Proceedings
of the 3rd International Modelica Conference,
Linköping, November 3-4, 2003,
Peter Fritzson (editor)

Mats Beckman and Johan Andreasson
Division of Vehicle Dynamics, Royal Institute of Technology, Sweden:
Wheel model library for use in vehicle dynamics studies
pp. 385-392

Paper presented at the 3rd International Modelica Conference, November 3-4, 2003, Linköpings Universitet, Linköping, Sweden, organized by The Modelica Association and Institutionen för datavetenskap, Linköpings universitet

All papers of this conference can be downloaded from
http://www.Modelica.org/Conference2003/papers.shtml

Wheel model library for use in vehicle dynamics studies

Mats Beckman and Johan Andreasson
KTH Vehicle Dynamics, Sweden
{mb,johan}@fkt.kth.se

Abstract

The implementation of a wheel model library is discussed. The modular structure and its benefits when configuring existing models and developing new ones is presented. The calculation of tyre-road properties is discussed, in particular the contact point estimation on uneven roads and detection of when the tyre looses contact with ground is explained. It is also shown how the implemented Magic Formula model for tyre force generation is validated and the influence of tyre dynamics on simulation time is examined.

1 Introduction

Working with vehicle dynamics modelling often requires a tyre model. A predefined library limits the effort and time needed to model a specific vehicle. This paper presents an extended and improved wheel library based on the library presented in [1].

The first attempt to solve a problem often uses a simple initial model and it is then favourable if the model can be enhanced with more complex features as more knowledge is gained. Thus, the user should be able to reconfigure the wheel models with a minimum of effort. As the complexity of the model increases, it is also desirable to be able to check sub parts separately. As a consequence, a modular structure is favourable and this is derived by identifying the tyre functions.

The models are intended to be used for vehicle simulations and will be included in future versions of the `VehicleDynamics` library [2].

2 Function identification

The function of a complete tyre can be divided into sub functions, each representing a specific tyre feature. This makes it easier to replace sub functions and reuse code. Some sub functions that can be related to the wheel are identified and described below.

Interface An interface handles the communication between the vehicle and the wheel. To easily switch tyre model, a common interface between the vehicle and the wheel is defined. Interfaces should be able to connect to one dimensional (1D), 2D and 3D vehicle models.

Contact point The location where the tyre forces are assumed to act is of substantial interest. Finding this point, orienting it and calculating its speed are necessary in many tyre models.

Vertical dynamics Vertical dynamics is required to model the relation between the contact point and the wheel carrier. This can be modelled as linear spring-damper but it is also possible to model it more elaborately allowing e.g. the wheel to loose ground contact.

Tyre forces The tyre forces acts in the contact point in the road-plane. They are commonly identified as longitudinal force, lateral force, rolling resistance, aligning torque and overturning moment. All or some of these are normally relevant when studying a vehicle dynamics problem.

Roads Examining a vehicle's behaviour may include the use of different topological maps e.g. roads. These roads may be analytic like cross slopes, sine waves, bumps or non analytic like a measured real road, or any analytic road with a random noise component added. The road may also hold more information apart from the topology, this could include entities like friction values or road normal.

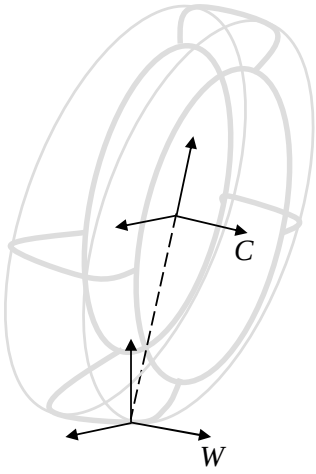

Figure 1: TYDEX frames W and C.

Graphics The visual behaviour of a model often gives the user valuable insights, it is therefore an aid if the wheel simulation can be easily visualised in a 2D or 3D environment, this includes both the visual appearance of the wheel as well some graphics representing the forces acting on it.

3 Definitions

Because of the tyre complexity, several reference frames are necessary to model its behaviour. To ensure that the model structure allows simple addition and reuse of components within new models, the modelling is based on DIN and TYDEX standards. According to the DIN representation the vehicle frame should be orientated so that x points forward, y to the left and z right up. The TYDEX definition of the carrier frame, C, and contact frame, W, is shown in figure 1. The carrier frame is fixed at the vehicle's suspension and the contact frame is located at the intersection of the carrier frame's z-axis and the road plane. For the representation of the graphics, a frame R is used to represent the rotation of the rim.

4 Implementation

In [1], each sub function of the wheel was implemented as a sub model that made it easy to reconfigure the wheel models by drag and drop functionality. The drawbacks were mainly that interfaces of the sub models required code repetition and that the structure made it easy to combine logically incompatible models.

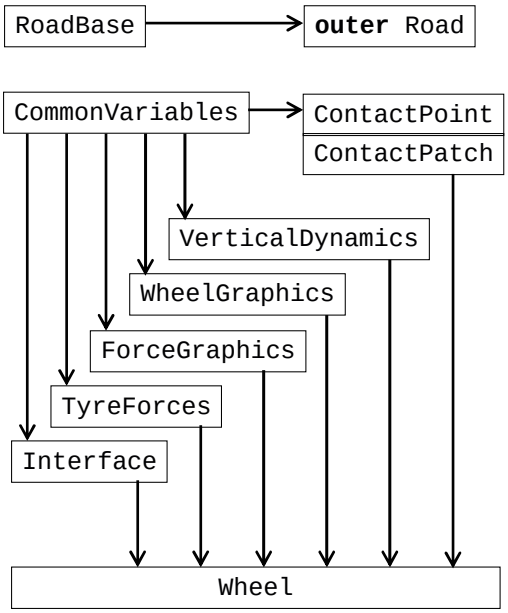

Figure 2: Model structure.

To deal with this problem, the wheel model structure was redesigned with a common variable set, as well as a redesign of the drag and drop configuration to an architecture based on multiple extension. This definition requires more understanding of tyre to set up a new wheel model, thus limiting the risk of improper implementations. Still it is easy to reconfigure an already existing model.

The functionality implemented in the wheel library are found in Figure 2. A brief description of these follows below.

4.1 Common Variables

When extending multiple models, care must be taken so variable collisions are avoided. This is achieved by identifying the wheel common variables in one model and then letting every other model extend this variable set. Included are parameters and variables that describes the properties of the wheel, independent of what kind of implementation is used. Quantities like slip are thus not included since there are several different definitions. Some of the included variables are: 1) Parameters like mass and inertia as well as geometric properties such as spin axis vector and a boolean defining if the wheel is mounted towards left or right so that the model can consider wheel asymmetries. 2)

States of the wheel such as the frames *C* and *W* as well as wheel spin, camber angle and velocities in the tyre-road contact.

4.2 Interfaces

The interface defines the communication between the wheel and the vehicle with two connectors representing the states and the flow through frame *C* and frame *R*, respectively.

The frame *C* connectors are available for one-, two- and three-dimensional mechanics. The three-dimensional connector is either from the old or the new [3] `MultiBody` library. For the two-dimensional case there is the `PlanarMultiBody` library [1] and the one-dimensional connector is standard `Translational`. The two latter interfaces use parameters and external inputs to define un-used dimensions.

The frame *R* connector is normally a one-dimensional standard `Rotational` which is sufficient for most applications and also compatible with the `PowerTrain` library. In some cases, it is relevant to have a more detailed description of the power train and thus, frame *R* is also available with the three-dimensional connectors mentioned above.

4.3 Tyre Forces

The tyre forces are described in a separate model. The wheels library today, contains the Magic Formula model [4, 5], the Rill model [6] and the brush model [5].

4.3.1 Magic Formula model

The Magic formula was originally presented in [4], the idea is to represent the tyre force $f(s)$ characteristics with a trigonometric function

$$f(s) = D\sin(C\arctan(Bs - E(Bs - \arctan Bs))) \quad (1)$$

This has been improved successively and considers now aspects such as camber, vertical load and transient behaviour[1]. The level of detail is controlled by user modes (UM), according to the specification in Table 1 The Magic Formula is a similarity approach, which means that it is based on the use of basic characteristics typically obtained from measurements. Through distortion, rescaling and multiplications, new relationships are obtained to describe off-nominal conditions. This classifies the Magic Formula as an semi-empiric model.

	steady state user modes
UM0	only vertical spring
UM1	pure longitudinal slip
UM2	pure lateral slip
UM3	longitudinal and lateral slip (not combined)
UM4	combined slip forces, steady state
	transient user modes
UM11	pure longitudital slip
UM12	pure lateral slip
UM13	longitudinal and lateral slip (not combined)
UM14	combined slip forces

Table 1: Specification of the Magic Formula user modes.

Magic Formula models the dynamic properties by calculating a dynamic slip in the longitudinal and lateral direction and then use the steady state force calculation.

4.3.2 Rill model

The Rill model calculates the slip in steady-state and calculates a corresponding tyre force with a curve fit using initial inclination $\partial f/\partial s(s = 0)$, location and magnitude of max force $f_{max} = f(s_{max})$ and location and magnitude of force when the whole contact patch is sliding $f_{slide} = f(s_{slide})$ as parameters. The nonlinear dependence of vertical load is handled by an interpolation between a set of the parameters for predefined load cases. This classifies the Rill model as semi-emperic.

Camber influence, roll resistance as well as overturning and aligning moment ar then defined based on geometrical considerations. Unlike the Magic Formula model, the dynamic effects are modelled as a spring-damper filter applied after the steady state forces have been calculated.

[1] The magic formula version implemented is 5.0.

4.3.3 Brush model

Unlike the Rill and Magic Formula models, which are semi-emperic, the brush model is analytical. The idea is to discretisise the tyre with elastic bristles that touches the road plane and can deflect in a direction parallel to the road surface. Their compliance represent the elasticity of the combination of carcass, belt and actual tread elements of the real tyre. The effect of each bristle is added to a set of forces and torques acting on the tyre.

4.4 Contact point calculation

Using frame C and information about the road profile, the `ContactPoint` calculates location and orientation of frame W, indicated in Figure 1. Additionally, the distance between the frames and its time derivative as well as the camber angle are calculated.

The orientation of the frames are related as described by their unit vectors:

$$\begin{aligned} {}^{W}e_z &= n \\ {}^{W}e_x &= {}^{C}e_y \times {}^{W}e_z \\ {}^{W}e_y &= {}^{W}e_z \times {}^{W}e_x \end{aligned} \tag{2}$$

where n is the road normal. The actual location of frame W can be found by iteration as suggested in [6] which was implemented in Modelica in [7]. The idea is to start at the location of frame C, r_C and define a first approximation of the contact point, $r_{P1} = -R_0 {}^{C}e_z$ where R_0 is the undeformed wheel radius. The (x,y)-coordinates are then used to find the actual road altitude, z, giving $r_{P2} = (r_{P2}[1], r_{P2}[2], z)$. Due to camber, tyre deflection, and road uneveness, r_{P2} is normally not located along the line between r_C and r_P1, r_P2 is then projected onto this line giving r_P3. However, if the road is uneven r_{P3} is no longer located at the road surface. Then r_{P3} is used as a new r_{P1} and the calculations can be iterated until the accuracy is sufficient.

However, the iteration may also diverge depending on the road surface, and the method has difficulties to cross sharp edges. Thus this method is not suitable when using e.g. meshed roads. Instead, the contact point is calculated using the deformation of the tyre from the previous time step. This allows the wheel to travel over unevennesses without causing numerical problems. Also a simple model that assumes a flat surface can be used to speed up simulations when apropriate.

4.5 Contact patch filtering

In reality, the contact between the road and tyre is spread over a patch about 1 dm^2, depending on tyre dimensions, pressure, load and cambering. The tyre force models that are based on a contact point representation all require that the actual patch is similar to the test conditions when the tyre parameters were estimated. Different tyre pressure and load is normally tested in a test rig and can thus be handled by the tyre force model. However, when the road unevenes is significant within the tyre patch range, these have to be accounted for by some kind of filtering. In [5] a filter is suggested that lets a set of solid ellipses travel over the road profile and geometric calculations then give a resulting road plane that is used for the tyre force calculations.

This method is not implemented since it is believed to be very time consuming. Instead, a simpler filtering is implemented based on either a rectangle or a cross. Assuming that the contact patch can be represented by rectangle, then the resulting road plane is calculated as:

$$\begin{aligned} k &= \sum_{i,j} k_{i,j} \\ z &= \frac{1}{k} \sum_{i,j} k_{i,j} z_{i,j} \\ \frac{\partial z}{\partial x} &= \frac{1}{k} \sum_{i,j} k_{i,j} \frac{z_{i,j} - z}{\Delta x} \\ \frac{\partial z}{\partial y} &= \frac{1}{k} \sum_{i,j} k_{i,j} \frac{z_{i,j} - z}{\Delta y} \end{aligned} \tag{3}$$

where $k_{i,j}$ is a weight distribution. The number of evaluation are $i \cdot j - 1$. To speed up the calculation a cross shape can be used instead reducing the number of evaluations to $i + j - 2$.

4.6 Vertical dynamics

The vertical dynamics handles the load carrying task of the tyre belt. Typically this is modelled as a spring-damper with the exception that the tyre only generates vertical force when in contact with the ground. In the basic case, this is formulated as

```
contact = R < R_0;
```

In this case, there is contact as long as the distance between the wheel centre and the ground, `R`, is shorter

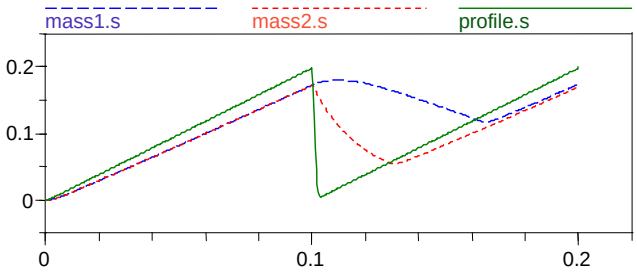

Figure 3: The simpler model (dash-dotted red) and the more advanced model (dashed blue) passing over a road profile (solid green).

than the undeformed tyre radius, `R_0`. However this assumes that the vertical dynamics of the tyre belt is infinitely fast and leads to problem when the road surface is uneven.

When the tyre dynamics needs to be considered the following model is used

```
v = der(R);
v1 = -R1*c/d;
der(R1) = if (v < v1 and contact)
     then v else v1;
contact = R1 >= R;
```

Here, an additional state `R1` is introduced to keep track on the actual deformation of the tyre. This state is limited so that once the tyre is compressed, it can cannot increase faster than the dynamics of the tyre. This needed when travelling over a road surface with a sudden quick altitude decrease, i.e. a pot hole. This is illustrated in Figure 3 where the two models are passing over a road profile[2]. When passing the bump, the simple contact model fails to detect the loss of contact and forces the tyre downwards in an unnatural way trace 2. The more advanced model consider the fast change of the road plane which results in a better behaviour, trace 1.

4.7 Road

The wheel models need information about the altitude and the road surface condition at the tyre-road contact. These should however be independent of the wheel model and thus this road information is stored in a separate model together with graphical information. Since the road and the tyre exchange data, the standard way in Modelica would be to have a road component which is connected to all wheels of a vehicle. This would however result in a close coupling of vehicle and road model. Instead the `inner/outer` Modelica language constructs are used, that only requires that the road model is defined at the top level of the vehicle model.

Information about the road condition is normally required once when generating tyre forces while the altitude may be called several times by both `ContactPoint` and `ContactPatch`. Thus it must be possible to call altitude and road condition separately and to deal with this, a basic road is defined as:

```
partial model RoadBase
  replaceable block Altitude  = BaseXY;
  replaceable block Condition = BaseXY;
  parameter Integer nAltitudes=0;
  parameter Integer nConditions=0;
  Altitude altitude[nAltitudes];
  Condition condition[nConditions];
end RoadBase;
```

Here, `BaseXY` is a `block` taking two inputs `Real x,y` and returns `Real z`. This can be used for both altitude and road condition.

In the `CommonVariables`, a model `Road` is defined as:

```
  outer model Road = RoadBase;
```

A model can then be instantiated whenever needed in the wheel model according to:

```
  Road road(nAltitudes=n1, nConditions=n2);
```

giving a `road` containing vectors `altitude` and `condition` with `n1` and `n2` elements, respectively. In the top model, the desired road can be selected by setting:

```
  inner model Road = DesiredRoad;
  Road road;
```

In this case, the actual road model in the instances of the wheel is `DesiredRoad`, which easily can be swapped to any other road extending the `RoadBase` using the `redeclare` syntax.

4.8 Graphics

The `Wheels` library contains graphics that represents the wheel as well as forces generated in the contact point. The visualisation of the road is stored in the road model described above.

[2]The profile is exaggerated to make the difference appear clearly.

5 Validation

The Magic Formula implementation has been validated using the magic formula tyre already implemented in ADAMS/Car [8]. ADAMS/Car does not have a convenient configuration method to let the user decide tyre inputs like load, slip and camber, the easiest way to validate was to pick a full vehicle manoeuvre and export the slip, tyre load and camber angle for each wheel and then use these values as input to the Modelica model. Thus making sure that the same inputs generate the same output, Figure 4.

The chosen maneouvre is a break in turn, which leads to both lateral and longitudinal slip. The braking force was set high enough to cause lock up, in Figure 4 this happens at $t = 3$s when κ reaches -1.

The validation model is realised as an interface. The variables unknown to the tested sub model are provided by the test interface. These variables are set as parameters or external inputs and are controlled by the user.

6 Usage

The `Wheels` library allows the user to use wheel models already implemented, to configure these and to implement own models within the structure. All wheel models extends an interface model, thus allowing the use of the `replaceable` syntax along with `choicesAllMatching`, presenting all compatible wheel models to the user. This makes it easier to handle wheel model changes in a full vehicle model. New models can be made and the structure makes reuse of elements from models already implemented intuitive and code effective.

The use of replaceable models makes it possible to do most testing with two flexible models that can be configured with drop-down boxes as illustrated in Figure 5. In the first rig the wheel can be either free, constrained or affected by forces or torques. Also the test road and of course the tested wheel can also be exchanged in the same manner. The second rig is a mass mounted on the wheel via a spring-damper, representing the suspension and the distributed body weight. Since only one wheel is used, this is often referred to as the *quarter car model*. Except for the vertical motion, the rig can be controlled as previously mentioned.

Figure 6 shows an animation where a wheel is

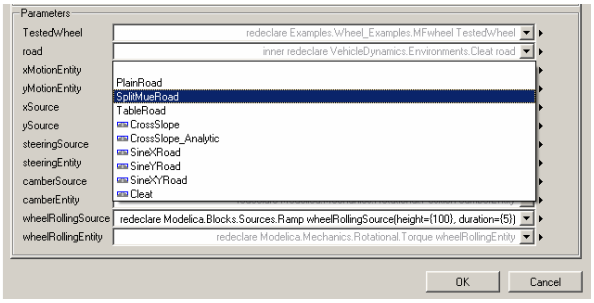

Figure 5: Parameter with drop down-boxes showing the available roads.

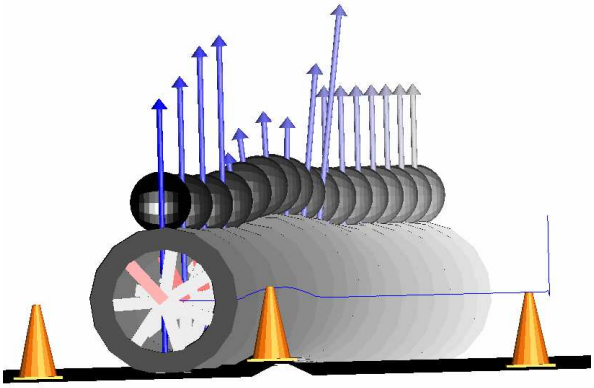

Figure 6: Animation view of tyre passing over a cleat.

used in a quarter car model. The model passes a cleat located by the centre cone, the normal force vector shows the tilt of the contact patch when the wheel ascends the cleat.

The CPU time required for different manoeuvres and different tyre configurations is measured to give an idea of the computational effort required. The maneouvre simulated is a start from standstill with an applied torque on the drive shaft. At time $t = 4$s, when speed is gained, a ramp signal is applied turning the wheel around its vertical axis 4^o in 0.1s. At time $t \approx 6.5$s the wheel meets a slope that is reducing the wheel's speed until it stops at time $t \approx 7.2$s and starts rolling back down, reaching the flat surface at time $t \approx 7.7$s.

The CPU time required for this maneouvre at a 1.5GHz Pentium4 with 512Mb ram is measured and presented in Figure 7. The models compared are Magic Formula user modes 14 (MF UM14) and 4 (MF UM4) as well as a modified user mode 14 with a simpler transient slip model. The Rill model is also compared to these.

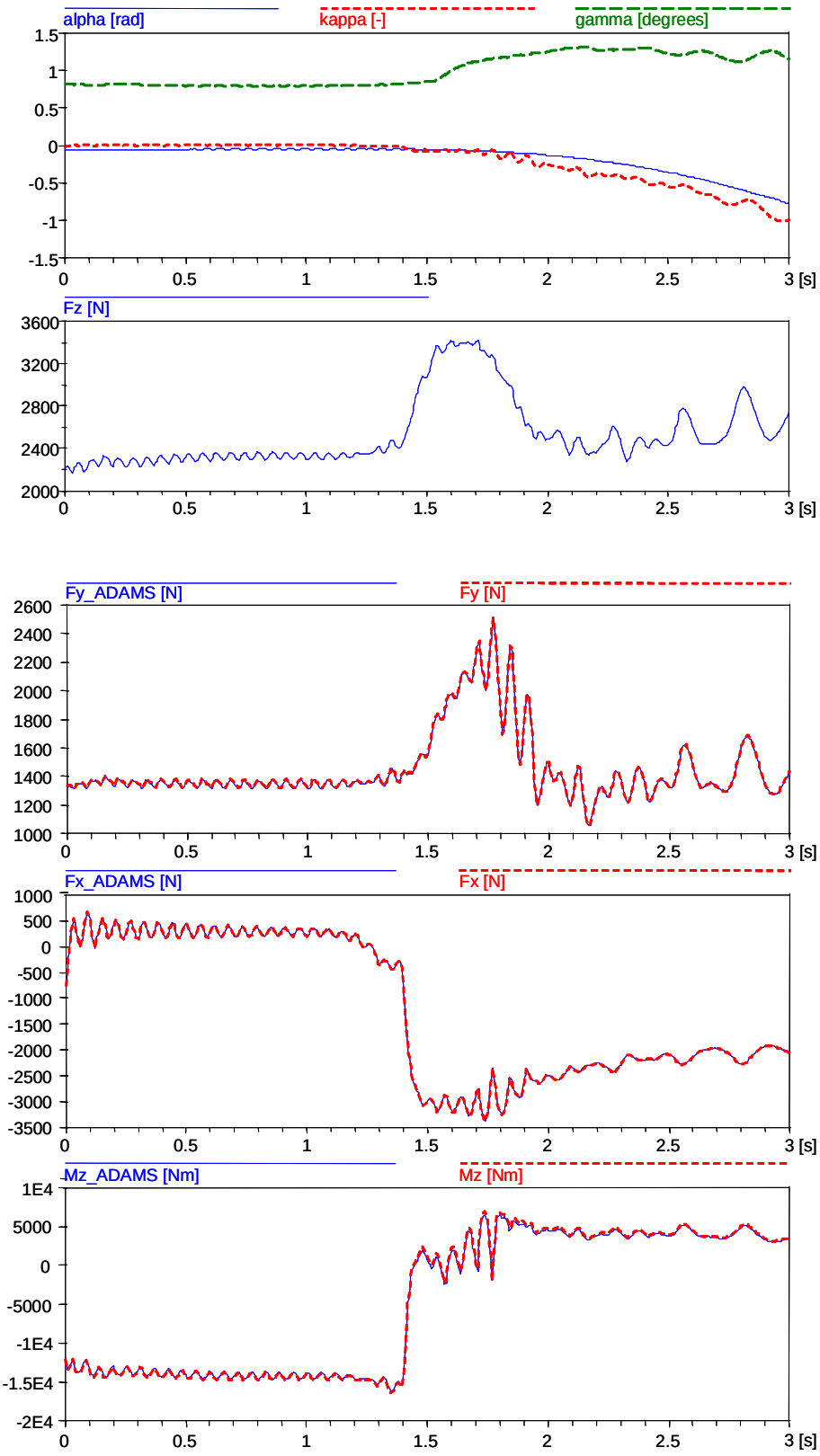

Figure 4: Validation plots comparing the result from the `Wheels` library and the ADAMS output.

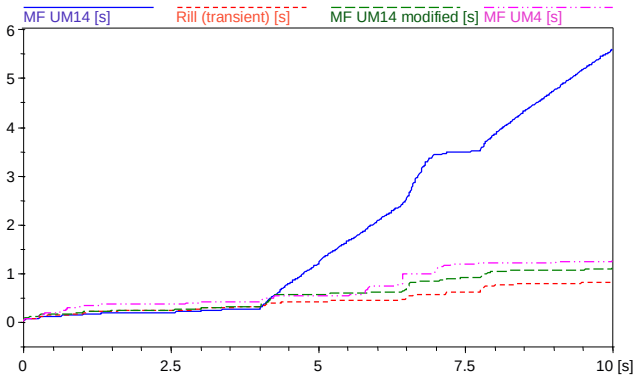

Figure 7: CPU time for different tyre models.

Figure 7 shows that UM14 requires a significantly more computational effort compared to UM4. The extra computing effort origins from the transient slip model implemented in UM14. The modified UM4 has a transient behavior modelled as a first order filter with coefficients only depending on wheel spin velocity. This model lacks physical characteristics that UM14 features such as relaxation length and load dependance. However the modified UM14 model is capable of starting from standstill which is not possible with a steady state model a the computional effort is on par with the original UM4. A disturbance to the UM14 models seems to oscillate for a longer period of time than other models like Rill. Besselink [5] has proposed a damping term that may address this issue.

7 Conclusions

The `Wheels` contain ready-to-use tyre models as well as components that can be used to design own models. The modular structure makes it easy to reconfigure existing models and to reuse code when adding new functionality.

Compared to the previous version of the library, the models are further validated and new functionalities are implemented such as a better vertical behaviour and ability to handle uneven roads. Additionally, interfaces to the new `MultiBody` library are added.

8 Acknowledgements

This work has been funded by KTH Vehicle Dynamics and the Driving Dynamics project within the Swedish National Research Programme ”The Green Vehicle/FCHEV”. The authors would also like to thank Martin Otter and Hans Olsson for valuable support.

References

[1] J. Andreasson and J. Jarlmark. Modularised Tyre Modelling in Modelica. In Peter Fritzson, editor, *Proceedings of the 2nd International Modelica Conference*, Oberpfaffenhofen, March 2002. The Modelica Association and Deutches Zentrum für Luft- und Raumfahrt.

[2] J. Andreasson. Vehicledynamics library. In Peter Fritzson, editor, *Proceedings of the 3rd International Modelica Conference*, Linköping, November 2003. The Modelica Association and Linköping University.

[3] M. Otter, H. Elmqvist, and S.E. Mattson. The new Modelica MultiBody library. In Peter Fritzson, editor, *Proceedings of the 3rd International Modelica Conference*, Linköping, November 2003. The Modelica Association and Linköping University.

[4] E. Bakker, H.B. Pacejka, and L. Lidner. A new tire model with application in vehicle dynamics studies. *SAE transactions, paper 890087*, pages 83–93, 1989.

[5] H.B. Pacejka. *Tyre and vehicle dynamics*. Butterworth Heinemann, 2002.

[6] G. Rill. *Simulation von Kraftfahrzeugen*. Vieweg, 1994.

[7] J. Andreasson, A. Möller, and M. Otter. Modeling of a racing car with Modelicas MultiBody library. In Peter Fritzson, editor, *Proceedings of the Modelica’2000 Workshop*, Lund, October 2000. The Modelica Association and Lund University.

[8] ADAMS, Mechanical Dynamics Inc. http://www.adams.com/.