Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2007
September 4-7, 2007, Las Vegas, Nevada, USA

DETC2007/VIB-35065

INTEGRATION METHOD OF CAD SYSTEMS

Dmitry Vlasenko *
Institute of Mobile Systems (IMS)
Otto-von-Guericke-University Magdeburg
Germany

Roland Kasper
Institute of Mobile Systems (IMS)
Otto-von-Guericke-University Magdeburg
Germany

ABSTRACT
This paper presents the modification of a non-iterative algorithm for the component-oriented simulation of the dynamics of multibodies. Now the method can be implemented for the component-oriented simulation of the dynamics of CAD systems with redundant constraints. Also the new version of the method is well-suitable for the implementation of sparse solvers. The algorithm was implemented in the **V**irtual **S**ystem **D**esigner (VSD) software, integrated with a CAD tool Autodesk Inventor. The simulation results of two CAD models: a car and a steam machine shows the method's stability and accuracy.

1. INTRODUCTION
Computer Aided Design (CAD) tools allow engineers to develop, visualize and test a complete machine before the first prototype is ever produced. The use of CAD systems reduces the development costs and the modeling time.

Since manufacturers need the simulation of multibody systems, designed in CAD tools, the coupling of CAD with multibody simulation software is strictly required. Different commercial tools such as Simpack or ADAMS allow the import of CAD data. Also was developed the integration of the CAD tool SolidWorks with Modelica simulation language [2] and with a Dynamic Analysis simulation tool DAP3D [1].

However, existing integration examples impose significant restrictions on CAD models: e.g. absence of redundant constraints [1] or a rigid fixing of one of the simulated bodies [2]. Moreover, nowadays for the simulation the motion of a CAD model the complete mechanical information (e.g. masses of bodies, types and parameters of joints) about the model's subsystems is needful. Therefore, existing tools can not be implemented in situations when components are made by foreign manufactures who wish to protect the commercial classified information of their components (e.g. manufactures of motors do not provide the complete mechanical information and original CAD models of motors to manufactures of manipulators).

In the last years we developed and implemented a method, based on the projection algorithm for absolute coordinates, which performs the component-oriented simulation of multibodies [6, 7]. In our method the model's partition, defined during the model's specification, remains 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:

- Each subsystem can be modeled, tested and compiled independently. This significantly decreases the time and cost of the models' development and test.
- The commercial 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. All submodel's internal data: parameters of constraints, forces, masses of internal bodies, etc. are unknown to the users of submodels.
- Critical effects like Coulomb friction, backslash etc. can be encapsulated inside a subsystem.
- Subsystems are ideal candidates for the partitioning of systems on multiple processors.
- Mechanical subsystems are represented by separate objects which interact via predefined interfaces with each

* Postdoctoral researcher and author of correspondence, Phone: +49 391-67-12-668, Email: Dmitri.Vlasenko@Masch-Bau.Uni-Magdeburg.DE.

other. Using such interface, simulation model can be easily extended by electronic and control components.

Now we are strongly interested in the further increasing of the numerical efficiency of the simulation process in VSD. In this paper we show the modification of our method, which uses the decomposition of sparse matrices during the simulation of multibodies. It is suitable for the implementation of sparse solvers or a special preprocessing module, performing a symbolic simplification of decomposition of matrices [5].

Also we discuss the simulation of models with redundant constraints. In many cases design engineers develop CAD models using the greater number of constraints than it is needful from the mechanical point of view. The redesign of CAD models and the elimination of redundant constraints by engineer is a very costly procedure. The correct simulation of CAD models with redundant constraints is shown in this paper.

2. SIMULATION STEPS

Fig. 1 shows the object-oriented method of the simulation of mechanical systems, implemented in VSD [6]. The base idea of the method is to perform the simulation of mechanical systems using the hierarchy of submodels that builds up the complete system.

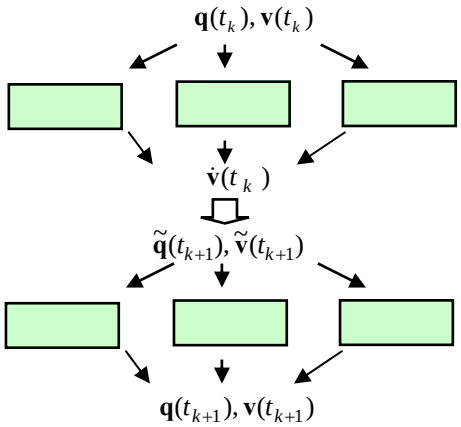

Figure 1. Data flow in simulation steps

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

1. Distributed calculation of the absolute accelerations $\dot{\mathbf{v}}(t_k)$.
2. Calculation of the absolute coordinates and velocities at 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 $\tilde{\mathbf{v}}(t_{k+1})$ at the new time step can be obtained.
3. Distributed stabilization of the absolute coordinates $\mathbf{q}(t_{k+1})$ and velocities $\mathbf{v}(t_{k+1})$.

3. HIERARCHICAL CALCULATION OF THE ACCELERATIONS

The distributed calculation of the accelerations consists of three steps, shown in Fig. 2:

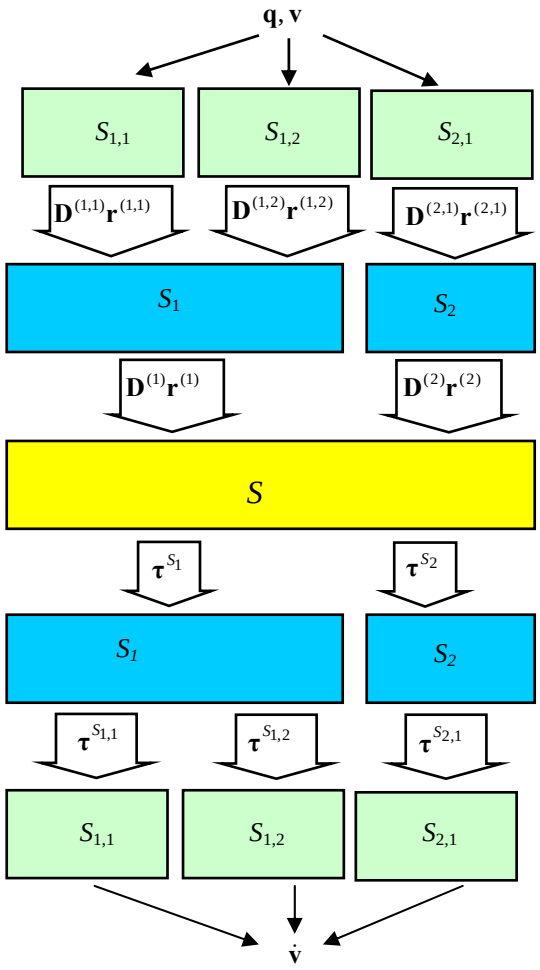

Figure 2. Hierarchical calculation of the accelerations

1. Starting from the lowest level of the hierarchy, each subsystem S generates the matrix $\mathbf{D}$ and the vector $\mathbf{r}$ using the equation of constraints connecting the parents. Here $\mathbf{D}$ and $\mathbf{r}$ show the linear dependency of $\dot{\mathbf{v}}_b^S$ on $\boldsymbol{\tau}$

$$\dot{\mathbf{v}}_b^S = \mathbf{D}\boldsymbol{\tau}^S + \mathbf{r} \qquad (1)$$

where $\dot{\mathbf{v}}_b^S$ is the vector of accelerations of subsystem's *bordering* bodies (i.e. bodies, connected outside the subsystem via external joints) and $\boldsymbol{\tau}$ is the vector of forces in subsystem's external links.

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

2. The subsystem of the highest level *S* gets **D** and **r** matrices from its parents and calculates the forces acting in the constraints connecting the parents. Then to each parent S_i the subsystems transmits the correspondent vector $\boldsymbol{\tau}^{S_i}$, where $\boldsymbol{\tau}^{S_i}$ is the vector of forces acting in the external constraints of S_i.
3. During the backward calculation of the accelerations each subsystem *S* gets the current value of $\boldsymbol{\tau}^S$ from its child. Then for each parent S_i the subsystem calculates $\boldsymbol{\tau}^{S_i}$ and transmits it to the parent. Subsystems of the lowest level of hierarchy calculate the absolute accelerations of its bodies.

The hierarchical projection of the absolute coordinates and velocities is performed in a similar way [6] and has the same order of complexity as the hierarchical calculation of the accelerations.

3.1. Equations of motion of a basic subsystem

Since the models are defined in CAD systems, it seems reasonable to use the absolute coordinates for the description of equations of motion. Moreover, if we use absolute coordinates, the equations of motion of models, can be partitioned to submodels accordingly the model's partition, defined during the model's design in CAD tool.

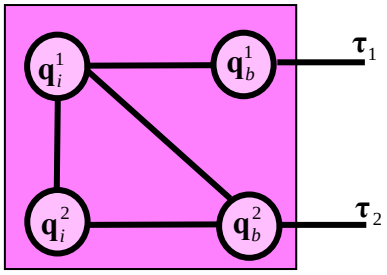

Figure 3: A subsystem of several connected bodies

Consider a *basic* subsystem *S* (i.e. the subsystem of the lowest level of the hierarchy), shown in Fig. 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:

$$\mathbf{g} = \begin{pmatrix} g_1(\mathbf{q}^S) & \dots & g_c(\mathbf{q}^S) \end{pmatrix}^T = \begin{pmatrix} 0 & \dots & 0 \end{pmatrix}^T \qquad (2)$$

where $\mathbf{q}^S$ is the 7*n*-vector of the absolute coordinates of the subsystem's bodies, consisting of Cartesian coordinates and Euler parameters of bodies in inertial frame.

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

$$\mathbf{q}^S = \begin{pmatrix} \mathbf{q}_b^S \\ \mathbf{q}_i^S \end{pmatrix} \qquad (3)$$

We partition the Jacobian matrix $\mathbf{G}$ into parts

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}_b & \mathbf{G}_i \end{pmatrix} = \begin{pmatrix} \dfrac{\partial \mathbf{g}}{\partial \mathbf{q}_b^S} & \dfrac{\partial \mathbf{g}}{\partial \mathbf{q}_i^S} \end{pmatrix} \qquad (4)$$

Here for simplicity of notation we omit the transformation matrix **T** describing the relation between the absolute coordinates and velocities: $\dot{\mathbf{q}}^S = \mathbf{T}(\mathbf{q}^S)\mathbf{v}^S$. The equations of motion, corresponding to the subsystem, are

$$\mathbf{M}_b \dot{\mathbf{v}}_b^S = \mathbf{f}_b + \boldsymbol{\tau}^S + \mathbf{G}_b^T \boldsymbol{\lambda} \qquad (5)$$

$$\mathbf{M}_i \dot{\mathbf{v}}_i^S = \mathbf{f}_i + \mathbf{G}_i^T \boldsymbol{\lambda} \qquad (6)$$

where $\mathbf{M}_b = diag(\mathbf{M}_1,...,\mathbf{M}_m)$ 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, $\mathbf{f}_b = \begin{pmatrix} \mathbf{f}_1^T & \cdots & \mathbf{f}_m^T \end{pmatrix}^T$ is the vector external forces acting on bordering bodies, $\mathbf{f}_i = \begin{pmatrix} \mathbf{f}_{m+1}^T & \cdots & \mathbf{f}_n^T \end{pmatrix}^T$ is the vector external forces acting on internal bodies.

Differentiating twice (2), we obtain the equations of motion on the acceleration level:

$$\mathbf{G}_b \dot{\mathbf{v}}_b^S + \mathbf{G}_i \dot{\mathbf{v}}_i^S - \mathbf{u} = 0 \qquad (7)$$

where $\mathbf{u} = -\dot{\mathbf{G}}\mathbf{v}^S$.

Substituting $\dot{\mathbf{v}}_b^S$, $\dot{\mathbf{v}}_i^S$ from (5) and (6) in (7), we obtain

$$\mathbf{G}_b \mathbf{M}_b^{-1}(\mathbf{f}_b + \boldsymbol{\tau}^S + \mathbf{G}_b^T \boldsymbol{\lambda}) + \mathbf{G}_i \mathbf{M}_i^{-1}(\mathbf{f}_i + \mathbf{G}_i^T \boldsymbol{\lambda}) = \mathbf{u} \qquad (8)$$

or, in the other form

$$\mathbf{G}\mathbf{M}^{-1}\mathbf{G}^T \boldsymbol{\lambda} = -\mathbf{G}_b^T \mathbf{M}_b^{-1} \boldsymbol{\tau}^S + \mathbf{a} \qquad (9)$$

where $\mathbf{a} = \mathbf{u} - \mathbf{G}\mathbf{M}^{-1}\mathbf{f}$.

If we find from (9) the dependency of $\boldsymbol{\lambda}$ on $\boldsymbol{\tau}$:

$$\boldsymbol{\lambda} = \mathbf{K}\boldsymbol{\tau}^S + \mathbf{b} \qquad (10)$$

then we can substitute it in (5), obtaining the dependency of $\dot{\mathbf{v}}_e$ on $\boldsymbol{\tau}$:

$$\dot{\mathbf{v}}_b^S = \mathbf{D}^S \boldsymbol{\tau}^S + \mathbf{r}^S \qquad (11)$$

where

$$\mathbf{D}^S = \mathbf{M}_b^{-1}\mathbf{G}_b^T\mathbf{K} + \mathbf{M}_b^{-1} \quad (12)$$

$$\mathbf{r}^S = \mathbf{M}_b^{-1}\mathbf{G}_b^T + \mathbf{f}_b \quad (13)$$

Consider now the process of calculation of **K** and **b** from (9). It can be easily checked that they can be calculated as the roots of the equation:

$$\mathbf{G}\mathbf{M}^{-1}\mathbf{G}^{\mathbf{T}}\mathbf{X} = \mathbf{B} \quad (14)$$

where $\mathbf{X} = \begin{pmatrix}\mathbf{K} & \mathbf{b}\end{pmatrix}$ and $\mathbf{B} = \begin{pmatrix}-\mathbf{G}_\mathbf{e}\mathbf{M}_\mathbf{e}^{-1} & \mathbf{a}\end{pmatrix}$.

Since the absolute coordinates of bodies are used, it follows that the matrices $\mathbf{M}^{-1}$ and $\mathbf{G}$ are sparse. The standard solution, using the sparsity of the matrices [4], is based on the Cholesky decomposition of $\mathbf{M} = \mathbf{L}\mathbf{L}^T$ where **L** is a non-singular lower triangular matrix. Then (14) can be written as

$$\mathbf{A}^T\mathbf{A}\mathbf{X} = \mathbf{B} \quad (15)$$

where $\mathbf{A} = \mathbf{L}^{-1}\mathbf{G}^T$.

If the matrix **A** is linearly independent (e.g. all rows of the Jacobian matrix **G** are independent), then using the QR-decomposition of $\mathbf{A} = \mathbf{Q}\begin{pmatrix}\mathbf{R}\\ \mathbf{0}\end{pmatrix}$, we can calculate **X** as

$$\mathbf{X} = \mathbf{R}^{-1}\left(\mathbf{R}^T\right)^{-1}\mathbf{B} \quad (16)$$

However, in the case of redundant constraints **G** has dependent rows! As it was noted, in many cases design engineers develop CAD models, using the more number of constraints than it is needful from the mechanical point of view. The redesign of CAD models and the elimination of redundant constraints by engineer is very costly procedure. Moreover, on high levels of the hierarchy we will need to solve similar equations, where **L** is positive-semidefinite.

That is why we propose to find the solution of (15) in the case when **A** has dependent columns. Consider this procedure more precisely.

Clearly, if **A** has dependent columns then the product $\mathbf{A}^T\mathbf{A}$ is singular and the solution (15) is not unique. In our case we need only an arbitrary solution with limited norm.

Performing the QR-decomposition with pivoting of **A**, we obtain [3]:

$$\mathbf{A}\boldsymbol{\Pi} = \mathbf{Q}\mathbf{R} \quad (17)$$

$$\mathbf{R} = \begin{pmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ 0 & 0\end{pmatrix} \quad (18)$$

Here **Q** is orthogonal matrix, **Π** is a permutation and $\mathbf{R}_1$ is a non-singular and upper triangular (r, r) matrix, where r=rank(**A**). From the definition of **Π** and **Q** follows that $\boldsymbol{\Pi}^T = \boldsymbol{\Pi}^{-1}$ and $\mathbf{Q}^T = \mathbf{Q}^{-1}$**.** Therefore, (15) can be written as:

$$\mathbf{A}^T\mathbf{A}\mathbf{X} = \boldsymbol{\Pi}\mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R}\boldsymbol{\Pi}^T\mathbf{X} = \boldsymbol{\Pi}\mathbf{R}^T\mathbf{R}\boldsymbol{\Pi}^T\mathbf{X} = \mathbf{B} \quad (19)$$

or, in the other form:

$$\mathbf{R}^T\mathbf{R}\tilde{\mathbf{X}} = \tilde{\mathbf{B}} \quad (20)$$

where $\tilde{\mathbf{X}} = \boldsymbol{\Pi}^T\mathbf{X}$, $\tilde{\mathbf{B}} = \boldsymbol{\Pi}^T\mathbf{B}$ are permutated vectors.

Denoting by **Y** a product $\mathbf{Y} = \mathbf{R}\tilde{\mathbf{X}}$, we can rewrite (20) as a system of equations:

$$\begin{aligned}\mathbf{R}\tilde{\mathbf{X}} &= \mathbf{Y}\\ \mathbf{R}^T\mathbf{X} &= \tilde{\mathbf{B}}\end{aligned} \quad (21)$$

or, in the other form:

$$\begin{bmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ 0 & 0\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{X}}_1\\ \tilde{\mathbf{X}}_2\end{bmatrix} = \begin{bmatrix}\mathbf{Y}_1\\ \mathbf{Y}_2\end{bmatrix} \quad (22)$$

$$\begin{bmatrix}\mathbf{R}_1^T & 0\\ \mathbf{R}_2^T & 0\end{bmatrix}\begin{bmatrix}\mathbf{Y}_1\\ \mathbf{Y}_2\end{bmatrix} = \begin{bmatrix}\tilde{\mathbf{B}}_1\\ \tilde{\mathbf{B}}_2\end{bmatrix} \quad (23)$$

Consider the matrix equation (23). The matrix $\mathbf{R}_1$ is non-singular, therefore, the upper part of the system follows that $\mathbf{Y_1}$ is uniquely defined:

$$\mathbf{Y}_1 = \left(\mathbf{R}_1^T\right)^{-1}\tilde{\mathbf{B}}_1 = \left(\mathbf{R}_1^{-1}\right)^T\tilde{\mathbf{B}}_1 \quad (24)$$

So, from the there lower part of the equation yields the restriction on the value of $\tilde{\mathbf{B}}_2$: $\tilde{\mathbf{B}}_2 = \mathbf{R}_2^T\left(\mathbf{R}_1^{-1}\right)^T\tilde{\mathbf{B}}_1$.

Substituting $\mathbf{Y_1}$ in (23), we obtain:

$$\begin{aligned}\mathbf{R}_1\tilde{\mathbf{X}}_1 + \mathbf{R}_2\tilde{\mathbf{X}}_2 &= \left(\mathbf{R}_1^{-1}\right)^T\tilde{\mathbf{B}}_1\\ \mathbf{0} &= \mathbf{y}_2\end{aligned} \quad (25)$$

We need and arbitrary limited solution, hence we can set $\tilde{\mathbf{X}}_2$ to zero and obtain from (25) :

$$\tilde{\mathbf{X}} = \begin{pmatrix}\mathbf{R}_1^{-1}\left(\mathbf{R}_1^{-1}\right)^T\tilde{\mathbf{B}}_1\\ 0\end{pmatrix} \quad (26)$$

Since $\boldsymbol{\Pi}\boldsymbol{\Pi}^T$**=I**, it follows that $\mathbf{X} = \boldsymbol{\Pi}\tilde{\mathbf{X}}$. Partitioning **Π** into submatrices **Π=(Π**$_1$**, Π**$_2$**),** we get: $\tilde{\mathbf{B}}_1 = \boldsymbol{\Pi}_1^T\mathbf{B}$. Therefore, we obtain:

$$\mathbf{X} = \boldsymbol{\Pi}\begin{pmatrix}\mathbf{R}_1^{-1}\left(\mathbf{R}_1^{-1}\right)^T\tilde{\mathbf{B}}_1\\ 0\end{pmatrix} = \boldsymbol{\Pi}_1\mathbf{R}_1^{-1}\left(\mathbf{R}_1^{-1}\right)^T\boldsymbol{\Pi}_1^T\tilde{\mathbf{B}} \quad (27)$$

After the calculation of **S** and **b**, the subsystem calculates **D** and **r** from (12), (13) and transmits them to its child.

3.2. 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,…, S_L shown in Fig. 4.

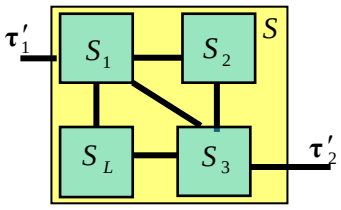

Figure 4: A subsystem consisting of several connected subsystems

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 $\mathbf{q}_b^{S_k}$ (*k*=1... *L*). We can reorder the vector $\mathbf{q}^S$ as

$$\mathbf{q}^S = \begin{pmatrix} \mathbf{q}_b^{S_1} \\ \vdots \\ \mathbf{q}_b^{S_L} \end{pmatrix} \tag{28}$$

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

$$\mathbf{g} = \begin{pmatrix} g_1(\mathbf{q}^S) & \cdots & g_c(\mathbf{q}^S) \end{pmatrix}^T = \begin{pmatrix} 0 & \ldots & 0 \end{pmatrix}^T \tag{29}$$

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

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}^{(1)} & \ldots & \mathbf{G}^{(L)} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial \mathbf{g}}{\partial \mathbf{q}_b^{S_1}} & \ldots & \dfrac{\partial \mathbf{g}}{\partial \mathbf{q}_b^{S_L}} \end{pmatrix} \tag{30}$$

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

$$\dot{\mathbf{v}}_b^{S_k} = \mathbf{D}^{S_k} \boldsymbol{\tau}^{S_k} + \mathbf{r}^{S_k} \qquad k = 1...L \tag{31}$$

Here $\boldsymbol{\tau}^{S_k}$ is the vector of forces acting on bodies bordering S_k, which occur 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, $\boldsymbol{\tau}^{S_k}$ can be represented as a sum

$$\boldsymbol{\tau}^{S_k} = \left(\mathbf{G}^{(k)}\right)^T \boldsymbol{\lambda} + \hat{\boldsymbol{\tau}}^{(k)} \tag{32}$$

where $\left(\mathbf{G}^{(k)}\right)^T \boldsymbol{\lambda}$ 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 (31) and grouping, we obtain the matrix equation

$$\dot{\mathbf{v}}^S = \hat{\mathbf{D}}(\mathbf{G}^T \boldsymbol{\lambda} + \hat{\boldsymbol{\tau}}) + \hat{\mathbf{r}} \tag{33}$$

where

$$\hat{\mathbf{D}} = diag\left(\mathbf{D}^{(1)}, \quad \ldots, \quad \mathbf{D}^{(L)}\right) \tag{34}$$

$$\hat{\mathbf{r}} = \left(\left(\mathbf{r}^{(1)}\right)^T \quad \ldots \quad \left(\mathbf{r}^{(L)}\right)^T\right)^T \tag{35}$$

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

$$\dot{\mathbf{v}}_b^S = \mathbf{P}\dot{\mathbf{v}}^S \tag{36}$$

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

$$\hat{\boldsymbol{\tau}} = \mathbf{P}^T \boldsymbol{\tau}^S \tag{37}$$

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

Substituting $\boldsymbol{\tau}^S$ from the formula (37) in (33), we get:

$$\dot{\mathbf{v}}^S = \hat{\mathbf{D}}\mathbf{G}^T \boldsymbol{\lambda} + \hat{\mathbf{D}}\mathbf{P}^T \boldsymbol{\tau}^S + \hat{\mathbf{r}} \tag{38}$$

Differentiating (29) twice, we obtain the equations of constraints on the acceleration level

$$\mathbf{0} = \mathbf{G}\dot{\mathbf{v}}^S + \dot{\mathbf{G}}\mathbf{v}^S = \mathbf{G}\dot{\mathbf{v}}^S - \mathbf{u} \tag{39}$$

Now we substitute $\dot{\mathbf{v}}^S$ from the equation (38) and get

$$\mathbf{G}\hat{\mathbf{D}}\mathbf{G}^T \boldsymbol{\lambda} = -\mathbf{G}\hat{\mathbf{D}}\mathbf{P}^T \boldsymbol{\tau}^S - \mathbf{G}\hat{\mathbf{r}} + \mathbf{u} \tag{40}$$

Like in the previous part, we need to find the dependency of $\boldsymbol{\lambda}$ on $\boldsymbol{\tau}$ in the form

$$\boldsymbol{\lambda} = \mathbf{K}\boldsymbol{\tau}^S + \mathbf{b} \tag{41}$$

Obviously, $\mathbf{K}$ and $\mathbf{b}$ can be calculated as the roots of the equation:

$$\mathbf{G}\hat{\mathbf{D}}\mathbf{G}^{\mathrm{T}}\mathbf{X} = \mathbf{B} \tag{42}$$

where $\mathbf{X} = \begin{pmatrix} \mathbf{K} & \mathbf{b} \end{pmatrix}$ and $\mathbf{B} = \begin{pmatrix} -\mathbf{G}\hat{\mathbf{D}}\mathbf{P}^T & \mathbf{u} - \mathbf{G}\hat{\mathbf{r}} \end{pmatrix}$. From the definition yields that all $\mathbf{D}^{(k)}$ are positive-semidefinite. Since $\hat{\mathbf{D}} = diag\left(\mathbf{D}^{(1)}, \quad \ldots, \quad \mathbf{D}^{(L)}\right)$, it follows that $\hat{\mathbf{D}}$ is also positive-semidefinite. Therefore, we can perform the Cholesky decomposition with pivoting of $\hat{\mathbf{D}} = \boldsymbol{\Pi}_1 \mathbf{L}\mathbf{L}^T \boldsymbol{\Pi}_1^T$ where $\mathbf{L}$ is a lower triangular matrix and $\boldsymbol{\Pi}_1$ is a permutation matrix. Like in the previous part, we obtain the matrix equation in the form

$$\mathbf{A}^T \mathbf{A}\mathbf{X} = \mathbf{B} \tag{43}$$

where $\mathbf{A} = \mathbf{L}^T \mathbf{\Pi}^T \mathbf{G}^T$. From the definition yields that **A** is a sparse matrix, whose columns can be linearly dependent. Using the procedure, shown in the previous part, we calculate **X**.

Finally, substituting $\boldsymbol{\lambda}$ in (38) and using (36), we obtain the desired dependency of accelerations $\dot{\mathbf{v}}_e$ on forces $\boldsymbol{\tau}$

$$\dot{\mathbf{v}}_b^S = \mathbf{D}\boldsymbol{\tau}^S + \mathbf{r} \tag{44}$$

where

$$\begin{aligned} \mathbf{D} &= \mathbf{P}\hat{\mathbf{D}}\mathbf{G}^T\mathbf{K} + \mathbf{P}\hat{\mathbf{D}}\mathbf{P}^T \\ \mathbf{r} &= \mathbf{P}\hat{\mathbf{D}}\mathbf{G}^T\mathbf{b} + \mathbf{P}\hat{\mathbf{r}} \end{aligned} \tag{45}$$

Using the equation (45), the subsystem calculates **D** and **r** and transmits them to its child.

With minor changes the same procedure can be used on the highest level of the hierarchy.

During the backward calculation of the accelerations each subsystem *S* gets the current value of $\boldsymbol{\tau}^S$ from its child. Then for each parent S_i the subsystem calculates from (32) the current values of $\boldsymbol{\tau}^{S_i}$ and transmits it to the parent. Subsystems of the lowest level of hierarchy calculate from (11) the absolute accelerations of their bodies.

4. POST-STABILIZATION OF CONSTRAINTS

Usually projections methods are used in the integration methods, based on the index-one formulation of the equations of motion

$$\dot{\mathbf{q}} = \mathbf{v} \tag{46}$$

$$\begin{pmatrix} \mathbf{M} & \mathbf{G}(\mathbf{q})^T \\ \mathbf{G}(\mathbf{q}) & 0 \end{pmatrix} \begin{pmatrix} \dot{\mathbf{v}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{f}(\mathbf{q}) \\ \mathbf{u}(\mathbf{q},\mathbf{v}) \end{pmatrix} \tag{47}$$

where **q** is the vector of coordinates; **v** is the vector of velocity variables; **f** is the vector of external forces; **g** is the vector of (holonomic) constraints; **M** is the mass matrix; $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers; **G** is the constraint Jacobian matrix; and $\mathbf{u} = -\dot{\mathbf{G}} \cdot \mathbf{v}$.

Clear, that the solution of (46)-(47) should satisfy the equations of constraints on the coordinate and on the velocity level

$$\mathbf{g}(\mathbf{q}) = 0 \tag{48}$$

$$\mathbf{G}(\mathbf{q})\mathbf{v} = 0 \tag{49}$$

After solving (47) for the accelerations $\dot{\mathbf{v}}$, the values of the coordinates $\tilde{\mathbf{q}}$ and the velocities $\tilde{\mathbf{v}}$ at the next time step are calculated, using standard ODE integration schemes (e.g. Runge-Kutta or multistep). But now we get so-called drift effect: equations of constraints (48)-(49) are not fulfilled completely. That is why we need to project the coordinates and the velocities

$$\begin{pmatrix} \mathbf{q} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{q}} \\ \tilde{\mathbf{v}} \end{pmatrix} + \begin{pmatrix} \Delta\mathbf{q} \\ \Delta\mathbf{v} \end{pmatrix} \tag{50}$$

where **q** and **v** are the stabilized coordinates and velocities and $\Delta\mathbf{q}$ and $\Delta\mathbf{v}$ are the stabilization displacements. Usually $\Delta\mathbf{q}$ and $\Delta\mathbf{v}$ are calculated as the minimal norm solutions of the equations

$$\mathbf{G}(\tilde{\mathbf{q}})\Delta\mathbf{q} = -\mathbf{g}(\tilde{\mathbf{q}}) \tag{51}$$

$$\mathbf{G}(\tilde{\mathbf{q}})\Delta\mathbf{v} = -\dot{\mathbf{g}}(\tilde{\mathbf{q}}, \tilde{\mathbf{v}}) \tag{52}$$

Really it is not necessary that the stabilizing displacements $\Delta\mathbf{q}$ or $\Delta\mathbf{v}$ will be minimal: we need only that they satisfy (51), (52) and has the same order as the accuracy of the used ODE integration scheme.

If redundant constraints are included in the system (i.e. **G** has dependent rows), the computation of the minimal norm solution of (51) is very costly because we need to perform the singular value decomposition or the complete orthogonal decomposition of **G** [3].

5. ALGORITHM OF THE DISTRIBUTED PROJECTION

Let us show how we can distributing the process of the stabilization at the component-oriented way. In our method we perform the projection of the coordinates and of the velocities. Since the stabilization of the coordinates and the velocities are similar, we consider in this paper only the stabilization of the coordinates. The minor difference is that on the coordinate level we need to stabilize normalization conditions on the Euler parameters of bodies.

Our method is based on the representation of the vector $\Delta\mathbf{q}_k$, projecting the coordinates of an arbitrary body, as the sum of vectors:

$$\Delta\mathbf{q}_k = \mathbf{y} - (\mathbf{z}^{(1)} + \mathbf{z}^{(2)} + ... + \mathbf{z}^{(m)}) \tag{53}$$

where **y** is the displacement of the body coordinates, stabilizing the normalization condition Euler parameters of the body, $\mathbf{z}^{(i)}$ is the displacement, stabilizing constraints in a subsystem S_i (i>0), where the body is included.

The distributed stabilization of constraints consists of five steps, similar to the steps of the distributed calculation of accelerations:

1. Each body calculates dependency matrices $\mathbf{U}_b$ and $\mathbf{c}_b$:

$$\Delta\mathbf{q}_k = \mathbf{U}_b\mathbf{z}_k + \mathbf{c}_b \tag{54}$$

where $\mathbf{z}_k$ is the displacement of the body, stabilizing equations of constraints on the higher level of the hierarchy of submodels. Then the body transmits $\mathbf{U}_b$ and $\mathbf{c}_b$ to its submodel of the first level of hierarchy.

2. During the forward hierarchical generations of stabilization's equations each subsystem *S* gets from its parents S_1, S_2…S_l their dependency matrices $\mathbf{U}_b^{S_i}$ and $\mathbf{c}_b^{S_i}$:

$$\Delta\mathbf{q}_b^{S_i} = \mathbf{U}_b^{S_i}\mathbf{z}_b^{S_i} + \mathbf{c}_b^{S_i} \qquad \forall i = 1..l \tag{55}$$

where $\Delta\mathbf{q}_b^{S_i}$ is the projection vector of bordering bodies of the *i*-th parent, $\mathbf{z}_b^{S_i}$ is the displacement of bordering bodies of the *i*-th parent, stabilizing the external constraints of the parent. Then the subsystem *S* generates dependency matrices $\mathbf{U}_b$ and $\mathbf{c}_b$:

$$\Delta\mathbf{q}_b^{S} = \mathbf{U}_b\mathbf{z}_b^{S} + \mathbf{c}_b \tag{56}$$

where $\Delta\mathbf{q}_b^{S}$ is the projection vector of bordering bodies of *S*, $\mathbf{z}_b^{S}$ is the displacement of bordering bodies of *S*, stabilizing the external constraints of *S*. Then the subsystem transmits $\mathbf{U}_b$ and $\mathbf{c}_b$ to its child.

3. On the highest level of the hierarchy the main subsystem *S* gets from its parents their dependency matrices $\mathbf{U}_b^{S_i}$ and $\mathbf{c}_b^{S_i}$ and calculates for each parent S_i the correspondent displacement $\mathbf{z}_b^{S_i}$, stabilizing the external constraints of the parent.
4. During the backward hierarchical calculation of displacements each subsystem *S* gets the current value of $\mathbf{z}_b^{S}$ from its child. Using the formula for $\mathbf{z}_b^{S}$, the subsystem calculates the displacement of bordering bodies of its parents $\mathbf{z}_b^{S_i}$, defined earlier. Then the subsystem sends $\mathbf{z}_b^{S_i}$ to its parents.
5. On the lowest level of the hierarchy each body gets the current value of $\mathbf{z}_k$ from its subsystem of the first level of the hierarchy. Using (54), the value of $\Delta\mathbf{q}_k$ is calculated and the coordinates of the body are projected:

$$\mathbf{q}_k = \tilde{\mathbf{q}} + \Delta\mathbf{q}_k \tag{57}$$

Consider now the procedure of the distributed post-stabilization of constraints more precisely.

5.1. Body level

Let us have a look at a row of (51), corresponding to the normalization condition of the Euler parameters of the *k*-th body:

$$\mathbf{G}_k(\tilde{\mathbf{q}}_k)\Delta\mathbf{q}_k = -g_k(\tilde{\mathbf{q}}_k) \tag{58}$$

where $\mathbf{q}^k = \begin{pmatrix} x_{k,1} & x_2 & x_3 & e_0 & e_1 & e_2 & e_3 \end{pmatrix}^T$ is the vector of bodies coordinates, $g^k = e_0^2 + e_1^2 + e_2^2 + e_3^2 - 1$ is the normalization condition of the Euler parameters of the body, $\mathbf{G}^k = \partial g^k / \partial \mathbf{q}^k$ is the Jacobian matrix of the condition,

Let us assume that the body is connected with other bodies, i.e. the coordinates of the body are included in the other equations of constraints. Now we represent the vector $\Delta\mathbf{q}_k$ as the sum of two vectors:

$$\Delta\mathbf{q}_k = \mathbf{y}_k + \mathbf{z}_k \tag{59}$$

where $\mathbf{y}_k$ is the displacement, stabilizing g_k, and $\mathbf{z}_k$ is the displacement of the body, stabilizing equations of constraints on the higher level of the hierarchy of submodels. If norms of $\mathbf{y}_k$ and $\mathbf{z}_k$ are minimal then we can be sure that the vector $\Delta\mathbf{q}_k$ is limited and is small enough.

Substituting $\Delta\mathbf{q}_k$ in (58), we get the dependency

$$\mathbf{G}_k\mathbf{y}_k = -\mathbf{g}_k - \mathbf{G}_k\mathbf{z}_k \tag{60}$$

Therefore, we obtain the dependency of $\mathbf{y}_k$ on $\mathbf{z}_k$:

$$\mathbf{y}_k = \mathbf{B}\mathbf{z}_k + \mathbf{c} \tag{61}$$

where **B** and **c** are the minimal norm solutions of the equations

$$\mathbf{G}_k\mathbf{c} = -\mathbf{g}_k \tag{62}$$

$$\mathbf{G}_k\mathbf{B} = -\mathbf{G}_k \tag{63}$$

Since we are interested in the minimal norm solution of (62) and (63) and , it follows that **B** and **c** can be calculated using the formulas

$$\mathbf{c} = -\mathbf{G}_k^T(\mathbf{G}_k\mathbf{G}_k^T)^{-1}\mathbf{g}_k = -\frac{1}{4(e_0^2 + e_1^2 + e_2^2 + e_3^2)}\mathbf{G}_k^T\mathbf{g}_k \tag{64}$$

$$\mathbf{B} = -\mathbf{G}_k^T(\mathbf{G}_k\mathbf{G}_k^T)^{-1}\mathbf{G}_k = -\frac{1}{4(e_0^2 + e_1^2 + e_2^2 + e_3^2)}\mathbf{G}_k^T\mathbf{G}_k \tag{65}$$

Substituting $\mathbf{y}_k$ in (59), we obtain

$$\Delta\mathbf{q}_k = \mathbf{U}\mathbf{z}_k + \mathbf{c} \tag{66}$$

where $\mathbf{U} = \mathbf{B} + \mathbf{E}$. After the calculation of **U** and **c** the body sends them to its child.

5.2. Subsystem level

Consider a subsystems *S*. The rows of (51), corresponding to the equations of constraints of *S*, have the form:

$$\mathbf{G}^S(\tilde{\mathbf{q}}^S)\Delta\mathbf{q}^S = -\mathbf{g}^S(\tilde{\mathbf{q}}^S) \tag{67}$$

where $\mathbf{q}^S$ is the vector of coordinates of subsystem's bodies; $\mathbf{g}^S(\mathbf{q}^S)$ is the vector of equations of subsystem's constraints; $\mathbf{G}^S = \partial \mathbf{g}^S / \partial \mathbf{q}^S$ is the Jacobian matrix of the constraints.

Let S_1, $S_2 \ldots S_k$ denote the parents of S. Obviously, $\mathbf{q}^S = \left(\mathbf{q}_b^{S_1} \quad \ldots \quad \mathbf{q}_b^{S_k}\right)^T$, where $\mathbf{q}_b^{S_i}$ is the vector of coordinates of bordering bodies of S_i. On the previous level of the hierarchy we already calculated the matrices $\mathbf{U}_b^{S_i}$, $\mathbf{c}_b^{S_i}$ for each parent S_i

$$\Delta \mathbf{q}_b^{S_i} = \mathbf{U}_b^{S_i} \mathbf{z}_b^{S_i} + \mathbf{c}_b^{S_i} \qquad \forall i = 1..k \tag{68}$$

Now we can write the formula for $\Delta \mathbf{q}^S$:

$$\Delta \mathbf{q}^S = \mathbf{U}'_b \mathbf{z}' + \mathbf{c}'_b \tag{69}$$

where

$$\begin{aligned} \mathbf{U}' &= diag(\mathbf{U}^{S_i}) \\ \mathbf{z}'_b &= \left(\mathbf{z}_b^{S_1} \quad \ldots \quad \mathbf{z}_b^{S_k}\right)^T \\ \mathbf{c}'_b &= \left(\mathbf{c}_b^{S_1} \quad \ldots \quad \mathbf{c}_b^{S_k}\right)^T \end{aligned} \tag{70}$$

Substituting this equation in (67), we get

$$\mathbf{H}\mathbf{z}' = \mathbf{b} \tag{71}$$

where $\mathbf{H} = \mathbf{G}^S \mathbf{U}'$ and $\mathbf{b} = -\mathbf{g}^S - \mathbf{G}^S \mathbf{c}'_b$.

Let $\mathbf{q}_b^S$ be the vector of coordinates of the bordering bodies in S. Let $\mathbf{q}_i^S$ be the vector of coordinates of the internal bodies in S. It is clear, that $\mathbf{q}^S$ consists of $\mathbf{q}_i^S$ and $\mathbf{q}_b^S$

$$\mathbf{q}^S = \mathbf{P}_1 \begin{pmatrix} \mathbf{q}_b^S \\ \mathbf{q}_i^S \end{pmatrix} = \begin{pmatrix} \mathbf{P}_{1,b} & \mathbf{P}_{1,i} \end{pmatrix} \begin{pmatrix} \mathbf{q}_b^S \\ \mathbf{q}_i^S \end{pmatrix} \tag{72}$$

where $\mathbf{P}_1$ is a permutation matrix. In future we will need the backward dependency:

$$\mathbf{q}_b^S = \mathbf{P}_{2,b} \mathbf{q}^S \tag{73}$$

$$\mathbf{q}_i^S = \mathbf{P}_{2,i} \mathbf{q}^S \tag{74}$$

Let us now represent $\mathbf{z}'$ as the sum of two vectors:

$$\mathbf{z}' = \mathbf{y}^S + \mathbf{z}^S \tag{75}$$

where $\mathbf{y}^S$ is the displacement, stabilizing the subsystem's constraints and $\mathbf{z}^S$ is the displacement, stabilizing equation of constraints on the higher levels of the hierarchy of subsystems. Since the internal bodies of S are not connected on the higher levels of the hierarchy, it follows that $\mathbf{z}^S$ has the structure

$$\mathbf{z}^S = \begin{pmatrix} \mathbf{P}_{1,b} & \mathbf{P}_{1,i} \end{pmatrix} \begin{pmatrix} \mathbf{z}_b^S \\ 0 \end{pmatrix} = \mathbf{P}_{1,b} \mathbf{z}_b^S \tag{76}$$

where $\mathbf{z}_b^S$ is the part of $\mathbf{z}^S$, corresponding to the displacements of bordering bodies of S.

Substituting (75) and (76) in (71), we get

$$\mathbf{H}\mathbf{y}^S = -\mathbf{P}_{1,b} \mathbf{z}_b^S + \mathbf{b} \tag{77}$$

Now we need to calculate the dependency of $\mathbf{y}^S$ on $\mathbf{z}_b^S$ in the form

$$\mathbf{y}^S = \mathbf{B}\mathbf{z}_b^S + \mathbf{d} \tag{78}$$

Since $\mathbf{y}^S$ should be small enough, it follows that the norms of matrices **B** and **c** should be minimal. That is why we calculate **B** and **c** as the part of the minimal norm solution of the equations:

$$\mathbf{H}\mathbf{X} = \mathbf{K} \tag{79}$$

where $\mathbf{X} = \begin{pmatrix} \mathbf{B} & \mathbf{d} \end{pmatrix}$ and $\mathbf{K} = \begin{pmatrix} -\mathbf{H}\mathbf{P}_{1,b} & \mathbf{b} \end{pmatrix}$.

Since the subsystem can include redundant constraints, it follows that rows of $\mathbf{G}^S$ can be linearly dependent. The matrix $\mathbf{U}'_b$ is a block-diagonal matrix, consisting of positive-semidefinite submatrices. Therefore, usually **H** is a sparse, rank-deficient matrix. We can not calculate the minimal norm solution of (79) using the QR-decomposition of **H**, but now we need to perform more costly calculations: the complete orthogonal decomposition (COD) or the singular value decomposition (SVD). The advantage of the computation of COD in comparison with the computation of SVD is that it is a non-iterative procedure, therefore, it is more suitable for the real-time simulation.

The complete orthogonal decomposition of **H** can be written as [3]:

$$\mathbf{H} = \mathbf{Q}_1 \begin{pmatrix} \mathbf{R}_1 & 0 \\ 0 & 0 \end{pmatrix} \mathbf{Q}_2^T \tag{80}$$

where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are orthogonal matrices and $\mathbf{R}_1$ is upper triangular. Then the least square solution of (79) is

$$\mathbf{X} = \mathbf{Q}_2 \begin{pmatrix} \mathbf{R}_1^{-1} & 0 \\ 0 & 0 \end{pmatrix} \mathbf{Q}_1^T \mathbf{K} \tag{81}$$

After the calculation of **B** and **d** we substitute (78) and (76) in (75) and obtain

$$\mathbf{z}' = \mathbf{y}^S + \mathbf{z}^S = \mathbf{B}\mathbf{z}_b^S + \mathbf{d} + \mathbf{P}_{1,b} \mathbf{z}_b^S = (\mathbf{B} + \mathbf{P}_{1,b}) \mathbf{z}_b^S + \mathbf{d} \tag{82}$$

Substituting $\mathbf{z}'$ in (69), we get

$$\Delta \mathbf{q}^S = \mathbf{U}\mathbf{z}_b^S + \mathbf{c} \tag{83}$$

where $\mathbf{U} = \mathbf{U}'(\mathbf{B} + \mathbf{P}_{1,b})$ and $\mathbf{c} = \mathbf{U}'\mathbf{d} + \mathbf{c}'_b$.

Finally, using (73), we express $\Delta \mathbf{q}_b^S$ as

$$\Delta \mathbf{q}_b^S = \mathbf{U}_b \mathbf{z}_b^S + \mathbf{c}_b \tag{84}$$

where $\mathbf{U}_b^S = \mathbf{P}_{2,b}\mathbf{U}^S$ and $\mathbf{c}_b^S = \mathbf{P}_{2,b}\mathbf{c}^S$.

After the calculation of $\mathbf{U}_b^S$ and $\mathbf{c}_b^S$ the subsystem sends them to its child. Also in future we will need the formula for the calculation of $\mathbf{z}'$

$$\mathbf{z}' = (\mathbf{B} + \mathbf{P}_{1,b})\mathbf{z}_b^S + \mathbf{c} \tag{85}$$

With minor changes the same procedure can be used on the highest level of the hierarchy.

During the backward hierarchical calculation of displacements each subsystem gets $\mathbf{z}_b$ from its child and calculates from (82) the vector $\mathbf{z}'$. At the end of the process each body calculates from (66) its stabilizing displacement $\Delta \mathbf{q}_k$.

6. SIMULATION OF A STEAM MACHINE

We implemented our method in VSD and performed the simulation of the high-realistic model of a three-cylinder steam machine, shown in Fig. 5. The machine was developed in Autodesk Inventor as an assembly, consisting of 13 bodies connected by 19 links.

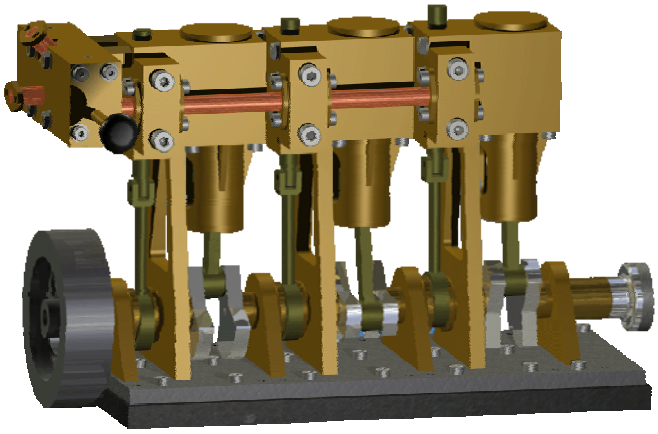

Figure 5. Autodesk Inventor model of a three-cylinder steam machine

An angular velocity of 1.96 radians per second was applied to the shaft of the mechanisms with a simulation time of 1 second.

In Fig. 6 is shown the velocity of the left piston in the z-direction. 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 Autodesk Inventor model's definition.

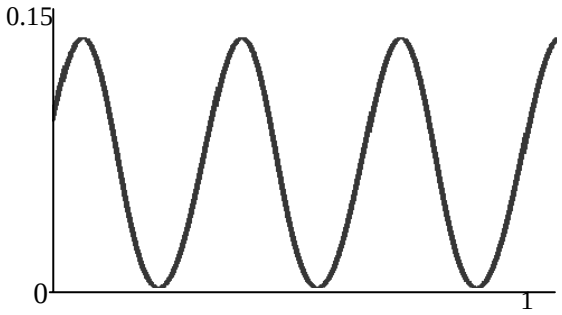

Figure 6. Velocity of the left piston in the z-direction

7. SIMULATION OF A CAR MODEL

We performed the simulation of the model of a car, shown in Fig. 7. The MODEL was developed in Autodesk Inventor as 3-level assembly, consisting of 17 bodies connected by 20 joints. A more detailed description of the model can be found in [6].

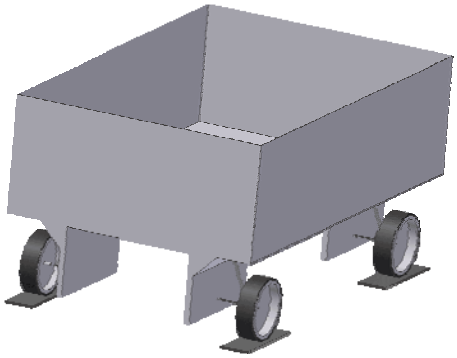

Figure 7: Autodesk Inventor car model

We performed the emulation of the passenger who gets into the car by the additional force *f* acting in *z*-direction, the value of which depends on time *t*:

$$f_z = \begin{cases} 0 & \text{when } t < 2.5 \\ 1000 \cdot (t\text{-}2.5)/0.1 & \text{when } t \in [2.5,\ \ 2.6] \\ 1000 & \text{when } t > 2.6 \end{cases} \tag{86}$$

The subsystem of the highest level of the hierarchy includes 12 joints, connecting 13 bodies. The density of the matrix **A** from (15) is 15.6% and the density of **H** from (79) is 20.4%.

Fig. 8 shows the changes of the *z*-coordinate of the car body. 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 Autodesk Inventor model's definition.

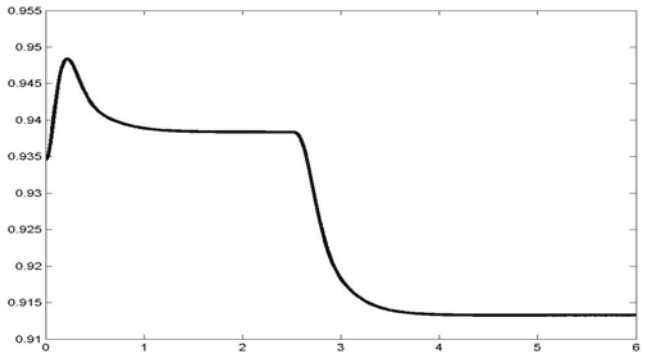

Figure 8: Z-coordinate of the car body

8. CONCLUSION AND FUTURE WORK

This paper introduces the modification of the method for the component-oriented simulation of dynamics of CAD models. The method was implemented in the VSD software, used for the simulation of the dynamics of a steam machine and of a 3D-car model. The simulation results show that method is stable and can be implemented for the wide set of multibody systems.

During the simulation of multibodies only the sparse matrices should be decomposed, that makes the method suitable for the implementation of sparse solvers or a special preprocessing module, performing a symbolic simplification of the decomposition of matrices [5]. The further integration of the preprocessing module with VSD will significantly increase the power of the software and the numerical efficiency of the software.

9. REFERENCES

[1] Bidalach, I., Portal, R., Dias, J.P., "Integration of CAD and Multibody Systems", *Proceedings of ECCOMAS Multibody Dynamics 2005*, Madrid, 21-24 June 2005

[2] Bunus, P., Engelson, V., Fritzson, P., "Mechanical Models Translation, Simulation and Visualization in Modelica", *Modelica Workshop Proceedings*, 2000

[3] Gene H. Golub, Charles F. van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins UP, 1996.

[4] Lubich, Ch., Nowak, U., Pöhle, U., Engstler, Ch.: "MEXX - Numerical software for the integration of constrained mechanical multibody systems". Mech. Struct. Mach. 23, p 473-495 (1995).

[5] Vlasenko, D., Kasper, R, "A new software approach for the simulation of multibody dynamics", *ASME Journal on Computational and Nonlinear Dynamics*, 2006, received, to appear.

[6] Vlasenko, D., Kasper, R.: *"Algorithm for Component Based Simulation of Multibody Dynamics", Technische Mechanik*, Band 26, Heft 2, (2006), pp. 92-105.

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