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

Hilding Elmqvist, Sven Erik Mattsson, Hans Olsson, Johan Andreasson, Martin Otter, Christian Schweiger, Dag Brück
Dynasim; Royal Institute of Technology; DLR:
Real-time Simulation of Detailed Automotive Models
pp. 29-38

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

Real-time Simulation of Detailed Automotive Models

Hilding Elmqvist[1], Sven Erik Mattsson[1], Hans Olsson[1],
Johan Andreasson[2], Martin Otter[3], Christian Schweiger[3], Dag Brück[1]

[1]Dynasim AB, Research Park Ideon, 223 70 Lund, Sweden, http://www.dynasim.se/,
{Hilding.Elmqvist, SvenErik.Mattsson, Hans.Olsson, Dag.Bruck}@Dynasim.se
[2]KTH Vehicle Dynamics, 100 44 Stockholm, Sweden
http://www.ave.kth.se/, Johan@fkt.kth.se
[3]German Aerospace Center (DLR), Institute of Robotics and Mechatronics, Oberpfaffenhofen,
82234 Weßling, Germany,
http://www.robotic.dlr.de/control/, {Martin.Otter, Christian.Schweiger}@dlr.de

Abstract

This paper describes typical modeling and real-time simulation issues that occur in automotive applications. Real-time simulations of detailed Modelica benchmark models for chassis and powertrain are presented. They demonstrate the powerful real-time capabilities of Dymola and the Modelica modeling language. One of the benchmark models for vehicle dynamics is a detailed model with 72 degrees-of-freedom with bushings in both the front and rear wheel suspensions. It was simulated in real-time with a sample rate of 500 Hz on the RT-LAB environment from OPAL-RT using a Pentium 4, 3066 MHz processor. This is made possible by Dymola's unique and elaborate symbolic processing of the model equations.

1 Introduction

Hardware-in-the-loop simulation (HILS) has become common practice in automotive development. In order to cope with the real-time constraints, only rough models are often used. In this paper, we present means to symbolically manipulate models with a high level of detail in such a way that the simulation can be performed in real-time. The effectiveness is demonstrated by several benchmark examples and by corresponding simulation results.

The methods are implemented in the simulation environment Dymola [3, 4] that uses the Modelica [7] modeling language for describing the models. It is described how Dymola solves certain difficult problems in hardware-in-the-loop simulation of automotive systems. Two types of benchmark models have been chosen to demonstrate the capabilities of Dymola: a transmission model and a set of vehicle dynamics models.

A transmission gearbox is somewhat special because the connection structure changes due to the engagement of clutches and brakes. Furthermore, effective inertias need to be calculated for each of the possible structures. Dymola handles this by appropriate preparation of the equations by symbolic methods before generating the code for the target HILS system.

Vehicle models of different complexities can be used for analysis. Traditionally, idealized models of wheel suspensions have been used, neglecting fast dynamics due to bushings and replacing them with ideal joints or just look-up tables. Dymola has special numeric methods to handle such cases. These methods require elaborate symbolic preprocessing of the equations. One of the benchmark models has 72 degrees-of-freedom with bushings in both the front and rear wheel suspensions. It was simulated in real-time with a sample rate of 500 Hz.

Dymola generates C code which can be used in Simulink and by use of RealTime Workshop downloaded to different HILS targets. Evaluation of the benchmark problems has been made on RT-LAB from OPAL-RT [8], demonstrating real-time performance of complex models.

2 Power train simulation

We will consider modeling and simulation of automatic gearboxes. The figure below shows a typical Modelica model of a gearbox (Lepelletier wheelset, 6-speed, from the commercial Modelica PowerTrain library [10] available from Dynasim; usable, e.g., for the automatic gear box ZF 6 HP 26

from ZF). The model includes planetary and Ravigneaux gear sets, clutches, brakes and inertias.

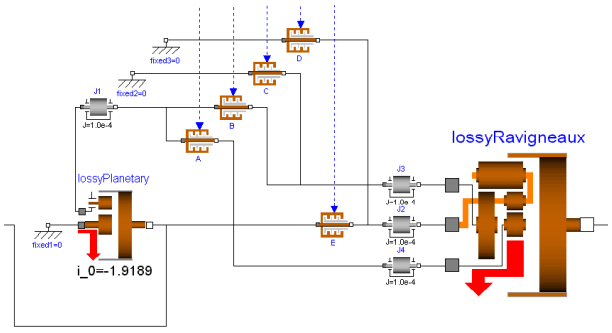

Figure 1: Gearbox model

2.1 Special problems

Simulation of gearbox models in real-time poses special problems. If detailed models of the friction of the clutches and brakes are used, the models become stiff. Typically, ideal friction models are used instead. This means that the number of degrees-of-freedom (DOF) changes if a clutch or brake is stuck or not. This can be handled by constraining the relative acceleration, when in stuck mode, to be zero.

Fast sampling

The differential equations of the gearbox need to be solved at a high speed. The electronic control unit (ECU) for the transmission typically samples its inputs and calculates new control signals every 10 milliseconds. In order to reduce effects of delays due to lack of synchronization, the model variables need to be determined every millisecond.

Accuracy and discontinuities

Special attention is needed to accurately calculate angular velocity. This is important because the angular velocities of the various wheel sets are typically output from the model to the hardware and input to the ECU. The control algorithm of the ECU acts differently when the angular velocity is close to zero. Thus it is important to calculate small velocities accurately. Another reason to achieve high accuracy is that one might otherwise get drift in the angle calculations. The difficulty in achieving high accuracy in the angular velocities close to zero is the highly nonlinear behavior when a clutch sticks. The torque of the clutch in sliding mode is calculated as a function of angular velocity. When the clutch sticks, the constraining torque is instead calculated in such a way that the relative angular acceleration stays zero. There are thus jumps in the relative angular acceleration.

Event handling

Integration algorithms for non-real-time simulation typically handle discontinuities, such as the one above for friction, by detecting when certain variables cross a boundary. They then calculate the time of the event by iteration and then change the step size to advance the time exactly to the time of the event (crossing). Also for real-time applications, the Dymola run-time system includes handling of calculation of the event time. This is done with little overhead and without iteration. The normal solving of the differential equations is for the real-time case performed with fixed step size. However, at an event the step size is decreased to hit the time of the event. In order to synchronize with real-time again, the size of the next step is increased such as the sum of the two steps around the event is equal to two normal steps. This procedure introduces a small synchronization error during one step, but gives better accuracy in the solution. It has successfully been utilized for gearbox HIL simulations for ECU testing.

Event propagation

After an event, for example if a clutch begins to slide, there might be an immediate event as a consequence. Another clutch might get stuck because its torque decreases below a certain threshold. Before a numerical solution of the differential equations is resumed, event propagation needs to be performed in order that all variables get consistent values. Dymola generates code for iterating the equations, called event iteration, until all Boolean mode variables have converged. This typically takes 1-3 extra evaluations of the equations, i.e., the calculation time to handle such an event might exceed the available time for the step. This is typically handled by configuring the HILS system to allow a certain number of overruns.

Effective inertia calculation

The effective inertias depend on the selected gear. Calculation of effective inertias shows up as systems of equations that need to be solved simultaneously.

Dymola symbolically converts the differential and algebraic equations (DAE) to an algorithm for calculating the derivatives. The integration algorithm uses the derivatives to update the state variables. Many times, the derivative algorithm is

just a sequence of assignment statements for algebraic variables and derivatives. However, the conditional constraint equations for torques and accelerations in the clutch and brake models implies that, in order to solve for the accelerations, a system of simultaneous equations needs to be solved. Dymola automatically calculates the coefficients of the linear system of equations and invokes a numerical solver for larger systems of equations. Small systems of equations are solved by producing symbolic code. The effective inertia typically shows up as the determinant of such a coefficient matrix. It should be noted that this is not a domain-specific procedure, but Dymola does it automatically by solving the systems of equations.

Underdetermined models

In certain cases, several clutches are engaged, giving parallel paths for the power. In such cases, the torque at each clutch cannot be determined individually; only the sum can be determined. Mathematically, this shows up as a singular system of equations. However, it is possible to find consistent solutions. Dymola determines one such consistent solution.

2.2 Transmission example

As a benchmark example, we will consider modeling of a 6 speed gearbox (Lepelletier wheel-set, e.g. ZF 6 HP 26) together with a simple vehicle and driver model. This model is suitable for carrying out driving cycle shift strategy analysis and is available in the Powertrain library. The hierarchical structure of the model and the 3D representation used for animation is shown in the picture below.

The engine model is based on steady-state engine maps. The ECU function included in this model controls idle and maximum speed, both constant limits, by a proportional controller. The transmission is a detailed model of an automatic transmission and incorporates a torque converter with a lock-up clutch. The gearbox itself is of Lepelletier type, which provides six different gear ratios. It is modeled using basic gearbox elements, inertia elements and different clutches and brakes. The different gear ratios are a result of applying different pressures to the clutches and the brakes in order to engage or disengage them.

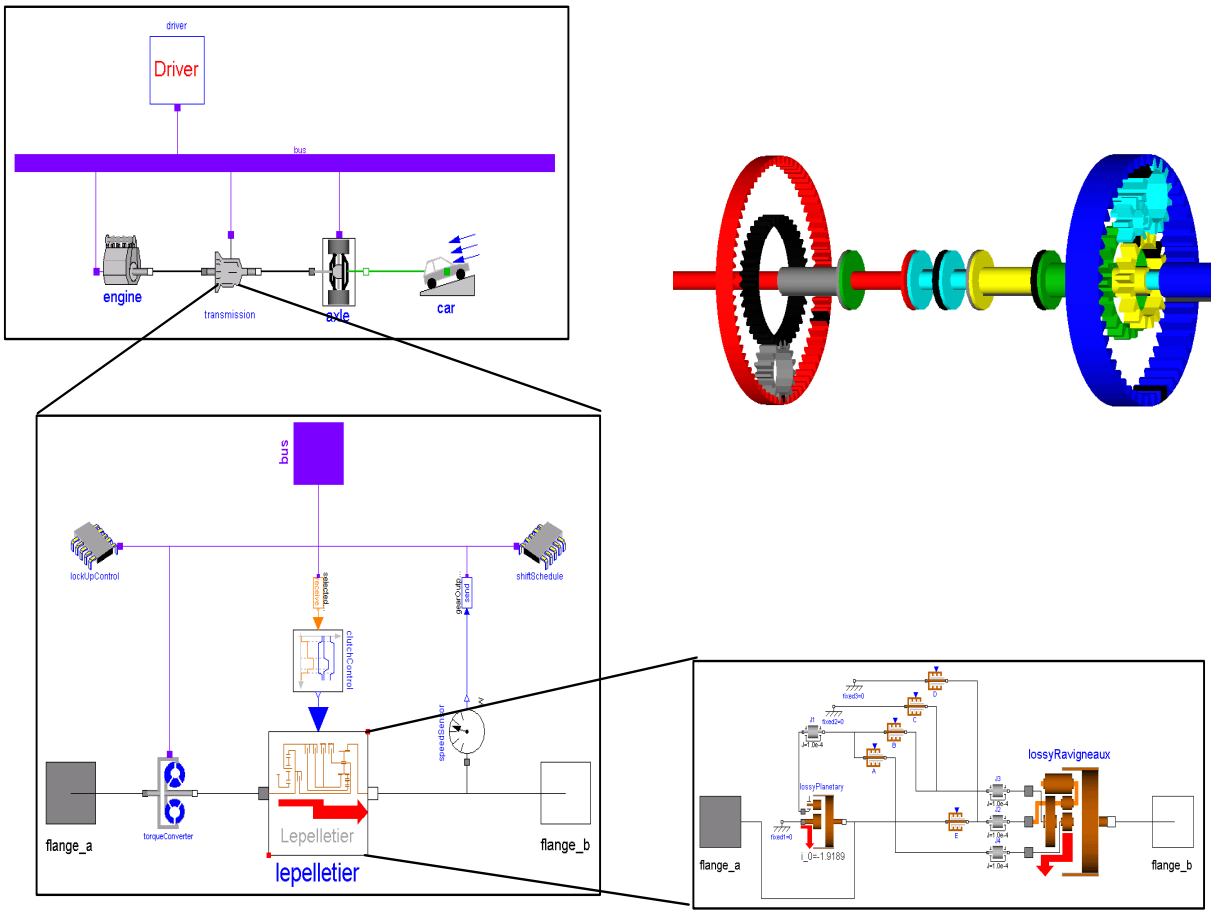

Figure 2: The transmission example with the gearbox model and its animation

The driveline model is essentially a rigid model with no compliance in the drive shafts and no tire-slip modeling. The vehicle is in this example modeled as a lumped mass and the resistance forces associated with the vehicle are modeled as different physical effects. The control system determines the shift point based on throttle position and vehicle speed when compared to the defined shift map. The driver model is based on a PI controller.

The model has 689 nontrivial equations and 15 state variables. There is a linear system of 77 simultaneous equations corresponding to the mass matrix inversion. After evaluating all parameter values and simplifying, the system reduces to 50 simultaneous equations. Symbolic manipulation reduces the size of the linear system of equations that has to be solved numerically to 7. The model was simulated with the explicit Euler method with a step size of 1 ms. As shown, the car follows the desired velocity very well.

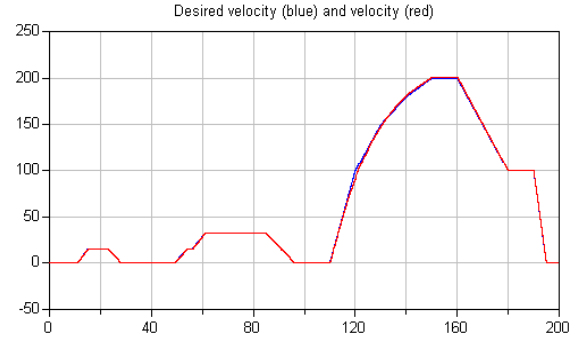

Figure 3: Desired velocity (blue) velocity (red)

The results are shown with a comparison to offline simulation using DASSL with a required relative tolerance of 10^{-6}. The difference is as shown below very small.

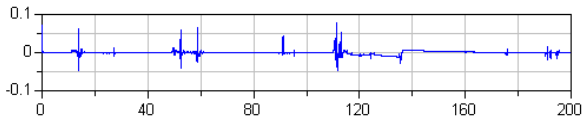

Figure 4: Velocity error (Explicit Euler – DASSL)

The gearshift is identical for explicit Euler and DASSL.

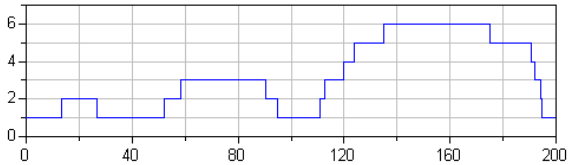

Figure 5: Gear shift

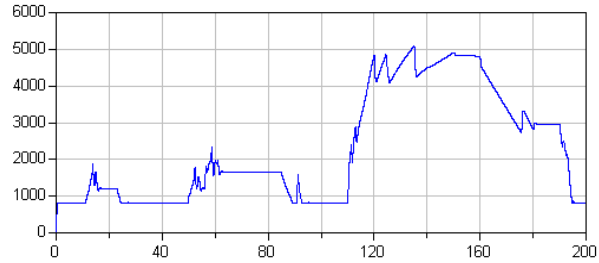

Figure 6: Engine speed

Also for engine speed, the agreement with the DASSL result is good.

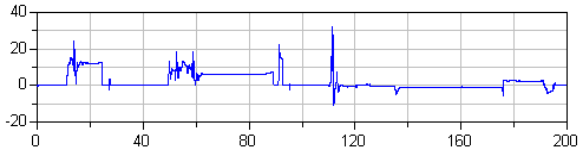

Figure 7: Engine speed error (explicit Euler – DASSL)

Real-time simulation

The benchmark model was run in the RT-LAB environment from OPAL-RT using a Pentium 4, 3066 MHz processorThe plot below shows the actual CPU time needed per step.

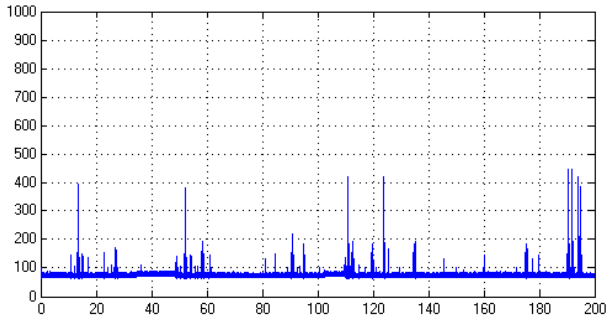

Figure 8: CPU time/step (microseconds)

The plot shows that the simulation runs in real time, because the time needed for each step is well below 1 ms. The CPU time needed per step is not constant, because of event handling due to locking or unlocking of clutches or brakes during gear shifting. Moreover, the linear system of size 7 being solved numerically has a coefficient matrix or a Jacobian, which does not depend on any continuous time variables, it changes only when there are discrete events. Its elements are in fact weighted sums of terms of the type

```
if axle.Break.locked then 1 else 0;
if transmission.wheelset_E.locked
   then 0 else 1;
```

Dymola exploits the fact that the Jacobian does not change during continuous time simulation. It

generates simulation code that only calculates the Jacobian and its LU-factorization during event iterations. This saves CPU time because the QR factorization is a major effort compared to the back substitution. The number of operations to factorize is proportional to the cube of the number of unknowns, i.e., $O(n^3)$, where n is the number of unknowns, which in this case is seven. Back substitution to calculate the solution when having the factorized Jacobian is much less computationally demanding.

To illustrate the importance of symbolic manipulation, a test was done where Dymola did not reduce the original system of 77 equations, but utilized that the Jacobian of the system only changed at discrete events. The plot below shows the actual CPU time needed per step.

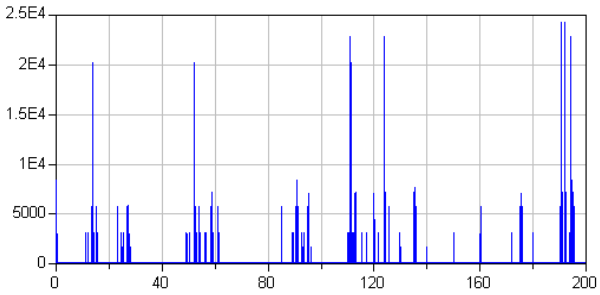

Figure 9: CPU time/step (microseconds) for the non-reduced case.

The plot shows that the CPU time needed per step varies a lot. This simulation does not run in real time. At certain steps the CPU time is nearly 25 ms. Much CPU time is needed, when there are discrete events and the Jacobian of the linear system with 77 unknowns needs to be calculated and LU-factorized. During continuous time simulation, the linear system is solved using the factorized Jacobian for back substitution, which is as shown a fast calculation.

3 Vehicle Dynamics Simulation

The free Modelica library VehicleDynamics [1] is used as basis for the evaluations in this report. This library is based on the multibody systems library ModelicaAdditions.MultiBody. The library is flexible since it is easy to replace wheel suspensions, tire models, etc. In particular, wheel suspensions are available with different levels of detail.

3.1 Special problems

Symbolic simplifications

Symbolic simplifications are very important for handling of multibody systems models. The model equations are written in the most general form. However, a motion could, for example, be constrained to be a rotation around a certain axis (e.g. {1,0,0}) in a local coordinate system. Parameters that are exactly zero are important to utilize symbolically; certain terms in the general model equations are cancelled and thus better efficiency can be achieved. The number of operations in the generated code is typically reduced by a factor of 3 to 10.

Mass matrix inversion

The differential-algebraic equations for a multibody system have a special structure. For a tree-structured mechanism, a large system of simultaneous equations involving accelerations, forces and torques will be present. It is important that such systems can be identified and reduced in size. It can typically be reduced in size to the number of degrees-of-freedom. This corresponds to finding the mass matrix of the mechanism.

Closed kinematic loops

Closed kinematic loops typically occur in suspensions with ideal joints. In such cases, the equations contain a nonlinear system of equations for each loop involving positions and orientations of the parts belonging to the loop. A linear system of equations involving velocities also appears. On acceleration level, equations from each loop appear in one large system of equations (corresponding to the mass matrix for tree-structured mechanisms accompanied with the constraint equations on acceleration level).

The non-linear system of equations is special in the sense that it involves trigonometric relations. It turns out that analytical solutions can be found [9]. The multibody library has been extended with composite joint models, for which the equations have been rewritten to give the analytical solution for a large class of kinematic loops occurring in vehicles and mechanisms.

Stiff models – Bushings

High fidelity models use bushing models instead of ideal joints. Such bushings are very stiff. This means that the differential equations are also stiff, i.e., that the corresponding linearized model has

eigenvalues in a large range. The explicit Euler method is not feasible for these models since a very small step size needs to be utilized (typically less than 50 microseconds). Implicit Euler allows a larger step size, but the accuracy is often not good enough. If neither the explicit nor the implicit Euler method is satisfactory, Dymola can utilize methods with higher order or mixed explicit/implicit methods for such models.

Tire models

The VehicleDynamics library [1, 2] contains two types of tire models: the standard tire model of Pacejka and the tire model of Rill. The Rill tire model is about 1 to 2 orders of magnitudes faster than the Pacejka tire model and should therefore be used when speed is important, such as for real-time simulation. The Rill tire model is based on the steady-state force/torque characteristics of a tire together with a simple transient tire deflection model.

3.2 Realtime Simulation Benchmarks

A mid-sized sedan with a front MacPherson suspension and a rear MultiLink suspension has been chosen as a benchmark model for vehicle dynamics simulations.

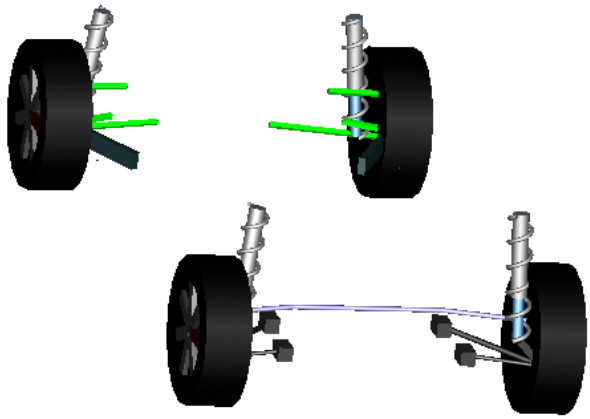

Figure 10: Front MacPherson suspension and rear MultiLink suspension.

The hierarchical structure of the vehicle models is shown in Figure 11.

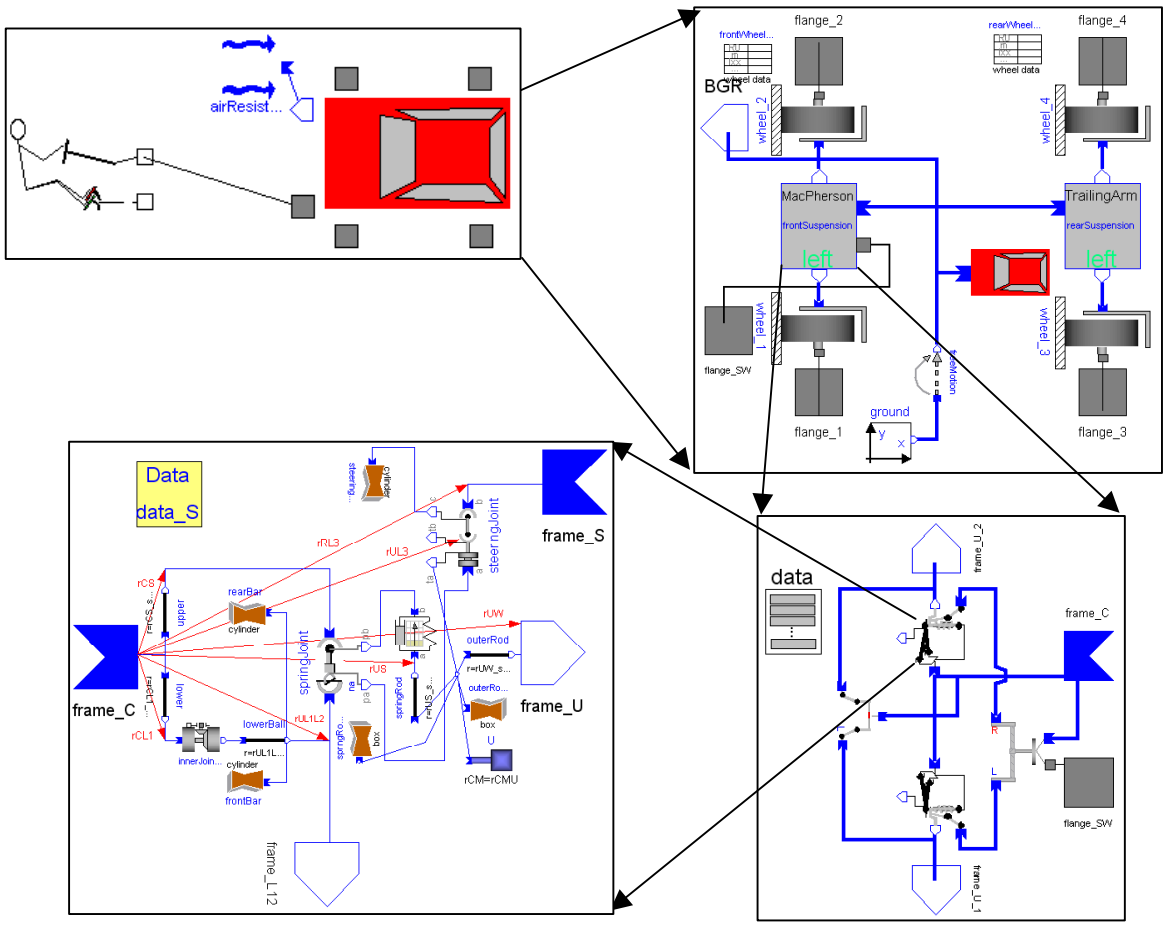

Figure 11: The hierarchical structure of the vehicle models

We have investigated models with different levels of detail.

1. Suspension is modeled by tables defining polynomials for Camber and toe-in angles. Steering is defined by an Ackermann function.
2. Suspension is modeled by linkages with ideal joints.
3. Suspension is modeled by linkages joined by bushings. The mass and inertia of the bar connecting two bushings are neglected.
4. Suspension is modeled by linkages joined by bushings where the small mass and inertia of the bar connecting two bushings are taken into account.

Level 1 – Linkage tables

The wheel suspensions are described by tables defining Camber and toe-in angles as functions of wheel bounce, i.e., a vertical motion of the wheel with constrained changes of the Camber and toe-in angles. This could easily be extended to handle also Camber and toe-in as functions of side force, which would make it possible to mimic the behaviour of suspensions with bushings and other flexible elements. This has been the common way to model vehicle dynamics in order to keep model complexity low for realtime simulation. Note, this method requires that the characteristics must either be measured, meaning that the suspension has to be built, or that the suspension characteristics have to be calculated from a more detailed model. This approach is justified if the simulation model is utilized, e.g., for improving controllers and ECUs for an existing vehicle. It is not useful, if the suspension and steering system shall be improved, e.g., based on optimization or parameter studies of a simulation model.

Steering is defined by an Ackermann function. The tables for Camber and toe-in angles are implemented as scaled polynomials. Dymola's symbolic engine differentiates these polynomials twice to handle the reduction of degrees-of-freedom.

The chassis has 6 degrees-of-freedom (DOFs), each wheel has 2 DOFs (bounce and rotation each) and the steering 1 DOF. The total DOF is 15. The tires each have 2 state variables for the deflection in x and y directions, i.e., 4*2 = 8 states. The total number of states for the vehicle dynamics itself is thus 2*15 + 8 = 38.

The steering in the benchmark model is a parameterized, given function which is filtered by a second order low pass filter to model driving behavior. The driver model of the benchmark model contains two additional state variables for the accelerator behavior. This is not used in this model since the vehicle maneuver is made with idle gear. The total number of state variables is thus 38 + 2 = 40.

Level 2 – Linkage with ideal joints

The table description used in level 1 is limited to only Camber and toe-in angles. It would of course be possible to extend to Castor angle trail as well as track width and wheel base translations. However, in many cases, in particular when trying new designs, it's easier to describe the suspension in terms of the linkage that is used.

The suspensions in level 2 consist of rigid mechanical components, i.e., all flexible elements, except for the struts, are replaced by ideal joints. Instead of a multi-link suspension, a trailing arm with similar geometry is used. The advantage over level 1 is that the suspension can be modelled with physical data and no precalculations or measurements are therefore needed.

The level 2 model uses a MacPherson type front wheel suspension, with the wishbone attached to the chassis via an ideal revolute joint (1 DOF). A strut is placed between the chassis and the wishbone via two spherical joints. The eigenrotation of the strut around its axis (1 DOF) is constrained by the distance constraint of an additional rod with two spherical joints on each end (1 constraint). One of the spherical joints of this rod is attached to the steering. In total, the suspension has therefore one degree of freedom, if the steering angle is given. The anti-roll bar is approximated by a spring/damper combination where the vertical force acting at its mount point on the lower part of the MacPherson strut is proportional to the relative vertical distance of the left and the right mount points. The rear suspension is a type of trailing arm with one DOF, the anti-roll bar is modeled like in the front suspension.

When using base elements of the MultiBody library to build up the MacPherson suspension, several non-linear algebraic loops appear. By using composite joint models (e.g., an aggregation of a revolute, a spherical and a universal joint) that contain analytic solutions of the non-linear kinematic relationships within the aggregation, the non-linear algebraic loops no longer occur in the generated code [9]. Note that this simplification is transparent to the end user.

The total DOF is 15 as for the level 1 model; The wheel bounce DOFs are replaced by the DOFs of the two trailing arm rotations and the two

wishbone rotations. The model has also 40 states. Note, that the elasticity of the tires in vertical direction has been modified slightly (both for the level 1 and the level 2 cars) in order to approximately compensate for the neglected bushings.

Level 3 – Linkage with bushings and massless bars

Using ideal joint models for the linkage is not accurate enough for severe driving conditions since bushings with certain flexibility are used in the real vehicle. Flexible elements are introduced in the suspensions of the level 3 model. The front suspension has bushings in the A-arm mounts. The rear multilink suspension has no ideal joints and the links are modelled as mass-less bars. If the mass and inertia of the rod connecting two bushings were not neglected 6 DOF would be added for every such pushrod. However, the mass and inertia are usually very small compared to the wheel and carrier masses, and therefore it is a good approximation to neglect the pushrod masses and inertias.

If the bushings were described solely by springs, then no states would be added, since springs in series connection lead to algebraic equations to solve for the spring deflections. Since bushings have a damping part, there are the states of the dampers (= 2*6). Once the states of one damper are known, the states of the other damper can be computed by relative kinematics. To summarise, a pushrod has 6 states, if the mass and inertia of the rod connecting the two bushings is neglected. There are 3 such bushing pairs at each rear wheel, i.e. the number of states is 2*3*6 = 36 states.

Additionally, the elasticity in the steering is taken into account by having a spring/damper system in the rack steering adding one additional DOF.

The total DOF is 36 and the model has 118 states.

Level 4 – Linkage with bushings and non-massless bars

A slightly more detailed model is obtained by not neglecting the masses of the push rods. The total DOF is 72 and the model has 2*72 + 8 + 2 = 154 states.

3.3 Simulation results

The benchmark models have been studied under a double lane change maneuver. The steering wheel has been operated as shown in Figure 12.

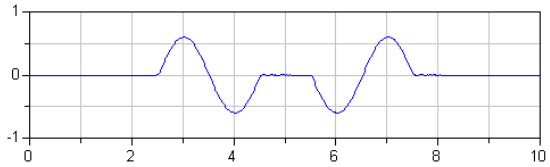

Figure 12: Steering wheel angle (rad)

We first show a comparison of the behavior of the four models. Below are shown plots of the side accelerations for the four cases.

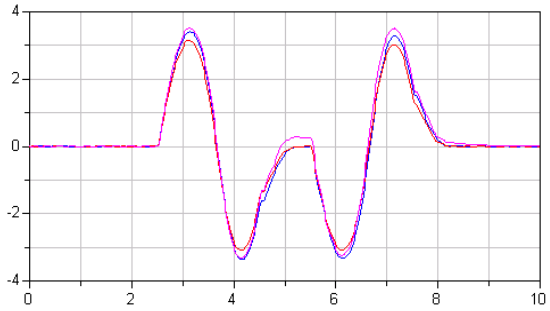

Figure 13: Side accelerations for level 1-4 models.

The level 3 and 4 models show a different behaviour than level 1 and 2. The differences can be spotted especially in the section between the lane changes: While the level 1 and 2 cars reach zero yaw and lateral acceleration, level 3 and 4 are too slow to get back to zero before the second lane change is started. This is essentially because of the elasticity in the suspensions. The level 1 and 2 models behave very similar. The tables used in level 1 were generated from suspensions close to those used in level 2. The behaviour of the level 3 and 4 models is practically identical. The oscillations of the links with small masses have very little effect on the deformation of the bushings that carry the wheel.

Real-time simulation

Let us discuss the problems of using these four models for real-time simulation.

It is possible to use explicit Euler with a step-size of 1 ms for the models of level 1 and 2. Comparisons with results from offline simulation with DASSL (relative tolerance=10^{-6}) show that the error in side acceleration is less than 0.25%. The major task when using the explicit Euler method is the calculation of the derivatives. Each of the level 1 model and the level 2 model has a

linear system of simultaneous equations corresponding to the mass matrix inversion. Dymola's symbolic processing reduces this system of equations to a system of about 10 equations. There are no nonlinear systems of equations, because the equations for the closed kinematics loops of level 2 have been solved analytically in the model library. The RT-LAB environment from OPAL-RT using a Pentium 4, 3066 MHz processor runs these two models easily in real-time, because it needs only 0.1 ms for an Euler step for the level 1 model and 0.3 ms for the level 2 model.

It is not possible to use explicit Euler to simulate the level 3 model or the level 4 model, because these models use bushing models instead of ideal joints. The bushings introduce very fast modes. Explicit Euler requires the step size to be smaller than the shortest time constant utilized (typically less than 50 microseconds). Typically, the fastest modes are not excited to a degree that it is necessay to resolve them for the intended purpose. In such cases the problem is referred as stiff. The *implicit* Euler method solves the numerical *stability* problem and allows larger step sizes to be used. It is the *accuracy* required that restricts how large step sizes can be used. Using the implicit Euler method, on the other hand, implies that a nonlinear system of equations needs to be solved at every step. The size of this system is at least as large as the size of the state vector, n. Solving large nonlinear systems of equations in real-time is somewhat problematic because the number of operations is $O(n^3)$ and the number of iterations might vary for different steps. Reducing the size of the nonlinear problem is advantageous. The method of inline integration [5, 6] was introduced to handle such cases. The discretization formulas of the integration method are combined with the model equations and structural analysis and computer algebra methods are applied on the augmented system of equations. Implicit Euler allows larger step size, but the accuracy is often not good enough. If neither the explicit nor the implicit Euler method is satisfactory, Dymola utilizes methods with higher order or mixed explicit/implicit methods for such models.

Each of the level 3 model and the level 4 model has a linear system of simultaneous equations corresponding to the mass matrix inversion. Dymola's symbolic processing reduces this system of equations to a system of about 20 equations.

The level 3 model and the level 4 model have been simulated with a special inline mixed explicit/implicit method, developed by Dynasim. This results in a nonlinear system of equations. For the level 3 model the size is about 130 and for the level 4 model the size is about 80. The level 4 model has 154 state variables. The large possible reduction of the size of the implicit non-linear system of equations from 154 to about 80 is due to the fact that certain subsystems are linear even after ammendment of the corresponding discretization formulas. Dymola automatically detects such structures during the structural analysis of the equations. The remaining nonlinear system of equations has to be solved by a Newton method; 2-3 iterations are typically needed, i.e. 3-4 residual calculations need to be performed. The step size was chosen to 2 ms. Comparisons with results from offline simulation with DASSL (relative tolerance=10^{-6}) show that the error in side acceleration is less than 0.5%.

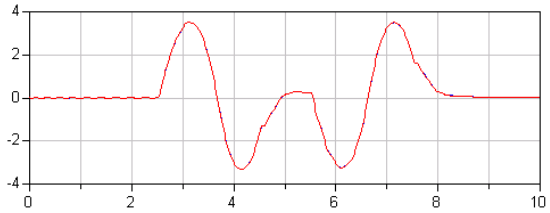

Figure 14: Side accelerations for the level 4 model

The difference between the results of the implicit method and DASSL is less than 0.5%

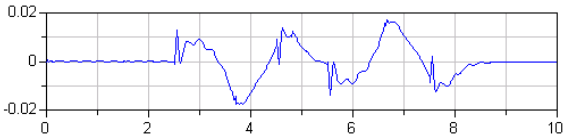

Figure 15: Side acceleration erros for the level 4 model (Euler – DASSL)

The realtime benchmarks were run on a computer equipped with a Pentium 4 processor running at 3066 MHz and a 333 MHz single-channel memory architecture.

As shown in Figure 16, the execution time is shorter for some time intervals, because of slower dynamics there requiring a smaller number of Newton iterations.

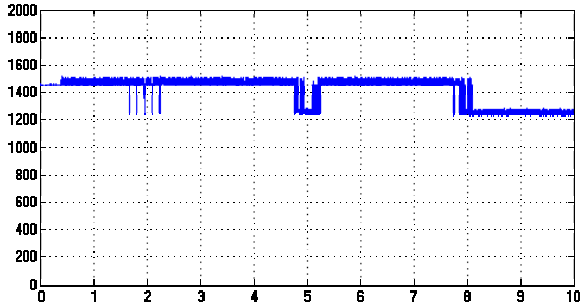

Figure 16: CPU time/step, when simulating level 4

It is worth noting that the level 4 model runs faster than the level 3 model, for which 1.7 ms per step is needed, although the level 4 model is more detailed. Obviously, the neglection of the push rod masses is not useful when Dymola's inline integration method together with its symbolic transformation capabilities are used. For offline simulations it is the opposite: the level 3 model runs faster as the level 4 model when using DASSL.

4 Conclusions

The paper has described typical efficiency issues in automotive real-time and HIL simulations. The examples given demonstrate the powerful real-time capabilities of Dymola and the Modelica modeling language. The models presented may indeed serve as benchmark examples as they are in the front-line of what can be simulated in real-time today. One of the benchmark models for vehicle dynamic simulation has 72 degrees-of-freedom with bushings in both the front and rear wheel suspensions. It was simulated in real-time with a sample rate of 500 Hz. The presented examples show that it is possible to simulate high-fidelity models in real-time for power trains and vehicle dynamics simulations. This is made possible by Dymola's unique and elaborate symbolic processing of the equations.

Acknowledgements

This work was in parts supported by *Bayerisches Staatsministerium für Wirtschaft, Verkehr und Technologie* under contract AZ300-3245.2-3/01 for the project *Test und Optimierung elektronischer Fahrzeug-Steuergeräte mit Hardware-in-the-Loop-Simulation.*

5 References

[1] Andreasson, J.: *VehicleDynamics library*, Proceedings of the 3rd International Modelica Conference, Modelica 2003, Modelica homepage: http://www.Modelica.org.

[2] Beckman, M. and J. Andreasson: *Wheel model library in Modelica for use in vehicle dynamics studies*, Proceedings of the 3rd International Modelica Conference, Modelica 2003, Modelica homepage: http://www.Modelica.org

[3] Brück, D., H. Elmqvist, S.E. Mattsson, H. Olsson: *Dymola for Multi-Engineering Modeling and Simulation*, Proceedings of Modelica 2002. Modelica homepage: http://www.Modelica.org.

[4] Dymola. *Dynamic Modeling Laboratory*, Dynasim AB, Lund, Sweden, http://www.Dynasim.se

[5] Elmqvist, H., F. Cellier, M. Otter: Inline Integration: *A new mixed symbolic/numeric approach for solving differential-algebraic equation systems*. Proceedings: European Simulation Multiconference. June 1995 Prague, pp: XXIII-XXXIV.

[6] H. Elmqvist, S.E. Mattsson, H. Olsson. *New Methods for Harware-in-the-loop Simulation of Stiff Models.* Proceedings of Modelica 2002. Modelica homepage: http://www.Modelica.org.

[7] Modelica, http://www.Modelica.org.

[8] OPAL-RT, http://www.opal-rt.com.

[9] Otter, M., H. Elmqvist, S.E. Mattssson: *The New MultiBody Library*. Proceedings of the 3rd International Modelica Conference, Modelica 2003. http://www.Modelica.org.

[10] PowerTrain Library 1.0 - Tutorial, German Aerospace Center (DLR), Oberpfaffenhofen, 2002, http://www.dynasim.se/www/PowerTrainTutorial.pdf/