TECHNISCHE MECHANIK, Band 16, Heft 2, (2006), 92-105
Manuskripteingang: 17. Mai 2006

Algorithm for Component Based Simulation of Multibody Dynamics

D. Vlasenko, R. Kasper

This paper presents the complete description of a non-iterative algorithm for the distributed component based simulation of the dynamics of multibodies. The simulation of well-partitioned systems has complexity O(N), where N is the total number of bodies. The algorithm was implemented in the Virtual Systems Developer (VSD) software. In contrast to other tools, this one maintains the object-oriented approach during the design and the simulation of models. Moreover an interface between VSD and a CAD tool like Autodesk Inventor was developed, where a 3D-car system was modeled. The simulation results of the car dynamics were verified in Simpack software.

1 Introduction

Simulation of multibodies has grown rapidly in recent years since engineers need to analyze increasingly complex mechanical systems. The dynamic simulation of constrained multibody systems is essential in robotics, biomechanics, vehicle and machinery design etc. The development and tests of computer models instead of real prototypes has the following significant advantages: decreased development cost, decreased development time, and increased quality of the product.

A multibody system consists of a finite number of rigid or flexible bodies which are connected by coupling elements like joints, springs, dampers and actuators. The coupling elements are assumed to be massless, so that the mass of the mechanical system is concentrated in the bodies. There is a large quantity of software by which the dynamics of multibodies can be simulated: ADAMS, SIMPACK, MEDYNA etc.

Trying to improve their usability and flexibility, modern simulation tools use the object-oriented method. In this approach a multibody system is represented as a hierarchy of submodels, connected by joints, springs and dampers. This approach significantly reduces the cost and development time of software, increasing reusability and abstraction, as was shown by Cellier (1996).

However, nowadays the object-oriented approach is used only on the design level. After a design engineer has developed a model, the simulation software generates equations of motion, describing the dynamics of the complete system. The problem is that in these equations all information about modularization is lost since the hierarchy of submodels is destroyed and the system is considered as a set of bodies connected by joints, springs and dampers. The simulation, based on these equations, is performed in a non-modular way.

In the last years we have developed and implemented the method, performing the object-oriented simulation of mechanical systems with holonomic constraints. The most detailed description of the method was written by Vlasenko (2006). In our method the model's partition, defined during the model's design, remains unchanged during the simulation, i.e. we use the simulation based on the hierarchy of subsystems.

The main advantages of the simulation on the basis of subsystems are:

- The subsystems can be modeled, tested and compiled independently. Then they can be used in a way similar to software components that encapsulate their internal structure and can be connected via interfaces. This significantly decreases the time and cost of the models' development and test. Redesign and reuse of components is more effective and easier.

- The commercially classified information of submodels is protected. A submodel works like a "black box" that has to provide only the strictly determined set of information via its interfaces. The submodel's internal data: parameters of constraints, forces, masses of internal bodies, etc. are unknown to the users of submodels. Therefore, in the future producers of mechanical (or, more commonly, mechatronic) components (e.g. motors, piezo-actuators) can provide the virtual submodels to a design office. And the design office can test and choose suitable components, using only the numerical simulation, without buying and developing real mechanical systems and components.

- Critical effects like Coulomb friction, backslash etc. can be encapsulated inside a subsystem.

- The simulation of big well-partitioned models costs *O(N)* numerical operations, where *N* denotes the total number of bodies in the simulation model.

- Subsystems are ideal candidates for the partitioning of systems on multiple processors.

In this paper we improve the previous version of the algorithm, described by Kasper, Vlasenko (2004). This time we do not use generalized coordinates and perform the simulation using only absolute coordinates. The advantages of this approach are:

- The stabilization of absolute coordinates and velocities can be distributed in a manner similar to the calculation of absolute coordinates. Therefore, the simulation process is completely distributed.

- The development of models is much easier and faster. During the modeling it is not necessary to partition the coordinates into dependent and independent ones, to define loop-closing constraints in a special manner, to set the order of bodies, choosing for each joint a basic and a dependent body. Models can be defined in standard CAD-tools and translated in the simulation software without any redesign. This approach minimizes development cost of the models, and the reusability of submodels.

We implemented the method and created an object-oriented software Virtual Systems Developer (VSD) simulating the dynamics of multibodies. The wide set of objects, describing different types of constraints and forces, was implemented in VSD: revolute joint, ball joint, stiff connection, gravity force, torque, springs etc. Trying to minimize the development time of models, also an interface between VSD and a CAD-like tool Autodesk Inventor was developed, by which mechanical systems can be modeled.

In this paper we show the simulation of a simplified car model in VSD. The comparison of our results with results obtained using the Simpack software, shows that VSD calculates the dynamics of multibodies correctly and accurately.

2 Description of Method

The basic idea of the method, shown in Figure 1, is to perform the simulation of mechanical systems using the hierarchy of submodels that constitute the complete system.

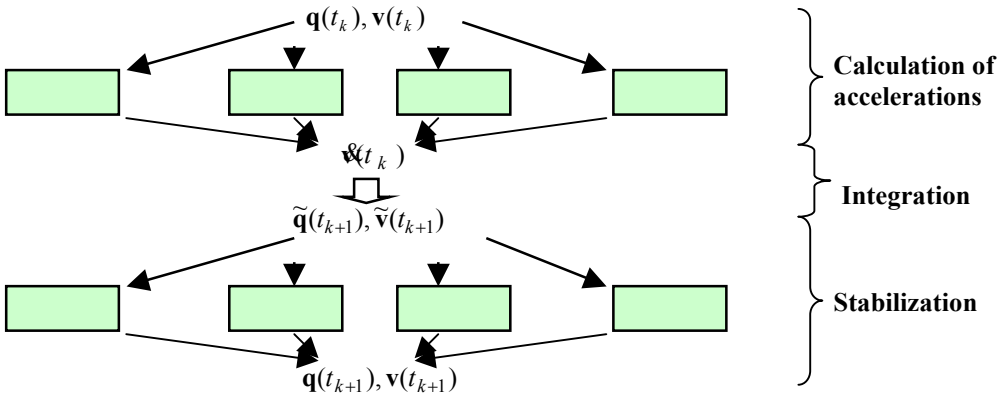

Figure 1. Simulation steps

Submodels of the first level in generally consist of connected bodies. Submodels of the next levels (called *children*) consist, without loss of generality, of connected submodels (called *parents*). Since the overwhelming majority of calculations proceeds inside of the submodels, it follows that the simulation can be distributed easily on several processors. During the simulation on each time step the following tasks have to be performed:

1. Distributed calculation of the absolute accelerations $\ddot{\mathbf{q}}(t_k)$.

2. Calculation of the absolute coordinates and velocities on the next time step. Using a favorite ODE integration scheme (e.g. Runge-Kutta or some multistep method), the value of the absolute coordinates $\tilde{\mathbf{q}}(t_{k+1})$ and velocities $\tilde{\mathbf{v}}(t_{k+1})$ on the new time step can be obtained.

3. Distributed stabilization of coordinates $\mathbf{q}(t_{k+1})$ and velocities $\mathbf{v}(t_{k+1})$.

3 Main Idea of the Distributed Calculation of Acceleration

The calculation of acceleration consists of three steps, shown in Figure 2. As it was shown by Vlasenko (2006), for large well-partitioned models this method costs *O(N)* numerical operations, where *N* denotes the total number of bodies in the simulation model.

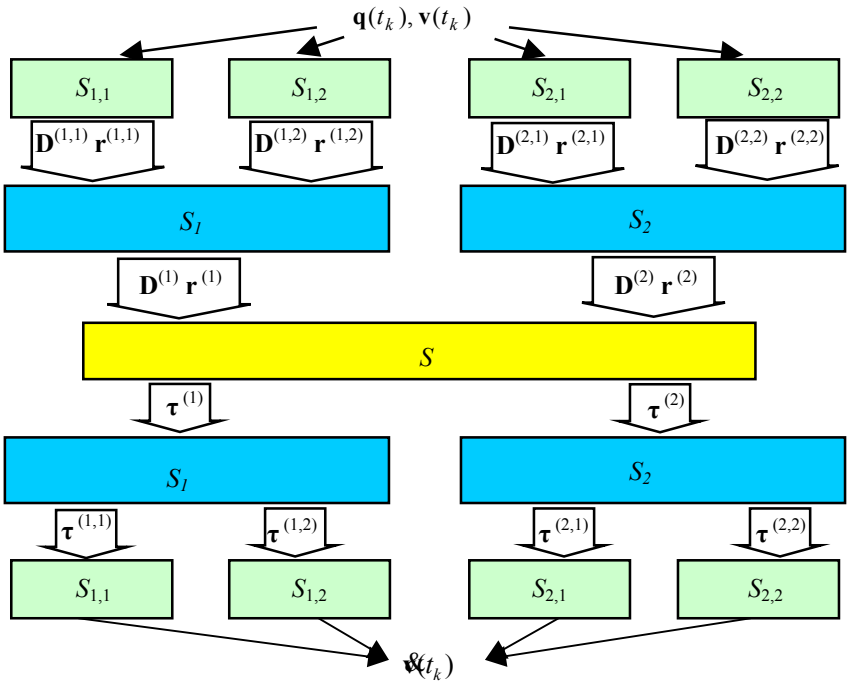

Figure 2. Calculation of acceleration steps

3.1 Hierarchical Generations of Equations of Motion

Each subsystem gets from its parents matrix $\mathbf{D}^{(i)}$ and vector $\mathbf{r}^{(i)}$ which describe the linear dependency of $\ddot{\mathbf{q}}_c^{(i)}$ on $\boldsymbol{\tau}^{(i)}$:

$$\ddot{\mathbf{q}}_c^{(i)} = \mathbf{D}^{(i)}\boldsymbol{\tau}^{(i)} + \mathbf{r}^{(i)} \tag{1}$$

where

$\ddot{\mathbf{q}}_c^{(i)}$ is the vector of accelerations of the *i*-th parent's bordering bodies,

$\boldsymbol{\tau}^{(i)}$ is the vector of forces acting in the *i*-th parent's external links.

For example, in Figure 2 the subsystem S_1 gets $\mathbf{D}^{(1,1)}$, $\mathbf{r}^{(1,1)}$, $\mathbf{D}^{(2,1)}$, $\mathbf{r}^{(2,1)}$ from its parents $S_{1,1}$, $S_{1,2}$ etc. Here a body is called *bordering* a subsystem, if it has constraints in the subsystem and is connected with external joints. A body is called *internal* to a subsystem, if it has constraints in the subsystem and is not connected to any external joint. The subsystem generates the matrix **D** and the vector **r** using the equation of constraints connecting the parents. Here **D** and **r** describe the linear dependency of $\dot{\mathbf{v}}_e$ on $\boldsymbol{\tau}$:

$$\dot{\mathbf{v}}_e = \mathbf{D}\boldsymbol{\tau} + \mathbf{r} \tag{2}$$

where

$\dot{\mathbf{v}}_e$ is the vector of accelerations of subsystem's bordering bodies,

$\boldsymbol{\tau}$ is the vector of forces in subsystem's external links.

Then the subsystem transmits **D** and **r** to its child.

3.2 Calculation of Forces on the Top of the Hierarchy

The subsystem of the highest level *S* calculates the vector of forces **t** acting in the constraints connecting its parents. Let $\boldsymbol{\tau}^{(i)}$ denote the vector of forces acting in external links of S_i, where S_i is the *i*-th parent of *S*. It is clear that $\boldsymbol{\tau}^{(i)}$ is a part of the vector **t.** In our algorithm *S* calculates $\boldsymbol{\tau}^{(i)}$ and transmits it to S_i (e.g. in Figure 2 the subsystem *S* transmits $\boldsymbol{\tau}^{(1)}$, $\boldsymbol{\tau}^{(2)}$ to its parents S_1, S_2 correspondingly).

3.3 Backward Hierarchical Calculation of the Absolute Accelerations

Subsequently, each subsystem gets the current value of $\boldsymbol{\tau}$ from its child (e.g. in Figure 2 the subsystem $S_{1,2}$ gets $\boldsymbol{\tau}^{(2)}$ from its child S_1). Then for each parent S_i the subsystem calculates $\boldsymbol{\tau}^{(i)}$ and transmits it to the parent. Subsystems of the lowest level of the hierarchy calculate the absolute accelerations of its bodies.

Finally, the absolute accelerations of all simulating bodies are obtained.

4 Equations of the Distributed Calculation of the Acceleration

4.1 Newton-Euler Equations of Motion

Equations of motion describing the dynamics of constrained multibody system, consisting of *n* bodies, in absolute coordinates can be written

$$\dot{\mathbf{q}} = \mathbf{T}(\mathbf{q})\mathbf{v} \tag{3}$$

$$\begin{pmatrix} \mathbf{M} & \mathbf{G}(\mathbf{q})^T \\ \mathbf{G}(\mathbf{q}) & 0 \end{pmatrix} \begin{pmatrix} \dot{\mathbf{v}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{f}(\mathbf{q}) \\ \mathbf{u}(\mathbf{q}, \mathbf{v}) \end{pmatrix} \tag{4}$$

$$\mathbf{g}(\mathbf{q}) = 0 \tag{5}$$

$$\|\boldsymbol{\theta}_k\| = 1 \qquad k = 1...n \tag{6}$$

where

$\mathbf{q} = \left(\mathbf{x}_1^T \quad \boldsymbol{\theta}_1^T \quad \mathrm{K} \quad \mathbf{x}_n^T \quad \boldsymbol{\theta}_n^T\right)^T$ is the vector of position variables, consisting of Cartesian position variables $\mathbf{x}_k$ and Euler parameters of body centroidal reference frames $\boldsymbol{\theta}_k$,

$\mathbf{v} = \left(\dot{\mathbf{r}}_1^T \quad \boldsymbol{\Omega}_1^T \quad \mathrm{K} \quad \dot{\mathbf{r}}_n^T \quad \boldsymbol{\Omega}_n^T\right)^T$ is the vector of velocity variables, consisting of absolute velocities $\dot{\mathbf{r}}_k$ and global angular velocities $\boldsymbol{\Omega}_k$,

$\mathbf{T}(\mathbf{q})$ is the block diagonal matrix describing the relation between the coordinates and velocities,

$\mathbf{f}$ is the vector of external forces (other than constraint forces),

$\mathbf{g}$ is the vector of (holonomic) constraints,

$\mathbf{M}$ is the mass matrix,

$\boldsymbol{\lambda}$ is the vector of Lagrange multipliers,

$\mathbf{G} = \left(\frac{\partial \mathbf{g}}{\partial \mathbf{q}}\right)\mathbf{T}$ is the matrix proportional to the constraint Jacobian,

$\mathbf{u} = -\dot{\mathbf{G}} \cdot \mathbf{v}$,

$\|\boldsymbol{\theta}_k\| = 1$ is the normalization condition on the Euler parameters of the bodies.

Readers interested in details of this formulation of the Newton-Euler equations of motion are referred to Chin (1995) and Shabana (2001) e.g..

In the future we call **G** the *transformed Jacobian*. Let us show how we can transform these equations, describing the dynamics of global multibody system, to the systems of equations, describing the dynamics of subsystems.

4.2 Equations of Motion of a Basic Subsystem

Consider a *basic* subsystem *S* (i.e. the subsystem of the lowest level of the hierarchy), shown in Figure 3, included in a complete simulating system. By *n* we denote the number of bodies in *S*. Let **g** denote the *c*-vector of equations of internal constraints:

$$\mathbf{g} = \left(g_1(\mathbf{q_S}) \quad \mathrm{K} \quad g_c(\mathbf{q_S})\right)^T = \left(0 \quad \mathrm{K} \quad 0\right)^T \tag{7}$$

where $\mathbf{q_S}$ is the $7n$-vector of the absolute coordinates of the subsystem's bodies.

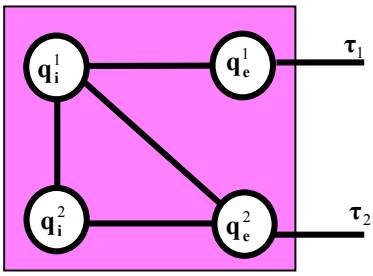

Figure 3. A subsystem of several connected bodies

Let $\boldsymbol{\tau}$ be the vector of Lagrange forces acting in external constraints. Suppose that first *m* bodies are connected with the complete system by external joints, i.e. first *m* bodies are bordering. Let $\mathbf{q_e}$ denote the $7m$-vector of absolute coordinates of bordering bodies. Let $\mathbf{q_i}$ denote the $7(n\text{-}m)$-vector of absolute coordinates of internal bodies. Obviously, $\mathbf{q_S}$ can be written as:

$$\mathbf{q_S} = \left(\mathbf{q_e}^T \quad \mathbf{q_i}^T\right)^T \tag{8}$$

We partition the transformed Jacobian $\mathbf{G} = \left(\frac{\partial \mathbf{g}}{\partial \mathbf{q_S}} \right) \mathbf{T}$ into parts

$$\mathbf{G} = \begin{pmatrix} \mathbf{G_e} & \mathbf{G_i} \end{pmatrix} = \begin{pmatrix} \frac{\partial \mathbf{g}}{\partial \mathbf{q_e}} \mathbf{T_e} & \frac{\partial \mathbf{g}}{\partial \mathbf{q}_{\mathbf{i}}} \mathbf{T}_{\mathbf{i}} \end{pmatrix} \tag{9}$$

Then a part of (4), corresponding to the subsystem *S*, is

$$\begin{pmatrix} \mathbf{M_e} & \mathbf{0} & \mathbf{G_e}^T \\ \mathbf{0} & \mathbf{M_i} & \mathbf{G_i}^T \\ \mathbf{G_e} & \mathbf{G_i} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \dot{\mathbf{v}}_{\mathbf{e}} \\ \dot{\mathbf{v}}_{\mathbf{i}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{f_e} + \boldsymbol{\tau} \\ \mathbf{f_i} \\ \mathbf{u} \end{pmatrix} \tag{10}$$

where

$\mathbf{M_e} = diag(\mathbf{M}_1,...,\mathbf{M}_m)$ is the mass matrix of bordering bodies

$\mathbf{M_i} = diag(\mathbf{M}_{m+1},...,\mathbf{M}_n)$ is the mass matrix of internal bodies

$\mathbf{f_e} = \begin{pmatrix} \mathbf{f}_1^T & \Lambda & \mathbf{f}_m^T \end{pmatrix}^T$ is the vector external forces acting on bordering bodies

$\mathbf{f_i} = \begin{pmatrix} \mathbf{f}_{m+1}^T & \Lambda & \mathbf{f}_n^T \end{pmatrix}^T$ is the vector external forces acting on internal bodies

$\mathbf{u} = -\dot{\mathbf{G}} \cdot \mathbf{v}$

This can be written as a system of equations

$$\dot{\mathbf{v}}_{\mathbf{e}} = \mathbf{M_e}^{-1}(\mathbf{f_e} + \boldsymbol{\tau} - \mathbf{G_e}^T \boldsymbol{\lambda}) \tag{11}$$

$$\dot{\mathbf{v}}_{\mathbf{i}} = \mathbf{M_i}^{-1}(\mathbf{f_i} - \mathbf{G_i}^T \boldsymbol{\lambda}) \tag{12}$$

$$\mathbf{G_e}\dot{\mathbf{v}}_{\mathbf{e}} + \mathbf{G_i}\dot{\mathbf{v}}_{\mathbf{i}} = \mathbf{u} \tag{13}$$

Substituting $\dot{\mathbf{v}}_{\mathbf{e}}$, $\dot{\mathbf{v}}_{\mathbf{i}}$ from (11) and (12) in (13), we obtain

$$\mathbf{G_e}\mathbf{M_e}^{-1}(\mathbf{f_e} + \boldsymbol{\tau} - \mathbf{G_e}^T \boldsymbol{\lambda}) + \mathbf{G_i}\mathbf{M_i}^{-1}(\mathbf{f_i} - \mathbf{G_i}^T \boldsymbol{\lambda}) = \mathbf{u} \tag{14}$$

or, in the other form

$$\mathbf{G}\mathbf{M}^{-1}\mathbf{G}^T \boldsymbol{\lambda} = \mathbf{G}\mathbf{M}^{-1}\mathbf{f} + \mathbf{G_e}^T \mathbf{M_e}^{-1} \boldsymbol{\tau} - \mathbf{u} \tag{15}$$

We can find the dependency of $\boldsymbol{\lambda}$ on $\boldsymbol{\tau}$ even when $\mathbf{G}\mathbf{M}^{-1}\mathbf{G}^T$ is singular (i.e. when **G** has dependent rows):

$$\boldsymbol{\lambda} = \mathbf{S}\boldsymbol{\tau} + \mathbf{b} \tag{16}$$

$$\mathbf{S} = (\mathbf{G}\mathbf{M}^{-1}\mathbf{G}^T)^+ \mathbf{G_e}^T \mathbf{M_e}^{-1} \tag{17}$$

$$\mathbf{b} = (\mathbf{G}\mathbf{M}^{-1}\mathbf{G}^T)^+ (\mathbf{G}\mathbf{M}^{-1}\mathbf{f} - \mathbf{u}) \tag{18}$$

where $(\mathbf{G}\mathbf{M}^{-1}\mathbf{G}^T)^+$ is a pseudoinverse of $\mathbf{G}\mathbf{M}^{-1}\mathbf{G}^T$ based on a singular value decomposition (SVD), shown by Press et al. (2002).
Substituting $\boldsymbol{\lambda}$ in (11), we obtain the relation between $\dot{\mathbf{v}}_{\mathbf{e}}$ and $\boldsymbol{\tau}$:

$$\dot{\mathbf{v}}_e = \mathbf{D}\boldsymbol{\tau} + \mathbf{r} \tag{19}$$

where

$$\mathbf{D} = \mathbf{M}_e^{-1} - \mathbf{M}_e^{-1}\mathbf{G}_e^T\mathbf{S} \tag{20}$$

$$\mathbf{r} = \mathbf{M}_e^{-1}\mathbf{f}_e - \mathbf{M}_e^{-1}\mathbf{G}_e^T\mathbf{b} \tag{21}$$

Using the equations (20) - (21), the subsystem calculates $\mathbf{D}$ and $\mathbf{r}$ and transmits them to its child.

4.3 Building Up the Hierarchy

Consider a *derived* subsystem S (i.e. a subsystem of the high level of the hierarchy) consisting of L parent subsystems: $S_1, S_2, \ldots, S_L$ shown in Figure 4.

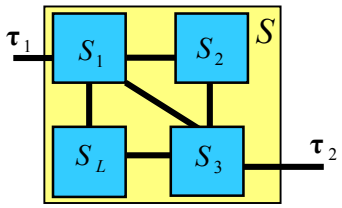

Figure 4. A subsystem consisting of several connected subsystems

Let $\mathbf{q_S}$ denote the n-vector of coordinates of bodies bordering the parents of S. Because of the definition of bordering bodies, it follows that the vector $\mathbf{q_S}$ is the union of vectors $\mathbf{q}_e^{(k)}$ (k=1..L). We can reorder the vector $\mathbf{q_S}$ as

$$\mathbf{q_S} = \begin{pmatrix} \mathbf{q}_e^{(1)} \\ \mathrm{M} \\ \mathbf{q}_e^{(L)} \end{pmatrix} \tag{22}$$

Let $\mathbf{g}$ denote the vector of equations of internal constraints between $S_1, S_2, \ldots, S_L$

$$\mathbf{g} = \begin{pmatrix} g_1(\mathbf{q_S}) & \Lambda & g_c(\mathbf{q_S}) \end{pmatrix}^T = \begin{pmatrix} 0 & \mathrm{K} & 0 \end{pmatrix}^T \tag{23}$$

By $\mathbf{G}$ we denote the transformed Jacobian

$$\mathbf{G} = \frac{\partial \mathbf{g}}{\partial \mathbf{q_S}}\mathbf{T} \tag{24}$$

Let $\boldsymbol{\lambda}$ denote the vector of the Lagrange multipliers associated with the constraints between subsystems $S_1, S_2, \ldots, S_L$. The equations of accelerations of subsystems are

$$\dot{\mathbf{v}}_S^{(k)} = \mathbf{D}^{(k)}\boldsymbol{\tau}^{(k)} + \mathbf{r}^{(k)} \qquad k = 1 \ldots L \tag{25}$$

Here $\boldsymbol{\tau}^{(k)}$ is the vector of forces acting on bodies bordering to S_k, and occurring in constraints external to S_k. Obviously, each of these constraints can be included in the system S or can be external to S. Therefore, $\boldsymbol{\tau}^{(k)}$ can be represented as a sum

$$\boldsymbol{\tau}^{(k)} = \left(\mathbf{G}^{(k)}\right)^T \boldsymbol{\lambda} + \hat{\boldsymbol{\tau}}^{(k)} \tag{26}$$

where

$\left(\mathbf{G}^{(k)}\right)^T \boldsymbol{\lambda} = \left(\frac{\partial \mathbf{g}}{\partial \mathbf{q}_S^{(k)}} \mathbf{T}_S^{(k)} \right)^T \boldsymbol{\lambda}$ is the vector of forces that occur in the constraints, included in *S,* and act on bodies bordering S_k

$\hat{\boldsymbol{\tau}}^{(k)}$ is the vector of forces that occur in the constraints external to *S,* and act on bodies bordering to S_k

By substituting $\boldsymbol{\tau}^{(k)}$ in the equation (25) and grouping, we obtain the matrix equation

$$\dot{\mathbf{v}}_S = \hat{\mathbf{D}}(\mathbf{G}^T \boldsymbol{\lambda} + \hat{\boldsymbol{\tau}}) + \hat{\mathbf{r}} \tag{27}$$

where

$$\hat{\mathbf{D}} = diag\left(\mathbf{D}^{(1)}, \quad \text{K} \ , \quad \mathbf{D}^{(L)}\right) \tag{28}$$

$$\hat{\mathbf{r}} = \left(\left(\mathbf{r}^{(1)}\right)^T \quad \text{K} \quad \left(\mathbf{r}^{(L)}\right)^T \right)^T \tag{29}$$

Let $\mathbf{q}_e \subset \mathbf{q}_S$ be the *m*-vector of coordinates of bodies bordering *S*. The dependency of $\dot{\mathbf{v}}_e$ on $\dot{\mathbf{v}}_S$ can be written in the matrix form

$$\dot{\mathbf{v}}_e = \mathbf{P}\dot{\mathbf{v}}_S \tag{30}$$

where **P** is a (*m,n*) matrix. Since not all bodies in *S* have external connections, we can write $\hat{\boldsymbol{\tau}}$ as a product:

$$\hat{\boldsymbol{\tau}} = \mathbf{P}^T \boldsymbol{\tau} \tag{31}$$

where $\boldsymbol{\tau}$ is the *m*-vector of forces acting in external constraints in *S.*

Substituting $\boldsymbol{\tau}$ from the formula (31) in (27), we get:

$$\dot{\mathbf{v}}_S = \hat{\mathbf{D}}\mathbf{G}^T \boldsymbol{\lambda} + \hat{\mathbf{D}}\mathbf{P}^T \boldsymbol{\tau} + \hat{\mathbf{r}} \tag{32}$$

Differentiating (23) twice, we obtain the equations of constraints on the acceleration level

$$\mathbf{0} = \mathbf{G}\dot{\mathbf{v}}_S + \dot{\mathbf{G}}\mathbf{v}_S = \mathbf{G}\dot{\mathbf{v}}_S + \mathbf{u} \tag{33}$$

Now we substitute $\dot{\mathbf{v}}_S$ from the equation (32) and get

$$\mathbf{G}\hat{\mathbf{D}}\mathbf{G}^T \boldsymbol{\lambda} = -\mathbf{G}\hat{\mathbf{D}}\mathbf{P}^T \boldsymbol{\tau} - \mathbf{G}\hat{\mathbf{r}} - \mathbf{u} \tag{34}$$

Using the same technique, as in the previous part, we can find the dependency of $\boldsymbol{\lambda}$ on $\boldsymbol{\tau}$

$$\boldsymbol{\lambda} = \mathbf{S}\boldsymbol{\tau} + \mathbf{b} \tag{35}$$

where

$$\begin{aligned} \mathbf{S} &= -(\mathbf{G}\hat{\mathbf{D}}\mathbf{G}^T)^+ \mathbf{G}^T \hat{\mathbf{D}}\mathbf{P}^T \\ \mathbf{b} &= -(\mathbf{G}\hat{\mathbf{D}}\mathbf{G}^T)^+ (\mathbf{G}\hat{\mathbf{r}} + \mathbf{u}) \end{aligned} \tag{36}$$

Finally, substituting $\boldsymbol{\lambda}$ in (32) and using (30), we obtain the desired dependency of accelerations $\dot{\mathbf{v}}_e$ on forces $\boldsymbol{\tau}$

$$\dot{\mathbf{v}}_e = \mathbf{D}\boldsymbol{\tau} + \mathbf{r} \tag{37}$$

where

$$\begin{aligned} \mathbf{D} &= \mathbf{P}\hat{\mathbf{D}}\mathbf{G}^T\mathbf{S} + \mathbf{P}\hat{\mathbf{D}}\mathbf{P}^T \\ \mathbf{r} &= \mathbf{P}\hat{\mathbf{D}}\mathbf{G}^T\mathbf{b} + \mathbf{P}\hat{\mathbf{r}} \end{aligned} \tag{38}$$

Using the equation (38), the subsystem calculates $\mathbf{D}$ and $\mathbf{r}$ and transmits them to its child.

4.4 Calculation of Forces on the Top of the Hierarchy

Iteratively repeating the previous step, we reach the top of the hierarchy. Let *S* denote a system of the highest hierarchical level. Using the algorithm similar to the one we considered in the previous chapter, we obtain the analogue of the equation (27)

$$\dot{\mathbf{v}}_S = \hat{\mathbf{D}}\mathbf{G}^T\boldsymbol{\lambda} + \hat{\mathbf{r}} \tag{39}$$

where

$\dot{\mathbf{v}}_S$ is the vector of coordinates of bodies bordering the parents of *S,*
$\mathbf{g}$ is the vector of equations of constraints in *S,*
$\mathbf{G}^T\boldsymbol{\lambda} = \left(\dfrac{\partial \mathbf{g}}{\partial \mathbf{q}_E}\mathbf{T}_E\right)^T\boldsymbol{\lambda}$ is the vector of forces acting in constraints in *S,*

$\hat{\mathbf{D}}$, $\hat{\mathbf{r}}$ are dependency matrices, transmitted by parents of *S.*

Therefore, using the algorithm similar to the one we considered in the previous chapter, we obtain $\boldsymbol{\lambda}$

$$\boldsymbol{\lambda} = -(\mathbf{G}\hat{\mathbf{D}}\mathbf{G}^T)^+(\mathbf{G}\hat{\mathbf{r}} + \mathbf{u}) \tag{40}$$

The forces acting in the constraints between the parents of *S* are:

$$\mathbf{t} = \mathbf{G}^T\boldsymbol{\lambda} = -\mathbf{G}^T(\mathbf{G}\mathbf{M}^{-1}\mathbf{G}^T)^+(\mathbf{G}^T\hat{\mathbf{r}} + \mathbf{u}) \tag{41}$$

Let $\boldsymbol{\tau}^{(k)}$ denote the vector of forces acting in the external links of S_k, where S_k is the *k*-th parent of *S.* Obviously, $\boldsymbol{\tau}^{(k)}$ is a part of the vector **t.** In our algorithm *S* calculates $\boldsymbol{\tau}^{(k)}$ and transmits it to S_k.

4.5 Backward Hierarchical Calculation of the Absolute Accelerations

Subsequently, each subsystem gets the current value of $\boldsymbol{\tau}$ from its child.

If the subsystem is derived, then, using $\boldsymbol{\tau}$, it calculates from (35) the vector $\boldsymbol{\lambda}$. Then for each parent S_k the subsystem calculates $\boldsymbol{\tau}^{(k)}$ from (31), (26) and transmits it to the parent.

Otherwise, on the lowest level of the hierarchy, the subsystem calculates the vector $\boldsymbol{\lambda}$ from (16). Substituting it in (12), we calculate the vector of accelerations $\dot{\mathbf{v}}_I$. From the equation (19) we get $\dot{\mathbf{v}}_e$.

Finally, the absolute accelerations of all bodies are calculated.

5 Main Idea of the Distributed Post-Stabilization

After the calculation of the accelerations, we perform the calculation of the absolute coordinates $\tilde{\mathbf{q}}(t_{k+1})$ and the velocities $\tilde{\mathbf{v}}(t_{k+1})$ for the new time step, using a favourite ODE integration scheme (e.g. Runge-Kutta or some multistep method).

However, for $\tilde{\mathbf{q}}(t_{k+1})$ our equations of constraints $\mathbf{g}(\tilde{\mathbf{q}}(t_{k+1})) = 0$ are not exactly satisfied, we have a drift problem. Therefore, we need to stabilize the coordinates, so that the end result is closer to the constraint manifold. Also we need to stabilize the velocities $\tilde{\mathbf{v}}(t_{k+1})$ in order to satisfy the equations of constraints on the velocity level $\mathbf{h}(\tilde{\mathbf{v}}(t_{k+1})) = 0$, where $\mathbf{h} = \frac{\partial \mathbf{g}}{\partial \tilde{\mathbf{q}}} \tilde{\mathbf{v}}$.

In our tool we improved the post-stabilization technique proposed by Ascher, Chin (1995). For the sake of simplicity, we show the distribution of the post-stabilization of velocities before the distribution of the post-stabilization of the coordinates, though on each time step we perform post-stabilization of velocities after the post-stabilization of the coordinates.

5.1 Stabilization of Velocities

Cline's (2003) formulation of the post-stabilization process, implemented on the velocity level, can be written as

$$\mathbf{v}(t_{k+1}) = \tilde{\mathbf{v}}(t_{k+1}) - \Delta\mathbf{v} \tag{42}$$

where $\mathbf{v}(t_{k+1})$ is the vector of stabilized velocities, $\tilde{\mathbf{v}}(t_{k+1})$ is the vector of unstabilized velocities, and $\Delta\mathbf{v}$ is the stabilizing vector which is calculated as the part of the solution of the matrix equation

$$\begin{pmatrix} \mathbf{M}(\mathbf{q}(t_{k+1})) & \mathbf{G}^T(\mathbf{q}(t_{k+1})) \\ \mathbf{G}(\mathbf{q}(t_{k+1})) & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{v} \\ \boldsymbol{\mu} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{h}(\tilde{\mathbf{v}}(t_{k+1})) \end{pmatrix} \tag{43}$$

Here $\mathbf{q}(t_{k+1})$ is the vector of stabilized coordinates and $\boldsymbol{\mu}$ is the vector of unknown parameters.

Comparing equations (4) and (43), we see that the matrices in the left parts of both equations are the same. Therefore, we can use the same distributed procedure of solving (43) that was used for the solving of (4). During the calculation of the accelerations, we performed the partitioning of the global vector of Lagrange forces $\mathbf{G}^T\boldsymbol{\lambda}$ on the subsystems. Now we need to do the same partitioning of the *velocities' shift* $\mathbf{G}^T\boldsymbol{\mu}$ from (43), using the same hierarchy of subsystems. For each subsystem we define the vector $\boldsymbol{\upsilon}$ of the velocities' shift, acting in the external constraints of the subsystem that is the analogue of Lagrange forces $\boldsymbol{\tau}$ acting in the external constraints of the subsystem.

Now, it is possible to perform the procedure similar to the distributed calculation of accelerations. Each subsystem calculates matrices **B** and **l** (analogues of **D** and **r**), which describe the linear dependency of $\Delta\mathbf{v}_\mathbf{e}$ on $\boldsymbol{\tau}^{(i)}$

$$\Delta\mathbf{v}_\mathbf{e} = \mathbf{B}\boldsymbol{\upsilon} + \mathbf{l} \tag{44}$$

Then the subsystem sends **B** and **l** to its child.

Having reached the highest level of the hierarchy, we start the backward hierarchical calculation of $\Delta\mathbf{v}_\mathbf{e}$, in a manner similar to the performance during the calculation of the accelerations. Each subsystem gets the current value of $\boldsymbol{\upsilon}$ from its child. Then for each parent S_i the subsystem calculates $\boldsymbol{\upsilon}^{(i)}$ and transmits it to the parent. On the lowest level of the hierarchy of subsystems we calculate $\Delta\mathbf{v}$ and stabilize the velocities of the bodies.

5.2 Stabilization of Coordinates

For simplicity of notation let **A** denote the constraint Jacobian $\left(\frac{\partial \mathbf{g}}{\partial \mathbf{q}}\right)$. Then the formulation of the post-stabilization of coordinates can be written as

$$\mathbf{q}(t_{k+1}) = \mathbf{q}(t_{k+1}) - \Delta\mathbf{q} \tag{45}$$

$$\begin{pmatrix} \mathbf{M}(\mathbf{q}(t_{k+1})) & \mathbf{A}^T(\tilde{\mathbf{q}}(t_{k+1})) \\ \mathbf{A}(\tilde{\mathbf{q}}(t_{k+1})) & 0 \end{pmatrix} \begin{pmatrix} \Delta\mathbf{q} \\ \boldsymbol{\eta} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{g}(\tilde{\mathbf{q}}(t_{k+1})) \end{pmatrix} \tag{46}$$

where $\mathbf{q}(t_{k+1})$ is the vector of stabilized coordinates $\tilde{\mathbf{q}}(t_{k+1})$ is the vector of unstabilized velocities, and $\Delta\mathbf{q}$ is the stabilizing vector.

It is obvious that this system of equations is similar to the formulation of the post-stabilization of velocities. However, we should take into account, that the vector **g** includes the equation of constraints connecting bodies as well as the normalization conditions on the Euler parameters of bodies

$$\|\boldsymbol{\theta}_k\| - 1 = 0 \tag{47}$$

Therefore, we need to start our distributed calculation of $\Delta\mathbf{q}$ not from the basic subsystems, but from the bodies. At the beginning of the forward hierarchical step each body calculates its matrices **N** and **z**, which set the dependencies of $\Delta\mathbf{q}$ of coordinates of bodies on the *coordinates' shift* $\boldsymbol{\varsigma}$ acting in the bodies' constraints

$$\Delta\mathbf{q} = \mathbf{N}\boldsymbol{\varsigma} + \mathbf{z} \tag{48}$$

Then the body sends **N** and **z** to its child. After that we repeat the procedure similar to the distribute stabilization of velocities: during the forward hierarchical step subsystems calculate their matrices **N**, **z** and send them to their children. During the backward hierarchical step subsystems get the current value of their $\boldsymbol{\varsigma}$ from their children and calculate $\boldsymbol{\varsigma}^{(i)}$ of their parents. After the subsystems of the lowest level of the hierarchy have calculated $\boldsymbol{\varsigma}$ of their bodies, we can calculate the vector $\Delta\mathbf{q}$ and stabilize the coordinates of the bodies.

6 Computation Complexity

Vlasenko (2006) showed that on each time step each subsystem performs $O(n^3+c^3)$ numerical operations during the calculation of acceleration, where n is the total numbers of simulating bodies and c denotes the number of constraints. Since the distribute stabilization is performed in the similar way like the calculation of accelerations, the complete complexity of the subsystem's simulation is $O(n^3+c^3)$.

Consider now a well-partitioned mechanical system *S,* including N bodies. Let all subsystems on all levels of the hierarchy have internal bodies and the number of bodies and internal constraints in each subsystem be limited by the global constant D. Therefore, the computation complexity of each subsystem is limited by $O(D^3)$. Since all subsystems have internal bodies, it follows that the total number of subsystems is limited by N. Finally, we obtain that the global complexity of the computation complexity is $O(N \cdot D^3)$.

7 Implementation

An object-oriented software Virtual Systems Developer (VSD) was developed based on the algorithm. In the tool a simulated mechanical system is split into functional parts representing real components. A wide set of objects, describing different types of constraints and forces, was developed: revolute joint, ball joint, stiff connection, gravity force, torque, springs etc.

Using Autodesk Inventor API, an integration of the software with Autodesk Inventor was performed. Design engineers can specify geometric and material data of simulation models in Inventor and then translate it into the

simulation tool. This approach minimizes the development cost of the model and the time of the training of end-users.

8 Example of Simulation

A simulation of the car model shown in Figure 5 was performed. This example illustrates all advantages of the method: the object-oriented simulation of multibodies, the stabilization of a closed-loop system, and the numerical efficiency of the distributed simulation.

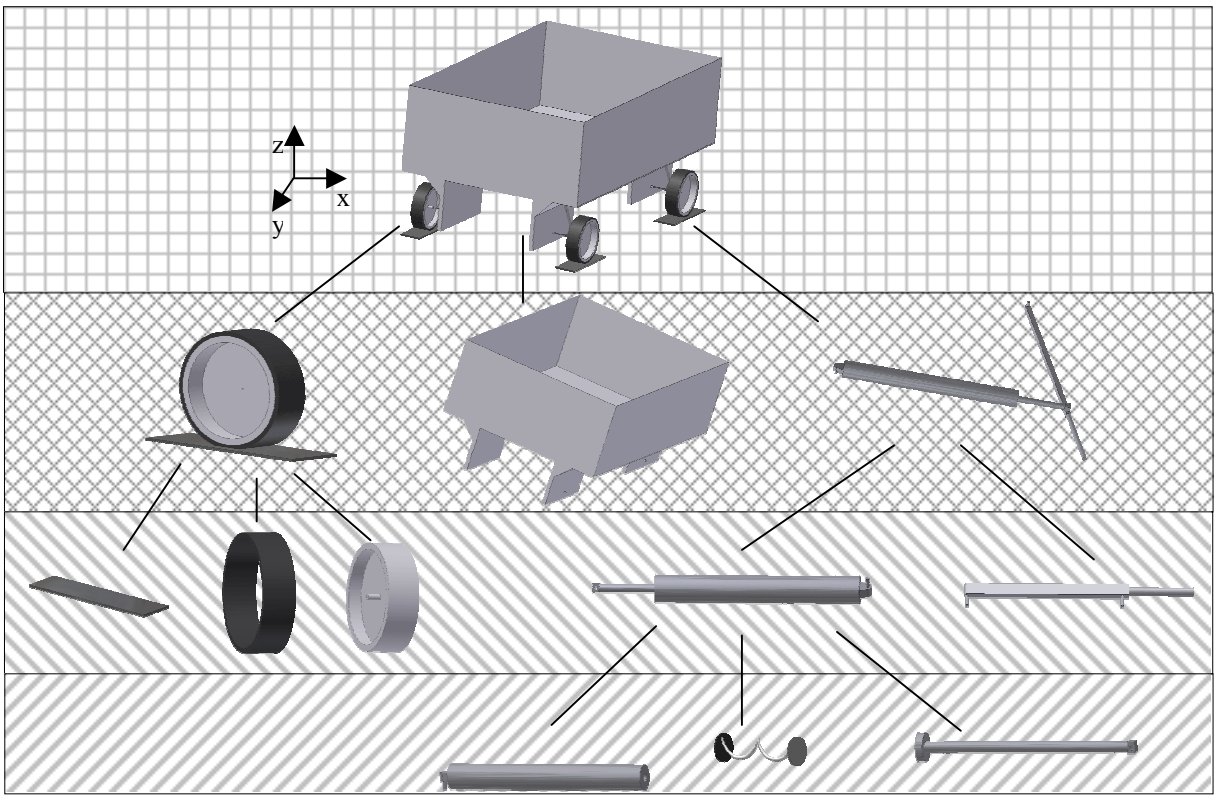

Figure 5. Car model

The complete model consists of 17 bodies coupled in several subsystems: four dampers, four suspensions, four wheel subsystems. The bodies are connected by 20 joints. The hierarchy of submodels has three levels: the subsystems of the first level are dampers and wheels, the subsystems of the second level are suspensions, and on the highest level the complete car is situated.

The tires of the car are simulated by springs with dampers. The model is stable by design because additional springs are placed between wheels and ground acting in x and y direction.

We performed the emulation of the passenger that gets into the car by the additional force f (measured in Newtons) acting in z-direction, the value of which depends on time t:

$$f_z = \begin{cases} 0 & \text{when } t < 2.5 \\ 1000 \cdot (t\text{-}2.5)/0.1 & \text{when } t \in [2.5,\quad 2.6] \\ 1000 & \text{when } t > 2.6 \end{cases} \tag{49}$$

The car is modeled by Autodesk Inventor and converted to VSD. The simulation time interval was chosen to be [0s, 6s]. The integrator uses the Runge-Kutta algorithm of the fourth order with a fixed time step equal to 0.001s. Figure 6 shows the changes of the z-coordinate of the car body, measured in meters.

Data of the simulation show that the algorithm is stable and the drift of the model has order 10^{-10}, which is equal to the accuracy of the model definition of Autodesk Inventor.

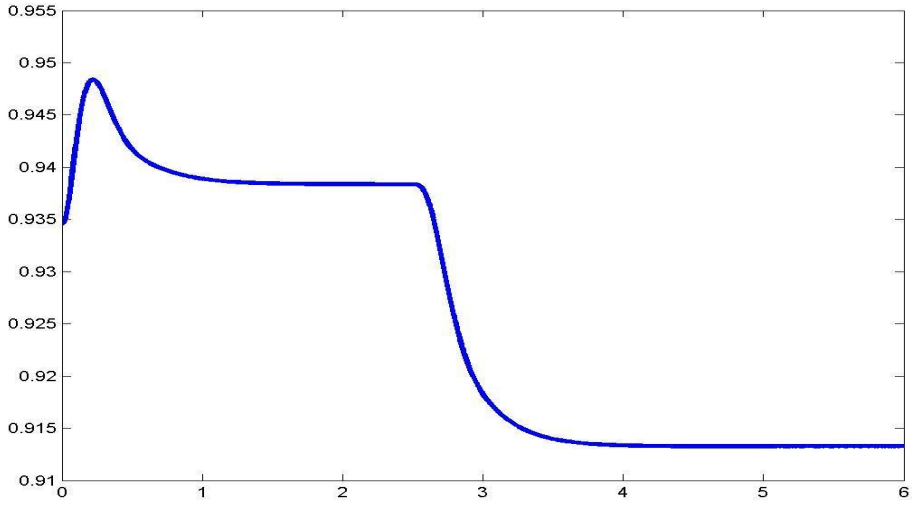

Figure 6. z-coordinate of the car body

For the validation of our simulation results we have built up the same model in Simpack and performed the simulation using Simpack's default integrator SODASRT, based on the DAE integrator DASSL. The comparison shows that the dynamics of the model was calculated correctly. The absolute difference between *z*-coordinates of the car body in the VSD and in Simpack has order 10^{-5} and is stable. This result is comparable with other tests of DASSL integrator (e.g. the coordinate error of the simulation of a 2-D car truck performed by Kunkel et. al (1997), the coordinate error of the simulation of Andrew's squeezing mechanism, presented by Hairer and Wanner (1996)).

In comparison with the simulation based on equations of motion (3) - (6), the partitioned simulation of the model is about four times faster.

9 Conclusion

This paper presents a new algorithm that can be used as a basis for a very powerful tool VSD for component-oriented simulation of the forward dynamics of multibody systems. It is based on the exact, non-iterative method, which is applicable to mechanisms of any joint type and any topology, including branches and kinematic loops. The simulation of well-partitioned systems has complexity O(*N*), where *N* is the total number of simulated bodies.

An integration of VSD with Autodesk Inventor was developed. Models can be defined in Inventor and translated into VSD without any redesign. This approach minimizes the development cost of models, and reusability of submodels.

Experimental data show the stability of the method. The drift of the car model with the closed-loop structure is limited for a long period of time and has order 10^{-10}, which is equal to the accuracy of the model definition Autodesk Inventor. The comparison of simulations results with results obtained by Simpack software, shows that the dynamics of the example was calculated correctly and accurately in VSD.

References

Ascher, U.; Chin, H.; Petzold, L. and Reich, S.: *Stabilization of constrained mechanical systems with DAEs and invariant manifolds*. The Journal of Mechanics of Structures and Machines, 23(2), 135-157(1995).

Cellier, F.E.: *Object-Oriented Modeling*: *Means for Dealing With System Complexity*. Proc. 15th Benelux Meeting on Systems and Control, Mierlo, The Netherlands, (1996), 53-64.

Chin, H.: *Stabilization methods for simulation of constrained multibody dynamics.* PhD thesis, Institute of Applied Mathematics, University of British Columbia (1995).

Cline, M. B. and Pai, D. K.: *Post-Stabilization for Rigid Body Simulation with Contact and Constraints.* Proc. IEEE Intl. Conf. on Robotics and Autom., (2003).

E. Hairer and G. Wanner. *Solving Ordinary Differential Equations. II. Stiff and Differential-Algebraic Problems.* Springer–Verlag, Berlin Heidelberg New York, 2nd edition (1996).

Kasper, R.; Vlasenko, D.: *Method for distributed forward dynamic simulation of constrained mechanical systems*. 5th EUROSIM Congress on Modelling and Simulation, S. 28, Paris. EUROSIM-FRANCOSIM-ARGESIM, Volume I (Book of Abstracts); Volume II (Full Papers CD Volume) (2004).

Kunkel, P.; Mehrmann, V.; Rath, W.; Weickert, J.: *A new software package for linear differential-algebraic equation.* SIAM J. Sci. Comput. 18, pp. 115-138 (1997).

Press, W. H.; Teukolsky, S. A. et al.: *Numerical Recipes in C++ – The Art of Scientific Computing*. Cambridge, MA, Cambridge University Press, (2002).

Shabana, A.A.: *Computational Dynamics*. Wiley, New York (2001).

Vlasenko, D.: *Component-oriented method for simulation of multibody dynamics.* PhD thesis, Institute of Mobile Systems, Otto-von-Guericke-University Magdeburg (2006).

Address: Prof. Dr.-Ing. Roland Kasper and Dr.-Ing. Dmitry Vlasenko, Institute of Mobile Systems (IMS), Otto-von-Guericke-University Magdeburg, Universitätsplatz 2, D-39106 Magdeburg.
E-Mail: Roland.Kasper@MB.Uni-Magdeburg.de; Dmitri.Vlasenko@MB.Uni-Magdeburg.de.