# Motorcycle Dynamics Library in Modelica

Filippo Donida, Gianni Ferretti, Sergio M. Savaresi, Francesco Schiavo, Mara Tanelli
Politecnico di Milano, Italy
Piazza Leonardo da Vinci 32, 20133 Milano

## Abstract

This paper presents a Modelica library developed for the dynamic simulation of a motorcycle, developed within the Dymola environment (see [1], [2], [3]) and tailored to test and validation of active control systems for motorcycle dynamics. As a matter of fact, as a complete analytical model for two-wheeled vehicles is not directly available due to the complexity of their dynamic behavior, a reliable model should be based on multibody modeling tools endowed with automated symbolic manipulation capabilities. In this work we illustrate the modular approach to motorcycle modeling and discuss the tire-road interaction model, which is the crucial part of the simulator. Moreover, we propose a virtual driver model which allows to perform all possible maneuvers.

*Keywords: Automotive Systems, Motorcycle Dynamics, Multibody Modeling, Dynamic Simulation.*

## 1 Introduction and Motivation

In this work we present a library for the dynamic simulation of a two-wheeled vehicle developed in Modelica, within the Dymola environment. This library is tailored to be employed for test and validation of active control systems for motorcycle dynamics.
The design of a control oriented simulator for two-wheeled vehicles is a very challenging task, as a complete analytical model is not directly available due to its complexity and its high sensitivity to parameters' variations. Accordingly, a reliable model should be based on multibody modeling tools endowed with automated symbolic manipulation capabilities (see *e.g.*, [4], [5], [6]).
As a matter of fact, in two-wheeled vehicle one has to handle strong dynamic coupling between the rigid bodies (front and rear frames, front and rear wheels) and the elastic joints (fork and front and rear suspensions), which make it difficult to devise appropriate reduced model for control purposes (see *e.g.*, [7], [8], [9]). The effort of analyzing well-defined driving conditions on a complete dynamic simulation model seems to be the key for a comprehensive control design for motorcycles. Such approach is well confirmed in the available literature (see *e.g.*, [7], [8], [9]).
In this work, we illustrate the modular approach to motorcycle modeling and present the library architecture discussing all the different packages. In particular, the core of the library lies in the tire-road interaction model, which manages the interaction between tires and road and has a major impact on both ride and handling properties of motorcycles (see *e.g.,* [10]).
Moreover, we propose a virtual driver model which allows to track a predefined trajectory and keep a target speed during different maneuvers.
The paper is organized as follows. Section 2 gives an overview of the overall library architecture, while Section 3 and Section 4 describe in detail the wheel-road interaction model and the chassis and suspension models, respectively. Section 5 discusses the employed aerodynamic model, whereas in Section 6 the road surface model is discussed. Section 7 introduces the implemented *Virtual Driver* model and its functionalities. Finally, in Section 8 some simulation results are presented to assess the validity of the proposed simulation model.

## 2 Motorcycle Dynamics Library Overview

The development of this library is based on the Modelica `Multibody` library. The `MotorcycleDynamics` is a library package for Dymola, developed in Modelica 2.2, which offers all the capabilities needed to perform virtual prototyping for a two-wheeled vehicle. The package shares basically the same structure of the `VehicleDynamics` library, see [11], and it is structured in a tree-based fashion. The root contains the standard motorcycle model and nine sub-packages

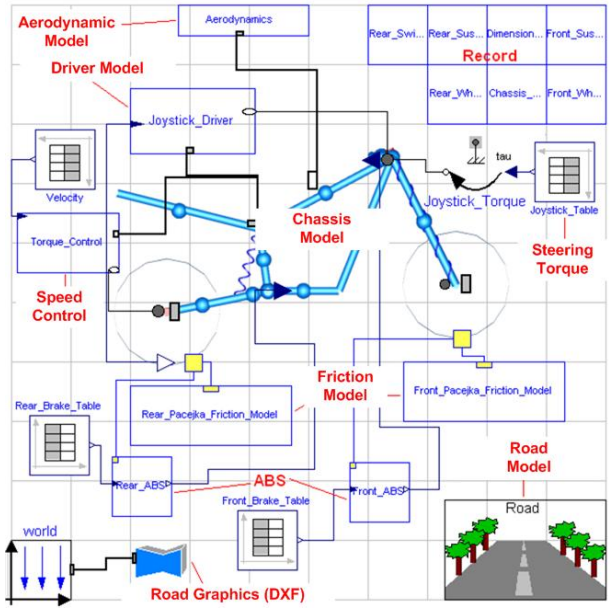



Figure 1: The complete simulation model block diagram.

- `Motorcycle`: is the eleven degrees of freedom motorcycle model with all the default components (*e.g.,* wheels, suspensions, and so on). Figure 1 shows a the complete block diagram of the motorcycle;
- the package `Chassis`: it contains the chassis standard structure and the corresponding geometrical data. It is possible to define new Chassis models and to customize the existing ones;
- the package `Suspension`: it contains the basic spring-damper model, `Base_Suspension`, the front suspension (which also include the front fork and the handlebars) and the rear suspension;
- the package `Rear_Swinging_Arm`: it defines the rear swinging arm, which, again, is fully parametrized and customizable;
- the package `Wheel`: it contains all the wheel geometrical data, the Wheel Road Interaction model, the friction model, the tire relaxation model and the interface model between the wheel and the road surface. Specifically, two different friction models are available to the user. The first is based on a linear approximation of the friction forces and it is reliable at low slip, while the second relies on the Pacejka formula and it's highly nonlinear;
- the package `Braking_Systems`: it includes the braking system model and an Anti-lock Braking System ABS control;
- the package `Driver`: it offers different drivers models and capabilities which allow to perform all the different maneuvers;
- the package `Environments`: it models the road surface and handles also its animation rendering;
- the package `Aerodynamics`: it models the lift and drag aerodynamics forces;
- the package `Example`: it contains some examples which allow the user to explore the library capabilities.

In the following Sections we will describe in detail the structure of each package so as to highlight its peculiarities.

## 3 Wheel-Road Interaction

The core of the Motorcycle Dynamics library lies in tire modeling and in defining the interaction between tires and road, which has a major impact on both ride and handling properties of motorcycles (see *e.g.,* [10]). In fact, the tire allows contact between the rigid part of the wheel (the hub) and the road surface, and it is the means for ensuring adherence to the road and for transferring to the ground longitudinal (*i.e.*, traction and braking) and lateral friction forces, which guarantee steerability.
From the conceptual point of view, the `Wheel_road_Interaction` model (which is

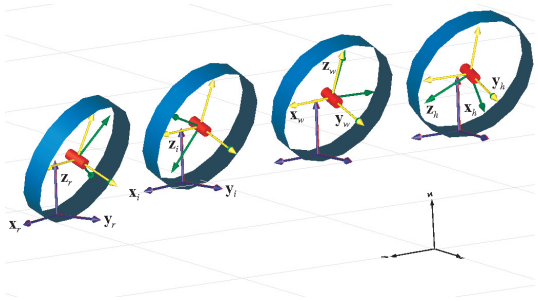


Figure 2: Wheel reference frames: `hub (green)`, `wheel (yellow)`, `P (red)` and `POC (blue)`.

part of the package `Wheel`) computes the forces arising at the contact between tire and road and the connection with the wheel holds *explicitly* through a mechanical connector. In order to compute these forces, it is fundamental to define the correct wheel reference frames. Namely, as it is shown in Figure 2, four reference frames are needed:

- `hub` : is the frame integral to the rigid part of the wheel, that is the hub reference frame;
- `wheel`: is the frame to describe the forward motion of the wheel;
- `P`: is the frame of the ideal contact point between wheel and ground, when no motion is applied to the wheel;
- `POC`: is the frame of the real contact point between wheel and ground, when the wheel is moving.

The said forces are in turn computed according to a description of the road, defined in the road model, which will be presented in Section 6. The connection between `Wheel_road_Interaction` model and `Road` model (which is part of the package `Environments`) is therefore performed *implicitly*, through the `inner/outer` statement, generally used to describe force fields. In this way, the description of the road can be made available to all wheels and vehicles in the scene.

The *hub* frame is attached to the wheel hub, it rotates with the wheel around the axis $\mathbf{y}_h$ and it is the frame associated to the connector trough which the wheel is attached to its rotational joint. The *wheel* frame is attached to the wheel center, with unit vector $\mathbf{y}_w$ coinciding with $\mathbf{y}_h$, while $\mathbf{x}_w$ is the unit vector associated with the wheel velocity direction - obtained as the intersection between the plane orthogonal to $\mathbf{y}_w$ and the road tangent plane at the ideal contact point. The unit vector $\mathbf{z}_w$ completes the frame

$$\mathbf{y}_w = \mathbf{y}_h \quad \mathbf{x}_w = \frac{\mathbf{y}_w \times \mathbf{n}}{|\mathbf{y}_w \times \mathbf{n}|} \quad \mathbf{z}_w = \mathbf{x}_w \times \mathbf{y}_w \,.$$

The *ideal* contact point frame has the origin laying on the road plane, along the direction identified by $\mathbf{z}_w$, the unit vector $\mathbf{z}_i$ is the road normal unit vector $\mathbf{n}$, $\mathbf{x}_i$ coincides with $\mathbf{x}_w$ and $\mathbf{y}_i$ simply completes the frame

$$\mathbf{z}_i = \mathbf{n} \quad \mathbf{x}_i = \mathbf{x}_w \quad \mathbf{y}_i = \mathbf{z}_i \times \mathbf{x}_i \,.$$

The location $\mathbf{r}_i$ of the origin of the ideal contact frame is computed as:

$$\mathbf{r}_i = \mathbf{r}_h - \Delta_z \mathbf{z}_w,$$

with $\mathbf{r}_h$ being the location of the origin of the hub frame and with $\Delta_z$ being the distance between the hub origin and the road plane in the $\mathbf{z}_w$ direction. The *ideal* and *real* contact frames are aligned with each other and differ only in the origin position; the location $\mathbf{r}_r$ of the real contact frame is displaced from $\mathbf{r}_i$ (see *e.g.,* [6], [12]) by the so-called tire trail. Hence

$$\mathbf{r}_r = \mathbf{r}_i + \mathbf{R}_r \begin{bmatrix} \delta_x \\ \delta_y \\ 0 \end{bmatrix},$$

where $\mathbf{R}_r$ is the rotation matrix from the real contact frame to the absolute frame. In turn, the displacements $\delta_x$, $\delta_y$ and $\delta_z$ have been computed[1] as in [6]:

$$\begin{aligned} \delta_x &= f_w R \, \mathrm{sgn}(v_x), \\ \delta_y &= (\Delta_z - R)\sin(\varphi), \\ \delta_z &= (\Delta_z - R)\cos(\varphi), \end{aligned}$$

where $R$ is the wheel radius, $v_x$ is the forward wheel ground contact point velocity and $f_w$ is the rolling friction coefficient [6], which can be computed as:

$$f_w = A + \frac{B}{p} + \frac{C}{p} v_x^2,$$

where $p$ is the tire pressure and $v_x$ the wheel forward velocity, *i.e.,*

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \mathbf{R}_r^T \mathbf{v}_r = \mathbf{R}_r^T \frac{d\mathbf{r}_r}{dt},$$

[1] The $\mathrm{sgn}(x)$ function has been smoothed by replacement with the function $\mathrm{sgn}(x) \approx \frac{2}{\pi}\arctan(kx)$, with $k \approx 10^{10}$.

and $A$, $B$, $C$ are suitable non-negative parameters.
The wheel roll angle $\varphi$ is given by

$$\varphi = -\arctan \frac{\mathbf{z}_w^T \mathbf{y}_r}{\mathbf{z}_w^T \mathbf{z}_r}.$$

The forces arising from the tire-road interaction can be decomposed into a vertical force $\mathbf{F}_z$, a longitudinal force $\mathbf{F}_x$ and a lateral force $\mathbf{F}_y$. The normal force $\mathbf{F}_z$ can be computed as:

$$\mathbf{F}_z = -\left(K_{el}\delta_z + D_{el}\frac{d\delta_z}{dt}\right)\mathbf{z}_r$$

with $K_{el}$ and $D_{el}$ being the tire elastic and damping constants which describe the tire elasticity properties.

As for the longitudinal and lateral forces description, two different models can be employed, according to the specific needs, namely a linear approximation model or a nonlinear model based on the Pacejka formula presented in [10].

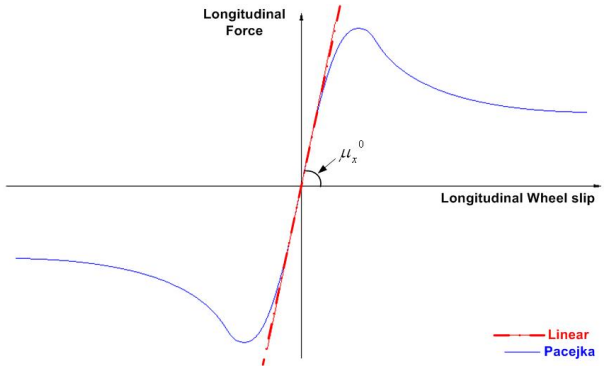



Figure 3: Plot of the longitudinal force $\mathbf{F}_x$ as a function of the wheel slip computed with the nonlinear Pacejka formula (solid line) and its linear approximation (dashed-dotted line).

The user can select either the linear or the Pacejka model, according to the current maneuver and to the analysis purpose of the simulator (see Figure 3 for a comparison between the longitudinal force $\mathbf{F}_x$ as a function of the wheel slip computed with the nonlinear Pacejka formula and its linear approximation). The longitudinal force $\mathbf{F}_x$, is either a traction force $\mathbf{F}_t$ - during acceleration - or a braking force $\mathbf{F}_b$. In the linear model, both traction and braking forces are computed as functions of the longitudinal wheel slip $\lambda$ (see Figure 3), as

$$\mathbf{F}_x = \mu_x^0 \lambda \mathbf{x}_r, \tag{1}$$

where

$$\mu_x^0 = \left.\frac{\partial |\mathbf{F}_x|}{\partial \lambda}\right|_{\lambda=0}$$

and

$$\lambda = \left( R \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}^T \mathbf{R}_r^T \omega_r - v_x \right) / v_x.$$

Finally, the lateral force $\mathbf{F}_y$ is computed as a linear function of the tire side-slip angle (see *e.g.,* [12]), *i.e.,* the angle between the rotational equivalent wheel velocity $\omega R$ and the wheel-ground contact point velocity $v_x$ and of the roll angle. Namely

$$\mathbf{F}_y = \left(k_\alpha \alpha - k_\varphi \varphi\right)\mathbf{F}_z = \left(K_\alpha \alpha - K_\varphi \varphi\right)\mathbf{y}_r, \tag{2}$$

where $\alpha$ is the tire side-slip angle and $\varphi$ is the roll angle. In Equation (2),

$$K_\alpha := \left.\frac{\partial |\mathbf{F}_y|}{\partial \alpha}\right|_{\alpha=0,\varphi=0} \quad K_\varphi := \left.\frac{\partial |\mathbf{F}_y|}{\partial \varphi}\right|_{\alpha=0,\varphi=0},$$

are the cornering and camber stiffness, respectively.
In the Pacejka model, instead, both longitudinal and lateral forces are highly nonlinear functions, namely

$$\mathbf{F}_x = \mathbf{F}_x(\mathbf{F}_z, \lambda, \alpha, \varphi), \tag{3}$$
$$\mathbf{F}_y = \mathbf{F}_y(\mathbf{F}_z, \lambda, \alpha, \varphi). \tag{4}$$

The Pacejka formulas for $\mathbf{F}_x$ and $\mathbf{F}_y$ are based on a semi–empirical model, and the analytical expressions of the forces depend on a huge number of parameters estimated from data. This model has been implemented according to the results proposed in [10], where extensive tests on different tires have been carried out so as to estimate the needed parameters via numerical optimization.
Moreover, the wheel-road interaction model has to take into account the tire relaxation dynamics, which describes the tire deformation due to elasticity properties of the tire material. This phenomenon is modeled computing the real longitudinal and lateral forces as

$$\dot{\mathbf{F}}_{xreal} = \frac{1}{\tau(t)}\left(\mathbf{F}_x - \mathbf{F}_{xreal}\right) \tag{5}$$
$$\dot{\mathbf{F}}_{yreal} = \frac{1}{\tau(t)}\left(\mathbf{F}_y - \mathbf{F}_{yreal}\right), \tag{6}$$

where $\mathbf{F}_x$ and $\mathbf{F}_y$ are as in (1) and (2) if the linear force model is selected, or as in (3) if the Pacejka model is employed. The filter time-varying time constant is $\tau(t) = s_{0l}/v_x$, where $s_{0l}$ is the tire-relaxation length, usually set equal to half of the tire circumference, and $v_x$ is the wheel-ground contact point velocity.
Finally, these interaction forces have to be balanced by the forces $\mathbf{F}_h$ (and torques $\boldsymbol{\tau}_h$) at the hub frame, thus

$$\begin{aligned} \mathbf{0} &= \mathbf{F}_h + \mathbf{F}_{xreal} + \mathbf{F}_z + \mathbf{F}_{yreal} \\ \mathbf{0} &= \boldsymbol{\tau}_h + (\mathbf{r}_r - \mathbf{r}_h) \times (\mathbf{F}_{xreal} + \mathbf{F}_z + \mathbf{F}_{xreal}). \end{aligned}$$

# 4 Chassis and Suspensions Modeling

In order to develop a reliable simulator, the geometry of the motorbike has to be carefully defined. As a matter of fact, very small changes in the center of gravity position may substantially alter the dynamic vehicle behavior. Moreover, we designed the library to be highly reusable and customizable.

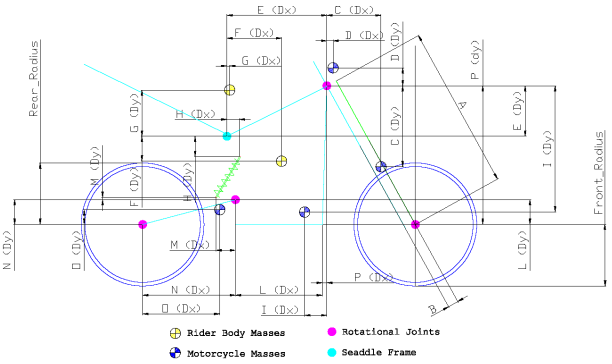



Figure 4: Motorcycle and driver geometry and center of gravity position.

To this end, each subpackage contains a record data structure (see also Figure 1) to store and define all geometric data, *e.g.,* masses, inertias and physical parameters for each component. For example, Figure 4 shows the chassis and driver geometric data which are stored in the `Base_Dimension_MassPosition_Data` record. The `Suspension` package contains the basic damper model `Base_Suspension`, the front suspension `Front_Suspension` (which comprises the front fork, the handlebars and their hard stops `Steer_Stopper`) and the rear suspension `Rear_Suspension`.
Both front and rear suspension employ the same mono-directional double spring-damper model

$$
\begin{aligned}
F_s &= k_s(x - x_0) \\
F_d &= c_d \dot{x} \\
F_{susp} &= -F_s - F_c - PreLoad,
\end{aligned}
$$

where

- $k_s$ is the spring elastic constant and $(x - x_0)$ is the spring compression/deflection;
- $F_s$ is the spring elastic force and it is a function of the spring compression/deflection;
- $c_d$ is the damping coefficient;
- $F_d$ is the damper force and it is a function of the compression/deflection velocity $\dot{x}$;
- *PreLoad* is a constant parameter which depends on the spring tuning and allows to change the static load distribution of the motorbike.

# 5 Aerodynamics

Two are the main aerodynamic forces which need to be taken into account in vehicle modeling, and are shown in Figure 5

- the *Drag Force* which is directed along the longitudinal axis;
- the *Lift Force* which is directed along the vertical axis.

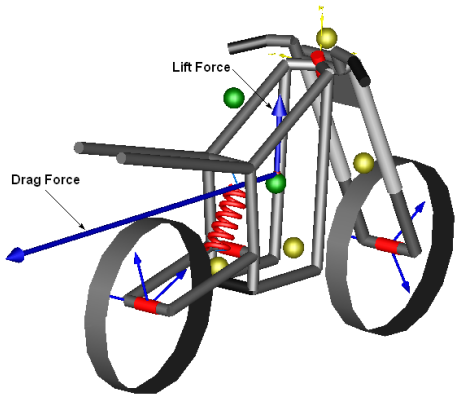



Figure 5: Lift and Drag Aerodynamic forces.

These forces are applied in a specified point called *pressure center* which is generally shifted forward with respect to the chassis center of gravity (see [6]).
The drag force affects the maximum speed and acceleration values and it is proportional to the square of the forward speed. It is computed as

$$\mathbf{F}_d = \frac{1}{2} \rho \, C_x \, A \, v^2, \tag{7}$$

where $\rho$ is the air density, $C_x$ is the drag coefficient, $A$ is the section of the motorbike area in the forward direction and $v$ is the vehicle forward velocity.
The lift force (also proportional to the square of the forward speed) reduces the vertical load on the front and rear tire. It is computed as

$$\mathbf{F}_l = \frac{1}{2} \rho \, C_l \, A \, v^2, \tag{8}$$

where $C_l$ is the lift coefficient and all the other parameters have the same meaning as in (7).

## 6 Road Surface Model and Rendering

The road model is part of the `Environments` package and it has been developed on the basis of the road model given in the `VehicleDynamics` library [11]. However, we extended that model by adding the $z$ coordinate. Specifically, the $z$ coordinate may vary as a function of the $(x,y)$ position, unlike in the model available in the `Dymola` library, where the $z$ coordinate can be set by the user but has to remain fixed for the whole simulation.

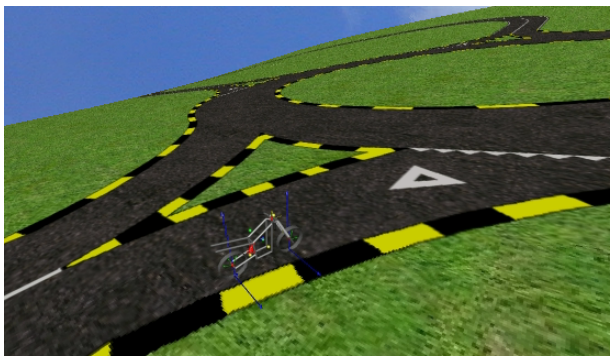


Figure 6: Screenshot of the motorbike and of a road model.

The road characteristics, accessible through the `Road` model, are the following:

- the vector **n**, perpendicular to the surface tangent to the road at the tire-road contact point;
- the quote of the contact point $z$;
- the friction coefficient $\mu$, which allows to model different road conditions (*e.g.,* dry and wet asphalt, icy road and so on).

Hence, the road model is essentially defined by a surface of equation $f(x,y,z) = 0$, from which the relevant normal unit vector can be computed as $\mathbf{n} = \nabla f/|\nabla f|$, and by the road surface characteristics (described via the friction coefficient $\mu$). The computation of the surface gradient has been automatically performed through the statement `partialderivative()`, described in [13].

As for the rendering, it is possible to visualize the road in the Dymola 3D animator via a *dxf* file created with an *ad hoc* Matlab routine and the open source software QuikGrid. Figure 6 and Figure 7 shows examples of road surfaces created with this procedure.

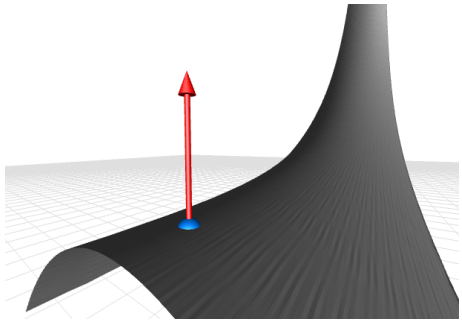


Figure 7: Example of a saddle-like road surface with the normal unit vector and height.

## 7 Virtual Driver

For the effective simulation of two-wheeled vehicles it is necessary to develop a virtual driver model. To this end, we complemented our library with a `Driver` package, which encloses all the control systems needed to perform a desired maneuver.

To this end, the motorcycle model is complemented with two control loops which take care, on one hand, of keeping a constant target speed throughout a specified maneuver and, on the other, of stabilizing the motorbike and following a predefined trajectory.

The closed loop speed control has been implemented as a Proportional-Integral control of the form

$$R(s) = \frac{k_P s + k_I}{s},$$

where the proportional and integral gain can be tuned so to model different drivers' behaviors. This controller takes as input the current forward vehicle velocity and outputs the appropriate traction or braking torque needed to follow the target speed set-point value.

The first task in order to model a virtual driver is to be able of following a desired roll angle profile, which allows to follow a road with turns. Accordingly, the proposed controller computes the steering torque to be applied as a function of the roll angle error, defined as

$$e_\varphi(t) = \varphi(t) - \varphi^o(t),$$

where the value of the set-point roll angle $\varphi^o$ is computed, at any given time instant, as

$$\varphi^o(t) = \frac{v(t)^2 C(t)}{g},$$

where $v(t)$ is the vehicle forward velocity, $C(t)$ is the instantaneous curvature of the trajectory and $g$ is the gravitational acceleration. Hence, the set-point

roll profile is obtained as the concatenation of the instantaneous roll equilibrium values, computed assuming steady-turning conditions [6]. The roll-angle controller has the form

$$\tau_s(t) = k_0\ddot{\varphi}(t) + k_1\dot{\varphi}(t) + k_2 e_\varphi(t), \qquad (9)$$

where $\tau_s$ is the steering torque to be applied in order to track the predefined roll. The presence of the first and second derivatives of the roll angle come from the observation that a real driver tunes his response not only based on current roll angle, but also according to the rolling velocity and acceleration.
If we consider, coherently with the steady turning assumption mentioned above, the set-point $\varphi^o(t)$ to be constant, then we can rewrite (9) as

$$\tau_s(t) = k_0\ddot{e}(t) + k_1\dot{e}(t) + k_2 e(t),$$

so that the transfer function of the roll controller can be expressed in the more familiar form

$$R_\varphi(s) = \frac{k_0 + k_1 s + k_2 s^2}{(1+sT_1)(1+sT_2)}, \qquad (10)$$

where, again, the time constants $T_1$ and $T_2$ can be tuned to obtain faster or slower tracking performance. Such roll controller is then complemented with two more terms, accounting for the direction and position errors, which have to be taken into account in order to track not only a roll profile, but also a desired trajectory, *e.g.,* to drive the bike on a race track. Accordingly, the final expression for the tracking controller (see also [14]) has the form

$$\tau_s(t) = k_0\ddot{\varphi}(t) + k_1\dot{\varphi}(t) + k_2 e_\varphi(t) + k_3\Delta P(t) + k_4\Delta\Psi(t),$$

where $\Delta P(t)$ represents the position error, while $\Delta\Psi(t)$ the direction error. These are computed discretizing the desired trajectory and evaluating the discrepancy between the current position and the closest trajectory point. Such point, though, is chosen according to a *look-ahead* rationale, *i.e.,* as the closest point computed over an error prevision time interval. It is worth noting that this trick is actually performed by human riders, which tend to adjust the vehicle direction looking forward on the road and not based on the current vehicle position. In our controller, we inserted the trajectory tracking performance requirement by computing the overall desired roll profile considering also the position and direction errors $\Delta P(t)$ and $\Delta\Psi(t)$. Such new set-point $\varphi^o(t)$ is then fed as input to the controller (10).

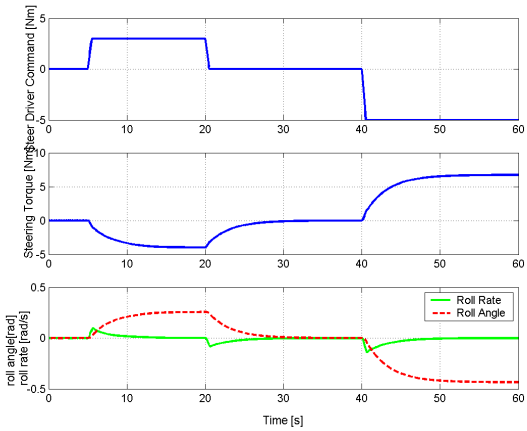
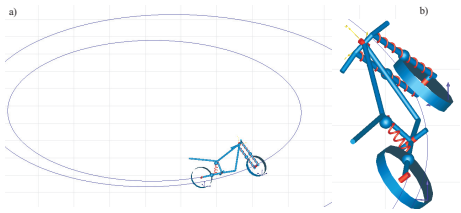



Figure 8: Steering torque input from the driver (top), steering torque applied by the stabilizer (middle), roll rate (bottom, solid) and roll angle (bottom, dashed).

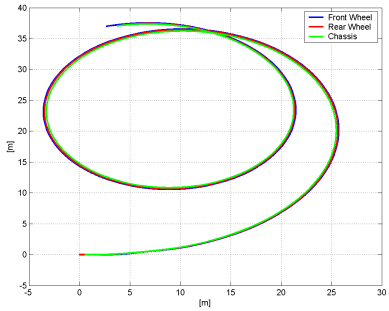
The second possible driver model allows the simulator user to input the desired trajectory as through a joystick, that is to provide as input to the simulator a steering torque profile to be followed. According to the given profile, then, a control loop has been designed which stabilizes the motorbike to a steady state tilt angle with zero roll velocity. As far as the stabilizer design is concerned, in fact, the user input steering torque is treated as a measurable disturbance, and the stabilizer provides as output the steering torque needed to achieve a steady state behavior of the roll angle. The performance of such controller are shown in Figure 8, where the user input is a sequence of three steering torque steps. As it can be seen, the stabilizer provides as output the correct steering torque which allows the motorbike to follow the profile entered by the user. As a consequence, the roll angle and the roll rate experience a transient after each step input before stabilizing to a steady state value.

# 8 Simulation Results

We now show some simulation result which assess the validity of the `MotorcycleDynamics` library. First of all, to validate the trajectory tracking performance a simulation experiment is presented where the motorcycle is commanded to run a circular trajectory, with a radius of 12.5 m at a target speed of 8 m/s. The results obtained with the Dymola 3D-Animator are shown in Figure 9(a), where the solid line depicts the trajectory of the rear hub frame origin, while the simulated trajectory of front wheel, rear wheel and chassis are shown in Figure 9(b). Note in particular the large in-



(a)



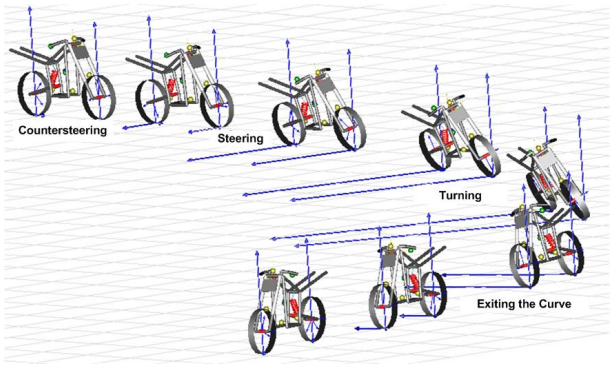

(b)

Figure 9: Simulated trajectory visualized via the Dymola 3D-Animator (top )and plot of the simulated trajectory of Front Wheel, Rear Wheel and Chassis (bottom).

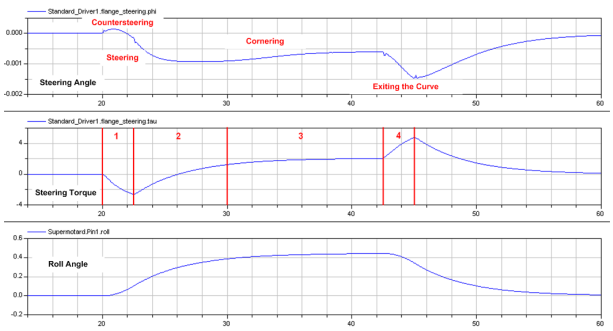
clination of the motorcycle (Figure 9(a)b), needed to run a trajectory with such a high curvature. To better understand the virtual driver behavior, Figure 10(a) shows, via the 3D animator, the four phases during a curve. Namely, the curve is entered with a countersteering, which makes the vehicle tilt correctly, then there is a steering phase which takes to the steady state cornering. Finally, the curve is exited and the vehicle is back to zero tilt angle. Figure 10(b) shows, in the same maneuver a plot of the steering angle, steering torque and roll angle. We also show in Figure 11(a) a screenshot of the 3D animator when a dangerous braking maneuver (panic brake) on a curve is performed and the subsequent fall of the motorbike. Figure 11(b) shows the correct behavior of the simulated roll angle and front wheel sideslip angle which correctly diverge when the bike falls. Finally, we command the motorbike to follow an eight trajectory on a hilly road, modeled via a hyperbolic paraboloid of the form

$$z = \left(\frac{x}{20}\right)^2 + \left(\frac{y}{30}\right)^2 + 0.55. \qquad (11)$$

Figure 12 shows the various phases of the maneuver and the simulated trajectory.




(a)




(b)

Figure 10: Screenshot of a turning maneuver (top) and plot of the steering angle, steering torque and roll angle in the same maneuver (bottom).

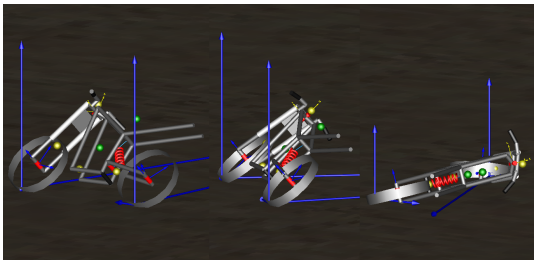
# 9 Concluding Remarks

This work presented the development of the `MotorcycleDynamics` library for Dymola, developed in Modelica 2.2, which offers all the capabilities needed to perform virtual prototyping for a two-wheeled vehicle. The overall architecture of the library has been thoroughly discussed, and its functionalities have been highlighted via simulation experiments.

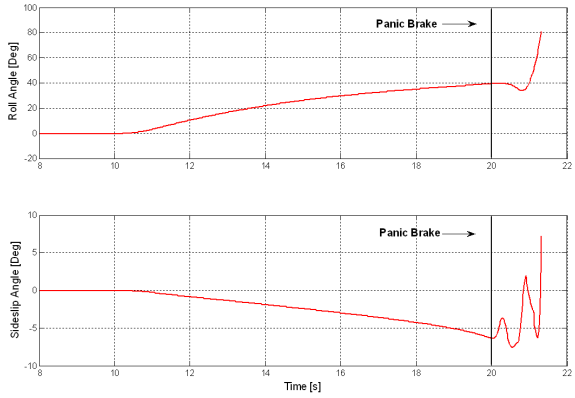
A controller for target speed and trajectory tracking, based on a virtual driver model has been presented and simulation results assessed its validity.

Future work will be devoted to validate the motorcycle model with experimental data and to exploit the library capabilities for active control system prototyping.

Moreover, we plan to extend the driver model so to insert the driver lean angle as an additional degree of freedom in the motorcycle model, in order to model the driver as *active*, capturing his real behavior and its effects on the overall vehicle dynamics.



(a)




(b)

Figure 11: Animation of a motorcycle fall during a braking maneuver on a curve (top) and corresponding roll angle and front wheel sideslip angle (bottom).

## References


[1] M. Tiller, *Introduction to Physical Modeling with Modelica*. Kluwer, 2001.

[2] M. Otter, H. Elmqvist, and S. Mattsson, "The new modelica multibody library," in *Proc. 3rd International Modelica Conference*, Linköping, Sweden, November 3-4, 2003, pp. 311–330.

[3] S. Mattsson, H. Elmqvist, and M. Otter, "Physical System Modeling with Modelica," *Control Engineering Practice*, vol. 6, pp. 501–510, 1998.

[4] R. Sharp, "The stability and control of motorcycles," *Journal of Mechanical Engineering Science*, vol. 13, pp. 316–329, 1971.

[5] V. Cossalter and R. Lot, "A motorcycle multibody model for real time simulations based on the natural coordinates approach," *Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility*, vol. 37, pp. 423–447, 2002.

[6] V. Cossalter, *Motorcycle Dynamics*. Milwaukee, USA: Race Dynamics, 2002.

[7] D. J. N. Limebeer, R. S. Sharp, and S. Evangelou, "The stability of motorcycles under acceleration and braking," *Proc. I. Mech. E., Part C, Journal of Mechanical Engineering Science*, vol. 215, pp. 1095–1109, 2001.

[8] V. Cossalter, R. Lot, and F. Maggio, "On the Stability of Motorcycle during Braking," in *SAE Small Engine Technology Conference & Exhibition*, Graz, Austria, September 2004, 2004, sAE Paper number: 2004-32-0018 / 20044305.

[9] G. Ferretti, S. Savaresi, F. Schiavo, and M. Tanelli, "Modelling and simulation of motorcycle dynamics for Active Control Systems Prototyping," in *Proceedings of the 5th MATHMOD Conference, Vienna, Austria*, 2006.

[10] R. S. Sharp, S. Evangelou, and D. J. N. Limebeer, "Advances in the modelling of motorcycle dynamics," *Multibody System Dynamics*, vol. 12, pp. 251–283, 2004.

[11] J. Andreasson, "Vehicle dynamics library," in *Proceedings of the 3rd International Modelica Conference, Linköping, Sweden*, 2003.

[12] H. Pacejka, *Tyre and Vehicle Dynamics*. Oxford: Buttherworth Heinemann, 2002.

[13] H. Olsson, H. Tummescheit, and H. Elmqvist, "Using automatic differentiation for partial derivatives of functions in modelica," in *Proc. 4th International Modelica Conference*, Hamburg-Harburg, Germany, March 7-84, 2005, pp. 105–112.

[14] L. Marescotti, "Modellazione del sistema pilota-veicolo a due ruote in ambiente integrato Matlab-Adams," Master's thesis, Universitá di Pisa, 2003.

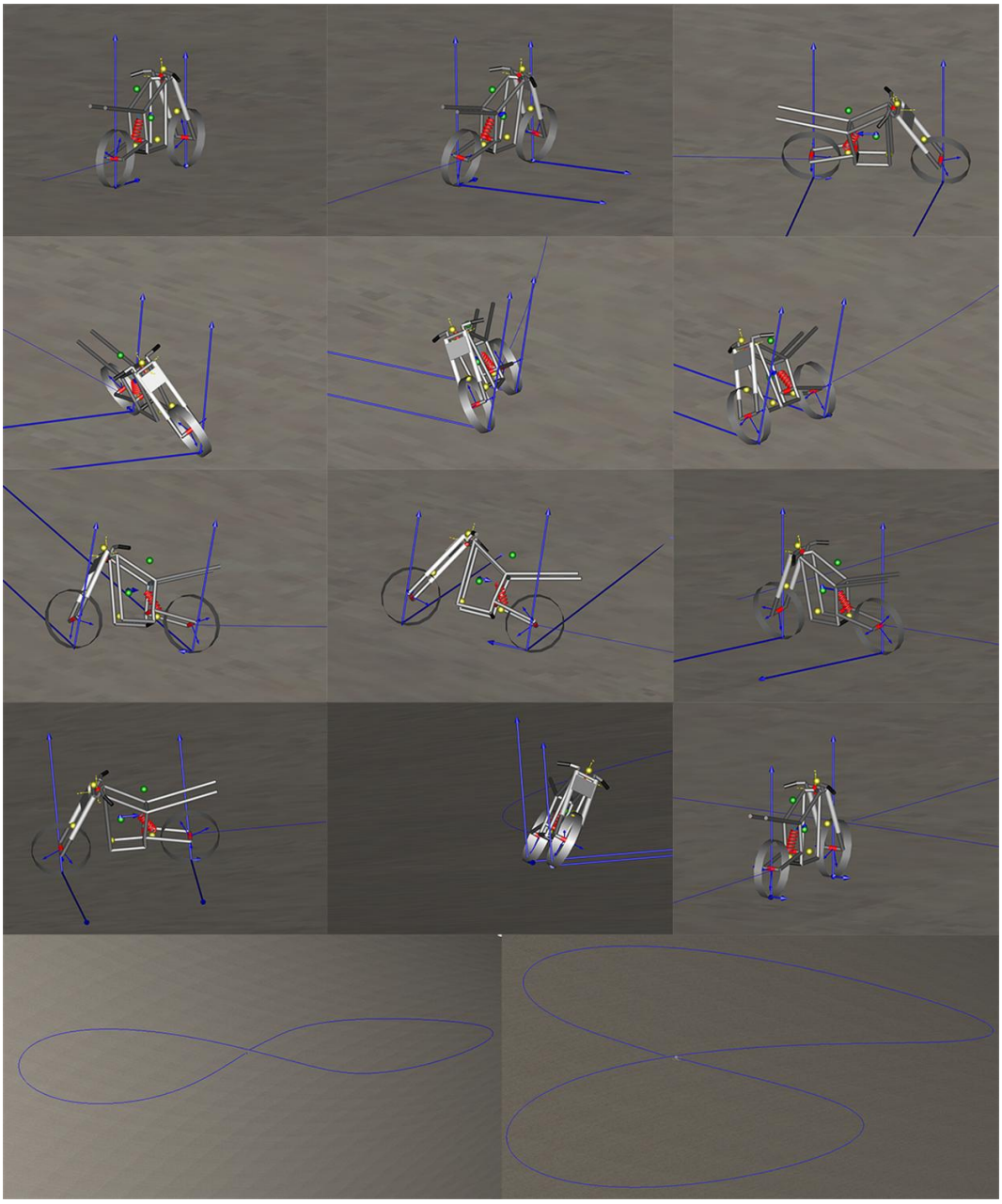


Figure 12: Screenshot of an eight maneuver and of the simulated trajectory.