) all follow this scheme. In the special case of orthonormal Euclidean coordinates the left matrix is an identity matrix and the right is a rotation matrix [13].

This manner of thinking about vectors will be used to work with non-Euclidean coordinate systems, which are coordinate systems where the linear form (1.1) does not suffice to describe a vector. For instance, we may have no more information about a coordinate system than some nonlinear vector-valued function $\boldsymbol{x}^p$. Defining and viewing vectors in a general coordinate-free sense allows us to approach this problem directly. Necessary consistency between frames of reference will be emergent from derivations similar to the one above for coordinate transformation, which will be extended to non-Euclidean coordinate systems. This proof strategy will appear repeatedly in Chapter 2, with notation introduced next.

## 1.3 General Notation

This dissertation involves theory from mechanics, vehicle dynamics, and differential geometry, fields which are at times notationally inconsistent within themselves, much less between one another. Some inconsistencies will appear here as well to maintain similarity to literature but are made clear by usage. For instance superscript $^g$ will denote a global, inertial frame, $\boldsymbol{g}$ the gravity vector, and $g(z, u, a)$ the algebraic constraint of a differential algebraic equation (DAE).

### 1.3.1 Vectors and Matrices

We will refer to vectors in the coordinate-free sense, that is $\boldsymbol{v}$ is a vector whereas $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ is an ordered set, not a vector. We will not refer to the latter as a "coordinated vector" or a "vector expressed in a frame of reference". Instead, we will either expand vectors in a frame of reference such as $\boldsymbol{v}^b = v_1^b \boldsymbol{e}_1^b + v_2^b \boldsymbol{e}_2^b + v_3^b \boldsymbol{e}_3^b$, or explicitly write out an ordered set, such as $\begin{bmatrix} v_1^b & v_2^b & v_3^b \end{bmatrix}$. Here the superscript on $\boldsymbol{v}^b$ is shorthand to illustrate that the vector $\boldsymbol{v}$ has been expanded in the basis $\boldsymbol{e}_{1,2,3}^b$. It does not change $\boldsymbol{v}$ whereas the ordered set does. This is most evident in the time derivative of $\boldsymbol{v}$. We use $\dot{}$ to denote scalar derivatives whereas $\partial_t$

denotes vector time derivatives. With this convention and the chain rule we have:

$$\dot{\boldsymbol{v}}^b = \sum_{k\in\{1,2,3\}} \dot{v}_k^b \boldsymbol{e}_k^b \tag{1.5}$$

$$\partial_t \boldsymbol{v} = \partial_t \boldsymbol{v}^b = \sum_{k\in\{1,2,3\}} \dot{v}_k^b \boldsymbol{e}_k^b + v_k^b \partial_t \boldsymbol{e}_k^b. \tag{1.6}$$

With this format, proper rules of differentiation are automatic, whereas special considerations and care are needed for differentiation of the ordered set form, eluded to in the previous section. Numerous examples of this notation will appear later in this dissertation, as most differential geometry derivations will be carried out by equating two vectors for all time, and then differentiating with respect to time to obtain new equalities. This is consistent with first order tensor calculus, but with notation more similar to standard vehicle dynamics.

Several notational assumptions are made: All vector time derivatives are with respect to an inertial frame of reference, which will be defined in Section 1.3.2. The distinction of what $\boldsymbol{v}$ and the $^b$ frame are will be determined by a given vehicle model. As eluded to in equation (1.5), we will use superscripts to indicate a frame of reference, ie. $\boldsymbol{v}^b$ will be velocity expressed in the $^b$ frame, often the body-fixed frame of a vehicle. We will use letter subscripts for partial derivatives, such as $\partial_s \boldsymbol{x}^p(s,y) = \boldsymbol{x}_s^p$, with numeric subscripts reserved for basis elements of a frame of reference. Boldface $\boldsymbol{e}$ is reserved for a unit vector which is part of an orthogonal basis.

As a result of this syntax, expressions involving vectors may appear scalar in nature. This should not be misinterpreted as the expressions themselves being scalar. Instead, vector expressions will be reduced to scalar ones by use of inner products with a basis, ie. $\boldsymbol{a} = \boldsymbol{b}$ yields three independent equations by taking inner products with respect to the basis $\boldsymbol{e}_{1,2,3}^b$.

We will not apply the same treatment to matrices, for instance when dealing with rotation matrices we will always consider their ordered set form and the standard matrix-matrix and matrix-vector products.

### 1.3.2 Position and Orientation

It is standard practice to represent the position and orientation of a rigid body in terms of:

1. A reference location on the body.
2. A triad of orthogonal unit vectors, assumed to be fixed with respect to the body.

We will refer to the combination of the two as a frame or frame of reference, with this one known as the body frame. We can define the body frame however so we please, but it is itself fictional. Position and orientation are always relative, for which we require a second frame. For this purpose we introduce an inertial frame of reference, also referred to as the global frame.

An inertial frame is defined simply as one for which Newton's laws of motion hold. In practice, this is as imprecise as the notion of a rigid body, and may take many different forms. For example, the walls of an enclosed room may provide a coordinate system that is inertial to a sufficient degree. At larger scales however, even the surface of the Earth is non-inertial due to its spin. Further still, the solar system itself is not inertial but rotates within The Milky Way. The choice of an inertial frame is application-dependent, one we assume has been made beforehand.

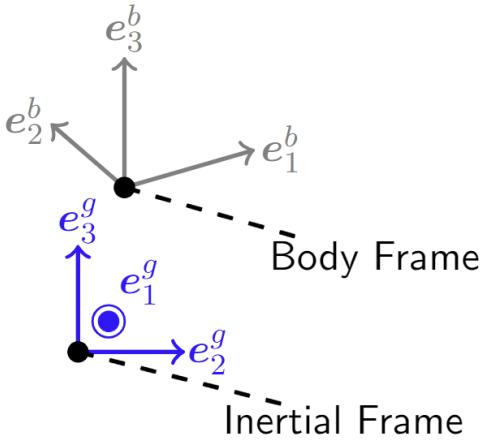



Figure 1.3: Schematic of inertial and body frames

With this pair of frames, we can introduce notions of position and orientation. Firstly, the position of the body frame relative to the inertial frame $\boldsymbol{x}$, is simply a vector from a fixed point in the inertial frame to the fixed origin of our body frame. Describing orientation is not as simple position, as we must relate $\boldsymbol{e}^b_{1,2,3}$ to $\boldsymbol{e}^g_{1,2,3}$. This can be done via a linear relationship in the form of a matrix $\mathbf{R}^{gb} \in \mathbb{R}^{3\times3}$:

$$\begin{bmatrix}\boldsymbol{e}_1^b & \boldsymbol{e}_2^b & \boldsymbol{e}_3^b\end{bmatrix} = \begin{bmatrix}\boldsymbol{e}_1^g & \boldsymbol{e}_2^g & \boldsymbol{e}_3^g\end{bmatrix} \mathbf{R}^{gb}. \tag{1.7}$$

It follows that $\left(\mathbf{R}^{gb}\right)_{ij} = \boldsymbol{e}_i^g \cdot \boldsymbol{e}_j^b$ and that is is a rotation matrix, ie. an element of the special orthogonal group in three dimensions (SO3) [13].

### 1.3.3 Velocity

Velocity exists in a purely geometric sense as as a means to describe how position and orientation may change. The time derivative of a vector is itself another vector hence the linear velocity vector $\boldsymbol{v}$. For orientation, since $\boldsymbol{e}^b_{1,2,3}$ are orthonormal there are limitations as to how they may change over time. These limitations are straightforwards to derive, and we do so to illustrate how vector calculus will be handled here.

Firstly, any change in a vector can be expressed as a new vector, and may itself be expressed in a basis, ie. we can write out:

$$\partial_t \boldsymbol{e}_1^b = \alpha_{11}\boldsymbol{e}_1^b + \alpha_{12}\boldsymbol{e}_2^b + \alpha_{13}\boldsymbol{e}_3^b \tag{1.8a}$$
$$\partial_t \boldsymbol{e}_2^b = \alpha_{21}\boldsymbol{e}_2^b + \alpha_{22}\boldsymbol{e}_2^b + \alpha_{23}\boldsymbol{e}_3^b \tag{1.8b}$$
$$\partial_t \boldsymbol{e}_3^b = \alpha_{31}\boldsymbol{e}_3^b + \alpha_{32}\boldsymbol{e}_2^b + \alpha_{33}\boldsymbol{e}_3^b, \tag{1.8c}$$

where we have simply chosen to write out the derivatives in the frame $\boldsymbol{e}^b_{1,2,3}$. For these vectors to remain orthonormal, we must have $\partial_t \left(\boldsymbol{e}_i^b \cdot \boldsymbol{e}_j^b\right) = 0 \ \forall \ i,j$. Expanding this in the

case where $i = j$ this tells us that $\alpha_{ii} = 0$, and in the case where $i \neq j$ tells us $\alpha_{ij} = -\alpha_{ji}$. This leaves the coefficients $\alpha_{ij}$ with three degrees of freedom, which we can summarize as:

$$\begin{bmatrix} \partial_t \boldsymbol{e}_1^b & \partial_t \boldsymbol{e}_2^b & \partial_t \boldsymbol{e}_3^b \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}_1^b & \boldsymbol{e}_2^b & \boldsymbol{e}_3^b \end{bmatrix} \hat{\boldsymbol{\omega}}^b \qquad \hat{\boldsymbol{\omega}}^b = \begin{bmatrix} 0 & -\omega_3^b & \omega_2^b \\ \omega_3^b & 0 & -\omega_1^b \\ -\omega_2^b & \omega_1^b & 0 \end{bmatrix}. \tag{1.9}$$

In this manner we've derived a standard result of SO3, that its ability to change is restricted to this matrix form. We will not show it, but the result $\omega_1^b \boldsymbol{e}_1^b + \omega_2^b \boldsymbol{e}_2^b + \omega_3^b \boldsymbol{e}_3^b = \boldsymbol{\omega}$ is a vector, just like linear velocity $\boldsymbol{v}$. This vector $\boldsymbol{\omega}$ is referred to as angular velocity, and is of the body frame relative to the inertial frame, as we have taken time derivatives with respect to the inertial frame.

## 1.4 Motion of a Newtonian Rigid Body

### 1.4.1 Kinematics

Having introduced both pose (position and orientation) and velocity, we can relate one to the other. For position we have $\partial_t \boldsymbol{x} = \boldsymbol{v}$ while for orientation we have Equation (1.9). These vector equalities may be used in any basis, however often times $\boldsymbol{x}$ is expressed in the inertial frame and $\boldsymbol{v}$ in the body frame. Furthermore, (1.9) links to $\mathbf{R}^{gb}$ by taking inner products of (1.9) with respect to $\boldsymbol{e}_{1,2,3}^g$. The standard result is:

$$\begin{bmatrix} \dot{x}_1^g \\ \dot{x}_2^g \\ \dot{x}_3^g \end{bmatrix} = \mathbf{R}^{gb} \begin{bmatrix} v_1^b \\ v_2^b \\ v_3^b \end{bmatrix} \tag{1.10a}$$

$$\partial_t \mathbf{R}^{gb} = \mathbf{R}^{gb} \hat{\boldsymbol{\omega}}^b. \tag{1.10b}$$

This is a well-known result of rigid body kinematics[1]: three equations for position, nine for orientation, and with linear and angular velocity expressed in the body frame.

### 1.4.2 Newtonian Mechanics

Newtonian mechanics are the classical form of dynamics, and will be the only form considered in this dissertation. We will use Newton's second law, which in its most fundamental form relates linear $\boldsymbol{p}$ and angular $\boldsymbol{l}$ momentum to force $\boldsymbol{F}$ and moment $\boldsymbol{K}$, all vector quantities.

$$\partial_t \boldsymbol{p} = \boldsymbol{F} \qquad \partial_t \boldsymbol{l} = \boldsymbol{K}. \tag{1.11}$$

[1]There are actually two results, depending on if $\mathbf{R}^{gb}$ or $\left(\mathbf{R}^{gb}\right)^T$ is used. This second case is solely an inversion of the matrix, and (1.10b) is then usually transposed with the sign of $\hat{\boldsymbol{\omega}}^b$ flipped to obtain the form $\partial_t \mathbf{R}^{gb} = \hat{\boldsymbol{\omega}}^b \mathbf{R}^{gb}$. This is purely a difference of convention.

Force and moment here are unknown quantities to be determined by a particular vehicle model. $\boldsymbol{p}$ and $\boldsymbol{l}$ are simplest to express in terms of the linear and angular velocity of the center of mass, combined with the mass $m$ and moment of inertia $I_{ij}^{bb}$ of a rigid body. In nearly all cases, the center of mass will be the origin of the body frame with which we have:

$$\boldsymbol{p} = m\boldsymbol{v}^b \qquad\qquad \boldsymbol{l} = \sum_{i,j\in\{1,2,3\}} \boldsymbol{e}_i^b I_{ij}^{bb} \omega_j^b. \tag{1.12}$$

Examples where this is not the case will be treated on a case-by-case basis. However, a common example are bodies with internally rotating parts, such as tires, engines and propellers. The effects of these may be approximated by adding terms to $\boldsymbol{l}$ that capture these effects, then applying (1.11) to the modified equations.

In the case where (1.12) holds true, (1.11) leads to the well-known Newton-Euler equations. For many vehicles, the body frame is chosen such that the moment of inertia matrix is diagonal, with which these simplify to:

$$\dot{v}_1^b + v_3^b\omega_2^b - v_2^b\omega_3^b = \frac{1}{m}F_1^b \tag{1.13a}$$

$$\dot{v}_2^b + v_1^b\omega_3^b - v_3^b\omega_1^b = \frac{1}{m}F_2^b \tag{1.13b}$$

$$\dot{v}_3^b + v_2^b\omega_1^b - v_1^b\omega_2^b = \frac{1}{m}F_3^b \tag{1.13c}$$

$$I_1^{bb}\dot{\omega}_1^b + \left(I_3^{bb} - I_2^{bb}\right)\omega_2^b\omega_3^b = K_1^b \tag{1.13d}$$

$$I_2^{bb}\dot{\omega}_2^b + \left(I_1^{bb} - I_3^{bb}\right)\omega_3^b\omega_1^b = K_2^b \tag{1.13e}$$

$$I_3^{bb}\dot{\omega}_3^b + \left(I_2^{bb} - I_1^{bb}\right)\omega_1^b\omega_2^b = K_3^b. \tag{1.13f}$$

Here we've denoted $I_{ii}^{bb}$ as $I_i^{bb}$ since there is no ambiguity in the diagonal case. The last three expressions are slightly different when the moment of inertia matrix is no longer diagonal, in ways which follow from (1.11).

For a single rigid body, Equation (1.13) is always present, independent of kinematic assumptions. For example, it is common practice to model a car as a rigid body stuck to a flat surface, in which case it is assumed that $v_3^b = \omega_1^b = \omega_2^b = 0$. This turns (1.13c), (1.13d), and (1.13e) into constraints instead of differential equations. Ignoring these constraints amounts to ignoring the operating limits of a vehicle, as they determine the weight distribution of the car. Ignoring these effects may result in loss of contact with the road, such as rolling over, pitching forwards or backwards, or going airborne, all of which invalidate the tangent contact motion model.

## 1.5 Example Vehicles

This section provides examples of several vehicles and how they may be modeled. These examples are cursory in nature and serve to highlight the general components of each model.

This is used to highlight the motion models that factor into each, and how the resulting vehicle models are altered when non-Euclidean motion models are used, for instance the weight distribution of a car. Many of the considerations of this section serve only to motivate how the motion models described in the previous section may be used to build models of real-world vehicles, and may not be necessary for the reader.

### 1.5.1 Point Mass

In some applications it is sufficient to treat vehicles as a point mass by ignoring rotational effects and focusing solely on translational dynamics. This may be useful in settings where the translational dynamics are the most important. For instance, the friction limits that govern how fast a car may travel around an icy turn may be studied using a point mass model provided rotational effects such as the vehicle rolling over do not appear first. Similarly with quadrotor racing, net thrust limits play the dominant role in how fast a vehicle can maneuver through a racetrack. With a full quadrotor model, the vehicle must rotate to apply a thrust in a different direction, introducing complex rotational dynamics. However, thrust magnitude limits may be enforced on a point mass model, with comparable results provided the rotational inertia of the quadrotor is small enough to track a trajectory computed with a point mass model. All of these applications are simplifications of the original behaviour of the vehicle, but are commonly done to obtain simplified models suitable for fast, real-time planning that retains most vehicle operation limits.

A point inherently has no notion of orientation, so equipping it with a body frame of reference is a somewhat arbitrary step. For instance, a point mass model on a flat road might be set up with its velocity described in an inertial frame, or in a frame of reference which follows the road. The former has simpler equations of motions, while the latter is more representative of how a car would move. Neither can be considered better than the other in general.

#### 1.5.1.1 Tangent Contact Point Mass

When constrained to remain in contact to a surface, several phenomena emerge. Most importantly, the force component normal to the surface is constrained by kinematic assumptions in a manner that emerges from equation (1.13c). Several forces may take part in the constrained $F_3^b$, such as gravity, aerodynamics, and a normal force with the surface. The resulting effect on normal force is often the most important, and expressions can be obtained to determine normal force as a function of external forces such as gravity and Coriolis forces from motion over a curved surface.

As an example of the implications of this, consider a car: The net normal force on the car immediately impacts the maximum longitudinal and lateral tire forces that may be obtained in practice. Exceeding these limits in practice results in loss of control, such as colliding

with an obstacle or skidding out during a turn. These risks must be detected quickly, which motivates the use of simplified models such as a point mass.

A point mass model might capture these friction limits by:

1. Introducing longitudinal and lateral forces on a point mass that are treated as inputs.
2. Modeling the net normal force on the point mass from its motion constraints.
3. Developing a motion planning algorithm that limits input forces depending on net normal force.

This is a logical approach but cannot be developed without a suitable motion model.

#### 1.5.1.2 Unconstrained Point Mass

Unconstrained point mass models take several forms. For instance, we might build a point mass model on a parametric surface with a spring law relating to its normal distance from the surface. The previous discussion would hold as a special case when on the surface, but the resulting model would be allowed to go airborne. This may be a useful model to develop for modeling a sled or inner tube on a well-packed and snowy surface.

A second paradigm emerges for aircraft, wherein a surface-relative motion model is only useful during take-off, landing, and potentially during very low altitude maneuvers. This motivates the curve-relative motion model of the previous section, which solely changes the coordinates used to describe where a vehicle is. When using a point mass, the only ambiguity left here is how to artificially describe its orientation, necessary to introduce input quantities for the forces acting upon it. This is purely a matter of user choice: inputs may be chosen in an inertial frame, or one that is relative to a non-Euclidean coordinate system induced by the curve.

### 1.5.2 Cars

For the purposes of control, cars are almost universally described with models that kinematically constrain them to a surface. Within this realm however, a plethora of options for modeling them has emerged. Three are introduced here which serve to best highlight the motion changes imparted upon the vehicle by non-Euclidean geometry, particularly weight distribution.

Note that in most literature on cars, trucks, and other ground vehicles, they are simply referred to as "vehicles". This terminology is ambiguous when other vehicles are or may be considered, such as airborne vehicles. In this dissertation they will instead be referred to as "car" models or "two track models", with the implication that they apply broadly to many four-wheeled ground vehicles encountered in practical experience.

Several geometric terms are necessary to describe the layout of the car, and are illustrated in Figure 1.4. We use $l_f$ and $l_r$ for the distance to the front and rear axle from the center of

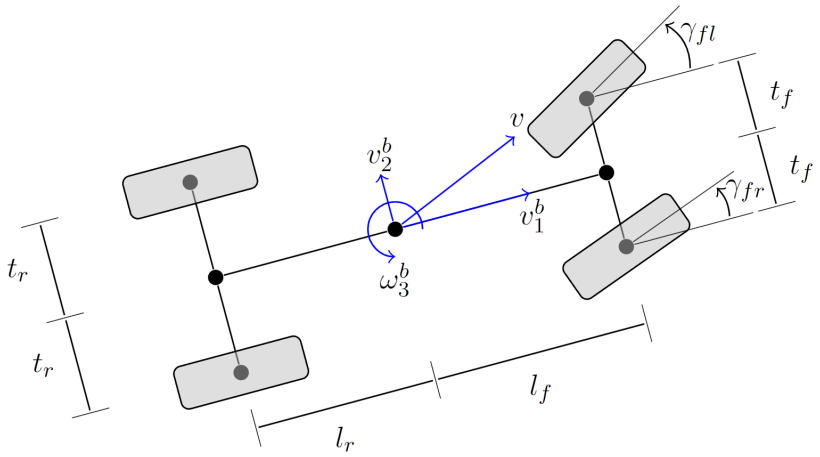



Figure 1.4: Two track layout of a car. The front of the car is to the right in the diagram, with the $1, 2, 3$ directions of the body frame corresponding to the front, left, and up directions from the perspective of a driver (ISO convention). The origin of the body frame is placed at the center of mass, which at a distance $h$ out of the page.

mass, which is used as the origin of the body frame. $t_f$ and $t_r$ denote half the width of the front and rear axles, respectively.

#### 1.5.2.1 Kinematic Two Track Model

A kinematic model of a car is unique in that is does away with any Newtonian dynamics. This is not to say these physics are absent, but that they do not enter through differential equations - they can only be added separately such as by imposing constraints on a control problem. This discrepancy comes with the advantage that they are by far the simplest models of cars which consider the presence of steering and rotational motion of the car, and always use the tangent contact motion model.

In most literature kinematic models are referred to as a kinematic bicycle model, with the assumption that the front pair of wheels, and rear pair of wheels, are each merged into single fictitious wheels. This is not the case in this dissertation: we will use the same equations of motion, but all four wheels will always be considered explicitly. As suggested by the previous paragraph, the presence of four instead of two wheels will have no effect on the equations of motion that develop. However, we will always consider four wheels to model weight distribution effects, which are necessary to consider on general road geometry, and largely ignored by a bicycle model.

A tangent contact motion model removes terms from the standard Newton Euler equations, leaving us with variables $v_1^b$, $v_2^b$, and $\omega_3^b$ as independent variables which change over time. In kinematic models, these are eliminated by the assumption that the velocity of each

tire on the car is parallel to the direction it is facing. This imposes constraints on how each tire may be orientated, but more importantly it constrains these three velocity components. In the most common setting, it is assumed that the front steering linkage is steered with angle $\gamma^f$ such that for a vehicle moving with signed speed $v$ we have:

$$\beta = \arctan\left(\frac{l_r}{l_r + l_f}\tan\gamma^f\right) \tag{1.14a}$$

$$v_1^b = v\cos\beta \tag{1.14b}$$

$$v_2^b = v\sin\beta \tag{1.14c}$$

$$\omega_3^b = v\frac{\cos\beta}{(l_r + l_f)\tan\gamma^f}. \tag{1.14d}$$

This allows us to replace the usual dynamic equations for a car with $v$ and $\gamma^f$, which may be treated as inputs, or given simplified equations of motion. For instance, $\dot{v}$ may be treated as the sum of a throttle input and the component of gravity opposing vehicle motion. Equations with the added degree of freedom to steer the rear wheels may be found in [3], but the form is similar: a new input is added for the rear axle steering, and the variables $v_1^b$, $v_2^b$, and $\omega_3^b$ are determined by signed speed and steering inputs.

As was the case with point mass models, capturing realistic friction limits is not inherent in the equations of motion of a kinematic model. Instead, these effects must be enforced separately, such as with additional constraints on a planning problem. The most straightforwards manner in which to do this is model the tire forces required to satisfy kinematic motion, and then constrain these forces.

For instance, the speed $v$ in a kinematic vehicle will change due to a sum of longitudinal tire forces on a car, aerodynamic drag, and gravity. The required longitudinal tire force can thus be calculated for a known $\dot{v}$.

Lateral tire forces are less straightforwards to model, but a simple and fairly realistic assumption is to model the presence of lateral tire forces to oppose two effects: circular motion of the vehicle, and the lateral component of gravity. Lateral tire forces are needed for both, the first may be approximated by simple circular motion equations, such as:

$$\frac{v^2\gamma^f}{l_r + l_f}, \tag{1.15}$$

while the second is known from the orientation of a vehicle.

Constraining both the longitudinal and lateral tire forces required by a kinematic model to a friction cone (ie. a circle with diameter dependent on the net normal force on the car) is a straightforwards and practical way to ensure that motions planned with a kinematic model are realistic in practice. Determining net normal force on a general 3D surface is discussed in Section 2.4.3, with introductory, general weight distribution considerations provided in Section 1.5.2.3.

#### 1.5.2.2 Dynamic Two Track Model

As opposed to kinematic car models, dynamic models directly consider the forces acting upon the vehicle. In particular, it is common practice to include:

1. Gravity forces upon a vehicle, the body-frame components of which change on a three-dimensional surface.
2. Aerodynamic forces, such as due to the speed of the vehicle and wind speed.
3. Tire forces, which ultimately govern how the vehicle is driven and steered.

These forces factor directly into dynamics expressed by Equation (1.13). Gravity and aerodynamic effects are fully determined by where the vehicle is and how fast it is moving. Unlike kinematic models, dynamic models can make operating limits of a tire part of the vehicle model itself. For instance, tire models such as the Pacejcka model [14] internally capture saturation of tire forces, and subsequent skidding. However, modeling tire forces requires us to know the weight distribution of the vehicle, discussed next.

#### 1.5.2.3 Weight Distribution of Tangent Car Models

For a body in tangent contact with a surface, the $F_3^b$, $K_1^b$, and $K_2^b$ equations from (1.13) turn into constraints on said forces and moments. This is meaningful because they impact the weight distribution of a car, which itself determines important physical effects such as rolling over or going airborne. While these effects are kinematically impossible for a tangent contact motion model - the motion model has no degrees of freedom to allow this - these operating limits can be modeled and subsequently controlled to avoid loss of contact in practice.

Several numerical approaches for relating the constrained force and moment to tire forces exist. In general the following process follows:

1. $F_3^b$, $K_1^b$, and $K_2^b$ are known.
2. Force and moment due to gravity, aerodynamics, non-normal tire forces are removed.
3. The resulting force and moment must be provided by tire normal forces, which are computed by an elasticity method.

We consider the approach developed and validated in [15]. We denote the required forces and moments from normal force effects as $F_{3,\text{req}}^b$ and so forth. The resulting normal forces

are then determined as:

$$N_f = \frac{l_r}{l_f + l_r} F^b_{3,\text{req}} - \frac{1}{l_f + l_r} K^b_{2,\text{req}} \tag{1.16a}$$

$$N_r = \frac{l_f}{l_f + l_r} F^b_{3,\text{req}} + \frac{1}{l_f + l_r} K^b_{2,\text{req}} \tag{1.16b}$$

$$\Delta = \frac{K^b_{1,\text{req}}}{2\left(t_r^2 + t_f^2\right)} \tag{1.16c}$$

$$N_{fr} = \frac{1}{2} N_f - t_f \Delta \tag{1.16d}$$

$$N_{fl} = \frac{1}{2} N_f + t_f \Delta \tag{1.16e}$$

$$N_{rr} = \frac{1}{2} N_r - t_r \Delta \tag{1.16f}$$

$$N_{rl} = \frac{1}{2} N_r + t_r \Delta. \tag{1.16g}$$

Overall, this relates the net force and moment on the vehicle to the normal force ($N$) on the front right (fr), front left (fl), rear right (rr), and rear left (rl) tires on the car.

There exists an algebraic loop in the sence that the four normal forces impact the longitudinal and lateral tire forces. These forces in turn impart a moment on the vehicle, changing the 1 and 2 moment that should be provided by normal forces. Three approaches are common to resolving this:

In one approach, the above expressions are used to compute equilibrium normal forces, which are passed through a first order filter for the normal force used to compute other tire forces. This is an attractive option as the overall equations of motion can remain ordinary differential equations.

In a second approach, the link between normal forces and tire forces can be encoded in the form of a differential algebraic equation. This takes the general form:

$$\dot{z} = f(z, u, a) \tag{1.17a}$$

$$0 = g(z, u, a), \tag{1.17b}$$

where $z$ contains the differential states of the vehicle (such as position and speed), $a$ the algebraic states (tire normal forces), and $u$ the inputs to the vehicle. Kinematics and dynamics are part of $f$ while $g$ models quasi-equilibrium weight distribution in the form of a constraint.

Finally, the third approach is to simply ignore this link. This makes no sense for a dynamic model, but is the go to approach for a kinematic model. In kinematic models, longitudinal and lateral tire forces are often approximated from inputs to the model, so there is no algebraic loop. Weight distribution can be computed directly from states and inputs of a kinematic vehicle model.

#### 1.5.2.4 Dynamic Model With Suspension

Although the previous cases can consider suspensions, their tangent contact motion model assumptions do not allow for actual suspension motion. This can only be changed by making different kinematic assumptions. Particularly, when assuming that a car is described relative to a surface, but not constrained by it, it is possible to introduce expressions for the displacement of suspension linkages and directly compute tire normal forces. This is the purpose of the unconstrained surface motion model from the start of this chapter.

Allowing for suspension motion avoids any algebraic loops as suspension forces are a function of current vehicle pose and velocity. Furthermore, the full Newton-Euler equations are used, making this a seemingly simple model to setup. Complexity enters in large part by the need to describe the vehicle relative to a surface, particularly when the surface is not flat. Terms such as the displacement and displacement rate of suspension link can be computed geometrically, but any resulting model has more state variables to describe vehicle pose and velocity.

It is immediate with the tangent contact motion model that a car is assumed to live within the tangent plane of the surface it is driving on. In the case of an unconstrained vehicle with suspension motion, a similar assumption may enter in how the displacement and displacement rate of suspension linkages are approximated, as on a curved surface displacement into the tangent plane of the surface will differ from displacement into the real physical surface.

### 1.5.3 Motorcycles

Motorcycles are fundamentally different from cars in that their ordinary motion, even without a driver, involves a clear additional degree of freedom - the ability for the motorcycle to camber. This degree of freedom immediately rules out any description of the motorcycle body as one that remains tangent to in and in contact with the surface being driven on.

However, we can develop a modified view of a motorcycle by introducing a frame of reference which does remain tangent to the surface being driven on. This is illustrated in Figure 1.5, where a motorcycle body is approximated as one that cambers about a known, fixed axis, which is located at a fixed normal offset from the road surface. This allows us to introduce a body frame $\boldsymbol{e}^b_{1,2,3}$ to which a tangent contact motion model may be applied.

However, the fact that the motorcycle is not fixed within this frame complicates application of Newtonian mechanics. This is not unlike describing the dynamics of a robot arm relative to its base, only in this case the base moves around in a manner prescribed by tangent contact. Development of the dynamics of this model is involved and discussed in Section 4.5.2.

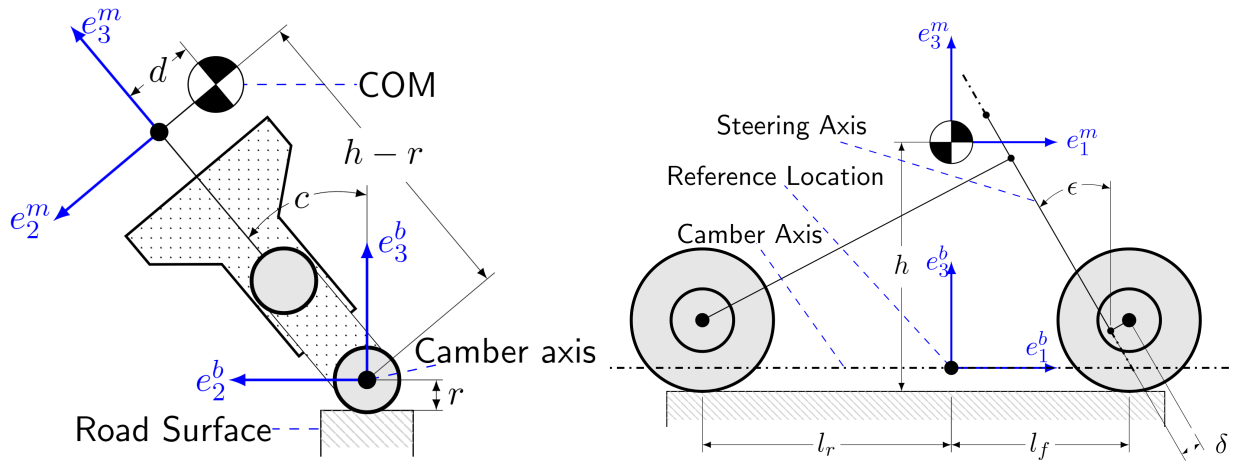



Figure 1.5: Rear (left) and side (right) views of motorcycle diagram. The right view is shown with zero camber angle ($c$). The center of mass (COM) may move due to camber of the motorcycle body, and motion of the driver. Here the latter is approximated with the displacement $d$ of the center of mass. Figures adapted from [16].

### 1.5.4 Quadrotors

Quadrotors are the first and only example of a truly airborne vehicle discussed in detail in this dissertation. Quadrotors are often modeled as a single rigid body. This is of course not true in practice - spinning propellers contribute large angular momentum to the vehicle. However, it is standard practice for half of the propellers to spin one direction, and the other half spin the other. Since often times all propellers spin at similar speeds, the resulting angular momentum contribution sums to zero.

It is common to model a quadrotor with four input thrusts corresponding to the thrust of each propeller, $T_1$ through $T_4$. The main result is to then relate these to the force and moment about the center of mass of the vehicle, often done [17] as:

$$F_1^b = -m\boldsymbol{g} \cdot \boldsymbol{e}_1^b \tag{1.18a}$$
$$F_2^b = -m\boldsymbol{g} \cdot \boldsymbol{e}_2^b \tag{1.18b}$$
$$F_3^b = T_1 + T_2 + T_3 + T_4 - m\boldsymbol{g} \cdot \boldsymbol{e}_3^b \tag{1.18c}$$
$$K_1^b = (T_1 + T_2 - T_3 - T_4)\, l \tag{1.18d}$$
$$K_2^b = (-T_1 + T_2 + T_3 - T_4)\, l \tag{1.18e}$$
$$K_3^b = (T_1 - T_2 + T_3 - T_4)\, c_\tau. \tag{1.18f}$$

Here $l$ is a geometric factor for the size of the quadrotor and $c_\tau$ is a propeller drag torque factor. For motion planning, one can consider treating thrusts as an input, and bounding them to an interval $\left[T_{min}, \frac{1}{4}\text{TWR}\right]$ with $T_{min}$ a lower bound on rotor thrust due to idle speeds

and TWR the thrust-to-weight ratio of the quadrotor, both model parameters. Net force and moment bounds are emergent from (1.18).

### 1.5.5 Other Possible Vehicles

Numerous other vehicles may be described by motion models introduced in this chapter, the full scope of which is not possible to include here. A few examples which will not be studied in any detail later are remarked upon here.

Many sliding objects may be approximated as tangent to and in contact with a surface. A few examples such as sledding and water-sliding have been remarked on already. Further examples exist of sliding objects constrained to one dimension, such as trains, roller-coasters, and sliding objects in certain industrial processes. These may be thought of as one-dimensional cases of the tangent-contact motion model, that is the heading and lateral coordinate of a train are fixed, and only the longitudinal axis has any freedom to move.

The same process which will be used to study motorcycles in Section 4.5.2 can be extended to handle spherical robots as well, which add a second degree of freedom for pitch motion.

Many aircraft can be studied with a single rigid frame of reference like a quadrotor, but require very different force laws to describe their aerodynamics. Furthermore, in the case of helicopters, additional terms are required to capture the momentum contribution of the spinning blades, which factor heavily into their dynamics. To a large extent, these vehicle-specific considerations are independent of the motion models developed for aircraft in this dissertation, as these motion models are practically a change of basis for how position and orientation are described, and naturally work for any airborne object.

## 1.6 Applying Motion Models to Vehicles

Understanding this dissertation and how to use it requires the reader to think differently about how vehicle models are structured and developed compared to usual vehicle models. This is because non-Euclidean geometry may be chosen independently of a vehicle, for instance one might drive the same car over multiple different road surfaces. Complete vehicle models of the same physical vehicle applied to different geometries should reflect that the same vehicle is being modeled. At the same time, it will be inconvenient to derive new vehicle models each time its operating domain changes. Addressing this requires a different perspective that accounts for both the vehicle and geometry separately, and is introduced in this section.

### 1.6.1 The Usual Approach

A reader familiar with vehicle dynamics literature likely expects at some point in this dissertation to find a neat set of equations that correspond to a complete model of a specific

vehicle. For instance, in [2], one of the key results is the dynamic bicycle model transcribed below:

$$\ddot{x} = \dot{\psi}\dot{y} + a_x \tag{1.19a}$$

$$\ddot{y} = -\dot{\psi}\dot{X} + \frac{2}{m}\left(F_{c,f}\cos\delta_f + F_{c,r}\right) \tag{1.19b}$$

$$\ddot{\psi} = \frac{2}{I_z}\left(l_f F_{c,f} - l_r F_{c,r}\right) \tag{1.19c}$$

$$\dot{X} = \dot{x}\cos\psi - \dot{y}\sin\psi \tag{1.19d}$$

$$\dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi. \tag{1.19e}$$

Such models are practical and user-friendly as they can be copied by the reader to their manuscript or programming language of choice, and then used for research. However, this format is not at all suitable for non-Euclidean motion models for several reasons.

First and foremost will be that in practical application, there are many ways to parameterize a surface, a curve, or other geometric feature. Deriving a complete vehicle model with respect to any single surface, or means of parameterizing a surface, is myopic and of use only to a limited range of users. For instance, off-road ground vehicles may opt to use an elevation map, but on-road ground vehicles may prefer surface parameterizations whose contours follow a lane that is to be followed.

Secondly, taking advantage of the previous point requires providing the user with equations which are general in nature. For instance in the mathematical treatment of motion models on surfaces, presented in Chapter 2, equations are left in a general form which can be filled in for any means of parameterizing a surface. As a result, they may be applied to an elevation map, a three-dimensional lane, and other non-Euclidean road surfaces. Such equations are by their very nature not ones that can be transcribed directly to standard numerical software.

Thirdly, in deriving general motion models, resulting motion and vehicle models will be more complicated than their Euclidean counterparts. As we look at an intuitive and practical example in Chapter 4, it will become rapidly obvious that non-Euclidean equations of motion are not practical to implement by hand, much less present to the reader to transcribe. For example, Figure 1.6 provides a printout of the equations for a four-wheeled dynamic car model on a non-Euclidean surface, which was derived using the software pipeline discussed in Chapter 4.

Finally, by allowing separate choice of non-Euclidean motion model and resulting vehicle model, any enthusiast willing to derive a single vehicle model by hand will quickly realize they cannot keep up with the combinatorial nature of different surfaces, different vehicles, and ultimately different applications.

As it will prove entirely unsustainable to think about models in standard formats such as (1.19), a new manner of thinking is necessary.

```
SX(@1=cos(a), @2=(@1*k), @3=(cos(a)-(y*@2)), @4=sin(a), @5=(@4*k), @6=(sin(a)-(y*@5)), @7=(sq(@3)+
sq(@6)), @8=(((((cos(ths)*sqrt((sq(@3)+sq(@6))))/@7)*v1)-(((sin(ths)*sqrt((sq(@3)+sq(@6))))/@7)*v2)
), @9=((sin(ths)*v1)+(cos(ths)*v2)), @10=1, @11=atan((tan(uy)/((0.192308*tan(uy))+@10))), @12=cos(
@11), @13=21.7654, @14=(@13*Nfr), @15=1.58, @16=0.1, @17=16, @18=-0.1, @19=(v1-(@18*w3)), @20=sin(
@11), @21=0.13, @22=(v2+(@21*w3)), @23=(((@12*@19)+(@20*@22)), @24=0.024, @25=1e-006, @26=((@23-(@2
4*w))/sqrt((sq(@23)+@25))), @27=14.4, @28=-0.5, @29=13, @30=9.7, @31=(@29*cos(atan((-(@30*@26)))))
, @32=atan((((@12*@22)-(@20*@19))/sqrt((sq(@23)+@25)))), @33=(((@14*sin((@15*atan(((@16*atan((-(@17
*@26))))-(@27*@26)))))*cos(atan(((@28*((@31*@32)+atan((-(@31*@32))))))-(@31*@32))))), @34=1.45, @3
5=-0.8, @36=23.4, @37=0.3, @38=10.62, @39=7.82, @40=(@38*cos(atan((-(@39*@32))))), @41=((@14*sin((
@34*atan(((@35*atan((-(@29*@32))))-(@36*@32))))))*cos(atan(((@37*((@40*@26)+atan((-(@40*@26)))))-(
@40*@26))))), @42=((@12*@33)-(@20*@41)), @43=atan((tan(uy)/((-0.192308*tan(uy))+@10))), @44=cos(@4
3), @45=(@13*Nfl), @46=(v1-(@16*w3)), @47=sin(@43), @48=(v2+(@21*w3)), @49=(((@44*@46)+(@47*@48)),
@50=(((@49-(@24*w))/sqrt((sq(@49)+@25))), @51=(@29*cos(atan((-(@30*@50))))), @52=atan((((@44*@48)-(
@47*@46))/sqrt((sq(@49)+@25)))), @53=(((@45*sin((@15*atan(((@16*atan((-(@17*@50))))-(@27*@50))))))*
+atan((-(@31*@32))))-(@31*@32))))), @34=1.45, @35=-0.8, @36=23.4, @37=0.3, @38=10.62, @39=7.82, @
40=(@38*cos(atan((-(@39*@32))))), @41=((@14*sin((@34*atan(((@35*atan((-(@29*@32))))-(@36*@32))))))
*cos(atan(((@37*((@40*@26)+atan((-(@40*@26)))))-(@40*@26))))), @42=((@12*@33)-(@20*@41)), @43=atan
((tan(uy)/((-0.192308*tan(uy))+@10))), @44=cos(@43), @45=(@13*Nfl), @46=(v1-(@16*w3)), @47=sin(@43
), @48=(v2+(@21*w3)), @49=(((@44*@46)+(@47*@48)), @50=(((@49-(@24*w))/sqrt((sq(@49)+@25))), @51=(@29
*cos(atan((-(@30*@50))))), @52=atan((((@44*@48)-(@47*@46))/sqrt((sq(@49)+@25)))), @53=(((@45*sin((@
15*atan(((@16*atan((-(@17*@50))))-(@27*@50))))))*cos(atan(((@28*((@51*@52)+atan((-(@51*@52))))))-(@
51*@52))))), @54=(@38*cos(atan((-(@39*@52))))), @55=(((@45*sin((@34*atan(((@35*atan((-(@29*@52))))-
(@36*@52))))))*cos(atan(((@37*((@54*@50)+atan((-(@54*@50))))))-(@54*@50))))), @56=(((@44*@53)-(@47*@
55)), @57=(@13*Nrr), @58=(v1-(@18*w3)), @59=(((@58-(@24*w))/sqrt((sq(@58)+@25))), @60=(@29*cos(atan
((-(@30*@59))))), @61=-0.13, @62=atan(((v2+(@61*w3))/sqrt((sq(@58)+@25)))), @63=(((@57*sin(((@15*ata
n(((@16*atan((-(@17*@59))))-(@27*@59))))))*cos(atan(((@28*((@60*@62)+atan((-(@60*@62))))))-(@60*@62
))))), @64=(@13*Nrl), @65=(v1-(@16*w3)), @66=(((@65-(@24*w))/sqrt((sq(@65)+@25))), @67=(@29*cos(ata
n((-(@30*@66))))), @68=atan(((v2+(@61*w3))/sqrt((sq(@65)+@25)))), @69=(((@64*sin(((@15*atan(((@16*at
an((-(@17*@66))))-(@27*@66))))))*cos(atan(((@28*((@67*@68)+atan((-(@67*@68))))))-(@67*@68))))), @70
=2.2187, @71=((@20*@33)+(@12*@41)), @72=(((@47*@53)+(@44*@55)), @73=(@38*cos(atan((-(@39*@62))))),
@74=(((@57*sin((@34*atan(((@35*atan((-(@29*@62))))-(@36*@62))))))*cos(atan(((@37*((@73*@59)+atan((-
(@73*@59))))))-(@73*@59))))), @75=(@38*cos(atan((-(@39*@68))))), @76=(((@64*sin((@34*atan(((@35*atan
((-(@29*@68))))-(@36*@68))))))*cos(atan(((@37*((@75*@66)+atan((-(@75*@66))))))-(@75*@66))))), @77=2
e-005, [@8, @9, ((w3+((((((((k*sin(a))*k)-(@1*dk))*y)-(sin(a)*k))*@6)-(((cos(a)*k)-((((k*cos(a))*
k)+(@4*dk))*y))*@3))/(sq(@3)+sq(@6)))*@8))+((((@5*@3)-(@2*@6))/(sq(@3)+sq(@6)))*@9)), (((((@42+@56
)+@63)+@69)/@70)+(w3*v2)), (((((@71+@72)+@74)+@76)/@70)-(w3*v1)), ((((((@16*@42)+(@21*@71))+((@18*
@56)+(@21*@72)))+((@16*@63)+(@61*@74)))+((@18*@69)+(@61*@76)))/0.027), ((T-(@24*@33))/@77), ((T-(@
24*@53))/@77), ((T-(@24*@63))/@77), ((T-(@24*@69))/@77)])
```

Figure 1.6: A printout of the expressions involved in a complete non-Euclidean vehicle model derived symbolically on a computer.

### 1.6.2 Non-Euclidean Approach

To use non-Euclidean motion models, the reader is required to think differently about how vehicle models are developed. To facilitate this, an intuitive example is provided here for developing a ground vehicle model on a surface.

Recall Figure 1.1, repeated in Figure 1.7, which illustrates the tangent contact motion model of a rigid body on a surface. When building a complete vehicle model, the motion model is but one key ingredient, among two others:

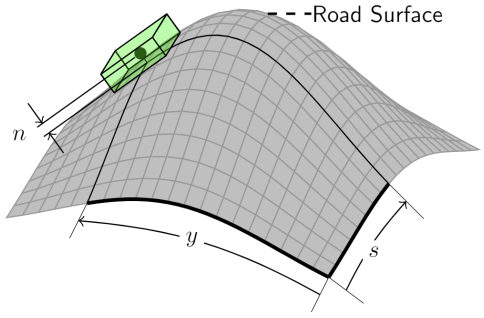



Figure 1.7: Illustration of tangent-contact motion model: A single rigid body tangent to an in contact with a road surface. The body may move across the surface, but may not move normal to it. The body must simultaneously translate and rotate when moving over a curved surface.

First, we must choose how to represent this surface. We might parameterize the surface as an elevation map, as in Figure 1.7, or using coordinates whose contours follow a lane. Examples will appear of these and other methods, and this aspect of choosing how a surface is parameterized will be referred to as a "surface class".

Second, we must also choose how the chosen motion model is driven or actuated. For instance we might use a kinematic or dynamic two track model from Section 1.5 to describe a car, or a

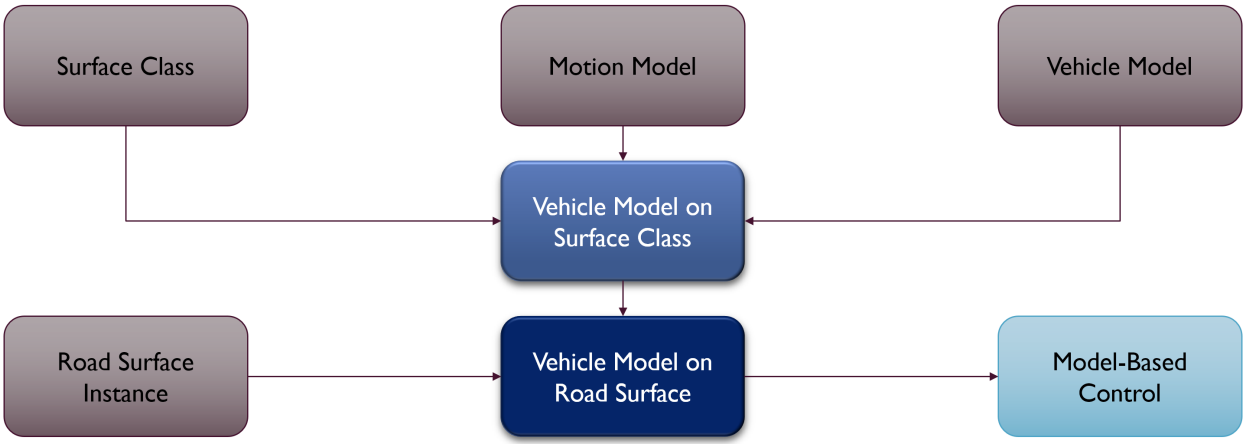



Figure 1.8: Conceptual vehicle model pipeline for non-Euclidean road surfaces. The same pipeline holds for other motion models, for instance the curve-relative motion model introduced in 1.1.2, with relabeling of the blocks.

motorcycle model from Section 1.5.3. This is fundamentally what is "unique" to the vehicle being modeled, whereas the surface and motion model are at least in part a result of external factors.

A final ingredient is necessary in the non-Euclidean case: While we can satisfactorily derive complete vehicle models using the three ingredients stated here, we need a real surface, curve, or other geometric feature to use motion models on. The complete model pipeline that results is summarized in Figure 1.8, which is filled in numerically in Chapter 4 with the motion models of Chapter 4. The same figure also holds for motion models that involve a curve instead of a road, with minor relabeling of the blocks present.

### 1.6.3 Links to Specific Vehicles and Applications

To aid the reader in finding the blocks needed to build several vehicle models, and ways in which these models may be used, the table below provides links to each for a few vehicle types.

| Vehicle | Motion Model | Vehicle Model | Example Application |
|---|---|---|---|
| Car | Tangent Contact (2.1) | Kinematic Two Track (1.5.2.1) | Off-road motion planning (5.2.2) |
| Racecar | Tangent Contact (2.1) | Dynamic Two Track (1.5.2.2) | Raceline Optimization (5.1.1) |
| Motorcycle | Tangent Contact (2.1) | Dynamic Motorcycle (1.5.3) | Raceline Optimization (5.1.1) |
| Quadrotor | Curve-Relative (3) | Four-Propeller (1.5.4) | Raceline Optimization (6.1 |

# Part I

# Non-Euclidean Motion Models

# Chapter 2
# Parametric Modeling of Vehicles on a 3D Surface

This chapter develops motion models for a rigid body relative to a surface. Two distinct cases are developed: one where the body is kinematically constrained to remain tangent to and in contact with the surface (Section 2.1), and a second in which no such constraints are present (Section 2.2). These model the motion of a body with or without roll, pitch, and heave motion, and were introduced conceptually in Section 1.1.1. In this chapter, these conceptual motion models are given a thorough mathematical treatment, and equations of motion to use them are derived.

To this end we introduce the parametric surface $\boldsymbol{x}^p(s, y) \to \mathbb{R}^3$, a two-dimensional surface in three-dimensional space, parameterized by variable $s$ and $y$. This parameterization is arbitrary, as there are infinitely many choices for a given surface. The approach developed here will be applicable to nearly any parameterization, allowing different choices for practical application. For instance, it is common practice on flat roads to choose parameterizations wherein $\boldsymbol{x}^p(s, 0)$ correlates to the center of a lane that should be followed, simplifying control tasks in this coordinate system. Similarly, parameterizations can be chosen that are easy to measure, such as an elevation map. No claim is made here as to the "best" parameterization for any given application, instead a general approach is derived which may be applied to particular applications and choices of surface parameterizations.

Following the mathematical development of each motion model, important remarks on choosing a surface parameterization are provided in Section 2.3, and common considerations for use with vehicles are provided in Section 2.4. The special case of surfaces which reference a centerline, such as a lane on a road, is discussed in Section 2.5 before concluding remarks are provided in Section 2.6. Numerical implementation and application of this chapter's motion models to vehicle models is developed later in Chapter 4.

## 2.1 Tangent Contact Motion Model

**Acknowledgement:** This section is derived in part from an Article published in Vehicle System Dynamics 2023, copyright Taylor & Francis, available online:
https://www.tandfonline.com/10.1080/00423114.2023.2205596

Our discussion of vehicles on 2D surfaces begins with the simplified picture illustrated in Figure 2.1, where we have added the orthogonal basis vectors $\boldsymbol{e}^b_{1,2,3}$ on top of the parametric surface. We assume that the vector $\boldsymbol{e}^b_3$ coincides with the normal vector of the surface $\boldsymbol{e}^p_n$. We assume that some reference location of the vehicle is located a distance $n$ from the parametric surface, which in most applications may be constant, though will be treated as if variable here.

The frame $\boldsymbol{e}^b_{1,2,3}$ will be referred to here as the "body frame", as in nearly all applications it can and should correspond to the orientation of the vehicle being modeled, such as the

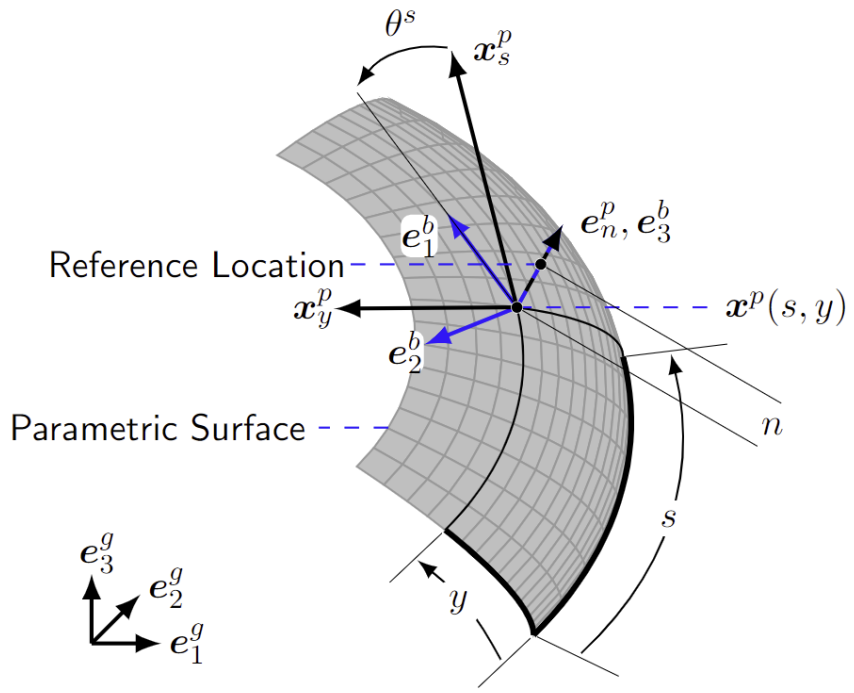



Figure 2.1: Parametric Surface and Vehicle Frame for Tangent Contact

longitudinal, lateral, and normal directions of a car.

### 2.1.1 Zero-Order Analysis

For zero-order analysis of the tangent contact approach we must relate position and orientation of the reference location and body frame to quantities from the parametric surface. We begin by defining the normal vector $\boldsymbol{e}_n^p$ in terms of the cross product of $\boldsymbol{x}_s^p$ and $\boldsymbol{x}_y^p$: the partial derivatives of the parametric surface.

$$\boldsymbol{e}_n^p = \frac{\boldsymbol{x}_s^p \times \boldsymbol{x}_y^p}{||\boldsymbol{x}_s^p \times \boldsymbol{x}_y^p||}. \tag{2.1}$$

We introduce the position[1] $\boldsymbol{x}$ of the reference location with respect to an inertial frame. With the assumptions so far, we can describe $x$ using the parametric surface instead of the typical Euclidean coordinate system $\mathbb{R}^3$:

$$\boldsymbol{x} = \boldsymbol{x}^p + n\boldsymbol{e}_n^p. \tag{2.2}$$

[1]The overloading of $\boldsymbol{x}$ and $\boldsymbol{x}^p$ is deliberate; $\boldsymbol{x}^p$ describes position (as much as can be) by the surface coordinate system.

This allows us to go from $s, y, n$ to $\boldsymbol{x}$, and will be instrumental in relating how the two vary with respect to time. The inverse map is challenging in general, and needs to be approached differently for different surface parameterizations. In all cases, restrictions on smooth vehicle motion will have the effect that $\boldsymbol{x}^p(s, y)$ is *locally*[2] the closest point to $\boldsymbol{x}$ on the surface. As a result, knowing sufficiently close values for $s, y, n$ is sufficient for finding them to arbitrary precision by optimization. We do not introduce formulae for how close is necessary, but remark that close approximations can be found for many practical surfaces: for elevation maps $s, y$ are close to their Euclidean counterparts $x_1^g, x_2^g$, and for parameterizations that characterize a lane the closest point on that lane can be a good initial guess for $s$.

To describe the orientation of the body frame we introduce the angle $\theta^s$ illustrated in Figure 2.1 with definition:

$$\cos(\theta^s) = \frac{\boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p}{\|\boldsymbol{x}_s^p\|} \qquad \sin(\theta^s) = \frac{-\boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p}{\|\boldsymbol{x}_s^p\|}. \tag{2.3}$$

Vectors $\boldsymbol{x}_s^p$ and $\boldsymbol{x}_y^p$ are the partial derivatives of the parametric surface with respect to $s$ and $y$, and form a basis for the tangent plane of the surface. $\theta^s$ fully describes the orientation of the vehicle, and may be computed as above. The inverse of this, computing $\boldsymbol{e}_{1,2,3}^b$ from $\theta^s$ can be found in two steps. First, $\boldsymbol{e}_3^b = \boldsymbol{e}_n^p$ as assumed by tangent contact. Next we consider the expansion of $\boldsymbol{e}_1^b$ and $\boldsymbol{e}_2^b$ in terms of $\boldsymbol{x}_s^p$ and $\boldsymbol{x}_y^p$:

$$\boldsymbol{e}_1^b = \alpha \boldsymbol{x}_s^p + \beta \boldsymbol{x}_y^p \qquad \boldsymbol{e}_2^b = \gamma \boldsymbol{x}_s^p + \delta \boldsymbol{x}_y^p, \tag{2.4}$$

where $\alpha$, $\beta$, $\gamma$ and $\delta$ are unknowns to solve for. We first split each equation into two equations by taking inner products with respect to $\boldsymbol{x}_s^p$ and $\boldsymbol{x}_y^p$:

$$\begin{bmatrix} \boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p \\ \boldsymbol{e}_1^b \cdot \boldsymbol{x}_y^p \end{bmatrix} = \mathbf{I} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \qquad \begin{bmatrix} \boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p \\ \boldsymbol{e}_2^b \cdot \boldsymbol{x}_y^p \end{bmatrix} = \mathbf{I} \begin{bmatrix} \gamma \\ \delta \end{bmatrix} \qquad \mathbf{I} = \begin{bmatrix} \boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p & \boldsymbol{x}_s^p \cdot \boldsymbol{x}_y^p \\ \boldsymbol{x}_s^p \cdot \boldsymbol{x}_y^p & \boldsymbol{x}_y^p \cdot \boldsymbol{x}_y^p \end{bmatrix}. \tag{2.5}$$

Here the symbol $\mathbf{I}$ denotes the first fundamental form of a parametric surface [18].

We merge and solve the two equations by introducing the matrix $\mathbf{J}$:

$$\begin{bmatrix} \alpha & \gamma \\ \beta & \delta \end{bmatrix} = \mathbf{I}^{-1}\mathbf{J} \qquad \mathbf{J} = \begin{bmatrix} \boldsymbol{x}_s^p \cdot \boldsymbol{e}_1^b & \boldsymbol{x}_s^p \cdot \boldsymbol{e}_2^b \\ \boldsymbol{x}_y^p \cdot \boldsymbol{e}_1^b & \boldsymbol{x}_y^p \cdot \boldsymbol{e}_2^b. \end{bmatrix}. \tag{2.6}$$

$\mathbf{J}$ will prove important as it is a Jacobian between the body frame and the parametric surface.

Recalling the definitions of $\alpha$ through $\gamma$ we have:

$$\begin{bmatrix} \boldsymbol{e}_1^b & \boldsymbol{e}_2^b \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_s^p & \boldsymbol{x}_y^p \end{bmatrix} \mathbf{I}^{-1}\mathbf{J}, \tag{2.7}$$

which we can compute knowing only $\theta^s$ and first order knowledge of the parametric surface.

[2]This distinction of locality is necessary on surfaces such as a figure eight, where the notion of a "closest point" becomes ambiguous.

$\mathbf{J}$ may be computed from $\theta^s$ using either of the following formulas:

$$\mathbf{J} = \begin{bmatrix} \cos(\theta^s)\left\|\boldsymbol{x}_s^p\right\| & -\sin(\theta^s)\left\|\boldsymbol{x}_s^p\right\| \\ \sin(\theta^s-\theta^p)\left\|\boldsymbol{x}_y^p\right\| & \cos(\theta^s-\theta^p)\left\|\boldsymbol{x}_y^p\right\| \end{bmatrix} \qquad \theta^p = -\sin^{-1}\left(\frac{\boldsymbol{x}_s^p\cdot\boldsymbol{x}_y^p}{\left\|\boldsymbol{x}_s^p\right\|\ \left\|\boldsymbol{x}_y^p\right\|}\right) \tag{2.8}$$

$$\mathbf{J} = \underbrace{\begin{bmatrix} \left\|\boldsymbol{x}_s^p\right\| & 0 \\ -\sin\theta^p\left\|\boldsymbol{x}_y^p\right\| & \cos\theta^p\left\|\boldsymbol{x}_y^p\right\| \end{bmatrix}}_{\mathbf{Q}} \begin{bmatrix} \cos\theta^s & -\sin\theta^s \\ \sin\theta^s & \cos\theta^s \end{bmatrix}. \tag{2.9}$$

The second expression constitutes a Q-R decomposition of $\mathbf{J}$, and may be useful when only $\theta^s$ changes. Incidentally, $\mathbf{J}\mathbf{J}^T = \mathbf{I}$. This completes our discussion of describing the pose of the body frame and reference location in the case of tangent contact.

In the process we have assumed that $\boldsymbol{x}^p$ is differentiable, and that its tangent vectors are nonzero and linearly independent. A parametric surface with the first property is "smooth", and the second property coincides with the notion of a "regular" surface. We have also assumed that the surface is orientable when we introduced $\boldsymbol{e}_n^p$. In the following sections we will also assume that the surface is time invariant. Formally, we have:

**Assumption 1** *$\boldsymbol{x}^p$ is smooth, regular, time-invariant and orientable.*

This assumption is necessary for all motion models in this chapter.

### 2.1.2 First-Order Analysis

First order analysis of tangent contact will establish relationships between the velocity of the body frame and how $s$, $y$, $n$, and $\theta^s$ change with respect to time. These expressions are geometric in nature: they are purely a function of the chosen coordinate system, not the vehicle in question. They can be exact, and in the present case we will present exact relationships. We derive said relationships by taking our zero-order expressions (Section 2.1.1) and differentiating them with respect to time.

We begin by considering orientation constraints on the body frame. Rather than using $\boldsymbol{e}_3^b = \boldsymbol{e}_n^p$, we consider the equivalent pair of expressions $\boldsymbol{e}_3^b\cdot\boldsymbol{x}_s^p = \boldsymbol{e}_3^b\cdot\boldsymbol{x}_y^p = 0$, as it will prove easier to analyze. Differentiating this with respect to time we obtain:

$$\frac{d}{dt}\left(\boldsymbol{x}_s^p\right)\cdot\boldsymbol{e}_n^p + \boldsymbol{x}_s^p\cdot\frac{d}{dt}\left(\boldsymbol{e}_3^b\right) = 0 \qquad \frac{d}{dt}\left(\boldsymbol{x}_y^p\right)\cdot\boldsymbol{e}_n^p + \boldsymbol{x}_y^p\cdot\frac{d}{dt}\left(\boldsymbol{e}_3^b\right) = 0. \tag{2.10}$$

When the time derivatives are expanded we obtain:

$$\underbrace{\begin{bmatrix} \boldsymbol{x}_{ss}^p\cdot\boldsymbol{e}_n^p & \boldsymbol{x}_{sy}^p\cdot\boldsymbol{e}_n^p \\ \boldsymbol{x}_{ys}^p\cdot\boldsymbol{e}_n^p & \boldsymbol{x}_{yy}^p\cdot\boldsymbol{e}_n^p \end{bmatrix}}_{\mathbf{II}} \begin{bmatrix} \dot{s} \\ \dot{y} \end{bmatrix} = \mathbf{J}\begin{bmatrix} -\omega_2^b \\ \omega_1^b \end{bmatrix}. \tag{2.11}$$

Here we have introduced **II** for the second fundamental form of the parametric surface [18]. This expression has an immediate result: it relates motion over the surface in terms of $\dot{s}$ and $\dot{y}$ to how the body frame must rotate ($\omega_1^b$ and $\omega_2^b$) to remain tangent to the surface.

Next, we differentiate (2.2) with respect to time:

$$\dot{s}\boldsymbol{x}_s^p + \dot{y}\boldsymbol{x}_y^p + \dot{n}\boldsymbol{e}_n^p + n\left(\omega_2^b\boldsymbol{e}_1^b - \omega_1^b\boldsymbol{e}_2^b\right) = \boldsymbol{v}, \tag{2.12}$$

We have introduced $\boldsymbol{v}$ to denote the velocity of the reference location with respect to an inertial frame. We introduce symbols $v_{1,2,3}^b$ to denote the components thereof in the body frame. By taking inner products of this expression with $\boldsymbol{x}_s^p$, $\boldsymbol{x}_y^p$, and $\boldsymbol{e}_n^p$ we obtain:

$$\mathbf{J}\left(\begin{bmatrix} v_1^b \\ v_2^b \end{bmatrix} + n\begin{bmatrix} -\omega_2^b \\ \omega_1^b \end{bmatrix}\right) = \mathbf{I}\begin{bmatrix} \dot{s} \\ \dot{y} \end{bmatrix} \qquad \dot{n} = v_3^b. \tag{2.13}$$

When we combine this with (2.11) we obtain:

$$\begin{bmatrix} \dot{s} \\ \dot{y} \end{bmatrix} = (\mathbf{I} - n\mathbf{II})^{-1}\,\mathbf{J}\begin{bmatrix} v_1^b \\ v_2^b \end{bmatrix} \qquad \dot{n} = v_3^b. \tag{2.14}$$

This expression relates velocity of the body frame to the time rate of change of $s$, $y$, and $n$. Invertibility of $(\mathbf{I} - n\mathbf{II})$ is guaranteed for $n = 0$ by our assumption that $\boldsymbol{x}^p$ is regular. When a surface is curved, it is always possible to find $n$ such that this expression is not invertible. In the one-dimnsional case, this $n$ corresponds to the radius of curvature. This requirement is often not an issue in practice, as it requires a reference location that is far above the surface compared to its curvature[3] However, in all cases it is possible to choose a reference location that is closer to the road such that this is not an issue.

Furthermore, we can combine (2.11) with (2.14) to obtain:

$$\begin{bmatrix} -\omega_2^b \\ \omega_1^b \end{bmatrix} = \mathbf{J}^{-1}\mathbf{II}\ (\mathbf{I} - n\mathbf{II})^{-1}\,\mathbf{J}\begin{bmatrix} v_1^b \\ v_2^b \end{bmatrix} \tag{2.15}$$

By construction, this expression is invariant to the parameterization of the surface, and depends only on its curvature and relative orientation of the vehicle.

The final element of our first-order surface analysis is an expression for the time derivative of $\theta^s$. We achieve this by differentiating (2.3) with respect to time, beginning with the cos term:

$$-\sin(\theta^s)\dot{\theta}^s\left\|\boldsymbol{x}_s^p\right\|^2 = \left(\frac{d}{dt}\left(\boldsymbol{e}_1^b\right)\cdot\boldsymbol{x}_s^p + \boldsymbol{e}_1^b\cdot\frac{d}{dt}\left(\boldsymbol{x}_s^p\right)\right)\left\|\boldsymbol{x}_s^p\right\| - \boldsymbol{e}_1^b\cdot\boldsymbol{x}_s^p\frac{d}{dt}\left\|\boldsymbol{x}_s^p\right\|, \tag{2.16}$$

[3] A possible scenario where this may happen is a tall unicycle on an undulating surface.

into which we can substitute the equation for $\sin(\theta^s)$ and expand the time derivatives:

$$\begin{aligned}\dot{\theta}^s = &\frac{\|\boldsymbol{x}_s^p\|}{\boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p} \frac{\omega_3^b \boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p + \boldsymbol{e}_1^b \cdot \boldsymbol{x}_{ss}^p \dot{s} + \boldsymbol{e}_1^b \cdot \boldsymbol{x}_{sy}^p \dot{y}}{\|\boldsymbol{x}_s^p\|} \\ &- \frac{\|\boldsymbol{x}_s^p\|}{\boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p} \frac{(\boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p)(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p)^{-1/2} (\frac{d}{dt}(\boldsymbol{x}_s^p) \cdot \boldsymbol{x}_s^p)}{\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p}.\end{aligned} \tag{2.17}$$

Factoring out terms,

$$\begin{aligned}\dot{\theta}^s = \omega_3^b &+ \frac{\boldsymbol{e}_1^b \cdot \boldsymbol{x}_{ss}^p - (\boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p)(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_{ss}^p)(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p)^{-1}}{\boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p} \dot{s} \\ &+ \frac{\boldsymbol{e}_1^b \cdot \boldsymbol{x}_{sy}^p - (\boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p)(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_{sy}^p)(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p)^{-1}}{\boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p} \dot{y}.\end{aligned} \tag{2.18}$$

Equation (2.18) is not useful since the denominator can be zero. However, if we expand $\boldsymbol{x}_s^p$, $\boldsymbol{x}_{ss}^p$ and $\boldsymbol{x}_{sy}^p$ into their $\boldsymbol{e}_1^b$, $\boldsymbol{e}_2^b$ and $\boldsymbol{e}_3^b$ components (see Appendix A) we obtain the expression

$$\begin{aligned}\dot{\theta}^s = \omega_3^b &+ \frac{(\boldsymbol{e}_1^b \cdot \boldsymbol{x}_{ss}^p)(\boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p) - (\boldsymbol{e}_2^b \cdot \boldsymbol{x}_{ss}^p)(\boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p)}{\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p} \dot{s} \\ &+ \frac{(\boldsymbol{e}_1^b \cdot \boldsymbol{x}_{sy}^p)(\boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p) - (\boldsymbol{e}_2^b \cdot \boldsymbol{x}_{sy}^p)(\boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p)}{\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p} \dot{y}.\end{aligned} \tag{2.19}$$

Equation (2.19) can be rewritten with the cross product:

$$\dot{\theta}^s = \omega_3^b + \frac{(\boldsymbol{x}_{ss}^p \times \boldsymbol{x}_s^p) \cdot \boldsymbol{e}_n^p}{\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p} \dot{s} + \frac{\left(\boldsymbol{x}_{yy}^p \times \boldsymbol{x}_s^p\right) \cdot \boldsymbol{e}_n^p}{\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p} \dot{y} \tag{2.20}$$

In the case of an orthogonal surface (one where $\boldsymbol{x}_s^p \cdot \boldsymbol{x}_y^p = 0$) these expressions relate to geodesic curvature (Appendix B).

### 2.1.3 Second-Order Analysis

Second-order analysis of the tangent contact model primarily focuses on studying the dynamics of a vehicle, which typically relate forces and moments to changes in velocity. Whereas our geometric first-order analysis was exact, dynamics of real physical systems are always approximate.

Given our tangent contact assumptions, variables $s$, $y$, $n$, and $\theta^s$ were a clear choice of variables to describe the pose of a body in tangent contact. The same cannot be said for describing the velocity of a body. One approach is to consider variables $v_1^b$, $v_2^b$, $v_3^b$, and $\omega_3^b$ to be independent, with $\omega_1^b$ and $\omega_2^b$ resulting from (2.20). This is the usual approach in vehicle dynamics, as these quantities are typically directly measured or estimated in real-world applications.

An alternative is to use $\dot{s}$, $\dot{y}$, $\dot{n}$, and $\dot{\theta}^s$ as independent variables to describe vehicle velocity. This approach is less straightforwards to apply in practice as measurement of $\dot{s}$ is generally challenging. However, this approach allows one to eliminate the notion of a body frame, which can be useful for systems that do not have a body frame, such as a point mass. As a result, we limit our discussion of this approach to variables $s$, $y$, and $n$, as $\theta^s$ loses its meaning. We discuss this approach first, then discuss the former.

#### 2.1.3.1 Parametric Frame Approach

Continuing the theme of our first-order analysis, we apply another round of differentiation to the tangent contact relationships, beginning with orientation:

$$\mathbf{II}\begin{bmatrix}\ddot{s}\\ \ddot{y}\end{bmatrix}+\begin{bmatrix}\boldsymbol{x}^p_{sss}\cdot\boldsymbol{e}^p_n & \boldsymbol{x}^p_{ssy}\cdot\boldsymbol{e}^p_n\\ \boldsymbol{x}^p_{ssy}\cdot\boldsymbol{e}^p_n & \boldsymbol{x}^p_{syy}\cdot\boldsymbol{e}^p_n\end{bmatrix}\begin{bmatrix}\dot{s}\\ \dot{y}\end{bmatrix}\dot{s}+\begin{bmatrix}\boldsymbol{x}^p_{ssy}\cdot\boldsymbol{e}^p_n & \boldsymbol{x}^p_{syy}\cdot\boldsymbol{e}^p_n\\ \boldsymbol{x}^p_{syy}\cdot\boldsymbol{e}^p_n & \boldsymbol{x}^p_{yyy}\cdot\boldsymbol{e}^p_n\end{bmatrix}\begin{bmatrix}\dot{s}\\ \dot{y}\end{bmatrix}\dot{y}=\mathbf{J}\begin{bmatrix}-\dot{\omega}^b_2-\omega^b_1\omega^b_3\\ \dot{\omega}^b_1-\omega^b_2\omega^b_3\end{bmatrix} \tag{2.21}$$

Next, for position we introduce $\boldsymbol{a}$ as the acceleration[4] of the vehicle reference point with respect to an inertial frame. We then differentiate (2.12) with respect to time:

$$\begin{aligned}\boldsymbol{a}=\boldsymbol{x}^p_s\ddot{s}+\boldsymbol{x}^p_y\ddot{y}+\boldsymbol{x}^p_{ss}\dot{s}^2+2\boldsymbol{x}^p_{sy}\dot{s}\dot{y}+\boldsymbol{x}^p_{yy}\dot{y}^2+\ddot{n}\boldsymbol{e}^p_n+2\dot{n}\left(\omega^b_2\boldsymbol{e}^b_1-\omega^b_1\boldsymbol{e}^b_2\right)\\ +n\left(\dot{\omega}^b_2\boldsymbol{e}^b_1-\dot{\omega}^b_1\boldsymbol{e}^b_2+\omega^b_2\left(\omega^b_3\boldsymbol{e}^b_2-\omega^b_2\boldsymbol{e}^b_3\right)-\omega^b_1\left(\omega^b_1\boldsymbol{e}^b_3-\omega^b_3\boldsymbol{e}^b_1\right)\right)\end{aligned} \tag{2.22}$$

Once more, we factor this into independent expressions by taking inner products with respect to $\boldsymbol{x}^p_s$, $\boldsymbol{x}^p_y$, and $\boldsymbol{e}^p_n$:

$$\begin{aligned}\begin{bmatrix}\boldsymbol{a}\cdot\boldsymbol{x}^p_s\\ \boldsymbol{a}\cdot\boldsymbol{x}^p_y\end{bmatrix}&=\mathbf{I}\begin{bmatrix}\ddot{s}\\ \ddot{y}\end{bmatrix}+\begin{bmatrix}\boldsymbol{x}^p_{ss}\cdot\boldsymbol{x}^p_s & \boldsymbol{x}^p_{sy}\cdot\boldsymbol{x}^p_s\\ \boldsymbol{x}^p_{ss}\cdot\boldsymbol{x}^p_y & \boldsymbol{x}^p_{sy}\cdot\boldsymbol{x}^p_y\end{bmatrix}\begin{bmatrix}\dot{s}\\ \dot{y}\end{bmatrix}\dot{s}+\begin{bmatrix}\boldsymbol{x}^p_{sy}\cdot\boldsymbol{x}^p_s & \boldsymbol{x}^p_{yy}\cdot\boldsymbol{x}^p_s\\ \boldsymbol{x}^p_{sy}\cdot\boldsymbol{x}^p_y & \boldsymbol{x}^p_{yy}\cdot\boldsymbol{x}^p_y\end{bmatrix}\begin{bmatrix}\dot{s}\\ \dot{y}\end{bmatrix}\dot{y}\\ &\quad-2\dot{n}\mathbf{J}\begin{bmatrix}-\omega^b_2\\ \omega^b_1\end{bmatrix}+n\mathbf{J}\begin{bmatrix}\dot{\omega}^b_2+\omega^b_1\omega^b_3\\ -\dot{\omega}^b_1+\omega^b_2\omega^b_3\end{bmatrix}\end{aligned} \tag{2.23}$$

$$\boldsymbol{a}\cdot\boldsymbol{e}^p_n=\ddot{n}+\begin{bmatrix}\dot{s} & \dot{y}\end{bmatrix}\mathbf{II}\begin{bmatrix}\dot{s}\\ \dot{y}\end{bmatrix}-n\left(\omega^b_1\right)^2-n\left(\omega^b_2\right)^2 \tag{2.24}$$

[4]Coordinate acceleration, as opposed to proper acceleration

Using (2.21), (2.11), and the fact that $\mathbf{J}\mathbf{J}^T = \mathbf{I}$ we simplify this pair of equations to:

$$\begin{aligned}\begin{bmatrix}\boldsymbol{a}\cdot\boldsymbol{x}_s^p\\ \boldsymbol{a}\cdot\boldsymbol{x}_y^p\end{bmatrix} &= (\mathbf{I}-n\mathbf{II})\begin{bmatrix}\ddot{s}\\ \ddot{y}\end{bmatrix} - 2\mathbf{II}\begin{bmatrix}\dot{s}\\ \dot{y}\end{bmatrix}\dot{n}\\ &+\begin{bmatrix}(\boldsymbol{x}_{ss}^p\cdot\boldsymbol{x}_s^p - n\boldsymbol{x}_{sss}^p\cdot\boldsymbol{e}_n^p) & \left(\boldsymbol{x}_{sy}^p\cdot\boldsymbol{x}_s^p - n\boldsymbol{x}_{ssy}^p\cdot\boldsymbol{e}_n^p\right)\\ \left(\boldsymbol{x}_{ss}^p\cdot\boldsymbol{x}_y^p - n\boldsymbol{x}_{ssy}^p\cdot\boldsymbol{e}_n^p\right) & \left(\boldsymbol{x}_{sy}^p\cdot\boldsymbol{x}_y^p - n\boldsymbol{x}_{syy}^p\cdot\boldsymbol{e}_n^p\right)\end{bmatrix}\begin{bmatrix}\dot{s}\\ \dot{y}\end{bmatrix}\dot{s}\\ &+\begin{bmatrix}\left(\boldsymbol{x}_{sy}^p\cdot\boldsymbol{x}_s^p - n\boldsymbol{x}_{ssy}^p\cdot\boldsymbol{e}_n^p\right) & \left(\boldsymbol{x}_{yy}^p\cdot\boldsymbol{x}_s^p - n\boldsymbol{x}_{syy}^p\cdot\boldsymbol{e}_n^p\right)\\ \left(\boldsymbol{x}_{sy}^p\cdot\boldsymbol{x}_y^p - n\boldsymbol{x}_{syy}^p\cdot\boldsymbol{e}_n^p\right) & \left(\boldsymbol{x}_{yy}^p\cdot\boldsymbol{x}_y^p - n\boldsymbol{x}_{yyy}^p\cdot\boldsymbol{e}_n^p\right)\end{bmatrix}\begin{bmatrix}\dot{s}\\ \dot{y}\end{bmatrix}\dot{y}\end{aligned} \tag{2.25}$$

$$\boldsymbol{a}\cdot\boldsymbol{e}_n^p = \ddot{n} + \begin{bmatrix}\dot{s} & \dot{y}\end{bmatrix}\left(\mathbf{II} - n\mathbf{II}\left(\mathbf{I}^{-1}\right)\mathbf{II}\right)\begin{bmatrix}\dot{s}\\ \dot{y}\end{bmatrix} \tag{2.26}$$

These expressions are lengthy, however they are exactly the sought after goal: second order equations for $s$, $y$, and $n$. From this it is straightforwards to develop a point mass model by treating $\boldsymbol{a}$ as an input, such as by treating its components in the global, inertial frame as inputs. We remark that the form above has a significant geometric interpretation, one shown separately in [19] with the key result in Equation 2.13 thereof. Terms that prefix second order derivatives of our coordinates constitute the metric tensor of the $s-y-n$ manifold and tell us how "stretched out" the manifold is. Meanwhile terms that multiply two coordinate derivatives (eg. $\dot{s}\dot{y}$ or $\dot{s}^2$) are Christoffel symbols of the manifold, give rise to fictitious Coriolis and centrifugal forces, and in loose terms tell us how "curved" the manifold is. When $n = 0$, the above reduce exactly to the metric tensor and Christoffel symbols of $\boldsymbol{x}^p$; the expressions for $n \neq 0$ hold for the special 3D manifold of $s$, $y$, and $n$ considered here.

#### 2.1.3.2 Body Frame Approach

When the body frame is retained and velocity is expressed therein, it is necessary to determine the time derivatives of $v_1^b$, $v_2^b$, $v_3^b$, and $\omega_3^b$. These are typically a function of forces, such as the tires on a car. However, in many cases it is important to model $\dot{\omega}_1^b$ and $\dot{\omega}_2^b$, as these factor into the weight distribution of a vehicle[5]. As with $\omega_1^b$ and $\omega_2^b$, there is an exact constraint on $\dot{\omega}_1^b$ and $\dot{\omega}_2^b$ to remain tangent to the surface. An exact expression may be obtained by using (2.25) to remove $\ddot{s}$ and $\ddot{y}$ from (2.21), and then expressing $\boldsymbol{a}$ in terms of $\dot{v}_1^b$, $\dot{v}_2^b$, and terms involving linear and angular velocity of reference location and the body frame.

However, we will opt instead to use the approximation that the terms of (2.15) involving the shape of the surface are nearly constant, that is:

$$\begin{bmatrix}-\dot{\omega}_2^b\\ \dot{\omega}_1^b\end{bmatrix} \approx \mathbf{J}^{-1}\mathbf{II}\ (\mathbf{I}-n\mathbf{II})^{-1}\,\mathbf{J}\begin{bmatrix}\dot{v}_1^b\\ \dot{v}_2^b\end{bmatrix}. \tag{2.27}$$

[5]As we have assumed that the body frame always remains tangent to the surface, we have effectively assumed that the suspension of a four-wheeled vehicle responds immediately to changes in roll and pitch moment required to remain tangent to the surface. This is discussed more in Section 2.4

This has the rough physical interpretation that the curvature of the surface changes gradually, which will not hold everywhere in practice. However, the notion of tangent contact for a vehicle makes a similar approximation in the sense that the surface itself must curve gradually relative to the size of the vehicle, for instance "tangent contact" is by no means satisfied for a vehicle on a speed bump, but may be a suitable approximation on a banked turn at a racetrack. Equation 2.27 makes a similar approximation on the rate of change thereof, one which was experimentally validated in [20].

The body frame approach is the usual approach for developing vehicle models since the presence of a body-fixed frame is convenient in engineering practice, even for systems approximated as a point mass. For this reason this approach will be the one used hereafter when using the tangent contact motion model, with Section 2.1.3.1 provided for reference.

### 2.1.4 Complete Motion Model

Equations summarizing the complete motion model are provided below, with the typical body frame approach for second order analysis assumed:

$$\boldsymbol{x} = \boldsymbol{x}^p + n\boldsymbol{e}_n^p \tag{2.28a}$$

$$\begin{bmatrix} \dot{s} \\ \dot{y} \end{bmatrix} = (\mathbf{I} - n\mathbf{II})^{-1}\,\mathbf{J}\begin{bmatrix} v_1^b \\ v_2^b \end{bmatrix} \tag{2.28b}$$

$$\dot{n} = v_3^b \tag{2.28c}$$

$$\dot{\theta}^s = \omega_3^b + \frac{(\boldsymbol{x}_{ss}^p \times \boldsymbol{x}_s^p)\cdot \boldsymbol{e}_n^p}{\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p}\dot{s} + \frac{\left(\boldsymbol{x}_{yy}^p \times \boldsymbol{x}_s^p\right)\cdot \boldsymbol{e}_n^p}{\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p}\dot{y} \tag{2.28d}$$

$$\begin{bmatrix} -\omega_2^b \\ \omega_1^b \end{bmatrix} = \mathbf{J}^{-1}\mathbf{II}\ (\mathbf{I} - n\mathbf{II})^{-1}\,\mathbf{J}\begin{bmatrix} v_1^b \\ v_2^b \end{bmatrix} \tag{2.28e}$$

$$\begin{bmatrix} -\dot{\omega}_2^b \\ \dot{\omega}_1^b \end{bmatrix} \approx \mathbf{J}^{-1}\mathbf{II}\ (\mathbf{I} - n\mathbf{II})^{-1}\,\mathbf{J}\begin{bmatrix} \dot{v}_1^b \\ \dot{v}_2^b \end{bmatrix}. \tag{2.28f}$$

These equations have a number of direct consequences, even without considering a specific vehicle.

First, many applications will assume $n$ to be constant, fixing $v_3^b$ as well.

Second, angular velocity constraints on $\omega_1^b, \omega_2^b$ and rates thereof factor into the Newton Euler equations (1.13). This substitution results in velocity-squared terms which alter the net normal force $F_3^b$ on the vehicle, and are the expected Coriolis forces for a curved surface. Similar effects are present on the roll and pitch moments $K_1^b$ and $K_2^b$. These are a part of the motion model, as they are present for any vehicle.

Finally, all expressions depend on the surface. Handling this dependence in a numerical implementation is the subject of Chapter 4.

## 2.2 Unconstrained Motion Model

It is natural to question the validity of the tangent contact vehicle model, particularly for vehicles that need to exhibit roll, pitch, and heave motion simultaneously. To this effect, we consider the process of the previous section, but without the tangent contact assumption.

In considering this, it will no longer be possible to describe the body frame with the single angle $\theta^s$, and we will always need $s$, $y$, and $n$ to describe position. We might equivalently describe pose with standard Euclidean coordinates, however we maintain the notion of a surface parameterization and coordinates relative to it for the important reason that these relative coordinates, even if not used as state variables in a model, will prove the most convenient to describe the physics of a vehicle, particularly tire and suspension forces. As a result, we will introduce a notion of complete 3D orientation of the body frame relative to the parametric surface, as opposed to just $\theta^s$, and determine expressions for how this relative orientation evolves with respect to time.

### 2.2.1 Zero-Order Analysis

As in the case of tangent contact, we have the position relationship:

$$\boldsymbol{x} = \boldsymbol{x}^p + n\boldsymbol{e}_n^p.$$

However, we must now introduce additional considerations for orientation. It is standard practice to discuss orientation in terms of a rotation matrix between two orthonormal frames of reference. The body frame $\boldsymbol{e}_{1,2,3}^b$ is already orthonormal, however this is not the case for the parametric surface, as $\boldsymbol{x}_s^p$ and $\boldsymbol{x}_y^p$ may not be normal or orthogonal to one another. To resolve this, we consider relative orientation from the Gram-Schmidt orthonormalization of $\boldsymbol{x}_s^p, \boldsymbol{x}_y^p, \boldsymbol{e}_n^p$, with the result being the basis:

$$\boldsymbol{e}_s^p = \frac{\boldsymbol{x}_s^p}{\|\boldsymbol{x}_s^p\|} \qquad \boldsymbol{e}_\perp^p = \frac{\boldsymbol{x}_y^p - \boldsymbol{e}_s^p\left(\boldsymbol{e}_s^p \cdot \boldsymbol{x}_y^p\right)}{\|\boldsymbol{x}_y^p - \boldsymbol{e}_s^p\left(\boldsymbol{e}_s^p \cdot \boldsymbol{x}_y^p\right)\|} \qquad \boldsymbol{e}_n^p = \boldsymbol{e}_n^p. \tag{2.29}$$

This basis is orthonormal by construction, and allows us to introduce a notion of relative orientation of the body frame similar to $\theta^s$ but which allows for arbitrary rotation in SO3. We denote this as $\mathbf{R}^{pb}$ with definition:

$$\begin{bmatrix}\boldsymbol{e}_1^b & \boldsymbol{e}_2^b & \boldsymbol{e}_3^b\end{bmatrix} = \begin{bmatrix}\boldsymbol{e}_s^p & \boldsymbol{e}_\perp^p & \boldsymbol{e}_n^p\end{bmatrix}\mathbf{R}^{pb}. \tag{2.30}$$

Our goal will be to determine effective angular velocities for the rotation group $\mathbf{R}^{pb}$, which may be applied to any parameterization of it.

Considerations for going to and from $\boldsymbol{x}$ and $s$, $y$, $n$ follow as in Section 2.1.1. For orientation conversion notice that we have introduced $\mathbf{R}^{pb}$ as a relationship between vectors.

In practice these vectors will usually be expressed in the global frame. Doing so expands (2.30) into three equations:

$$\underbrace{\begin{bmatrix} \boldsymbol{e}_1^b \cdot \boldsymbol{e}_1^g & \boldsymbol{e}_2^b \cdot \boldsymbol{e}_1^g & \boldsymbol{e}_3^b \cdot \boldsymbol{e}_1^g \\ \boldsymbol{e}_1^b \cdot \boldsymbol{e}_2^g & \boldsymbol{e}_2^b \cdot \boldsymbol{e}_2^g & \boldsymbol{e}_3^b \cdot \boldsymbol{e}_2^g \\ \boldsymbol{e}_1^b \cdot \boldsymbol{e}_3^g & \boldsymbol{e}_2^b \cdot \boldsymbol{e}_3^g & \boldsymbol{e}_3^b \cdot \boldsymbol{e}_3^g \end{bmatrix}}_{\mathbf{R}^{gb}} = \underbrace{\begin{bmatrix} \boldsymbol{e}_s^p \cdot \boldsymbol{e}_1^g & \boldsymbol{e}_\perp^p \cdot \boldsymbol{e}_1^g & \boldsymbol{e}_n^p \cdot \boldsymbol{e}_1^g \\ \boldsymbol{e}_s^p \cdot \boldsymbol{e}_2^g & \boldsymbol{e}_\perp^p \cdot \boldsymbol{e}_2^g & \boldsymbol{e}_n^p \cdot \boldsymbol{e}_2^g \\ \boldsymbol{e}_s^p \cdot \boldsymbol{e}_3^g & \boldsymbol{e}_\perp^p \cdot \boldsymbol{e}_3^g & \boldsymbol{e}_n^p \cdot \boldsymbol{e}_3^g \end{bmatrix}}_{\mathbf{R}^{gp}} \mathbf{R}^{pb}. \tag{2.31}$$

These new matrix terms have immediate physical interpretation: The left is the orientation of the body frame relative to the global frame, and the right is the orientation of the orthonormalized parametric surface frame relative to the global frame. Either $\mathbf{R}^{gb}$ or $\mathbf{R}^{pb}$ may be treated as state variables in a model, as they are related by $\mathbf{R}^{gp}$, which is known from $s$, $y$, and $n$.

No choice is assumed to be better here. $\mathbf{R}^{pb}$ is necessary to model tire and suspension forces, meaning the expressions for those forces are simplest if $\mathbf{R}^{pb}$ is a state variable. However, the time derivative of $\mathbf{R}^{pb}$ is more complicated than that of $\mathbf{R}^{gb}$, as it includes surface-curvature terms. These terms are derived in the next subsection.

As in the case of our tangent contact analysis, it will be instrumental to introduce a Jacobian between the body frame and the parametric surface. We denote this as $\mathbf{J}_3$ to symbolize its similar meaning, with the major difference here that it will be a $3 \times 3$ matrix instead. We define it as:

$$\mathbf{J}_3 = \begin{bmatrix} \boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p & \boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p & \boldsymbol{e}_3^b \cdot \boldsymbol{x}_s^p \\ \boldsymbol{e}_1^b \cdot \boldsymbol{x}_y^p & \boldsymbol{e}_2^b \cdot \boldsymbol{x}_y^p & \boldsymbol{e}_3^b \cdot \boldsymbol{x}_y^p \\ \boldsymbol{e}_1^b \cdot \boldsymbol{e}_n^p & \boldsymbol{e}_2^b \cdot \boldsymbol{e}_n^p & \boldsymbol{e}_3^b \cdot \boldsymbol{e}_n^p \end{bmatrix}. \tag{2.32}$$

As in the tangent contact case, it can be decomposed into a lower-triangular matrix multiplied by a rotation matrix. This can be shown by taking inner products of (2.30) with respect to $\boldsymbol{x}_s^p$, $\boldsymbol{x}_y^p$, and $\boldsymbol{e}_n^p$:

$$\begin{bmatrix} \boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p & \boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p & \boldsymbol{e}_3^b \cdot \boldsymbol{x}_s^p \end{bmatrix} = \begin{bmatrix} ||\boldsymbol{x}_s^p|| & 0 & 0 \end{bmatrix} \mathbf{R}^{pb} \tag{2.33}$$

$$\begin{bmatrix} \boldsymbol{e}_1^b \cdot \boldsymbol{x}_y^p & \boldsymbol{e}_2^b \cdot \boldsymbol{x}_y^p & \boldsymbol{e}_3^b \cdot \boldsymbol{x}_y^p \end{bmatrix} = \begin{bmatrix} -\sin\theta^p \left\|\boldsymbol{x}_y^p\right\| & \cos\theta^p \left\|\boldsymbol{x}_y^p\right\| & 0 \end{bmatrix} \mathbf{R}^{pb} \tag{2.34}$$

$$\begin{bmatrix} \boldsymbol{e}_1^b \cdot \boldsymbol{e}_n^p & \boldsymbol{e}_2^b \cdot \boldsymbol{e}_n^p & \boldsymbol{e}_3^b \cdot \boldsymbol{e}_n^p \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \mathbf{R}^{pb}. \tag{2.35}$$

Stacking these on top of one another, we obtain $\mathbf{J}_3$:

$$\mathbf{J}_3 = \begin{bmatrix} ||\boldsymbol{x}_s^p|| & 0 & 0 \\ -\sin\theta^p ||\boldsymbol{x}_y^p|| & \cos\theta^p ||\boldsymbol{x}_y^p|| & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{R}^{pb}. \tag{2.36}$$

### 2.2.2 First-Order Analysis

In the case of tangent contact, we used the constraint $\boldsymbol{e}_n^p = \boldsymbol{e}_3^b$ to help study time derivatives of either. In this case we cannot, so we introduce the general result[6] for the time derivative of $\boldsymbol{e}_n^p$.

First observe that

$$\boldsymbol{e}_n^p \cdot \boldsymbol{x}_s^p = 0 \tag{2.37a}$$

$$\boldsymbol{e}_n^p \cdot \boldsymbol{x}_y^p = 0, \tag{2.37b}$$

which is very similar to our tangent contact constraints. Differentiating with respect to time we have:

$$\frac{d}{dt}\left(\boldsymbol{e}_n^p\right) \cdot \boldsymbol{x}_s^p = -\boldsymbol{e}_n^p \cdot \left(\boldsymbol{x}_{ss}^p \dot{s} + \boldsymbol{x}_{sy}^p \dot{y}\right) \tag{2.38a}$$

$$\frac{d}{dt}\left(\boldsymbol{e}_n^p\right) \cdot \boldsymbol{x}_y^p = -\boldsymbol{e}_n^p \cdot \left(\boldsymbol{x}_{sy}^p \dot{s} + \boldsymbol{x}_{yy}^p \dot{y}\right), \tag{2.38b}$$

from which we conclude:

$$\begin{bmatrix} \frac{d}{dt}\left(\boldsymbol{e}_n^p\right) \cdot \boldsymbol{x}_s^p \\ \frac{d}{dt}\left(\boldsymbol{e}_n^p\right) \cdot \boldsymbol{x}_y^p \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_{ss}^p \cdot \boldsymbol{e}_n^p & \boldsymbol{x}_{sy}^p \cdot \boldsymbol{e}_n^p \\ \boldsymbol{x}_{ys}^p \cdot \boldsymbol{e}_n^p & \boldsymbol{x}_{yy}^p \cdot \boldsymbol{e}_n^p \end{bmatrix} \begin{bmatrix} \dot{s} \\ \dot{y} \end{bmatrix} = -\mathbf{II} \begin{bmatrix} \dot{s} \\ \dot{y} \end{bmatrix}. \tag{2.39}$$

Now suppose we want the form $\frac{d}{dt}\boldsymbol{e}_n^p = a\boldsymbol{x}_s^p + b\boldsymbol{x}_y^p$. Then:

$$\begin{bmatrix} \boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p & \boldsymbol{x}_s^p \cdot \boldsymbol{x}_y^p \\ \boldsymbol{x}_s^p \cdot \boldsymbol{x}_y^p & \boldsymbol{x}_y^p \cdot \boldsymbol{x}_y^p \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \frac{d}{dt}\left(\boldsymbol{e}_n^p\right) \cdot \boldsymbol{x}_s^p \\ \frac{d}{dt}\left(\boldsymbol{e}_n^p\right) \cdot \boldsymbol{x}_y^p \end{bmatrix}. \tag{2.40}$$

Finally:

$$\frac{d}{dt}\boldsymbol{e}_n^p = \begin{bmatrix} \boldsymbol{x}_s^p & \boldsymbol{x}_y^p \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = -\begin{bmatrix} \boldsymbol{x}_s^p & \boldsymbol{x}_y^p \end{bmatrix} \mathbf{I}^{-1}\mathbf{II} \begin{bmatrix} \dot{s} \\ \dot{y} \end{bmatrix}, \tag{2.41}$$

which is the general result.

It now follows that the time derivative of the position equality is:

$$\dot{s}\boldsymbol{x}_s^p + \dot{y}\boldsymbol{x}_y^p + \dot{n}\boldsymbol{e}_n^p + n\left(-\begin{bmatrix} \boldsymbol{x}_s^p & \boldsymbol{x}_y^p \end{bmatrix} \mathbf{I}^{-1}\mathbf{II} \begin{bmatrix} \dot{s} \\ \dot{y} \end{bmatrix}\right) = v_1^b \boldsymbol{e}_1^b + v_2^b \boldsymbol{e}_2^b + v_3^b \boldsymbol{e}_3^b. \tag{2.42}$$

When simplified by taking inner products with $\boldsymbol{x}_s^p$, $\boldsymbol{x}_y^p$, and $\boldsymbol{e}_n^p$ we arrive at a similar form to that from tangent contact:

$$\begin{bmatrix} \mathbf{I} - n\mathbf{II} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{s} \\ \dot{y} \\ \dot{n} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p & \boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p & \boldsymbol{e}_3^b \cdot \boldsymbol{x}_s^p \\ \boldsymbol{e}_1^b \cdot \boldsymbol{x}_y^p & \boldsymbol{e}_2^b \cdot \boldsymbol{x}_y^p & \boldsymbol{e}_3^b \cdot \boldsymbol{x}_y^p \\ \boldsymbol{e}_1^b \cdot \boldsymbol{e}_n^p & \boldsymbol{e}_2^b \cdot \boldsymbol{e}_n^p & \boldsymbol{e}_3^b \cdot \boldsymbol{e}_n^p \end{bmatrix}}_{\mathbf{J}_3} \begin{bmatrix} v_1^b \\ v_2^b \\ v_3^b \end{bmatrix}. \tag{2.43}$$

[6] $\boldsymbol{e}_n^p$ is often referred to as the Gauss map of a surface, similar formulae to the result here may be found in many differential geometry texts, however are often limited to orthogonal surface parameterizations.

Here "0" in the block matrix corresponds to an appropriately sized block of zeros.

This relationship is very similar to that of tangent contact, with the difference that the Jacobian involved now has additional degrees of freedom. However, it is still straightforwards to evaluate in terms of $\mathbf{R}^{pb}$ and the parametric surface, per Equation (2.36).

To complete our first order analysis, we must now determine the time derivative of $\mathbf{R}^{pb}$, so that it is possible to use a parameterization of $\mathbf{R}^{pb}$ as state variables in developing a vehicle model. We will express the result in an effective angular velocity that determines the time derivative of $\mathbf{R}^{pb}$, which will depend both on the parametric surface and on $\boldsymbol{\omega}^b$. To derive the contribution from the parametric surface, we first recognize that the time derivative of $\boldsymbol{e}_s^p$, $\boldsymbol{e}_\perp^p$, and $\boldsymbol{e}_n^p$ satisfy the form:

$$\frac{d}{dt}\begin{bmatrix}\boldsymbol{e}_s^p & \boldsymbol{e}_\perp^p & \boldsymbol{e}_n^p\end{bmatrix} = \begin{bmatrix}\boldsymbol{e}_s^p & \boldsymbol{e}_\perp^p & \boldsymbol{e}_n^p\end{bmatrix}\begin{bmatrix}0 & -\omega_n^p & \omega_\perp^p \\ \omega_n^p & 0 & -\omega_s^p \\ -\omega_\perp^p & \omega_s^p & 0\end{bmatrix}. \tag{2.44}$$

This is the case as they are orthonormal for all time. The angular velocity terms $\omega_s^p$, $\omega_\perp^p$, and $\omega_n^p$ are due to the curvature of the surface and the rate at which the vehicle moves over it. To determine expressions for the first two components: $\omega_s^p$ and $\omega_\perp^p$ we recall the relationship derived earlier for $\frac{d}{dt}\boldsymbol{e}_n^p$ and equate it to the relationship above. This can then be solved in a straightforwards manner, taking inner products with respect to $\boldsymbol{x}_s^p$ and $\boldsymbol{x}_y^p$ to isolate for $\omega_s^p$ and $\omega_\perp^p$:

$$\omega_\perp^p \boldsymbol{e}_s^p - \omega_s^p \boldsymbol{e}_\perp^p = \frac{d}{dt}\boldsymbol{e}_n^p \tag{2.45}$$

$$\omega_\perp^p \boldsymbol{e}_s^p - \omega_s^p \boldsymbol{e}_\perp^p = -\begin{bmatrix}\boldsymbol{x}_s^p & \boldsymbol{x}_y^p\end{bmatrix}\mathbf{I}^{-1}\mathbf{II}\begin{bmatrix}\dot{s} \\ \dot{y}\end{bmatrix} \tag{2.46}$$

$$\begin{bmatrix}\|\boldsymbol{x}_s^p\| & 0 \\ -\sin\theta^p \left\|\boldsymbol{x}_y^p\right\| & \cos\theta^p \left\|\boldsymbol{x}_y^p\right\|\end{bmatrix}\begin{bmatrix}\omega_\perp^p \\ -\omega_s^p\end{bmatrix} = -\mathbf{II}\begin{bmatrix}\dot{s} \\ \dot{y}\end{bmatrix} \tag{2.47}$$

$$\begin{bmatrix}-\omega_\perp^p \\ \omega_s^p\end{bmatrix} = \begin{bmatrix}\frac{1}{\|\boldsymbol{x}_s^p\|} & 0 \\ \frac{\tan\theta^p}{\|\boldsymbol{x}_s^p\|} & \frac{1}{\cos\theta^p\|\boldsymbol{x}_y^p\|}\end{bmatrix}\mathbf{II}\begin{bmatrix}\dot{s} \\ \dot{y}\end{bmatrix}. \tag{2.48}$$

We have in fact already found $\omega_n^p$ in the analysis of the tangent contact model. Using (2.20) and our convention for $\omega_n^p$ we have:

$$\omega_n^p = -\frac{(\boldsymbol{x}_{ss}^p \times \boldsymbol{x}_s^p)\cdot \boldsymbol{e}_n^p}{\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p}\dot{s} - \frac{\left(\boldsymbol{x}_{yy}^p \times \boldsymbol{x}_s^p\right)\cdot \boldsymbol{e}_n^p}{\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p}\dot{y}. \tag{2.49}$$

This tells us how $\boldsymbol{e}_s^p$, $\boldsymbol{e}_\perp^p$, and $\boldsymbol{e}_n^p$ vary with respect to time. The effective angular velocity for the rate of change of $\mathbf{R}^{pb}$ then follows from standard rotation theory:

$$\begin{bmatrix}\omega_1^b \\ \omega_2^b \\ \omega_3^b\end{bmatrix} - \left(\mathbf{R}^{pb}\right)^T \begin{bmatrix}\omega_s^p \\ \omega_\perp^p \\ \omega_n^p\end{bmatrix}. \tag{2.50}$$

Although there are numerous ways to parameterize the rotation group SO3, of which $\mathbf{R}^{pb}$ is a member, all of them provide relationships to relate angular velocity to how their parameters vary with respect to time. As a result, this is sufficient to describe the time derivative of $\mathbf{R}^{pb}$, and completes the first-order analysis of the unconstrained body approach.

### 2.2.3 Second Order Analysis

In the case of the unconstrained body model it is unnecessary to discuss second order effects from the surface. This is because the velocity of the body frame and reference location are entirely unconstrained, unlike the case of tangent contact. Expressions for how linear and angular velocity change with respect to time are emergent from mechanics, for instance the net force and moment on the vehicle and how the momentum thereof changes with respect to time.

One can of course extend the analysis of a point mass in Section 2.1.3.1 to include orientation as well, and derive second order equations. However, in the present case the result of such a process is not the elimination of the body frame of reference but the replacement of equations for the time rate of change of $v^b_{1,2,3}$ and $\omega^b_{1,2,3}$ with second order equations for $s$, $y$, $n$, and $\mathbf{R}^{pb}$ that are far more complicated than necessary. While there may be potential advantages to this, we consider the more practical approach of explicitly using the ever-present body frame to be far superior for practical applications. A clear example of this is $\mathbf{R}^{pb}$ itself: $\mathbf{R}^{pb}$ does not live within a vector space, but its derivative does, and may be cleanly described by $\omega^b_{1,2,3}$. Introducing $\omega^b_{1,2,3}$ directly as opposed to $\dot{\mathbf{R}}^{pb}$ means we need not concern ourselves with how to parameterize $\dot{\mathbf{R}}^{pb}$, much less determine an equation for it involving long algebraic expressions derived from $\boldsymbol{x}^p$.

### 2.2.4 Complete Motion Model

The complete unconstrained motion model is summarized as using Equation 2.43 to describe the pose of a vehicle:

$$\begin{bmatrix} \dot{s} \\ \dot{y} \\ \dot{n} \end{bmatrix} = \begin{bmatrix} (\mathbf{I} - n\mathbf{II})^{-1} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{J}_3 \begin{bmatrix} v^b_1 \\ v^b_2 \\ v^b_3 \end{bmatrix}. \tag{2.51a}$$

It is then up to the user to determine if they wish to build a vehicle model with vehicle orientation expressed relative to the global frame of reference or with respect to the surface. In the former case, Equation (1.10b) provides equations of motion for how this orientation changes. When using orientation relative to the surface as a vehicle state, the effective angular velocity of this rotation group is modified by the curvature of the surface, as provided by equation (2.50).

## 2.3 Choice of Surface Parameterization

The motion models presented hereto have assumed the existence of a surface parameterization, with restrictions in Assumption 1. This leaves great flexibility as to the choice of how a surface is parameterized. For instance, parameterizations may be chosen that simplify control objectives or are easy to measure. One might parameterize a road such that the contour $y = 0$ coincides with the center of the road while other level sets of $y$ coincide with designated lanes on the road. Meanwhile, for measuring a road surface it is often easy to use an elevation map, which may be used directly as a motion model.

No claims are made here as to the "best" choice of $\boldsymbol{x}^p$ for any given surface or application. However, several properties of $\boldsymbol{x}^p$ can simplify the numerical results introduced so far.

Most importantly is the notion of an orthogonal surface parameterization, one in which $\boldsymbol{x}_s^p \cdot \boldsymbol{x}_y^p = 0$. In this case, $\mathbf{I}$ is diagonal and $\theta^p = 0$. This simplifies computation and inversion of both. Furthermore Equation 2.20 for $\dot{\theta}^s$ has a special interpretation in terms of the geodesic curvature of the surface, remarked upon in Appendix B.

In the case of a planar surface (though not necessarily flat), we have $\mathbf{II} = 0$, trivializing all angular velocity expressions.

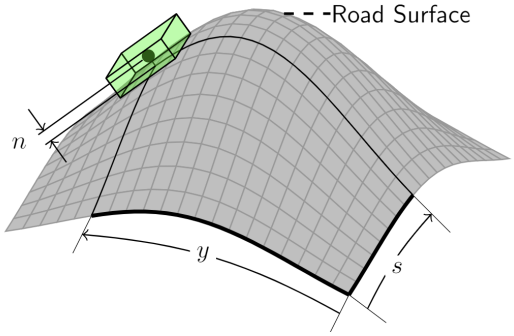
### 2.3.1 Interpretation and Choice of Surface

There is further flexibility in how the parametric surface $\boldsymbol{x}^p$ is interpreted. An obvious choice of $\boldsymbol{x}^p$ is the road surface that is driven over. However, one might also choose $\boldsymbol{x}^p$ to be a surface which always contains a known reference location on a vehicle. The former has the advantage that is can be directly measured in the real world. However, the latter may be found in a data-driven manner, especially in iterative settings. Figure 2.2 illustrates the two, where the main difference is the introduction of the variable $n$ to allow for a normal offset from the surface in the road-surface case. For the purposes of derivation, we will always assume $n \neq 0$ and that $n$ may not be constant, with those cases emergent from simplifying the stated results.

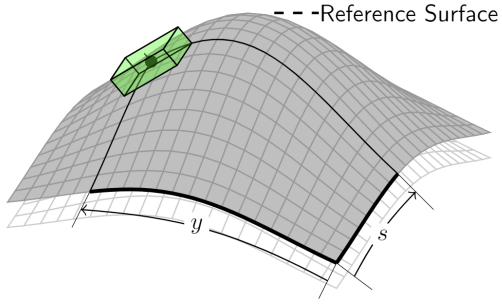
### 2.3.2 Orientation Symmetry and Elevation Maps

In both surface motion models we referenced $\boldsymbol{x}_s^p$ when using surface coordinates to establish a notion of relative heading. We could just as well have chosen $\boldsymbol{x}_y^p$. When $\boldsymbol{x}_s^p$ and $\boldsymbol{x}_y^p$ are orthogonal there is an obvious link and corresponding symmetry between choosing one or the other. However, when they are not orthogonal, and more generally $\mathbf{R}^{pb}$, may become asymmetric with respect to other notions of orientation.

This can be seen in the case of an elevation map: $\boldsymbol{x}^p = s\boldsymbol{e}_1^g + y\boldsymbol{e}_2^g + h(s,y)\boldsymbol{e}_3^g$. In the case of an elevation map, it is both possible and often desirable to describe the heading of a vehicle with its yaw angle relative to a Euclidean reference frame. For instance, consider




(a) Illustration road-surface interpretations of $\boldsymbol{x}^p$ with the vehicle at a normal offset $n$ from the surface.




(b) Illustration of reference-surface interpretation $\boldsymbol{x}^p$. There is no normal offset $n$ but $\boldsymbol{x}^p$ is no longer the physical road.

Figure 2.2: Road and reference surface interpretations of $\boldsymbol{x}^p$.

the heading angle $\phi$ defined by:

$$\cos\phi = \boldsymbol{e}_1^b \cdot \boldsymbol{e}_1^g \qquad\qquad \sin\phi = \boldsymbol{e}_1^b \cdot \boldsymbol{e}_2^g \tag{2.52}$$

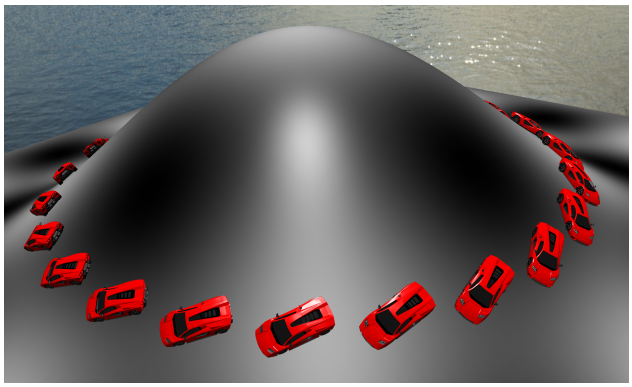
If we take an elevation map which is radially symmetric about the vertical axis through $s = y = 0$, place a vehicle on it with a given $\phi$, and then rotate the vehicle about the surface axis of symmetry we would see that $\phi$ changes with 1:1 correspondence with the rotation. However, the same cannot be said for $\theta^s$. $\theta^s$ can change nonlinearly as the angle between $\boldsymbol{x}_s^p$ and $\boldsymbol{x}_y^p$ changes. For example, Figure 2.3 was produced by rotating a vehicle about a radially symmetric surface with $\phi$ rotated as well. As seen, $\theta^s$ undergoes a periodic, nonlinear distortion, which results from the shape of the surface and the non-orthogonal surface parameterization.

## 2.4 Vehicle Considerations

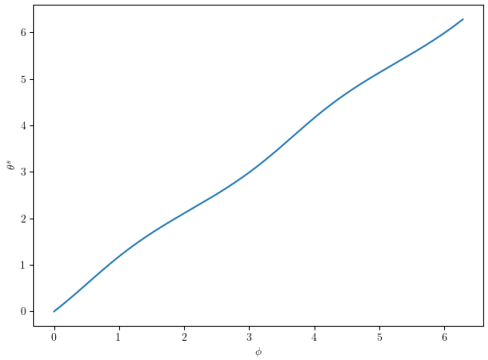
The road considerations developed so far are not tied to any specific type of vehicle, so long as the kinematic assumptions on how the vehicle moves are realistic. In this section we discuss application considerations that hold for some of the most common vehicles that may be considered, and remark upon how these considerations may impact other vehicles as well.

### 2.4.1 Contact with the Road, and Loss Thereof

Unlike flat surfaces, non-Euclidean ones can very well cause a vehicle to lose contact with the road, not just by rolling or pitching over, but by simply going airborne. It may then be problematic that in Section 2.1 we assumed a vehicle remains on the surface and tangent to it. This means that such a model will never leave the surface or roll or pitch, even if the



(a) Radially symmetric surface with several vehicles rotated about the axis of symmetry.




(b) $\theta^s$ and $\phi$ as the vehicle is rotated by angle $\phi$ about the axis of symmetry.

Figure 2.3: Variation of $\theta^s$ as a vehicle is rotated about a radially-symmetric surface parameterized as an elevation map. As shown on the left, the vehicle is rotated with apparently continuous yaw angle $\phi$; due to the coordination of the surface the angle $\theta^s$ changes nonuniformly as shown on the right.

real vehicle would. However, these limits can be modeled and subsequently controlled to be avoided in practice, which is a major control application of such models.

This is most evident for a four-wheeled vehicle, where we can use the constrained normal, roll, and pitch motion of a tangent contact vehicle to model the net force and moment that act in these directions, which is discussed in Section 2.4.3. From these, we can model the normal force on all four tires. Avoiding loss of contact with the road then amounts to keeping these normal forces positive, as any vehicle motion that would result in negative normal force on one or more tires would clearly result in the real vehicle losing contact with the road.

This requirement of road contact is less applicable to the unconstrained model of Section 2.2. As a matter of fact, Section 2.2 can be used to build vehicle models that allow the vehicle to go airborne, such as during a jump or other stunts. It may also be applied to vehicles that remain airborne all the time, though the notion of a surface in such cases is likely to be useful only at low altitude; a better approach for aircraft is developed in Section 3.

### 2.4.2 Gravity

One of the first impacts of driving on a non-Euclidean surface is that the components of gravity are not constant in the body frame. This has the greatest impact on dynamic models, but should be considered by all models for physically realistic behavior[7]. In Section 2.1.1 we introduced equations that allow $\boldsymbol{e}^b_{1,2,3}$ to be computed from $s, y, \theta^s$. These expressions are a

[7]Unless of course, they are meant for a zero-gravity environment.

function of the parametric surface, and results were left in terms of $\boldsymbol{x}_s^p$, $\boldsymbol{x}_y^p$, and $\boldsymbol{e}_n^p$. From these it is straightforwards to find the body frame components; for instance if gravity is $\boldsymbol{g}$, then the $\boldsymbol{e}_{1,2}^b$ components of gravity are simply:

$$\begin{bmatrix} \boldsymbol{e}_1^b \cdot \boldsymbol{g} & \boldsymbol{e}_2^b \cdot \boldsymbol{g} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_s^p \cdot \boldsymbol{g} & \boldsymbol{x}_y^p \cdot \boldsymbol{g} \end{bmatrix} \mathbf{I}^{-1}\mathbf{J} \tag{2.53}$$

This has direct implications on the dynamics and weight distribution of vehicles, which are discussed in following sections.

### 2.4.3 Weight Distribution of Tangent Contact Vehicle Models

Section 1.5.2.3 introduced a method to compute the tire normal forces on a four-wheeled vehicle based on the $\boldsymbol{e}_{1,2}^b$ components of moment and $\boldsymbol{e}_3^b$ component of force on the vehicle which must result from tire normal forces. At the time, it was assumed that $F_3^b$, $K_1^b$, and $K_2^b$ were known, and that other sources of force and moment, such as tires and aerodynamics, were subtracted to obtain the contribution that must arise from tire normal forces.

At the time these three forces and moment were not yet known, however having developed a tangent contact motion model we may now fill them in. To do so, consider the following parts of the Newton-Euler equations:

$$\begin{aligned} F_3^b &= m\dot{v}_3^b + mv_2^b\omega_1^b - mv_1^b\omega_2^b \\ K_1^b &= I_1^{bb}\dot{\omega}_1^b + \left(I_3^{bb} - I_2^{bb}\right)\omega_2^b\omega_3^b \\ K_2^b &= I_2^{bb}\dot{\omega}_2^b + \left(I_1^{bb} - I_3^{bb}\right)\omega_3^b\omega_1^b. \end{aligned}$$

For a tangent contact model, we have either constrained the resulting terms ($v_3^b$, $\omega_1^b$, $\omega_2^b$), or left them as state variables ($v_1^b$, $v_2^b$, $\omega_3^b$), and can simplify these equations. For example, the $F_3^b$ equation simplifies to:

$$F_3^b = m\begin{bmatrix} v_1^b & v_2^b \end{bmatrix} \mathbf{J}^{-1}\mathbf{II}\left(\mathbf{I} - n\mathbf{II}\right)^{-1}\mathbf{J}\begin{bmatrix} v_1^b \\ v_2^b \end{bmatrix}. \tag{2.55}$$

This is a key result - the net normal force on the vehicle described in terms of the curvature of the surface and the speed of the vehicle on the surface. While such effects should have been expected from the start, here they are made precise. Determining the net normal force on a vehicle then amounts to subtracting other forces from the net force, such as gravity and aerodynamic down-force, to obtain the net $\boldsymbol{e}_3^b$ force due to the normal forces of all tires.

Expressions for $K_1^b$ and $K_2^b$ follow similarly for a tangent contact vehicle model but are not expanded here. In summary however, we may model the weight distribution of a vehicle and use it as discussed in Section 1.5.2.3. Numerical implementation of this is discussed in Section 4.4.3.

### 2.4.4 Weight Distribution of Unconstrained Vehicle Models

In Section 2.2, we developed a motion model for a vehicle which may roll pitch and heave relative to the road surface. This concept was introduced earlier in Section 1.5.2.4. The main difference in modeling weight distribution is that the presence of actual roll pitch and heave motion of the vehicle chassis means suspension forces may be modeled directly, such as by spring-damper systems.

The simplest manner in which to do this is to model the displacement of all tires relative to the tangent plane of the surface. This assumes that the curvature of the surface is small relative to the length of a vehicle, and is common practice for control-oriented vehicle models. Computing these displacements is a purely geometric exercise, one that follows from the normal distance $n$ of a vehicle from the road as well as its relative orientation $\mathbf{R}^{pb}$. A similar procedure follows to model the displacement rate of suspension elements, and is discussed later in Section 4.5.1.

## 2.5 Centerline Surfaces

The motion models developed in this chapter have required minimal assumptions on the road surface, and no assumption that the surface have lanes, roads, or similar geometric features. However, these features are common in practice and it is often useful to encode these features in the parameterization of a surface. For example, if the level curves of $\boldsymbol{x}^p$ with constant $y$ correspond to lanes on a road, following a lane is encoded by the surface parameterization as keeping $y$ constant while increasing (or decreasing) $s$.

This section discusses the most common practical form of this: parameterizing a surface $\boldsymbol{x}^p(s, y)$ in terms of a curve $\boldsymbol{x}^c(s)$ with variable $y$ corresponding to some form of distance from the curve. We will refer to these as "centerline" surfaces, with $\boldsymbol{x}^c$ being the centerline, which may correspond physically to the center of a road, a lane on a road, a trajectory which should be followed, or other important construct for control. Examples from existing literature are used to develop this subcategory of surfaces in a step-by-step manner.

### 2.5.1 The Frenet Frame

The Frenet frame is the most widely used form of non-Euclidean geometry to describe vehicle motion in literature. In it, a flat surface is parameterized with a curve $\boldsymbol{x}^c(s)$, assumed to be parameterized by its arc length, with lateral distance $y$ from the curve, illustrated in Figure 2.4. This is an intuitive frame of reference to use wherein the tracking error $y$ and $\theta^s$ are considered directly, and their equations of motion allow for synthesis of vehicle controllers.

Differential geometry analysis of the Frenet frame, and many other centerline surfaces, follows from considering $\boldsymbol{x}^p$ in the form:

$$\boldsymbol{x}^p = \boldsymbol{x}^c + y\boldsymbol{e}_y^c \tag{2.56}$$

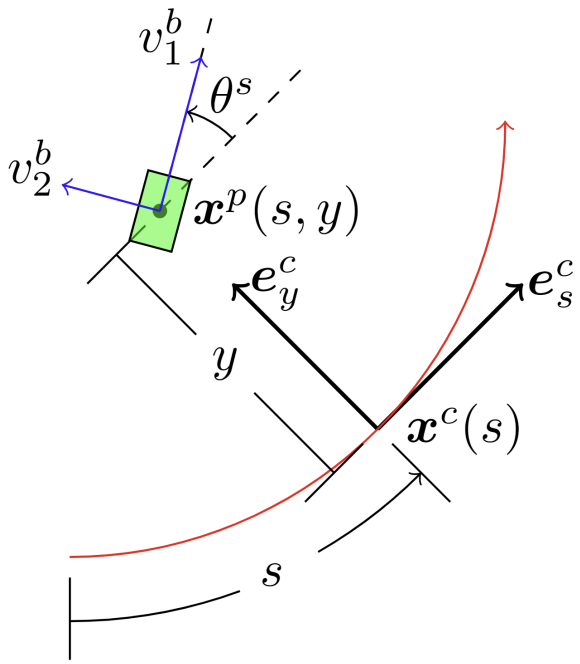



Figure 2.4: Illustration of the Frenet frame, with centerline $\boldsymbol{x}^c$, centerline tangent $\boldsymbol{e}_s^c$ and orthogonal lateral direction $\boldsymbol{e}_y^c$. The green box corresponds to a vehicle, and $\boldsymbol{x}^c$ always lies within a 2D plane.

This form is more generally referred to as a ribbon surface [6, 8], as when $\boldsymbol{x}^c$ and $\boldsymbol{e}_y^c$ are not confined to a plane, the resulting surface has the shape of a ribbon. In the case of the Frenet frame, the fundamental relations needed to use (2.56) are those that relate how $\boldsymbol{e}_s^c$ and $\boldsymbol{e}_y^c$ change along the curve. This is known from the curvature $\kappa$ of the centerline:

$$\partial_s \boldsymbol{e}_s^c = -\kappa \boldsymbol{e}_y^c \tag{2.57}$$
$$\partial_s \boldsymbol{e}_y^c = \kappa \boldsymbol{e}_s^c \tag{2.58}$$

From this, the partial derivatives of $\boldsymbol{x}^p$ are immediate:

$$\boldsymbol{x}_s^p = (1 - y\kappa)\, \boldsymbol{e}_s^c \tag{2.59a}$$
$$\boldsymbol{x}_y^p = \boldsymbol{e}_y^c \tag{2.59b}$$
$$\boldsymbol{x}_{ss}^p = -y\boldsymbol{e}_s^c \partial_s \kappa + (1 - y\kappa)\, \kappa \boldsymbol{e}_y^c \tag{2.59c}$$
$$\boldsymbol{x}_{sy}^p = \boldsymbol{x}_{ys}^p = -\kappa \boldsymbol{e}_s^c \tag{2.59d}$$
$$\boldsymbol{x}_{yy}^p = 0 \tag{2.59e}$$

Application of a tangent contact motion model is straightforwards, with the well-known result that:

$$\dot{s} = \frac{v_1^b \cos(\theta^s) - v_2^b \sin(\theta^s)}{1 - \kappa y} \tag{2.60a}$$
$$\dot{y} = v_1^b \sin(\theta^s) + v_2^b \cos(\theta^s) \tag{2.60b}$$
$$\dot{\theta}^s = \omega_3^b - \kappa \dot{s} \tag{2.60c}$$

This discussion of the Frenet frame has not introduced any new material, and the main contribution here is a strict geometric derivation of these equations, whereas their derivation is often haphazard in literature [21]. However, we have introduced a new geometric perspective of the Frenet frame, one that can be extended naturally to ever more complicated surfaces, a process that is demonstrated next.

### 2.5.2 Ribbon Surfaces

The Frenet frame can be generalized first by allowing $\boldsymbol{x}^c$ and $\boldsymbol{e}_y^c$ to be chosen more arbitrarily in three dimensions. This appeared first in [6] and [8]. In their approaches, the orthonormal basis $\boldsymbol{e}_{s,y,n}^c$ is chosen as a function of $s$, with the curve $\boldsymbol{x}^c$ being the integral of $\boldsymbol{e}_s^c$. The shape of the road surface is then defined by Equation (2.56). Irrespective of how $\boldsymbol{e}_{s,y,n}^c$ are chosen,

the partial derivatives of this surface are:

$$\boldsymbol{x}^p_s = \boldsymbol{e}^c_s + y\partial_s \boldsymbol{e}^c_y \tag{2.61a}$$
$$\boldsymbol{x}^p_y = \boldsymbol{e}^c_y \tag{2.61b}$$
$$\boldsymbol{x}^p_{ss} = \partial_s \boldsymbol{e}^c_s + y\partial^2_s \boldsymbol{e}^c_y \tag{2.61c}$$
$$\boldsymbol{x}^p_{sy} = \boldsymbol{x}^p_{ys} = \partial_s \boldsymbol{e}^c_y \tag{2.61d}$$
$$\boldsymbol{x}^p_{yy} = 0 \tag{2.61e}$$

The basis $\boldsymbol{e}^c_{s,y,n}$ may be chosen directly with Euler angles [22] or implicitly in terms of curvature coefficients with the Darboux frame [6, 8], with the partial derivatives above following from algebra.

It should be pointed out that previous work on ribbon surfaces [6, 8] does not develop ribbon surface motion models using the partial derivatives of a surface. Instead, their approaches are in essence ones which take the Frenet frame equations and add new terms to capture the fact that a three-dimensional road is being considered. This is in some sense the reverse of the process introduced in this chapter: rather than going from a general surface to a specific one, a specific one is taken and new terms are added to try to describe a more general one. This is an error-prone process, particularly when trying to maintain the founding assumptions of a motion model, such as remaining tangent to the surface. Such errors are hard to spot, as they appear primarily in the numerical motion models becoming inaccurate during extreme conditions, whereas they are altogether avoided in the general parametric surface approach presented here.

### 2.5.3 Roads with Curved Cross-Section

Ribbon surfaces are limited in that their cross-section at a given $s$ is a line, whereas roads in practice may have curved cross section. This was first introduced in [7], where the cross-section of a road is allowed to have a curved profile.

We can describe this generally by introducing a new form for $\boldsymbol{x}^p$, one with a second offset in the direction $\boldsymbol{e}^c_n$:

$$\boldsymbol{x}^p = \boldsymbol{x}^c + y\boldsymbol{e}^c_y + h(s,y)\boldsymbol{e}^c_n \tag{2.62}$$

This form places no restrictions upon $h$ apart from it being differentiable. For instance, the partial derivatives of this surface are:

$$\boldsymbol{x}^p_s = \boldsymbol{e}^c_s + y\partial_s \boldsymbol{e}^c_y + (\partial_s h)\,\boldsymbol{e}^c_n + h\partial_s \boldsymbol{e}^c_n \tag{2.63a}$$
$$\boldsymbol{x}^p_y = \boldsymbol{e}^c_y + (\partial_y h)\,\boldsymbol{e}^c_n \tag{2.63b}$$
$$\boldsymbol{x}^p_{ss} = \partial_s \boldsymbol{e}^c_s + y\partial^2_s \boldsymbol{e}^c_y + \left(\partial^2_s h\right)\boldsymbol{e}^c_n + 2\,(\partial_s h)\,(\partial_s \boldsymbol{e}^c_n) + h\partial^2_s \boldsymbol{e}^c_n \tag{2.63c}$$
$$\boldsymbol{x}^p_{sy} = \boldsymbol{x}^p_{ys} = \partial_s \boldsymbol{e}^c_y + (\partial_y h)\,(\partial_s \boldsymbol{e}^c_n) + (\partial_s \partial_y h)\,\boldsymbol{e}^c_n \tag{2.63d}$$
$$\boldsymbol{x}^p_{yy} = \left(\partial^2_y h\right)\boldsymbol{e}^c_n \tag{2.63e}$$

These expressions are lengthy, and it is no surprise that [7] used small-angle approximations to simply their approach. Chapter 4 develops a largely automated procedure to manage this and other sources of complexity when working with surface motion models.

## 2.6 Concluding Remarks on Surfaces

At this point it may appear that there are numerous challenges to practical application of three-dimensional surface parameterizations and corresponding motion models. For example, Equation (2.63) contains numerous coefficients which themselves need to be expanded to determine the partial derivatives of a parametric surface. These derivatives enter into ever more complicated expressions for a chosen motion model, and this complexity is repeated every time a new surface parameterization is chosen.

Managing this complexity has been one of the major shortcomings of previous motion models. Despite the generality of the motion model in [7], it is perhaps unsurprising that later work [23] has still often assumed linear road cross-sections. This stems from the lack of a systematic manner in which to use and apply their motion model. This chapter has presented motion models in a systematic and general manner by treating a surface in a general and flexible manner, and deriving motion models in the same general context.

This may be appealing and practical in a purely theoretical sense, but numerous practical implementation challenges have not yet been addressed. Perhaps most important of all, the amount of algebra that is required to develop a motion model for a given surface has grown and grown throughout this entire chapter.

This problem however has a convenient solution. Due to the generality of this chapter's parametric surface motion model, a software pipeline can be set up to derive motion models symbolically on a computer in a fraction of a second, as opposed to the error-prone and inevitably futile task of deriving motion models on more and more surfaces by hand.

This approach is introduced in Chapter 4, but has already had a number of consequences in the delivery of this dissertation. For one, at no point ever has or will a vehicle model be written out in totality as a set of equations that may be transcribed to computer code and run. This is a substantial different from existing literature, wherein it is common to write out six or more equations which fully describe a vehicle model [2].

This difference in exposition has been made because the aspects that model a vehicle and those that model a surface are two fundamentally different components: we should be able to change the surface and preserve the assumptions made to model the vehicle. Delivering several pages of equations that model a four-wheeled car driving on a particular three-dimensional surface would be contrary to the goals of this work, and still leave the reader with the all important question of how to use this for their own surface, or perhaps their own vehicle. At this point these questions should appear daunting and formidable, but it will be because of this abstraction that the algebraic implementation work is offloaded to a computer in Chapter 4.

# Chapter 3
# Path-Parametric Modeling of an Airborne Vehicle

**Acknowledgement:** Parts of this chapter appeared previously in [9].

While the previous chapter developed models appropriate for both ground and air vehicles, the need for a surface can be inconvenient for aircraft. For example, interactions with a surface are important to model when taking off or landing, but are hardly present for most operational maneuvers.

This chapter develops a second class of non-Euclidean motion models: ones which reference a spatial curve rather than a spatial surface. These motion models will be similar in nature to the centerline surface of Section 2.5 and will have notational overlap concerning the centerline $\boldsymbol{x}^c$. The key geometric difference is that the $y$ and $n$ coordinates will be Euclidean at fixed $s$. This is in contrast to surfaces where even if cross-sections at fixed $s$ are linear, the $n$ coordinate is always normal to the surface, the direction of which changes for any centerline in torsion. In essence, the main difference between this chapter and the unconstrained motion model of Section 2.2 applied to a centerline surface is the $n$ coordinate will point in a fixed direction at any $s$ coordinate.

Note that this chapter's motion model does not change the dynamics of the aircraft being modeled and may be viewed entirely as a change of coordinate system for how pose is expressed. The same was true for the unconstrained surface motion model in Section 2.2, with the key difference being the use of a surface or a curve. As before, this chapter's motion model may be used to simplify control tasks, such as following a route, passing through a gate, and avoiding obstacles, which appears in Chapter 6.

## 3.1 Path-Parametric Motion Model

Geometry for this chapter's motion model is shown in Figure 3.1, where we maintain the notion of a global frame $\boldsymbol{e}^g_{1,2,3}$ and a body frame $\boldsymbol{e}^b_{1,2,3}$, but have dispensed with the parametric surface $\boldsymbol{x}^p$. Instead, we have introduced a parametric curve $\boldsymbol{x}^c(s)$, and equipped it with an orthonormal basis $\boldsymbol{e}^c_{s,y,n}$, with the $\boldsymbol{e}^c_s$ vector tangent to the curve. This is deceptively similar to using a parametric surface $\boldsymbol{x}^c(s)+y\boldsymbol{e}^c_y(s)$, with the important difference that with surfaces the $n$ coordinate was normal to the surface, which could vary along both $s$ and $y$. Here, the $n$ coordinate is along $\boldsymbol{e}^c_n$, which is a function of $s$ only. Notice that we have chosen both the curve $\boldsymbol{x}^c$ and some means of specifying the directions $\boldsymbol{e}^c_y$ and $\boldsymbol{e}^c_n$, as there is a degree of freedom which allows those two vectors to be rotated about the centerline. This additional degree of freedom will be determined shortly.

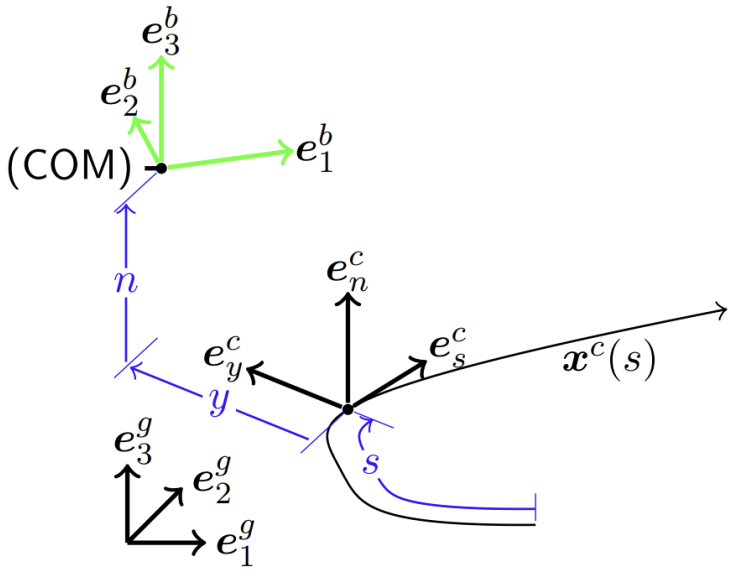



Figure 3.1: Path-parametric coordinate system for aircraft. Adapted from [9].

### 3.1.1 Zero-Order Analysis

Zero-order analysis determines non-Euclidean versions of position and relative orientation in this motion model. Mathematically, position in this motion model is described as:

$$\boldsymbol{x} = \boldsymbol{x}^c(s) + y\boldsymbol{e}_y^c(s) + n\boldsymbol{e}_n^c(s), \tag{3.1}$$

To facilitate practical use we will not assume that $\boldsymbol{x}^c(s)$ is an arc length parameterization, and we will only assume knowledge of $\boldsymbol{r}^c(s)$: a vector close to $\boldsymbol{e}_y^c$ and within the $\boldsymbol{e}_s^c - \boldsymbol{e}_y^c$ plane, but which may not be orthonormal with respect to $\boldsymbol{e}_s^c$. To proceed with analysis it is necessary to orthonormalize $\boldsymbol{x}_s^c$ and $\boldsymbol{r}^c$, and then obtain $\boldsymbol{e}_n^c$ orthonormal to the two:

$$\boldsymbol{e}_s^c = \frac{\boldsymbol{x}_s^c}{\|\boldsymbol{x}_s^c\|} \tag{3.2a}$$

$$\boldsymbol{e}_y^c = \frac{\boldsymbol{r}^c - \boldsymbol{e}_s^c\left(\boldsymbol{e}_s^c \cdot \boldsymbol{r}^c\right)}{\boldsymbol{r}^c - \boldsymbol{e}_s^c\left(\boldsymbol{e}_s^c \cdot \boldsymbol{r}^c\right)} \tag{3.2b}$$

$$\boldsymbol{e}_n^c = \boldsymbol{e}_s^c \times \boldsymbol{e}_y^c. \tag{3.2c}$$

This is sufficient to define the path-parametric motion model to zero-order effects: we have fully determined its shape, and done so in a manner which is general and easy to apply to known geometric environments. As was the case with surfaces, $s$, $y$, and $n$ can be converted to Euclidean position, and vice-versa. The former follows from (3.1) whereas for the latter it is straightforwards to show that $\boldsymbol{x}^c(s)$ is locally the closest point along the centerline to a known position $\boldsymbol{x}$ with with the same $s$ coordinate. As a result one can convert Euclidean

position to non-Euclidean position by first finding $s$ using an optimization problem or a grid of points along the centerline. Once $s$ is known, $y$ and $n$ may be found from the $\boldsymbol{e}_y^c$ and $\boldsymbol{e}_n^c$ components of position relative to the centerline.

As was the case for the unconstrained surface motion model in Section 2.2, it is possible to use this section's motion model with vehicle orientation described relative to a fixed global frame of reference, or relative to the non-Euclidean reference frame and basis vectors $\boldsymbol{e}_{s,y,n}^c$. This is entirely optional in practice. The difference is purely a rotation from a zero-order perspective and involves standard mathematics [13]. From a practical perspective, describing orientation relative to an inertial frame is standard, but describing it relative to the non-Euclidean frame of reference may be useful for tracking a trajectory which includes a reference orientation.

### 3.1.2 First-Order Analysis

First order analysis determines how non-Euclidean position and orientation change over time due to both vehicle motion and chosen geometry. No second-order analysis will be necessary for the same reasons as in Section 2.2.

#### 3.1.2.1 Position

For first-order analysis to determine how $s$, $y$, and $n$ change over time, it is useful to view this motion model as a case of the Darboux frame [18], wherein the standard definition of the frame relates differentials of $\boldsymbol{e}_{s,y,n}^c$ along the curve to one another:

$$\partial_s \boldsymbol{e}_s^c = \kappa_n^c \boldsymbol{e}_y^c + \kappa_y^c \boldsymbol{e}_n^c \tag{3.3a}$$

$$\partial_s \boldsymbol{e}_y^c = \kappa_s^c \boldsymbol{e}_n^c - \kappa_n^c \boldsymbol{e}_s^c \tag{3.3b}$$

$$\partial_s \boldsymbol{e}_n^c = -\kappa_y^c \boldsymbol{e}_s^c - \kappa_s^c \boldsymbol{e}_y^c. \tag{3.3c}$$

Here the $\kappa$ terms are curvature of the moving frame of reference, which in the Darboux frame are assumed to be known. The Darboux frame further assumes that $s$ is an arc-length parameterization. By change of variables, when $s$ is not arc length we have:

$$\partial_t \boldsymbol{e}_s^c = \left(\kappa_n^c \boldsymbol{e}_y^c + \kappa_y^c \boldsymbol{e}_n^c\right) \|\boldsymbol{x}_s^c\| \, \dot{s} \tag{3.4a}$$

$$\partial_t \boldsymbol{e}_y^c = \left(\kappa_s^c \boldsymbol{e}_n^c - \kappa_n^c \boldsymbol{e}_s^c\right) \|\boldsymbol{x}_s^c\| \, \dot{s} \tag{3.4b}$$

$$\partial_t \boldsymbol{e}_n^c = \left(-\kappa_y^c \boldsymbol{e}_s^c - \kappa_s^c \boldsymbol{e}_y^c\right) \|\boldsymbol{x}_s^c\| \, \dot{s}. \tag{3.4c}$$

The above expression may be used to determine $\kappa_s^c$, $\kappa_y^c$, and $\kappa_n^c$ in terms of $\boldsymbol{x}^c$ and $\boldsymbol{r}^c$:

$$\begin{bmatrix} \kappa_y^c \\ \kappa_s^c \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_s^c \cdot \boldsymbol{e}_s^c & \boldsymbol{x}_s^c \cdot \boldsymbol{e}_y^c \\ \boldsymbol{r}^c \cdot \boldsymbol{e}_s^c & \boldsymbol{r}^c \cdot \boldsymbol{e}_y^c \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{e}_n^c \cdot \boldsymbol{x}_{ss}^c \\ \boldsymbol{e}_n^c \cdot \boldsymbol{r}_s^c \end{bmatrix} \frac{1}{\|\boldsymbol{x}_s^c\|} \tag{3.5a}$$

$$\kappa_n^c = -\frac{(\boldsymbol{x}_{ss}^c \times \boldsymbol{x}_s^c) \cdot \boldsymbol{e}_n^c}{\|\boldsymbol{x}_s^c\|^3}. \tag{3.5b}$$

A full derivation is presented in [9], along with the resulting time derivatives of $s$, $y$, and $n$, which complete position kinematics in this frame of reference:

$$\dot{s} = \frac{\boldsymbol{v} \cdot \boldsymbol{e}_s^c}{\left(1 - \kappa_y^c n - \kappa_n^c y\right) \|\boldsymbol{x}_s^c\|} \tag{3.6a}$$

$$\dot{y} = \boldsymbol{v} \cdot \boldsymbol{e}_y^c + n\kappa_s^c \|\boldsymbol{x}_s^c\| \, \dot{s} \tag{3.6b}$$

$$\dot{n} = \boldsymbol{v} \cdot \boldsymbol{e}_n^c - y\kappa_s^c \|\boldsymbol{x}_s^c\| \, \dot{s}. \tag{3.6c}$$

This tells us how to describe motion over time in the non-Euclidean frame of reference. No restrictions have been placed upon our choice of $\boldsymbol{x}^c$ and $\boldsymbol{r}^c$ apart from needing nonzero $\boldsymbol{x}_s^c$ and $\boldsymbol{r}^c$, and needing $\boldsymbol{r}^c$ and $\boldsymbol{x}_s^c$ to not be collinear. These cases are pathological and of little interest when chosen directly. Using equation (3.6) for a vehicle model requires the $\dot{s}$ expression to be nonsingular, this restriction is discussed in Section 3.2

#### 3.1.2.2 Orientation

A second set of kinematic relations may be derived for orientation. These equations are optional in practical use. Vehicle orientation relative to an inertial frame, or relative to the path-parametric frame are equally valid choices when building a vehicle model. Both are useful as the former relates directly to gravity force components in the vehicle body frame while the latter directly tells us the components of velocity needed for (3.6). However, since the two are related by the orientation of the path-parametric frame relative to the global frame, which is known from its definition, either may be used as a vehicle state in a model.

In the case that vehicle orientation state is described relative to the path, the angular velocity of the vehicle relative to the global frame is insufficient to describe how orientation relative to the path changes precisely because the path may curve and bend, changing the basis that vehicle orientation is described relative to. We have already seen this in (3.4), but must relate this to an "effective" angular velocity on the rotation group between the path-parametric frame and the vehicle body frame. Following the convention that $\left(\mathbf{R}^{gb}\right)_{ij} = \boldsymbol{e}_i^g \cdot \boldsymbol{e}_j^b$ and defining $\boldsymbol{\eta}^c$ to be the angular velocity of the path-parametric frame relative to the global frame, we have:

$$\mathbf{R}^{gb} = \mathbf{R}^{gc}\mathbf{R}^{cb} \tag{3.7a}$$

$$\partial_t \mathbf{R}^{gb} = \mathbf{R}^{gb}\hat{\boldsymbol{\omega}}^b \tag{3.7b}$$

$$\partial_t \mathbf{R}^{gc} = \mathbf{R}^{gc}\hat{\boldsymbol{\eta}}^c \tag{3.7c}$$

$$\boldsymbol{\eta}^c = \left(\kappa_s^c \boldsymbol{e}_s^c - \kappa_y^c \boldsymbol{e}_y^c + \kappa_n^c \boldsymbol{e}_n^c\right) \|\boldsymbol{x}_s^c\| \, \dot{s}. \tag{3.7d}$$

Here $\mathbf{R}^{gb}$ represents the orientation of a vehicle relative to the global frame and $\mathbf{R}^{cb}$ its orientation relative to the path-parametric frame. $\mathbf{R}^{gc}$ is the orientation of the path-parametric frame relative to the global frame, is fully defined by the non-Euclidean geometry chosen,

and links the two possible means of representing vehicle orientation. The time derivative of $\mathbf{R}^{gb}$ follows from standard rotation theory from Section 1.3.3 while the time derivative of $\mathbf{R}^{gc}$ follows from the Darboux frame definition and previous analysis to determine $\kappa$ terms in terms of $\boldsymbol{x}^c$ and $\boldsymbol{r}^c$. To determine the time-derivative of $\mathbf{R}^{cb}$, the goal of this analysis, first we have:

$$\partial_t \mathbf{R}^{gb} = \partial_t \left( \mathbf{R}^{gc} \mathbf{R}^{cb} \right) = \partial_t \mathbf{R}^{gc} \mathbf{R}^{cb} + \mathbf{R}^{gc} \partial_t \mathbf{R}^{cb}. \tag{3.8}$$

Expanding the known derivatives of $\mathbf{R}^{gb}$ and $\mathbf{R}^{gc}$ and rearranging we have:

$$\mathbf{R}^{gc} \mathbf{R}^{cb} \hat{\boldsymbol{\omega}}^b = \mathbf{R}^{gc} \hat{\boldsymbol{\eta}}^c \mathbf{R}^{cb} + \mathbf{R}^{gc} \partial_t \mathbf{R}^{cb} \tag{3.9a}$$

$$\implies \partial_t \mathbf{R}^{cb} = \mathbf{R}^{cb} \left( \hat{\boldsymbol{\omega}}^b - (\mathbf{R}^{cb})^T \hat{\boldsymbol{\eta}}^c \mathbf{R}^{cb} \right). \tag{3.9b}$$

This is equivalent to replacing angular velocity coefficients

$$\begin{bmatrix} \omega_1^b \\ \omega_2^b \\ \omega_3^b \end{bmatrix}, \tag{3.10}$$

with [13]

$$\begin{bmatrix} \omega_1^b \\ \omega_2^b \\ \omega_3^b \end{bmatrix} - \left( \mathbf{R}^{cb} \right)^\top \begin{bmatrix} \eta_s^p \\ \eta_y^p \\ \eta_n^p \end{bmatrix}, \tag{3.11}$$

when used for any parameterization of the SO3 rotation group.

## 3.2 Regularity Limits

The path-parametric approach is inherently limited to where equations of motion are defined. By inspection of (3.6), this is the case when $1 - \kappa_y^c n - \kappa_n^c y$ is nonzero; when it is zero the equation of motion for $\dot{s}$ becomes singular. Enforcing this is necessary for proper numerical resolution of any differential equation built upon this path-parametric model, and may be done using a constraint such as:

$$\kappa_y^c n + \kappa_n^c y \leq \lambda < 1, \tag{3.12}$$

This limit may be viewed geometrically as in Figure 3.2. As the limit enforced by (3.12) is approached, the non-Euclidean coordinate system collapses to a point or a line, meaning that a dimension has been lost. Equivalently, increasing the $s$ coordinate will result in no change in Euclidean position, meaning that the inverse, a change in Euclidean position, will produce a discontinuous change in non-Euclidean position. This is not desirable for building vehicle models which should be smooth and continuous in nature.

Regularity limits are a fundamental limiting factor as to where non-Euclidean coordinates may be used for a vehicle model. For instance, following the illustration in Figure 3.2, a

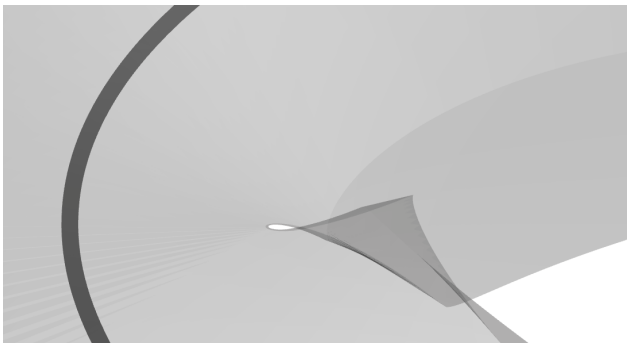


Figure 3.2: Illustration of regularity limits adapted from [9]. A translucent tube has been extruded along the centerline (black line). The tube has been shrunk at places for $\lambda = 0.9$. Noticeably, on the inside of the turn the tube nearly intersects itself, which is where regularity limits are reached and equations of motion become singular. The small opening appears since $\lambda < 1$. Elsewhere the tube overlaps itself. This occurs as regularity limits are only local restrictions, not global restrictions, for instance a figure 8 would overlap as well.

vehicle model build in a Euclidean coordinate system may pass to the left or right of the small opening which illustrates the regularity limit. A non-Euclidean vehicle model however must pass to the left of this opening to ensure that the vehicle model remains continuous and well-defined. Passing to the right of this opening is not possible without additional coordinate systems, for instance a multi-phase problem might switch from one coordinate system to another in the darker regions where coordinates overlap before and after regularity limits become critical. In practical applications, it may be necessary to preprocess the choice of centerline to avoid regularity limits that restrict vehicle motion unnecessarily.

## 3.3 Concluding Remarks on Motion Models

Whereas the previous chapter developed motion models which reference a surface, this one developed a motion model which references a curve. This was motivated primarily by aircraft which often do not interact with a surface while airborne, but for control purposes may interact with a reference curve or trajectory to follow.

Similar to how the parameterization of a surface could be chosen to encode special information in its coordinates, such as the contour of a lane, we can encode special information in the path-parametric motion model. For instance we can choose local $y - n$ coordinates that reference gravity as in [12], or we might choose coordinates that hold special meaning with respect to geometric features such as obstacles and the desired orientation with which the path should be followed. We also developed mathematics to build vehicle models either with orientation relative to a fixed global frame, or relative to the path itself, furthering the ability to encode not only desired position but desired orientation in how non-Euclidean

geometry is set up and then used.

This completes the motion models developed in this dissertation. However, it does not complete our discussion of how they are set up. Section 1.1 introduced all motion models conceptually, and at this point they have been developed mathematically. Equations have been provided for all motion models, but these equations are not necessarily easy to implement and use for computational purposes. As discussed earlier in Section 1.6, using these motion models involves complexity not seen in Euclidean vehicle models. As discussed in the same section, this complexity motivates thinking differently about how vehicle models are derived and implemented. In the next chapter, this perspective is motivated and developed for the surface motion models, with a proposed methodology for practical numerical implementation. Note that while surface motion models are used as an example, the conceptual approach applies equally to the motion model of this chapter, with differences in the geometric inputs and mathematical steps.

# Chapter 4
# Implementation of Parametric Surface Models

The past two chapters introduced mathematics to describe vehicle motion using non-Euclidean geometry. The expressions introduced therein differ from many existing texts on dynamics in that they were never fully expanded. Particularly in the case of parametric surfaces, generic terms such as $\mathbf{I}$ were left in expressions without any regard to a specific surface or parameterization thereof. This has left the question of how to implement these terms and any corresponding vehicle model, unanswered.

This chapter is devoted to answering the question of how to implement non-Euclidean models. Emphasis is placed on the parametric surface models of Chapter 2, but the approach and programming concepts introduced here translate to the aircraft coordinate systems of Chapter 3 with the major change that the fundamental building blocks are $\boldsymbol{x}^c$ and $\boldsymbol{r}^c$ instead of $\boldsymbol{x}^p$. This chapter will follow the systematic approach outlined in Section 1.6, particularly the diagram in Figure 1.8.

Code listing are provided throughout this chapter which illustrate the programming concepts developed here. Full implementations in Python are available online, with links provided at the end of the chapter.

It should be emphasized that the expressions given so far are complete and geometrically correct. The missing elements are twofold: First, *how* is $\boldsymbol{x}^p$ parameterized? One might choose to use an elevation map (A Monge patch in differential geometry speak) for driving in an off-road environment. In an urban driving setting it may be appropriate to use a centerline which follows a given lane with a lateral offset along the lane, similar to Section 2.5. In general, this is a function class for $\boldsymbol{x}^p$ and we will refer to this as the "surface class". The second ingredient is a particular instance of a given surface class, for instance a real offload environment, urban lane, or other mapped geometry. To understand how these factor into a non-Euclidean vehicle model, we start with an example.

## 4.1 Motivating Example: A Road Parameterized by its Boundaries

To illustrate the process used to implement non-Euclidean motion models we consider the following surface parameterization not considered before: Suppose we parameterize a 3D road using its left and right boundaries. We describe this using two interpolation functions $\boldsymbol{x}^l(s)$ and $\boldsymbol{x}^r(s)$ for the left and right boundaries of the track. These may themselves be implemented as cubic splines, Bezier curves, or other smooth interpolation functions, but this will not appear until implementation for a specific instance of this surface class.

For now we consider the surface class itself: $\boldsymbol{x}^p$ described as follows:

$$\boldsymbol{x}^p(s, y) = (1 - y)\boldsymbol{x}^r(s) + y\boldsymbol{x}^l(s). \tag{4.1}$$

Intuitively, we have assumed to know the left and right boundaries of the surface, chosen

interpolating functions for each, and connected the two curves with a line wherever their arguments are identical.

This is a very intuitive choice of surface class, but prior to the developments of Chapter 2 it would have been challenging to develop kinematic equations from (4.1), even if restricted to a flat plane. In the present context however, one starts by computing the partial derivatives of $\boldsymbol{x}^p$:

$$\boldsymbol{x}^p_s = (1-y)\boldsymbol{x}^r_s + y\boldsymbol{x}^l_s \tag{4.2a}$$
$$\boldsymbol{x}^p_y = -\boldsymbol{x}^r + \boldsymbol{x}^l \tag{4.2b}$$
$$\boldsymbol{x}^p_{ss} = (1-y)\boldsymbol{x}^r_{ss} + y\boldsymbol{x}^l_{ss} \tag{4.2c}$$
$$\boldsymbol{x}^p_{sy} = -\boldsymbol{x}^r_s + \boldsymbol{x}^l_s \tag{4.2d}$$
$$\boldsymbol{x}^p_{ys} = -\boldsymbol{x}^r_s + \boldsymbol{x}^l_s \tag{4.2e}$$
$$\boldsymbol{x}^p_{yy} = 0 \tag{4.2f}$$

This is a decent start, but from this point subsequent expressions become ever more complex and tedious to fill in. Consider for instance the first fundamental form:

$$\mathbf{I} = \begin{bmatrix} (1-y)^2\boldsymbol{x}^r_s\cdot\boldsymbol{x}^r_s + 2y(1-y)\boldsymbol{x}^r_s\cdot\boldsymbol{x}^l_s + y^2\boldsymbol{x}^l_s\cdot\boldsymbol{x}^l_s & (y-1)\boldsymbol{x}^r_s\cdot\boldsymbol{x}^r + (1-y)\boldsymbol{x}^r_s\cdot\boldsymbol{x}^l - y\boldsymbol{x}^l_s\cdot\boldsymbol{x}^r + y\boldsymbol{x}^l_s\cdot\boldsymbol{x}^l \\ (y-1)\boldsymbol{x}^r_s\cdot\boldsymbol{x}^r + (1-y)\boldsymbol{x}^r_s\cdot\boldsymbol{x}^l - y\boldsymbol{x}^l_s\cdot\boldsymbol{x}^r + y\boldsymbol{x}^l_s\cdot\boldsymbol{x}^l & \boldsymbol{x}^r\cdot\boldsymbol{x}^r - 2\boldsymbol{x}^r\cdot\boldsymbol{x}^l + \boldsymbol{x}^l\cdot\boldsymbol{x}^l \end{bmatrix} \tag{4.3}$$

Without shrinking the default font size, it would not fit on the page. This rapid growth in algebraic complexity poses severe challenges for implementing motion model terms such as Equation (2.14). Not only does it involve $\mathbf{II}$, which requires us to compute the normal vector of $\boldsymbol{x}^p$, but there is an inverse:

$$\begin{bmatrix}\dot{s}\\ \dot{y}\end{bmatrix} = (\mathbf{I} - n\mathbf{II})^{-1}\,\mathbf{J}\begin{bmatrix}v^b_1\\ v^b_2\end{bmatrix} \qquad\qquad \dot{n} = v^b_3.$$

Even for this seemingly simple choice of surface parameterization we have almost immediately found ourselves with a daunting and tedious problem: performing massive amounts algebra by hand. This is not an impossible task, but it is error-prone and inevitably insufficient: there is no limit to how many different surface classes can be chosen, and we have not yet so much as reached the point of implementation for any actual surface, much less the development of a vehicle model.

The rest of this chapter introduces a computer algebra approach for surface implementation to address this. This provides a general and systematic manner in which to implement parametric surface classes, and to instantiate and use specific instances of a surface class.

Subsequently, we show that computer algebra can go one step further and be used to derive vehicle models. We argue that this is a necessary development to manage the complexity that arises when considering vehicle models, as it allows for separate choice of a surface and a vehicle without insurmountable amounts of hand-calculated algebra.

In what follows, it is assumed that the reader has a basic familiarity with symbolic algebra and automatic differentiation, as can be performed by numerous modern software packages. As much as possible the discussion presented is agnostic to any particular software package. Code examples are presented which use CasADi [24], which is the author's own choice of symbolic algebra software due to its performance and integration of ODE solvers for simulation and optimization routines for model-based control.

## 4.2 Surface Classes and Parameterization Terms

Equation 4.2 has told us everything we need to know to implement a non-Euclidean motion model[1]. With sufficient computer algebra (examples later) all necessary terms can be derived. However it is important to recognize that there are a set of terms in 4.2 which can only be provided by a particular instance of the surface class. In the present example these are:

$$p^s = \left[\boldsymbol{x}^l, \boldsymbol{x}^l_s, \boldsymbol{x}^l_{ss}, \boldsymbol{x}^r, \boldsymbol{x}^r_s, \boldsymbol{x}^r_{ss}\right]. \tag{4.4}$$

While these are known functions of $s$ for a given surface, they are unknown in the general sense. We will refer to these as the parameterization terms of a parametric surface. They are determined symbolically by the choice of a surface class, and numerically by a given instance of that surface class.

It may be tempting to do away with these terms by only ever working with specific instances of a surface class. For instance substituting known functions for $\boldsymbol{x}^l$ and $\boldsymbol{x}^r$ into (4.1) and proceeding. However, this approach is not suitable for cases where the surface changes over time due to measurement updates, and may become intractable for extremely long surfaces where $\boldsymbol{x}^l$ and $\boldsymbol{x}^r$ would have enormous symbolic expressions.

Instead, the approach developed here will handle surfaces in a two stage approach:

First, motion models, and subsequent vehicle models, are derived in an abstract sense for the surface class considered. Properties of the surface, and the resulting vehicle model will depend on $p^s$ in addition to the usual $s, y, n$ for surfaces, and generic state and input for vehicles. For example, with a state $z$ and input $u$ one arrives at an abstract vehicle model in the form:

$$\dot{z} = f_{\text{abs}}(z, u, p^s) \tag{4.5}$$

Second, these abstract models will be filled in for specific surface instances by substitution of the surface parameterization terms. This may be done directly, obtaining a corresponding vehicle model:

$$\dot{z} = f(z, u) \tag{4.6}$$

[1]Higher order derivatives appeared in Section 2.1.3.1 but we omit that motion model for the time being due to its limited usefulness.

## 4.3 Motion model Implementation

In this section, algorithmic steps to automate the development of a motion model are introduced. General outlines are given that hold for any surface class, and examples are presented for the example surface class introduced earlier in the chapter: a road parameterized with curves for its left and right boundaries.

Three main steps are introduced in this section, which completely define a non-Euclidean motion model for the considered surface class. Using a different surface class will amount to changing the first step, with the rest of the steps differences captured by symbolic algebra. This holds true as well for the development of a vehicle model, discussed in the next section. Overall, this section outlines the fundamental aspects of a non-Euclidean vehicle dynamics software stack.

### 4.3.1 Core Surface Class Elements

The most fundamental quantities to determine for any surface class are its parameterization terms and the partial derivatives of the resulting parametric surface. For the boundary-parameterized surface (4.1) these were simple enough to determine by hand: (4.2) and (4.4). In some cases however, computing the partial derivatives may be prohibitively difficult, and automatic differentiation may be employed.

Below is a code listing which demonstrates this: symbolic variables are created for the parametric surface coordinates $s, y, n$, as well as the parameterization terms. $\boldsymbol{x}^p$ has an explicit symbolic expression, and its partial derivatives are found using its partial derivatives with respect to $s$ and $y$. Partial derivatives with respect to $y$ are straightforward while the chain rule is used for $s$ derivatives.

```
import casadi as ca

# parametric variables
s = ca.SX.sym('s')
y = ca.SX.sym('y')
n = ca.SX.sym('n')

# left boundary and its partial derivatives with respect to s
xl   = ca.SX.sym('xl',   3)
dxl  = ca.SX.sym('dxl',  3)
ddxl = ca.SX.sym('ddxl', 3)

# right boundary and its partial derivatives with respect to s
xr   = ca.SX.sym('xr',   3)
dxr  = ca.SX.sym('dxr',  3)
ddxr = ca.SX.sym('ddxr', 3)

# parameterization - dependent terms for symbolic expressions
param_terms = ca.vertcat(xr, dxr, ddxr, xl, dxl, ddxl)
```

```

# parametric surface xp
xp = xr * (1-y) + xl * y

# first derivatives
xps = ca.jacobian(xp, xl) @ dxl + \
      ca.jacobian(xp, xr) @ dxr
xpy = ca.jacobian(xp, y)

# second derivatives
xpss = ca.jacobian(xps, xl) @ dxl + \
       ca.jacobian(xps, dxl) @ ddxl + \
       ca.jacobian(xps, xr) @ dxr + \
       ca.jacobian(xps, dxr) @ ddxr
xpsy = ca.jacobian(xps, y)
xpys = xpsy
xpyy = ca.jacobian(xpy, y)
```

Substituting one surface class for another at this stage is as simple as choosing a different set of parameterization terms and finding new expressions for partial derivatives.

### 4.3.2 General Surface Elements

Next, there are numerous terms which appear in the motion models developed thus far which may now be derived symbolically from the partial derivatives of $\boldsymbol{x}^p$. This is agnostic to the chosen surface class: surface specific details are captured by symbolic algebra and described in terms of the surface coordinates and parameterization terms.

```
# normal vector
xpn = ca.cross(xps, xpy)
xpn = xpn / ca.norm_2(xpn)

# full 3D position
x = xp + n * xpn

# angle between xps and xpy
thp = - ca.arcsin(xps.T @ xpy / ca.norm_2(xps) / ca.norm_2(xpy))

# orthonormalized vectors
eps = xps / ca.norm_2(xps)
xpp = xpy - eps * (xpy.T @ eps)
epp = xpp / ca.norm_2(xpp)
epn = xpn
Rp = ca.horzcat(eps, epp, epn)

# first fundamental form of parametric surface
one = ca.vertcat(
    ca.horzcat(xps.T @ xps, xps.T @ xpy),
```

```
    ca.horzcat(xps.T @ xpy, xpy.T @ xpy)
)

# second fundamental form of parametric surface
two = ca.vertcat(
    ca.horzcat(xpss.T @ xpn, xpsy.T @ xpn),
    ca.horzcat(xpys.T @ xpn, xpyy.T @ xpn)
)

# coefs for angular velocity about xpn from s and y change
ws = (ca.cross(xpss, xps).T @ xpn) / (xps.T @ xps)
wy = (ca.cross(xpsy, xps).T @ xpn) / (xps.T @ xps)

# QR factorization of J
Q = ca.vertcat(
    ca.horzcat(ca.norm_2(xps), 0 ),
    ca.horzcat(-ca.sin(thp)*ca.norm_2(xpy), ca.cos(thp) * ca.norm_2(xpy))
)

# tangent contact motion model terms:
ths = ca.SX.sym('ths')
J = ca.vertcat(ca.horzcat(ca.cos(ths)       * ca.sqrt(xps.T @ xps),
                          -ca.sin(ths)      * ca.sqrt(xps.T @ xps)),
               ca.horzcat(ca.sin(ths - thp) * ca.sqrt(xpy.T @ xpy),
                           ca.cos(ths - thp) * ca.sqrt(xpy.T @ xpy)))
R = ca.horzcat(ca.horzcat(xps, xpy) @ (ca.inv(one) @ J), xpn)
```

### 4.3.3 Evaluation of Symbolic Motion Model

The outcome of a motion model is to determine pose equations for motion of a vehicle, as well as constraints on its motion. In the case of a tangent contact motion model, this means determining $\dot{s}$, $\dot{y}$, and $\dot{\theta}^s$, as well as finding expressions for $\omega_1^b$ and $\omega_2^b$. These equations were introduced in Section 2.1, and are implemented here:

```
# inputs
vb1 # longitudinal vehicle velocity
vb2 # lateral vehicle velocity
wb3 # yaw angular velocity
n   # constant value for n


# parametric position derivative
dsdy = ca.inv(one - n * two) @ J @ ca.vertcat(vb1,vb2)
s_dot = dsdy[0]
y_dot = dsdy[1]

# parametric orientation derivative
ths_dot = wb3 + ws * s_dot + wy * y_dot
```

```
15
16 # constrained angular velocity terms
17 w2w1 = ca.inv(J) @ two @ dsdy
18 wb2 = -w2w1[0]
19 wb1 = w2w1[1]
20
21 # outputs
22 s_dot
23 y_dot
24 ths_dot
25 wb1
26 wb2
```

Evaluation of other motion models, such as the unconstrained approach in Section 2.2, follow an identical overall procedure with additional surface terms that may be added to the previous section, as well as additional inputs from the vehicle.

This completes the implementation of a motion model. In the example here, the resulting, necessary components have been determined as a function of $s$, $y$, and the surface class parameterization terms. Using a different surface class in this pipeline is as simple as swapping out the first block with a new choice of surface class. Applying this result for a particular surface in amounts to substituting the parameterization terms for known functions, and is discussed in Section 4.4 when vehicle models are developed.

In practice, the code block above may appear as a function, with the inputs at the top of the listing provided as arguments, and the function called by a vehicle model being built upon the motion model. This process of building a vehicle model upon a symbolic motion model is described in detail in the next section. Before moving on however, we discuss an important class of surfaces: ones whose shape is implicit.

### 4.3.4 Implicit Surfaces

In the context of differential geometry there are many forms of surfaces which are parameterized implicitly, such as the level sets of a function. There exists a particular form of implicit surface which is commonly used in existing vehicle literature: the Frenet frame. The Frenet frame describes a vehicle on a flat surface in a so-called error reference frame: progress $s$ along a known curve and error $y$ corresponding to distance from the curve. Several other examples were given in Section 2.5.

This may be approached in the manner of the previous sections using a spline for a curve with lateral offset direction orthogonal to the tangent of the curve. However, much existing applications of the Frenet frame require an arc-length parameterized curve, which is difficult to obtain in practice. Instead, most formulations are based on the curvature of the curve and its current heading. In this form, the shape of the curve is implicit - it may be found as an integral of the heading angle of the curve but is not known otherwise.

The lack of a closed-form symbolic form for the shape of a surface is not an issue for

applying a motion model. The steps taken so far have relied only on the derivatives of the surface, not the surface itself. For instance, consider the code below, which illustrates a Frenet frame surface (Section 2.5.1) class defined by heading angle $a$ with curvature $\partial_s(a) = k$:

```
s = ca.SX.sym('s')
y = ca.SX.sym('y')
n = ca.SX.sym('n')

a = ca.SX.sym('a')
k  = ca.SX.sym('k')
dk = ca.SX.sym('dk')

# centerline orientation
es = ca.vertcat(ca.cos(a), ca.sin(a), 0)
ey = ca.vertcat(-ca.sin(a), ca.cos(a), 0)
en = ca.vertcat(0, 0, 1)

# partial derivatives of centerline basis vectors
des = ca.jacobian(es, ca.horzcat(a)) @  ca.vertcat(k)
dey = ca.jacobian(ey, ca.horzcat(a)) @  ca.vertcat(k)

ddey = ca.jacobian(dey, ca.horzcat(a)) @ ca.vertcat(k ) + \
       ca.jacobian(dey, ca.horzcat(k )) @ ca.vertcat(dk)

# partial derivatives of parametric surface
xps = es + y * dey
xpy = ey

xpss = ddey*y + des
xpsy = dey
xpys = xpsy
xpyy = ca.SX.zeros(3,1)
```

This code does not have a symbolic expression for $\boldsymbol{x}^p$, something that was present for the earlier surface class example. It is however sufficient to proceed with all the steps up to here. Not knowing the shape of a surface explicitly from its parameterization poses challenges for practical application, as the shape of the road surface is only known via integration. This makes it very challenging to fit such implicitly-defined surfaces to known road geometry, requiring large optimization problems such as in [8]. An important contribution of the motion models in this dissertation is that they can be implemented in a seamless manner for explicit surfaces. For example, we can develop an alternative Frenet frame where the input is not the heading angle $a$ with $s$ restricted to arc length, but an explicit centerline $\boldsymbol{x}^c(s)$ with $s$ some arbitrary parameterization of the curve. For instance, the code listing below is tailored to $\boldsymbol{x}^c$ being a cubic spline:

```
s = ca.SX.sym('s')
y = ca.SX.sym('y')
```

```
n = ca.SX.sym('n')

xc   = ca.SX.sym('xc',   2)
dxc  = ca.SX.sym('dxc',  2)
ddxc = ca.SX.sym('ddxc', 2)
d3xc = ca.SX.sym('d3xc', 2)

# centerline basis vectors
esc = dxc / ca.norm_2(dxc)
eyc = ca.vertcat(-esc[1], esc[0], 0)
eyc = eyc / ca.norm_2(eyc)
enc = ca.vertcat(0, 0, 1)

# surface and its partial derivatives
xp = xc + y * eyc
xps = ca.jacobian(xp, xc) @ dxc \
    + ca.jacobian(xp, dxc) @ ddxc
xpy = ca.jacobian(xp, y)

xpss = ca.jacobian(xps, xc) @ dxc \
    + ca.jacobian(xps, dxc) @ ddxc \
    + ca.jacobian(xps, ddxc) @ d3xc
xpsy = ca.jacobian(xps, y)
xpys = xpsy
xpyy = ca.jacobian(xpy, y)
```

This may only be a matter of convenience for some applications, but generally allows flexibility in the inputs to describe a surface, meaning different techniques and methods can be chained together with more flexibility than otherwise possible.

## 4.4 Vehicle Model Implementation

We have just seen how the choice of a surface class can be implemented and used in a symbolic algebra pipeline to develop a motion model. Importantly, most of this pipeline was agnostic to the surface class being used, making it easy to swap in and out new surface classes. In the present section this pipeline is built upon to develop a vehicle model, now in a manner that is agnostic to the surface entirely: outputs from the motion model are used in a manner which, courtesy of symbolic algebra, will appear entirely independent on the surface itself. This is of course not an indication that the behaviour of the vehicle will be the same on all surfaces, but that the implementation of a vehicle model can be transferred seamlessly from one surface class to the next, and later from one physical surface to another.

We follow through this section with an example vehicle model - a two track dynamic vehicle model with weight distribution over all four wheels in tangent contact with the road surface. As discussed in section 1.5.2.3, the resulting model is a differential algebraic equation. In the present case, we will assume that the front wheels are steered with input

steering angle $\gamma$ and that the slip ratio $\sigma$ of each tire is treated as an input.

With these considerations, we can set up what will become the differential state, input, and algebraic state of the system:

```
#s, y, ths from motion model

# model inputs
gamma = ca.SX.sym('gamma')       # front steering angle
sigma_fr = ca.SX.sym('sigma_fr') # front right slip ratio
sigma_fl = ca.SX.sym('sigma_fl') # front  left slip ratio
sigma_rr = ca.SX.sym('sigma_rr') #  rear right slip ratio
sigma_rl = ca.SX.sym('sigma_rl') #  rear  left slip ratio

# algebraic tire normal force states
Nfr = ca.SX.sym('Nfr')
Nfl = ca.SX.sym('Nfl')
Nrr = ca.SX.sym('Nrr')
Nrl = ca.SX.sym('Nrl')

# differential state vector
z = ca.vertcat(s,y,ths,vb1,vb2,wb3)

# input vector
u = ca.vertcat(gamma,sigma_fr,sigma_fl,sigma_rr,sigma_r)

# algebraic state vector
a = ca.vertcat(Nfr,Nfl,Nrr,Nrl)
```

Completing this vehicle model will amount to finding both a differential equation for $z$ and an algebraic constraint involving $a$. This discussion is split into three steps: pose equations, velocity equations, and weight distribution, which together fully determine the differential algebraic equation.

### 4.4.1 Pose Evolution

We begin by introducing motion model effects to determine how the pose of a vehicle model changes. For a tangent contact model, this is purely an exercise in relating velocity to $\dot{s}, \dot{y}, \dot{\theta}^s$. For example, with a dynamic vehicle model in tangent contact we have:

```
# motion model evaluation
s_dot, y_dot, ths_dot, wb1, wb2 = road_model(vb1, vb2, wb3, n = 0)
```

This fully captures the motion model in a short and manageable amount of code, leaving the complex details of the motion model to underlying motion model code and symbolic algebra.

### 4.4.2 Velocity Evolution

Next, we compute the forces acting upon the vehicle, such as gravity, aerodynamics, and tire forces. These in turn determine the dynamics of the vehicle. Gravity forces follow directly from the orientation of the vehicle, which is known from the motion model. For example for a tangent contact motion model this was introduced in the code listing of Section 4.3.2. Aerodynamic and tire models are broad in nature, and represented with generic functions here. For instance, the position of the tires relative to the vehicle reference location have been omitted in the pseudocode but must be considered in practice.

```
# gravity force from orientation
Fg = - m * g * R[2, :]

# drag force and moment from vehicle velocity
Fd, Kd = aero_model(vb1, vb2)

# tire forces and moments about body from slip angles and ratios
Ft_fr, Kt_fr = tire_model(sigma_fr, Nfr, vb1, vb2, wb1, wb2, wb3)
Ft_fl, Kt_fl = tire_model(sigma_fl, Nfl, vb1, vb2, wb1, wb2, wb3)
Ft_rr, Kt_rr = tire_model(sigma_rr, Nrr, vb1, vb2, wb1, wb2, wb3)
Ft_rl, Kt_rl = tire_model(sigma_rl, Nrl, vb1, vb2, wb1, wb2, wb3)

# net tire force and moment
Ft = Ft_fr + Ft_fl + Ft_rr + Ft_rl
Kt = Kt_fr + Kt_fl + Kt_rr + Kt_rl

# net force and moment
F = Fg + Fd + Ft
K = Kd + Kt

# velocity dynamics
vb1_dot = F[0] / m + wb3 * vb2
vb2_dot = F[1] / m - wb3 * vb1
wb3_dot = (K[3] - (Ib2 - Ib1) * wb1 * wb2) / Ib3
```

Coriolis terms from the surface on the vehicle have entered through the $\omega_1^b$ and $\omega_2^b$ terms above, as they were determined previously by the motion model. These effects appear more prominently in the next sub-section for weight distribution, but their incorporation is as simple as using the expressions for $\omega_1^b$ and $\omega_2^b$ obtained symbolically from the motion model.

### 4.4.3 Weight Distribution of Tangent Models

At the start of this section we introduced variables for the normal force of each tire of the car. These normal forces are however subject to weight distribution effects, governed by the roll, pitch, and heave moments and forces on the vehicle. Mathematics for this were introduced in Section 2.4.3, which here we introduce in the symbolic code pipeline to compute equillibrium normal forces for all four tires.

```
# inputs: inertial parameters and velocities, known from vehicle state or
    motion model

# approximate angular accelerations
dwb1wb2 = J_inv @ two @ ca.inv(one - n * two) @ J @ ca.vertcat(vb1_dot,
    bv2_dot)
wb2_dot = -dwb1wb2[0]
wb1_dot = dwb1wb2[1]

# net forces and moments
Fb3 = m * (vb2*wb1 - vb1*wb2)
Kb1 = Ib1 * wb1_dot + (Ib3-Ib2)*wb2*wb3
Kb2 = Ib2 * wb2_dot + (Ib1-Ib3)*wb3*wb1

# subtract forces and moments due to tire forces (Ft, Kt), drag (Fd, Kd)
    and gravity (Fd) to obtain force/moment from normal forces
Fb3_N = Fb3 - Fd[2] - Fg[2]
Kb1_N = Kb1 - Kd[0] - Kt[0]
Kb2_N = Kb2 - Kd[1] - Kt[1]

# compute Nf, Nr, Delta
Nf = lr / (lr + lf) * Fb3_N - Kb2_N / (lr + lf)
Nr = lf / (lr + lf) * Fb3_n + Kb2_N / (lr + lf)
Delta = Kb1_N / 2 / (tf**2 + tr**2)

# compute tire normal forces
Nfr_eq = Nf/2 - tf * Delta
Nfl_eq = Nf/2 + tf * Delta
Nrr_eq = Nr/2 - tr * Delta
Nrl_eq = Nr/2 + tr * Delta
```

These equilibrium normal forces may be used as constraints on the original, symbolic normal force variables, forming a differential algebraic equation. They may also be filtered, such as by a first order lag, to obtain differential equations for the symbolic normal forces, obtaining an ordinary differential equation for the vehicle.

### 4.4.4 Completing a Vehicle Model

Returning to the tangent contact two track model, at this point, all ingredients of the tangent two track dynamic vehicle model are in place, and we need only concatenate all elements of the differential state derivative $\dot{z}$ and algebraic constraint $g$.

```
# differential state derivative
z_dot = ca.vertcat(s_dot,y_dot,ths_dot,vb1_dot,vb2_dot,wb3_dot)

# algebraic constraint (g=0)
g = ca.vertcat(Nfr - Nfr_eq,Nfl - Nfl_eq,Nrr - Nrr_eq,Nrl - Nrl_eq)
```

This fully determines the equations of motion of the vehicle model, meaning its numerical implementation is complete. However, we have left the effects of a particular surface unmodeled as the surface class parameterization terms $p^s$ remain undetermined. This final step is introduced next.

### 4.4.5 Substitution of Parameterization Terms

One final step to practical use of a vehicle model remains: applying it to an instance of the chosen surface class. This all-important step has intentionally been left towards the very end because it has decoupled the process of building up a vehicle model from any specific surface. This can be seen in that all vehicle-specific parts build on top of the surface class in abstract and symbolic mannerisms. In other words, were the surface class to change, none of the code blocks in Section 4.4 need to change because the surface changes are handled by symbolic algebra.

To use any vehicle model we must first know exactly what the parameterization terms are for a given surface. Continuing the example of a road parameterized by its left and right boundaries from Section 4.3, we must be able to evaluate the parameterization terms of the surface in terms of $s$, and in some cases $y$. For example, suppose functions exist for each element of $p^s$ such that the following code is valid:

```
eval_param_terms = ca.vertcat(
    xr(s),dxr(s),ddxr(s),
    xl(s),dxl(s),ddxl(s),
)
```

Equipped with this knowledge, $p^s$ is determined by the state of the vehicle model, as $s$ is part of the pose variables describing the vehicle. We may then take a vehicle model developed for the surface class in general, and eliminate $p^s$ to obtain a vehicle model on an instance of that surface class. For example, in the code listing below, functions for $\dot{z}$ and $g$ are first found as functions of $[z, u, a, p^s]$. Subsequently, $p^s$ is removed using the known expression for it in terms of $s$, which is then used to obtain functions for $\dot{z}$ and $g$ in terms of only $z$, $u$, and $a$, which encapsulates a vehicle model on the particular surface in question.

```
f_zdot_abs = ca.Function(
    'z_dot_abs',            # name of function
    [z, u, a, param_terms], # inputs
    [z_dot]                 # outputs
)
f_g_abs = ca.Function(
    'g_abs',
    [z, u, a, param_terms],
    [h]
)

z_dot_sub = f_zdot_full(z, u, a, eval_param_terms)
```

```
g_sub = f_g_full(z, u, a, eval_param_terms)

# z_dot_sub now only depends on z, u, and a
f_zdot = ca.Function(
    'z_dot',
    [z, u, a]
    [z_dot_sub]
)
f_g = ca.Function(
    'g',
    [z, u, a],
    [g_sub]
)
```

Finally we have achieved our goal: functions that encompass a vehicle model on a chosen parametric surface class in the general sense, as well as the same vehicle model on a specific surface of that class. This process has been long and laborious relative to existing vehicle models on flat surfaces, but it was not without reason. Most importantly, the choice of a surface has been made independent of the vehicle model. Additional surface classes and corresponding surfaces may be added in a developer-friendly and user-friendly manner, and the same holds true for vehicle models as well.

## 4.5 Advanced Vehicle Implementation

The previous section gave a step-by-step example of setting up a dynamic vehicle model with a tangent contact motion model. In this section, advanced considerations are given for other vehicle models, such as motorcycles and the unconstrained motion model of Section 2.2.

### 4.5.1 Suspension Motion

Suppose for the moment that instead of using the tangent contact motion model we used the unconstrained variant of Section 2.2. Implementing the pose and velocity evolution of this motion model follows from that section in a manner functionally identical to the tangent contact motion model seen in Section 4.3. However, modeling weight distribution is fundamentally different in that suspension transients are present and modeled explicitly, as opposed to the quasi-equilibrium weight distribution models needed for tangent contact models.

As an example of how suspension transients may be implemented, consider building a model wherein spring-damper forces are introduced for the displacement and displacement rate of the four tires on a car. In a simplified model, we will consider displacements with respect to the tangent plane of the road surface. These are determined by the geometric dimensions of the car, its relative orientation $\mathbf{R}^{pb}$, and its normal offset $n$, and may be found symbolically. Furthermore, displacement rates may be found via differentiation of the

displacement, and the two used together in a spring-damper force law. An example code listing is provided below:

```
    # inputs
    p # pose, (s,y,n)
    p_dot # known pose evolution (s_dot, y_dot, n_dot)
    r # variables that parameterize vehicle orientation relative to the surface, such as euler angles or quat
    R_rel # relative orientation matrix, a function of r
    r_dot # time derivative of r from vehicle velocity and surface curvature
    k # spring coefficient
    b # damper coefficient

    # position of tire relative to center of mass when at rest on a flat surface (suspension partially displaced)
    xb = ca.vertcat(lf, tf, -h)

    # suspension displacement at rest
    d = m * g / k / 4

    # position in parametric frame from surface: xb is rotated and normal offset "n" is added
    x_rel = ca.vertcat(0,0,n) + R_rel @ xb

    # suspension displacement
    axial_displacement = x_rel[2] / R_rel[2,2] - d

    # suspension displacement rate
    axial_displacement_rate = \
        ca.jacobian(axial_displacement, p) @ p_dot \
        + ca.jacobian(axial_displacement, r) @ r_dot

    # suspension force - zeroed if negative since that implies road contact is lost
    N = ca.fmax(
        0,
        -k * axial_displacement - b * axial_displacement_rate
    )
```

The resulting normal force is an explicit function of vehicle state: its pose and velocity, as well as the parameterization terms of the surface class. As a result, steady state tire models may be applied directly and used to determine the net force and moment on the car, fulfilling the necessary inputs for vehicle dynamics.

### 4.5.2 Motorcycles

**Acknowledgement:** Parts of this section appeared previously in [16].

Motorcycles are considerably harder to model than four-wheeled vehicles due to camber motion of the motorcycle body, separate motion of the driver, and possibly suspension deformations within the motorcycle body. This is evidently not a vehicle which remains tangent to a road surface, which was the founding basis of developing non-Euclidean motion models for vehicles. However, for motion planning purposes motorcycles are often approximated as a body whose longitudinal axis remains tangent to and in contact with the surface, but which may camber relative to it. This allows us to assign a frame of reference on the motorcycle which does satisfy the tangent contact motion model, which the motorcycle cambers relative to, as was introduced in Section 1.5.3. This is fundamentally different from four-wheeled vehicles in that the $^b$ frame is no longer directly linked to the center of mass of a vehicle, and the overall linear and angular momentum of the vehicle is more complicated to express.

To manage this from an implementation standpoint, we use the following procedure:

1. Apply a tangent contact motion model to the $^b$ frame.
2. Determine the momentum of the motorcycle, which will depend both on the $^b$ frame and on driver motion and motorcycle camber.
3. Differentiate the momentum to find net force and moments.
4. Compute Net force and moment separately from tire forces, gravity, and aerodynamics.
5. Equate the modeled forces and the momentum forces, forming a DAE model of the motorcycle.

While more complicated than the procedure to build a DAE model of a four-wheeled vehicle, this is very similar in nature to it, and is illustrated in several example steps. For this example we assume the driver's effect is to move the center of mass laterally with distance $d$, as was illustrated in Section 1.5.3.

```
    s, y, n                 # from road parameterization
    s_dot, y_dot, ths_dot   # from motion model
    wb1, wb2, vb3           # from motion model

    # velocity of b frame
    vb1 = ca.SX.sym('vb1')
    vb2 = ca.SX.sym('vb2')
    wb3 = ca.SX.sym('vb3')
    vb = ca.vertcat(vb1, vb2, vb3)
    vb1_dot = ca.SX.sym('vb1_dot')
    vb2_dot = ca.SX.sym('vb2_dot')
    vb3_dot = 0
    vb_dot = ca.vertcat(v1_dot, v2_dot, vb3_dot)

    # new variables for camber and driver motion
    c = ca.SX.sym('c') # camber angle
```

```
    c_dot = ca.SX.sym('c_dot')
    c_ddot = ca.SX.sym('c_ddot')
    d = ca.SX.sym('d') # driver motion of COM
    d_dot = ca.SX.sym('d_dot')
    d_ddot = ca.SX.sym('d_ddot')

    # helpers for body frame and motorcycle frame
    eb = ca.SX.sym('eb', 3)
    eb_dot = -hat(ca.vertcat(wb1,wb2,wb3)) @ eb
    R_c = ca.horzcat(
            ca.vertcat(1, 0, 0),
            ca.vertcat(0,ca.cos(c), -ca.sin(c)),
            ca.vertcat(0, ca.sin(c), ca.cos(c)))
    em = R_c.T @ eb

    # position of COM in body frame
    r_com = em[1] * d + em[2] * (h - r)

    # velocity of COM w.r.t. inertial frame
    v_com = vb.T @ eb + \
        ca.jacobian(r_com, eb) @ eb_dot + \
        ca.jacobian(r_com, d) @ d_dot + \
        ca.jacobian(r_com, c) @ c_dot

    # momentum rate of change
    v_com_dot = ca.jacobian(v_com, vb[:2]) @ vb_dot[:2] + \
        ca.jacobian(v_com, wb[2]) @ wb_dot[2] + \
        ca.jacobian(v_com, eb) @ eb_dot + \
        ca.jacobian(v_com, d) @ d_dot + \
        ca.jacobian(v_com, d_dot) @ d_ddot + \
        ca.jacobian(v_com, c) @ c_dot + \
        ca.jacobian(v_com, c_dot) @ c_ddot
    p_dot = m * v_dot

    # force required for conservation of linear momentum
    # splits p_dot into a vector with components for each basis vector
    F_required = ca.jacobian(p_dot, eb).T
```

Numerically, we have derived the net force components on the motorcycle as a function of state and input variables of the motorcycle. Factors from the motion model have been included automatically, and the overall equations of motion of the motorcycle can be obtained automatically by computer algebra. A similar procedure can be set up for the net moment on the vehicle, with the final result that the algebraic constraint on a DAE involves equating forces and moments as modeled on the motorcycle to those obtained above from mechanics.

This process can be extended seamlessly to handle other single-contact systems, particularly spherical robots.

### 4.5.3 Multi-Contact Systems

A variety of systems involve contact at multiple points that cannot be approximated as simply as was done for four-wheeled cars and two-wheeled motorcycles. These include legged robotics, trailers, and trains, among other vehicular systems. These are beyond the current scope of implementation of the motion models in this dissertation, in large part because transferring kinematic assumptions from one body to a neighboring one cannot be done with local surface knowledge alone, such as the tangent plane and curvature. This is left as potential future work and not addressed here.

## 4.6 Concluding Remarks on Implementation

This chapter introduced a numerical process to implement the surface motion models of Chapter 2. Most importantly, this procedure allows one to think about vehicles and surfaces independently, a necessity given the difficulty to derive any single model by hand and the combinatorial explosion of different surfaces, motion models, and vehicles, a fact motivated earlier in Section 1.6.

As promised in the same section, at no point have we written out a vehicle model in full: there are no sets of standalone equations which may be copied and used independently. Rather, this chapter has asked the reader to think differently about how vehicle models are built, with the surface and vehicle being two distinctly modular parts. This is a substantial break from existing literature, one made specifically to aid the user of non-Euclidean motion models.

To aid the reader in using these motion models themselves, and potentially saving them the need to implement them from scratch, implementations of motion models, vehicle models, and several applications thereof are available online using Python as the programming language and CasADi [24] for symbolic algebra.

Ground vehicles on surfaces: github.com/thomasfork/Nonplanar-Vehicle-Control/
Aircraft relative to curves: github.com/thomasfork/aircraft_trajectory_optimization

For readers not interested in Python, CasADi provides interfaces to a number of other languages, allows for models to be transferred from one language to another, and supports C code generation to speed up model evaluation. In short, vehicle models may be derived in Python and transferred to C, C++, or MATLAB. More information on CasADi may be found on their web page web.casadi.org/

In the next part of this dissertation, several key applications are introduced for both ground vehicles and aircraft, as well as potential future research.

# Part II

# Applications

# Chapter 5
# Applications to Ground Vehicles

The first part of this dissertation developed control-oriented vehicle models for operation on non-Euclidean geometry. The second part of this dissertation presents examples of how these models may be used for planning, control, and other common vehicle model applications.

It should be recognized that the applications discussed here have all been the subject of prior research, and that in most cases the core novelty lies in a precise and methodological way to consider more general geometry than before. This is particularly true of ground vehicles, where the key contribution is the ability to apply these control techniques to new operating domains.

In some cases this transition will be seamless, and require no new considerations beyond replacing one vehicle model with another. In others however, new effects will need to be considered, such as completely losing contact at the top of a hill.

In this chapter, examples of several common ground vehicle control problems are used to show how three dimensional roads may be considered using non-Euclidean motion models. Subsequently, Chapter 6 discusses applications to aircraft.

## 5.1 Racing

The need for considering non-Euclidean road geometry in racing should be self-evident: road geometry may allow for faster vehicle operation, or require reduced vehicle speed, depending on the shape of the racetrack. In any of these cases, using a more accurate road model will yield improved results, allow for faster lap times where appropriate, and may even be used to design the shape of the racetrack. Not considering non-Euclidean road geometry will at best leave one at a disadvantage, and at worse result in unsafe vehicle operation or catastrophic loss of control.

Racing is in many ways the ideal target application of the road models developed in Chapter 2. Racetracks are typically free of bumps, potholes, and other geometric features that would invalidate a tangent contact road model. Furthermore, many racecars are designed with hardly any ground clearance, meaning that the behaviour assumed by the tangent contact road model is expected and desired in the first place. These assumptions are of course broken in practice: racecars may crash and sometimes go airborne as a result. This is however an application where vehicles should be controlled to avoid such behaviour. Even though we have made major assumptions in building a vehicle model for control, we can and should control the vehicle so these assumptions remain valid.

### 5.1.1 Raceline Computation

A standard problem across many different forms of racing is to determine the fastest possible lap for a given vehicle. This may be done for any number of reasons, such as to provide a

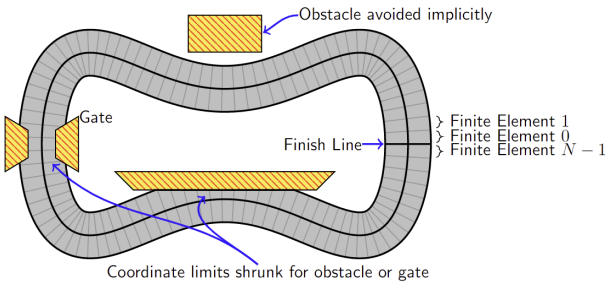



Figure 5.1: Schematic of a racetrack discretized for raceline optimization. Adapted from [9].

reference for a feedback controller to track, to alter design variables of a vehicle to improve its lap time, or to train human drivers. A review of this area may be found in [25]. My own work has focused on the extension of raceline optimization to arbitrary non-Euclidean racetracks with application to cars [26] and motorcycles [16].

Racelines are usually computed by means of large nonlinear optimization programs which solve a problem of the form:

$$\min_{T,x(t),u(t)} T \tag{5.1}$$

$$\text{subject to } \dot{x} = f(x,u) \tag{5.2}$$

$$x \in \mathbb{X} \tag{5.3}$$

$$u \in \mathbb{U} \tag{5.4}$$

$$x(0) = x(T) \tag{5.5}$$

$$u(0) = u(T). \tag{5.6}$$

Here $f$ is an ordinary differential equation which models the dynamics of a vehicle. $x$ and $u$ are the states and inputs of the vehicle model, and the sets $\mathbb{X}$ and $\mathbb{U}$ capture any constraints imposed on the vehicle. The overall goal is to minimize the total elapsed time $T$ to complete a lap, with periodic constraints on $x$ and $u$.

This optimization problem is often implemented by discretizing a racetrack spatially, such that a fixed number of finite elements are associated to the entire length of the racetrack, as shown in Figure 5.1. A differential vehicle model is then solved over each finite element using one of several numerical methods [27], such as orthogonal collocation [28, ch. 10]. Toolboxes exist [29] to aid the setup thereof, or they can be set up from scratch, such as using CasADi [24]. Examples in this chapter were set up using CasADi, with resulting optimization problems solved using IPOPT [30] with the linear solver MUMPS [31].

Discretizing a racetrack in space has several important effects, for instance that the time duration of each finite element must be variable, rendering the overall problem nonconvex. However, spatially-fixed finite elements trivialize the effects of obstacles as shown in Figure 5.1, where racetrack boundaries may be shrunk to avoid collision. This is straightforwards

when using non-Euclidean geometry, as the left and right boundary constraints shown in Figure 5.1 are box constraints on the coordinate system shown in the same figure. Furthermore, as discussed in chapter 4, non-Euclidean vehicle models involve terms from the coordinate system, such as the curvature of the racetrack centerline in Figure 5.1. These terms are variable in general, but for finite elements fixed in space they are constant. As a result, while fixing elements in space has introduced one source of nonconvexity, it removes two sources of nonconvexity which would be hard to control otherwise, and makes the overall problem easier to set up.

The structure of a raceline optimization problem does not need to change to handle non-Euclidean geometry, however several important effects need to be considered, which are discussed next.

Non-Euclidean dynamics are captured by setting up a differential vehicle model, that is they are captured by the function $f$. However, this model will only be valid in some range of operation, for instance when all tire normal forces are positive, that is the vehicle does not go airborne. As a result, it is necessary in all cases to impose constraints on normal forces, or use a road model that allows for airborne behaviour (Section 2.2).

Several other operating limits must be considered as well: road friction, maximum suspension forces, powertrain input and output, and heat dissipation may all prove to be limiting factors in practice, and will result in unrealistic behaviour if not considered. In general, the structure of existing raceline problems is preserved with new models and constraints to reflect a non-Euclidean racetrack.

Examples of racelines computed for several vehicles are shown in Figures 5.2, 5.3, and 5.4, illustrating the variety of vehicles for which racelines may be computed. In Figure 5.2, a raceline is computed with a high fidelity vehicle model, which is recomputed in Figure 5.3 with a simplified point mass model. The two racelines are nearly identical, however the high fidelity vehicle model exhibits drifting behaviour, which emerges from optimization and the vehicle model. The point mass model in Figure 5.3 solves roughly 15 times faster, but misses this behaivour. Figure 5.4 illustrates a motorcycle raceline adapted from [16]. Source code to produce these and other racelines is available at github.com/thomasfork/Nonplanar-Vehicle-Control/

### 5.1.2 Game Theoretic Racing and Enabling Overtaking

It stands to reason that considering non-Euclidean road geometry is not only useful in the offline sense of computing a raceline, but in the online one as well. With the growth of autonomous racing competitions [32, 33] there is ample potential for the use of non-Euclidean road models on racetracks with such road geometry, such as cambered turns. In [26] I illustrated that considering non-Euclidean road geometry gives one an edge over vehicles which approximate all roads as flat, and may enable overtaking maneuvers on suitable road geometry. Future research may leverages these performance improvements for head-to-head racing techniques, such as [34].

Figure 5.2: Raceline computed for a car on a non-Euclidean racetrack, with snapshots shown every tenth of a second. Vehicle dynamics were modeled with a two-track tire layout, combined-slip Pacejcka tire model, and quasi-equillibrium weight distribution with the tangent contact road model. The 3D racecar asset was made by Sketchfab user Lexyc16, is licensed CC BY-NC 4.0, and can be downloaded from skfb.ly/opvyH

Figure 5.3: Raceline computed using a point mass model on the same non-Euclidean racetrack as Figure 5.2, with snapshots shown every tenth of a second. Dynamics were modeled by treating longitudinal and lateral forces as inputs, with a friction cone similar to the macroscopic limits on the dynamic vehicle model of Figure 5.2.

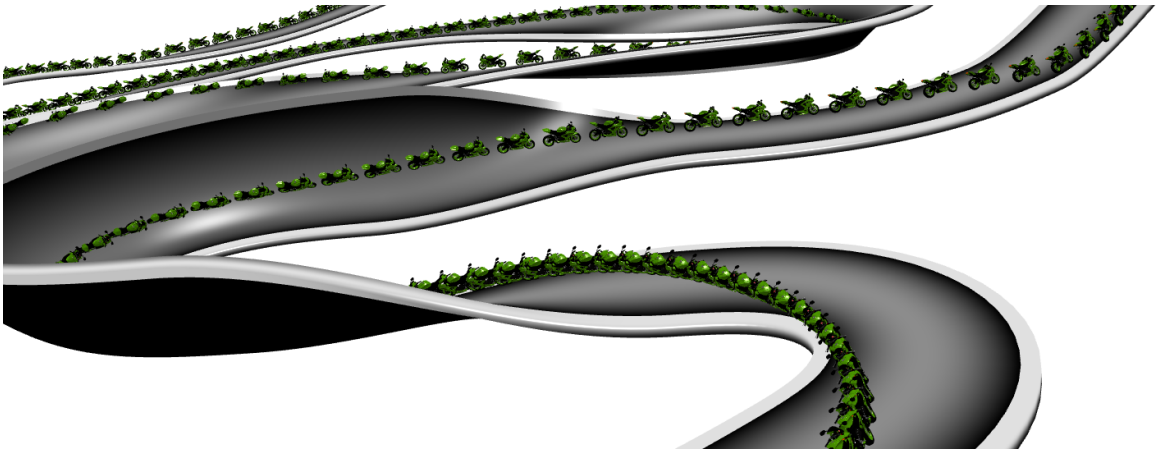


Figure 5.4: Raceline computed for a motorcycle on a non-Euclidean racetrack, with snapshots shown every tenth of a second. Environment and motorcycle model are adapted from [16]. The visual motorcycle model was made by Sketchfab user nouter2077 and is licensed CC BY 4.0 and can be downloaded from skfb.ly/oIYFv.

Below the level of path planning and control, considering road geometry in a realistic sense offers performance improvements to the drivetrain of a vehicle, such as torque vectoring and stability control. These are discussed in section 5.3.

## 5.2 Autonomous Vehicles

One of the main research goals for ground vehicles in the present day is the development of autonomous vehicles. It is inevitable however, that some autonomous vehicles, particularly those being developed for industrial applications such as mining, encounter road geometry which is non-Euclidean in nature. As a result, several current and future possible research avenues are discussed along with the implications of my own work.

### 5.2.1 Motion Planning and Control

Numerous model-based planning and control approaches have been developed for autonomous vehicles. Considerations of road topology are present in many approaches, but are often simplified. For example, [35] only considers changes in gravity due to the slope of a road.

General road models such as developed in this dissertation enable the development of similarly general vehicle models, which may be applied to any number of planning and control problems, including those already in literature. The potential scope of this is incredibly broad, and an example of path-planning for following a road was illustrated in [22].

The key enabling factor to deploy such systems to the real world is estimation of road

parameters in real time, either from a map of road information or live sensory feeds from a vehicle. These remain open problems to solve and are an area of potential future research.

### 5.2.2 Off-Road Route and Path Planning

Enabling mobility on off-road terrain is one of the key areas of application of my work on road models. Whereas prior road models were developed for limited, non-Euclidean scenarios for racing [6, 7], no comparable approach existed for high speed vehicles operating on off-road terrain. This is a key advantage for the road models developed in Chapter 2, as they are general in nature, and can be easily applied to smooth off-road terrain. While rough and rocky surfaces are common in off-road environments, these features typically fall into one of several categories:

1. Small enough that they may be ignored, such as due to being smoothed out due to tire and suspension compliance.
2. Large enough that they must be avoided altogether.
3. Somewhere in the middle, but for the purposes of planning and control they may be treated as disturbances.

Assumptions such as these are common in practice, but contentious in nature, as they are specific to a given vehicle and environment and may vary greatly in practice. Consider however an example problem: an off-road vehicle must maneuver to the top of a sandy hill. Several features are present, such as trees or bushes, which are in effect, obstacles that must be avoided. Along the way, uneven sand causes the vehicle to jostle around unpredictably, however macroscopic behaviour, such as the slope of the hill, can be predicted reliably. Many such settings can be envisioned, such as in automated reconnaissance and monitoring, transportation services for people and goods, and construction. Each setting is bound to require specific considerations, but all share the common theme of regularly interacting with non-Euclidean road surfaces. Given the multitude of methods used for route and path planning, it is impossible to remark upon all the potential applications. Instead, a short example is presented, with remarks on several directions of potential future research.

Suppose we wish to compute a time-optimal route from one point to another point, avoiding obstacles along the way, and coming to a full stop at the end. This is straightforwards to set up as an optimization problem such as in [36], which also discusses how to set up collision avoidance constraints. Here, the key novelty is the ability to consider a surface that is no longer flat, extending the sorts of problems which may be solved. For example, Figure 5.5 shows a trajectory for ascending a hill in the presence of obstacles. Algorithmically, the problem is identical in form to those in [36], yet we have extended it to a completely new operating domain, one which previously would have been intractable.

This generalization is not without its perils. Point-to-point optimization problems are nonconvex and not guaranteed to converge, and allowing the shape of the road surface to

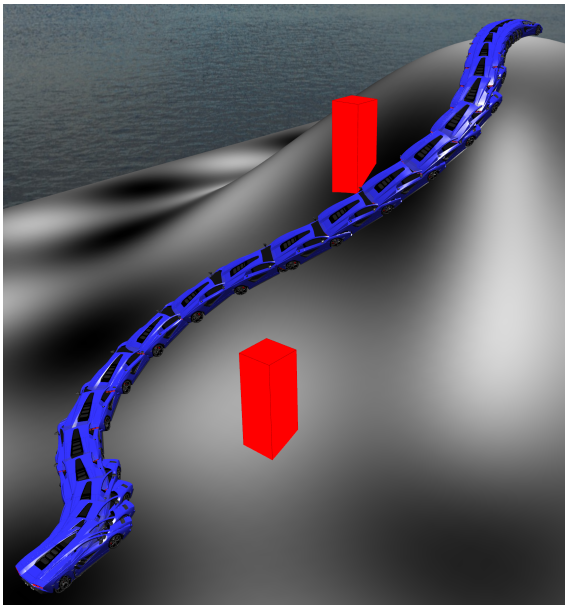


Figure 5.5: Trajectory computed to maneuver to the top of a hill in the presence of two obstacles. Snapshots of the vehicle are shown every third of a second. The trajectory was computed using a kinematic bicycle model subject to friction cone and two-track road contact constraints (the normal force on all four tires must be positive). The 3D racecar asset was made by Sketchfab user Lexyc16, is licensed CC BY-NC 4.0, and can be downloaded from skfb.ly/opvyH

vary leads to many possible scenarios where convergence is suboptimal, or not present at all. This is not a new problem however, and many methods have been proposed in the research community to address them. For instance, search based methods [37] approach multi-modality and nonconvexity by attempting to explore many possible options using a search algorithm. These algorithms may in turn be used to warmstart a local optimization method, such as used in the hill climb example here. All of these are potential applications for non-Euclidean road models and future research.

## 5.3 Vehicle Systems

**Acknowledgement:** Parts of the material presented in this section have appeared in [38].

The applications presented so far have focused on the high-level planning of a vehicle, such as its handling and macroscopic motion. However, the vast majority of vehicle control systems in deployment operate at far lower levels. For instance traction control, torque vectoring, and many other powertrain management technologies operate without knowledge of the road surface they are driving on.

This is a stark contrast to the previous developments of the dissertation, all of which assumed some knowledge of the road surface, which may not be available in practice. This section explores this with the task of brake control on a four-wheeled vehicle, first by approaching what can be done without road knowledge, and then what can be done with road knowledge.

Section 5.3.1 approaches the first half of this problem using the problem of brake control. It will be shown that road knowledge can be obtained implicitly by means of existing sensor technologies, particularly an accelerometer. When properly used this results in near-optimal performance, even without direct knowledge of the road.

Direct knowledge of the road enables more powerful control methods to be developed, particularly predictive control which anticipates future behaviour of a vehicle. This is investigated in Section 5.3.2, where a safety system is developed to anticipate when a vehicle must slow down to avoid future loss of control due to vehicle speed and road geometry, and take appropriate countermeasures.

### 5.3.1 Electronic Brake Force Distribution

Electronic brake force distribution (EBD) is the task of distributing a net force and moment target over all of the brakes of a vehicle. In this context it is assumed that four degrees of freedom are present, corresponding to the four corners of a car. However, the methods introduced here may be similarly applied to a motorcycle, or extended to the additional degrees of freedom of a truck with a trailer.

The key job of EBD is to respect, as much as possible, the operating limits of the corners of the vehicle, particularly the threshold at which tires skid or lock in place. This is because

at these extremes, tire forces decrease, performance is degraded, and in extreme cases a vehicle may spin out of control.

Numerous technologies have been developed to address these following EBD. For instance anti-lock braking systems (ABS) are designed to avoid wheel lock when braking. Traction control (TC) systems avoid excessive wheel skid when accelerating. Finally, electronic stability control (ESC) attempts to correct for a vehicle which is about to spin out of control.

These technologies have proven their effectiveness and are common in modern passenger vehicles, but ultimately raise all manner of performance questions. For instance, ESC was developed in large part to prevent vehicles from spinning out of control during maneuvers that involve a hard turn and hard braking. Turning without braking, or braking without turning, may not result in instability by themselves, making it worth questioning whether or not EBD was responsible for causing vehicle instability.

#### 5.3.1.1 Brake Force Limits

Before investigating this, we will investigate how brake force limits may be estimated in a practical manner suitable for general road geometry. Brake force limits arise from three effects: First, the friction limit between the tire and the road. Second, the normal force on a tire. Third, the lateral tire force present on a tire. All three of these coexist in the form of what is referred to as a friction cone, which captures limits on the maximum longitudinal and lateral tire forces as a function of normal force and friction.

Since braking is purely longitudinal, and cannot directly change lateral tire forces, knowing these three quantities provides us with an estimate of how large a brake force can be applied by a tire, which in turn can be correlated to actuator forces on a brake pad.

Determining these quantities is challenging in practice. We suppose for the moment that an estimate of friction limits is readily available, and consider the task of estimating tire forces. This can be approached using a clever observation: The raw output of an accelerometer is not coordinate acceleration but proper acceleration. Viewed through the lens of Newtonian mechanics, this means that an accelerometer measures every acceleration except that due to gravity:

$$\boldsymbol{a}_{proper} = \frac{\boldsymbol{F}}{m} - \boldsymbol{g}. \tag{5.7}$$

For example, consider a car parked on a flat road. The accelerometers output is $1g$ in the vertical direction, measuring the $1g$ net normal force on the vehicle. This observation is incredibly useful in practice, because the dominant sources of force on a vehicle (in the classical sense) are gravity, tire forces, and aerodynamics. In this sense, we have a measurement of net tire forces with additional effects due to aerodynamics.

This is immediately useful as net tire forces may be used to estimate weight distribution and subsequently brake force limits. Net longitudinal and lateral tire forces produce roll and pitch moments which shift how the net normal force is distributed over all four tires.

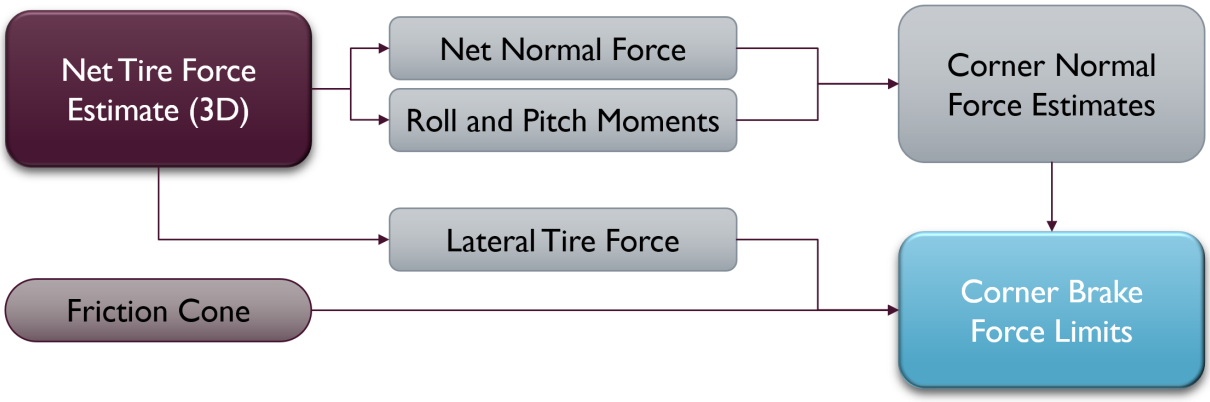



Figure 5.6: Diagram of brake force limit estimation.

Subsequently, the net lateral tire force may be used with a friction cone estimate to estimate the longitudinal brake force limits of all four tires.

This is a straightforwards and appealing process, but has a number of vulnerabilities.

First, a vehicle with a soft suspension will physically roll and pitch, meaning the axes of an accelerometer are no longer aligned with the road. This might be counterbalanced using suspension stiffness factors to estimate and correct for this motion, or instrumented suspension might measure this directly.

Second, aerodynamic forces inherently confound the measurement of tire forces. For instance aerodynamic downforce increases the net normal force, while longitudinal drag increases the tire force necessary to move a vehicle. When these are large they may need to be estimated and corrected for separately, such as aerodynamic measurements in high performance settings, obtaining normal forces from instrumented suspension elements, or estimating longitudinal drag from drivetrain power consumption and vehicle acceleration.

#### 5.3.1.2 EBD Algorithm and Performance

Having estimated the maximum allowable brake force for each corner, a target net force and moment upon the vehicle may be allocated within these limits. This can be done by setting up a small optimization problem to match the target force and moment as closely as possible while respecting constraints upon the for corner brake. Once brake forces are solved for, they may be applied to physical brake actuators. This EBD pipeline is outlined in Figure 5.6 and 5.7, with emphasis on the novel process to estimate brake force limits.

The key advantage of this approach is the reliable estimation of brake force limits. Overestimating brake force limits causes skidding and potential loss of control of a vehicle while underestimating brake force limits causes a vehicle to decelerate much less than possible. This is illustrated in a simulated test in Figure 5.8, which involves impulsive braking of a vehicle on a straight, undulating, downhill road surface.

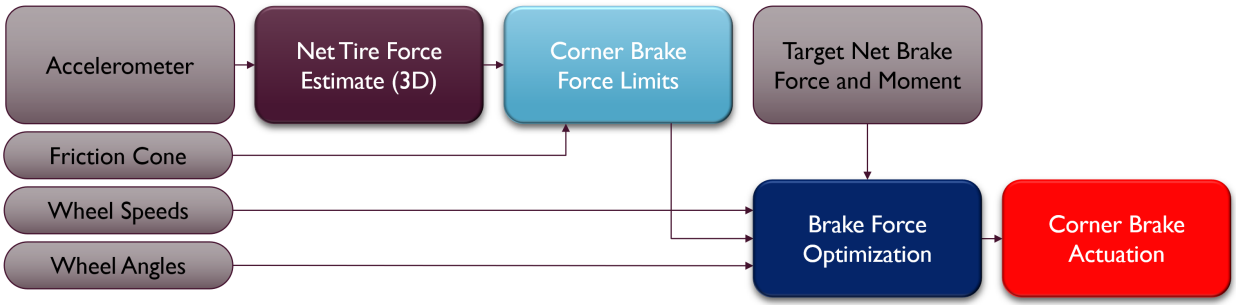



Figure 5.7: Overall diagram of non-Euclidean electronic brake force distribution system.

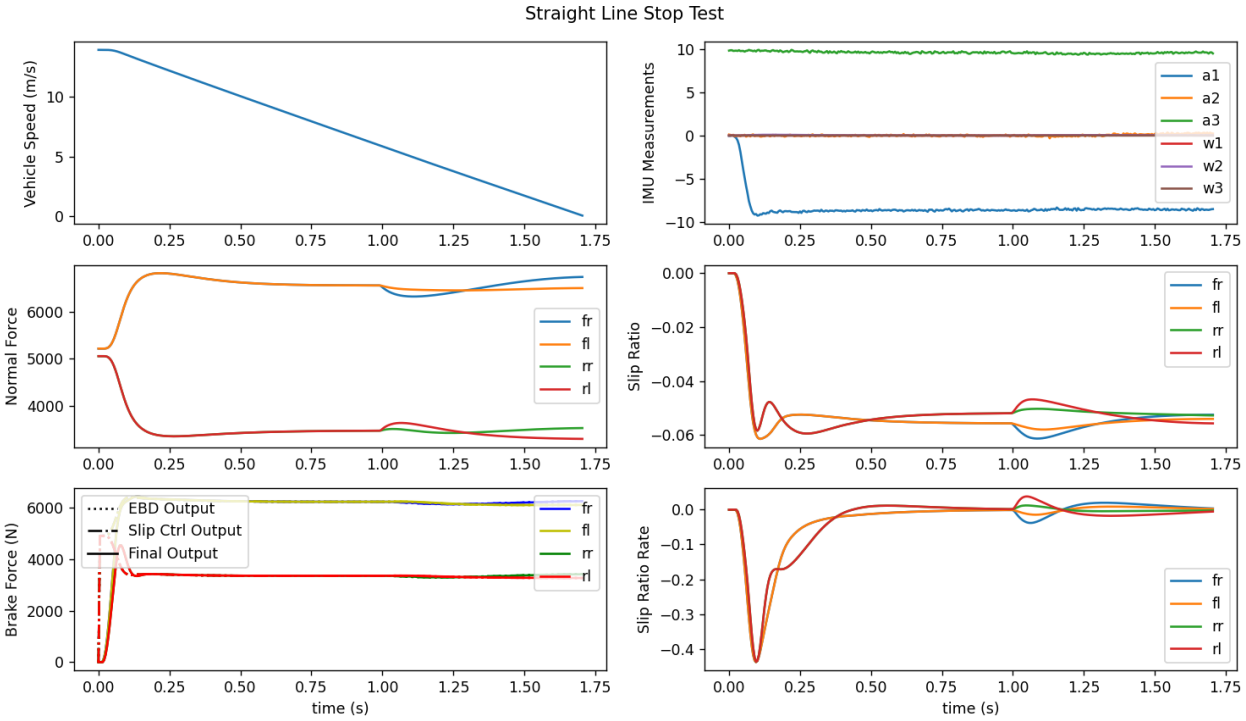



Figure 5.8: Results from a simulated test scenario of the non-Euclidean EBD introduced here. The key outcome, slowing a vehicle to a rapid stop, is accomplished in a safe and controlled manner despite frequent roll and pitch variations in the road surface. This is evidenced by the uniform slip ratios on the four tires on the middle right subplot, which reflect the equal distribution of brake effort over all corners of the vehicle, stopping the vehicle quickly without causing it to skid.

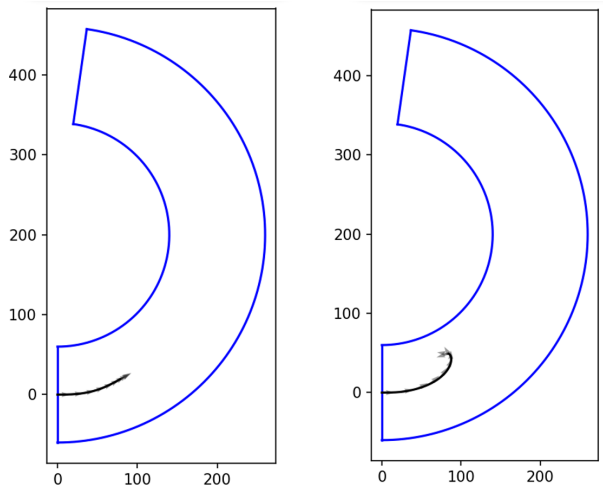



Figure 5.9: non-Euclidean (left) and planar (right) EBD systems during a simulated brake maneuver on a high-speed turn. Arrows indicate the heading of the vehicle, with the right figure illustrating that the vehicle spins out of control during the turn. The non-Euclidean EBD system incorporates inertial weight distribution knowledge, as a result of which brake forces are balanced and the vehicle remains stable. The planar EBD does not, resulting in the vehicle spinning out of control without adding stability control as a secondary control stage.

These EBD performance improvements play an even more profound role in brake control during a hard turn, where lateral tire forces both alter the weight distribution of a vehicle and reduce the maximum possible brake force on each tire. This is a major strength of the accelerometer data interpretation proposed in Section 5.3.1.1: net lateral tire force may be estimated directly, and is subject to little confounding due to aerodynamic forces. The importance of this is illustrated in Figure 5.9, where the non-Euclidean EBD system is tested against a planar version. While the non-Euclidean EBD system maintains stable control of the vehicle during a high-g turn, the planar EBD system loses control as it does not maintain balanced brake forces. Partway through the planar EBD test, ABS engages to avoid wheel lock, which imbalances the brake forces on the vehicle and causes it to spin out. This is the standard reason for stability control, such as ESC. While ESC is still a necessary safety feature for all vehicles, the tendency of the non-Euclidean EBD system to avoid causing vehicle instability is a major performance advantage, illustrated in the stopping distance and oversteering results shown in Figures 5.10 and 5.11.

Beyond improving the distribution of brake force over a vehicle, the consideration of weight distribution and tire operating limits of a vehicle on a real road surface enables a variety of other potential applications and performance improvements.

Firstly, the uniform controlled behaviour of all four tires implies that nonuniform be-

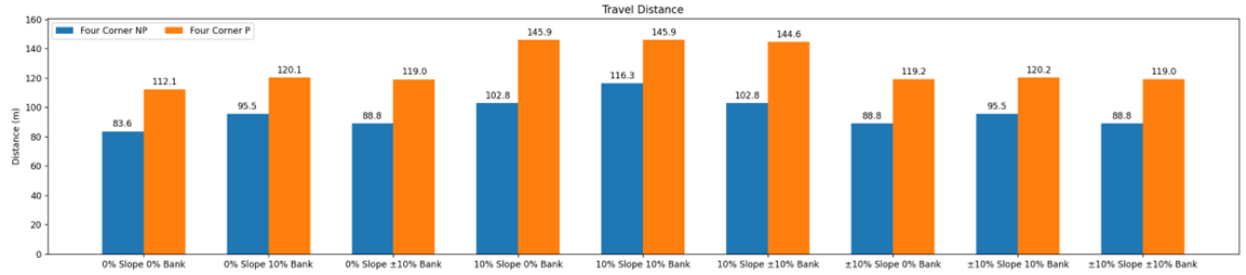



Figure 5.10: Stopping distance comparison for planar and non-Euclidean EBD systems on a high speed cornering test. Horizontal axis labels indicate the slope and bank of the turn, which was changed for different tests.

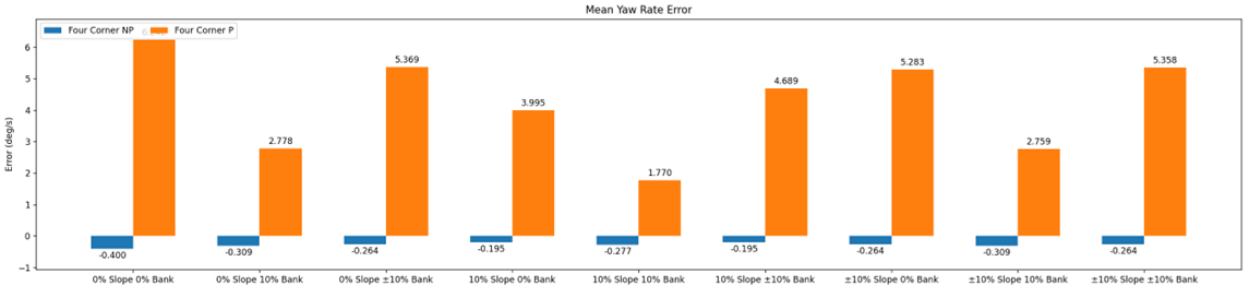



Figure 5.11: Oversteering comparison for planar and non-Euclidean EBD systems on a high speed cornering test. Horizontal axis labels indicate the slope and bank of the turn, which was changed for different tests. Oversteering was measured by comparing the yaw rate of the vehicle to that of a kinematic bicycle, with positive corresponding to a larger yaw rate than the kinematic bicycle.

haviour is a strong indicator of nonuniform conditions, such as changes in road friction or tire condition. These may be used to develop improved methods for estimating road conditions and predicting the need for vehicle service.

Secondly, the improved cornering stability of the non-Euclidean EBD algorithm, and its massive improvement in performance, suggest that vehicle performance and safety may be improved greatly on existing vehicles. This is not limited to braking systems, as similar considerations can be made for torque vectoring of engines and motors.

Thirdly, lower level control systems, such as ABS and ESC, may benefit dramatically from improved estimation of the weight distribution of a vehicle, as this greatly impacts the skid behaviour of individual corners of a vehicle.

### 5.3.2 Safety Systems

While we have now shown that electronic brakeforce distribution may be improved by considering road geometry without explicit road knowledge, there is fundamentally only so much that can be done in that setting. In particular, EBD and other systems rely heavily on a driver to anticipate future behaviour of a vehicle and act accordingly, such as slowing down for a turn.

With knowledge of the road and surroundings of a vehicle we can develop systems which alert or override a drivers behaviour when it is unsafe. This is the field of advanced driver assist (ADAS) systems, and has seen numerous developments such as adaptive cruise control to follow highway lanes in the presence of traffic, as well as alert systems for sharp turns and vehicles in blind spots.

Given the emphasis on non-Euclidean road geometry, in this section we develop and test a safety system designed to proactively detect unsafe vehicle behaviour that will result from road geometry. For example, lateral road banking during a turn plays a major role in how fast a vehicle can safely travel. While one might derive analytic expressions for maximum safety speed for a turn with constant radius and bank, real roads require fully considering continuous variation in their shape.

This is exactly what general-purpose non-Euclidean road models enable, allowing safety systems to be developed which explicitly consider the effects of road geometry on safe operation of a vehicle. One example is presented in [38], which overrides user brake commands when reducing the speed of a vehicle is immediately necessary to slow down a vehicle.

An example solution of the model predictive control (MPC) problem set up in [38] is shown in Figure 5.12. In it, a safe speed profile has been determined for a spatial curve the vehicle should follow over a non-Euclidean road surface. Accompanying this on the right is a plot of the intervention profile: how much the user brake force must change along the future vehicle trajectory to achieve this speed profile. As shown, immediate intervention is necessary to reduce the speed of the vehicle to safe operating conditions, here determined by speed-dependent friction and rollover limits along the given trajectory.

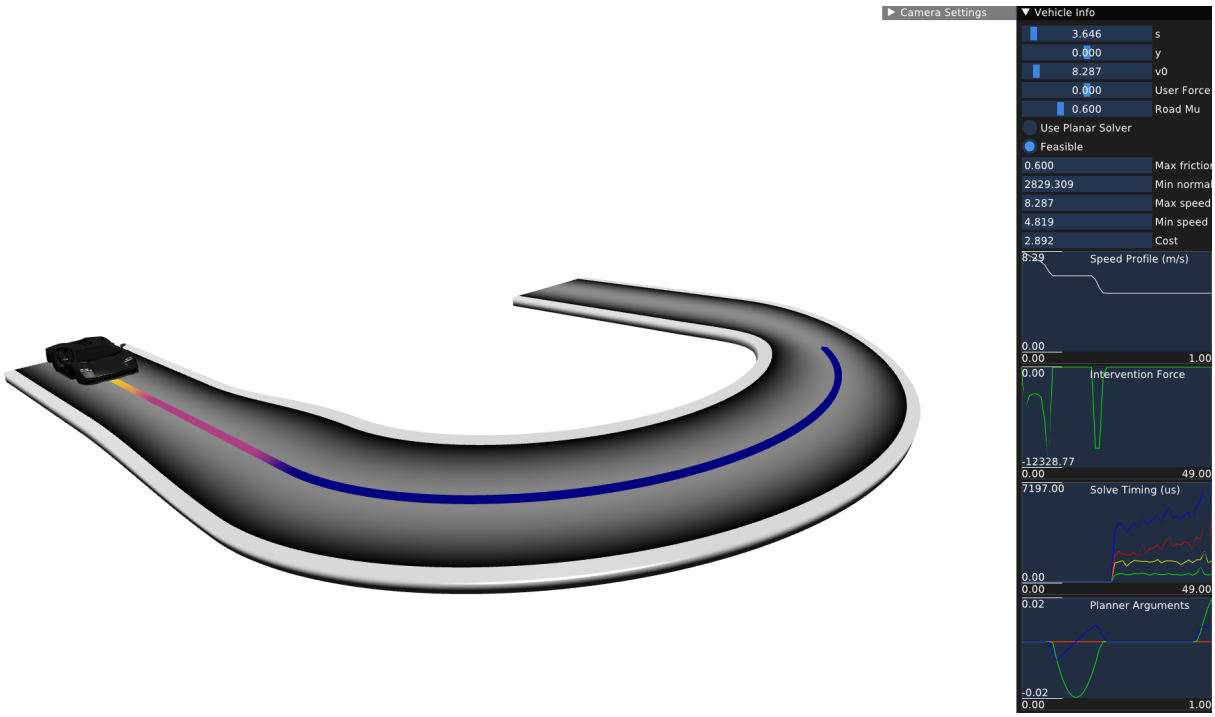



Figure 5.12: Example of one solution of the MPC safety system developed in [38]. The 3D racecar asset was made by Sketchfab user Lexyc16, is licensed CC BY-NC 4.0, and can be downloaded from skfb.ly/opvyH

Other safety systems may incorporate additional information, such as pedestrians, other vehicles, traffic rules, and intersections. All of these examples may be enhanced to consider non-Euclidean road geometry in a similar procedure to how the safety system in [38] was developed.

## 5.4 Other Potential Applications

Several other applications of non-Euclidean motion models, particularly those not yet explored are discussed here.

### 5.4.1 Machine Learning

While the previous applications discussed in this chapter largely fall within classical and conventional approaches to control, future application to machine learning is possible and should be noted. A particularly interesting development in the field of machine learning is physics-informed neural networks (PINN) [39]. These networks have proven successful at modeling fluid dynamics [40], heat transfer [41], and power systems [42], among other applications.

PINNs encode model equations within the neural network itself, and could potentially encode a motion model, or more complete vehicle model within their structure. While many typical applications of PINNs are to solve a differential equation and make predictions, such as weather forecasting, there is ample opportunity to develop machine learning approaches for control that leverage the structure and results of non-Euclidean motion models.

### 5.4.2 Fast Simulation

Control-oriented motion models as developed in this dissertation are predominantly motivated by their applicability for control. However, this does not prevent their use for simulation, provided that road and vehicle model assumptions are realistic for the simulation in question. As a result, the control-oriented models developed here may prove useful for generating large amounts of simulated data, such as for machine learning.

# Chapter 6
# Applications to Aircraft

Many of the applications to ground vehicles discussed in the previous chapter have parallels for aircraft, such as autonomy, racing, and tracking control. This chapter discusses a few of these with examples, and suggests several other potential areas of future research and application.

Note that aircraft which are taking off or landing from an airstrip bear some resemblance to a ground vehicle, at least temporarily. As a result, the motion models of Chapter 2 may be applied to certain aircraft. For example, the modeling considerations necessary to predict and avoid loss of contact with a road for a racecar can be reused to consider an aircraft which is trying to lose contact with a runway. While the control inputs to both vehicles are drastically different, both may be approximated with the same motion model.

Compared to ground vehicles, airborne aircraft have the distinction that they can easily and equivalently be described in Euclidean or non-Euclidean coordinate systems. As remarked in Chapter 3, using non-Euclidean geometry is optional and may not be appropriate for all applications. The remainder of this chapter discusses several such applications, along with the tradeoffs that can make Euclidean geometry preferable in certain applications.

## 6.1 Raceline Computation

Acknowledgement: Parts of this section appeared previously in [9].

As is the case for ground vehicles, computing minimum-time laps for aircraft may be done for optimal control of an autonomous vehicle, comparing the performance of different platforms, or training a human pilot. Unlike computing racelines for ground vehicles, aircraft may compute racelines in a Euclidean or non-Euclidean coordinate system. This is precisely the focus of [9], which introduce both approaches. In brief, the non-Euclidean approach is intrinsically limited by where said coordinates are defined, that is by the regularity limits discussed previously and expressed in Equation 3.12. However, the non-Euclidean approach has the advantage that it may be extended naturally to obstacle avoidance, a feat shown in [9]. Several raceline examples are presented in Figures 6.1, 6.2, and 6.3, with full results available in [9]. Source code for [9] is provided at github.com/thomasfork/aircraft_trajectory_optimization

In all figures the 3D quadrotor asset was made by Sketchfab user the_Thorminator, is licensed CC BY 4.0, and can be downloaded from skfb.ly/orIx8

## 6.2 Trajectory Tracking

A standard control problem for virtually all aircraft with some degree of autonomy is following a target trajectory. This is not limited to full autonomy, as many aircraft exhibit some degree of autonomy, such as autopilots on commercial aircraft. Furthermore, many

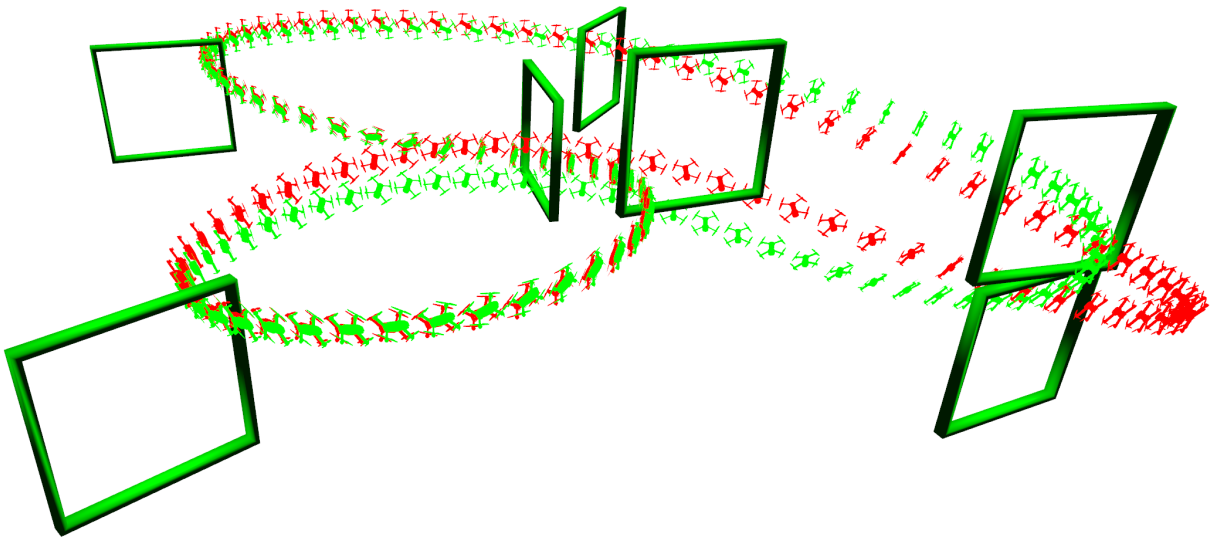


Figure 6.1: Euclidean (green) and non-Euclidean (red) racelines for a quadrotor on an aerial racetrack. The method used to solve for both is described in [9]. The non-Euclidean raceline is forced to make a wider turn on the right due to regularity limits.

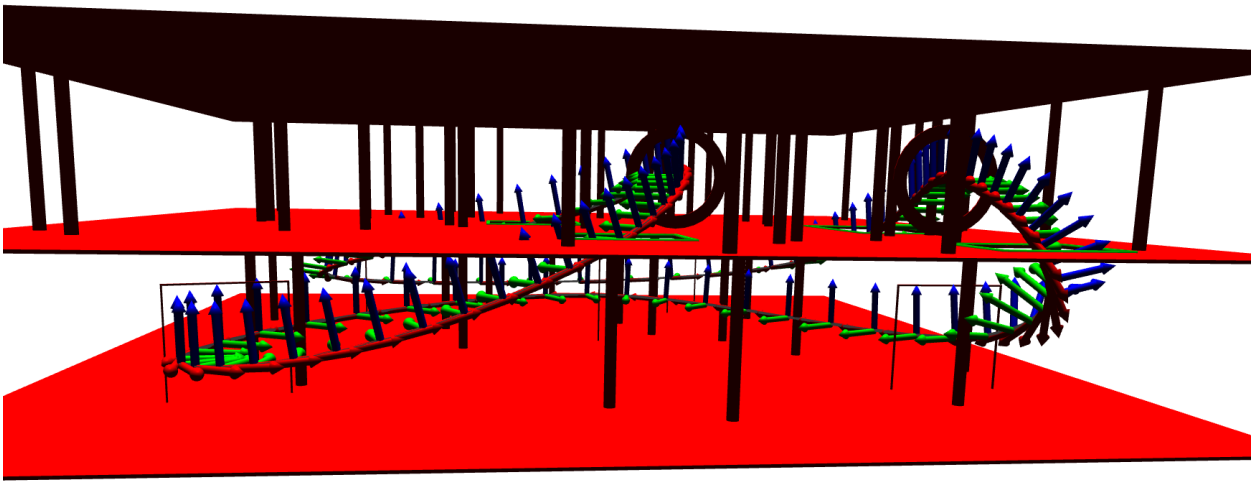


Figure 6.2: Obstacle-rich environment (red) adapted from [43] and used for raceline computation with obstacle avoidance in [9]. A centerline for a non-Euclidean coordinate system is shown with red, green, and blue arrows for $\boldsymbol{e}^c_{s,y,n}$ respectively.

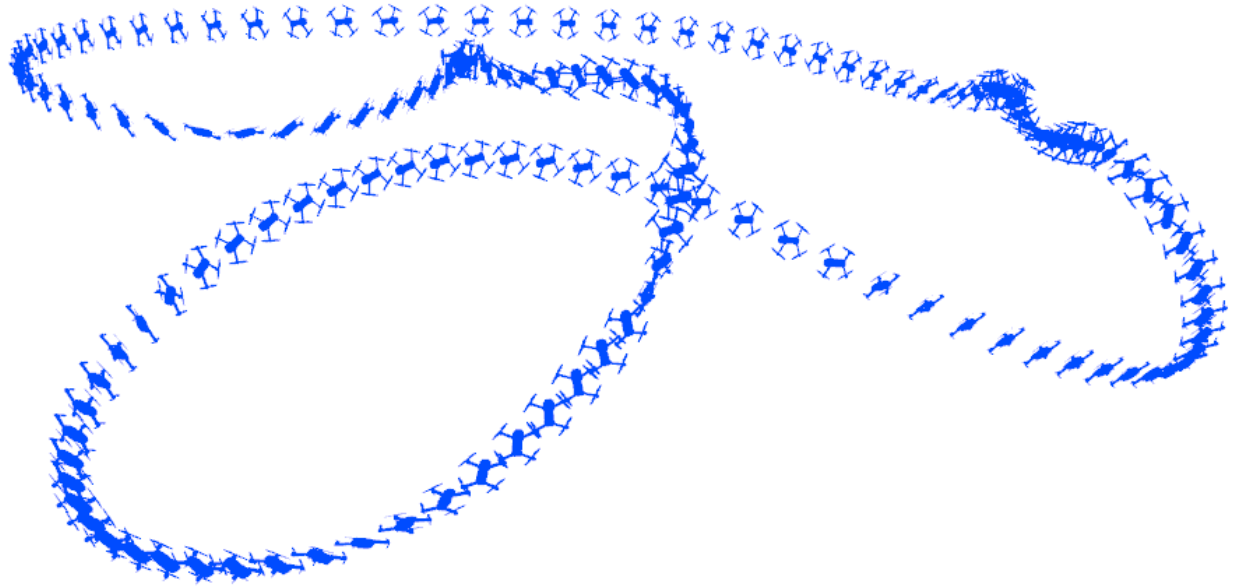


Figure 6.3: Raceline computed using augmented non-Euclidean approach through the environment shown in Figure 6.2. The method and results are presented in [9], and the obstacle-rich environment has been hidden to illustrate the maneuvers which result to avoid collision.

teleoperated systems, such as high-altitude military drones, incorporate some form of intermediary control between user inputs and actuator control, both to stabilize the vehicle and compensate for latency when teleoperated from long distances.

Various approaches have been developed to follow a spatial trajectory using non-Euclidean geometry to encode the trajectory, which can be done directly with the approach outlined in Chapter 3, adapted from [9]. These results allow for trajectory-tracking approaches such as developed in [11] to be applied with more generality and ease of use than before.

Several approaches in literature are not compatible, particularly contouring control approaches [44] which are geometrically incomplete. In [44], only a single lateral coordinate relative to the spatial reference curve is introduced, meaning knowledge as to whether or not one is below, above, left, or right of the reference is absent. This may be desirable in some settings as fewer variables are introduced, but in cases where it is important to stricter control on tracking error in a particular direction, such as when passing an obstacle, this may be insufficient.

# Chapter 7
# Conclusion and Future Work

## 7.1 Conclusion

This dissertation developed motion models which describe a vehicle relative to a surface or a curve. Said models did not exist previously, or existed in an unsatisfactory manner. For instance in the context of cars, only certain road shapes could be studied, and some works included small-angle or other approximations on top of motion assumptions. In this work however, motion models were developed from general and flexible foundations to which motion assumptions were applied in a strict and methodological manner.

In turn, these motion models enable the development of vehicle models for both ground vehicles and aircraft. Special emphasis has been made on motion assumptions and vehicle models which are used for control, such as the widely-used assumption that a car remains tangent to and in contact with the road it is driving on. The key novelty here is not any new model of any particular vehicle, but the ability to apply existing approaches to model specific vehicles on more general geometry. For instance much literature has been devoted to the modeling of racecars and passenger vehicles, which may now be used to develop models of said vehicles on general road geometry.

Furthermore, developing new and more general vehicle models allows them to be used for new and more general applications. Chapters 5 and 6 showed how new non-Euclidean vehicle models can be used to both extend existing control problems to new geometry and enable new control problems which could not be studied otherwise.

Importantly, in the process of developing motion models, vehicle models, and model applications it has been instrumental to think differently about how vehicle models are built and how geometry factors into said models. This is all-important to developing vehicle models that can be applied to different geometries, such as different road surfaces or aerial curves, and is essential to using these motion models in practice. This was introduced conceptually in Section 1.6 and numerically in Chapter 4, with Python implementations and applications available online:

Ground vehicles on surfaces: github.com/thomasfork/Nonplanar-Vehicle-Control/
Aircraft relative to curves: github.com/thomasfork/aircraft_trajectory_optimization

Finally, the research avenues opened in this dissertation have by no means reached their close. This work was inspired by the need for new tools to approach challenging problems, and having laid such a foundation, there are many areas for future research and application, discussed next.

## 7.2 Future Work

From the perspective of fundamental research, there are numerous systems of interest which do not fall within the motion models developed in this dissertation. Most evident of these are multi-body systems which cannot be approximated with a single rigid body. For example, quadruped robots involve several points of contact with the surface they interact with. Similarly, a tractor-trailer vehicle involves two separate rigid bodies, both of which may remain approximately tangent to the surface they are driving on, but have a fixed link between the two. This link is straightforwards to enforce in Euclidean settings, but not so when using non-Euclidean geometry as correspondence between the non-Euclidean coordinates of each body goes beyond local information on the shape of the surface, such as its curvature.

Similar theory extensions hold in applying the motion models from this dissertation to new vehicles. For instance, surface motion models may be applied to spherical robots [45] and aircraft motion models may be applied to helicopters and airplanes, and numerous control applications exist for each vehicle.

Beyond fundamental theory extensions, there are many other possibilities for future applied work:

In automotive research, many decades of work has focused on lateral and longitudinal behaviour, and non-Euclidean motion models may prove instrumental in the vertical control of the same. Much of this hinges on having access to information on the shape of the road surface, either from prior map data or online sensor fusion, both of which are areas of potential future work. However, not all is lost without a local map of the road surface. As was shown in Section 5.3.1, we can develop vehicular systems which consider three-dimensional road surfaces through inertial measurements, as was the case for electronic-brakeforce-distribution. This promising observation may lend itself to future work in commercial vehicle control systems without external road information.

The original inspiration for the line of research that turned into this dissertation was the modeling and control of racecars. In some ways, racecars are an ideal target application for this work as 1) racetracks are smooth, controlled, and often mapped-out surfaces, 2) racecars are engineered with stiff suspensions and low ground clearance, making tangent-contact motion assumptions fairly realistic, and 3) there is great interest in studying the performance of racecars, and developing autonomous racecars [46]. These efforts are excellent applications of non-Euclidean motion models, as racetracks are not always flat, and the shape of such racetracks must be considered to operate at maximum performance.

Further ground vehicle applications are abundant in the planning and control of off-road vehicles. Methods based on vehicle models which may use the motion models developed here[47] are immediate candidates for future work, as considerations for road shape and maintaining contact with the road may be considered formally in settings not previously possible. Furthermore, methods that are not directly model-based, such as ones based on point clouds [48] and graph search [37], may be extended in manners which consider the

same operating limits. As an example of how this might be done, Section 5.3.2 developed a safety system using non-Euclidean motion models without building a complete vehicle model. Rather, operating limits were directly enforced on a path with variable vehicle velocity, and similar thought processes may be possible for search-based methods as well.

Furthermore, there is an abundance of work on path-planning and following for aircraft. While non-Euclidean motion models are not necessary to describe the motion of aircraft, they have seen much interest for path-following [11, 44] as non-Euclidean coordinates can use geometry to encode the task of following a curve. The contributions of this work may prove useful in extending such work to include similar concepts for orientation and to allow for easier practical application of non-Euclidean coordinates. Furthermore, [9] made the case for using non-Euclidean coordinates for path planning, particularly in the presence of obstacles, an application for which there is abundant potential for future work, especially for online control.

Finally, this list is certainly incomplete. In the same sense as this work required the fusion of mechanics, geometry, vehicle dynamics, and controls, there are no doubt other multidisciplinary applications not yet discovered.

# Bibliography


[1] D. Limebeer and E. Warren, "A review of road models for vehicular control," *Vehicle system dynamics*, vol. 61, no. 6, pp. 1449–1475, 2023.

[2] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 1094–1099.

[3] R. Rajamani, *Vehicle Dynamics and Control*, 2nd. Springer, 2011.

[4] M. Guiggiani, *The Science of Vehicle Dynamics, Handling, Braking, and Ride of Road and Race Cars*, 3rd. Cham, Switzerland: Springer, 2023.

[5] R. N. Jazar, *Vehicle Dynamics, Theory and Application*. Springer, 2007.

[6] R. Lot and F. Biral, "A curvilinear abscissa approach for the lap time optimization of racing vehicles," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 7559–7565, 2014, 19th IFAC World Congress, ISSN: 1474-6670.

[7] S. Lovato, M. Massaro, and D. Limebeer, "Curved-ribbon-based track modelling for minimum lap-time optimisation," *Meccanica*, vol. 56, no. 8, pp. 2139–2152, 2021.

[8] G. Perantoni and D. J. N. Limebeer, "Optimal Control of a Formula One Car on a Three-Dimensional Track—Part 1: Track Modeling and Identification," *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 5, May 2015.

[9] T. Fork and F. Borrelli, "Euclidean and non-euclidean trajectory optimization approaches for quadrotor racing," *arXiv preprint arXiv:2309.07262*, 2023.

[10] S. Spedicato and G. Notarstefano, "Minimum-time trajectory generation for quadrotors in constrained environments," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1335–1344, 2018.

[11] J. Arrizabalaga and M. Ryll, "Towards time-optimal tunnel-following for quadrotors," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 4044–4050.

[12] J. C. H. Ramírez and M. Nahon, "A gravity-referenced moving frame for vehicle path following applications in 3d," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4393–4400, 2021.

[13] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[14] H. Pacejka, *Tire and Vehicle Dynamics*, 3rd. Butterworth-Heinemann, 2012.

[15] A. Rucco, G. Notarstefano, and J. Hauser, "Development and numerical validation of a reduced-order two-track car model," *European Journal of Control*, vol. 20, no. 4, pp. 163–171, 2014.

[16] T. Fork and F. Borrelli, "A general 3d road model for motorcycle racing," *arXiv preprint arXiv:2406.01726*, 2024.

[17] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, eabh1221, 2021.

[18] M. P. do Carmen, *Differential geometry of curves and surfaces*. Prentice Hall, 1976.

[19] R. Adler, M. Bazin, and M. Shiffer, *Introduction to General Relativity*. McGraw-Hill Book Company, 1965.

[20] T. Fork, H. E. Tseng, and F. Borrelli, "Models for ground vehicle control on nonplanar surfaces," *Vehicle System Dynamics*, pp. 1–25, 2023.

[21] A. Micaelli and C. Samson, "Trajectory tracking for unicycle-type and two-steering-wheels mobile robots," INRIA, Research Report RR-2097, 1993. [Online]. Available: `https://hal.inria.fr/inria-00074575`.

[22] T. Fork, H. E. Tseng, and F. Borrelli, "Models and predictive control for nonplanar vehicle navigation," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 749–754.

[23] S. Lovato and M. Massaro, "A three-dimensional free-trajectory quasi-steady-state optimal-control method for minimum-lap-time of race vehicles," *Vehicle System Dynamics*, vol. 60, no. 5, pp. 1512–1530, 2022.

[24] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2018.

[25] D. J. N. Limebeer and M. Massaro, *Dynamics and optimal control of road vehicles*. Oxford University Press, 2018.

[26] T. Fork, H. E. Tseng, and F. Borrelli, "Vehicle models and optimal control on a nonplanar surface," in *15th International Symposium on Advanced Vehicle Control*, 2022, pp. 749–754.

[27] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[28] L. T. Biegler, *Nonlinear Programming, Concepts, Algorithms, and Applications to Chemical Processes*. SIAM, 2010.

[29] M. A. Patterson and A. V. Rao, "Gpops-ii: A matlab software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Trans. Math. Softw.*, vol. 41, no. 1, Oct. 2014.

[30] A. Wächter and L. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, Mar. 2006.

[31] P. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent, "A fully asynchronous multifrontal solver using distributed dynamic scheduling," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 1, pp. 15–41, 2001.

[32] J. Betz *et al.*, "Tum autonomous motorsport: An autonomous racing software for the indy autonomous challenge," *Journal of Field Robotics*, vol. 40, no. 4, pp. 783–809, 2023.

[33] J. Betz *et al.*, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 458–488, 2022.

[34] E. L. Zhu and F. Borrelli, "A sequential quadratic programming approach to the solution of open-loop generalized nash equilibria," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 3211–3217.

[35] X. Yan, R. Zhang, J. Ma, Y. Ma, *et al.*, "Considering variable road geometry in adaptive vehicle speed control," *Mathematical Problems in Engineering*, vol. 2013, 2013.

[36] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2020.

[37] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[38] T. Fork, F. Camozzi, X.-Y. Fu, and F. Borrelli, "Predictive braking on a nonplanar road," *arXiv preprint arXiv:2406.01724*, 2024.

[39] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics–informed neural networks: Where we are and what's next," *Journal of Scientific Computing*, vol. 92, no. 3, p. 88, 2022.

[40] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (pinns) for fluid mechanics: A review," *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1727–1738, 2021.

[41] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks for heat transfer problems," *Journal of Heat Transfer*, vol. 143, no. 6, p. 060 801, 2021.

[42] G. S. Misyris, A. Venzke, and S. Chatzivasileiadis, "Physics-informed neural networks for power systems," in *2020 IEEE power & energy society general meeting (PESGM)*, IEEE, 2020, pp. 1–5.

[43] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5719–5726, 2022.

[44] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3340–3356, 2022.

[45] T. E. Honein and O. M. O'Reilly, "Explorations of the holonomy of a rolling sphere," *Proceedings of the Royal Society A*, vol. 480, no. 2282, p. 20 230 684, 2024.

[46] A. Wischnewski *et al.*, "Indy autonomous challenge-autonomous race cars at the handling limits," in *12th International Munich Chassis Symposium 2021: chassis. tech plus*, Springer, 2022, pp. 163–182.

[47] S. Yu, C. Shen, and T. Ersal, "Nonlinear model predictive planning and control for high-speed autonomous vehicles on 3d terrains," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 412–417, 2021.

[48] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart, "Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments," *Journal of Field Robotics*, vol. 34, no. 5, pp. 940–984, 2017.

# Appendix

# Appendix A
# Expanded Derivation from Eqn. (2.18) to (2.19)

Adapted from [22].

Starting with (2.18):

$$\dot{\theta}^s = \omega_3^b + \frac{\boldsymbol{e}_1^b \cdot \boldsymbol{x}_{ss}^p - (\boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p)(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_{ss}^p)(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p)^{-1}}{\boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p}\dot{s} + \frac{\boldsymbol{e}_1^b \cdot \boldsymbol{x}_{sy}^p - (\boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p)(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_{sy}^p)(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p)^{-1}}{\boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p}\dot{y}.$$

We recognize that the $\boldsymbol{e}_3^b$ component of $\boldsymbol{x}_s^p$ is zero and the $\boldsymbol{e}_3^b$ components of $\boldsymbol{x}_{ss}^p$ and $\boldsymbol{x}_{sy}^p$ are removed by the dot product with $\boldsymbol{x}_s^p$. We introduce placeholder variables $(a, b, c, d, q, r)$ used only here and only for the sake of concise notation defined such that:

$$\boldsymbol{x}_s^p = a\boldsymbol{e}_1^b + b\boldsymbol{e}_2^b \tag{A.1a}$$

$$\boldsymbol{x}_{ss}^p = c\boldsymbol{e}_1^b + d\boldsymbol{e}_2^b + \boldsymbol{e}_3^b(\boldsymbol{x}_{ss}^p \cdot \boldsymbol{e}_3^b) \tag{A.1b}$$

$$\boldsymbol{x}_{sy}^p = q\boldsymbol{e}_1^b + r\boldsymbol{e}_2^b + \boldsymbol{e}_3^b(\boldsymbol{x}_{sy}^p \cdot \boldsymbol{e}_3^b). \tag{A.1c}$$

With this in mind, (2.18) becomes:

$$\dot{\theta}^s = \omega_3^b + \frac{c - a(ac+bd)\frac{1}{a^2+b^2}}{b}\dot{s} + \frac{q - a(aq+br)\frac{1}{a^2+b^2}}{b}\dot{y}. \tag{A.2}$$

And we can rearrange over several steps:

$$\dot{\theta}^s = \omega_3^b + \frac{c(a^2+b^2) - ca^2 - abd}{b(a^2+b^2)}\dot{s} + \frac{q(a^2+b^2) - qa^2 - abr}{b(a^2+b^2)}\dot{y}. \tag{A.3}$$

$$\dot{\theta}^s = \omega_3^b + \frac{cb^2 - abd}{b(a^2+b^2)}\dot{s} + \frac{qb^2 - abr}{b(a^2+b^2)}\dot{y}. \tag{A.4}$$

$$\dot{\theta}^s = \omega_3^b + \frac{cb - ad}{a^2+b^2}\dot{s} + \frac{qb - ar}{a^2+b^2}\dot{y}. \tag{A.5}$$

$$\dot{\theta}^s = \omega_3^b + \frac{(\boldsymbol{e}_1^b \cdot \boldsymbol{x}_{ss}^p)(\boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p) - (\boldsymbol{e}_2^b \cdot \boldsymbol{x}_{ss}^p)(\boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p)}{\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p}\dot{s} + \frac{(\boldsymbol{e}_1^b \cdot \boldsymbol{x}_{sy}^p)(\boldsymbol{e}_2^b \cdot \boldsymbol{x}_s^p) - (\boldsymbol{e}_2^b \cdot \boldsymbol{x}_{sy}^p)(\boldsymbol{e}_1^b \cdot \boldsymbol{x}_s^p)}{\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p}\dot{y}.$$

Which brings us to (2.19).

# Appendix B
# Remark on Geodesic Curvature

The cross products in (2.20) should not be confused with geodesic curvature. In general they are different. In the case of an orthogonal surface parameterization ($\boldsymbol{x}_s^p \cdot \boldsymbol{x}_y^p = 0$) the cross products reduce to geodesic curvature as shown here. From [18, Sec. 4-3, Eqn. 1] we can write out the second partial derivatives of the surface in terms of Christoffel symbols:

$$\boldsymbol{x}_{ss}^p = \Gamma_{ss}^s \boldsymbol{x}_s^p + \Gamma_{ss}^y \boldsymbol{x}_y^p + (\boldsymbol{x}_{ss}^p \cdot \boldsymbol{e}_n^p)\boldsymbol{e}_n^p \qquad \boldsymbol{x}_{sy}^p = \Gamma_{sy}^s \boldsymbol{x}_s^p + \Gamma_{sy}^y \boldsymbol{x}_y^p + (\boldsymbol{x}_{sy}^p \cdot \boldsymbol{e}_n^p)\boldsymbol{e}_n^p \tag{B.1}$$

With an orthogonal surface parameterization, this lets us simplify $\dot{\theta}^s$ to:

$$\dot{\theta}^s = \omega_3^b + \frac{\Gamma_{ss}^y ||\boldsymbol{x}_s^p|| ||\boldsymbol{x}_y^p||}{\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p}\dot{s} + \frac{\Gamma_{sy}^y ||\boldsymbol{x}_s^p|| ||\boldsymbol{x}_y^p||}{\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p}\dot{y} \tag{B.2}$$

The Christoffel symbols are related to the derivatives of the first fundamental form [18, Sec. 4-3, Eqn. 2]. In the case of an orthogonal surface parameterization, we have

$$\Gamma_{ss}^y = -\frac{\frac{\partial}{\partial y}(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p)}{2\boldsymbol{x}_y^p \cdot \boldsymbol{x}_y^p} \qquad \Gamma_{sy}^y = \frac{\frac{\partial}{\partial s}(\boldsymbol{x}_y^p \cdot \boldsymbol{x}_y^p)}{2\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p} \tag{B.3}$$

Together the angular velocity expression is:

$$\dot{\theta}^s = \omega_3^b - \frac{\frac{\partial}{\partial y}(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p)}{2(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p)||\boldsymbol{x}_y^p||}\dot{s}||\boldsymbol{x}_s^p|| + \frac{\frac{\partial}{\partial s}(\boldsymbol{x}_y^p \cdot \boldsymbol{x}_y^p)}{2(\boldsymbol{x}_y^p \cdot \boldsymbol{x}_y^p)||\boldsymbol{x}_s^p||}\dot{y}||\boldsymbol{x}_y^p||. \tag{B.4}$$

The geodesic curvature of a curve of fixed $y$ moving along $s$ in an orthogonal surface parameterization [18, pg. 254] is:

$$(\kappa_g)_s = -\frac{\frac{\partial}{\partial y}(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p)}{2(\boldsymbol{x}_s^p \cdot \boldsymbol{x}_s^p)||\boldsymbol{x}_y^p||} \tag{B.5}$$

and of fixed $s$ moving along $y$ is:

$$(\kappa_g)_y = \frac{\frac{\partial}{\partial s}(\boldsymbol{x}_y^p \cdot \boldsymbol{x}_y^p)}{2(\boldsymbol{x}_y^p \cdot \boldsymbol{x}_y^p)||\boldsymbol{x}_s^p||} \tag{B.6}$$

Note that the above are functions of both $s$ and $y$. With this we can simplify the expression for $\dot{\theta}^s$ to:

$$\dot{\theta}^s = \omega_3^b + (\kappa_g)_s \dot{s}||\boldsymbol{x}_s^p|| + (\kappa_g)_y \dot{y}||\boldsymbol{x}_y^p|| \tag{B.7}$$

Which is the desired result - parametric angular velocity in terms of geodesic curvature.

Note that in the classical example of the Frenet frame, $(\kappa_g)_y = 0$, and remarkably, $\frac{(\kappa_g)_s}{||\boldsymbol{x}_s^p||}$ is constant at a given $s$ and equal to the geodesic curvature of the centerline.

# Notation

**Systems**

| | |
|---|---|
| $z$ | Differential States |
| $u$ | Inputs |
| $a$ | Algebraic States |
| $f$ | Differential equation, ie. $\dot{z} = f(z, u, a)$ |
| $g$ | Algebraic equation, ie. $0 = g(z, u, a)$ |

**Frames**

| | |
|---|---|
| ${}^{g}$ | Inertial frame of reference, also referred to as "global" frame |
| ${}^{b}$ | Body frame, typically the usual body-fixed frame at the center of mass |
| ${}^{p}$ | Parametric frame induced by non-Euclidean surface |
| ${}^{c}$ | Parametric frame induced by non-Euclidean curve |

**Some Vehicle Variables**

| | |
|---|---|
| $m$ | Vehicle Mass |
| $I$ | Moment of inertia coefficient |
| $l_f$ | Front wheelbase length |
| $l_r$ | Rear wheelbase length |
| $t_f$ | Half-width of front axle |
| $t_r$ | Half-width of rear axle |
| $h$ | Height of COM above ground |
| $v$ | Signed speed for kinematic models |
| $\gamma^f$ | Front steering angle |
| $\beta$ | Sideslip angle |
| $N$ | Normal force |
| $c$ | Camber angle for motorcycles |
| $d$ | COM displacement of motorcycle due to rider |
| $T$ | Propeller thrust |
| $l$ | Quadrotor size parameter |
| $c_\tau$ | Propeller drag torque factor |

**Vectors**

| | |
|---|---|
| $\boldsymbol{e}$ | Basis vector |
| $\boldsymbol{x}$ | Position |
| $\boldsymbol{v}$ | Linear Velocity |
| $\boldsymbol{\omega}$ | Angular Velocity |
| $\boldsymbol{F}$ | Force |
| $\boldsymbol{K}$ | Moment |

| | |
|---|---|
| $\boldsymbol{g}$ | Gravity |

**Parametric Surfaces**

| | |
|---|---|
| $s$ | First surface coordinate |
| $y$ | Second surface coordinate |
| $n$ | Normal offset from surface |
| $\boldsymbol{x}^p$ | Parametric surface, function of $s$ and $y$ |
| $\boldsymbol{x}^p_s$ | Partial derivative of surface with respect to $s$ |
| $\boldsymbol{x}^p_y$ | Partial derivative of surface with respect to $y$ |
| $\boldsymbol{e}^p_n$ | Normal vector of parametric surface |
| $\boldsymbol{x}^p_{ss}$ | $\partial_s\ \boldsymbol{x}^p_s$ |
| $\boldsymbol{x}^p_{sy}$ | $\partial_y\ \boldsymbol{x}^p_s$ |
| $\boldsymbol{x}^p_{ys}$ | $\partial_s\ \boldsymbol{x}^p_y$ |
| $\boldsymbol{x}^p_{yy}$ | $\partial_y\ \boldsymbol{x}^p_y$ |
| $\mathbf{I}$ | First fundamental form of $\boldsymbol{x}^p$ |
| $\mathbf{II}$ | Second fundamental form of $\boldsymbol{x}^p$ |
| $\boldsymbol{e}^p_s$ | Unit vector in direction of $\boldsymbol{x}^p_s$ |
| $\boldsymbol{e}^p_\perp$ | Orthonormalization of $\boldsymbol{x}^p_y$ w.r.t $\boldsymbol{e}^p_s$ |
| $\mathbf{R}^{gp}$ | Rotation matrix between reference frame induced by surface and global reference frame |

**Tangent Contact Road Model**

| | |
|---|---|
| $\theta^s$ | Heading angle realative to $\boldsymbol{x}^p_s$ |
| $\mathbf{J}$ | Jacobian between $\boldsymbol{x}^p$ and body frame |
| $\mathbf{Q}$ | "Q" element of Q-R decomposition of $\mathbf{J}$ |

**Unconstrained Road Model**

| | |
|---|---|
| $\mathbf{R}^{pb}$ | Rotation matrix between vehicle body and reference frame induced by surface |
| $\mathbf{J}_3$ | Jacobian for unconstrained road model |

**Centerlines**

| | |
|---|---|
| $s$ | Coordinate parameterizing the centerline |
| $y$ | Lateral coordinate |
| $n$ | Normal coordinate |
| $\boldsymbol{x}^c$ | Centerline, function of $s$ |
| $\boldsymbol{x}^c_s$ | Partial derivative of centerline with respect to $s$ |
| $\boldsymbol{r}^c$ | Lateral direction for non-Euclidean coordinate system |
| $\boldsymbol{e}^c_s$ | Unit vector in direction of $\boldsymbol{x}^c_s$ |
| $\boldsymbol{e}^c_y$ | Lateral unit vector orthogonal to $\boldsymbol{e}^c_s$ |
| $\boldsymbol{e}^c_n$ | Normal vector orthogonal to $\boldsymbol{e}^c_s$ and $\boldsymbol{e}^c_y$ |
| $\mathbf{R}^{gc}$ | Rotation matrix between centerline and global reference frames |
| $\mathbf{R}^{cb}$ | Rotation matrix between vehicle body and centerline reference frames |