UNIVERSITÀ DI PADOVA  FACOLTÀ DI INGEGNERIA

TESI DI LAUREA

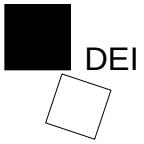
NonLinear Predictive Control for a NonHolonomic Nonminimum Phase System

Relatore: Prof. Ruggero Frezza
Correlatore: Prof. Alessandro Beghi

Laureando: Alessandro Abate

Anno Accademico 2001-2002

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA



DIPARTIMENTO DI ELETTRONICA E INFORMATICA

"... See Alex, let me tell you that ***feedback*** *is somehow the motor of the whole world!"*

Abstract

This work addresses the problem of ***nonlinear control of nonholonimic, nonminimum-phase systems***.

More specifically, the problem of *trajectory tracking for a motorcycle-like system* will be considered.

The literature on this topic appears to be rather limited, both with regard to the theorical side, as well as concerning the applications.

The path is considered to be fully known in advance, in other words it's by itself a deterministic variable. A mathematically rigorous, yet simplified model of the system will be defined and the controller will adopt some features of the ground-breaking paradigm of model-based predictive control.

Differently from previous approaches, the focus will be on defining feasible trajectories for the internal dynamics which, through an optimization procedure, will cause precise convergence to the desired path. This idea poses clearly defined bounds on the internal dynamics, thus preventing the whole system from a situation of instability, and also avoids, through the aforementioned optimization, the need of pointing out analytical definitions of feasible external connection trajectories which appear to be much harder to be precisely defined.

The whole controller has undergone a process of enhancement and improvement up to its computational weight and algorithmical complexity in order to be integrated into an existing commercial simulator. New problems connected with the use of a model for a "real" system had to be faced and solved.

Future trends and ideas for the whole projects are eventually described and suggested.

Simulations for both the theorical model and the real world application are provided, analyzed and fully discussed.

Sommario

L'oggetto di questa tesi è lo studio della tematica del ***controllo non lineare per sistemi anolonomi a fase non minima****.*

Più precisamente si focalizzerà l'attenzione al problema dell'inseguimento di traiettoria da parte di un sistema modellizzabile come una motocicletta.

In letteratura l'argomento appare alquanto nuovo, sia dal lato teorico, data la mancanza di euristiche o metodologie specifiche, che da un punto di vista di sviluppo di algoritmi applicativi.

La traiettoria, ovvero la strada, è considerata totalmente nota a priori, ovvero la si considera deterministicamente fissata.

Si adotterà , per il sistema in analisi un modello semplificato ma rigoroso, e lo schema di controllo proposto farà sue alcune caratteristiche dalla nuova metodologia del "controllo predittivo basato su modello".

Differentemente dagli approcci precedenti, l'idea chiave è stata quella di sviluppare traiettorie virtuali per la dinamica interna del veicolo (ovvero per il rollio) che, attraverso una procedura di ottimizzazione, portano ad un'effettiva convergenza verso la traiettoria desiderata, ovvero ad un effettivo inseguimento della strada. L'idea si è dimostrata efficace in quanto pone dei limiti alla dinamica del rollio del veicolo, prevenendo dunque potenziali situazioni di instabilità dello stesso; inoltre risolve il problema del progetto di traiettorie di connessione fisiche nel piano, il quale appare impreciso sia da un punto di vista formale che sotto l'aspetto delle prestazioni.

L'intera struttura di controllo è stata sottoposta ad un processo di progressivo miglioramento e di ottimizzazione per ciò che riguarda il peso computazionale, con l'intento finale di integrarla in un software commerciale esistente sul mercato. Si sono di conseguenza dovuti risolvere nuovi problemi pratici al momento dell' applicazione effettiva dello schema sviluppato.

Si suggeriscono nuove idee e potenziali miglioramenti per l'algoritmo.

Infine, si presentano e discutono simulazioni riguardanti il metodo sia nel suo sviluppo teorico, che nel suo utilizzo pratico.

Contents

List of Figures

Chapter 1

Introducing the problem

Riding a bicycle is not that hard. Intuitively our body teaches us how to balance and steer a bicycle, although we may not understand the physical reasons or laws behind this phenomenon.
Attempting to model such a system and learning how to "mathematically" steer it is indeed much more difficult.

A bicycle, as well as a motorcycle, is an intrinsically unstable system because it can easily reach an unstable state and fall over. This makes it much harder to control than a car, for instance, as the driver must not only go in the right direction, but also maintain his or her balance. Moreover, this is a complex system to model, as it can be only described through the influence of many forces and momenta, which are all included in a series of quite convoluted differential equations. From a purely kinetic point of view, it can be clustered into four different parts: *the rear axle, the front axle, the rear wheel and the front wheel.* Reasoning over all of its constraints, we can conclude that as a whole it is a system with *three degrees of freedom.* These degrees can be put in relation with the possible movements of the vehicle: longitudinal feed (forward and backward motion), rolling (lateral leaning) and steering (left and right). In a real setting, we also must consider the fact that tires are never ideal, but indeed they are subject to slipping or skidding: this non-ideality (dynamic friction relative to the ground, existing in every active situation) is indeed the main cause of the actual movement of the whole vehicle.

As it can be inferred from the previous deiscussion, the *modelling* of a motorcycle can present many inherent difficulties: a good job of the control engineer must mediate between a very adherent model, and a rough one. With the first choice, indeed, it could be possible to include into the modelization many distinctive characteristics, which could turn out in the future to be necessary to justify some behaviors. Conversely, a too precise model could turn out to be not enough robust and be suitable not to all the different models of vehicles. In our work, besides, we'll make a tradeoff of modelling errors for simplicity and tractability.

As the reader will come to know, there's a fairly large leap between a theorical solution to the problem and a practical one, and many times the integration of

some mathematical results, proven to be sound and well posed, into an empirical setting require updating the model itself.

Our approach was to tackle the problem of *path following for those nonminumum phase, nonholonomic systems* in a theorical framework through the *theory of nonlinear control*, and to implement the solution into an existing software interface.

We will first introduce the reader through the mathematical setting of the problem, after having defined the quantities which will come into play. We will also further explain some of the physical properties of these dynamical systems. We then give a detailed description of the proposed algorithm, or control scheme. Lastly, after giving a brief description of a software product that simulates the behavior of motorbikes, we'll analyze how the proposed algorithm was embedded into the product and all the improvements that we attained. One chapter will be dedicated to a mathematical derivation for another new, more complex model of the system. A discussion and an analysis of the possible future ideas will close the paper.

Chapter 2

A theorical setting

2.1 The Model

The selection of a model for the system is vital to the final performance of the whole controller. We consider a standard motorcycle model already used in similar analyzes, as illustrated in figure 2.1.

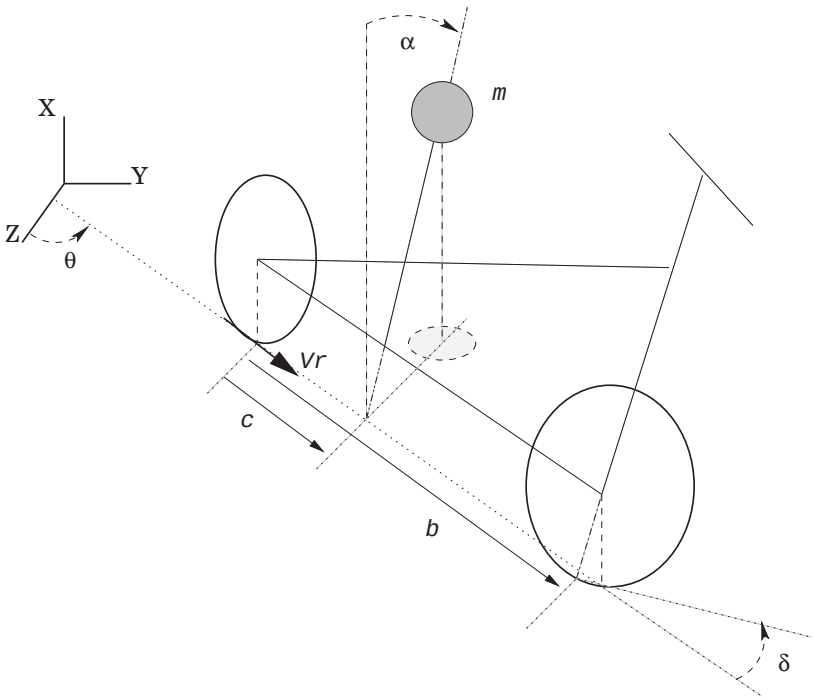


Figure 2.1: The Model for the Motorbicycle

The distance between the rear wheel contact point with the ground and the ground projection of the center of mass m along the *contact line* is c; the distance between the two wheels is b.

We make the following assumptions: the wheels have negligible mass, inertial moments, radius and width (i.e. they are assumed to be "lenticular"); also, they

roll through static friction without any slip or skid effect, and there is no associated damping factor with them (i.e., the system does not pitch). The steering wheel has a vertical axis as long as the bicycle's plane is normal to the flat ground; also, we neglect any influence of inertial moments due to the steering mechanism, especially in the derivation of the dynamics equations. All the massive effects are clustered into the center of mass, which is set at a height p. In other words, we assume to deal with what is often characterized as a *point-mass bicycle*.

2.1.1 Reference frames

Considering the dynamics of our system, we are free to refer its equations either to a ground-fixed inertial frame, or to a non-inertial, bicycle-fixed one (see figure 2.2).

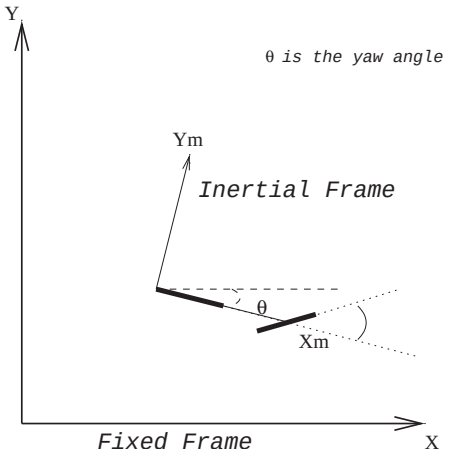


Figure 2.2: The Different Reference frames

The inertial frame has two of its dextrorotatory axes lying on the flat ground plane, and its third pointing vertically and opposite to the direction of gravity. The non-inertial frame is dextrorotatory as well, and has the x axis pointing forward in the direction of the bicycle, the z axis coinciding with the inertial equivalent, and the y one in the reasonable third direction, normal to the previous two.

We define *contact line* as the intersection of the vehicle's plane and the ground plane. The orientation of this line (pointed in accordance with the x axis of the body frame) with respect to the x axis in the inertial frame is named *yaw angle*, ϑ.

As long as the vehicle's plane of symmetry has some inclination with the normal to the flat ground, we have a non-zero angle, known as *roll angle*, α.
As mentioned above, our bicycle has always null *pitch angle*.

Rotating the steering wheel allows us to define a *steering angle* β, which can be easily connected with the curvature of the rear wheel, σ. More precisely, we can parameterize the steering angle through the following transformation:

$$\sigma = \frac{\tan(\beta)}{b} \tag{2.1}$$

A more accurate consideration will make the reader realize that the steering angle does not always match the angle of rotation of the steering shaft; indeed, calling ψ this last variable, we have that:

$$\tan\beta\cos\alpha = \tan\psi \tag{2.2}$$

In the case when α is null, we obtain obviously that $\beta = \psi$.

2.2 Cinematics and Dynamics Equations

We can refer our equations describing the vehicle's motion either to the body frame, or to the inertial one. Also the dynamics relations can be referred to the moving or to the fixed reference system, thus highlighting the influence of different forces.

Assuming that the vehicle has a longitudinal velocity v_r along the x-direction in the body frame, and a lateral one v_l, parallel to the y-axis, using a well known orthogonal matrix we easily state that

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) \\ \sin(\vartheta) & \cos(\vartheta) \end{bmatrix} \begin{bmatrix} v_r \\ v_l \end{bmatrix}, \tag{2.3}$$

thus obtaining a 1 to 1 relation between the two couples of velocities. Note that the matrix is always invertible, due to its structure. Along with v_r and v_l, the quantities ϑ and σ form the set of *generalized velocities*. These are the derivatives of the *generalized coordinates* referred to a non-inertial system. Expressing the external forces applied to the system in terms of the generalized coordinates, we define the *generalized forces* and distinguish them from the physical forces. These new quantities help defining the dynamic equations of a system referred to any non-inertial frame. For instance, considering a robot manipulator with its joints as generalized coordinates, we have that the generalized forces are the torques applied to the joint axes.

In our case, the two generalized forces are respectively the reaction due to friction that the ground plane applies to the rear wheel, τ^r, and the torque τ^σ applied through the steering shaft.

2.2.1 Nonholonomic Constraints

In Robotics, a *constraint* is a sort of restriction to the movements of a system,i.e. a reduction to its degrees of freedom or to the paths it can follow. Generally speaking, a constraint is said to be *holonomic* if it restricts the motion of the

system to a smooth hypersurface in its configuration space; algebraically, it can be expressed as

$$h_i(q) = 0, \qquad i = 1, ..., k; \tag{2.4}$$

where k defines the number of limits imposed to the system and h_i is a mapping from the configuration space to $\mathbb{R}$.

Assuming the full rank of the Jacobian of (2.4)

$$\frac{\partial h}{\partial q} = \begin{pmatrix} \frac{\partial h_1}{\partial q_1} & \cdots & \frac{\partial h_1}{\partial q_n} \\ \ddots & & \\ \frac{\partial h_n}{\partial q_1} & \cdots & \frac{\partial h_n}{\partial q_n} \end{pmatrix} \tag{2.5}$$

we infer that those constraints reduce the degrees of freedom of the system of n dimensions.

Through equation (2.4) we can define a set of *constraint forces*, which are linear combinations of the gradients of all the constraint functions h_i,

$$F = \frac{\partial h}{\partial q}^T \lambda, \tag{2.6}$$

and they're thought as acting normal to the constraint surface. No work is originated from them as long as the movements of the systems follow feasible trajectories.

If these constraints regard velocities instead of coordinates, they can be expressed as

$$C(q)\dot{q} = 0, \tag{2.7}$$

where $C(q) \in \mathbb{R}^{k \times n}$ stands for a set of k velocity constraints. Such constraints are named *Pfaffian Constraints*. Assumptions on their linear independence are similar to those aforementioned.

It makes sense to wonder whether these constraints on velocities can be integrated into corresponding algebraic restrictions on the coordinates: when this happens the Pfaffian Constraint is called **Holonomic**; conversely, we define a **Nonholonomic Constraint** as one of the form (2.7), which cannot be integrated. This implies that the set of constrained velocities does not match the (dimension of) the set of constrained coordinates.

2.2.2 Cinematics and Dynamics Equations, cont'd

In order to derive some dynamic equations for our model, we must take in consideration the effect of the constraints. We assume the idealized wheels roll without slip and are free to rotate around their z-axis, due to the steering action.

Remembering the definition of the velocities belonging to the system, considering also the roll and partitioning them into $\dot{r} = [\dot{\alpha}, v_r, \dot{\sigma}]$ and $\dot{s} = [\dot{\vartheta}, v_l]$, we have the following *nonholonomic constraints*:

$$\begin{bmatrix} \dot{\vartheta} \\ v_l \end{bmatrix} + \begin{bmatrix} 0 & -\sigma & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\alpha} \\ v_r \\ \dot{\sigma} \end{bmatrix} = 0; \tag{2.8}$$

From the first row, we have that $\dot{\vartheta} = \sigma v_r$, while the second gives $v_l = 0$, thus verifying our aforecited assumptions. An important entity is the curvilinear abscissa, which is given in our case by $s(t) = \int_0^t v_r(\tau)d\tau$.

Now that we are ready to write down the equations, let's refresh the reader's mind about *Lagrangian Equations*.

2.2.3 Lagrange's Equations

Two are the main motivations which naturally lead to the powerful lagrangian formalism in Mechanics : first, the expression of the dynamics equations in an arbitrary frame; second, the cancellation of reaction forces for constrained systems. There exist many approaches for their derivation, here we shall exploit the energy properties of a mechanical system.

Newton's second law for system of n particles gives

$$\mathbf{F} = \mathbf{m}\ddot{\mathbf{r}}, \qquad \mathbf{r} \in \mathbb{R}^{3\times n}, \tag{2.9}$$

where the bold quantities are n-dimensional vectors. These particles may undergo a set of *holonomic constraints* limiting their degrees of freedom. Constraints influence them through *constraint forces* $\mathbf{\Phi}$, which must appear in a relation of the form (2.9). As discussed above, a set of constraints induces a *vincular manifold or surface* in $\mathbb{R}^n$, on which $\nabla\mathbf{F}$[1] is never null. In other words, forces are tangent to these manifolds. We must then update equation (2.9) into

$$\mathbf{F} + \mathbf{\Phi} = \mathbf{m}\ddot{\mathbf{r}}, \qquad \mathbf{r} \in \mathbb{R}^{3\times n}. \tag{2.10}$$

A further analysis may show that $\mathbf{\Phi}$ can be expressed with a basis for the constraint forces and suitable *Lagrange multipliers*.

Relation (2.10) in general has two unknown variables, the constraint forces and the particles' movements. Therefore, we have to look for other relations allowing us to solve in terms of all the components. We could, for instance, project the partial derivatives of the coordinates along the vincular manifold, knowing that this work is always null, as previously stated.

As the reader might be aware, this problem appears to be quite computationally lengthy. Let's select l coordinates q_i, with $l = 3 \cdot n - c$ and c=dim($\mathbf{\Phi}$); we can express a relation similar to (2.9) in terms of them. Obviously, we're forced to look for a new set of *generalized forces* for these *generalized coordinates*. Independent of the choice of these coordinates, the new equations have the same form.

[1] We define the gradient at point x of a scalar function $\mathbf{F}$ in $\mathbb{R}^n$ as $\nabla\mathbf{F}(x) = \sum_{i=1}^{n} \frac{\partial \mathbf{F}}{\partial x_i}(x)e_i$.

Defining a function $L = T - V$, called **Lagrangian**, where T and V are the kinetic and potential energies, respectively. It can be demonstrated that these equations have the following form:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial qi} = \Phi_i, \qquad i = 1, \ldots, m, \tag{2.11}$$

where Φ_i is the external force along coordinate i.
These are the **Lagrange's Equations** in the general form.

2.2.4 Cinematics and Dynamics Equations, cont'd

Applying the aforementioned Lagrangian approach, we obtain the following Lagrangian for our system:

$$L = -mgps_\alpha + \frac{1}{2}J(\alpha,\sigma)\dot{\sigma}^2 + \frac{m}{2}\Big((v_r + ps_\alpha\dot{\vartheta})^2 + (v_l - p\dot{\alpha}c_\alpha + c\dot{\vartheta})^2 + (-p\dot{\alpha}s_\alpha)^2\Big), \tag{2.12}$$

where $J(\alpha,\sigma)$ is the moment of inertia of the front steering wheel about the steering axis, and we have used $[\sin(\alpha) = s_\alpha, \cos(\alpha) = c_\alpha]$ [2].

By substitution, we can include the constraints (2.8) into (2.12), obtaining

$$L_c = -mgps_\alpha + \frac{1}{2}J(\alpha,\sigma)\dot{\sigma}^2 + \frac{m}{2}\Big((v_r + ps_\alpha\sigma v_r)^2 + (c\sigma v_r - pc_\alpha\dot{\alpha})^2 + p^2\dot{\alpha}^2 s_\alpha^2\Big), \tag{2.13}$$

Now, the reader might know that in general the *Constrained Lagrangians* are not Lagrange Equations (for an easy demonstration please check [18]). Through some technicalities (see [13]) and disregarding the term J we can attain a correct, reduced and decoupled formulation:

$$\begin{aligned} \dot{\sigma} &= w_\sigma; \\ M\begin{pmatrix} \ddot{\alpha} \\ \dot{v}_r \end{pmatrix} &= F + B\begin{pmatrix} w_\sigma \\ u_r \end{pmatrix}, \end{aligned} \tag{2.14}$$

where

$$M = \begin{pmatrix} p^2 & -cpc_\alpha\sigma \\ -cpc_\alpha\sigma & 1 + (c^2 + p^2 s_{alpha}^2)\sigma^2 + 2p\sigma s_\alpha \end{pmatrix}$$

$$F = \begin{pmatrix} gps_\alpha + (1 + p\sigma s_\alpha)pc_\alpha\sigma v_r \\ -(1 + p\sigma s_\alpha)2pc_\alpha\sigma v_r\dot{\alpha} - cp\sigma s_\alpha\dot{\alpha}^2 \end{pmatrix}$$

$$B = \begin{pmatrix} cpc_\alpha v_r & 0 \\ -(c^2\sigma + ps_{alpha}(1 + p\sigma s_\alpha))v_\alpha & 1/m \end{pmatrix}$$

[2] All the letters have undergone a definition in the beginning of the present section.

Here $w_\sigma = \dot{\sigma}$ is intended to be a control, as well as the force u_r exerted by the rear wheel to the flat ground, through the engine's thrust.

From now on, we'll focus our attention just on the first row of equation (2.14), which actually describes the dynamic relation concerning the equilibrium of the bicycle. Let's rewrite this part separately to highlight it:

$$\boxed{p\ddot{\alpha} = gps_\alpha + (1 + p\sigma s_\alpha)pc_\alpha \sigma v_r^2 + cpc_\alpha(v_r\dot{\sigma} + \dot{v_r}\sigma).} \qquad (2.15)$$

2.3 Nonlinear Control

Considering, as in [21], the classical single-input single-output system

$$\begin{cases} \dot{x} = f(x) + g(x)u \\ y = h(x); \end{cases}$$

with $x \in \mathbb{R}^n$, f , g smooth vector fields on $x \in \mathbb{R}^n$ and h a smooth nonlinear function, i.e. an infinitely differentiable function, we can define the derivatives of the function h along the vector fields f and g through the concept of the *Lie Derivative*. Differentiating the second member of (2.16) w.r.t. time, we obtain:

$$\begin{aligned} \dot{y} = & \ \tfrac{\partial h}{\partial x} f(x) + \tfrac{\partial h}{\partial x} g(x)u \\ \doteq & \ L_f h(x) + L_g h(x)u. \end{aligned} \qquad (2.16)$$

Here $L_f h(x)$ and $L_g h(x)$ are the *Lie Derivatives* of h w.r.t. respectively f and g.

Through an intuitive *feedback linearization* is possible to render the system (2.16) linear, granted that $L_g h(x)$ is bounded away from zero in a particular open subset of $\mathbb{R}^n$. Contrariwise, differentiating again (2.16) we have:

$$\begin{aligned} \ddot{y} = & \ \tfrac{\partial L_f h}{\partial x} f(x) + \tfrac{\partial L_f h}{\partial x} g(x)u \\ \doteq & \ L_f^2 h(x) + L_g L_f h(x)u. \end{aligned} \qquad (2.17)$$

An attentive reader might have noticed the trend: we can reach a new linearization of (2.17) under the case $L_g L_f h(x) \neq 0$ into that open subset.

Generalizing the procedure, we can look for that index $\zeta > 0$ such that $L_g L_f^{\zeta-2} h(x) = 0$ in the open subset and $L_g L_f^{\zeta-1} h(x) \neq 0$ in there. The nonlinear system (2.16) is defined to have *strict relative degree* ζ in a point belonging to the mentioned set in case the two previous conditions hold.

Now, under the case $\zeta < n$ inside an open subset, it's possible to determine a *Normal Form* for the SISO system. Define:

$$\begin{aligned}\phi_1(x) &= \quad h(x),\\ \phi_2(x) &= \quad L_f h(x),\\ &\vdots\\ \phi_\zeta(x) &= \quad L_f^{\zeta-1} h(x).\end{aligned} \tag{2.18}$$

The terms $\phi_i(x)$ are new coordinates corresponding to y and its derivatives and do not depend on u. Also, it can be proven [3] that the system made up of the derivatives of (2.18) is linearly independent; by Frobenius theorem [4] we can define $n-\zeta$ functions $\eta_1(x), \eta_2(x), \ldots, \eta_{n-\zeta}(x)$ such that the system

$$\begin{bmatrix} dh(x) \\ dL_f h(x) \\ \vdots \\ dL_f^{\zeta-1} h(x) \\ d\eta_1(x) \\ d\eta_2(x) \\ \vdots \\ d\eta_{n-\zeta} \end{bmatrix} \tag{2.19}$$

has rank n in the same open subset (which could at worst be a single point). Thus, we can define a local *diffeomorphism*[5] by

$$\Phi : x \mapsto \begin{pmatrix} h(x) \\ L_f h(x) \\ \vdots \\ L_f^{\zeta-1} h(x) \\ \eta_1(x) \\ \eta_2(x) \\ \vdots \\ \eta_{n-\zeta} \end{pmatrix} \tag{2.20}$$

Referring to the two parts of the whole system as ξ and η, we can express the system (2.16) as

[3] Through the use of the so called *Lie Brackets* and the theory of distributions (which we skip to cover; the reader can find a precise analysis in [21]).

[4] Given a set of smooth vector fields $X_1, X_2, \ldots, X_n$, we define a distribution $\Delta(x) = span(X_i)$; the theorem states that *A nonsingular distribution is completely integrable if and only if it is involutive*, where a distribution is *involutive* if and only if the *Lie Bracket* of two of its vectors lies in the same distribution and is *integrable* if, having dimension $d < n$, $\exists\ n-d$ functions forming a complementary *codistribution*.

[5] A map f is a *diffeomorphism* if it is a *homeomorphism* (i.e. a one-to-one continuous map with continuous inverse) if both f and f^{-1} are smooth.

$$\begin{aligned}
\dot{\xi}_1 &= \xi_2, \\
\dot{\xi}_2 &= \xi_3, \\
&\vdots \\
\dot{\xi}_\zeta &= b(\xi,\eta) + a(\xi,\eta)u, \\
\dot{\eta} &= q(\xi,\eta), \\
y &= \xi_1
\end{aligned} \tag{2.21}$$

This system description is referred to as the **"normal" form**.

In the fortunate case where the relative degree is exactly n, we obtain a form for the system (2.16) identical to (2.21), except for the absence of the part $\dot{\eta} = q(\xi,\eta)$. It is possible to establish some conditions for this to happen. In general we'll talk about the *State Space Exact Linearization* Problem when, given a system of the form (2.16) with no output, we look for a function $h(x)$ s.t. the relative degree of the resulting system is exactly n.

It's time for an important definition, as in [21] (here we'll skip all the specifications on asymptotic or exponential behavior):

Definition: The system (2.16) is said to be *minimum phase* at x_0 if the equilibrium point $\eta = 0$ is locally stable.

Everything smooth so far, but the central starting point in our paper is that the model described above, (2.14) , and in general each other model for vehicles like bicycles, doesn't verify this property, i.e. it's namely **non minimum phase**.

As the reader might have already studied, for linear systems with an usual transfer function, the corresponding definition involves the presence of zeros in the right half-plane of the S-transform of the impulse response. In the case of discrete systems this condition refers zeros external to the unit circle.
Even if apparently this seems just one among a wide set of conditions, it's indeed a pretty tough one. For instance, stabilization and tracking for minimum-phase systems has many pretty effective methods, while similar problems for non-minimum phase systems have just been superficially addressed. We'll give some contributions toward this goal.

2.3.1 Jacobian Linearization

We've already spent some lines hinting at the difference between the *Linearization by State Feedback* procedure and the *Jacobian Linearization*, so for the sake of completeness let's devote some more efforts to briefly summarize this second one.

While, as previously described, the *linearization by state-feedback* aims at obtaining an *exactly linear* input-output response for the nonlinear system, the *jacobian* one just helps to characterize "linearly" a system around some particular points of the phase plan. Usually this points are equilibrium ones, which means that the system has a bias in remaining close to them, and indeed this linearization turns out to be consistent just in a small neighborhood of these points.

Given the classical relation for the state part of a SISO (scalar) system of the form

$$\dot{x} = f(x, u), \tag{2.22}$$

and having determined equilibrium values x_0, u_0, that is, values for which $\dot{x_0} = 0 = f(x_0, u_0)$, we proceed with a small-signal linearization expanding the nonlinear equation in terms of perturbations from these equilibrium values letting $x = x_0 + \delta x$ and $u = u_0 + \delta u$, so that

$$\dot{x_0} + \delta\dot{x} \cong f(x_0, u_0) + F\delta x + G\delta u, \tag{2.23}$$

where

$$F = \left[\frac{\partial f}{\partial x}\right]_{x_0,u_0}, \qquad G = \left[\frac{\partial f}{\partial u}\right]_{x_0,u_0}. \tag{2.24}$$

Subtracting the equilibrium solution, this reduces to

$$\delta\dot{x} = F\delta x + G\delta u, \tag{2.25}$$

which is a linear ODE approximating the dynamics of the motion about the equilibrium point. Always keep in mind that this expression is only valid in a close neighborhood of the very same equilibrium point.

2.4 Model-Based Predictive Control

This is a relatively new trend in Controls, and as many dawning methods it comes with many expectations, as well as many blurry aspects. No deep theorical research has passed through so far, and many aspects have to be wholly grasped yet. Still, many applications ensure that it is quite promising. Our work will conglomerate some of its features.

Consider a nonlinear control system

$$\begin{cases} \dot{x} = f(x(t), u(t)), \\ x(t_0) = x_0; \end{cases} \tag{2.26}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$ are the state and control vectors respectively. We impose a bound on the input, $u(t) \in U$, where $U \subset \mathbb{R}^m$.

In general, the MPC problem can be stated as:
For any state x at time t, find a continuous function $\hat{u}(\tau, x(t)) : [t, t+T] \to U$ such that the performance index

$$J = x(t)^T P x(t) + \int_0^T \hat{x}(t+\tau)^T Q \hat{x}(t+\tau) + \hat{u}(t+\tau)^{t+\tau} R \hat{u}(t+\tau) d\tau \tag{2.27}$$

is minimized where $Q \geq 0$, $P, R > 0$.

Then the nonlinear MPC law is determined by $u(t) = \hat{u}(t, x(t))$.
The two main problems this method arises are *Computational Burden* and *Stability*. For a deeper insight about these two aspects refer to [3] and [4]. We will return to these topics describing our new approach later on.

2.5 Differential Flatness

This concept is very interesting and plays indeed a key role in our approach, as the reader will appreciate in the following parts of the paper. It was introduced by *Fliess et. al.* and has a direct application in the control of vehicles. For a more in-depth look, the reader is invited to refer to [7, 8, 9].

Roughly speaking, **flatness** is *trajectory invertibility. Flat systems* are those systems that are "equivalent" in a certain way to systems with no dynamics, described by a set of independent variables, namely the *flat output*. In other words, having an a-priori trajectory in the output, the state and the input of the system can be uniquely derived through algebraic relationships. This bijective relationship simplifies all the handling of the systems, as we're sure we can operate in either one of the two descriptions. The tradeoff is that for nonlinear systems there's no systematic way to find a suitable flat output and there is much computational complexity.

Still, it can be demonstrated that our model for the motorcycle verifies all the mathematical assumptions for the differential flatness. This concept comes together with another important one, that of Inverse Dynamics.

2.6 Inverse Dynamics

Given a control system, the problem of determining a state trajectory such that the output resulting from that state trajectory is an assigned function is named *inversion problem*. The feasible state trajectory is called the *inverse* corresponding to the desired output.

As the reader might easily grasp, differential flatness is an important condition for being able to work with the inverse dynamics; indeed, we will make sure to express any output function as an algebraic equation for the internal state, as defined in the section on nonlinear control.

This concept of *inverse dynamics* will turn out to be a key one for the development of the proposed control scheme; sometimes through the paper, we'll work directly with the inverted equations, implying to be able to switch to the "direct" part whenever we need to refer to and work with the trajectory on the flat output.

2.7 Splines and Bézier Curves

We shall be interested in deploying $2D$ curves on the plane represented parametrically as

$$\mathbf{C}(s) = (X(s), Y(s)),$$

where $X(s)$ and $Y(s)$ are functions of the parameter s. These functions are usually polynomials, or more correctly *piecewise polynomial* curves; thus the parameter, throughout its extension $s_{min} \leq s \leq s_{max}$, will be broken into a sequence of values, or *knots*.

Now, *Splines Curves* are simply functions which are suited to pass through a sequence of points in the space. This ordered set of points can then be regarded as a vector for the curve's parameter. Using a sufficient grade of the polynomials, we can have the points' constraints satisfied exactly, or even impose some more constraints on the function's derivatives, to obtain smooth connections between the segments.[6]

We can choose to represent the curve's segments as:

$$\begin{aligned} \mathbf{Q}_{i-1}(s) &= (X_{i-1}(s), Y_{i-1}(s)) \\ &= \Big((1-s)x_{i-2} + sx_{i-1}, (1-s)y_{i-2} + sy_{i-1}\Big) \\ &= (1-s)\mathbf{V}_{i-2} + s\mathbf{V}_{i-1}; \\ \mathbf{Q}_{i}(s) &= (X_{i}(s), Y_{i}(s)) \\ &= \Big((1-s)x_{i-1} + sx_{i}, (1-s)y_{i-1} + sy_{i}\Big) \\ &= (1-s)\mathbf{V}_{i-1} + s\mathbf{V}_{i} \end{aligned} \tag{2.28}$$

We obtain in this case a *linear B-Spline*. It has the property of the so called *local control*, i.e. altering the position of one vertex only affects a part of the curve to change. These points are named *control vertices*. In this way, we can represent any curve in the plane changing the control vertices; the letter "B" stands for *basis* by virtue of this fact.

A *degree b Bézier Curve* is defined, similar to *B-Splines*, as:

$$Q(u) = \sum_{i=0}^{d} V_i P_{i,d}(u), \qquad 0 \leq u \leq 1, \tag{2.29}$$

where

[6] This method is regarded as *Hermite Interpolation*. We skip all the very easy algebraic explanations thereabout.

$$P_{i,d}(u) = \binom{d}{i} u^i (1-u)^{d-i} \tag{2.30}$$

are known as *Bernstein Polynomials*. The following is a proposition which will show itself to be of fundamental success for the heuristic we are proposing through this report.

Fact: *A Bézier Curve lies within the convex hull of its defining control vertices.*

Proof: Using the Binomial Theorem, first we write for a degree d Bézier curve:

$$\begin{aligned} 1 &= [(1-u)+u]^d \\ &= \textstyle\sum_{i=0}^{d} \binom{d}{i} u^i (1-u)^{d-i} \\ &= (1-u)^d + du(1-u)^{d-1} + \ldots + du^{d-1}(1-u) + u^d \\ &= P_{0,d}(u) + P_{1,d}(u) + \ldots + P_{d-1,d}(u) + P_{d,d}(u). \end{aligned} \tag{2.31}$$

Thus the $P_{i,d}(u)$ sum to one. As $0 \leq u \leq 1$ the quantities u and $(1-u)$ are both nonnegative; therefore $P_{i,d}(u)$ are nonnegative as well. From all this we infer that any Bézier Curve must lie within the convex hull of its control vertices. *QED*

Chapter 3

Previous Approaches

3.1 Introduction

In this chapter we will describe and explain all the previous approaches for controlling a nonlinear, nonminimum-phase system subject to nonholonomic constraints. We will only partially touch a couple of methods which pose heavy conditions on the system's characteristics; then we'll spend quite a lot of time to the approach by Neil Getz. Thereafter, we'll analyze a method suggested by Dirk Von Wissel, and finally we will assess a methodology already used in some technical software. All this work turned out to be fundamental to the understanding of the problem. Also, the new algorithm we're proposing is just the final fruit of a long development which went through the improvement of many parts of the following heuristics. We shall not describe the whole long path to the eventual solution, but we will give many explanations about all the problems and drawbacks, as well as of course all the main achievements.

3.2 Nonlinear Methods

There are two main approaches to solving the problem, *system inversion* and *output regulation*.

The first one, proposed by Devasia, Chen and Paden, applies to systems that have strict relative degree γ and thus have the form (2.21). The reference trajectory has to be bounded and defined through the whole time interval. The main assumption is that the system's *zero dynamics* can be linearized (see section 2.3.1) to a *kinematically hyperbolic matrix*. This is a condition on the stability for the whole system. After many technicalities, the authors come to define a *non-causal* feedforward control law, and also a feedback action for the error coordinates (for a deeper analysis, please check [21] and [22]). the main drawbacks are its non-causal form (the reference trajectory should be defined ahead of time), and the fact that it practically works only for trajectories that are small in amplitude and slowly varying. In other words, it looks like being quite awkward to apply.

The second method, developed by Byrnes and Isidori, makes the fundamental assumption that the desired trajectory is generated by a so-called *exosystem* described by

$$\begin{aligned} \dot{w} &= s(w), \\ y_D &= r(w); \end{aligned}$$

Thus we have to regulate (asymptotically) to zero the output e of the system

$$\begin{aligned} \dot{x} &= f(x) + g(x)u, \\ \dot{w} &= s(w), \\ e &= h(x) - r(w); \end{aligned}$$

This is obtained through many convoluted calculations; on overall, it appears suffering from the main drawbacks of the previous method: non-causality and excessive computational weight.
We shall disregard these approaches in the future, trying to obtain something more substantial and less mathematically burdensome.

3.3 Control for Autonomous Bicycles

In his many papers on the *Control for Autonomous Bicycles* (cf. for instance [12], [13], [14]), Neil Getz has suggested a ground-breaking approach to the problem, proposing a feedback control law for stable tracking of smooth trajectories.

3.3.1 Tracking of roll-angle trajectories and velocities

The author assumes a model for the bicycle identical to the one described in our introduction. A control scheme is proposed for tracking smooth roll-angle trajectories with non-zero rear-wheel velocity (which aren't really tight constraints at all), even if the initial conditions are far away from the ideal ones; in other words, it's been constructed an *internal tracking controller*. The convergence to the desired trajectory is exponential.

Although the derivation of the model follows different paths, the final form looks exactly like the one described in page 21. Now, in the case $v_r \neq 0$ and B is invertible, it's possible to choose as input

$$\begin{bmatrix} w_\sigma \\ u_r \end{bmatrix} = B^{-1}(-F + Mv), \tag{3.1}$$

thus obtaining a linearized form

$$\begin{bmatrix} \ddot{\alpha} \\ \dot{v}_r \end{bmatrix} = v. \tag{3.2}$$

Defining $\alpha_d(t)$ and $v_d(t)$ the desired time-trajectories for the roll angle and the velocity, assumed sufficiently smooth for our purposes, we can choose suitable coefficients c_i in order to obtain the following form for the variable v:

$$v = \begin{bmatrix} \ddot{\alpha_d} - c_{11}(\dot{\alpha} - \dot{\alpha d}) - c_{10}(\alpha - \alpha_d) \\ \dot{v_d} - c_{20}(v_r - v_d) \end{bmatrix}. \tag{3.3}$$

In the case the polynomials formed by the chosen coefficients have roots with non-positive real part, we obtain a control law assuring exponential convergence to the desired trajectories.
This nonlinear controller in other words gives balance to the non-minimum phase system, but it doesn't solve the problem of *tracking trajectories in the x, y plane, while retaining balance at the same time.*

The attentive reader could notice that, defined a path in the x, y plane, one could derive a time-varying function of the desired roll angle in the following way: it's always possible to derive the *curvature radius* R of a given the path in terms of x and y. The curvature at the rear wheel is then $\sigma = 1/R$. Now, given a relation of the form (2.15), page 21, we could in theory solve for α, thus obtaining the desired roll trajectory. Having still to realize how to algebraically solve this equation, if it were possible to obtain the $\alpha_d(t)$, the tracker would indeed stabilize the vehicle, but it wouldn't definitely cause an exact tracking of the flat trajectory, as random errors would be integrated in an open-loop cycle without the possibility to correct them.

3.3.2 Tracking a time parameterized path with stability

Let's reiterate the equation describing the internal dynamics of the bicycle:

$$\boxed{p\ddot{\alpha} = gps_\alpha + (1 + p\sigma s_\alpha)pc_\alpha\sigma v_r^2 + cpc_\alpha(v_r\dot{\sigma} + \dot{v_r}\sigma).} \tag{3.4}$$

It's intuitive to define an *equilibrium angle* α_e as a solution to the implicit relation

$$\begin{aligned} 0 = \;& gps_{\alpha_e} + (1 + p\sigma s_{\alpha_e})pc_{\alpha_e}\sigma v_r^2 + cpc_{\alpha_e}(v_r\dot{\sigma} + \dot{v_r}\sigma) \\ = \;& F_\alpha(\alpha, \sigma, v_r, \dot{v_r}, \dot{\sigma}). \end{aligned} \tag{3.5}$$

Clearly, it would be reasonable to assume that our desired roll matched this equilibrium solution, as we would be indeed sure that our vehicle retains stability at this speed and under this current steering action.

Now the problem is to solve the previous equation. It is intuitive that an algebraic solution is pretty hard to be derived; furthermore, we want to obtain an online result, i.e. we're interested in the computational weight. Neil Getz has suggested a brilliant solution to this problem.

Dynamic Inversion of Nonlinear Maps

Through the following method it will be possible to obtain a running estimate for α_d. In this paper we'll limit our analysis to a brief description, just to give the reader an idea on how the problem has been tackled(for a deeper and complete understanding, please refer to [14]).

Dynamic Inversion is a methodology for inverting nonlinear maps. Let's assume we're given a map $F : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}^n$ and we know somehow that there exists an unique isolated solution $\vartheta_*(t)$ of $F(\vartheta, t) = 0$. We'll match this map with a new one, namely a *dynamic inverse*, $G(\varsigma, \vartheta, t)$, such that the new system

$$\dot{\vartheta} = -G[F(\vartheta, t), \vartheta, t]$$

has a solution asymptotically convergent to $\vartheta_*(t)$.

In the simple case of a real function $F : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ looking like figure (3.1),

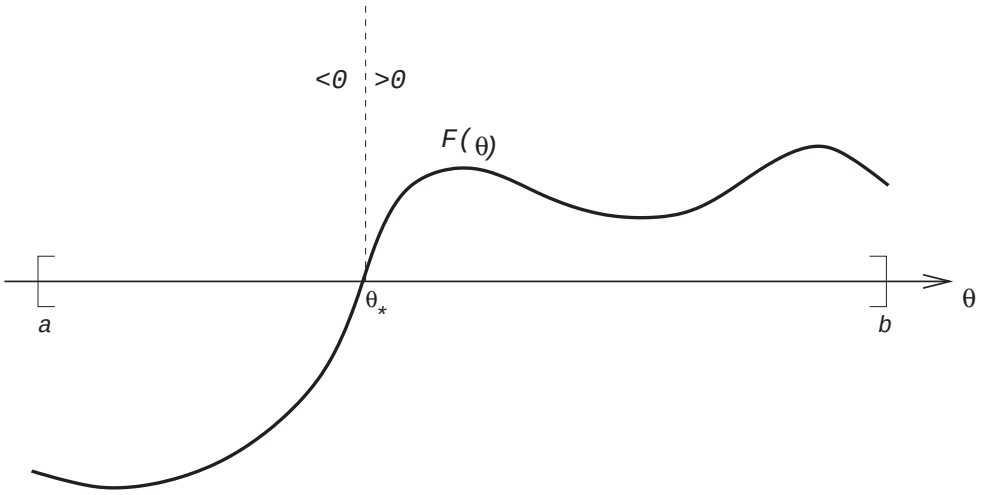
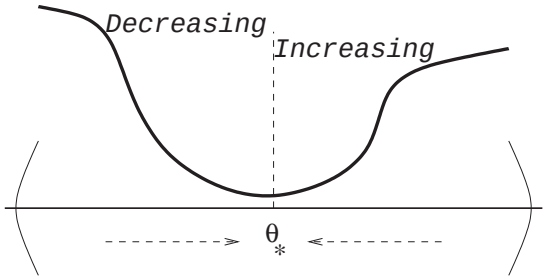


Figure 3.1: Case Function

we know it has a single isolated solution to $\vartheta \mapsto F(\vartheta)$, $F(\vartheta) = 0$, although at an unknown point ϑ_*. We claim that $G(\vartheta) = F(\vartheta)$ is the dynamic inverse; indeed, the solution of the system $\dot{\vartheta} = -F(\vartheta)$ asymptotically converges to θ_* (to realize this, just observe figure (3.2)).

In the general case, the dynamic inverse will be time-varying and vector-valued, and there are some sufficient conditions(mainly on the regularity of F around its solution) assuring its existence. Also, given the complex structure of this inverse, it can be approximated under some assumptions. The next step is to define a dynamic system whose state is an estimator for the root ϑ_* of $F(\vartheta, t) = 0$. A couple of theorems state that, once found the dynamic inverse $G[\varsigma, \vartheta, t]$, there exist an adequate $\mu > 0$ such that the solution to

$$\dot{\vartheta} = -\mu G[F(\vartheta, t), \vartheta, t], \qquad \vartheta(0) = \vartheta_0$$



Solution to the Dynamical System around the root.

Figure 3.2: Asymptotic Convergence for the System built over the Case Function.

approximates $\vartheta_*(t)$ with estimation error decaying to zero.

It's also possible to clamp together the two procedures, thus obtaining an online procedure for deriving both a dynamic inverse as well as an approximation for the root of the equilibrium system. Finally, applied algorithms manage to to estimate the derivatives of the aforementioned roots.

3.3.3 Application of the Dynamic Inversion

Here we will apply the concepts just explained to an example, that of tracking a path with stability.

We're interested in finding a running solution to the equation (3.5). As previously stated, a system whose state is the desired estimator $\hat{\alpha}$ is

$$\dot{\hat{\alpha}} = -\mu F_{alpha}(\hat{\alpha}, \sigma, v_r, \dot{v_r}, \dot{\sigma}) + L_{\hat{\alpha}\alpha_d}|_{\alpha_d=\alpha}.$$

A Jacobian linearization[1] of this differential equation would make the reader realize that there's exponential convergence of the solution to α_d in a convenient neighborhood.

It's even possible, as anticipated, to estimate the derivatives of $\hat{\alpha}$, therefore applying a feedback linearization control as that of section (3.3.1) results in approximate tracking of ground plane trajectories with balance. The convergence to the desired trajectory for the external system and to the equilibrium angle for the internal dynamics is even exponential until a certain time, when it persists bounded in a small region around the desired values.

Remark

Even if this second approach looks definitely more precise in tracking the desired path than the first one, especially with reference to the convergence performance, we still argue that possible errors might be not cancelled in the process of integrating the equations. More precisely, observing that the two feedback stabilization

[1]For a definition of this important concept, check sec. 2.3.1

actions work for the roll angle and the velocity, we can realize that there's no direct relation of the vehicle's position towards the road; this means that possible errors (for instance due to calculations) on the system's position might be not compensated just through the use of the roll regulation. These errors might also be indefinitely integrated, thus eventually bringing to an increasingly imprecise tracking for the reference path. In other words, the vehicle retains its balance, but loses the track of the road, as it's given a wrong reference; the feedback from the path is explicit through the roll angle but is static in the *x,y* coordinates.

This apparent problem also has an impact on the next method.

3.3.4 Tracking Implicit Trajectories for a class of Nonminimum-Phase Systems

As usual, we want to track a predefined output trajectory, maintaining the internal state within a given bound, thus solving all the problems due to its nonminimum-phase characteristics.

Starting from a model with the same appearance of (2.14), page 20, the author defines some coordinate changes in order to obtain a so called *External/Internal form*. This form allows a clear distinction between the internal dynamics (those referred to the roll angle), and the external one (those referring to the vehicle's kinetics); this form can be considered a modification of the known *normal form*, and it has some useful properties, like the possibility to split the analysis in two parts (one for the external and another for the internal part of the system) and the convertibility to a dual structure.

This new form looks like:

$$\begin{cases} \begin{bmatrix} x^{(3)} \\ y^{(3)} \end{bmatrix} = \begin{bmatrix} -2\dot{v}_r s_\vartheta - v_r c_\vartheta \dot{\vartheta} \\ 2\dot{v}_r c_\vartheta - v_r s_\vartheta \dot{\vartheta} \end{bmatrix} \dot{\vartheta} + \begin{bmatrix} c_\vartheta & -v_r s_\vartheta \\ s_\vartheta & v_r c_\vartheta \end{bmatrix} \begin{bmatrix} u^r \\ u^\vartheta \end{bmatrix} \\ \ddot{\alpha} = \frac{g}{p} s_\alpha + \frac{1}{p}(1 + \frac{p\vartheta \dot{s}_\alpha}{v_r}) c_\alpha \dot{\vartheta} v_r + \frac{c}{p} c_\alpha u^\vartheta, \end{cases}$$

where we have used slightly different notations and $u^r := \ddot{v}_r$, $u^\vartheta := \ddot{\vartheta}$.

The idea is to define, according to the possibilities offered by the E/I form, an *Internal and an External Tracking Controller*; these will both be in the fashion of that in section 3.3.1.

Choosing coefficients ϵ_i and ε_i such that the polynomials $x^3 + \epsilon_3 x^2 + \epsilon_2 x + \epsilon_1$ and $x^2 + \varepsilon_2 x + \varepsilon_1$ have roots with negative real parts and having smooth enough referencee trajectories (x_d, y_d) and α_d, we can define:

$$\begin{bmatrix} u^r_{contr} \\ u^\vartheta_{contr} \end{bmatrix}_{ext} = \begin{bmatrix} c_\vartheta & s_\vartheta \\ -s_\vartheta / v_r & c_\vartheta / v_r \end{bmatrix} \left(\begin{bmatrix} -2\dot{v}_r s_\vartheta - v_r c_\vartheta \dot{\vartheta} \\ 2\dot{v}_r c_\vartheta - v_r s_\vartheta \dot{\vartheta} \end{bmatrix} \dot{\vartheta} + \begin{bmatrix} x_d^3(t) \\ y_d^3(t) \end{bmatrix} - \right.$$
$$\left. - \sum_{i=1}^{3} \epsilon_i \begin{bmatrix} x^{i-1} - x_d^{i-1}(t) \\ y^{i-1} - y_d^{i-1}(t) \end{bmatrix} \right)$$

as the *External Tracking Controller* and, as in section 3.3.1,

$$u^{\vartheta}_{contr,int} = \Big(\frac{c}{p}c_\alpha\Big)^{-1}\Big(-\frac{g}{p}s_\alpha - \frac{1}{p}\Big(1+\frac{p\dot{\vartheta}s_\alpha}{v_r}\Big)c_\alpha\dot{\vartheta}v_r + \ddot{\alpha_d} - \varepsilon_2(\dot{\alpha}-\dot{\alpha_d}) - \varepsilon_1(\alpha-\alpha_d)\Big)$$

as the *Internal Tracking Controller*.

Searching for a closed form for α_d, the author applies the *dynamic inverter* to equation (3.5). He thereafter defines an *Equilibrium Manifold* in which the equilibrium point becomes an attractor through the application of the Internal Tracking Controller.

A controller providing tracking of smooth reference trajectories while retaining balance has been obtained. *Issues might be raised concerning computational easiness and load, as well as effective tracking performance.* Indeed, the author quotes the necessity of addressing a *computational-time axis* along with a *dynamic-time one*. Also, it's conceded that the solutions provided aren't *optimal* in any sense. Finally, some doubts concerning the algorithm's robustness near singularities for the inverse kinetics aren't totally eliminated yet.

Remark

It's interesting to ponder whether it's really necessary to obtain such precise dynamical solutions to equation (3.5), coping with evident complexity and computational burden. We'll show in the next sections that approximate solutions can achieve still precise results for the tracking, conceding the fact that they might be not rigorously correct. Thus we can abandon mathematical rigour for speed and easiness in implementation. We'll come back to this point later.

3.4 DAE Control of Dynamical Systems

This work can be synthesized as an extension of the previous research: it accepts many concepts from the work of Getz and still introduces some novelties, mainly regarding the *Descriptor Predictive Control (**DPC** [2])* and the study of obstacle avoiding trajectories for driving through a cluttered environment. The author is Dirk von Wissel.

3.4.1 DPC

This method is a combination of a feedforward and a feedback action. We can state that MPC is a development of this technique. DPC allows to have a state feedback even if the controller is expressed through an implicit formula; this fact makes the system's inversion, often pretty hard for nonlinear systems, almost useless. The control is often computed numerically rather than algebraically,

[2] For a better understanding, please refer to [22].

as a solution to a *Differential Algebraic Equation (DAE)*. As a matter of fact, author's attention is then attracted by the search of some reliable methods for DAE's integration.

3.4.2 Application of the concept

The author shows how the inverse linearization method works mainly just for minimum-phase systems, and how it can be practically applied almost only to linear systems.

The use of DPC can instead be used with any complex dynamical system; here it's used a constant control over the sampling period, eventually updated at each sampling period. It can be demonstrated that the controller's performance equals that of system's inversion, i.e. it's asymptotic. A preliminary feedback is also used to give more strength and robustness to the control.

This method is then employed to control a *Riderless Bicycle*. The model the author addresses is almost the same as the one described in the theory section, eq. (2.15), except that it doesn't neglect the moment of inertia for steering. A so called *separated form* (i.e. highlighting the distinction between cinematics and dynamics) is privileged.

An open-loop, *bang-bang control* is then proposed for making the bicycle turn a prescribed angle. This is actually applied only to the bicycle's cinematics, thus we have always to check for balance: the two-points boundary value problem turns out to be quite expensive.

It's way better to implement an asymptotic tracking for the roll angle, in the fashion of Getz's, through the use of DPC.

The second step is tracking for x, y paths with non-zero velocities, retaining balance. The previous approach is again used, although conceived as a DPC, rather than a feedback linearization. This works in a slightly more general case than in Getz's, but still raises many doubts on the actual implementation. Additionally, no formal demonstration of the stability is given.

3.5 Current Algorithms for Controlling a Motorcycle

In the present section we'll introduce the reader to some ideas which, implemented in some current software, have allowed to control some NonMinimum Phase System, and more precisely a Motorcycle.

3.5.1 A Fuzzy Approach

This very effective approach distinguishes between different kinds of control schemes, implementing three diverse controllers: the first is a *longitudinal controller*, intended to regulate the vehicle's speed, the second is a *lateral controller* which

presides over the trajectory following, while the last one is a *balance controller* which, in the fashion of Getz's approach, provides stability, avoiding the bicycle to fall. There's also a so-called *yaw controller*, which acts in order to prevent the vehicle from falling under unlikely situation, as for instance abrupt accelerations or brakes, or sudden lateral disturbances; we won't focus on this very last part, though. All the elements are eventually clustered to work together through an intelligent fuzzy regulator.

3.5.2 Connection Trajectories on the flat output

Let's start with the core part of the controller, i.e. the *lateral* section: here some ideas must provide the tracking of the desired path. Let's assume we have out motorcycle slightly aside of the path, running with a certain velocity which we can assume to be constant without any loss of generality (see figure 3.3).

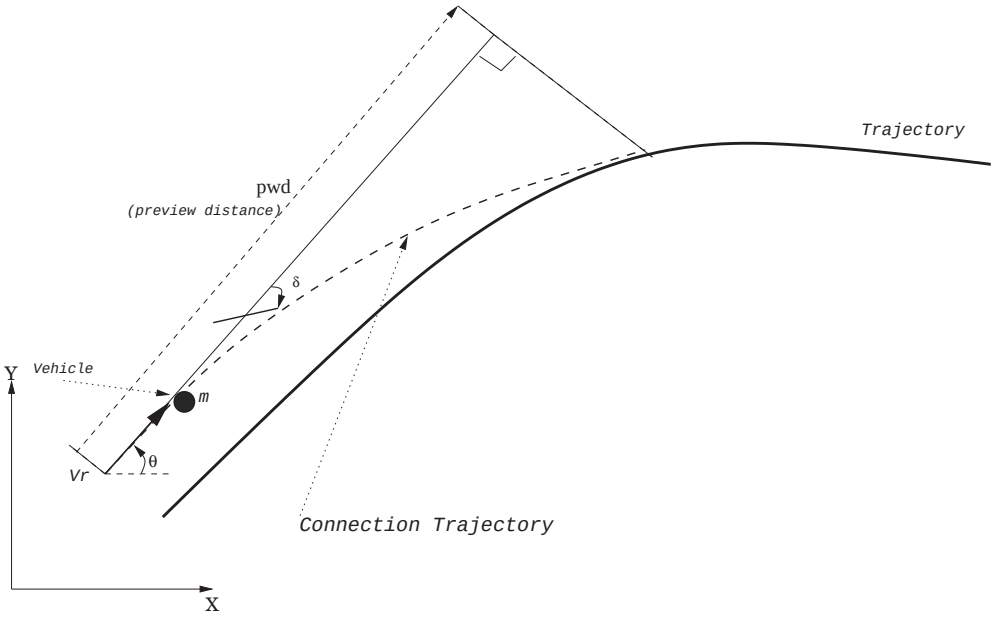


Figure 3.3: Project of a Connection Trajectory

Now, we obviously want the bicycle to converge to the path. It's very adherent to the physical reality to stare ahead of the vehicle at a particular distance, usually dependent of the current speed. We shall henceforth name this distance ***preview distance***, or **pwd** for short.

It's even more realistic to put in relation the point ahead of the driver at the *pwd* with a nearby point on the path; here many approaches can be thought of (for instance, the point minimizing the Euclidean distance, or the orthogonal projection as in figure). The idea here is to define a *connection trajectory* among a pool of functions, bringing from the current state to the defined point ahead. As we have already studied in the theory section, a trajectory in the x,y plane can be conducted to a function dependent of the curvature σ of the vehicle. From the

curvature we can easily derive a time-dependent function for the steering action β through $\sigma = \tan(\beta)/b$, where b is the *bike wheel base*. We can apply this control for any time we want, taking into consideration that we'll be in open-loop, and recalculate this at any time we want, thus giving as a feedback the state of the motorcycle [3].

What kind of *connection trajectories* are more suitable? The first idea is to deploy some polynomials; we have to keep in mind that the motorcycle has the following states, at a precise time t^*: $\alpha(t^*)$,$\frac{d\alpha(t^*)}{dt}$ and $\alpha(t^* + T)$. Therefore, if we consider a third order polynomial, we determine three of them as given, and we use the fourth one as an *optimization index*.

Considering the point lying on the trajectory calculated as in figure 3.3, we can define a weighting function based on the Euclidean distance between this position and the final point of all the polynomials belonging to the predefined set.

The attentive reader might have noticed the trend: we apply a kind of MPC, in that we employ an optimization function to apply an open-loop control, based on the state of the system as a feedback.

We shall refresh this trajectory with any frequency we desire, this would be a project parameter as in MPC. Also, the *preview distance* can be conceived as a project parameter: we can build up a more foresighted controller (longer *pwd*), as well as a more reactive one (shorter *pwd*).

This approach seems certainly to be very convenient, and indeed the idea works out pretty well, except that it does not resolve the issue of stability.

3.5.3 Bank Angle Stabilizer

Let's undertake Getz's approach for the *roll angle stability*, i.e. state-feedback linearize the famous eqn. (2.15), page 21, to obtain exponential convergence to the desired roll-angle trajectory. In this case the idea is also to determine the desired α with an easier formula, not having to make use of the *dynamic inverter* and therefore speeding up all the calculations. A problematic issue naturally raising here, as remembered before, is the choice for the coefficients rendering the polynomial with negative real-parts roots.

3.5.4 Put the Pedal To the Metal

The last part of the regulating scheme is the *longitudinal controller*. We want out vehicle to match a certain desired speed, possibly time varying [4].

[3] Here please note that the *connection trajectory* depends from the vehicle's state, i.e. from its x,y,ϑ positions as initial conditions.

[4] In reality the software product takes into account the possibility of changing gear, with the consequent speed modifications. We won't report any detail here, as this would go beyond the scope of the current paper.

The approach, similar to the one used for the lateral regulator, defines a *speed preview distance* and devises a first order polynomial as the connection function, i.e. applies a *proportional and derivative* control to obtain the desired velocity at the *speed-pwd*.

3.5.5 The Fuzzy Approach

It's necessary to "blend together" the three stages. Disregarding the third one, which stands by itself as it just acts on the velocity derivative, let's focus on the first two (having as the actuator the steering shaft). The idea here is to make the *lateral controller* act more in case the motorbike loses the path or in case of a curve, while the *roll stabilizer* is activated through a dynamic weight, inversely proportional to the second derivative of the roll.

3.5.6 Analysis and Evaluation

The method clearly detaches from Getz's approach, which as far as now has been the only one giving rather good and ground-breaking results. The heuristic works indeed quite well, especially regarding robustness and stability. Still, the fuzzy approach looks from the beginning too much tailored to the specific problem, and it poses the question whether it will work for any manoeuver. Also, the choice of the two coefficients for the state-feedback control are left to the end user and looks too much empiric.

One main drawback is given by the choice of the polynomials as the connection trajectories: they're reasonable as for easiness and to avoid computational bottlenecks, but they almost always represent *unfeasible trajectories*. We shall come back to this point describing how to entangle this bundle with the proposed new approach. We'll also describe later (read section 4.2.2) the main cause of this failure: **countersteering**. Let's say, just for now, that the method isn't theoretically wrong only for restricted, smooth cases, like straight motion, or circular curves.

The reader should be aware of the fact that, in case the trajectory isn't feasible, the control we're applying through the term σ in equation (2.15) is leading the roll to an unstable state, which we'll have to steer back to the equilibrium through the application of the state feedback. This means a higher calculus load, as well as a theorically incorrect approach.

Just a couple of words for the *longitudinal controller*: it's not very effective, particularly in the case of high frequency changes in the real velocity, like those due to the gear shift. Conceding that the idea of making use of a *preview speed* may look like being quite appealing, we have to underline that more straightforward implementations would lead to definitely better results. This will lead us to redefine this part in later parts of our work.

Chapter 4

The New Method

4.1 The setting

In this chapter, a new control methodology will be proposed.

To first accomplish this, we must first define exactly which side we have tackled our task from. It is necessary to make a distinction between *trajectory tracking* and *path following*.

The former problem concerns a reference trajectory which isn't fixed a-priori, but has indeed proper and partially unknown dynamics; let us assume that we have a *causal* knowledge of this reference trajectory that can possibly be affected by some sort of noise, which is said to be wholly referred to the input. Noise can arise from a limited knowledge of the output (as in vision for example), as well as from any sort of signal corruption. This setting spontaneously paves the way for the application of statistical filtering.

With path following, conversely, we're given a deterministic trajectory, or path, which can actually be time-dependent, i.e. non-autonomous, but which can be possibly fully known a-priori. We shall not be forced to make use of all this information at any time step, but this eliminates any necessity to use stochastical control. Moreover, noises are said to be referred to the output.

Our problem is limited to the second setting, i.e. we shall be interested in ***Path Following for NonHolonomic, NonMinimum Phase Systems*** through the application of Nonlinear Control Methodologies. This task-horizon isn't limited at all, as too few methods can be found in literature regarding these problematics, with also questionable results. A possible shift toward the first issue might make the actual problems we want to deal with be partly concealed by other difficulties, thus let's stick for the moment to this second task. Results can be subject to future extension.

4.2 The Main Idea

So far we have analyzed mainly three methods providing working results: Getz's, von Wissel's and the one based on *Connection Trajectories*. Robust applications to actual software products have worked mainly only through the first and the third methods. The first one has been theorically proven to be stable, even if the robustness is still questionable. Moreover, it suffers from Computational Burden issues. The third method has performed rather well, especially regarding its robustness (which is actually quite astonishing), although the fuzzy approach looks pretty blurry. Also, no stability has been yet proved (and in our opinion the fuzzy structure makes this last quite a tough task!).

The new approach proposed throughout this paper encompasses some of the very smart concepts from the third method and almost skips all the theory (and the calculations) which underlies the first approach, still retaining some of its key features.

Summarizing it in short, **it eliminates all the problems of roll balancing**.

This result is obtained *devising connection trajectories for the system's roll angle*. More precisely, we make use of the concept of a preview horizon, although we are specifically interested in a *preview time*, rather than a *preview distance*. Defining opportunely the desired equilibrium roll angle at the preview time, and exploiting this roll and possibly any number of its derivatives as the vehicle's state, we can devise some *connection trajectories* for the roll itself.

This new yet simple idea provides a simplification for many problems we had previously described and attains some new results and performances, as we'll explain and thoroughly describe in the following sections.

4.2.1 Path Definition

As said before, the desired path is predefined as a function of the curvilinear abscissa and the curvature radius. We assume that the desired velocity v_d is constant[1], therefore we can state that the path is a function of time and again the curvature radius. Assuming an initial kinetic condition for the bicycle, for instance $x_0 = 0$, $y_0 = 0$, $\vartheta_0 = 0$, we obtain the following path on the (x, y) coordinates:

from $\dot{\vartheta} = \sigma v_d$,
we have $\vartheta(t) = \vartheta_0 + \int_0^t \dot{\vartheta}(\tau) d\tau$;
moreover:
$x(t) = x_0 + \int_0^t v_d(\tau) \cos(\vartheta(\tau)) d\tau$,
$y(t) = y_0 + \int_0^t v_d(\tau) \sin(\vartheta(\tau)) d\tau$;

this is a classical definition of a curve in the (x, y) plane in function of time[2].

[1] As we shall see in upcoming paragraphs, this assumption can be eventually relaxed.

[2] Please note that we could keep it defined w.r.t the curvilinear abscissa, still obtaining the

4.2.2 Devising Connection Trajectories

As previously stated, the new method makes use of an MPC Heuristics, and in doing so it has to exploit the idea of *Connection Trajectories*. This time, however, we'll refer to this concept applied not on plane trajectories (or, in other words, trajectories devised directly for the steering action), but indeed on roll trajectories, i.e. referring to the vehicle's bank angle.

Let's first go back to illustrate a concept which will enable the reader understand why this new idea is much more convenient than the one exploited in the previous approach.

The Countersteering Phenomenon

A subtle reasonment over the equations for the balance of the motorcycle makes us understand a specific phenomenon due to the nonminimum phase characteristics of the vehicle: *Countersteering*.

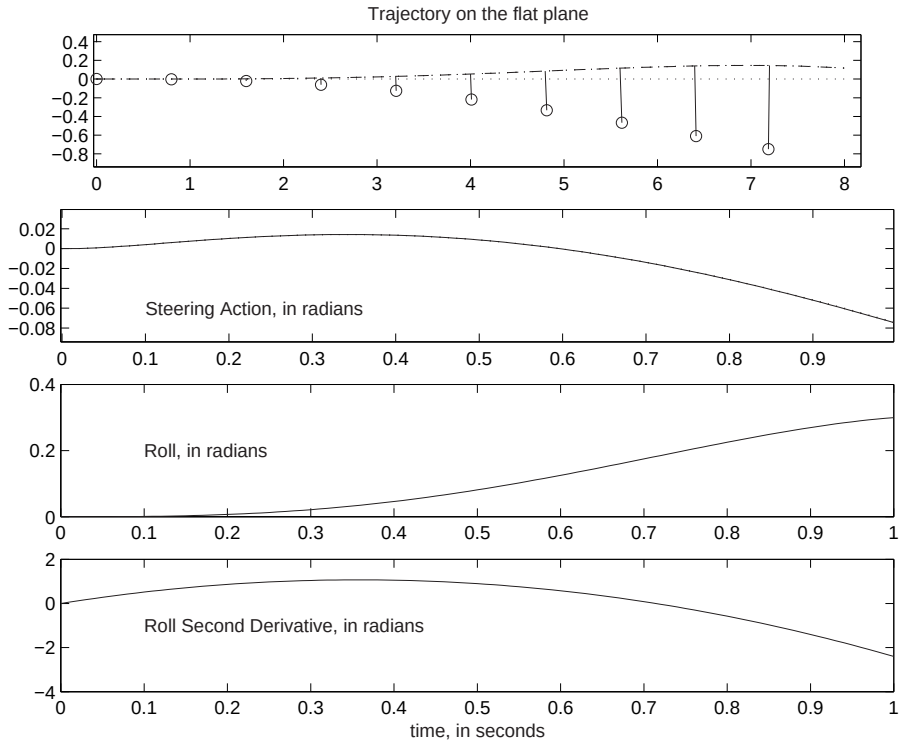


Figure 4.1: The Countersteering Phenomenon.

Roughly speaking, countersteering involves rotating the driving shaft opposite to the direction the driver wants the vehicle to go to. For instance, starting from an upright position and willing to bend the vehicle towards the right, one has first to make a (slight) turn to the left, and then converge to the desired direction.

very same curve on the flat plane.

Also, dynamically starting from any steer angle, with the corresponding roll equilibrium position, this behavior is still required to leave this equilibrium situation. Often this manoeuver is not practically performed, for instance while driving a bike, as the driver uses his body to lean the vehicle in the desired direction. Still, this behavior can be seen while observing an experienced motorcycle driver, like during pro races.

All this has naturally a mathematical explanation, which we're going to describe in the following. Taking into consideration the classic equation (2.15) and assuming that v_r is constant, i.e. $\dot{v_r} = 0$, we obtain:

$$\begin{aligned} 0 &= gs_\alpha + c_\alpha \sigma v_r^2 + c_\alpha s_\alpha p \sigma^2 v_r^2 + c c_\alpha v_r \dot{\sigma} \\ &= gs_\alpha + c_\alpha s_\alpha p \sigma^2 v_r^2 + c_\alpha v_r (\sigma v_r + c\dot{\sigma}); \end{aligned} \tag{4.1}$$

now, considering the case $\alpha > 0$ and small enough, we see that the first two terms are always greater than zero and that the equation can be solved only if the last term in parentheses is negative; in other words, we require that

$$\sigma v_r + c\dot{\sigma} < 0.$$

If we assume we started from an upright position, i.e. $\sigma = 0$, we see the equation admits a solution in the case $\dot{\sigma} < 0$; in other words we have to first steer contrary to the required direction of motion.

Same thing happens in the opposite case, i.e. when we want to reach a roll angle with opposite sign.

In figure 4.1 we plotted a MATLAB output, referring to the following situations: $\alpha_0 = 0$, $\dot{\alpha_0} = \ddot{\alpha_0} = 0$, $\sigma_0 = 0$, *preview time* $T = 1[s]$ and $\alpha_T = 0.3[rad]$, $\dot{\alpha_T} = 0.2[\frac{rad}{s}]$; a fourth order polynomial was deployed as *connection trajectory* (read later sections for further detail thereon).

As quite evident from the plots[3], starting from an upright position over the reference trajectory (dashed straight in the upper graph), we want to bend right, in order, say, to turn right as well. To do that, before turning right (second plot), we have to first slightly steer left: the effect of this action is a steady shift of the roll angle toward positive values (third plot). The overall outcome on the vehicle's dynamics is a slight turn on the left, before bending to the right with the correct inclination.

A deeper analysis of the equation may show that this is valid under general circumstances, i.e. starting from any position of the steering shaft and any (corresponding) bank angle.

For instance, considering figure 4.2, we are in the situation: $\alpha_0 = -0.08136[rad]$, $\dot{\alpha_0} = \ddot{\alpha_0} = 0$, $\sigma_0 = 0.0125[rad]$ [4], *preview time* $T = 1[s]$ and $\alpha_T = 0[rad]$, $\dot{\alpha_T} = 0[\frac{rad}{s}]$; a fourth order polynomial was deployed as *connection trajectory*.

[3] Please get acquainted with the reference frames for the angles involved reading section 2.1 and observing figure 2.1.

[4] The motorbicycle is assumed to run at $8\frac{m}{s}$ in a curve of radius 80 m $=\frac{1}{0.0125}$ m; the relative roll angle has been obtained through the solution of equation (3.5).

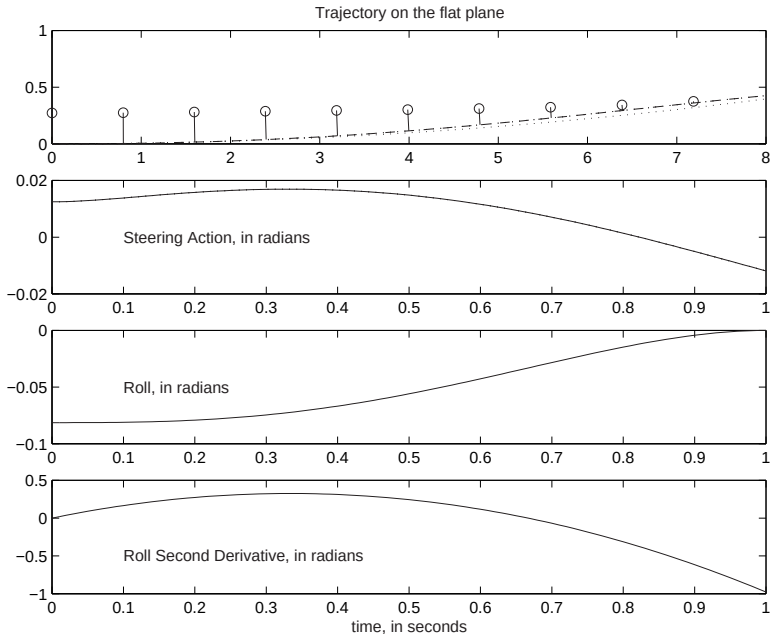


Figure 4.2: General Validity of the Countersteering Phenomenon.

As evident from the second plot, it's necessary to steer even further left to regain balance on the roll angle, before bending to the right.

Why is Countersteering so important?

The reader might remember that we called the *Connection Trajectories on the Flat Output* in one of the previous approaches (check Section 3.5.2, page 37) ***Non Feasible***. The Coutersteering effect justifies and explains this nickname. Indeed, we described the use of very simple functions, like polynomials (more precisely paraboloids) as connection trajectories: it's clear now that they aren't suited to describe with their shape this quite complex effect. Therefore they are often inadequate to obtain a theorically correct steering action. Moreover, they tend to steer the motorcycle to an unstable state (exactly in the opposite side), as well as to a further point in the plane.

The reader has to ponder that this problem doesn't affect Getz's or von Wissel's approaches, which always present a correct behavior from this point of view [5].

The motivations to use Connection Trajectories for the Roll Angle should now be more clear: given the uneasiness of the feasible functions for the steering, it's proven by experience that, while driving, the driver performs easy and smooth rolling movements. Consequently, we can exploit relatively easy to define functions, like polynomials, and be sure that they'll match pretty closely the actual

[5] The convergence to the desired path is indeed exponential (albeit only to a desired region closed to the final point), while retaining balance for the vehicle.

ones. Also, the reader might notice that the Countersteering phenomenon doesn't affect the roll, but it's indeed a feature of the steering dynamics.

4.2.3 How to disregard the Stability Problems all at once

The choice of working out Connection Trajectories for the Roll has another major basis: it allows us to avoid all the problems related to the vehicle's stability issues. To understand this, we make use of ***Bèzier Curves*** *as Connection Functions*: they're indeed no more computationally complex than a simple polynomial [6] and they also have some more properties which arise from their structure(see section 2.7).

From now on, we'll consider a four-points function: the first one is the vehicle's state at time t^*, $\alpha_0 = \alpha(t^*)$, the last is the desired roll at the preview time T, $\alpha(t^* + T)$; the other two points will be the *control points*. We might associate these points with other dynamic states for the vehicle, as well as consider them as simple points we can act upon (see figure 4.3).

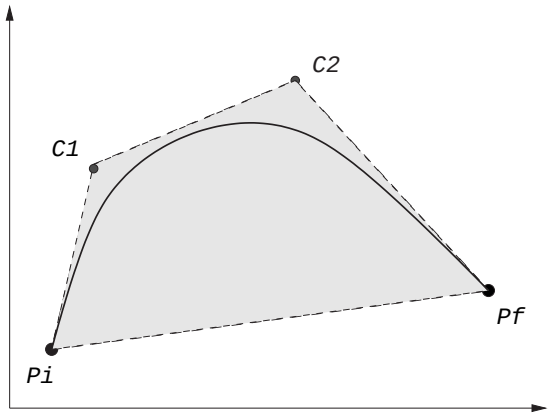


Figure 4.3: A simple Bèzier Curve, controlled by the two points C_1 and $C2$.

Now, considering $t^* = 0$, we develop the following relations, which could be

[6] Indeed, they're just polynomials expressed in a particular form.

obtained formally through the theory as well:

$$\begin{aligned}
being \quad & P_0 = P_i \quad , \quad P_3 = P_f, \\
and \quad & P_3 = C_1 \quad , \quad P_4 = C_2; \\
for \quad & 0 \leq t \leq T, \\
P_0^{'}(t) = \quad & P_0(t)t + P_1(T-t) \\
P_1^{'}(t) = \quad & P_1(t)t + P_2(T-t) \\
P_2^{'}(t) = \quad & P_2(t)t + P_3(T-t); \\
P_0^{''}(t) = \quad & P_0^{'}(t)t + P_1^{'}(T-t) \\
P_1^{''}(t) = \quad & P_1^{'}(t)t + P_2^{'}(T-t); \\
P_0^{'''}(t) = \quad & P_0^{''}(t)t + P_1^{''}(T-t);
\end{aligned} \tag{4.2}$$

we obtain through substitution

$$P_0^{'''}(t) = P_0 t^3 + 3P_1 t^2(1-T) + 3P_2 t(1-T)^2 + P_3(1-T)^3.$$

We shall then consider from now on equations for the roll like the following one, simply based on coefficients:

$$\alpha(t) = \eta_0 t^3 + \eta_1 t^2(T-t) + \eta_2 t(T-t)^2 + \eta_3(T-t)^3 \tag{4.4}$$

Now, it should be clear that if the vehicle is balanced and is required to reach a possible state at the preview time, imposing bounds on the two *control points* will ensure that the dynamics for the roll will stay bounded. This is quite an important point: ***if we start working directly on the vehicle's roll, we can skip all the balancing procedures devised for the preceding methods.***

We'll define these coefficients in the following sections. Let's first give an explanation on how these functions are intended to be exploited.

4.3 MPC in action: the receding horizon and the state feedback

As already stated, the *Model Based Predictive Control* algorithm can be summarized in the following steps:

1. At time t^* devise a control action over some future interval (we'll directly refer to the aforementioned *preview time* T henceon);

2. Apply the first step of this control operation in an open loop fashion;

3. Refresh the *kinetic and dynamic states of the vehicle* and work out a control scheme using this information as *feedback* [7];

4. Skip to step 2.

The reader might understand that after the very first step the control scheme closes the loop, thanks to the feedback we exploit from the vehicle's current states. That's the important point about MPC: *it encompasses a rather simple algorithmic scheme with the powerful effects of feedback.*

Now, we still have to understand which can be regarded as the states of the vehicle. Let's devote a whole paragraph for that.

4.3.1 The Vehicle's States and the Connection Trajectory

The first kinetic states for the vehicles are very naturally its position and orientation, or *yaw*; thus, at time t^*, we'll make use of the variables x, y, ϑ.

As for its dynamic states, it's up to us to decide how many of them to exploit. At time t^* we'll look at $\alpha(t^*)$ and any time derivatives of it, as requested by equation (2.15).

We then have to define once and for all how to obtain the desired roll angle at the preview time $t^* + T$.

Solving the equation for the Desired Roll

As we want to obtain balance, we'll look at the equation

$$0 = gps_\alpha + (1 + p\sigma s_\alpha)pc_\alpha\sigma v_r^2 + cpc_\alpha(v_r\dot{\sigma} + \dot{v_r}\sigma), \tag{4.5}$$

as previously discussed in the theorical settings.

At this time we have to decide how to solve this equation: we can either embrace Getz's method, estimating the *dynamical inverse*, or propose some simplifications. Conceding that the approach proposed in section 3.3.1 is theoretically definitely more correct (and is indeed mandatory in case we want to demonstrate the stability of our approach), we shall nevertheless focus on a slight variation of the equation. We'll be indeed interested in obtaining a very simple and closed-form solution, which could be utilized running through the whole algorithm.

Let's first observe that, after having singled out each term in (4.5), the one which can be easily disregarded is

$$p\sigma^2 s_\alpha c_\alpha v_r^2$$

[7] Under the case that the information from the vehicle isn't exactly know, we might still use certain kinds of estimation of it; this will be the case for the desired roll at the *preview time*, when we'll not solve the differential equation exactly, but indeed we'll simplify it conveniently.

because of the presence of the term s_α and the square of σ. Eliminating the terms p, we obtain the simplified relation

$$0 = gs_\alpha + c_\alpha \sigma v_r^2 + cc_\alpha(v_r\dot{\sigma} + \dot{v_r}\sigma). \tag{4.6}$$

Dividing by c_α[8] and separating the term in α, we get

$$\tan(\alpha) = -\frac{1}{g}\sigma(v_r^2 + c(\dot{\sigma}v_r + \sigma\dot{v_r})), \tag{4.7}$$

which offers an easy solution,

$$\alpha = -\arctan\Big(\frac{1}{g}\sigma(v_r^2 + c(\dot{\sigma}v_r + \sigma\dot{v_r}))\Big). \tag{4.8}$$

It can be shown that the difference between this solution and the one obtained through dynamical inversion is, for nonpathological cases, less then one degree. Thus obtaining this last solution is not worth the effort. We could even obtain a more simple form disregarding the derivatives in the preceding form, still without making a sensible mistake, but we'll stick to equation (4.7) from now on.

The reader might remember that Getz proposed a method for obtaining the time derivatives of the dynamical inverse. In case we'll be interested in deriving equation (4.8), we'll naively proceed in that: might this seem quite awkward in theory, it doesn't anyways produce any problem when solved through the usual numerical calculus algorithms[9].

Power is nothing without Feedback!

We have seen that the MPC approach requires the use of connection functions for the bank angle, as well as the *feedback action* of the vehicle's dynamic states. Here we'll see how these two aspects blend.

Taking into consideration functions like (4.4), we have to specify the coefficients. We can easily do that through the vehicle's states and its derivatives, as well as thanks to the desired roll plus its derivatives.

Considering a third order function,

$$\begin{aligned}\alpha(t) &= \eta_0 t^3 + \eta_1 t^2(T-t) + \eta_2 t(T-t)^2 + \eta_3(T-t)^3 \\ &= (\eta_0 - \eta_1 + \eta_2 - \eta_3)t^3 + (\eta_1 - 2\eta_2 + 3\eta_3)Tt^2 + \\ &\quad +(\eta_2 - 3\eta_3)T^2 t + \eta_3 T^3\end{aligned} \tag{4.9}$$

and deriving it,

$$\begin{aligned}\frac{d\alpha(t)}{dt} &= 3(\eta_0 - \eta_1 + \eta_2 - \eta_3)t^2 + 2(\eta_1 - 2\eta_2 + 3\eta_3)Tt + \\ &\quad +(\eta_2 - 3\eta_3)T^2\end{aligned} \tag{4.10}$$

[8] Which makes sense as this term is not null in any ordinary situation.

[9] We refer here to the widely known methods of Euler, or Runge Kutta - 4^{th} order.

and assuming without loss of generality $t^* = 0$, we can regard as *vehicle's states* its current roll and its derivative, i.e. $\alpha(0) = \alpha_0$ and $\dot{\alpha}(0) = \alpha_1$. As already stated, solving for α in equation (4.8), we attain the *desired roll at the preview time* T, i.e. $\alpha(T) = \alpha_2$.

In conclusion we get the following system:

$$\begin{aligned} \eta_3 T^3 &= \alpha_0 \\ (\eta_2 - 3\eta_3) T^2 &= \alpha_1 \\ \eta_0 T^3 &= \alpha_2 \end{aligned} \tag{4.11}$$

In other words, this system specifies three of the four coefficients in our connection trajectory for the roll. It's our choice to look at higher degree functions, or to a larger number of states for the vehicle (and more precisely to the time derivatives of the quantities above), or to any combination of the two quantities, given that *the degree of the function is higher than or equal to the number of the states we examine.*

The choice of considering more states implies requesting a more precise and smooth definition for the connection function, i.e. a trajectory which could be precise also "at higher orders". The drawback of considering more states is surely the higher number of calculations we'd be required to perform. Hands on the project, the vehicle is supposed to output more information, and we have to be able to process it on the fly.

In our example, we still have to justify our choice of leaving one of the coefficients undetermined a priori. Let's understand why.

4.4 MPC in action: the Optimization Step

We realize what is now missing to complete the puzzle: the remaining coefficient plays the role of adding one degree of freedom, as well as bounding the function inside a desired region, namely such that the vehicles doesn't approach near-to-fall situations. Simple calculations would show how this is possible. In other words we exploit this coefficient as if we could directly act upon the two control points of the Bézier Curve, and, by the theorem mentioned in section 2.7 in page 26, we're sure to obtain limited dynamics.

Combining the two main ideas we have conveyed so far, that is deploying Bézier Curves for the bank angle to bound their dynamics and using the power of feedback, we practically see that each time we work out a connection trajectory, we indeed obtain *a limited set of them, or an ensemble* (see figure (4.4)), which is parameterized by the unknown left coefficient(s). Refer for instance to figure (4.4) for a set of connection trajectories (the fact that they don't intersect is not casual, as we shall see soon).

More precisely, we see how this coefficient changes the shape of the function, still maintaining the end conditions.

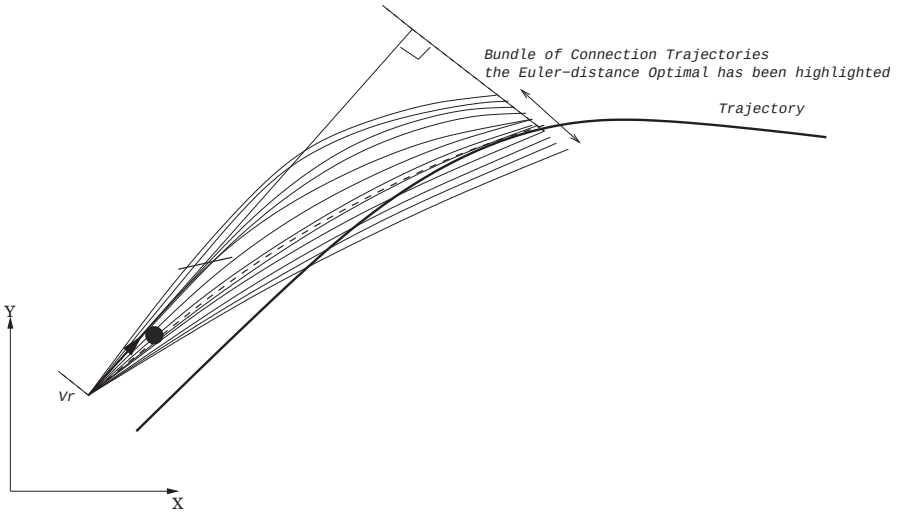


Figure 4.4: Ensemble of Connection Trajectory: we choose the optimal inside this pool.

4.4.1 Time Parameterization

A rational way to influence the outline of the function is to exploit a time parameterization. For the classical Bèzier Curve just considered, the time evolves linearly; how would we reshape this evolution into a nonlinear behavior, as portrayed in figure (4.5)?

In other words, having a function for the roll $\alpha(\tau)$, we could think of using another function for the time, like $\tau = \tau(t)$.

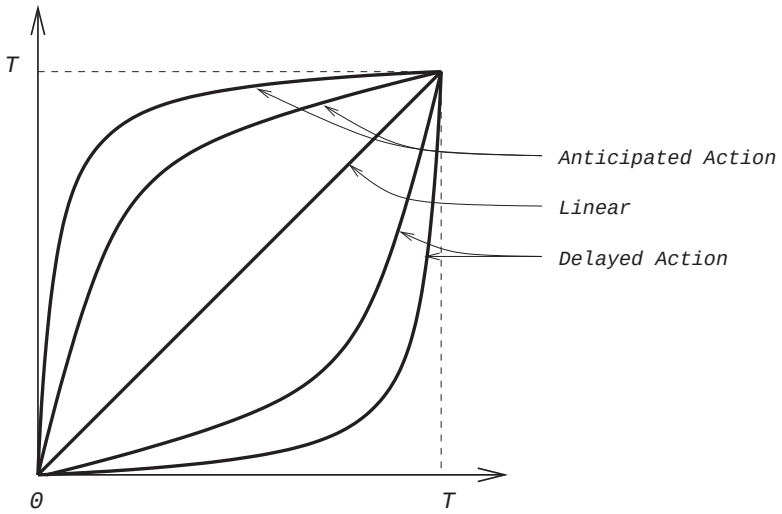


Figure 4.5: Time Reparameterization

That would mean influencing at the very end the steering action. Therefore we can have the left coefficient depend on the kind of control we might want to

apply, for instance more steady or conversely smoother. More precisely, in case the parametrization function has a small derivative in the first instants, we'll know that the control is going to be rather calm initially; conversely, in case the same function increases abruptly, we know we'll be applying a brusque steer. Keep in mind that eventually we'll be effectively applying only the first step of the steering action, as in the MPC fashion; still, experience shows that it's the whole outline of the connection trajectory that influences the general behavior of the control algorithm.

From this perspective, the different functions belonging to the ensemble would mean a diverse kind of control action. How can we single out the most suitable option?

4.4.2 The Performance Index

Let's go back for a moment to the dynamics equation (2.15) already considered. It has been conceived as a nonlinear differential equation for α so far but, as already said, it can also be comfortably seen as a relation w.r.t σ, being the term α known in advance.
Assuming we can somehow obtain a correct solution to this equation through the knowledge of $\alpha(t)$, $t^* \leq t \leq t^* + T$, we can now manipulate a function $\sigma(t)$, $t^* \leq t \leq t^* + T$.
Thus, from the relation,

$$\dot{\vartheta} = \sigma v_r$$

and knowing the state $\vartheta_0 = \vartheta(t^*)$, we can obtain

$$\vartheta(t) = \vartheta_0 + \int_{t^*}^{t} \sigma(\tau) v_r(\tau) d\tau, \qquad t^* \leq t \leq t^* + T.^{10}$$

Similar to what developed in section 4.2.1, we also obtain for the motion of the vehicle on the plane:
for $t^* \leq t \leq t^* + T$,

$$\begin{aligned} x(t) &= x(t^*) + \int_{t^*}^{t} v_r(\tau) \cos(\vartheta(\tau)) d\tau; \\ y(t) &= y(t^*) + \int_{t^*}^{t} v_r(\tau) \sin(\vartheta(\tau)) d\tau. \end{aligned} \tag{4.12}$$

Now, keeping in mind that we've assumed v_d constant, we can relate the points on the desired path at time $t^* + T$, say $x_d(t^* + T)$ and $y_d(t^* + T)$ and the actual points $x(t^* + T)$ and $y(t^* + T)$. Let's therefore define:

$$J(t^*) = (x_d(t^* + T) - x(t^* + T))^2 + (y_d(t^* + T) - y(t^* + T))^2.$$

[10] Remember that in our case we have assumed that the velocity is even constant.

This is proven to be a *Convex Performance Index*. Note from figure (4.4) that, if we use this method and be precise in integrating the equations, we obtain a bundle of non-intersecting curves; this means that the optimization index, depending on the Euler distance, will be intrinsically convex, having one single global minimum.

Matching all the paradigms of MPC, we'll choose the undetermined coefficient(s) that minimizes this Index, that is the index which corresponds to a trajectory for the roll which refers to a steering action which brings the closest possible to the desired points in space.

This corresponds in other words to the *Optimization Step*.

4.4.3 Suitable Functions for the Time Parametrization

Backtracking to the problem of devising a suitable *time parametrization function*, we face the problem of choosing on which might respect the boundary conditions, i.e. the states for α at t^* and $t^* + T$.

For instance, in case we wanted to pick un some degree-varying polynomials,

$$\tau(t) = t^{\nu}, \qquad -\infty < \nu < +\infty,$$

we would have problems under the case $|\nu| < 1$, as it time derivative would boost for $t = 0$. In other words, after plugging this time parametrization into the expression for α, we would have an unstable function for $t \longrightarrow 0$. The other boundary values could be kept the same through convenient choice of the coefficients.

A second idea is to make use of a rescaled arctangent, but calculations might show that, choosing coefficients such that the initial conditions for the derivatives at t^* aren't modified, we can't avoid to affect negatively those at $t^* + T$.

The probably best choice appears to be the easiest: exploiting a low order polynomial with null zero-grade coefficient. For the sake of example, in the case we desired to reparameterize a cubic polynomial with one free parameter (i.e., defining three out of four coefficients), we'd need a second order polynomial, that is a parable, with null zero-order coefficient:

$$\tau(t) = \epsilon t^2 + \varepsilon t;$$

we'll further on define the two parameters through conditions on $\dot{\tau}(t^*) = \varepsilon$ and $\tau(t^* + T) = \epsilon(t^* + T)^2 + \varepsilon(t^* + T)$.

4.4.4 Mimicking the Time Parametrization

Instead of making use of time parametrization, we could create out trajectories' bundle in other, more exotic ways.

A first idea has been to *modulate* part of the connection trajectory through a *cosinusoidal function*, just to boost the first part and smoothly damp the foregoing

behavior. The time span during which this parametrization acts is variable, and actually represents a parameter.

Another idea has been to start from a piecewise linear polynomial to be further smoothed out in its connection points: the very first part would be a straight line with inclination corresponding to $\dot{\alpha}(t^*)$, while the last one would be a horizontal line at the height $\alpha(t^*+T)$, or even a straight line slanted according to the value $\dot{\alpha}(t^*+T)$. In between, there would be a connection polynomial controlled at the starting and ending points. In figure 4.6 this is depicted clearly: the control points here are $C1$ and $C2$.

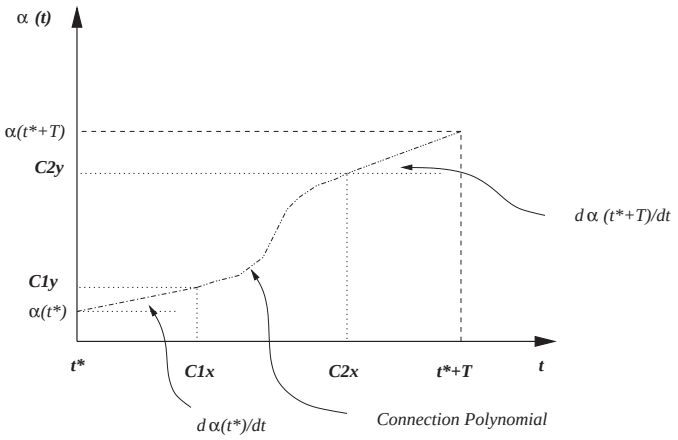


Figure 4.6: Connection Trajectory for the Roll: another example.

4.4.5 Degree of Precision

The attentive reader could question which (dynamic) *States* to consider for the vehicle; indeed, it can be still considered a state any time derivative of α at time t^*. From the preceding discussions, it can be understood that it would be compelling to increase the degree of the connecting polynomial to keep in consideration all those states [11].

It can be shown that, examining a higher number of states, we can attain a more precise control, i.e. we can be much more informed about the outcome of the control-to-be. For example, it can be proven, as well as derived staring at equation (2.15), that the properties for σ are reflected onto those for $\ddot{\alpha}$. As a matter of fact, if we express that term in function of σ, we have:

$$\dot{\sigma} = \frac{p\ddot{\alpha}}{cc_\alpha v_r} - \frac{gps_\alpha}{pcc_\alpha v_r} - \frac{\sigma \dot{v}r}{v_r} - \frac{\sigma v_r}{c} - \frac{c^2 ps_\alpha v_r}{c}$$

Reasoning over the magnitude of the terms comparing in the expression, we can discern that, through convenient actions on $\ddot{\alpha}$, we can influence the behavior

[11] Indeed, we've mentioned that the function's degree has to be higher or at most equal to the number of states.

of the first term. Check figures 4.1 and 4.2 to have some physical justification to that.

Thus, in the process of optimizing for $\ddot{\alpha}$, we can be sure about what we're going to attain for σ. This can turn out to be of a certain help.

As the reader might have recognized, the disadvantage is always either computational and physical[12]. Also, after a particular limit, the differences between all the controls appear to be too small to take care of, meaning that they correspond to similar steering actions.

4.5 Speeding up the Algorithm: A Hamiltonian Approach

We have already highlighted the fact that, given a *connection trajectory* for the roll, α_{opt}, we're supposed to solve the classical equation (2.15) to obtain the *control action* in terms of σ. Moreover, we have to pass through all the integrations to attain the vehicle's position at the *preview time* during the optimization process.

There's a canny mathematical trick to boost this whole process. Let's first dub the product

$$\sigma v_r = u.^{13}$$

Equation 2.15 now looks like

$$p\ddot{\alpha} = gs_\alpha + c_\alpha u v_r + pc_\alpha s_\alpha u^2 + cc_\alpha \dot{u}; \tag{4.13}$$

we can divide all the members by cc_α, given that $\alpha \neq \frac{\pi}{2}$ (which is always the case for normal situations),

$$\frac{p\ddot{\alpha} - gs_\alpha}{cc_\alpha} = \frac{v_r}{c}u + \frac{p}{c}s_\alpha u^2 + \dot{u}; \tag{4.14}$$

let's define

$$u = \frac{q}{s}, \qquad where \quad s \neq 0, \quad \forall t;$$

thus the derivative is

$$\dot{u} = \frac{\dot{q}s - \dot{s}q}{s^2}. \tag{4.15}$$

Plugging these two terms into (4.14) and proceeding with some manipulations we get

$$s^2\frac{p\ddot{\alpha} - gs_\alpha}{cc_\alpha} = \frac{v_r}{c}qs + \frac{p}{c}s_\alpha q^2 + \dot{q}s - q\dot{s}. \tag{4.16}$$

[12] For this second case, we have to have a sufficient number of sensors on the vehicle for detecting all the states.

[13] This notation resembles one used in Getz's work, and turns out to be indeed very convenient in some cases.

It's easy to split this equation into a system

$$\begin{cases} \dot{q} = & -\frac{v_r}{c}q + \frac{p\ddot{\alpha} - gs_\alpha}{cc_\alpha}s \\ \dot{s} = & \frac{p}{c}s_\alpha q, \end{cases}$$

or, neatly,

$$\begin{bmatrix} \dot{q} \\ \dot{s} \end{bmatrix} = \begin{bmatrix} -\frac{v_r}{c} & \frac{p\ddot{\alpha} - gs_\alpha}{cc_\alpha} \\ \frac{p}{c}s_\alpha & 0 \end{bmatrix} \begin{bmatrix} q \\ s \end{bmatrix}. \tag{4.17}$$

Through these passages, we've reached a *two-dimensional first order differential equation* in the unknown q and s, which is way easier to solve than the starting equation (2.15).

Indeed, our equation is in the form

$$\begin{bmatrix} \dot{q} \\ \dot{s} \end{bmatrix}(t) = A(t) \begin{bmatrix} q \\ s \end{bmatrix}(t); \tag{4.18}$$

integrating the time-varying "coefficient",

$$B(t) = \int A(\tau)d\tau, \tag{4.19}$$

the general solution to the system is

$$\begin{bmatrix} q \\ s \end{bmatrix}(t) = e^{B(t)} \begin{bmatrix} q \\ s \end{bmatrix}(t_0), \tag{4.20}$$

where t_0 is the starting point and $\begin{bmatrix} q \\ s \end{bmatrix}(t_0)$ is the *initial condition*, which itself corresponds to some physical conditions on u, or in other words on σ and v_r[14].

We have many ways to computationally work out these integrals, namely through approximation methods, from the *first order Euler Integral* to the *fourth order Runge-Kutta Algorithm*. In the first case the equations look simple and easy, thus it's worthwhile to record them.

Assuming the integration step is T_c and that the starting point is $t_0 = 0$, we recursively have:

$$\begin{aligned} B(kT_c) = & \quad B((k-1)T_c) + \\ + & \quad T_c \begin{bmatrix} -\frac{v_r}{c} & \frac{p\ddot{\alpha}((k-1)T_c) - g\sin(\alpha((k-1)T_c))}{c\cos(\alpha((k-1)T_c))} \\ \frac{p}{c}\cos(\alpha((k-1)T_c)) & 0 \end{bmatrix}; \\ & \quad k \geq 0. \end{aligned} \tag{4.21}$$

The solution looks like

$$\begin{bmatrix} q \\ s \end{bmatrix}(kT_c) = e^{B(kT_c)} \begin{bmatrix} q \\ s \end{bmatrix}(0). \tag{4.22}$$

[14]For instance, setting $s = 1/\sigma$, we can be sure to refer always to nonsingular situations.

Backtracking to the previous notations,

$$u(kT_c) = \frac{q(kT_c)}{s(kT_c)} = \sigma(kT_c)v(kT_c), \tag{4.23}$$

where $v(kT_c) = v_d$ is a constant value.
We obtain

$$\sigma(kT_c) = \frac{u(kT_c)}{v_d} = \frac{q(kT_c)}{v_d s(kT_c)}. \tag{4.24}$$

This is one of our effective *Control Actions*[15].

From the kinetic states of the motorbike, namely $x(0) = x_0$, $y(0) = y_0$ and $\vartheta(0) = \vartheta_0$, we can determine the position at the *preview time*:
for k>0,

$$\begin{aligned} \vartheta(kT_c) &= \vartheta((k-1)T_c) + T_c\sigma((k-1)T_c)v_d \\ x(kT_c) &= x((k-1)T_c) + T_c v_d \cos(\vartheta((k-1)T_c)) \\ y(kT_c) &= y((k-1)T_c) + T_c v_d \sin(\vartheta((k-1)T_c)) \end{aligned} \tag{4.25}$$

Now, as stated before, from these final states we can proceed with the optimization paradigm.

As we'll always have to pass through equation (2.15), this method plays a key role in the whole algorithm: it's understandable that it would be easier to solve algorithmically a first order differential equation than a second order one.

For the sake of honesty, we have to concede that eventually the implementation of the direct method of integration of the original equation (with numeric algorithms) isn't that bad. The results are comparable, at least when the whole dynamics isn't really fast or close to unstable situations. In general this new method is more robust though, thus we'll keep sticking to that from now on.

4.5.1 An apparently smart idea

In the case we want to apply the *Hamiltonian Approach*, we can have the possibility to privilege *speed* rather than *space*. As already explained, even though the Hamiltonian Method shows to be definitely more robust, it suffers from the many computations needed to build up the B matrix. Indeed, we need all the values through $B(0), B(T_c), \ldots, B(kT_c)$ to further obtain $[q(kT_c), s(hT_c)]$.Also, we need to build one dedicated B matrix for every different parameterized α in the ensemble.

One idea to eliminate this bottleneck has been to implement a *lookup table* which might store some values, like the second term in equation (4.21) for all

[15] Remember here that we substitute into the matrix $A(t)$ the expressions for α, where all but some coefficients (we choose the actual number through the polynomial's grade) are determined from the initial conditions, i.e. the states, of the vehicle.

possible values of $\alpha(kT_c)$, varying with the undefined coefficient(s). This method has main drawbacks: first, it requires a booting procedure at the beginning of the algorithm to actually build up this weighty table; second, the massive table, which complexity depends exponentially on the number of parameters, the integration step and the length of the preview time, turns out to fill too much memory resource[16]. Also, memorizing some values of the B matrix means predefining some possible values for the coefficients of the connection trajectory; in other words, we have to approximate the states of the vehicle, sampling a range of possible values for each of them. This means that we use an imprecise feedback action, or that we devise connection trajectories starting from wrong initial conditions which have been sampled. This is in other words a further error we are prone to introduce, along with all those computationally dependent.

4.6 Other Fruits of Brainstorming

In this section we spend some time in describing some other ideas which cropped out during the time spent on the project; some of them haven't actually been included in the final version of the proposed algorithm, but they're part of that "mental wire" which helps understanding the reason of many choices.

Starting from the optimization function, one further issue has been the following: how can we be sure that our Performance Index is actually a convex function? Facing this doubt, we thought about realizing a famous search algorithm: **Simulated Annealing**

4.6.1 Simulated Annealing

This method has the aim of finding a minimum in a very general system and the main advantage that has the ability to avoid becoming trapped at local minima. The algorithm employs a random search which not only accepts changes that decrease objective function , but also some changes that increase it with a probability

$$p = exp\left(-\frac{\delta f}{T}\right),$$

where δf is the increase in f and T is a control parameter.

The idea here has been to apply both the classical search method and the Simulated Annealing to the problem of finding the optimal solution to our problem. The results turning out to match, we understood that our search space was enough "convex" to allow us apply the simplest (and fastest) search method, without the peril to be stuck to incorrect local minima.

[16] The apparent further trouble of having a table with a predefined velocity, namely fixed through the first term of (4.21), can be worked out memorizing only positions $(1, 2)$ and $(2, 3)$ of the array. Apart from this, the whole algorithm would work also for time-varying actual speeds.

4.6.2 Other Search Methods

More methods have been suggested and effectively employed to speed up the optimization procedure; in other words, is that possible to somehow prune the search tree(due to the coefficients in the functions) of some of its branches, being still sure that the disregarded solutions refer to inefficient steering actions?
The subtle point to grasp here is that the first step where we perform a full search through the coefficients' tree submits an optimal solution; then, as we assume that the system runs smoothly and that the integration steps give nonsingular solutions, we're prone to believe that the next optimal solution will be found "next" to the first one. Indeed, if the sampling time isn't that wide, the new initial conditions for the whole system will stay close to the ones considered for the previous step; there'll be also a proximity referring to the desired value on the trajectory to track.

Here comes into use the idea of **Local Search**: the optimization process will run only through a local set around the previous result. It's up to the programmer to decide the width of this set. It's even possible to zoom into it, augmenting the discretization of the same interval.

Another pretty easy, yet amazingly powerful search method is the ***Dichotomic Search***: as studied in Numerical Calculus, this algorithm proceeds splitting the search span apart and taking the "best half". This assumes the convexity of the function over which we perform our search, otherwise we could get stuck at local minima; nevertheless, as seen, it turns out that in our case this search method works regularly. This last algorithm is indeed the most suitable for our purposes and the best up to precision and running time.

4.6.3 A new and more complex Optimization Function

As will be described in a later section, there's been another idea for the Optimization Parameter, namely to exploit more than one reference point on the path to compare with a set of points at the preview distance. Let's take a look at figure (3.3): it's quite clear that the approximation of considering the preview distance linearly proportional to the preview time through a constant velocity (the known current speed examined at time t^*) is not only a possibly wrong approximation[17], but also a misleading one: if the track is bending the vehicle won't ever be able to reach exactly the desired point over the path, which is predetermined through the path definition; what we'll be obtaining is instead a point calculated through the known projection method.

The bootstrapping idea here has been to allow *one more degree of freedom on the preview distance.* If the track were a perfect curvature, in fact, we would aim at a point which is, over the track, distant $\frac{2\pi}{4}pwd = \frac{pi}{2}pwd \simeq 1.5pwd$, instead of an actual point which is pwd ahead on the curvilinear abscissa. In other words,

[17] Indeed the speed could vary due to a steering action, or to a gear change, or to some skid effects, as we'll see in one of the next chapters.

we would overestimate the distance made of more than 50% its value.
Thus, if we plan to build trajectories for alpha with a varying final point (i.e. the *preview time*), we would definitely obtain more precise results. So, the *Optimization Function* stays the same up to the values given by the track's definition, but is somehow less conservative on the values obtained through calculation, which won't lie on a straight line any more.
Another feasible modification could be to compare not only one point at the preview distance (possibly obtained as just described), but, say, three of them, one of whom taken ahead of the first one, the other behind it. This would pose a condition on the derivative of the roll angle at the preview time, which can be easily obtained in a simple loose way. The reader should remember that this would pose one further condition on the states of the connection trajectory, which by itself would have to have a higher degree.

Those two ideas, improving the performance, clearly present the drawback of being more complex and computationally intensive.

4.7 The Algorithm in a glimpse

We summarize and plot a scheme of the whole Algorithm in figure 4.7.

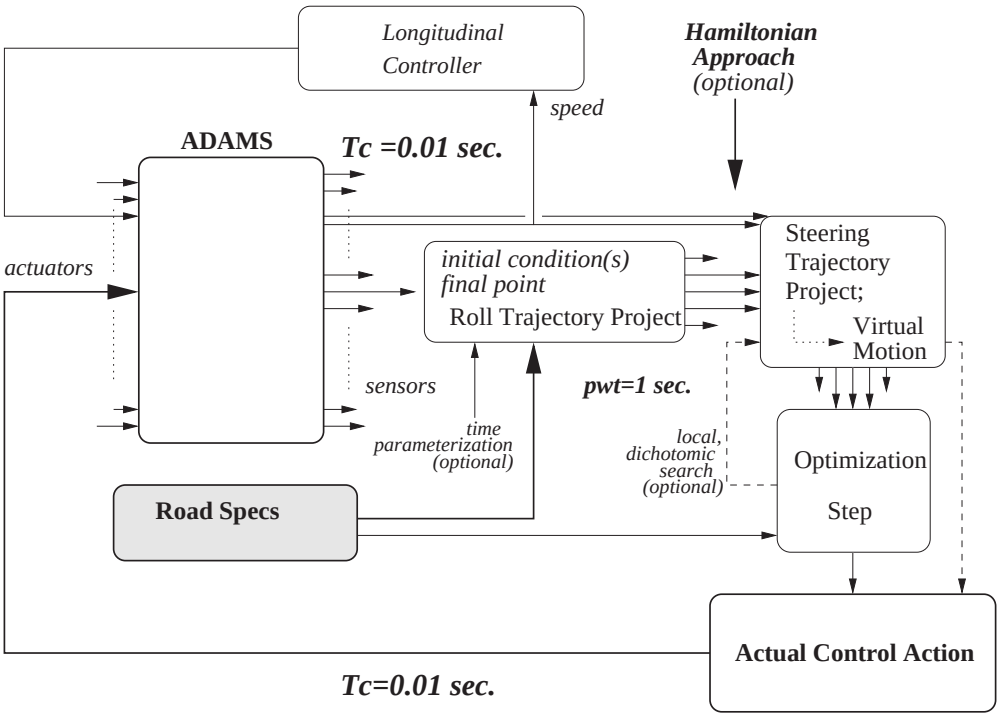


Figure 4.7: A visual representation of the proposed control scheme.

Chapter 5

The Control Algorithm applied to the real world

5.1 Introduction

So far, we have described the structure of our controlling scheme, justified some of its features and unveiled how to demonstrate its stability.

Some of the previous approaches have been booted out mainly for their awkward behavior or clumsy performance. So the question is: *does this new approach work when applied to a real situation?* The answer is positive, and this chapter gives grounds for that.

5.2 The ADAMS Software

The ADAMS Motorcycle Software is a powerful tool for simulating the behavior of a motorcycle. Its main application is for Virtual Prototyping in Motorcycle and Scooter Development. Put more simply, the aim of this product is to virtually test the performance of a vehicle under certain situations, thus avoiding the need to build actual physical prototypes of it. The claimed advantages of this approach are lower test costs, faster engineering decisions and higher performance rate, with less possible structural problems.

The vehicle has to be modelled very precisely from a mechanical point of view, thanks to the finite element approach. Many characteristics, such as the components' materials, some mechanical features or non linear qualities, are included into the system itself, which turns out to be eventually very realistic.

The vehicle is fully parameterized on design variables and geometric locations, and all these parameters can be accessed by the customer. Also, many vehicles models are deployed, thus allowing pretty various analyzes.

Summarizing, ADAMS Motorcycle gives the manufacturers the ability to simulate full vehicles in a virtual test environment.

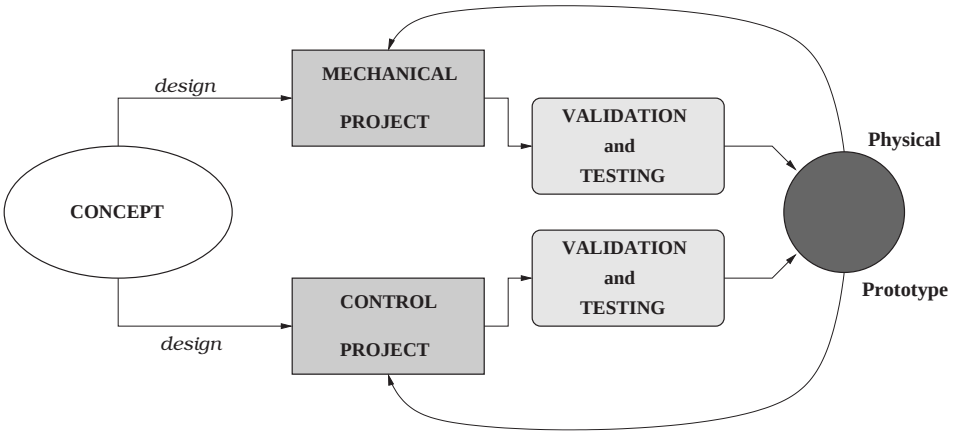


Figure 5.1: Traditional Mechanical and Control Design.

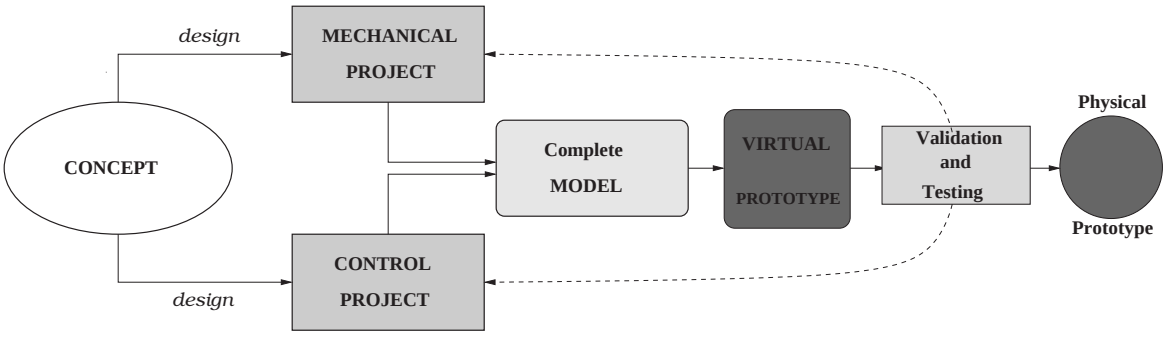


Figure 5.2: Integrated Approach to the Mechanical and Control Design.

At this point it's clear that, having a model and being able to define a road profile[1] and some performance benchmarks, we just need a *pilot*: that's where the controller comes into play.

The model has many *sensors* and *actuators* and conveys informations about its states to the controller part, along with obtaining commands from it to deploy through its actuators. This is the feedback step we always claim the importance of.

The controller works under SIMULINK, possibly through MATLAB (using S-Functions) and also through RTW, the automatic code generator.

Let's take a look from a distance on how the controller used to be conceived, and on how the new one is instead.

5.3 The old Controller

The previous version of the controller was based on a fuzzy approach (see figure 5.3). This would merge a *longitudinal* controller for the velocity with a *lateral* one for the position referred to the track and the balance and finally with a *yaw controller*[2].

The approach accepted is the one of the *Connection Trajectories* in the flat plane, i.e. for the steering action in σ. As described in section 3.5.2, it works rather good in tracking the desired path, needing a balance controller to prevent the system from falling. The difference with Getz's approach is that here the two steering actions are not integrated, but just switched in between in a fuzzy fashion, depending on the magnitude value the second derivative of α assumes.

The longitudinal controller makes use of a perhaps overexploited idea of *preview distance for the speed*. As we've said, the desired velocity is expressed in terms of the curvilinear abscissa s; if we look ahead of a certain *preview time* and hypothesize a constant speed, we can devise a very simple connection trajectory for the speed as well: in our case, that would be a simple straight line (i.e., a first order polynomial). This method is also important for the robustness to speed changes due to the gear shifts.

5.4 The new Controller

Trough this section, we'll show how to change the structure of the software to take into account the effects of the new controller. We'll see how the new arrangement will from one side result way neater, from the other need some modifications .

[1] The 2D-road profile has many similarities to the one used for our theorical studies: the road is either defined by segments as function of "s" coordinates or by points in *(x,y)* coordinates. The road curvature is assumed to be smooth, and this is obtained through the use of *Clotoids* as connections between straight lines and constant-radius curves.

[2] We'll disregard this last one for now, albeit it plays a key role on the substantial performance of the old system.

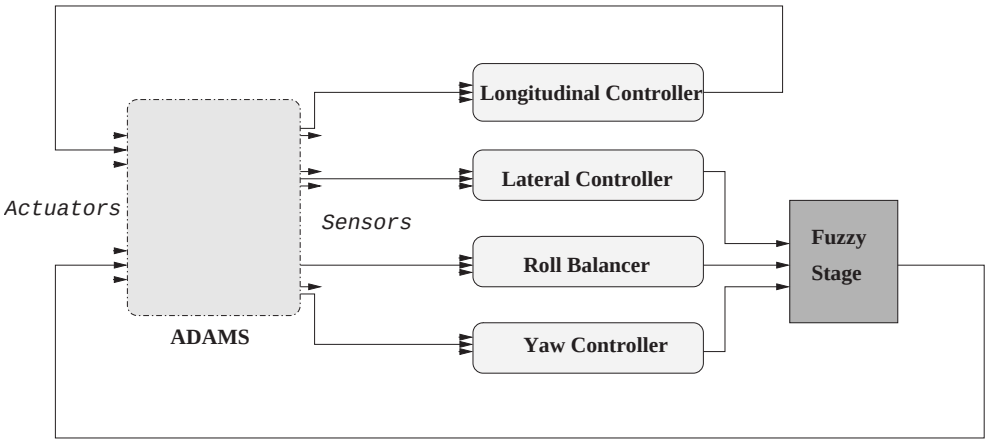


Figure 5.3: Block Structure for the old Controller.

First, as already stated, the new approach will prevent from using the roll balancer, as the connection trajectories for the bank angle eliminate away all at once all the problems which come along with the minimum phase feature of the system. Also, this new method appears to be pretty robust toward those behaviors once solved through the yaw controller, like brusque braking or abrupt lateral noise provoking an undesired steering action.

See figure 5.4 for a glimpse to the new method's structure.

The new approach solves then many of the issues referred to the method once known as *lateral controller*.

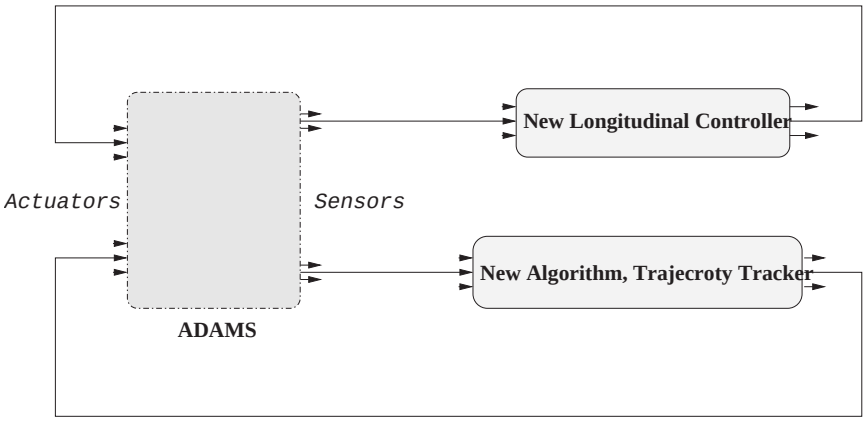


Figure 5.4: Block Structure for the New Controller.

5.4.1 Description

While the old controller was wholly implemented in SIMULINK, the new algorithm needs to be run under the MATLAB environment. This poses the question of how to switch between the twos: we need indeed some values provided by the

sensors, such as for instance the vehicle's states, as well as some forces or mechanical parameters, like the tires' slip factors. Besides, our control action has to be exerted through some actuators, thus still present in the SIMULINK setting.

The algorithm by itself, on the contrary, needs the speed and the precision of MATLAB. Consequently, the choice has been to exploit the usefulness of the *S-Function* structure[3].

This structure provides the programmer with the automatic possibility of defining some states, as well as receiving some inputs and defining outputs, along with the feature of refreshing them through any calculations at each time step.

The timing issue has been assessed choosing the same clock for the SIMULINK program and the underlying MATLAB algorithm.

At each time step (actually $T_c = 0.01[s]$) the sensors provide information about the motorcycle, like the necessary states. From many of these values, the actual optimization algorithm is applied on a *preview time* which can actually be set or predefined (it's customary to leave it at $1secs$). Many variables are refreshed, and other values are given as outputs in the form of a control action, which will be used only in its first term (MPC approach). Also all the information about the kinetic states of the bike are provided by the vehicle's sensors (included all those referring to the connection trajectories); here the path is predefined and possibly modifiable by the end user.

The control values, passed to the actual "mechanical" simulator, modify the position of the motorbike and so pass through the *real system*, which we assume to be modelled by our theorical kinetics and dynamics equations.

All this procedure is repeated each time step, as in the fashion of the MPC approach.

Here one can understand how important role the modelization procedure plays: too a precise model would be way much complex and not suitable for real-time use, as well as rigid to unexpected behaviors. Conversely, a loose model could be inadequate to take into account some features. We claim that our model attains good performances, with some notable exceptions we had to figure out how to solve.

5.4.2 Tires Modelization

In our theorical analysis, we've always assumed that the tires had no lateral slip, indeed we made the hypotheses that the whole system underwent the so called *nonholonomic constraints*. Here this assumption fails to ba valid; it can be seen that skidding effects are always present while the motorcycle is bending or turning, and they're indeed the very cause of the movements inside a curve.

The reader might remember that, ideally,

$$\sigma = \frac{\tan(\beta)}{b}, \tag{5.1}$$

[3]Please note that future releases will include the possibility of the automatic language generator(RTW) for optimization purposes.

where σ is the motorbike's curvature (i.e. the inverse of the radius defined through the normal to the front and rear wheel) and β is the effective steering angle obtained moving the steering shaft [4]. In other words,

$$\beta = \arctan(\sigma b). \tag{5.2}$$

Now, we can have some skidding effect both on the front and the rear wheel; dubbing λ_f and λ_r respectively the front and rear slip coefficients, there's a relation that generalizes the previous one:

$$\beta = \arctan(\frac{b\sigma - \sin(\lambda r)}{\cos(\lambda_r)}) + \lambda_f. \tag{5.3}$$

Clearly, equation (5.2) is a particular case that can be obtained from (5.3).

The problem was to correct the imperfect values obtained and due to the skidding effects; equation (5.3) was used in the following fashion: we already know that the actual control is expressed in form of σ and then fed as an actuator signal; from the previous relation, let's obtain β with those values for λ_f and λ_r (obtained from two dedicated sensors) and, from relation (5.2) the corresponding σ again; now, let's refer all the calculations during the optimization procedure to this value.

That's a classical example where the model was unable to justify some new behaviors and needed some further modifications and refinements.

5.4.3 The new Longitudinal Controller

A very large quantity of test performed during the time passed over the project, as well as some experience and sensibility earned throughout the whole activity made clear that the idea to employ a preview time also for the speed, while matching the winning idea used for other parts, offered low quality achievements. This both for matching the desired velocity and during the frequent gear changes.

Here the solution was nice and easy: *PID control*. Having set up the necessary blocks to attain it, we just had to figure out which were the time constants for the system and calibrate the gains and the pole positions in our transfer functions.

Given the necessity to place a *saturator* right at the output of the whole system, we provided an *anti-saturation* block [5] to improve the system's performance and avoid the problems referred to the saturation itself.

From figure (5.5), one can understand that when the actuator saturates the feedback loop opens, due to the non-zero error $u - v$, and decreases v with a velocity which is inversely proportional to Tt. In other words, the integrator arrives discharged at the possible saturation condition and consequently doesn't produce any overshoot.

[4] The reader should remember also another important relation, the one relating the steering angle with the roll angle: $\tan\beta\cos\alpha = \tan\psi$, where ψ is the rotation angle for the steering shaft.

[5] This scheme is widely known as *Anti-reset wind-up*.

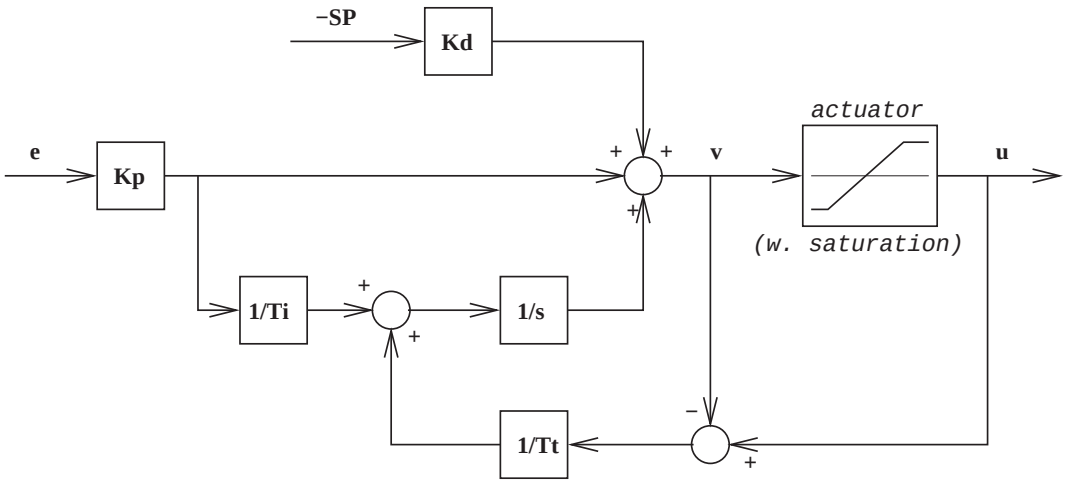


Figure 5.5: Scheme for the *anti reset wind up*.

Chapter 6

System's Modelling: an Application

6.1 Introduction

As we mentioned in the introductory section, as well as throughout the theorical chapter, one pretty interesting engineering problem is *System's Modelling*. The issue here is always to choose between accuracy and robustness: a very precise model could take into account many behaviors of the system but at the same time fail under the case of unexpected happenings, i.e. unforeseen events which could affect and perturb the same system. Conversely, a somehow rough modelization, being by itself pretty robust, could be inadequate to describe some occurrences that we expect from the real world.
In our case, we decided to make use of a model for the motorcycle in some way simple, but definitely really robust and enough precise to mirror all the events we were interested in.
Throughout this chapter, we're introducing a new model, which could be suitable to highlight some new features related to the *driver*. We'll then emphasize some possible applications and a control synopsis for this new model.

6.2 The new Model

The new task here is to model the driver, along with the vehicle; having used a very simple *Inverted Pendulum* for the motorcycle, the most direct idea here is to exploit an *Inverted Double Pendulum*.

We'll dub the vehicle's mass M, while the driver's one m. The angle that m describes referred to the straight line passing through the mass M and the contact point with the ground will be γ, and the roll angle for the vehicle will be α from now on. The distance of the mass M to the contact point with the ground will be p, while the space between M and m is going to be q.

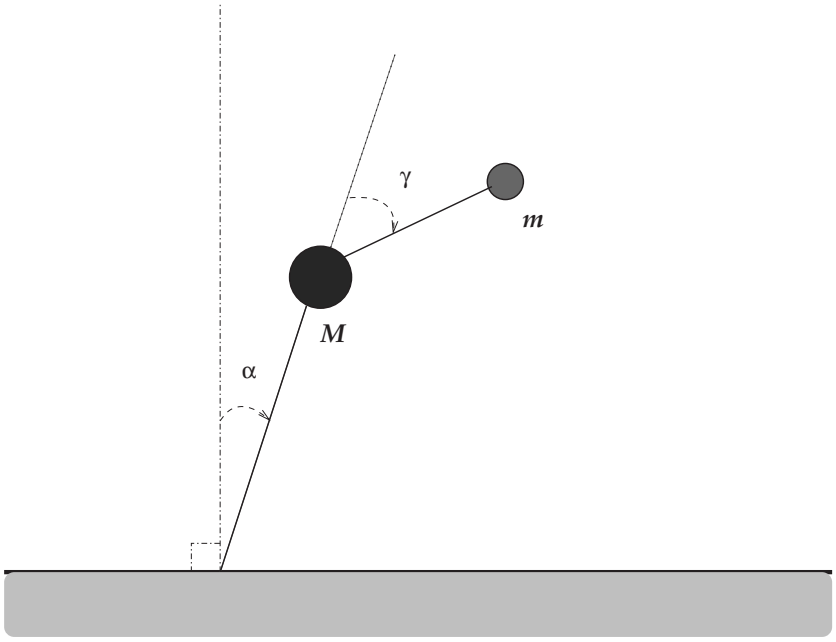


Figure 6.1: The new Model.

6.3 Mathematical derivation of the Dynamics Equations

In the theory review, we have gone through the *Lagrange's Equations* and highlighted how they can be used to describe any system in any coordinates we wanted. these new coordinates are named *Generalized Coordinates*: here we want to write new *Dynamics Equations* referred to these coordinates.
We'll tackle the problem from the most convenient side: expressing the system's states, i.e. position, velocity and acceleration, in function of these new coordinates, as well as define some *Generalized Forces* acting along them.
Let's start with the *Lagrange's Equations* for the two masses; being $L = T - V$,

$$L_M = \tfrac{1}{2}Mp^2\dot{\alpha}^2 - Mgpc_\alpha; \tag{6.1}$$

$$L_m = \tfrac{1}{2}m\left(p^2\dot{\alpha}^2 + q^2(\dot{\alpha}+\dot{\alpha})^2 + 2pq\dot{\alpha}(\dot{\alpha}+\dot{\gamma})c_\gamma\right) - \\ -mg(pc_\alpha + qc_{\alpha+\gamma}); \tag{6.2}$$

Unfolding all the necessary calculations, we first consider the coordinate α:

$$\begin{aligned}
\frac{\partial L_M}{\partial \alpha} &= Mgps_\alpha; \\
\frac{\partial L_M}{\partial \dot{\alpha}} &= Mp^2\dot{\alpha}; \\
\frac{\partial L_m}{\partial \alpha} &= mg(ps_\alpha + qs_{\alpha+\gamma}); \\
\frac{\partial L_m}{\partial \dot{\alpha}} &= m\left(p^2\dot{\alpha} + q^2(\alpha \dot{+} \gamma) + pq(2\dot{\alpha}+\dot{\gamma})c_\gamma\right);
\end{aligned}$$

The same computations with respect to γ:

$$\begin{aligned}
\frac{\partial L_M}{\partial \gamma} &= 0; \\
\frac{\partial L_M}{\partial \dot{\gamma}} &= 0; \\
\frac{\partial L_m}{\partial \gamma} &= -mpq\dot{\alpha}(\dot{\alpha}+\dot{\gamma})s_\gamma; \\
\frac{\partial L_m}{\partial \dot{\gamma}} &= mq(q(\dot{\alpha}+\dot{\gamma})+pc_\gamma\dot{\alpha});
\end{aligned}$$

Now, working out the terms in the definition of the Lagrange's Equations:

$$\begin{aligned}
\frac{d}{dt}\frac{\partial L_M}{\partial \dot{\alpha}} &= Mp^2\ddot{\alpha}; \\
\frac{d}{dt}\frac{\partial L_m}{\partial \dot{\alpha}} &= m\left(p^2\ddot{\alpha}+q^2(\ddot{\alpha}+\ddot{\gamma})+pq\left((2\ddot{\alpha}+\ddot{\gamma})c_\gamma-(2\dot{\alpha}+\dot{\gamma})s_\gamma\dot{\gamma}\right)\right); \\
\frac{d}{dt}\frac{\partial L_M}{\partial \dot{\gamma}} &= 0; \\
\frac{d}{dt}\frac{\partial L_m}{\partial \dot{\gamma}} &= mq\left(q(\ddot{\alpha}+\ddot{\gamma})+p(-s_\gamma\dot{\gamma}\dot{\alpha}+c_\gamma\ddot{\alpha})\right);
\end{aligned}$$

As described in the theory section, we now have to go look for an expression of the *Generalized Forces* acting along the coordinates α and γ.

It turns out to be quite tough to single out each force acting along these directions, especially because of the reaction forces between the two masses, as well as due to the fact that the system is not *Inertial*; this means we'd have to consider also the dragging forces and the Coriolis forces also.

It's conceptually way easier and straightforward to use Newton's law for the whole force acting on each point, and then project it along the generalized coordinates. Considering figure 6.2, we have:

$$\vec{r}_m = x_0 + jy_0 + ce^{j\vartheta} - j(ps_\alpha + qs_{\alpha+\gamma})e^{j\vartheta}; \tag{6.3}$$

$$\begin{aligned}
\vec{v}_m = v_r e^{j\vartheta} + jce^{j\vartheta}\dot{\vartheta} - jpc_\alpha\dot{\alpha}e^{j\vartheta} + ps_\alpha\dot{\vartheta}e^{j\vartheta} - \\
-jqc_{\alpha+\gamma}\dot{\alpha}e^{j\vartheta} - jqc_{\alpha+\gamma}\dot{\gamma}e^{j\vartheta} + qs_{\alpha+\gamma}\dot{\theta}e^{j\vartheta};
\end{aligned} \tag{6.4}$$

$$\begin{aligned}
\vec{a}_m = \dot{v_r}e^{j\vartheta} + jv_r\dot{\vartheta}e^{j\vartheta} - c\dot{\vartheta}^2e^{j\vartheta} + jce^{j\vartheta}\ddot{\vartheta} + \\
+jps_\alpha\dot{\alpha}^2e^{j\vartheta} - jpc_\alpha\ddot{\alpha}e^{j\vartheta} + pc_\alpha\dot{\alpha}\dot{\vartheta}e^{j\vartheta} + pc_\alpha\dot{\alpha}\dot{\vartheta}e^{j\vartheta} + ps_\alpha\ddot{\vartheta}e^{j\vartheta} + \\
+jps_\alpha\dot{\vartheta}^2e^{j\vartheta} + jqs_{\alpha+\gamma}\dot{\alpha}^2e^{j\vartheta} + jqs_{\alpha+\gamma}\dot{\alpha}\dot{\gamma}e^{j\vartheta} - jqc_{\alpha+\gamma}\ddot{\alpha}e^{j\vartheta} + \\
+qc_{\alpha+\gamma}\dot{\alpha}\dot{\vartheta}e^{j\vartheta} + jqs_{\alpha+\gamma}\dot{\alpha}\dot{\gamma}e^{j\vartheta} + jqs_{\alpha+\gamma}\dot{\gamma}^2e^{j\vartheta} - jqc_{\alpha+\gamma}\ddot{\gamma}e^{j\vartheta} + \\
+qc_{\alpha+\gamma}\dot{\gamma}\dot{\vartheta}e^{j\vartheta} + qc_{\alpha+\gamma}\dot{\alpha}\dot{\vartheta}e^{j\vartheta} + qc_{\alpha+\gamma}\dot{\gamma}\dot{\vartheta}e^{j\vartheta} + \\
+jqs_{\alpha+\gamma}\dot{\vartheta}^2e^{j\vartheta} + qs_{\alpha+\gamma}\ddot{\vartheta}e^{j\vartheta}.
\end{aligned} \tag{6.5}$$

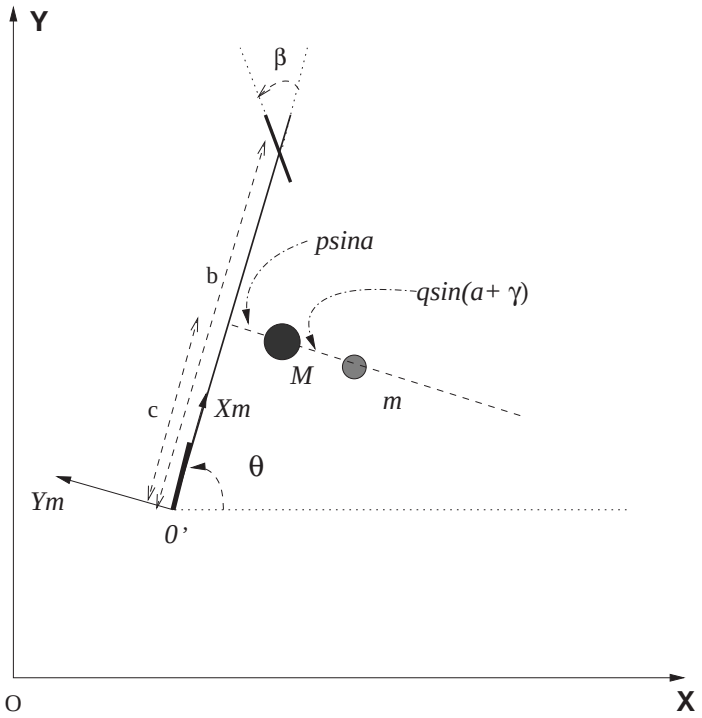


Figure 6.2: Vehicle's top view.

Note that the first term in the equation for the velocity is one of the *NonHolonomic Constraints*[1].
As already mentioned, the second constraint is $\dot{\vartheta} = \sigma v_r$.
Here we'll make some more conjectures, namely that $\ddot{\alpha} = \dot{\alpha} = \ddot{\gamma} = \dot{\gamma} = 0$; these hypotheses are actually those implied for the derivation of the preceding model, and correspond to assuming *null Coriolis Forces*. Moreover we'll consider *the steering shaft with no momentum associated with*, as well as *no gyroscopic effects onto the wheels*.
We obtain for the last term,

$$\begin{aligned} \vec{a_m}_{|_{hp}} = & \left(\dot{v_r} - c\sigma^2 v_r^2 + ps_\alpha(\sigma\dot{v_r}) + qs_{\alpha+\gamma}(\sigma\dot{v_r})\right) e^{j\vartheta} + \\ + & \; j\left(\left(v_r^2\sigma + c(\sigma\dot{v_r}) + ps_\alpha\sigma^2 v_r^2 + qs_{\alpha+\gamma}\sigma^2 v_r^2\right) e^{j\vartheta}\right); \end{aligned} \tag{6.6}$$

We have just obtained a simplified expression for the acceleration of the driver's mass; applying Newton's law to obtain the *Generalized Forces* requires to attain an expression of,say, the *Generalized Acceleration*. This is obtained by projecting the previous expression along the generalized coordinates, i.e. in out case along the plane described by the movement of α or γ. Also, it should be clear that, at his point, the term dependent on the yaw ϑ can be disregarded; it affects indeed

[1]More precisely, dubbing $z_0 = x_0 + jy_0$, we have that, being $v_\perp = 0$, $\dot{z_0} = v_r e^{j\vartheta}$; in other words the vehicle, as assumed, doesn't skid apart.

only the phase of the vector $\vec{a_{m|_{hp}}}$. Hence, we obtain,

$$\vec{a_{m|_{hp}|_{j,\vartheta=0}}} = (1 + (ps_\alpha + qs_{\alpha+\gamma})\sigma)\,\sigma v_r^2 + c(\dot{\sigma}v_r + \sigma\dot{v_r}). \tag{6.7}$$

Proceeding in the same way, we obtain for the mass M,

$$\vec{a_{M|_{hp}|_{j,\vartheta=0}}} = (1 + p\sigma s_\alpha)\sigma v_r^2 + c(\dot{\sigma}v_r + \sigma\dot{v_r}). \tag{6.8}$$

Now, the *Lagrange's Equations* appear to have the form:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\alpha}} - \frac{\partial L}{\partial \alpha} = \Phi_\alpha; \tag{6.9}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\gamma}} - \frac{\partial L}{\partial \gamma} = \Phi_\gamma. \tag{6.10}$$

As referred in the Theory Section, Φ_i is the *Generalized Force* related to the generalized coordinate i.
As of now, we just have to express the *Generalized Forces* in a suitable and correct way, having already derived a close form for the respective accelerations. As suggested in [15] and [2], it's necessary to take into consideration a virtual displacement referred to the new coordinates and multiply it by the force obtained through Newton's Equation. More precisely, in the case of the driver's mass, we have (see figure 6.1):

$$\begin{aligned} x_m = &\quad ps_\alpha + qs_{\alpha+\gamma}; \\ x_m^{'} = &\quad ps_\alpha + qs_{(\alpha+\gamma)+\delta\gamma} = \\ = &\quad ps_\alpha + q\left(s_{\alpha+\gamma}c_{\delta\gamma} + s_{\delta\gamma}c_{\alpha+\gamma}\right) \approx \\ \approx &\quad ps_\alpha + qs_{\alpha+\gamma} + qc_{\alpha+\gamma}\delta\gamma, \end{aligned}$$

as we're considering a differentials and, being $\delta\gamma \approx 0$, $c_{\delta\gamma} \approx 1$ and $s_{\delta\gamma} \approx 0$. We obtain

$$x_m^{'} - x_m = qc_{\alpha+\gamma}\delta\gamma.$$

We define the *Virtual Displacement*[2] as:

$$\delta x\,|_\gamma = \frac{x_m^{'} - x_m}{\delta\gamma}, \tag{6.11}$$

and the *Generalized Force* referred to coordinate γ is:

$$\Phi_\gamma = m\|\vec{a_{m|_{hp}|_{j,\vartheta=0}}}\|qc_{\alpha+\gamma}. \tag{6.12}$$

Working out all the math for α, we similarly get:

$$\Phi_\alpha = \left(M\|\vec{a_{M|_{hp}|_{j,\vartheta=0}}}\| + m\|\vec{a_{m|_{hp}|_{j,\vartheta=0}}}\|\right)pc_\alpha. \tag{6.13}$$

[2]Here we exploited the concept of *Virtual Displacement* only in function of our current purposes; for a deeper insight of this important topic refer to [15] and [2].

reordering all the previous equations and plugging some results into them, the *New Dynamics Equations for the Model* turn out to be:

$$\begin{gathered}(Mp^2 + mp^2 + mq^2 + 2mpqc_\vartheta)\,\ddot{\alpha} + (mq^2 + mpqc_\gamma)\,\ddot{\gamma} - \\ -mpq\,(2\dot{\alpha} + \dot{\gamma})\,s_\gamma\dot{\gamma} - Mgps_\alpha - mg\,(ps_\alpha + qs_{\alpha+\gamma}) = \\ M\,((1 + \sigma ps_\alpha)\,\sigma v_r^2 + c\,(\dot{\sigma}v_r + \sigma v_r)) + \\ +m\,((1 + (ps_\alpha + qs_{\alpha+\gamma})\,\sigma)\,\sigma v_r^2 + c\,(\dot{\sigma}v_r + \sigma\dot{v_r}))\,pc_\alpha;\end{gathered} \tag{6.14}$$

$$\begin{gathered}(mq^2 + mpqc_\gamma)\,\ddot{\alpha} + mq^2\ddot{\gamma} - mpqs_\gamma\dot{\gamma}\dot{\alpha} + \\ +mpq\dot{\alpha}\,(\dot{\alpha} + \dot{\gamma})\,s_\gamma - mgqs_{\alpha+\gamma} = \\ m\,(1 + (ps_\alpha + qs_{\alpha+\gamma})\,\sigma)\,\sigma v_r^2 + \\ +c\,(\dot{\sigma}v_r + \sigma\dot{v_r})\,qc_{\alpha+\gamma}.\end{gathered} \tag{6.15}$$

Note that these equations have been derived under all the assumptions holding also for the calculation of the old model; as a matter of fact, restricting those relations through hypotheses like $m = 0$ or $q = 0$, we attain the equation (referred to the angle α) (2.15), of page 21. As a counterproof to all this math, we could try to derive these formulas through the approach in [12] and [13]: writing the Lagrangians as here, we have to constraint them, thus obtaining a formula which could not be used directly to derive the dynamics equations[3]. We must therefore pass through a set of transformations, named *d'Alembert* maps, to finally get the equations referred to our desired coordinates.
Using some software programs like $MAPLE^{®}$, one can check the validity of our results.

6.4 New Control Heuristics

After having defined a consistent and sound model, we have to suggest a control methodology which might encompass new characteristics of the whole system and possibly improve the performances.

A couple of approaches have been thought thereabout, the more brilliant of which is the following: starting from our new model, we can try to think it as a new big unique mass (the sum of the twos), plus a structure of the simple inverted pendulum, with a certain height calculated through the center of mass of the double system. Writing down the equations and simplifying them, we obtain a relation similar to (2.15), only depending on new parameters. We can then operate in parallel between this new relation and the *new dynamics equations*: we can obtain a certain behavior for the whole system operating either on the velocity, or the steering shaft, or even on the new mass position (movement of the pilot). We get in other words a new degree of freedom, which we can fully

[3] See the Theory section.

exploit.
This can be just an example of heuristics for the control action.

We leave this new goal to future works.

Chapter 7

Results, Conclusions and Future Work

7.1 Outcomes

In this chapter we'll focus our attention to the performance analysis of the whole algorithm as evident from simulations and real-world results.

7.1.1 Simulations

Having implemented the core of the algorithm in a MATLAB file, we had somehow to test it. Thanks to the theorical meditations lying beneath the heuristic and having somehow proven the stability of the method, we were quite confident about the simulations outcomes.

The idea was to implement many version of a motorcycle simulator through the SIMULINK interface, which could mimic the real ADAMS software. Obviously, our simulator had no mechanical features, but could only cover the dynamic behavior of the system. In other words, through simulations we were still controlling a *point-mass vehicle*, subject to some instability constraints on its *roll*, but free from all its burdens regarding mechanical nonlinearities.

Seen from another side, we just verified that we were able to validate the roll dynamics equations, i.e. maintain the vehicle's stability, along with tracking any feasible and considerate trajectory on the plane.

As stated in section 4, we had to build up a reference path, which we did through first defining a *velocity profile* and a *curvature profile*, all referred to the *curvilinear abscissa*. Then, we implemented a *road builder*, which created the path in the x,y reference frame (even if always referred to the abscissa), as well as calculating the *desired roll* in terms of the same abscissa.

As already said, the algorithm produces a control action in terms of the *steering angle* and feeds the simulator with it; conversely, the very simulator processes this data and outputs the new dynamic states for the vehicles, as well as the ki-

netic ones (position). We used a *sampling time step* of $T_c = 0.01[sec]$[1], and a *preview time* of $T = 1[sec]$, albeit this last term was prone to be modified as desired[2].

Next, some plots referring to simulations will be displayed. Let's define some elements: usually the desired trajectory appears thicker than the other items; the position of the rear wheel contact point with the ground is a star, while the projection of the vehicle's mass on the flat plane is a little circle; detaching from the vehicle's state point are either some connection trajectories or just the optimal one chosen, as well as a straight dashed line slanted accordingly to the yaw ϑ (from which the reference point(s) on the desired trajectory are calculated).

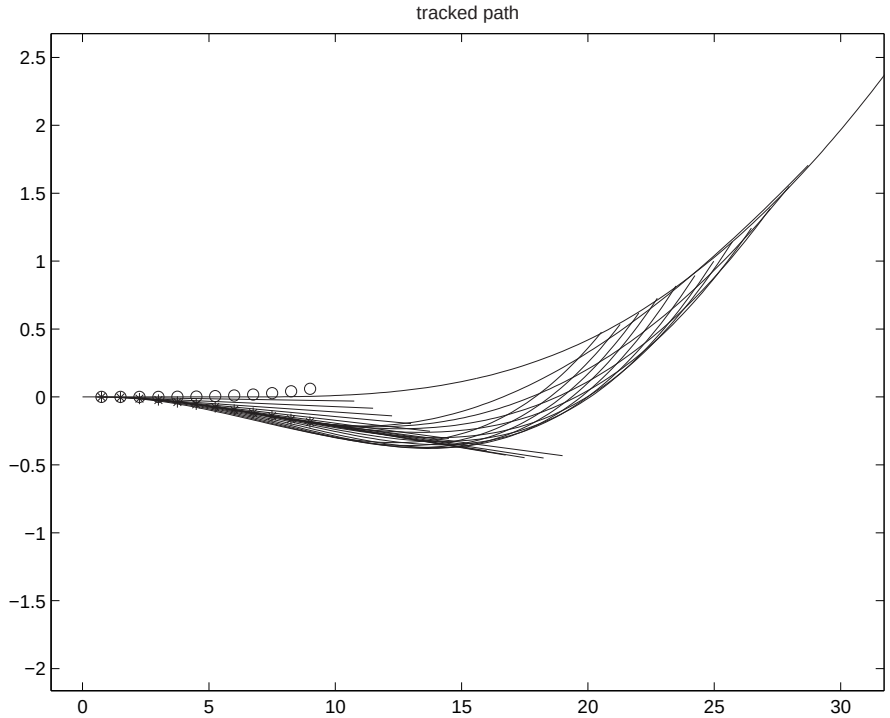


Figure 7.1: The Countersteering Phenomenon.

From figure 7.1, we have that the motorcycle is required to make a turn left, starting right over the desired path; the *Countersteering Phenomenon* is here evident: to reach a negative required roll, the driver first has to steer right, i.e. negative, and then positive after some fractions of second. It's kind of interesting to notice that this behavior blossoms out just from the dynamics equations, nobody imposed that, it's just a natural manoeuver which the driver is required to perform.

In figure 7.2 again, we make use of a more sophisticated parametrization to

[1] Remember the paradigm of *Model Based Predictive Control*.

[2] Actually the real *preview distance* is dynamically modified linearly accordingly to the velocity, or even possibly quadratically depending on the instant acceleration.

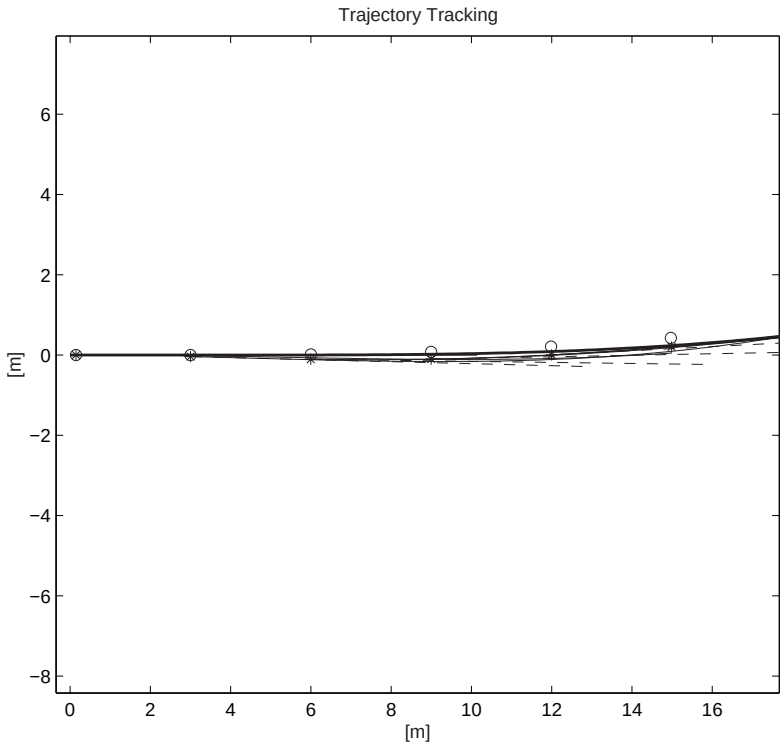


Figure 7.2: The Countersteering Phenomenon.

obtain the desired tracking, and again it's noticeable the countersteering effect, as well as a very precise performance for the tracking itself.

In figure 7.3 the vehicle is in a state not over the path and is "looking ahead" to devise some connection trajectories. Here we're using simple cubic polynomials for the roll, thus depending only on one parameter (more precisely, the derivative at the arrival point; no time parametrization is here exploited). Applying the optimization procedure, the outcome is simply the best trajectory on the plane based on the end point distance, which in turns corresponds to the optimal control in σ.

In figures 7.4 and 7.5 the reader can assess the tracking performance, which we judged to be pretty accurate. Here the motorbike is speeding at $20\frac{m}{s}$, that is roughly around $70\frac{km}{h}$, and entering in a curve of radius 80 meters. Please notice that, for the sake of neatness and clarity, we limited the plot to once every 10 steps, i.e. the state of the vehicle and the connection trajectories are drawn every 0.1 seconds.

In figure 7.6 the reader can take a look at the shift in roll happening when the curvature of the road changes sign. The manoeuver is rather steady and definitely robust.

Plot 7.7 refers to the evolution of the roll angle and the steering action through time.

The previous three figures describe some robustness tests performed, namely

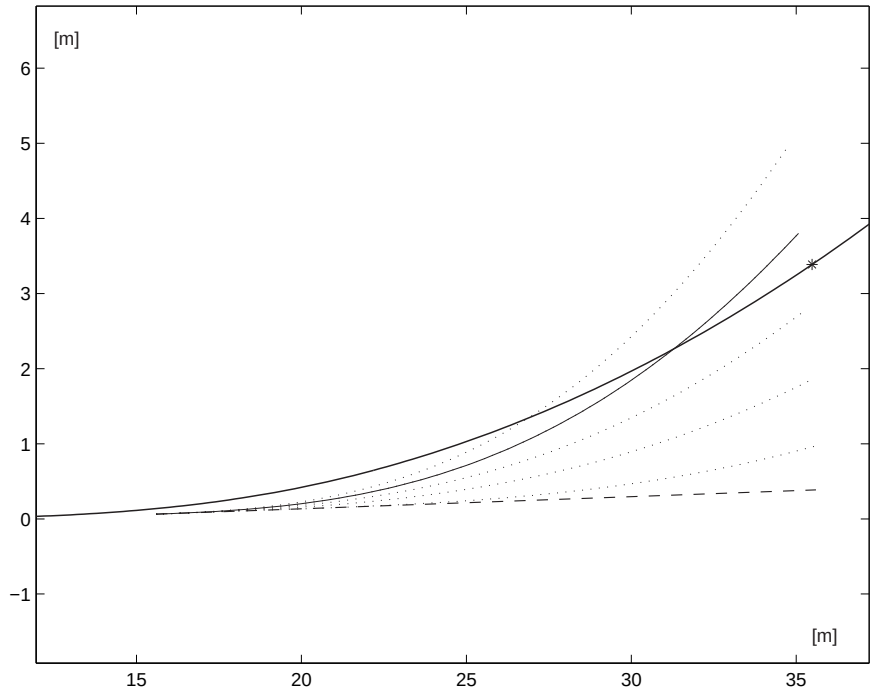


Figure 7.3: A bundle of trajectories, with the optimal one highlighted.

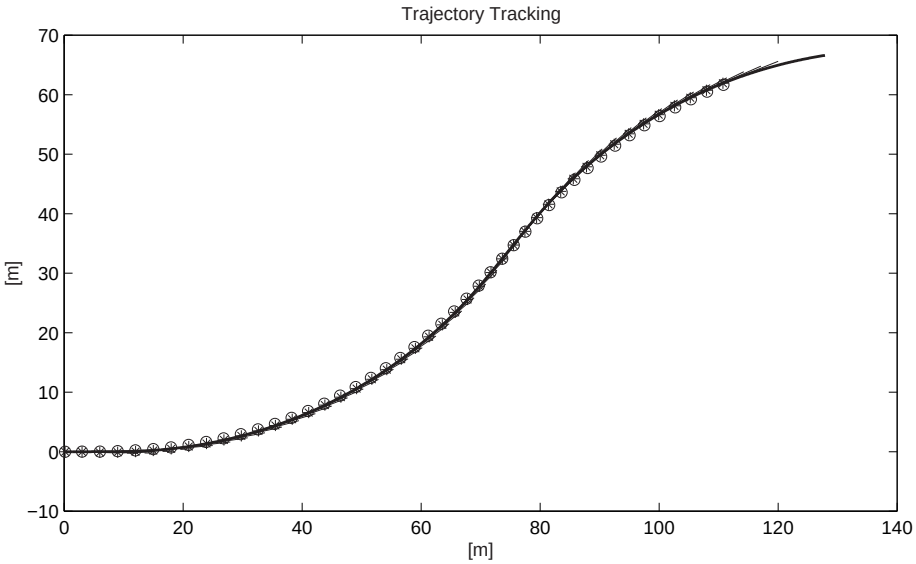


Figure 7.4: Trajectory tracking: an example.

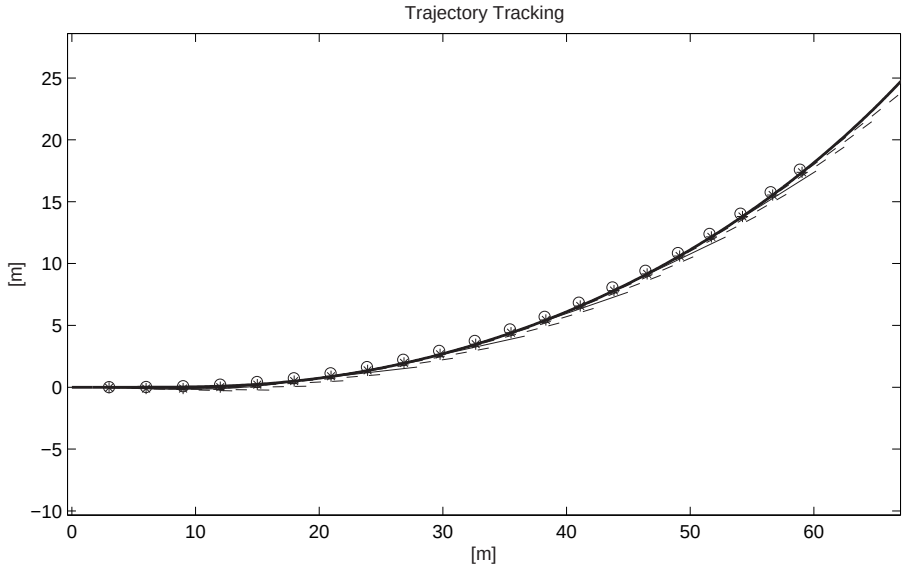


Figure 7.5: Trajectory tracking: another example.

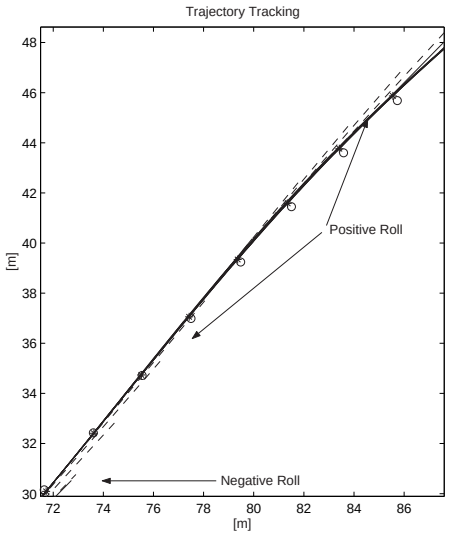


Figure 7.6: Trajectory tracking: shift on the roll sign.

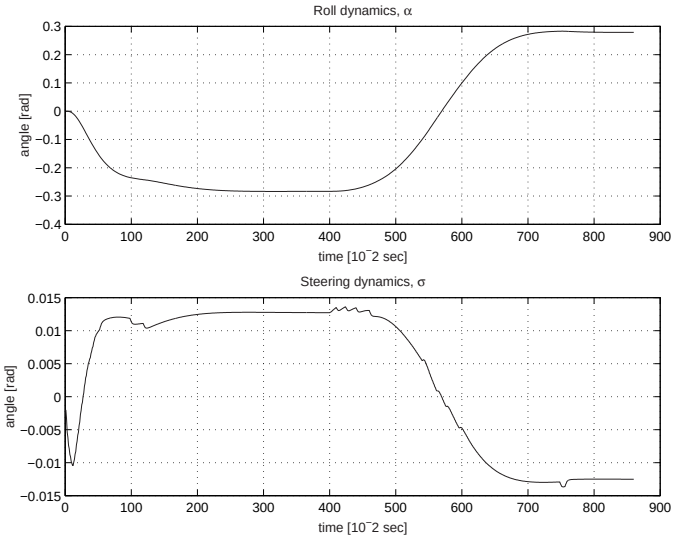


Figure 7.7: Trajectory tracking: Progress of the roll α and the steering angle σ.

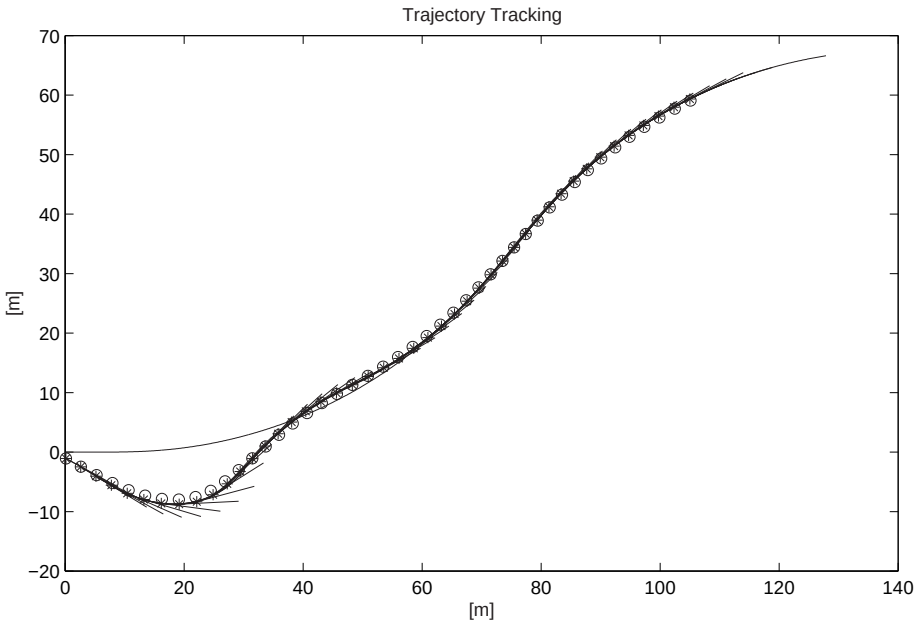


Figure 7.8: Trajectory tracking: Robustness Test.

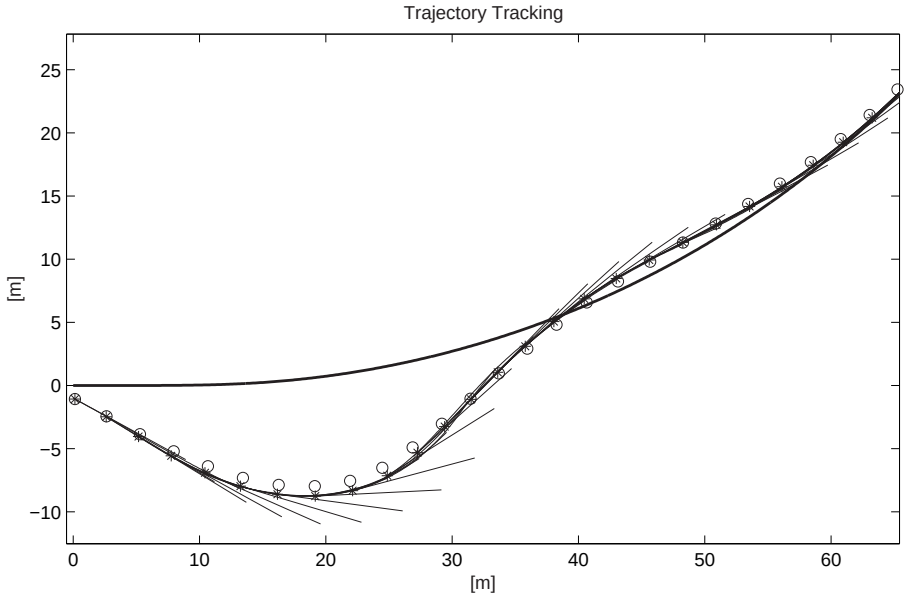


Figure 7.9: Trajectory tracking: Robustness Test (zoom in).

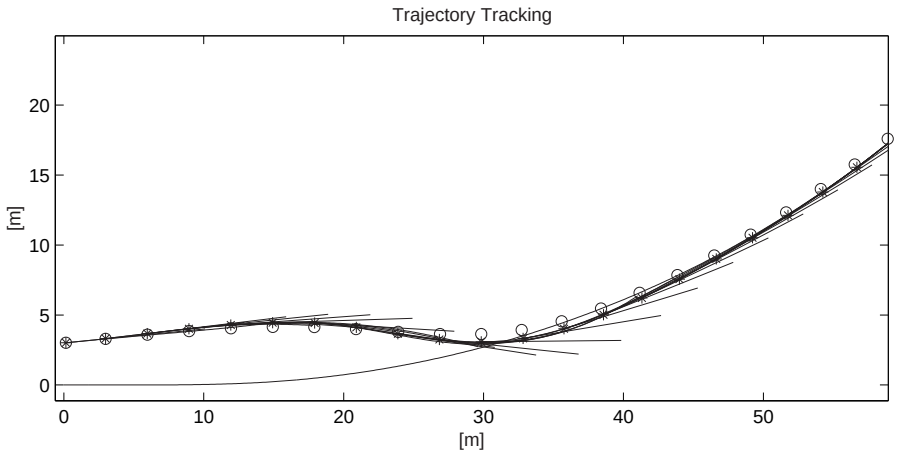


Figure 7.10: Trajectory tracking: Robustness Test, with different initial conditions.

here the ability of the controller to recover from anomalous situations can be assessed.

Figures 7.8 and 7.9 refer to a vehicle starting not over the path and slightly out of the path direction, while plot 7.10 described an opposite start up. In both cases, but especially in the first one, it can be noted how the motorbike recovers and converges smoothly to the path.

The reader might be aware that this behavior has been obtained through a cubic polynomial as connection trajectory for the roll, and with a smooth time parametrization. Search for the optimal has been kept local, although one might first think that a global one could definitely help the swing in the control action.

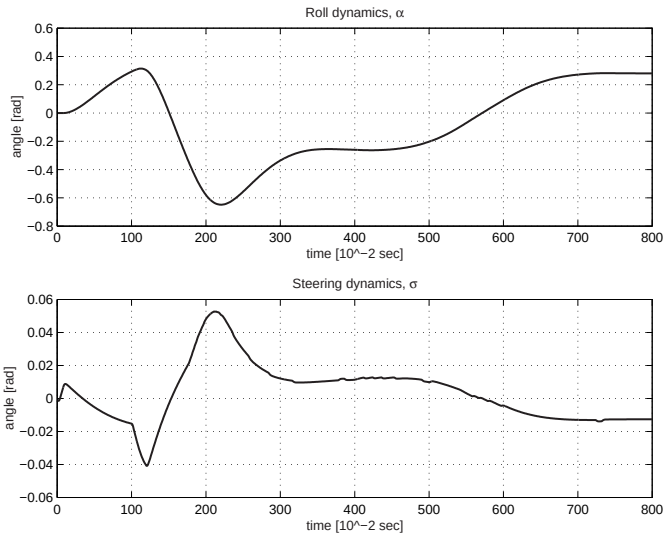


Figure 7.11: Trajectory tracking: Signals evolution during the robustness test.

Finally, in figure 7.11 the reader can check out the time evolution of the two signals α and σ.

7.1.2 ADAMS Performance

It should be quite indisputable that the simulations done in MATLAB are from one side pretty promising if referred to the algorithm's heuristic, but from the other side incapable of testing the method applied to a real-world problem; our *simulator*, indeed, is based on the very same equations we used to build up the algorithm, thus we're unable to test its robustness on a real system which undergoes equations only similar to ours.

Integrating our new method into the ADAMS software [3] gave us the possibility to check its consistency when coping with reality. Here it comes to mind the discussion on the importance of the *modelization* procedure: will our model be enough accurate to power up a real motorbike, being our method stable enough

[3] For more details on this software, please refer to section 5, page 61.

when many of the assumed premises won't hold any more? Again, will our model be enough robust when coping with unexpected nonlinearities of the whole system?

As described in section 5, we actually had to have our system refined in some of its characteristics, for instance when facing the problem of tires'slip, over- and under-steering.

Nevertheless, in general, we didn't have to make such drastic variations on the very same algorithm, thus its application turned out to be rather straightforward.

The outcomes appeared to be fulfilling our expectations, and our main achievements can be summarized as follows:

- Better performance at low speed, from $3\frac{m}{s}$ on;
- Improved performance at small curvature radius' roads;
- Neater control signals, especially the steering action;
- More compact and clean setting for the algorithm (take a look at figures (5.3) and (5.4)); most important, no more use of the *fuzzy approach*;
- Good performance to disturbances, like lateral thrusts or abrupt decelerations;
- Due to the idea of devising *connection trajectories* for the roll, the vehicle cannot actually fall down, unless there's a singularity in the solutions of the internal ODEs; at worst the vehicle doesn't track the path close;
- Actual understanding of the physical behavior of the tires'skid effects, and project of a method to compensate them;
- New method for the *longitudinal controller*, i.e. for the speed regulator, as described in section 5;
- Comparable robustness to the previous method; this was the previous release's most positive aspect;
- Ability to run through opportunely devised closed circuits;

- Debug of some software errors referring to the previous release;

Summarizing, we can claim that our new algorithm has passed all the benchmarks for state-of-the-art software products, and thus can be exploited for commercial purposes.

7.2 Conclusions

The goal of this paper has been to describe how a new heuristic approach can be first embedded into a method and an actual algorithm for tracking a deterministic and predefined track on the flat ground with precision, while retaining the vehicle's balance, accordingly to the dynamics equations. Also, a further step has been to apply this new methodology for a real problem, that of devising an *automatic pilot* for a virtual motorcycle or, in other words, to engineer the controlling scheme which could underlie a software product dedicated to motorcycle simulations.

After describing the control synopsis, we actually constructed step by step the whole algorithm and explained how to demonstrate its stability and robustness.

Many different ideas have been described, all of which are not included in the final optimized version of the same algorithm, but which provide a better understanding of the problem.

Another step has been the integration of this algorithm into a real software product, with all the correlated problems regarding real-world nonlinearities. The goal of obtaining an actual brand new product has been fully attained, along with more refined features, thus eventually obtaining a new improved and enhanced version of the product.

7.3 Future Work

We cannot claim to have reached a final outcome; both the theorical study and the practical work are prone to further refined improvements. First, there might be some new approach in defining the system's model, which could possibly bring new results. For instance, in section 6 we mathematically worked out a new model which could take into account also the pilot; clearly, it would be reasonable to exploit it and devise some controlling schemes which could make use of this new degree of freedom, as suggested in that chapter.

Attention will be surely brought to the problem of demonstrating the vehicle's stability, and results will be assessed in future works.

From an applicative point of view, it's clear that many other ideas could help in improving the software efficiency and performance. We coped in rendering the whole system modular and dynamic, and we concede that we just partially achieved our task: some constants still have to be triggered by hand, and it's up to the engineer's sensibility to understand which value could be the most suitable. In general, results will be at least linearly correlated to the time spent on the software, but we're sure the advantages this software brings are worth the effort.

Bibliography

[1] R.H. Bartels, J.C. Beatty, B.A. Barsky. *An Introduction to Splines for use in Computer Graphics and Geometric Modeling.* Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1987.

[2] G. Benettin, L. Galgani, A. Giorgilli. *Appunti di Meccanica Razionale.* Edizioni Libreria Progetto, Padova, 1999. In Italian.

[3] W.H. Chen, D.J. Ballance, J. O'Reilly. *Model Predictive Control of Nonlinear Systems: Computational Burden and Stability.* CSC Report: CSC-99007, Sept. 1999.

[4] W.H. Chen, D.J. Ballance, J. O'Reilly. *On Attraction Domain of Model Predictive Control of Nonlinear Systems with Input/State Constraints.* CSC Report: CSC-99009, Sept. 1999.

[5] V. Cossalter. *Cinematica e Dinamica della Motocicletta.* Edizioni Libreria Progetto, Padova, 1999.

[6] G. De Nicolao, L. Magni, R. Scattolini. *Stabilizing Receding-Horizon Control of Nonlinear Time-Varying Systems.* IEEE Transactions on Automatic Control, vol. 42, No. 7, July 1998.

[7] M. Fliess, H. Sira-Ramirez, R. Marquez. *Regulation of Non-Minimum Phase Outputs: a Flatness based Approach.* In "Perspectives in Control. Theory and Applications". Springer Verlag.

[8] M. Fliess, J. Lévine, P. Martin, P. Rouchon. *A Lie-Bäcklund Approach to Equivalence and Flatness of Nonlinear Systems.* IEEE Transactions on Automatic Control.

[9] M. Fliess, J. Lévine, P. Martin, P. Rouchon. *Nonlinear Control and Diffieties, with an application to physics.* Contemporary Mathematics, Volume 219, 1998.

[10] E. Fornasini, G. Marchesini. *Appunti di Teoria dei Sistemi.* Edizioni Libreria Progetto, Padova, 1994. In Italian.

[11] R. Frezza, G. Picci, S. Soatto. *A Lagrangian Formulation of Nonholonomic Path Following*. Appeared in "*The Confluence of Vision an Control*", A. S. Morse et al., Springer Verlag, 1998.

[12] N.H. Getz. *Control of Balance for a Nonlinear Nonholonomic Non-Minimum Phase Model of a Bicycle*. American Control Conference, Baltimore, June 1994.

[13] N.H. Getz. *Control of an Autonomous Bicycle*. IEEE International Conference on Robotics and Automation, Nagoya, May 1995.

[14] N.H. Getz. *Dynamic Inversion of Nonlinear Maps with Applications to Nonlinear Control and Robotics*. PhD Thesis, University of California at Berkeley, 1995.

[15] D. Graffi. *Elementi di MECCANICA RAZIONALE*. Pàtron, Bologna, 1964.

[16] A. Isidori. *Nonlinear Control Systems,* 3^{rd} *Edition*. Springer Verlag, 1994.

[17] Yi Ma. *Vision Guided Navigation for a Nonholonomic Mobile Robot*. University of California at Berkeley, May 1997.

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

[19] M. Murray, S. Sastry. *Nonholonomic Motion Planning: Steering using Sinusoids*. University of california at Berkeley.

[20] G. Prokop. *Modeling Human Vehicle Driving by Model Predictive Online Optimization*. Vehicles Systems Dynamics, 2001.

[21] S. Sastry. *Nonlinear Systems. Analysis, Stability and Control*. Springer Verlag, 1999.

[22] D. von Wissel. *DAE Control of Dynamical Systems: Example of a Riderless Bicycle*. PhD Thesis, École Nationale Supérieure des Mines de Paris, 1996.