2004-01-0768

Realtime Simulation of Detailed Vehicle and Powertrain Dynamics

Hilding Elmqvist, Sven Erik Mattsson, Hans Olsson
Dynasim AB, Sweden

Johan Andreasson
KTH Vehicle Dynamics, Sweden

Martin Otter, Christian Schweiger
German Aerospace Center (DLR), Germany

Dag Brück
Dynasim AB, Sweden

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 1 kHz 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.

INTRODUCTION

Simulation has become common practice in automotive developments because it speeds up the development cycle and decreases costs.

A typical application is evaluation, fine-tuning and testing of Electronic Control Units (ECU). For example, fine-tuning of the control of the gear shifting is essential to optimize comfort. An ECU is often a proprietary component and the vendor does not provide the control algorithms of the ECU. Hardware-in-the-loop simulation, HILS, must be used, where the setup consists of the ECU hardware and a real-time simulation of all other components interacting dynamically with it. This requires very efficient simulation. For example, a typical ECU gearbox for gearshift control has a sampling time of 10 ms implying that the simulation needs to produce values each 1 ms or faster.

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. Several benchmark examples and simulation results demonstrate the effectiveness.

The methods are implemented in the simulation environment Dymola [3, 5] that uses the Modelica [10] modeling language for describing the models. This paper describes how Dymola solves certain difficult problems in HILS 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 engaging 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 the generated code is downloaded to the target HILS system.

The task of the suspension is to isolate the vehicle from unwanted road-induced vibrations and at the same time make sure that the wheel-road contact is optimal. To assure good comfort, elastic elements, bushings, are introduced along the path of the vibration energy flow from ground to vehicle. However, the bushings also have the drawback that they allow the wheel to move slightly and thus decrease the driver's control of the vehicle. For racecars this would not be acceptable and thus, no bushings are used in these cases. For normal cars and trucks, it is instead important to make sure that the position and stiffness of the bushings are such that they together affect the wheel in a proper way.

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. Simulations reported in the paper show that a model without and with bushings give visibly different results in side accelerations when making a double lane shift maneuver. Neglecting the effects of bushings when designing a suspension may thus result in unwanted behavior of the vehicle in critical situations.

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 1 kHz.

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 [11], demonstrating real-time performance of complex models.

SPECIAL PROBLEMS IN AUTOMOTIVE HILS

This section gives an overview and explanation of some special difficulties in HILS in automotive applications. The underlying technology to solve these problems and especially the symbolic processing that dymola performs are described in the appendix.

TRANSMISSION MODELS

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 [15, 16, 18] library available from Dynasim) used e.g. to model a ZF 6 HP 26. The model includes planetary and Ravigneaux gear sets, clutches, brakes and inertias.

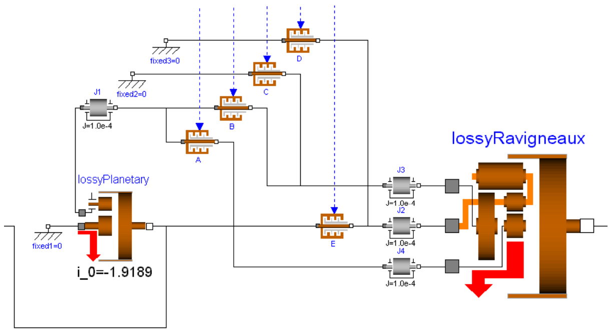

Figure 1. Gearbox model.

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.

<u>Fast sampling</u> - The differential equations of the gearbox needs 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.

<u>Accuracy and discontinuities</u> - 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.

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. In off-line simulation one can interpolate in the last steps in order to accurately find the event one just passed. Such interpolation would cause severe over-runs for HILS. Dymola therefore predicts the event point. The prediction of the event must be done with care to accurately find the event. Without event detection, it would be necessary to introduce non-physical effects that have to be tuned in a step-size dependent way.

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 predicted 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 HILS simulations for ECU testing.

<u>Event propagation</u> - 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 torque 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 symbolically. 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 systems of equations - 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.

CHASSIS MODELS

A model library called VehicleDynamics [1] has been developed for chassis modeling. A beta release is used for the evaluations in this paper. It is based on the Modelica multibody systems library. Both libraries are available with a standard Dymola distribution. The library is flexible since it is easy to replace wheel suspensions, etc. In particular, wheel suspensions are available with different levels of detail.

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 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 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. 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.

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 very small step size needs to be utilized (typically less than 50 microseconds). Implicit Euler allows a larger step size, but the accuracy is sometimes not good enough. If neither the explicit nor the implicit Euler method is satisfactory, Dymola utilizes methods with higher accuracy for such models.

Tire models - The VehicleDynamics library contains two types of tire models [2]: the standard tire model of Pacejka [13] and the tire model of Rill [17]. 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 model of transient tire deflection.

REAL-TIME SIMULATION BENCHMARKS

TRANSMISSION MODELS

Components for modeling of transmissions are available in the PowerTrain library. As a benchmark example, we will consider modeling of a 6 speed gearbox (Lepelletier wheelset, 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 is shown in Figure 2.

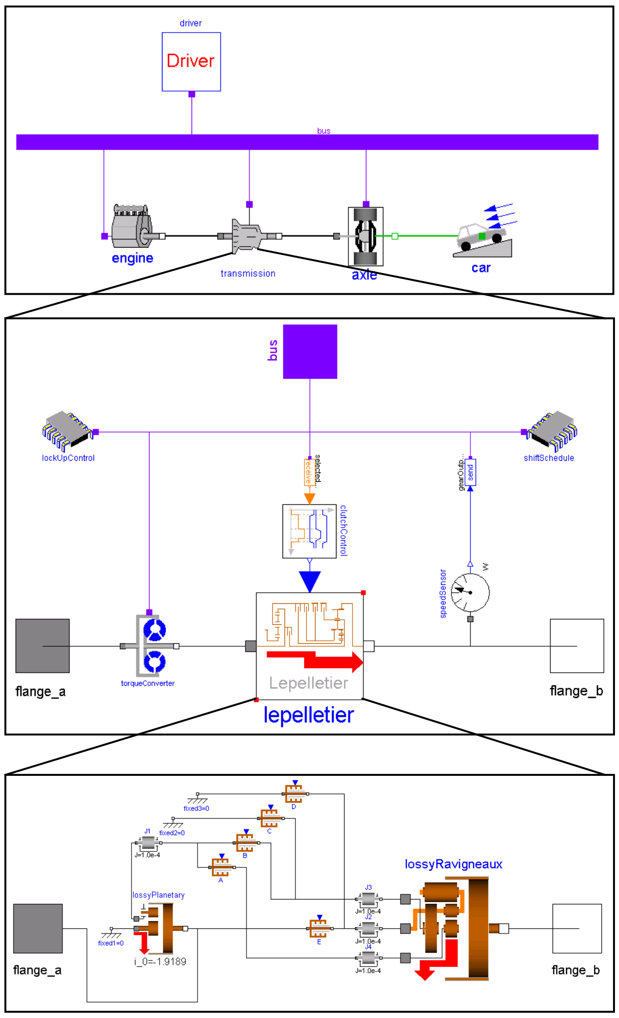

Figure 2. The transmission example with the gearbox model.

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. The driveline model is essentially a rigid model with no compliance in the drive shafts and no tire-slip modeling. The vehicle is 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 around 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 system 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.

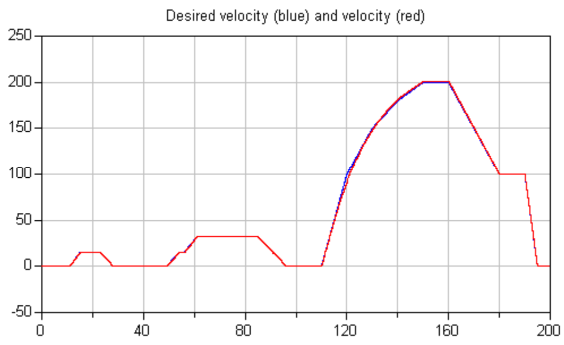

Figure 3. Desired velocity and velocity.

The results are shown with a comparison to offline simulation using DASSL requiring a relative tolerance of 10^{-6}. The difference between explicit Euler and the DASSL result is as shown below very small.

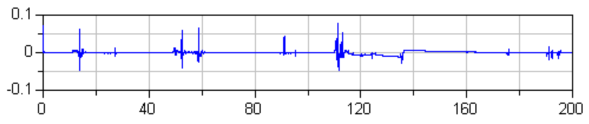

Figure 4. Velocity error (Explicit Euler – DASSL)

The gearshift is identical for explicit Euler and DASSL.

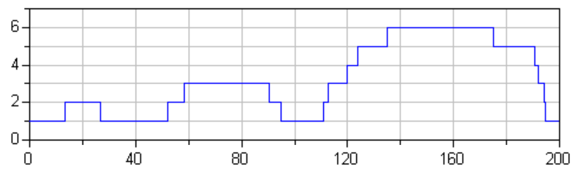

Figure 5. Gear shift.

The engine speed varies as

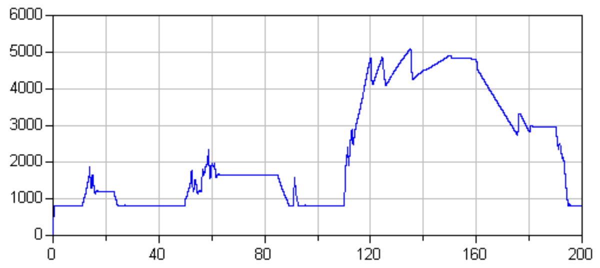

Figure 6. Engine speed.

The agreement with DASSL result is good.

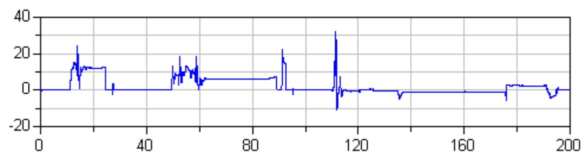

Figure 7. Engine speed error (explicit Euler – DASSL).

Dymola generates automatically a 3 D representation for animation as shown in the picture below.

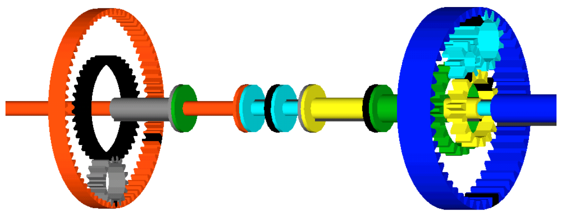

Figure 8. Animation of the gearbox.

Real-time simulation - The benchmark model was run in the RT-LAB environment from OPAL-RT using a Pentium 4, 3066 MHz processor.

The plot below shows the actual CPU time needed per step.

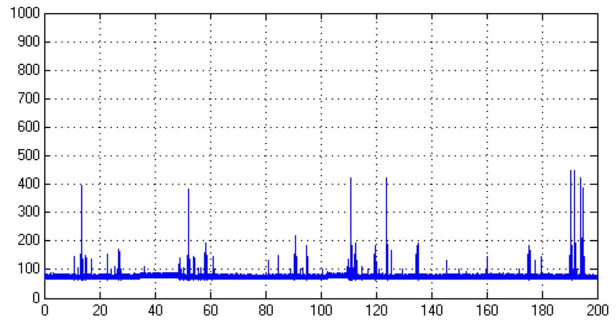

Figure 9. CPU time/step (microseconds)

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 to 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 manipulate the original system of 77 equations. However, it realized that the Jacobian of the system only changed to discrete events. The plot below shows the actual CPU time needed per step.

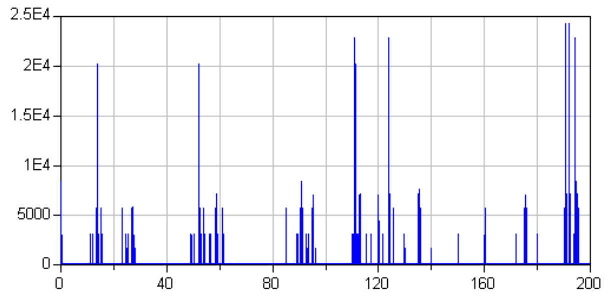

Figure 10. CPU time/step [microseconds] for the non-reduced case.

The plot shows that 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.

CHASSIS MODELS

As a benchmark model for vehicle dynamics simulation, we have chosen a mid-sized sedan with a front MacPherson suspension and a rear MultiLink suspension.

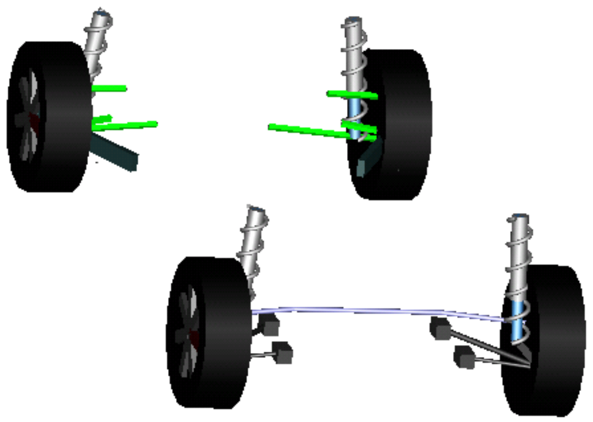

Figure 11. Car suspensions used in benchmark model.

A model library called VehicleDynamics [1] has been developed for chassis modeling. The library is flexible since it is easy to replace wheel suspensions, etc. In particular, wheel suspensions are available with different levels of detail. The hierarchical structure of the vehicle models used in this benchmark is shown in Figure 12.

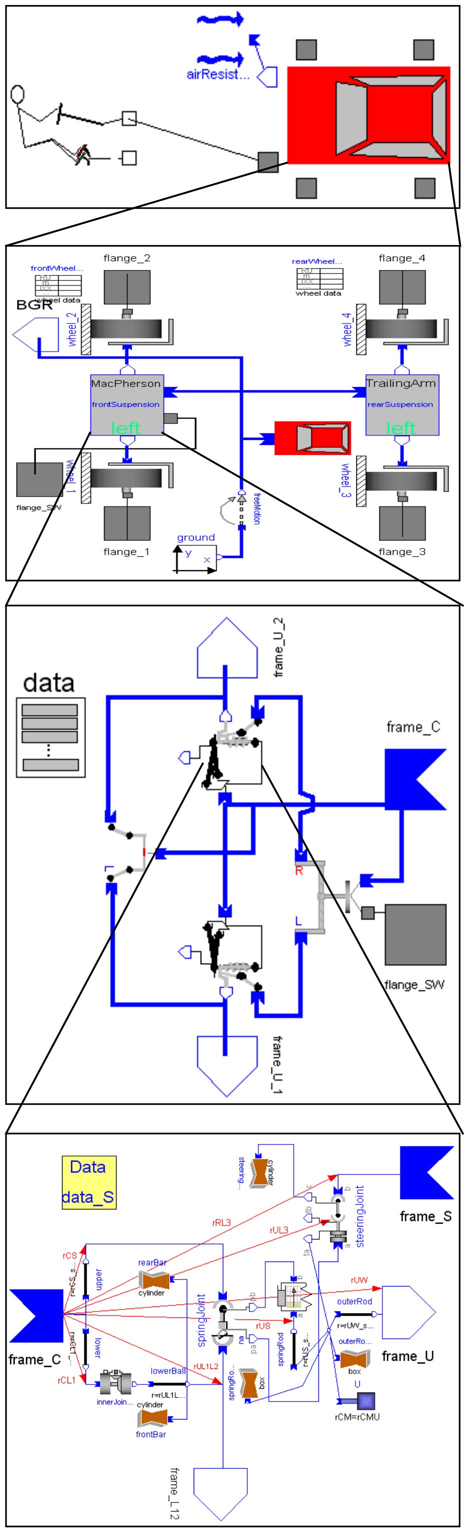

Figure 12. The hierarchical structure of the vehicle models.

We have investigated models with different levels of detail.

1. The suspension is modeled by tables defining polynomials for Camber and toe-in angles. Steering is defined by an Ackermann function (model VehicleDynamics.Examples.SedanCar_Level1_LinkageTables).
2. The suspension is modeled by linkages with ideal joints (model VehicleDynamics.Examples.SedanCar_Level2_RigidLinkages).
3. The suspension is modeled by linkages joined by bushings. The mass and inertia of the bar connecting two bushings are neglected (model VehicleDynamics.Examples.SedanCar_Level3_Bushings).
4. The suspension is modeled by linkages joined by bushings where the small mass and inertia of the bar connecting two bushings is taken into account (model VehicleDynamics.Examples.SedanCar_Level4_MassRods).

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 behavior of suspensions with bushings and other flexible elements. However, the drawback with this method is 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.

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 complexity of the model is characterized in the following table showing the number of degrees-of-freedom (DOFs).

	Individual DOFs	Component counts	Total DOFs
Chassis	6	1	6
Wheel bounce	1	4	4
Wheel rotation	1	4	4
Steering	1	1	1
Total			**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 2*15 + 8 = 38.

The steering in the benchmark model is a 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 Caster 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 modeled 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. Note that this simplification is transparent to the end user. Details are given in [12].

	Individual DOFs	Component count	Total DOFs
Chassis	6	1	6
Wishbone rotation	1	2	2
Steering	1	1	1
Trailing arm rotation	1	2	2
Wheel rotation	1	4	4
Total			**15**

The table above gives the degrees of freedom of this vehicle model. Similar to the level 1 car, there are 4*2 = 8 states for the tire deflections and 2 unused states. As a result, this 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 number of degrees of freedom is shown below.

	Individual DOFs	Component count	Total DOFs
Chassis	6	1	6
Front wheel carrier	6	2	12
Steering	1	1	1
Rack steering	1	1	1
Rear wheel carrier	6	2	12
Wheel rotation	1	4	4
Total			**36**

There are 4*2 = 8 states for the tire deflections and 2 unused states. As a result, this model has 2*36 + 36 + 8 + 2 = 118 states.

Level 4 – Linkage with bushings and non-massless bars - A slightly more detailed model is obtained by including the masses of the push rods.

	Individual DOFs	Component counts	Total DOFs
Chassis	6	1	6
Front wheel carrier	6	2	12
Steering	1	1	1
Rack steering	1	1	1
Rear wheel carrier	6	2	12
Pushrods of rear wheel carrier	6	2*3	36
Wheel rotation	1	4	4
Total			**72**

The number of degrees of freedom is shown in the table above and this model has 2*72 + 8 + 2 = 154 states.

Simulation results - The benchmark models have been studied under a double lane change maneuver. Figure 13 shows how the steering wheel was operated.

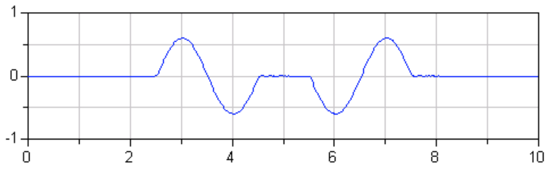

Figure 13. Steering wheel angle [rad].

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

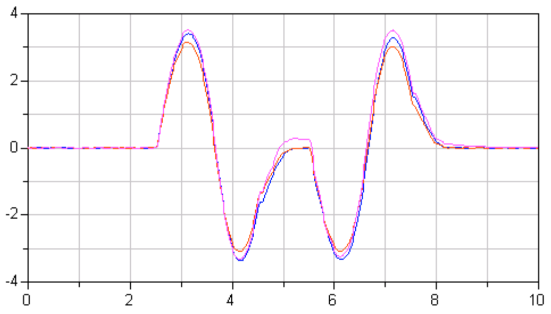

Figure 14. Side accelerations for level 1-4 models.

The level 3 and 4 models show a different behavior 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 behavior 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.

Dymola generates automatically a 3 D representation for animation as shown in Figure 15 from the information included in the model components.

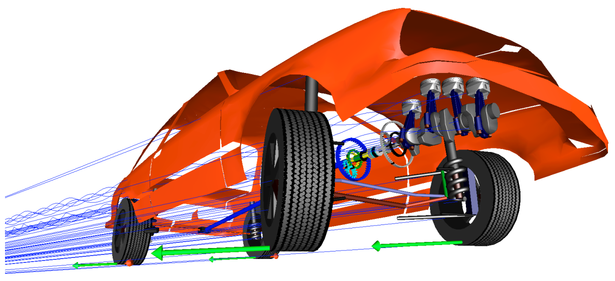

Figure 15. Animation of the car.

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.

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.

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 leading to very stiff systems. 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 necessary 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. As described previously Dymola uses inline integration and symbolic manipulation to make the calculations efficient.

The level 3 model and the level 4 model have been simulated using the inlined implicit Euler 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 90. 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 90 is due to the fact that certain subsystems are linear even after amendment 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 1 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%.

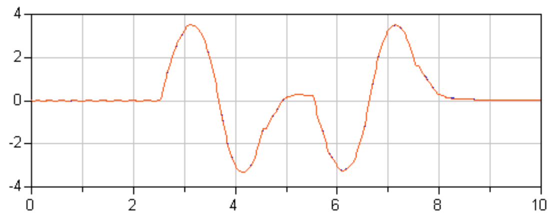
Figure 16. Side accelerations for the level 4 model.

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

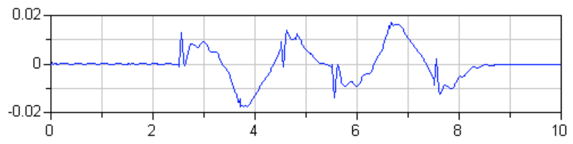
Figure 17. Side acceleration errors for the level 4 model (Euler – DASSL)

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

As shown in Figure 18, the execution time is shorter for some time intervals, because of slower dynamics there requiring a smaller number of Newton iterations.

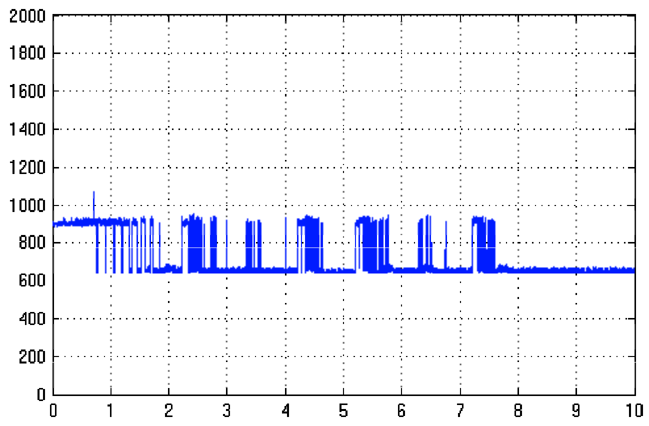
Figure 18. CPU time/step [microseconds], when simulating level 4.

When time is around 0.8, there is one overrun, but this is caught up in the next step. It is possible to use a step size of 2 ms and still get accurate simulation results.

It is worth noting that the level 4 model runs faster than the level 3 model, for which 1.3 ms per step is needed, although the level 4 model is more detailed. Obviously, neglecting 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 than the level 4 model when using DASSL.

INTEGRATED CHASSIS AND POWERTRAIN MODEL

To demonstrate the object-oriented methodology and the possibilities for collaboration, the independently developed models were put together to form a complete vehicle with chassis and powertrain. This is easily achieved by just connecting the models as shown in the figure below.

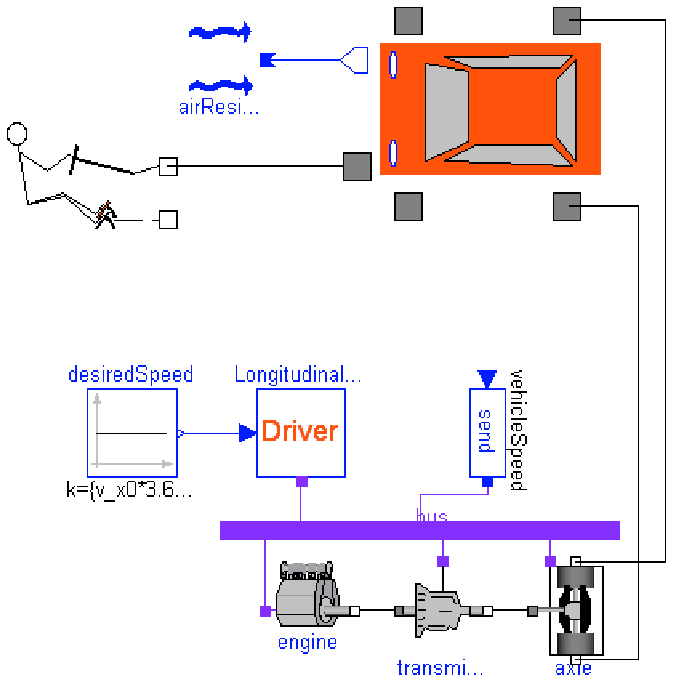

Figure 19. Integrated chassis and powertrain model.

The level 2 model together with the powertrain model has 3054 nontrivial equations and 53 state variables. There is a linear system of 490 simultaneous equations corresponding to the mass matrix inversion. These equations are reduced by Dymola to a system with a 18 x 18 matrix equation.

The model has been simulated with inline explicit Euler method since there is no stiffness in the model. The step size was chosen to 1 ms. The result is shown below. For comparison the results from the previous simulation without powertrain is included.

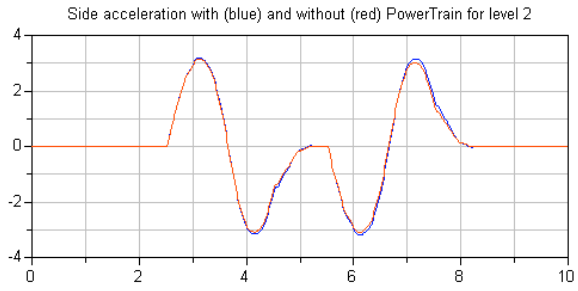

Figure 19. Side acceleration.

In this case, the driver model tries to keep the velocity constant during the double lane shift maneuver as shown below. In the case without powertrain, the velocity was decreased.

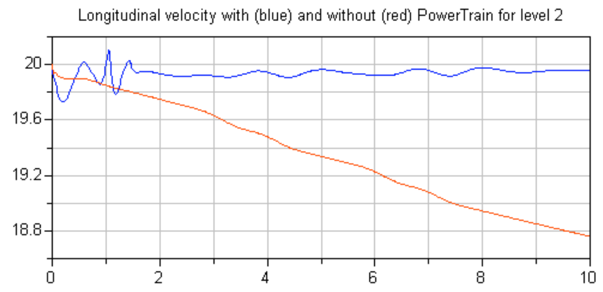

Figure 20. Longitudinal velocity.

Real-time simulation - When the model was run on the RT-LAB platform, the needed CPU time was less than 1 ms. Thus the model runs in real-time.

CONCLUSION

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 1 kHz. 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.

ACKNOWLEDGMENTS

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*.

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. Duff, I.S, A.M. Erisman, and J.K. Reid: *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, 1986.
5. Dymola. *Dynamic Modeling Laboratory*, Dynasim AB, Lund, Sweden, http://www.Dynasim.se
6. 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.
7. H. Elmqvist, S.E. Mattsson, H. Olsson. *New Methods for Hardware-in-the-loop Simulation of Stiff Models*. Proceedings of Modelica 2002. Modelica homepage: http://www.Modelica.org.
8. Hairer, Wanner: *Solving Ordinary Differential Equations II*, Springer Verlag
9. Mattsson, S.E. and G. Söderlind: *Index reduction in differential-algebraic equations using dummy derivatives*. SIAM Journal of Scientific and Statistical Computing, Vol. 14 pp. 677-692, 1993.
10. Modelica, http://www.Modelica.org.
11. OPAL-RT, http://www.opal-rt.com.
12. Otter, M., H. Elmqvist, S.E. Mattssson: *The New MultiBody Library*. Proceedings of the 3rd International Modelica Conference, Modelica 2003. http://www.Modelica.org.
13. Pacejka, H.B.: *Tyre and vehicle dynamics*. Butterworth Heinemann, 2002.
14. Pantelides C.: *The consistent initialization of differential-algebraic systems*. SIAM Journal of Scientific and Statistical Computing, pp. 213-231, 1988.
15. Pelchen, C., C. Schweiger, M. Otter. *Modeling and Simulating the Efficiency of Gearboxes and of Planetary Gearboxes.* Proceedings of Modelica 2002. Modelica homepage: http://www.Modelica.org/.
16. PowerTrain Library 1.0 - Tutorial, German Aerospace Center (DLR), Oberpfaffenhofen, 2002, http://www.dynasim.se/www/PowerTrainTutorial.pdf/
17. Rill, G.: *Simulation von Kraftfahrzeugen*. Vieweg, 1994
18. Schweiger, C. and M. Otter. *Modelling 3D Mechanical Effects of 1D Powertrains.* Proceedings of Modelica 2003. Modelica homepage: http://www.Modelica.org/.
19. Tarjan, R.E. *Depth first search and linear graph algorithms*, SIAM J. Comput. 1, 146-160 (1972).

CONTACT

The first author, Dr Hilding Elmqvist, can be contacted by mail to Dynasim AB, Research Park Ideon, SE 224 70 Lund, Sweden or by e-mail to Elmqvist@Dynasim.se.

APPENDIX - DYMOLA TECHNOLOGY FOR HILS

The Modelica Language Specification defines how a Modelica model shall be mapped into a mathematical description as a mixed system of Boolean equations, differential-algebraic equations (DAE) and discrete equations. There are no general-purpose solvers for such problems. There are numerical DAE solvers, which could be used to solve the continuous part. However, if a DAE solver is used directly to solve the original model equations, the simulation will be very slow. The traditional approach has been to manually manipulate the model equations to ODE form. Dymola automates all this time-consuming and error-prone work and generates efficient code also for large and complex models. Symbolic processing is a unique feature of Dymola to make simulations efficient. Dymola converts the differential-algebraic system of equations symbolically to state-space forms, i.e. solves for the derivatives. Efficient graph-theoretical algorithms are used to determine which variables to solve for in each equation and to find minimal systems of equations using tearing to be solved simultaneously (algebraic loops). The equations are then, if possible, solved symbolically or code for efficient numeric solution is generated. Discontinuous equations are properly handled by translation to discrete events as required by numerical integration routines.

SORTING AND ALGEBRAIC LOOPS

For example in ACSL and Simulink, there are assignment statements for each variable and these are sorted in a computational order that a variable is calculated before being used. A Modelica model defines behavior in terms of genuine equations and Dymola has to assign an equation for each variable as a part of the sorting procedure, which also identifies algebraic loops.

To be able to process problems with hundred thousand unknowns, the idea is to focus on the structural properties, i.e. which variables that appear in each equation rather than how they appear. This information can be represented by a "structure" Jacobian, where for a system of equations, $h(x) = 0$, each element i, j, is zero if x_j does not appear in the expression h_i, otherwise it is one. The sorting procedure is to order unknowns and equations to make the structure Jacobian become Block Lower Triangular, BLT. A BLT partitioning reveals the structure of a problem. It decomposes a problem into subproblems, which can be solved in sequence. There are efficient algorithms, see e.g. [4], for constructing BLT partitions with diagonal blocks of minimum size. Each non-scalar block constitutes an algebraic loop. This sorting procedure identifies all algebraic loops in their minimal form that is unique. The sorting procedure is done in two steps. The first step is to assign each variable, x_j, to a unique equation, $h_i = 0$ such that x_j appears in this equation. It can be viewed as permuting the equations to make all diagonal elements of the structure Jacobian non-zero. If it is impossible to pair variables and equations in this way then the problem is structurally singular. The second step of the BLT partition procedure is to order pairs of a variable and an equation to make the structural matrix BLT. The basic algorithm was given by Tarjan [19].

REDUCTION OF SIZE AND COMPLEXITY

A Modelica model has typically many simple equations, $v_1 = v_2$ being the result of connections. These are easy to exploit for elimination.

From the BLT partition it is rather straightforward to find unknowns that actually are constant and can be calculated and substituted at translation. This may have considerable impact on the complexity of the problem that has to be solved numerically. For example, a multibody component is developed for free motion in a 3-dimensional space. When using it we connect it to other components and set parameters implying restrictions on its motion. For example, it may be restricted to move in a plane. It means that coefficients in the equations become zero and terms disappears. This in turn may make algebraic loops to decompose into smaller loops or even disappear.

Algebraic loops are, if possible, solved symbolically or code for efficient solution is generated. In order to obtain efficient simulation, it is very important to reduce the size of the problem sent to a numerical solver. The work to solve a system of equations increases rapidly with the number of unknowns, because the number of operations is proportional to the cube of n, i.e. $O(n^3)$, where n is the number of unknowns. One approach to reduce size the size is called tearing. Let z represent the unknowns to be solved from the system of equations. Let z be partitioned as z_1 and z_2 such that

$$\begin{aligned} Lz_1 &= f_1(z_2) \\ 0 &= f_2(z_1, z_2) \end{aligned}$$

where L is lower triangular with constant, non-zero diagonal elements. A numerical solver needs then only consider z_2 as unknown. A numerical solver provides guesses for z_2 and would like to have the f_2 residuals calculated for these guesses. When having a value for z_2, it is simple to calculate z_1 from the first set of equations. Note that it is very important to avoid divisions by zero and the assumption that the diagonal elements are constant and non-zero guarantees this. It is then straightforward to calculate the f_2 residuals. The z_1 variables are in fact hidden for the numerical solver. The general idea of tearing is to decompose a problem into two sets, where it is easy to solve for the first set when the solution to the second set is known and to iterate over the second set. The aim is of course to make the number of components of z_2 as small as possible. It is a hard (NP-complete) problem to find the minimum. However, there are fast heuristic approaches to find good partitions of z. If the equations are linear, they can be written as

$$\begin{aligned} Lz_1 &= Az_2 + b_1 \\ 0 &= Bz_1 + Cz_2 + b_2 \end{aligned}$$

and it is possible to eliminate z_1 to get $Jz_2 = b$, where

$$J = C + BL^{-1}A$$
$$b = b_2 + BL^{-1}b_1$$

This may be interpreted as Gauss elimination of z_1. The procedure may be iterated.

When solving a linear equation system, a major effort is to calculate and LU or QR factorize the Jacobian, *J*. Back substitutions are much less computationally demanding. In some cases the elements of the Jacobian does not vary continuously with time. The Jacobian may for example only change at events and it is then only necessary to calculate and factorize it during event iterations and not during continuous simulation. In other cases, it may depend only on parameters and constants and then it needs only to be calculated once, at the start of a simulation.

When using Newton methods for non-linear equation systems, it is necessary to calculate the Jacobian. If this is made numerically from residuals, then n residual calculations are needed. Dymola provides analytic Jacobians. These are more accurate and much less computationally demanding, because there are many common subexpressions to exploit. Modelica provides facilities to provide derivatives also for external functions.

INDEX REDUCTION

When solving an ordinary differential equation (ODE) the problem is to integrate, i.e. to calculate the states when the derivatives are given. Solving a DAE may also include differentiation, i.e. to calculate the derivatives of given variables. Such a DAE is said to have high index. It means that the overall number of states of model is less than the sum of the states of the components. Higher index DAEs are typically obtained because of constraints between models. To support reuse, model components are developed to be "general". Their behavior is restricted when they are used to build a model and connected to other components. Take as a very simple example two rotating bodies with inertia J_1 and J_2 connected rigidly to each other. The angles and the velocities of the two bodies should be equal. Not all four differentiated variables can be state variables with their own independent start values. The connection equation for the angles, $\varphi_1 = \varphi_2$, must be differentiated twice to get a relation for the accelerations to allow calculation of the reaction torque.

The reliability of a direct numerical solution is related to the number of differentiations needed to transform the system algebraically into ODE form. Today's numerical integration algorithms for DAEs, such as used by most simulators, can handle systems where equations needed to be at most differentiated once. However, reliable direct numerical solutions for non-linear systems are not known if two or more differentiations are required. Furthermore, if mixed continuous and discrete systems are solved, the hybrid DAE must be initialized at every event instants. In this case, it is in general not sufficient to just fulfill the original DAE. Instead, also some differentiated equations have to be fulfilled, in order that a consistent initialization is fulfilled. Direct numerical methods have problems at events to determine consistent restart conditions of higher index systems.

Higher index DAEs can be avoided by restricting how components may be connected together and/or include manually differentiated equations in the components for the most common connection structures. The drawback is (1) physically meaningful component connections may no longer be allowed in the model or (2) unnecessary "stiff" elements have to be introduced in order that a connection becomes possible. For example, if a stiff spring is introduced between the two rotating bodies discussed above, the problem has no longer a higher index.

Since most Modelica libraries are designed in a truly object-oriented way, i.e. every meaningful physical connection can also be performed with the corresponding Modelica components, this leads often to higher index systems, especially in the mechanical and thermo-fluid field.

Dymola transforms higher index problems by differentiating equations analytically. The standard algorithm by Pantelides [14] is used to determine how many times each equation has to be differentiated. Selection of which variables to use as state variables is done statically during translation or in more complicated cases during simulation with the dummy derivative method [9]. Let us make the example above a bit more realistic and put a gearbox with fixed gear ratio n between the two bodies. Dymola differentiates the position constraint twice to calculate the reaction torque in the coupling, and it is sufficient to select the angle and velocity of either body as state variables. The constraint leads to a linear system of simultaneous equations involving angular accelerations and torques. The symbolic solution contains a determinant of the form "$J_1 + n^2J_2$". Dymola thus automatically deduces how inertia is transformed through a gearbox.

INLINE INTEGRATION

Real-time simulation of physical models is a growing field of applications for simulation software. One goal is to be able to simulate more and more complex models in real-time with fast sampling rates. Many of those models are multi-engineering models, which means, that they contain components from more than one engineering domain. Mechanic, electric, hydraulic or thermodynamic components are often coupled together in one model. This leads to a large span of time-constants in the model. The usual use of the explicit Euler method is not appropriate because the fastest time-constant determines the computational effort (step size) for the simulation. In order to maintain stability of the integration method, the step size must be less than the smallest time constant. 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 to 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 that 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 [6, 7] 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. For a robotics model with 66 states, the size of the nonlinear system of equations could be reduced to only 6. The large possible reduction of the size of the implicit non-linear system of equations is due to the fact that certain subsystems might be linear even after ammendment of the corresponding discretization formulas. Dymola is able to automatically detect such structures during the structural analysis of the equations.

HIGHER ORDER METHODS

In order to get sufficient accuracy for large steps it was necessary to extend the basic method to higher order methods. Higher order methods indicate that they have order greater than one, and the ones considered have orders 2 to 4.

The higher order methods implemented for the new method are L-stable singly diagonally implicit Runge-Kutta methods [8]. The L-stability implies that they are stable for all stable linear systems and do not exhibit oscillations for very stiff systems. The class of methods, singly diagonally implicit Runge-Kutta methods, require the solution to the same equation systems as implicit Euler.