Simulation Algorithms in Vehicle System Dynamics

MARTIN ARNOLD

SUMMARY

Multi-body dynamics may be considered as integration platform for simulation in vehicle system dynamics. In the present report we discuss classical numerical simulation techniques of multi-body dynamics, their use in multi-body system simulation packages and extensions to typical problems of vehicle system dynamics like continuous and discrete controllers and multi-physical applications.

1. INTRODUCTION

Model based simulation in vehicle system dynamics relies on advanced methods for model setup, robust and efficient numerical solution techniques and powerful simulation tools for industrial applications.

Most frequently the mechanical components are described by rigid or flexible multi-body system models that interact with electrical, hydraulic and other system components [1, 2], see also [3] for a summary of recent developments and [4] for a comprehensive overview on methods and algorithms that are tailored to industrial simulation packages. The first systematic treatment of numerical problems and numerical methods in multi-body dynamics from the viewpoint of vehicle system dynamics is the monograph of Eich–Soellner and Führer [5], see also [6].

In its simplest form the equations of motion of a multi-body system are given by a non-linear second order system of ordinary differential equations (ODEs) of moderate dimension that may be solved numerically by standard methods.

The structure of the model equations gets more complicated if the multi-body system has kinematically *closed loops* [7]. Multi-body system models of wheel suspensions may be considered as a typical example in road vehicle simulation. The complex kinematics is modelled by *constraints*, the resulting model equations form a second order differential-algebraic equation (DAE).

Robust and efficient DAE time integration methods are the backbone of state-of-the-art multi-body system simulation software. Another important issue in simulation is the handling of *discrete* controllers that are beyond the classical time continuous

Address correspondance to: Martin Arnold, Martin–Luther–University Halle–Wittenberg, Department of Mathematics and Computer Science, Institute of Numerical Mathematics, 06099 Halle (Saale), Germany.

world of mechanics.

Advanced multi-body system simulation packages have been open to multi-disciplinary applications for a long time. The classical approach to the integration of hydraulic and electrical system components in multi-body models are special force elements with time continuous or time discrete inner state variables [3].

Alternatively, novel modelling and simulation techniques for multi-physical technical systems may be used in more complex multi-disciplinary applications. Here, the multi-disciplinary aspect is considered by universal modelling languages like Modelica [8] or by the coupling of several mono-disciplinary simulation packages in a co-simulation environment [9].

In the present report we give an overview on classical and more advanced simulation algorithms in vehicle system dynamics that are closely related to multi-body numerics and multi-body system simulation packages.

The material is organized as follows: Basic simulation algorithms in multi-body system analysis are summarized in Section 2. In Section 3 we discuss the general structure of model equations including closed-loop systems, inner state variables of force elements and time discrete sub-systems. Special DAE time integration methods for the robust and efficient numerical solution of these model equations are presented in Section 4. Finally, in Section 5 we focus on algorithms and tools for multi-disciplinary applications.

Throughout the present report the analysis of distributed physical phenomena like the elastic deformation of car components or the temperature field in disk brakes is restricted to low-dimensional modal approximations.

2. BASIC SIMULATION ALGORITHMS

If a mechanical multi-body system is described by a minimum set of generalized coordinates $\boldsymbol{y}(t) \in \mathbb{R}^{n_y}$ the equations of motion form a second order ODE

$$\boldsymbol{M}(\boldsymbol{y})\ddot{\boldsymbol{y}}(t) = \boldsymbol{f}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t) \tag{1}$$

with the symmetric, positive definite mass matrix $\boldsymbol{M}(\boldsymbol{y})$ containing mass and inertia properties of all bodies and the vector of applied and gyroscopic forces $\boldsymbol{f}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t)$ [7, 10].

Despite its implicit structure Eq. (1) can be solved numerically with a complexity that grows only linearly with the number N of bodies in the system. These $\mathcal{O}(N)$*–formalisms* exploit the topology of the multi-body system to evaluate the right-hand side $\boldsymbol{M}^{-1}(\boldsymbol{y})\boldsymbol{f}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t)$ for given $t, \boldsymbol{y}, \dot{\boldsymbol{y}}$ (*explicit* formalisms, e. g. [11, 12]) or the residual $\boldsymbol{r}(\boldsymbol{y}, \dot{\boldsymbol{y}}, \boldsymbol{a}, t) := \boldsymbol{M}(\boldsymbol{y})\boldsymbol{a} - \boldsymbol{f}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t)$ for given $t, \boldsymbol{y}, \dot{\boldsymbol{y}}, \boldsymbol{a}$ (*residual* formalisms, e. g. [13]) efficiently.

In contrast to structural dynamics the standard time integration methods of multi-body dynamics are not tailored to the second order structure of (1) since additional first order equations have to be considered frequently in practical applications, see Section 3. With the velocity vector $\boldsymbol{v} := \dot{\boldsymbol{y}}$ the equations of motion get the state space form

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{\varphi}(\boldsymbol{x}, t) \quad \text{with} \quad \boldsymbol{x}(t) := \begin{pmatrix} \boldsymbol{y}(t) \\ \boldsymbol{v}(t) \end{pmatrix}, \ \boldsymbol{\varphi}(\boldsymbol{x}, t) := \begin{pmatrix} \boldsymbol{v} \\ [\boldsymbol{M}^{-1}\boldsymbol{f}](\boldsymbol{y}, \boldsymbol{v}, t) \end{pmatrix}. \quad (2)$$

Static analysis The computation of static equilibria is often the first step in the analysis of a multi-body system model. They define, e. g., the working point for the linearization of (1) and provide initial values for the dynamical simulation. An equilibrium position $\boldsymbol{y}^*$ is determined by the equilibrium conditions $\dot{\boldsymbol{y}} = \ddot{\boldsymbol{y}} = \boldsymbol{0}$ at $t = t_0$:

$$\boldsymbol{0} = \boldsymbol{f}(\boldsymbol{y}^*, \boldsymbol{0}, t_0)\,. \quad (3)$$

Eqs. (3) form a system of n_y non-linear equations for the n_y unknowns $\boldsymbol{y}^*$ that is solved by Newton's method [14] being available as free software in a number of implementations that meet the demands on robustness and efficiency in engineering applications, see, e. g., the results of the MINPACK project [15] at **http://www.netlib.org/minpack**.

Other static analysis methods are closely related to the equilibrium problem (3): the computation of nominal forces in vehicle dynamics [5] and the trimming of aircraft models in aeronautics [16]. In both applications the actual values of some parameters $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ in the force elements are left undefined in the model setup: $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t; \boldsymbol{\theta})$. Later, these parameters are adjusted to meet equilibrium conditions

$$\boldsymbol{0} = \boldsymbol{f}(\boldsymbol{y}_0, \boldsymbol{0}, t_0; \boldsymbol{\theta}^*) \quad (4)$$

for a pre-defined position $\boldsymbol{y}_0$ of the multi-body system. The actual values $\boldsymbol{\theta}^*$ of nominal forces or trim parameters are important criteria for the evaluation of a vehicle or aircraft design.

Eq. (4) has the same mathematical structure as the classical equilibrium problem (3) and could in principle be solved again by Newton's method. However, in practical applications the system (4) is often underdetermined. Following the approach of Levenberg and Marquardt, (4) may be substituted by

$$\|\boldsymbol{f}(\boldsymbol{y}_0, \boldsymbol{0}, t_0; \boldsymbol{\theta}^*)\|_2 + \alpha\|\boldsymbol{\theta}^*\|_2 \to \min$$

to guarantee uniqueness of $\boldsymbol{\theta}^*$. Here, $\alpha > 0$ denotes a small scalar regularization parameter [17].

Linearization The linearization of the equations of motion (1) is the key to the linear stability analysis near an equilibrium and to other methods of linear system analysis [2]. Writing the right hand side of (1) as $\boldsymbol{f}(\boldsymbol{y}, \dot{\boldsymbol{y}}, \boldsymbol{u}(t))$ with system inputs

$\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$ the linearization at an equilibrium $\boldsymbol{y} = \boldsymbol{y}^*$, $\dot{\boldsymbol{y}} = \boldsymbol{0}$, $\boldsymbol{u} = \boldsymbol{0}$ yields

$$\boldsymbol{M}\ddot{\bar{\boldsymbol{y}}} = -\boldsymbol{D}\dot{\bar{\boldsymbol{y}}} - \boldsymbol{K}\bar{\boldsymbol{y}} + \boldsymbol{B}\boldsymbol{u}(t) \tag{5}$$

with

$$\boldsymbol{M} = \boldsymbol{M}(\boldsymbol{y}^*), \ \ \boldsymbol{D} = -\frac{\partial \boldsymbol{f}}{\partial \dot{\boldsymbol{y}}}(\boldsymbol{y}^*, \boldsymbol{0}, \boldsymbol{0}), \ \ \boldsymbol{K} = -\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(\boldsymbol{y}^*, \boldsymbol{0}, \boldsymbol{0}), \ \ \boldsymbol{B} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}(\boldsymbol{y}^*, \boldsymbol{0}, \boldsymbol{0}).$$

The Jacobians $\partial \boldsymbol{f}/\partial \boldsymbol{y}$, $\partial \boldsymbol{f}/\partial \dot{\boldsymbol{y}}$ and $\partial \boldsymbol{f}/\partial \boldsymbol{u}$ are approximated by classical finite differences [14]. We obtain for the i-th column of $\partial \boldsymbol{f}/\partial \boldsymbol{y}$

$$\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(\boldsymbol{y}^*, \boldsymbol{0}, \boldsymbol{0}) \right)_{\cdot, i} \approx \frac{\boldsymbol{f}(\boldsymbol{y}^* + \Delta \cdot \boldsymbol{e_i}, \boldsymbol{0}, \boldsymbol{0}) - \boldsymbol{f}(\boldsymbol{y}^*, \boldsymbol{0}, \boldsymbol{0})}{\Delta} \tag{6}$$

with the i-th unit vector $\boldsymbol{e_i}$ and a small scalar parameter Δ that satisfies $0 < |\Delta| \ll 1$.

The numerical effort for the linearization (5) is dominated by the difference approximations (6) with a total of $2n_y + n_u + 1$ evaluations of $\boldsymbol{f}$ that require $2n_y + n_u + 1$ evaluations of each force element in the multi-body system model. Substantial savings are achieved by the simultaneous difference approximation of several columns of $\partial \boldsymbol{f}/\partial \boldsymbol{y}$, $\partial \boldsymbol{f}/\partial \dot{\boldsymbol{y}}$ and $\partial \boldsymbol{f}/\partial \boldsymbol{u}$ that requires some graph theoretical preparations based on the topology of the multi-body system [18, 4].

Time Integration The state space form (2) of the equations of motion may be integrated by any standard ODE solver [19]. Explicit Runge–Kutta methods like the 5th order method of Dormand and Prince (free FORTRAN code **DOPRI5** [20] and Matlab's default ODE solver **ode45**) and predictor–corrector methods of Adam's type (free FORTRAN code **LSODE** at **http://www.netlib.org/odepack** and Matlab solver **ode113**) proved to be favourable in non-stiff applications.

More frequently, the model equations in vehicle system dynamics appear to be *stiff* [19, 21] because of stiff springs and strongly damping force elements in most multi-body vehicle models. Backward differentiation formulae (BDF, also: Gear's method) are the most often used time integration methods for stiff technical systems. Starting from initial values $\boldsymbol{x}_0 = \boldsymbol{x}(t_0)$ the numerical solution is obtained time step by time step solving

$$\frac{1}{h_n} \sum_{j=0}^{k_n} \alpha_{n,j} \boldsymbol{x}_{n+1-j} = \boldsymbol{\varphi}(\boldsymbol{x}_{n+1}, t_{n+1}) \tag{7}$$

w. r. t. $\boldsymbol{x}_{n+1} \approx \boldsymbol{x}(t_{n+1})$, ($n = 0, 1, \ldots$).

In (7) the stepsize of time step $t_n \to t_{n+1}$ is denoted by $h_n := t_{n+1} - t_n$. The parameter k_n defines the order of the method, $1 \leq k_n \leq 6$, and $\alpha_{n,j}$ are the BDF coefficients that are determined by k_n and by the stepsizes h_n, h_{n-1}, $\ldots$ [19]. The well known backward Euler method

$$\frac{1}{h_n}(\boldsymbol{x}_{n+1} - \boldsymbol{x}_n) = \boldsymbol{\varphi}(\boldsymbol{x}_{n+1}, t_{n+1})$$

is a special case of (7), $k_n = 1$.

For given state vectors $\boldsymbol{x}_{n+1-j} \approx \boldsymbol{x}(t_{n+1-j})$, $(\, j = 1, \ldots, k\,)$ the BDF equations (7) define $\boldsymbol{x}_{n+1}$ implicitly as solution of the system of n_x non-linear equations

$$\mathbf{0} = \boldsymbol{\Phi}(\boldsymbol{x}) := \frac{\alpha_{n,0}}{h_n}\boldsymbol{x} - \boldsymbol{\varphi}(\boldsymbol{x}, t_{n+1}) + \frac{1}{h_n}\sum_{j=1}^{k_n} \alpha_{n,j}\boldsymbol{x}_{n+1-j}\,. \tag{8}$$

The application of Newton's method to (8) yields

$$\boldsymbol{x}_{n+1}^{(l+1)} := \boldsymbol{x}_{n+1}^{(l)} - \boldsymbol{J}^{-1} \cdot \boldsymbol{\Phi}(\boldsymbol{x}_{n+1}^{(l)})\,, \quad (\, l \geq 0\,) \tag{9}$$

with an initial guess $\boldsymbol{x}_{n+1}^{(0)}$ that is obtained by polynomial extrapolation of $\boldsymbol{x}_n$, $\boldsymbol{x}_{n-1}, \ldots$. In (9), matrix J approximates the Jacobian of (8):

$$\boldsymbol{J} \approx \frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{x}}(\boldsymbol{x}_{n+1}^{(l)}) = \frac{\alpha_{n,0}}{h_n}\boldsymbol{I}_{n_y} - \frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{x}}(\boldsymbol{x}_{n+1}^{(l)}, t_{n+1})\,. \tag{10}$$

Typically, the approximation of $\partial\boldsymbol{\varphi}/\partial\boldsymbol{x}$ by finite differences according to (6) dominates the overall computing time in the dynamical simulation because of the large number of function evaluations of $\boldsymbol{\varphi}$ that require the evaluation of all force elements in the multi-body system and the computation of $[\boldsymbol{M}^{-1}\boldsymbol{f}](\boldsymbol{y}, \boldsymbol{v}, t)$ by the $\mathcal{O}(N)$–formalism, see (2). Therefore these time consuming re-evaluations of the Jacobian J have to be avoided as far as possible [19, 4].

In practical computations stepsize h_n and order k_n are adapted automatically to meet the user defined error tolerances. The combination of stepsize and order control with the algorithms for Newton's method (9) and for the Jacobian re-evaluations results in a fairly complicated structure of state-of-the-art BDF solvers. Free FORTRAN codes are **LSODE** at **http://www.netlib.org/odepack** and **DASSL** at **http://www.netlib.org/ode**. The Matlab BDF solver is **ode15s**.

From the viewpoint of multi-body dynamics the solver **DASSL** [22] is especially interesting because of its interface in residual form

$$\mathbf{0} = \boldsymbol{F}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) \tag{11}$$

that is a generalization of (2) with $\boldsymbol{F}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) = \dot{\boldsymbol{x}} - \boldsymbol{\varphi}(\boldsymbol{x}, t)$ and optionally allows also the use of very efficient residual formalisms [13]:

$$\boldsymbol{x}(t) := \begin{pmatrix} \boldsymbol{y}(t) \\ \boldsymbol{v}(t) \end{pmatrix}, \quad \boldsymbol{F}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) := \begin{pmatrix} \dot{\boldsymbol{y}} - \boldsymbol{v} \\ \boldsymbol{M}(\boldsymbol{y})\dot{\boldsymbol{v}} - \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{v}, t) \end{pmatrix}.$$

Furthermore, **DASSL** may also be used in the time integration of closed loop systems, see Section 4 below.

BDF for (11) have the form

$$\mathbf{0} = \boldsymbol{F}(\boldsymbol{x}_{n+1}, \frac{1}{h_n} \sum_{j=0}^{k_n} \alpha_{n,j} \boldsymbol{x}_{n+1-j}, t_{n+1}) \tag{12}$$

with the Jacobian

$$\boldsymbol{J} \approx \frac{\alpha_{n,0}}{h_n} \frac{\partial \boldsymbol{F}}{\partial \dot{\boldsymbol{x}}}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) + \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) \tag{13}$$

instead of (10). To keep the efficiency of classical BDF solvers in the application to multi-body systems **DASSL**'s standard algorithm for Jacobian re-evaluation has to be modified to exploit the special block structure of $\partial \boldsymbol{F} / \partial \dot{\boldsymbol{x}}$, see [4, 23]:

Explicit formalism: $\dfrac{\partial \boldsymbol{F}}{\partial \dot{\boldsymbol{x}}} = \begin{pmatrix} \boldsymbol{I}_{n_y} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I}_{n_y} \end{pmatrix}$, residual formalism: $\dfrac{\partial \boldsymbol{F}}{\partial \dot{\boldsymbol{x}}} = \begin{pmatrix} \boldsymbol{I}_{n_y} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{M} \end{pmatrix}$.

3. GENERAL STRUCTURE OF MODEL EQUATIONS

The numerical methods that are discussed in Section 2 may be extended from the equations of motion (1) to more complex model equations that are typical of applications in vehicle system dynamics.

Flexible multi-body systems The elastic deformation of flexible bodies is not covered by the classical equations of motion (1) of rigid multi-body systems. In the moving frame of reference formulation [24, 25, 26] the configuration $\vec{u}(\vec{x}, t)$ of a flexible body $(.)^{(i)}$ in $\mathbb{R}^3$ is represented by the gross motion of a body fixed reference frame with coordinates $\boldsymbol{y}_{\mathrm{r}}^{(i)}(t)$ and the (small) elastic deformation $\vec{w}(\vec{x}, t)$ w. r. t. this reference configuration:

$$\vec{u}^{(i)}(\vec{x}, t) = \vec{r}(\boldsymbol{y}_{\mathrm{r}}^{(i)}(t)) + \vec{w}^{(i)}(\vec{x}, t) \,. \tag{14}$$

Following a Ritz approach a low-dimensional modal approximation of $\vec{w}^{(i)}(\vec{x}, t)$ is used:

$$\vec{w}^{(i)}(\vec{x}, t) \approx \sum_{j=1}^{\sigma_i} q_j^{(i)}(t)\, \vec{w}_j^{(i)}(\vec{x}) \,. \tag{15}$$

In general, the σ_i modes $\vec{w}_j^{(i)}$ are selected based on a finite element analysis of the flexible body $(.)^{(i)}$. Typical ansatz functions $\vec{w}_j^{(i)}$ are eigenmodes in the frequency range up to $50 \ldots 100\,$Hz, static modes and frequency response modes [26, 27].

With (14) and (15) the equations of motion for flexible multi-body systems get the basic structure (1) with a coordinate vector $\boldsymbol{y}(t)$ that contains the generalized coordinates of rigid bodies and the elastic coordinates $\boldsymbol{y}_{\mathrm{r}}^{(i)}(t)$ and $(\, q_1^{(i)}(t), \, \ldots \, , q_{\sigma_i}^{(i)}(t) \,)$

of flexible bodies. $\boldsymbol{M}(\boldsymbol{y})$ and $\boldsymbol{f}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t)$ are extended by the modal mass, damping and stiffness matrices and by the coupling terms between gross motion and elastic deformation $\vec{w}$, see [25, 28]. In industrial applications these data are generated automatically using standardized interfaces to finite element tools [29].

Today, the selection of an appropriate set of modes $\{ \vec{w}_1^{(i)}, \ldots, \vec{w}_{\sigma_i}^{(i)} \}$ still relies much on engineering intuition. Algorithms for (semi-)automatic mode selection are a topic of active research, see [27] for a criterion that is based on a linear error analysis, [28] for a more general approach that could be used in an adaptive mode selection strategy and [30] for the use of *thermal response* modes in thermoelastic applications.

Recently, the moving frame of reference formulation has been extended to a modal multifield approach that considers elastic deformation as well as electrostatic fields in piezo-elements and spatial temperature distributions in thermoelastic problems like the thermal deformation of disk brakes [30].

Inner state variables Frequently, force elements in multi-body system models describe engineering systems that have their own, internal dynamics (*dynamical* force elements [5]). Typical examples include hydraulic components, control devices and advanced tyre models. From the mathematical viewpoint these force elements are characterized by *inner* state variables. Elements like hydraulic components have time continuous state variables $\boldsymbol{c}(t)$ with state equations

$$\dot{\boldsymbol{c}}(t) = \boldsymbol{d}(\boldsymbol{c}, \boldsymbol{y}, \dot{\boldsymbol{y}}, t) . \tag{16}$$

The internal dynamics of discrete controllers is characterized by time discrete state variables $\boldsymbol{r}$ with $\boldsymbol{r} = \boldsymbol{r}_j$ in the sampling interval $[T_j, T_{j+1})$ and

$$\boldsymbol{r}_{j+1} = \boldsymbol{k}(\boldsymbol{r}_j, \boldsymbol{r}_{j-1}, \ldots, \boldsymbol{y}, \dot{\boldsymbol{y}}, T_{j+1}) . \tag{17}$$

In most applications the sampling points are equally spaced: $T_j = j \cdot \Delta_t$, typical sampling rates are in the range of $\Delta_t = 1\,\text{ms} \ \ldots \ 10\,\text{ms}$.

In the multi-body system, these force elements contribute to the force vector $\boldsymbol{f}$ in (1):

$$\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{y}(t), \dot{\boldsymbol{y}}(t), \boldsymbol{c}(t), \boldsymbol{r}_j, t) \quad \text{if} \quad t \in [T_j, T_{j+1}) . \tag{18}$$

With (17) and (18) the equations of motion are not longer continuous in time but form a hybrid system with continuous and discrete components. In the dynamical simulation the variable order, variable stepsize integrators of Section 2 may be used only within a single sampling interval $[T_j, T_{j+1}]$. At $t = T_{j+1}$ the integration has to be stopped, the discrete variables $\boldsymbol{r}$ are updated according to (17) and the time integration may be continued for $t \in [T_{j+1}, T_{j+2}]$.

The frequent re-initializations of the ODE solvers have a very negative effect on their performance that is compensated in part by an adapted algorithm for Jacobian re-evaluations [4]. Alternatively, special Runge–Kutta type methods are proposed for a more efficient re-initialization after the sampling point T_{j+1}, see [31].

Time events The discontinuous changes (17) of state variables are special *time events* [19]. The term "time event" refers to a discontinuity ("event") in state variables or model equations at an isolated time $t = T^*$ during simulation. For model equations (1) with $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{y}, \dot{\boldsymbol{y}}, \boldsymbol{c}, \boldsymbol{r}_j, \boldsymbol{u}(t); \boldsymbol{\theta})$ typical time events result from switching processes with jumping system inputs $\boldsymbol{u}(t)$ and from contact problems including stick-slip phenomena with force elements that change their force parameters $\boldsymbol{\theta}$ or even their internal structure during simulation (unilateral force elements, varying friction coefficients $\boldsymbol{\theta}, \ldots$ [5, 32]).

For switching inputs $\boldsymbol{u}(t)$ the time T^* is known in advance but, in general, T^* has to be determined during simulation. State-of-the-art ODE solvers locate time events automatically using *switching functions* that have to be provided in addition to the model equations (1), see [5, 22].

In case of a time event the time integration is stopped at $t = T^*$, system variables $\boldsymbol{y}$, $\boldsymbol{c}$, $\boldsymbol{r}_j$ and system parameters $\boldsymbol{\theta}$ are updated (if necessary) and the simulation is continued after a re-initialization of the solver. Again, frequent time events may slow down the time integration substantially.

Kinematically closed loops In the setup of multi-body system models it is important to select an appropriate set of coordinates $\boldsymbol{y}$. For tree structured systems joint coordinates $\boldsymbol{y}$ that describe the degrees of freedom in the joints of the multi-body system form a minimum set of generalized coordinates resulting in equations of motion in ODE form (1).

If, however, the multi-body system has kinematically closed loops then the joint coordinates $\boldsymbol{y}$ are not longer independent of each other. Loop closing joints restrict the system's configuration to joint coordinates $\boldsymbol{y}$ satisfying n_g *constraints*

$$\mathbf{0} = \boldsymbol{g}(\boldsymbol{y}, t)\,. \tag{19}$$

Coordinate partitioning methods [7, 33] locally select a sub-set of linearly independent joint coordinates and compute the remaining (dependent) joint coordinates solving the system of non-linear equations (19). From the numerical viewpoint it proved to be more favourable to keep *all* joint coordinates in the equations of motion that get the form

$$\boldsymbol{M}(\boldsymbol{y})\,\ddot{\boldsymbol{y}}(t) = \boldsymbol{f}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t) - \boldsymbol{G}^\top(\boldsymbol{y}, t)\boldsymbol{\lambda}\,, \tag{20a}$$

$$\mathbf{0} = \boldsymbol{g}(\boldsymbol{y}, t) \tag{20b}$$

with constraint forces $-\boldsymbol{G}^\top(\boldsymbol{y}, t)\boldsymbol{\lambda}$ that are determined by the constraint matrix $\boldsymbol{G}(\boldsymbol{y}, t) := (\partial \boldsymbol{g}/\partial \boldsymbol{y})(\boldsymbol{y}, t)$ and the Lagrangian multipliers $\boldsymbol{\lambda}(t) \in \mathbb{R}^{n_g}$, see [12]. The common assumption $\operatorname{rank} \boldsymbol{G}(\boldsymbol{y}, t) = n_g$ (*Grübler* condition) excludes redundant constraints (20b). Eqs. (20) form a second order differential-algebraic equation (DAE), the *descriptor form* of the equations of motion [22].

The descriptor form is not restricted to joint coordinates $\boldsymbol{y}$. Using absolute (Cartesian) coordinates $\boldsymbol{y}$ *any* joint results in a constraint (20b). The resulting equations of motion (20) have a much larger dimension than in the joint coordinate case but $\boldsymbol{M}(\boldsymbol{y})$, $\partial \boldsymbol{f}/\partial \boldsymbol{y}$, $\partial \boldsymbol{f}/\partial \dot{\boldsymbol{y}}$, $\boldsymbol{G}^\top$ are sparse matrices with a structure that reflects the topology of the multi-body system [34]. The counterpart to $\mathcal{O}(N)$–formalisms that eliminate redundant coordinates in a joint coordinate formulation [11, 12] are *topological solvers* for the absolute coordinate formulation that allow an efficient elimination of redundant coordinates in (9), see [34, 31].

In DAE terminology the descriptor form (20) is an index-3 system, see [21, 22]. The constraints (20b) imply additional restrictions on the state variables $\boldsymbol{y}(t)$ and $\boldsymbol{\lambda}(t)$. These *hidden* constraints are obtained by differentiation of (20b):

$$\begin{aligned} \mathbf{0} &= \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{g}(\boldsymbol{y}(t),t) = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}}(\boldsymbol{y},t)\cdot\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}t}(t) + \frac{\partial \boldsymbol{g}}{\partial t}(\boldsymbol{y},t) \\ &= \boldsymbol{G}(\boldsymbol{y},t)\,\dot{\boldsymbol{y}}(t) + \boldsymbol{g}^{(\boldsymbol{I})}(\boldsymbol{y},t)\,, \end{aligned} \tag{21a}$$

$$\mathbf{0} = \frac{\mathrm{d}^2}{\mathrm{d}t^2}\boldsymbol{g}(\boldsymbol{y}(t),t) = \boldsymbol{G}(\boldsymbol{y},t)\,\ddot{\boldsymbol{y}}(t) + \boldsymbol{g}^{(\boldsymbol{II})}(\boldsymbol{y},\dot{\boldsymbol{y}},t) \tag{21b}$$

with functions $\boldsymbol{g}^{(\boldsymbol{I})}$, $\boldsymbol{g}^{(\boldsymbol{II})}$ that summarize higher order derivatives of $\boldsymbol{g}$ and the partial derivatives w. r. t. t. The solution of (20) has satisfy the constraints (20b) on position level as well as the hidden constraints (21) on velocity and acceleration level.

Initial values $\boldsymbol{y}_0$, $\dot{\boldsymbol{y}}_0$, $\boldsymbol{\lambda}_0$ have to be *consistent* with all these constraints:

$$\begin{aligned} &\mathbf{0} = \boldsymbol{g}(\boldsymbol{y}_0,t_0)\,, \quad \mathbf{0} = \boldsymbol{G}(\boldsymbol{y}_0,t_0)\,\dot{\boldsymbol{y}}_0 + \boldsymbol{g}^{(\boldsymbol{I})}(\boldsymbol{y}_0,t_0)\,, \\ &\mathbf{0} = \boldsymbol{G}(\boldsymbol{y}_0,t_0)\,\ddot{\boldsymbol{y}}_0 + \boldsymbol{g}^{(\boldsymbol{II})}(\boldsymbol{y}_0,\dot{\boldsymbol{y}}_0,t_0) \end{aligned} \tag{22}$$

with $\boldsymbol{M}(\boldsymbol{y}_0)\,\ddot{\boldsymbol{y}}_0(t) = \boldsymbol{f}(\boldsymbol{y}_0,\dot{\boldsymbol{y}}_0,t_0) - \boldsymbol{G}^\top(\boldsymbol{y}_0,t_0)\boldsymbol{\lambda}_0$.

The computation of consistent initial values $\boldsymbol{y}_0$, $\dot{\boldsymbol{y}}_0$, $\boldsymbol{\lambda}_0$ is part of the initialization in the time integration of (20). If initial values y_{0,i_j} are given for $n_y - n_g$ independent position coordinates y_{i_j}, $(\,j = 1,\dots,n_y - n_g\,)$, the remaining initial values y_{0,i_j}, $(\,j = n_y - n_g + 1,\dots,n_y\,)$ are uniquely determined by (22) and may be computed by Newton's method. Furthermore, (22) defines systems of linear equations for the initial values $\dot{y}_{0,i_j}$, $(\,j = n_y - n_g + 1,\dots,n_y\,)$ of n_g dependent velocity components and for the initial values $\boldsymbol{\lambda}_0$.

Advanced DAE models The descriptor form (20) is linear w. r. t. $\boldsymbol{\lambda}$. This structural property is lost if the model has friction forces that depend on the constraint forces $-\boldsymbol{G}^\top\boldsymbol{\lambda}$. The resulting force vector $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{y},\dot{\boldsymbol{y}},\boldsymbol{\lambda},t)$ has to satisfy the generalized

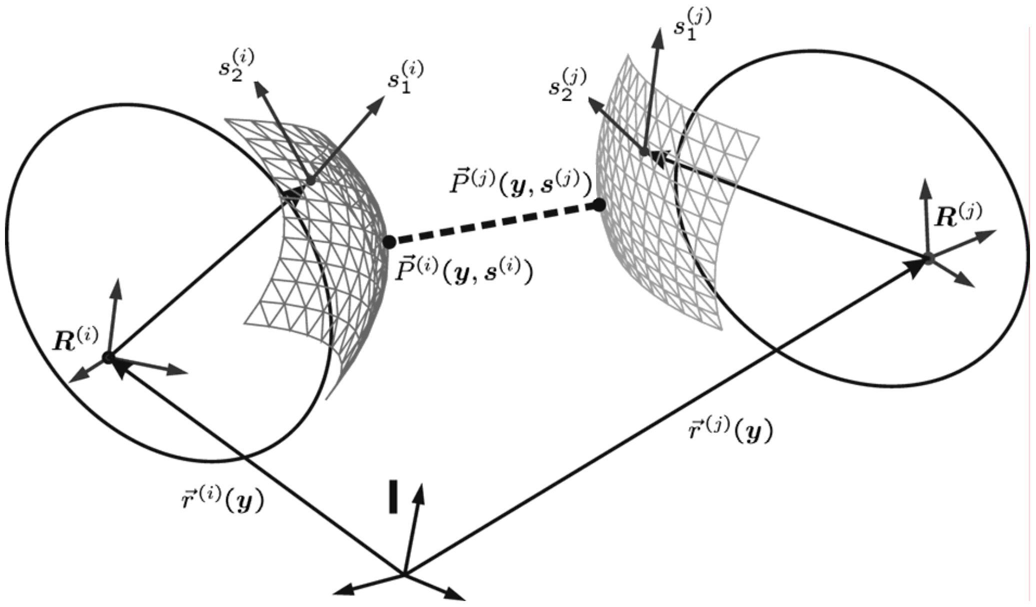
$s_2^{(i)}$
$s_1^{(i)}$
$s_1^{(j)}$
$s_2^{(j)}$
$\vec{P}^{(j)}(\boldsymbol{y},\boldsymbol{s}^{(j)})$
$\vec{P}^{(i)}(\boldsymbol{y},\boldsymbol{s}^{(i)})$
$\boldsymbol{R}^{(j)}$
$\boldsymbol{R}^{(i)}$
$\vec{r}^{(j)}(\boldsymbol{y})$
$\vec{r}^{(i)}(\boldsymbol{y})$
I

Fig. 1. Formulation of contact conditions using surface parameters $\boldsymbol{s}$ as additional algebraic variables.

Grübler condition

$$\text{rank}\,\boldsymbol{G}(\boldsymbol{y},t) = \text{rank}\Big(\boldsymbol{G}(\boldsymbol{y},t)\,\boldsymbol{M}^{-1}(\boldsymbol{y})\,\Big(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\lambda}}(\boldsymbol{y},\dot{\boldsymbol{y}},\boldsymbol{\lambda},t) - \boldsymbol{G}^\top(\boldsymbol{y},t)\Big)\Big) = n_g \quad (23)$$

to guarantee the unique solvability of (20).

Another extension of (20) shows the large potential of DAE formulations in view of user-friendly algorithms for model setup. Consider rigid bodies $(.)^{(i)}$ and $(.)^{(j)}$ that are in contact. To formulate the contact conditions parametrizations $\boldsymbol{s}^{(i)}, \boldsymbol{s}^{(j)} \in \mathbb{R}^2$ of the body surfaces are introduced. Then the position of the points $\vec{P}^{(i)}$ on the surface of Body $(.)^{(i)}$ in $\mathbb{R}^3$ is determined by the position and orientation of this body, i. e., by the multi-body system coordinates $\boldsymbol{y}$, and by the surface parameter $\boldsymbol{s}^{(i)}$: $\vec{P}^{(i)} = \vec{P}^{(i)}(\boldsymbol{y},\boldsymbol{s}^{(i)})$, see Fig. 1.

If both bodies are strictly convex then there is a uniquely defined contact point $\vec{P}^{(i)}(\boldsymbol{y},\boldsymbol{s}^{(i)}) = \vec{P}^{(j)}(\boldsymbol{y},\boldsymbol{s}^{(j)})$ that belongs to the surfaces of both bodies. The contact point is characterized by the condition that the normal vector of Body $(.)^{(i)}$ at $\vec{P}^{(i)}$ is parallel to the normal vector of Body $(.)^{(j)}$ at $\vec{P}^{(j)}$, see [12]. This geometrical condition defines implicitly the contact point coordinates $\boldsymbol{s} = (\boldsymbol{s}^{(i)},\boldsymbol{s}^{(j)})$ as solution $\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{y},t)$ of a non-linear system

$$\mathbf{0} = \boldsymbol{h}(\boldsymbol{y},\boldsymbol{s},t) \quad (24)$$

with non-singular Jacobian $\partial\boldsymbol{h}/\partial\boldsymbol{s}$, see [5, 6].

Contact and friction forces act at the contact point. Furthermore, the contact point coordinates $\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{y}, t)$ are used in the formulation of the contact condition

$$0 = \vec{P}^{(i)}(\boldsymbol{y}, \boldsymbol{s}^{(i)}) - \vec{P}^{(j)}(\boldsymbol{y}, \boldsymbol{s}^{(j)}) \tag{25}$$

that defines constraints $\mathbf{0} = \boldsymbol{g}(\boldsymbol{y}, t)$ in (20). In a classical implementation each evaluation of $\boldsymbol{g}(\boldsymbol{y}, t)$ would require the solution of (24) to get the contact point coordinates $\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{y}, t)$ that have to be inserted in the contact condition (25).

As an alternative, the DAE framework allows to add simply the non-linear equations (24) to the equations of motion (20) and to append the contact conditions (25) directly to the constraints:

$$\boldsymbol{M}(\boldsymbol{y})\,\ddot{\boldsymbol{y}}(t) = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{s}, \dot{\boldsymbol{y}}, \boldsymbol{\lambda}, t) - \boldsymbol{G}^{\top}(\boldsymbol{y}, \boldsymbol{s}, t)\boldsymbol{\lambda}\,, \tag{26a}$$

$$\mathbf{0} = \boldsymbol{h}(\boldsymbol{y}, \boldsymbol{s}, t)\,, \tag{26b}$$

$$\mathbf{0} = \boldsymbol{g}(\boldsymbol{y}, \boldsymbol{s}, t)\,. \tag{26c}$$

With this formulation, the solution of (24) w. r. t. $\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{y}, t)$ is left to the DAE solver [4, 6]. Note, that the constraint matrix is now

$$\begin{aligned}\boldsymbol{G}(\boldsymbol{y}, \boldsymbol{s}, t) &= \frac{\mathrm{D}}{\mathrm{D}\boldsymbol{y}}\boldsymbol{g}(\boldsymbol{y}, \boldsymbol{s}(\boldsymbol{y}, t), t) = \Big[\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{s}}\,\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{y}}\Big](\boldsymbol{y}, \boldsymbol{s}, t) \\ &= \Big[\,\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{y}} - \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{s}}\Big(\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{s}}\Big)^{-1}\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{y}}\,\Big](\boldsymbol{y}, \boldsymbol{s}, t)\end{aligned} \tag{27}$$

because of

$$\mathbf{0} = \frac{\mathrm{D}}{\mathrm{D}\boldsymbol{y}}\boldsymbol{h}(\boldsymbol{y}, \boldsymbol{s}(\boldsymbol{y}, t), t) = \Big[\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{y}} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{s}}\,\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{y}}\Big](\boldsymbol{y}, \boldsymbol{s}, t)\,. \tag{28}$$

From the mathematical viewpoint, Eq. (24) is just a special case of algebraic equations

$$\mathbf{0} = \boldsymbol{b}(\boldsymbol{w}, \boldsymbol{y}, \dot{\boldsymbol{y}}, \boldsymbol{\lambda}, t) \tag{29}$$

with variables $\boldsymbol{w}(t) \in \mathbb{R}^{n_b}$ that are defined implicitly by (29) if $\partial\boldsymbol{b}/\partial\boldsymbol{w}$ is non-singular. The mathematical modelling of force elements may be simplified substantially using such auxiliary variables $\boldsymbol{w}$. A typical example is the modelling of joint friction, see [4] for a detailed discussion.

With variables $\boldsymbol{w}$ the force vector in (26) gets the general form $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{s}, \dot{\boldsymbol{y}}, \boldsymbol{\lambda}, \boldsymbol{w}, t)$ and the Jacobian $\partial\boldsymbol{f}/\partial\boldsymbol{\lambda}$ in the generalized Grübler condition (23) has to be substituted by

$$\Big[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\lambda}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w}}\Big(\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{w}}\Big)^{-1}\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{\lambda}}\Big](\boldsymbol{y}, \boldsymbol{s}, \dot{\boldsymbol{y}}, \boldsymbol{\lambda}, \boldsymbol{w}, t)\,.$$

Summary Based on the multi-body system approach the model equations in vehicle system dynamics are formulated as hybrid system of differential-algebraic equations

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{y})\,\ddot{\boldsymbol{y}}(t) &= \boldsymbol{f}(\boldsymbol{y},\boldsymbol{s},\dot{\boldsymbol{y}},\boldsymbol{c},\boldsymbol{r}_j,\boldsymbol{\lambda},\boldsymbol{w},t) - \boldsymbol{G}^\top(\boldsymbol{y},\boldsymbol{s},t)\boldsymbol{\lambda}\,, && \text{(30a)}\\ \dot{\boldsymbol{c}}(t) &= \boldsymbol{d}(\boldsymbol{c},\boldsymbol{y},\boldsymbol{s},\dot{\boldsymbol{y}},\boldsymbol{r}_j,\boldsymbol{\lambda},\boldsymbol{w},t)\,. && \text{(30b)}\\ \boldsymbol{0} &= \boldsymbol{b}(\boldsymbol{w},\boldsymbol{y},\boldsymbol{s},\dot{\boldsymbol{y}},\boldsymbol{c},\boldsymbol{r}_j,\boldsymbol{\lambda},t)\,, && \text{(30c)}\\ \boldsymbol{0} &= \boldsymbol{h}(\boldsymbol{y},\boldsymbol{s},\boldsymbol{r}_j,t)\,, && \text{(30d)}\\ \boldsymbol{0} &= \boldsymbol{g}(\boldsymbol{y},\boldsymbol{s},\boldsymbol{r}_j,t) && \text{(30e)} \end{aligned}$$

with $t \in [T_j, T_{j+1})$ and discrete state equations

$$\boldsymbol{r}_{j+1} = \boldsymbol{k}(\boldsymbol{r}_j,\boldsymbol{r}_{j-1},\ldots,\boldsymbol{y},\boldsymbol{s},\dot{\boldsymbol{y}},\boldsymbol{c},\boldsymbol{\lambda},\boldsymbol{w},T_{j+1})\,. \tag{31}$$

The continuous part (30) of the model equations forms an index-3 DAE with a complex but characteristic structure that will be exploited in the numerical solution, see Section 4.

It is important that the contact point coordinates $\boldsymbol{s}$ in the constraints (30e) are defined by an equation that is independent of $\dot{\boldsymbol{y}}$ and $\boldsymbol{\lambda}$, see (30d). Otherwise the model equations would not have been solvable unless strong additional regularity conditions were satisfied.

4. DAE TIME INTEGRATION IN VEHICLE SYSTEM DYNAMICS

The time integration of DAE model equations (30) is based on classical ODE methods combined with *index reduction* and *projection* techniques. The time integration methods will be discussed in detail for DAEs (20) that are re-written as first order systems in residual form (11) with

$$\boldsymbol{x}(t) = \begin{pmatrix} \boldsymbol{y}(t)\\ \boldsymbol{v}(t)\\ \boldsymbol{\lambda}(t)\end{pmatrix},\quad \boldsymbol{0} = \boldsymbol{F}(\boldsymbol{x},\dot{\boldsymbol{x}},t) = \begin{pmatrix} \dot{\boldsymbol{y}} - \boldsymbol{v}\\ \boldsymbol{M}(\boldsymbol{y})\dot{\boldsymbol{v}} - \boldsymbol{f}(\boldsymbol{y},\boldsymbol{v},t) + \boldsymbol{G}^\top(\boldsymbol{y},t)\boldsymbol{\lambda}\\ \boldsymbol{g}(\boldsymbol{y},t)\end{pmatrix}. \tag{32}$$

At the end of the section we will come back to DAEs of the more general form (30).

Throughout the present section a benchmark problem from railway vehicle dynamics is used to illustrate the numerical problems in DAE time integration. The model describes a rigid wheelset with conic wheels moving with constant speed along a straight track. Starting with a small initial lateral displacement y the wheelset oscillates in lateral direction, see Fig. 2. This so called hunting motion is a well known phenomenon in railway vehicle dynamics.

The equations of motion are formulated in DAE form (26) with the position coordinates $\boldsymbol{y}(t) \in \mathbb{R}^6$ of the wheelset that correspond to its six degrees of freedom.

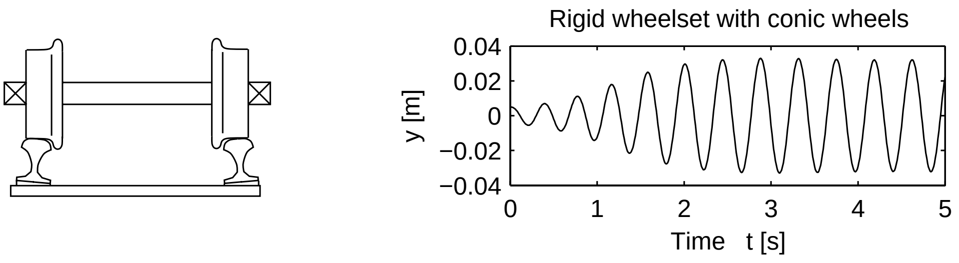

Fig. 2. Lateral displacement y of a rigid wheelset performing a hunting motion.

Two constraints (26c) guarantee permanent contact between the two wheels and the rails [6]. Exploiting symmetries in the geometry of wheel and rail the contact point between a single wheel and the rail may be described by a scalar position coordinate $s(t)$ resulting in two equations (26b) that determine implicitly $\boldsymbol{s} = (s_l, s_r)^\top \in \mathbb{R}^2$ for the contact point coordinates on left and right wheel [35].

DAE time integration by ODE methods Eqs. (32) suggest to apply BDF (12) and other ODE methods straightforwardly to the time integration of DAEs. Simple test problems like (32) with $\boldsymbol{M} = \boldsymbol{I}_{n_y}$, $\boldsymbol{f} = \boldsymbol{0}$ and linear constraints

$$\boldsymbol{0} = \boldsymbol{g}(\boldsymbol{y}, t) = \boldsymbol{C}\boldsymbol{y} - \boldsymbol{z}(t)$$

show that this approach fails in general: According to (20a) and (21b) we get $\ddot{\boldsymbol{y}}(t) = -\boldsymbol{C}^\top\boldsymbol{\lambda}$ and $\boldsymbol{C}\ddot{\boldsymbol{y}}(t) = \ddot{\boldsymbol{z}}(t)$ resulting in Lagrangian multipliers $\boldsymbol{\lambda}(t) = -(\boldsymbol{C}\boldsymbol{C}^\top)^{-1}\ddot{\boldsymbol{z}}(t)$. The BDF solution $\boldsymbol{x}_n = (\boldsymbol{y}_n, \boldsymbol{v}_n, \boldsymbol{\lambda}_n)^\top$ satisfies (12) and hence also

$$\boldsymbol{0} = \boldsymbol{g}(\boldsymbol{y}_{\boldsymbol{n}}, t_n) = \boldsymbol{C}\boldsymbol{y}_n - \boldsymbol{z}(t_n)\,, \quad (\, n > 0\,)$$

since the constraints (20b) are part of the residual $\boldsymbol{F}$ in (32). A simple computation shows that the application of backward Euler method to (32), i. e., BDF (12) with $k_n = 1$, yields

$$\boldsymbol{\lambda}_{n+1} = \frac{h_n + h_{n-1}}{2h_n}\boldsymbol{\lambda}(t_{n+1}) + \mathcal{O}(h_n) + \mathcal{O}(\frac{h_{n-1}^2}{h_n})\,.$$

For varying stepsizes the numerical solution $\boldsymbol{\lambda}_{n+1}$ is completely wrong since it does not converge to the analytical solution $\boldsymbol{\lambda}(t_{n+1})$ if $h_n \to 0$ and h_{n-1} is fixed, see also [36].

For some higher order methods convergence can still be guaranteed [21] but the errors in the numerical solution are typically substantially larger than in the ODE case. This is illustrated by simulation results for the hunting motion of the rigid wheelset in Fig. 2. In this application, the BDF integrator **DASSL** applied to (26) fails completely. As an alternative the implicit Runge–Kutta solver **RADAU5** was used in the numerical tests [21]. For **RADAU5**, experience shows that the error of the numerical solution in

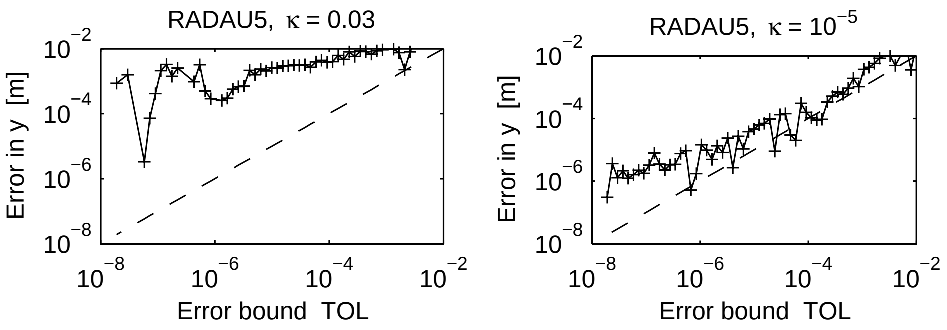

Fig. 3. ODE solver **RADAU5** applied to the differential-algebraic equations of motion (20) of a rigid wheelset.

ODE applications is usually kept well below the user-defined error tolerances TOL. But in the application to DAE (26) the relative error remains even for extremely small error bounds always in the size of $0.1\ldots1.0\,\%$. As a typical example, the left plot of Fig. 3 shows the error in the lateral displacement y for various values of TOL.

To analyse these unsatisfactory results of **RADAU5** we modify one of the internal parameters of the solver. BDF (12) and implicit Runge–Kutta methods define $\boldsymbol{x}_{n+1}$ as solution of a system of non-linear equations. In the practical implementation this system is solved iteratively by Newton's method that is stopped if the iteration error is less than $\kappa\cdot$ TOL. In this stopping criterion the user-defined error tolerance for time integration (TOL) is scaled by a constant $\kappa\leq1$ that is a free control parameter of the solver. Default values are $\kappa=0.33$ in **DASSL** [22] and $\kappa=0.03$ in **RADAU5** [21].

The right plot of Fig. 3 shows that the error in time integration is reduced drastically and remains now roughly below the error bounds if κ is set to the very small value $\kappa=10^{-5}$. The comparison of left and right plot in Fig. 3 illustrates that the direct application of **RADAU5** to DAEs (26) makes the solver very sensitive w. r. t. (small) iteration errors in Newton's method.

An analytical perturbation analysis for (32) shows that the errors in $\boldsymbol{y}_n$, $\boldsymbol{v}_n$ and $\boldsymbol{\lambda}_n$ are of size $\mathcal{O}(\theta)+\mathcal{O}(\theta^2/h^2)$, $\mathcal{O}(\theta/h)$ and $\mathcal{O}(\theta/h^2)$, respectively, if the errors in the constraints (20b) are bounded by some small positive constant $\theta\ll1$, see [37]. Note, that these error bounds *grow* unboundedly for decreasing time stepsize $h:=\min_m h_m$.

This error amplification is typical of the direct application of ODE solvers to DAE (32). It may be considered as discrete analogue of corresponding error bounds $\mathcal{O}(\max_t\|\boldsymbol{\theta}(t)\|)+\mathcal{O}(\max_t\|\dot{\boldsymbol{\theta}}(t)\|^2)$, $\mathcal{O}(\max_t\|\dot{\boldsymbol{\theta}}(t)\|)$ and $\mathcal{O}(\max_t\|\ddot{\boldsymbol{\theta}}(t)\|)$ for the analytical solution $\boldsymbol{y}(t)$, $\boldsymbol{v}(t)$ and $\boldsymbol{\lambda}(t)$ of (32) with perturbed constraints

$$\boldsymbol{\theta}(t)=\boldsymbol{g}(\boldsymbol{y}(t),t)$$

and $\max_t\|\boldsymbol{\theta}(t)\|\ll1$, see [37].

For perturbations like $\boldsymbol{\theta}(t) = \varepsilon \sin \omega t$ that oscillate with small amplitude $\varepsilon \ll 1$ and high frequency $\omega \gg 1$ the errors in the solution $(\boldsymbol{y}(t), \boldsymbol{v}(t), \boldsymbol{\lambda}(t))$ are much larger than the perturbation itself since $\|\dot{\boldsymbol{\theta}}(t)\| = \mathcal{O}(\omega\varepsilon)$, $\|\ddot{\boldsymbol{\theta}}(t)\| = \mathcal{O}(\omega^2\varepsilon)$. A mechanical interpretation of these results is the well known fact that high frequency kinematic excitations $\boldsymbol{\theta}(t)$ in loop closing joints may cause large joint and constraint forces in a multi-body system.

Index reduction and projection In the direct application of ODE methods to DAE (32) the robustness of the solvers may be improved substantially by small values of κ that result in (very) small iteration errors in Newton's method, see the right plot of Fig. 3. On the other hand values of κ that are less than 10^{-3} increase the number of Newton steps per time step substantially and may slow down the ODE solver dramatically.

Instead of applying ODE solvers directly to (32) it proved to be much more advantageous to transform the equations of motion *analytically* before time integration. These analytical transformations are guided by the perturbation analysis. In (20) the Lagrangian multipliers $\boldsymbol{\lambda}(t)$ are defined implicitly by the constraints (20b). Differentiating these constraints $\mathbf{0} = \boldsymbol{g}(\boldsymbol{y}(t), t)$ twice the acceleration constraints (21b) are obtained that define together with $\ddot{\boldsymbol{y}}(t) = \boldsymbol{M}^{-1}(\boldsymbol{y})(\boldsymbol{f} - \boldsymbol{G}^\top \boldsymbol{\lambda})$, see (20a), the system of linear equations

$$\mathbf{0} = -[\boldsymbol{G}\boldsymbol{M}^{-1}\boldsymbol{G}^\top](\boldsymbol{y}, t)\,\boldsymbol{\lambda} + [\boldsymbol{G}\boldsymbol{M}^{-1}\boldsymbol{f}](\boldsymbol{y}, \dot{\boldsymbol{y}}, t) + \boldsymbol{g}^{\boldsymbol{(II)}}(\boldsymbol{y}, \dot{\boldsymbol{y}}, t) \qquad (33)$$

for the Lagrangian multipliers $\boldsymbol{\lambda}(t)$. The $r = 2$ differentiation steps introduce the large error terms $\mathcal{O}(\|\ddot{\boldsymbol{\theta}}(t)\|)$ and $\mathcal{O}(\theta/h^2)$ in the analytical and numerical solution, respectively. In DAE terminology the system (20) forms a DAE of index $r + 1 = 3$ that is called the *index-3 formulation* of the equations of motion [22].

For consistent initial values satisfying (22) the solution of the equations of motion (20) remains unchanged if the constraints (20b) on position level are substituted by the constraints (21a) on velocity level since

$$\boldsymbol{g}(\boldsymbol{y}(t), t) = \underbrace{\boldsymbol{g}(\boldsymbol{y}_0, t_0)}_{=0,\ \text{see (22)}} + \int_{t_0}^{t} \underbrace{\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{g}(\boldsymbol{y}(\tau), \tau)}_{=0,\ \text{see (21a)}} \,\mathrm{d}\tau = 0\,. \qquad (34)$$

From the numerical viewpoint the *index-2 formulation* (20a)/(21a) of the equations of motion is attractive because only $r = 1$ differentiation of the velocity constraints (21a) is necessary to get the system (33) of linear equations defining $\boldsymbol{\lambda}(t)$.

With a perturbation $\boldsymbol{\theta}(t)$ in the velocity constraints (21a) the error terms for this index-2 DAE are of size $\mathcal{O}(\max_t \|\boldsymbol{\theta}(t)\|\, \|\dot{\boldsymbol{\theta}}(t)\|)$ for $\boldsymbol{y}(t)$, $\boldsymbol{v}(t)$ and of size $\mathcal{O}(\max_t \|\dot{\boldsymbol{\theta}}(t)\|)$ for $\boldsymbol{\lambda}(t)$. The error bounds for the numerical solution are $\mathcal{O}(\theta) + \mathcal{O}(\theta^2/h)$ for $\boldsymbol{y}_n$, $\boldsymbol{v}_n$ and $\mathcal{O}(\theta/h)$ for $\boldsymbol{\lambda}_n$. They are smaller by the (small) factor h than the corresponding error bounds for the original index-3 DAE (20a,b), see [37]. Applying **RADAU5** with the standard setting $\kappa = 0.03$ to the wheelset example of Fig. 2 in

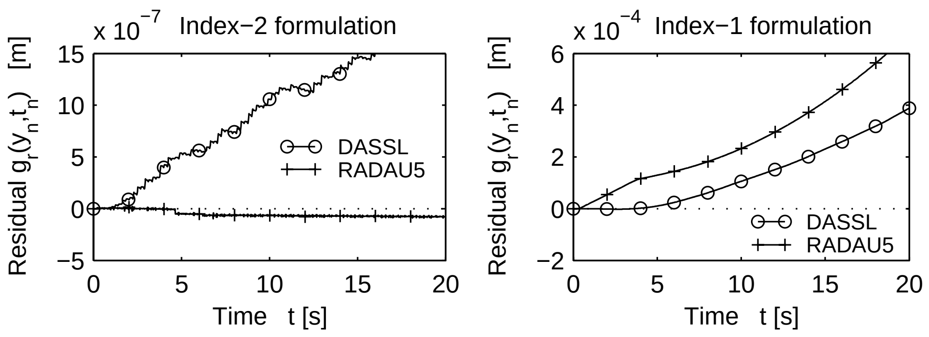

Fig. 4. Drift-off effect in the dynamical simulation of the rigid wheelset of Fig. 2 resulting in an increasing distance $g_r(\boldsymbol{y}, t)$ between the right wheel and the rail, i. e., in an increasing error in the constraints $\mathbf{0} = \boldsymbol{g} = (g_l, g_r)^\top$ that are defined by the contact conditions for left and right wheel.

index-2 formulation (20a)/(21a) the error of the numerical solution is kept well below the user-defined error tolerances TOL.

In the same way the *index-1 formulation* (20a)/(21b) of the equations of motion is obtained substituting (20b) by the constraints (21b) on acceleration level. The error bounds for $\boldsymbol{\lambda}(t)$ and $\boldsymbol{\lambda}_n$ are slightly improved to $\mathcal{O}(\max_t \|\boldsymbol{\theta}(t)\|)$ and $\mathcal{O}(\theta)$, respectively. BDF and implicit Runge–Kutta solvers applied to the index-1 formulation are as robust and efficient as in the classical ODE case.

In DAE terminology the substitution of the constraints (20b) by one of its time derivatives is called *index reduction*. Index reduction by differentiation improves the solver's robustness substantially but suffers in long-term simulations from the *drift-off effect* that is illustrated by Fig. 4.

Because of (34) the *analytical* solution of the index-2 formulation satisfies the original constraints (20b) exactly for all $t \geq t_0$. In the *numerical* solution the integrand $(\mathrm{d}\boldsymbol{g}/\mathrm{d}t)(\boldsymbol{y}(\tau), \tau)$ in (34) is still bounded by a small constant $\epsilon > 0$ but it does *not* vanish identically. Therefore, the error in (20b) may increase linearly in time t:

$$\|\boldsymbol{g}(\boldsymbol{y}_n, t_n)\| \leq \|\boldsymbol{g}(\boldsymbol{y}_0, t_0)\| + \int_{t_0}^{t_n} \epsilon \,\mathrm{d}t = \epsilon \cdot (t_n - t_0)\,; \tag{35}$$

the numerical solution $\boldsymbol{y}_n$ *drifts* off the manifold $\{\, (\boldsymbol{\eta}, t) \,:\, \boldsymbol{g}(\boldsymbol{\eta}, t) = \mathbf{0} \,\}$ that is defined by the constraints (20b) on position level. The error bound ϵ summarizes discretization and round-off errors and the iteration errors of Newton's method.

For the index-1 formulation a quadratic error growth $\|\boldsymbol{g}(\boldsymbol{y}_n, t_n)\| \leq \epsilon \cdot (t_n - t_0)^2$ has to be expected since the constraints (20b) on position level are substituted by their second derivatives (21b). Practical experience shows that the actual value of ϵ depends on the solver and on the user-defined error tolerances TOL. In general, however, there is always a linear drift in the time integration of the index-2 formulation and a quadratic drift for the index-1 formulation, see Fig. 4 for a typical example.

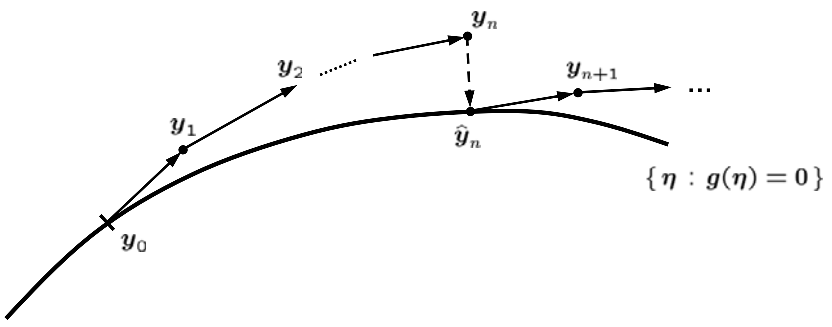

Fig. 5. Time integration with projection steps for DAEs (20) with $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{y})$.

An early attempt to avoid both the numerical problems for the index-3 formulation (20) and the drift-off effect in the index-2 and index-1 formulation goes back to the work of Baumgarte [38] who substituted the constraints (20b) by a linear combination of all three constraints (20b), (21a) and (21b). Because of the problems to select suitable coefficients for this linear combination (*Baumgarte coefficients*) the practical use of Baumgarte's approach is restricted to small scale models, see also [39].

Today, it is state-of-the-art to avoid the drift-off effect by *projection* techniques [40, 41]. During the time integration of the index-2 formulation the residual $\|\boldsymbol{g}(\boldsymbol{y}_n, t_n)\|$ in the constraints (20b) is monitored. If the residual exceeds some user-defined small error bound $\varepsilon > 0$ then $\boldsymbol{y}_n$ is projected onto the manifold $\{ \boldsymbol{\eta} : \boldsymbol{g}(\boldsymbol{\eta}, t_n) = \mathbf{0} \}$ and the time integration is continued with the projected position coordinates $\hat{\boldsymbol{y}}_n$ instead of $\boldsymbol{y}_n$, see Fig. 5.

Mathematically, the projection defines a minimization problem

$$\|\boldsymbol{\eta} - \boldsymbol{y}_n\| \to \min_{\{ \boldsymbol{\eta} : \boldsymbol{g}(\boldsymbol{\eta}, t_n) = \mathbf{0} \}} \tag{36}$$

that can be solved efficiently by simplified Newton iterations to get $\hat{\boldsymbol{y}}_n$, see [41].

For the index-1 formulation the projection of both $\boldsymbol{y}_n$ and $\boldsymbol{v}_n$ helps to avoid the drift in the constraints (20b) and (21a) on position and velocity level [40].

Gear–Gupta–Leimkuhler formulation In the complex applications of vehicle system dynamics the use of classical explicit projection methods like (36) is restricted to Runge–Kutta and other one-step methods since the efficient implementation in advanced BDF solvers with order and stepsize control is non-trivial.

Instead of implementing explicit projection steps in the solver the equations of motion (20) are reformulated in a way that contains implicitly the projection onto the constraint manifold $\{ \boldsymbol{\eta} : \boldsymbol{g}(\boldsymbol{\eta}, t_n) = \mathbf{0} \}$. The *Gear–Gupta–Leimkuhler formulation* (or *stabilized index-2 formulation*) of the equations of motion considers the constraints (20b) and (21a) on position and velocity level simultaneously [42]:

$$\begin{aligned} \dot{\boldsymbol{y}}(t) &= \boldsymbol{v} - \boldsymbol{G}^\top(\boldsymbol{y},t)\boldsymbol{\mu}\,, \\ \boldsymbol{M}(\boldsymbol{y})\,\dot{\boldsymbol{v}}(t) &= \boldsymbol{f}(\boldsymbol{y},\boldsymbol{v},t) - \boldsymbol{G}^\top(\boldsymbol{y},t)\boldsymbol{\lambda}\,, \\ \mathbf{0} &= \boldsymbol{g}(\boldsymbol{y},t)\,, \\ \mathbf{0} &= \boldsymbol{G}(\boldsymbol{y},t)\boldsymbol{v} + \boldsymbol{g}^{(\boldsymbol{I})}(\boldsymbol{y},t)\,. \end{aligned} \tag{37}$$

The increasing number of equations is compensated by a correction term $-\boldsymbol{G}^\top\boldsymbol{\mu}$ with auxiliary variables $\boldsymbol{\mu}(t) \in \mathbb{R}^{n_g}$. The correction term vanishes identically for the analytical solution ($\boldsymbol{\mu}(t) \equiv \mathbf{0}$) and remains in the size of the user-defined error tolerances TOL for the numerical solution.

Eqs. (37) form an index-2 DAE that may be solved robustly and efficiently by BDF [42] and implicit Runge–Kutta methods [21]. However, the error estimates in classical ODE solvers tend to overestimate the local errors in the algebraic components $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ of DAE (37), see [36]. Therefore the components $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ should not be considered in the automatic stepsize control of BDF solvers [43]. For implicit Runge–Kutta solvers the error estimates for $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are scaled by the small factor h_n.

The efficient evaluation of the correction term $\boldsymbol{G}^\top(\boldsymbol{y},t)\boldsymbol{\mu}$ for given $\boldsymbol{\mu},\boldsymbol{y},t$ is non-trivial since multi-body formalisms evaluate matrix-vector products $\boldsymbol{G}^\top\boldsymbol{\lambda}$ and $\boldsymbol{G}\boldsymbol{v}$ with $\mathcal{O}(N)$ complexity but do not evaluate the matrix $\boldsymbol{G}(\boldsymbol{y},t)$ itself.

The key to the efficient solution of (37) was found in the analysis of a special class of overdetermined DAEs [44] that are closely related to DAEs (37), see [45]. BDF solvers and implicit Runge–Kutta solvers for DAEs (11) approximate the Jacobian $(\alpha_{n,0}/h_n)(\partial\boldsymbol{F}/\partial\dot{\boldsymbol{x}}) + (\partial\boldsymbol{F}/\partial\boldsymbol{x})$ to compute $\boldsymbol{x}_{n+1}$ iteratively by Newton's method, see (9) and (13). In (37) this Jacobian has a 4×4 block structure reflecting the four equations in (37) and the partitioning $\boldsymbol{x} = (\boldsymbol{y},\boldsymbol{v},\boldsymbol{\lambda},\boldsymbol{\mu})^\top$. Block $(3,1)$ is an approximation of $(\partial\boldsymbol{g}/\partial\boldsymbol{y})(\boldsymbol{y},t) = \boldsymbol{G}(\boldsymbol{y},t)$, its transpose may be used to get the matrix-vector product $\boldsymbol{G}^\top\boldsymbol{\mu}$. This algorithm was implemented in the BDF solver **ODASSL** [44] that exploits furthermore the special structure of (37) to compute $\boldsymbol{\mu}_{n+1}$ more efficiently than in the standard **DASSL** implementation.

Advanced DAE models The numerical algorithms for DAE time integration that we discussed in detail for classical constrained mechanical systems with equations of motion (20) may be applied as well to the more complex hybrid differential-algebraic model equations (30)/(31) that are typical of vehicle system dynamics.

In the model equations in residual form (32) the vector $\boldsymbol{x}(t)$ is now given by $\boldsymbol{x} = (\boldsymbol{y},\boldsymbol{s},\boldsymbol{v},\boldsymbol{c},\boldsymbol{\lambda},\boldsymbol{w})^\top$. Time events and the discrete state equations (31) are handled by the methods that are known from ODE theory, see Section 2.

The differential equations (30b) and the algebraic equations (30c) and (30d) are added to the residual $\boldsymbol{F}(\boldsymbol{x},\dot{\boldsymbol{x}},t)$. As a consequence, the BDF solution $\boldsymbol{x}_{n+1}$ of (30) satisfies the non-linear equations (30d) exactly:

$$\mathbf{0} = \boldsymbol{h}(\boldsymbol{y}_{n+1}, \boldsymbol{s}_{n+1}, \boldsymbol{r}_j, t_{n+1})\,,$$

see (12). The BDF solution $\boldsymbol{s}_{n+1}$ coincides with the function value $\boldsymbol{s}(\boldsymbol{y}_{n+1}, t_{n+1})$ of the function $\boldsymbol{s}(\boldsymbol{y}, t)$ that is implicitly defined by

$$\mathbf{0} = \boldsymbol{h}(\boldsymbol{y}, \boldsymbol{s}(\boldsymbol{y}, t), \boldsymbol{r}_j, t)$$

if $\partial\boldsymbol{h}/\partial\boldsymbol{s}$ is non-singular, see (24) and (30d). BDF compute for a given geometric configuration $\boldsymbol{y}$ of the multi-body system the contact point coordinates $\boldsymbol{s}$ exactly up to the (small) iteration errors in Newton's method. The same result may be shown for the algebraic variables $\boldsymbol{w}$ in (30c).

With the regularity assumptions of Section 3 the model equations (30) form an index-3 DAE. Index reduction and projection techniques have to be applied. It is important to note that the non-linear equations (30c) and (30d) may be solved directly w. r. t. the algebraic variables $\boldsymbol{w}$ and $\boldsymbol{s}$. In contrast to the constraints (30e) the non-linear equations (30c) and (30d) do *not* imply hidden constraints on the state variables like (21).

The index reduction is based on time derivatives of the constraints $\mathbf{0} = \boldsymbol{g}(\boldsymbol{y}, \boldsymbol{s}, t)$. It does not involve any derivatives of the non-linear equations $\mathbf{0} = \boldsymbol{b}(\boldsymbol{w}, \ldots)$ and $\mathbf{0} = \boldsymbol{h}(\boldsymbol{y}, \boldsymbol{s}, t)$. The hidden constraints on the level of velocity coordinates are

$$\mathbf{0} = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{g}(\boldsymbol{y}(t), \boldsymbol{s}(t), t) = \boldsymbol{G}(\boldsymbol{y}, \boldsymbol{s}, t)\,\boldsymbol{v} + \boldsymbol{g}^{(\boldsymbol{I})}(\boldsymbol{y}, \boldsymbol{s}, t)$$

with matrix $\boldsymbol{G}$ of (27) since $\dot{\boldsymbol{s}}(t)$ is a linear function of $\dot{\boldsymbol{y}}(t) = \boldsymbol{v}(t)$ that may be obtained differentiating $\mathbf{0} = \boldsymbol{h}(\boldsymbol{y}(t), \boldsymbol{s}(t), t)$ implicitly w. r. t. t, see also (28).

Using a rather general concept of DAE index reduction [46] the Gear–Gupta–Leimkuhler formulation (37) is generalized to complex DAE models (30), see [47]. Here, the Gear–Gupta–Leimkuhler formulation (37) is extended by (30b,c,d). To compensate the simultaneous consideration of the constraints on position level and the hidden constraints on velocity level there is, as before, just one correction term. It has to be added to the kinematic equations $\dot{\boldsymbol{y}}(t) = \boldsymbol{v}$ that get the form

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{v} - \boldsymbol{G}^{\top}(\boldsymbol{y}, \boldsymbol{s}, t)\boldsymbol{\mu}$$

with the constraint matrix $\boldsymbol{G}$ of (27) and auxiliary variables $\boldsymbol{\mu}(t) \in \mathbb{R}^{n_g}$. Again, the correction term vanishes identically for the analytical solution and remains in the size of the user-defined error tolerances TOL for the numerical solution.

For this extended system the Jacobian $\boldsymbol{J}$ in (13) has a 7×7 block structure with $(\partial\boldsymbol{g}/\partial\boldsymbol{y})$, $(\partial\boldsymbol{g}/\partial\boldsymbol{s})$, $(\partial\boldsymbol{h}/\partial\boldsymbol{s})$, $(\partial\boldsymbol{h}/\partial\boldsymbol{y})$ as blocks $(6,1)$, $(6,2)$, $(5,2)$ and $(5,1)$, respectively. In a modified version of the BDF solver **ODASSL** these blocks are used for the efficient evaluation of the correction term $\boldsymbol{G}^{\top}\boldsymbol{\mu}$ with matrix $\boldsymbol{G}$ of (27), see [48].

Summary The robust and efficient dynamical simulation in vehicle system dynamics may be based on DAE time integration methods with **DASSL** like BDF solvers that are applied to the Gear–Gupta–Leimkuhler formulation of the model equations. The approach is used successfully in industrial multi-body system simulation packages [4].

5. ALGORITHMS AND TOOLS FOR THE SIMULATION OF MULTI-PHYSICAL PROBLEMS

Typical applications of vehicle system dynamics are far beyond the field of classical multi-body analysis. Nevertheless, the methods and software tools of multi-body dynamics have been used very successfully as integration platform for large, complex models in vehicle system dynamics including mechanical, hydraulic and electronic components.

Alternatives are general purpose simulation tools like Simulink and modelling languages for complex physical systems like ACSL or Modelica[1]. Furthermore, the coupling of two or more mono-disciplinary simulation tools from different fields of application has been proved to be very useful in industrial applications (*simulator coupling* or *co-simulation*).

Extensions of multi-body system simulation packages In the early days of industrial multi-body system simulation the focus was on specialized force elements with or without inner state variables $\boldsymbol{c}$ and $\boldsymbol{r}_j$, see Section 3 and [3].

Today, it might be even more important to embed the dynamical simulation of the mechanical system components efficiently in process chains for the simulation of the overall behaviour of engineering systems. Fig. 6 shows typical bi-directional interfaces between specialised simulation tools. The interfaces are used for data exchange in the pre- and post-processing of dynamical simulation.

The simulation of flexible multi-body systems relies on the interfaces to finite element (FE) tools of structural dynamics. The interface from FE tool to multi-body system tool provides mass, damping and stiffness data for the pre-processing of modes $\vec{w}^{(i)}$ in the Ritz ansatz (15), see [29]. The interface from multi-body system tool to FE tool supports the post-processing of multi-body system simulation data. Load vectors are transferred to the fatigue analysis in standard FE tools [50].

Modal reduction techniques are also used successfully in the analysis of certain aerodynamic effects. For this *close coupling* of computational fluid dynamics (CFD) and (flexible) multi-body dynamics the classical interfaces between FE tools and multi-body system tools have been extended by an interface to CFD tools [16].

[1]Matlab, Simulink and Real-Time Workshop are trademarks of The MathWorks, Inc., ACSL is a trademark of The AEgis Technologies Group, Inc., and Modelica is a trademark of the Modelica Association.

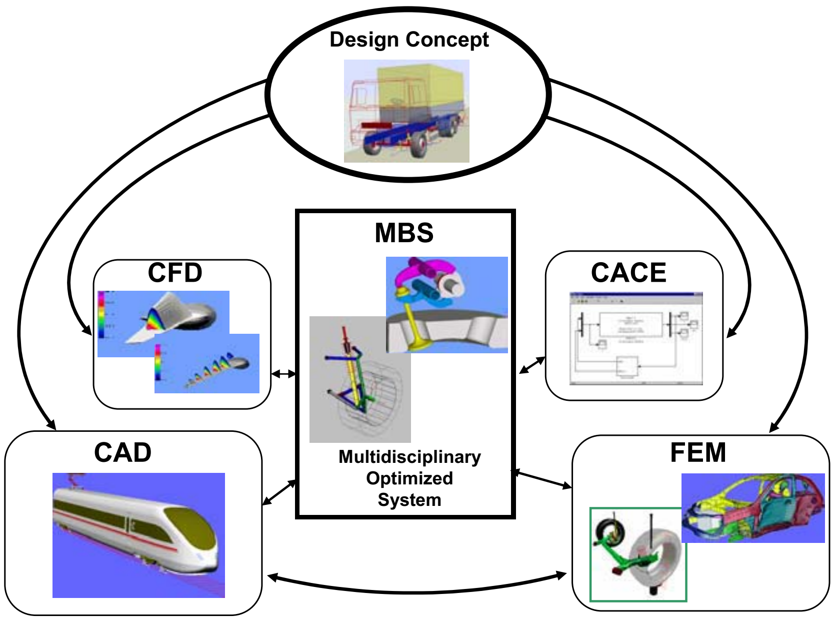

Fig. 6. Integrated virtual design using multi-body system simulation software and its bi-directional interfaces, see [49].

The interfaces between CAD and multi-body system tools support an efficient and fail-safe model setup using the geometrical data of CAD models. The opposite interface transfers all relevant geometrical modifications of the multi-body system model automatically back to the CAD model [51].

Controller synthesis is a central problem in many applications of vehicle system dynamics. Close links between the methods and tools for the dynamical simulation of mechanical components and the tools of Computer aided control engineering (CACE) are therefore essential. Fig. 7 summarizes a number of typical interfaces between multi-body system software and CACE tools.

Already in the early nineties the increasing complexity of controllers in mechatronic system components motivated the integration of multi-body system models in CACE tools starting with the export of the system matrices $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ describing the linearized equations of motion, see (5). Later, this interface was extended by the export of FORTRAN source code for non-linear equations of motion (2). In both cases the time integration for the full system including multi-body system model and controller is performed in the CACE tool.

Since the solvers of CACE tools are not tailored to the differential-algebraic model equations (30) it is often more attractive to perform the time integration of the full system including the controller in the multi-body system tool. State-of-the-art CACE tools support this approach by a code export interface. A typical example is the Real-

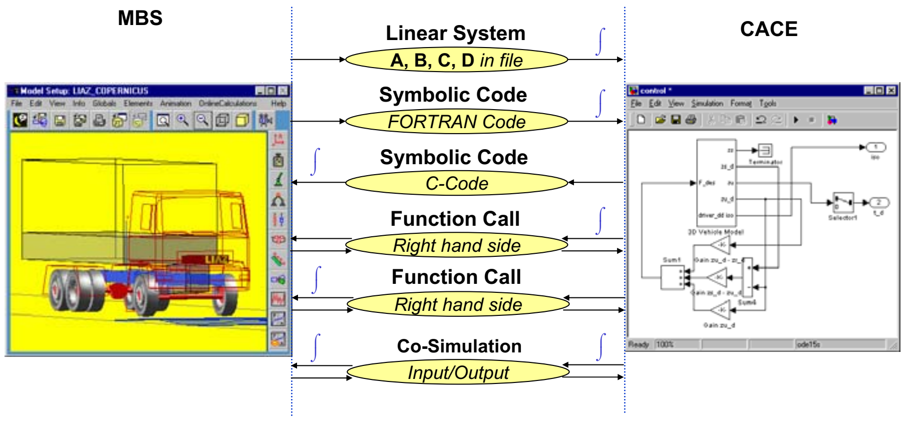

Fig. 7. Typical interfaces between multi-body system tools and CACE tools [52].

Time Workshop of The MathWorks, Inc., that generates and executes stand-alone C code for developing and testing algorithms modelled in Simulink.

As an alternative to code export the tools of multi-body system analysis and CACE may be coupled as well by a function call interface that evaluates during time integration repeatedly the right hand sides of the model equations for various given input data, see [52] for a more detailed discussion. The last row of Fig. 7 refers to the co-simulation interface that will be discussed below.

General purpose simulation tools and modelling languages Using interfaces between well developed mono-disciplinary simulation tools the experienced engineer may study multi-physical problems without leaving his usual software environment. Latest new developments of the powerful specialized tools are immediately available and the access to highly developed specialized solvers is straightforward.

Alternative approaches resolutely focus on the multi-domain aspect of complex multi-physical applications. The most widespread general purpose simulation tool from system theory is Simulink that has been developed as platform for multi-domain simulation and model-based design of dynamic systems. It has a block-diagram user interface for model setup and a number of toolboxes for special fields of application like SimMechanics and SimDriveline.

The different concepts behind specialized multi-body system tools on the one hand and general purpose simulation tools from system theory on the other is illustrated by the screenshots of user interfaces in Fig. 7.

The increasing complexity of industrial multi-physical applications results in a revival of modelling languages that are designed to bundle the experience of specialists from various engineering domains. Modern modelling languages like Modelica [53] are object oriented and do not follow the classical input–output scheme of traditional block-oriented tools (*non-causal* modelling).

Instead of classical ODE model equations the model equations get the form of differential-algebraic equations. Modelica is a rapidly developing non-proprietary modelling language that covers a wide range of applications like electrical circuits, vehicle dynamics, hydraulic and pneumatic systems and automotive powertrains, see [8] and the actual material at the Modelica website [53] for more details of this ongoing development.

Simulator coupling Specialized mono-disciplinary tools and general purpose tools follow completely different strategies for the setup of multi-physical models and for the formulation of model equations. They have, however, in common that the full set of model equations is solved numerically in *one* simulation tool by *one* solver.

The modular structure of coupled multi-physical problems may also be exploited explicitly by coupling two or more well-established mono-disciplinary tools for model setup *and* for time integration (*simulator coupling* or *co-simulation*). In this way the subsystems are handled by *different* solvers and each solver is tailored to the corresponding subsystem.

The communication between subsystems is restricted to discrete synchronization points T_j. For each subsystem all necessary information from other subsystems has to be provided by interpolation or — if data for interpolation have not yet been computed — by extrapolation from $t \leq T_j$ to the actual *macro step* $T_j \to T_{j+1}$.

From the viewpoint of a multi-body system tool the coupling variables to other simulation tools may be considered as a special type of discrete variables $\boldsymbol{r}_j$ in (31): In the multi-body system tool the values of the coupling variables $\boldsymbol{r}_j$ are kept constant during the whole macro step $T_j \to T_{j+1}$. For all other simulation tools in the co-simulation environment the update formula (31) involves the time integration from the synchronization point $t = T_j$ to the synchronization point $t = T_{j+1}$ to get $\boldsymbol{r}_{j+1} = \boldsymbol{k}(\ldots)$.

As a typical example of simulator coupling Fig. 8 shows simulation results for a heavy duty truck with semi-active suspensions [54]. The truck is modelled as a multi-body system with 41 bodies and 64 (mechanical) degrees of freedom. There are semi-active dampers in the axle suspension and in the cabin suspension that are controlled by an extended groundhook concept and by the skyhook law, respectively. Both controllers are modelled in Simulink and should be optimized to minimize dynamic road-tyre forces and to maximize comfort.

In this application the code export interfaces, see Fig. 7, are not attractive since the multi-body system model contains closed loops resulting in constraints (30e) that can not be handled by the standard solvers of Simulink. On the other hand structure and parameters of the controllers are modified during the optimization process. It is therefore much more convenient to keep the controllers in Simulink instead of exporting their C code.

In [54], the problem was solved successfully by co-simulation of Simulink and the multi-body system tool SIMPACK [4]. The SIMPACK solver numerically integrates

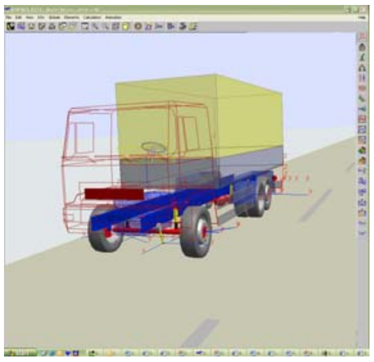
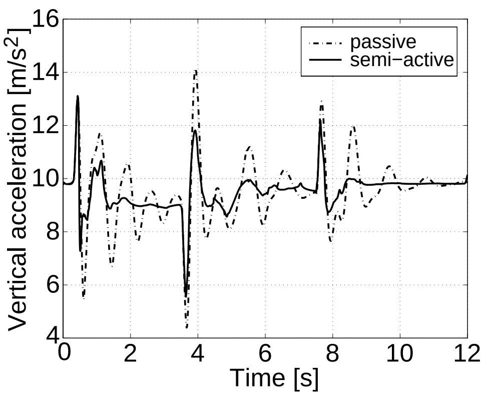

Fig. 8. Semi-active truck suspensions: Vertical acceleration of the cabin (ISO filtered) [54].

the mechanical part of the model, the control part is integrated in Simulink. The co-simulation sampling frequency is set to 200 Hz resulting in a constant *macro stepsize* $H = T_{j+1} - T_j = 5.0$ ms.

SIMPACK and Simulink run in two parallel processes. At the synchronization points T_j the controller outputs, i. e., the electrical signals defining the damper characteristics, are sent from Simulink to SIMPACK using inter process communication (IPC). By the same IPC interface Simulink gets the actual vertical position, velocity and acceleration of truck chassis and cabin as controller input from SIMPACK. When the data exchange is completed the solvers of Simulink and SIMPACK separately perform the macro step $T_j \rightarrow T_{j+1}$. Co-simulation continues with the data exchange at synchronization point T_{j+1} and so on.

The robust and reliable co-simulation contributed to the successful optimization of the controller design. A comparison of a truck with passive dampers and a truck with optimized semi-active dampers shows that the ride comfort is improved substantially, see the right plot of Fig. 8 for the simulation of a typical driving maneuver [54].

This case study illustrates that co-simulation techniques allow the convenient simulation and optimization of coupled problems in a classical software environment. Practical experience shows, however, that the coupling of different solvers may occasionally result in numerical instability. Furthermore, interpolation and extrapolation introduce additional discretization errors. The macro stepsize H has to be selected carefully. In typical standard applications stability and accuracy may be guaranteed if H is in the range between 0.1 ms and 10 ms.

For certain classes of coupled problems the instability phenomenon has been analysed in great detail. Several modifications of the co-simulation techniques help to improve their stability, accuracy and robustness also for larger macro stepsizes [55, 56].

Summary Complex multi-physical applications in vehicle system dynamics are beyond the field of multi-body dynamics. Successful simulation strategies use data interfaces or co-simulation interfaces between mono-disciplinary tools or models that are generated by general purpose simulation tools or modelling languages.

ACKNOWLEDGEMENTS

The author acknowledges gratefully the inspiring working atmosphere in late Professor W. Kortüm's Vehicle System Dynamics group at DLR German Aerospace Center and in the SIMPACK development team (1997–2002). Furthermore, he is indebted to C. Führer (Lund) for introducing him during several years to the topic of the present paper and to P. Rentrop (Munich) for his permanent interest in the author's scientific work.

REFERENCES

1. W. Kortüm and W. Schiehlen. General purpose vehicle system dynamics software based on multibody formalisms. *Vehicle System Dynamics*, 14:229–263, 1985.
2. W. Kortüm and P. Lugner. *Systemdynamik und Regelung von Fahrzeugen*. Springer–Verlag, Berlin Heidelberg New York, 1994.
3. W. Kortüm, W.O. Schiehlen, and M. Arnold. Software tools: From multibody system analysis to vehicle system dynamics. In H. Aref and J.W. Phillips, editors, *Mechanics for a New Millennium*, pages 225–238, Dordrecht, 2001. Kluwer Academic Publishers.
4. W. Rulka. *Effiziente Simulation der Dynamik mechatronischer Systeme für industrielle Anwendungen*. PhD thesis, Vienna University of Technology, Department of Mechanical Engineering, 1998.
5. E. Eich-Soellner and C. Führer. *Numerical Methods in Multibody Dynamics*. Teubner–Verlag, Stuttgart, 1998.
6. B. Simeon, C. Führer, and P. Rentrop. Differential-algebraic equations in vehicle system dynamics. *Surveys on Mathematics for Industry*, 1:1–37, 1991.
7. A.A. Shabana. *Computational Dynamics*. John Wiley & Sons, Inc., New York, 2nd edition, 2001.
8. M.M. Tiller. *Introduction to Physical Modeling with Modelica*. Kluwer Academic Publishers, Boston Dordrecht London, 2001.
9. R. Kübler and W. Schiehlen. Modular simulation in multibody system dynamics. *Multibody System Dynamics*, 4:107–127, 2000.
10. W.O. Schiehlen. *Technische Dynamik*. Teubner Studienbücher, Stuttgart, 1985.
11. H. Brandl, R. Johanni, and M. Otter. A very efficient algorithm for the simulation of robots and similar multibody systems without inversion of the mass matrix. In P. Kopacek, I. Troch, and K. Desoyer, editors, *Theory of Robots*, pages 95–100, Oxford, 1988. Pergamon Press.
12. R.E. Roberson and R. Schwertassek. *Dynamics of Multibody Systems*. Springer–Verlag, Berlin Heidelberg New York, 1988.
13. A. Eichberger. *Simulation von Mehrkörpersystemen auf parallelen Rechnerarchitekturen*. Fortschritt-Berichte VDI Reihe 8, Nr. 332. VDI–Verlag, Düsseldorf, 1993.
14. C.T. Kelley. *Solving Nonlinear Equations with Newton's Method*. SIAM, Philadelphia, 2003.
15. J.J. Moré, D.C. Sorensen, B.S. Garbow, and K.E. Hillstrom. The MINPACK project. In W.J. Cowell,

editor, *Sources and Development of Mathematical Software,* pages 88–111. Prentice–Hall, Englewood Cliffs, N.J., 1984.

16. W.R. Krüger and M. Spieck. Aeroelastic effects in multibody dynamics. *Vehicle System Dynamics*, 41:383–399, 2004.
17. C.T. Kelley. *Iterative Methods for Optimization*. SIAM, Philadelphia, 1999.
18. T.F. Coleman, B.S. Garbow, and J.J. Moré. Software for estimating sparse Jacobian matrices. *ACM Transactions on Mathematical Software*, 10:329–345, 1984.
19. U. Ascher and L.R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential–Algebraic Equations*. SIAM, Philadelphia, 1998.
20. E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations. I. Nonstiff Problems*. Springer–Verlag, Berlin Heidelberg New York, 2nd edition, 1993.
21. E. Hairer and G. Wanner. *Solving Ordinary Differential Equations. II. Stiff and Differential-Algebraic Problems*. Springer–Verlag, Berlin Heidelberg New York, 2nd edition, 1996.
22. K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical solution of initial–value problems in differential–algebraic equations*. SIAM, Philadelphia, 2nd edition, 1996.
23. A. Fuchs and M. Arnold. Efficient corrector iteration for implicit time integration in multibody dynamics. NATO Advanced Study Institute on Virtual Nonlinear Multibody Systems, Prague 23 June - 3 July, 2002.
24. A.A. Shabana. *Dynamics of Multibody Systems*. Cambridge University Press, Cambridge, 2nd edition, 1998.
25. O. Wallrapp. Linearized flexible multibody dynamics including geometric stiffening effects. *Mechanics of Structures and Machines*, 19:385–409, 1991.
26. R. Schwertassek and O. Wallrapp. *Dynamik flexibler Mehrkörpersysteme*. Vieweg, 1999.
27. S. Dietz. *Vibration and Fatigue Analysis of Vehicle Systems using Component Modes*. Fortschritt-Berichte VDI Reihe 12, Nr. 401. VDI–Verlag, Düsseldorf, 1999.
28. B. Simeon. *Numerische Simulation gekoppelter Systeme von partiellen und differential-algebraischen Gleichungen in der Mehrkörperdynamik*. Fortschritt-Berichte VDI Reihe 20, Nr. 325. VDI–Verlag, Düsseldorf, 2000.
29. O. Wallrapp. Standardization of flexible body modeling in multibody system codes, Part I: Definition of standard input data. *Mechanics of Structures and Machines*, 22:283–304, 1994.
30. A. Heckmann, M. Arnold, and O. Vaculín. A modal multifield approach for an extended flexible body description in multibody dynamics. *Multibody System Dynamics*, accepted for publication, 2004.
31. R. von Schwerin. *MultiBody System SIMulation – Numerical Methods, Algorithms, and Software,* volume 7 of *Lecture Notes in Computational Science and Engineering*. Springer, Berlin Heidelberg, 1999.
32. F. Pfeiffer and Ch. Glocker. *Multibody Dynamics with Unilateral Contacts*. Wiley & Sons, New York, 1996.
33. R.A. Wehage and E.J. Haug. Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems. *J. Mech. Design*, 104:247–255, 1982.
34. N. Orlandea. *Development and Application of Node–Analogous Sparsity–Oriented Methods for Simulation of Mechanical Dynamic Systems*. PhD thesis, University of Michigan, 1973.
35. M. Arnold and H. Netter. Wear profiles and the dynamical simulation of wheel-rail systems. In M. Brøns, M.P. Bendsøe, and M.P. Sørensen, editors, *Progress in Industrial Mathematics at ECMI 96*, pages 77–84. Teubner, Stuttgart, 1997.
36. L.R. Petzold. Differential/algebraic equations are not ODEs. *SIAM J. Sci. Stat. Comput.*, 3:367–384, 1982.
37. M. Arnold. A perturbation analysis for the dynamical simulation of mechanical multibody systems. *Applied Numerical Mathematics*, 18:37–56, 1995.
38. J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer*

Methods in Applied Mechanics and Engineering, 1:1–16, 1972.

39. U.M. Ascher, H. Chin, and S. Reich. Stabilization of DAEs and invariant manifolds. *Numer. Math.*, 67:131–149, 1994.
40. E. Eich. Convergence results for a coordinate projection method applied to mechanical systems with algebraic constraints. *SIAM J. Numer. Anal.*, 30:1467–1482, 1993.
41. Ch. Lubich, Ch. Engstler, U. Nowak, and U. Pöhle. Numerical integration of constrained mechanical systems using MEXX. *Mech. Struct. Mach.*, 23:473–495, 1995.
42. C.W. Gear, B. Leimkuhler, and G.K. Gupta. Automatic integration of Euler–Lagrange equations with constraints. *J. Comp. Appl. Math.*, 12&13:77–90, 1985.
43. L.R. Petzold and P. Lötstedt. Numerical solution of nonlinear differential equations with algebraic constraints II: practical implications. *SIAM J. Sci. Stat. Comput.*, 7:720–733, 1986.
44. C. Führer. Differential-algebraische Gleichungssysteme in mechanischen Mehrkörpersystemen. Theorie, numerische Ansätze und Anwendungen. PhD Thesis, TU München, Mathematisches Institut und Institut für Informatik, 1988.
45. C. Führer and B. Leimkuhler. Numerical solution of differential–algebraic equations for constrained mechanical motion. *Numer. Math.*, 59:55–69, 1991.
46. P. Kunkel, V. Mehrmann, W. Rath, and J. Weickert. GELDA: A software package for the solution of general linear differential algebraic equations. *SIAM J. Sci. Comp.*, 18:115–138, 1997.
47. M. Arnold, V. Mehrmann, and A. Steinbrecher. Index reduction of linear equations of motion in industrial multibody system simulation. Submitted for publication, 2004.
48. M. Arnold. Numerical problems in the dynamical simulation of wheel-rail systems. *Z. Angew. Math. Mech.*, Proceedings of ICIAM 95, Issue 3:151–154, 1996.
49. A. Veitl. *Integrierter Entwurf innovativer Stromabnehmer*. Fortschritt-Berichte VDI Reihe 12, Nr. 449. VDI–Verlag, Düsseldorf, 2001.
50. S. Dietz, H. Netter, and D. Sachau. Fatigue life prediction by coupling finite-element and multibody systems calculations. In *Proceedings of DETC'97, ASME Design Engineering Technical Conferences DETC/VIB–4229*, Sacramento, California, 1997.
51. W. Trautenberg. *Bidirektionale Kopplung zwischen CAD und Mehrkörpersimulationssystemen*. PhD thesis, Munich University of Technology, Department of Mechanical Engineering, 1999.
52. O. Vaculín, M. Valášek, and W.R. Krüger. Overview of coupling of multibody and control engineering tools. *Vehicle System Dynamics*, 41:415–429, 2004.
53. Modeling of Complex Physical Systems. Website of Modelica and the Modelica Association. http://www.modelica.org.
54. O. Vaculín, M. Valášek, and W. Kortüm. Multi-objective semi-active truck suspension by spatial decomposition. In H. True, editor, *Proc. of the 17th IAVSD Symposium on The Dynamics of Vehicles on Roads and on Tracks*, pages 432–440. Supplement to Vehicle System Dynamics, Vol. 37, Swets & Zeitlinger B.V., 2003.
55. R. Kübler and W. Schiehlen. Two methods of simulator coupling. *Mathematical and Computer Modelling of Dynamical Systems*, 6:93–113, 2000.
56. M. Arnold and M. Günther. Preconditioned dynamic iteration for coupled differential-algebraic systems. *BIT Numerical Mathematics*, 41:1–25, 2001.