MULTIBODY DYNAMICS 2005, ECCOMAS Thematic Conference
C.L. Bottasso, P. Masarati, L. Trainelli (eds.)
Milano, Italy, 25–28 June 2007

SPARSE MATRIX METHOD FOR COMPONENT-ORIENTED DYNAMIC SIMULATION OF MULTIBODIES IN VSD SOFTWARE

Dmitry Vlasenko, Roland Kasper

*Institute of Mobile Systems (IMS), Otto-von-Guericke-University Magdeburg
Universitätsplatz 2, D-39016 Magdeburg, Germany
e-mails: Dmitri.Vlasenko@Masch-Bau.Uni-Magdeburg.DE,
Roland.Kasper@Masch-Bau.Uni-Magdeburg.DE

Keywords: Object-oriented simulation, multibody, QR-decomposition, sparse matrices, symbolic computations.

Abstract. *This paper presents a new algorithm for the component-oriented simulation of motion of multi-rigid-body systems, based on the decomposition of sparse matrices. The implementation of the method was sufficiently performed in **V**irtual **S**ystem **D**esigner (VSD) software, which is able to calculate the dynamics of CAD models because of its integration with a CAD tool Autodesk Inventor. The presented algorithm is well-suited for the use of a preprocessing module, performing the symbolic simplification of decompositions of sparse matrices, which significantly reduces the numerical costs of the simulation. Example of a CAD model of a double insulator chain illustrates the effectiveness of the method.*

1 INTRODUCTION

In the last years we developed and implemented the method, performing the component-oriented simulation of holonomic rigid body systems [1, 2]. In contrast to standard algorithms we preserve during the simulation the partitions of models, defined during the design of the models, 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 modelled, tested and compiled independently. This significantly decreases the time and cost of the models' development and test. Redesign and reuse of components is more effective and easier.
- 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.
- The simulation of big good-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.

We implemented the method and created an object-oriented software Virtual Systems Designer (VSD), integrated with a CAD tool Autodesk Inventor, simulating the dynamics of CAD models of mechanical systems.

Now we are strongly interested in the further increasing of the numerical efficiency of the simulation process in VSD. One of the most time-expensive routines of the simulation process in VSD is the calculation of accelerations. Some standard methods for the undistributed simulation of multibodies significantly improve their numerical efficiency taking into account the sparse structure of matrices [3]. Unfortunately, sparse methods were not effective in VSD, since the decomposition of matrices with a high non-zero density was needed.

In this article we show the improvement of the method, which performs the calculation of accelerations, based on the solution of sparse linear systems.

We also developed a preprocessing module in Maple software, performing the symbolic simplification of decompositions of matrices, which has several advantages in comparison with standard sparse solvers:

- Sparse structure of matrices is used completely without any run time overhead.
- The numerical operations with numerical elements of matrices are performed already during the translation.
- Additional operations with arrays of indexes (like in usual sparse solvers) are not needed.

We used VSD for the calculation of dynamics of a CAD model of a double insulator chain. The results show that the integration of the preprocessing module with the new version of VSD significantly reduces the computation cost of the simulation of multibodies.

2 SIMULATION STEPS

Fig. 1 shows the object-oriented method of the simulation of mechanical systems, implemented in VSD [2]. 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.

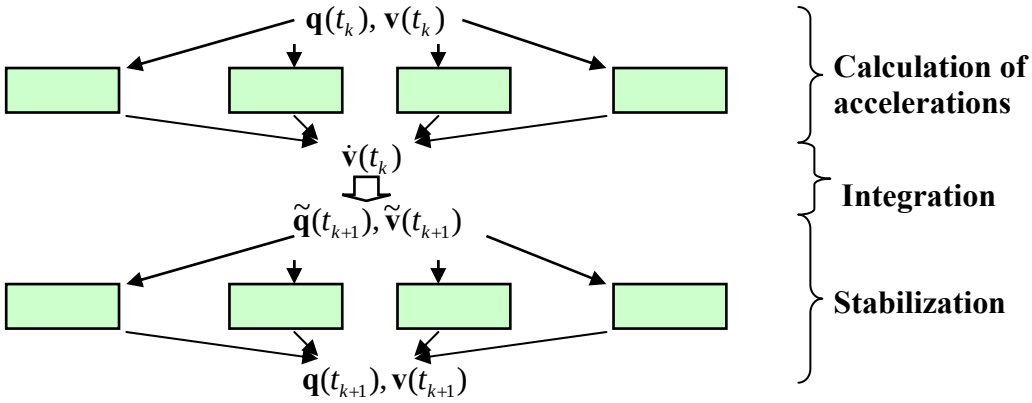

Figure 1: Data flow in simulation steps

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:

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

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

   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.

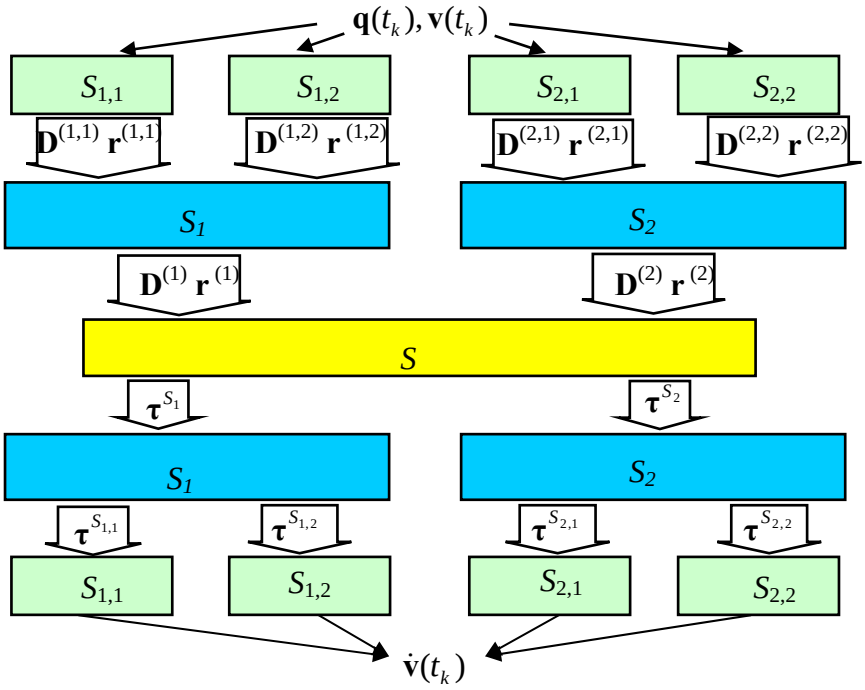

Figure 2: Hierarchical calculation of the accelerations

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 [2] 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.

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.

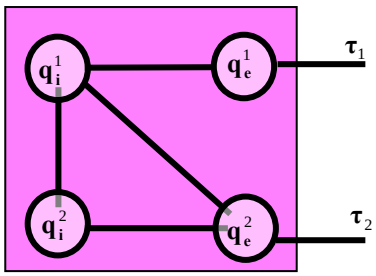

Figure 3: A subsystem of several connected bodies

Let $\mathbf{g}$ denote the c-vector of equations of internal constraints:

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

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:

$$\mathbf{q}^S = \begin{pmatrix} \mathbf{q}_b^S \\ \mathbf{q}_i^S \end{pmatrix} \tag{3}$$

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

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

Here for simplicity of notation we omit the transformation matrix $\mathbf{T}$ 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

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

$$\mathbf{M}_i \dot{\mathbf{v}}_i^S = \mathbf{f}_\mathbf{i} + \mathbf{G}_i^T \boldsymbol{\lambda} \tag{6}$$

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:

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

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

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

or, in the other form

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

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

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

$$\boldsymbol{\lambda} = \mathbf{K}\boldsymbol{\tau}^S + \mathbf{b} \tag{10}$$

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

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

where

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

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

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:

$$\mathbf{G}\mathbf{M}^{-1}\mathbf{G}^{\mathrm{T}}\mathbf{X} = \mathbf{B} \tag{14}$$

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 **G** are sparse. The standard solution, using the sparsity of the matrices [3], 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

$$\mathbf{A}^T\mathbf{A}\mathbf{X} = \mathbf{B} \tag{15}$$

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

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

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 [4]:

$$\mathbf{A}\mathbf{\Pi} = \mathbf{Q}\mathbf{R} \tag{17}$$

$$\mathbf{R} = \begin{pmatrix}\mathbf{R}_1 & \mathbf{R}_2\\ 0 & 0\end{pmatrix} \tag{18}$$

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 $\mathbf{\Pi}^T = \mathbf{\Pi}^{-1}$ and $\mathbf{Q}^T = \mathbf{Q}^{-1}$**.** Therefore, (15) can be written as:

$$\mathbf{A}^T\mathbf{A}\mathbf{X} = \mathbf{\Pi}\mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R}\mathbf{\Pi}^T\mathbf{X} = \mathbf{\Pi}\mathbf{R}^T\mathbf{R}\mathbf{\Pi}^T\mathbf{X} = \mathbf{B} \tag{19}$$

or, in the other form:

$$\mathbf{R}^T\mathbf{R}\tilde{\mathbf{X}} = \tilde{\mathbf{B}} \tag{20}$$

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

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

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

or, in the other form:

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

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

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:

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

So, from the 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:

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

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

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

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

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

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: $S_1, S_2, \ldots, S_L$ shown in Fig. 4.

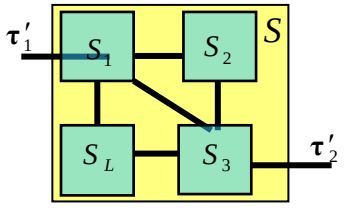

Figure 4: A subsystem consisting of several connected subsystems

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 **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 **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 **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, **K** and **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 $\hat{\mathbf{D}} = diag\left(\mathbf{D}^{(1)}, \ \ldots, \ \mathbf{D}^{(L)}\right)$, 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 **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 $\mathbf{A} = \mathbf{L}^T\boldsymbol{\Pi}^T\mathbf{G}^T$. 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 $\dot{\mathbf{v}}_{\mathbf{e}}$ 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 PREPROCESSING MODULE

As it was shown in the previous section, the most numerically expensive procedure proceeded during the distributed calculation of the accelerations is the QR-decomposition of the block-sparse matrices A from the equations (15), (43). Nowadays exists several solvers, performing the efficient numerical solution of sparse linear systems: Sparspak [5], UMFPACK [6], Yale Sparse Matrix Package [7], Harwell Subroutine Library [8].

It seems more efficient to perform a preprocessing simplification of the QR-decomposition instead of use of standard numerical methods for sparse matrices. We have developed in Maple a preprocessing module, which gets the mechanical parameters of a simulating system from Autodesk Inventor, generates correspondent matrices in symbolic form and produces the C-code, describing the matrices' decomposition. Then the C-code is compiled into .dll libraries, which are used by VSD during the simulation of dynamics of the model.

This approach has several advantages pointed out in the introduction. Generating C-code directly not only avoids calculating with zero elements, but also allows to preprocess all nu-

merical parts of expressions. Indexing of matrices is avoided completely as linear code is generated.

5 DOUBLE INSULATOR CHAIN EXAMPLE

Consider a double insulator chain example [9, 3], developed as a 3-level hierarchy of subsystems, shown in Fig. 5. Each chain consists of four insulators, connected by revolute joints. The first end of each chain is coupled with a triangular distance holder; the second is coupled with the ground. The holder is connected with a high voltage line, which is modeled as a force (shown as a red arrow), acting on the holder. We consider the situation when a bird lands on a high-voltage transmission line and added the force (shown as a blue arrow), acting on the distance holder in vertical direction.

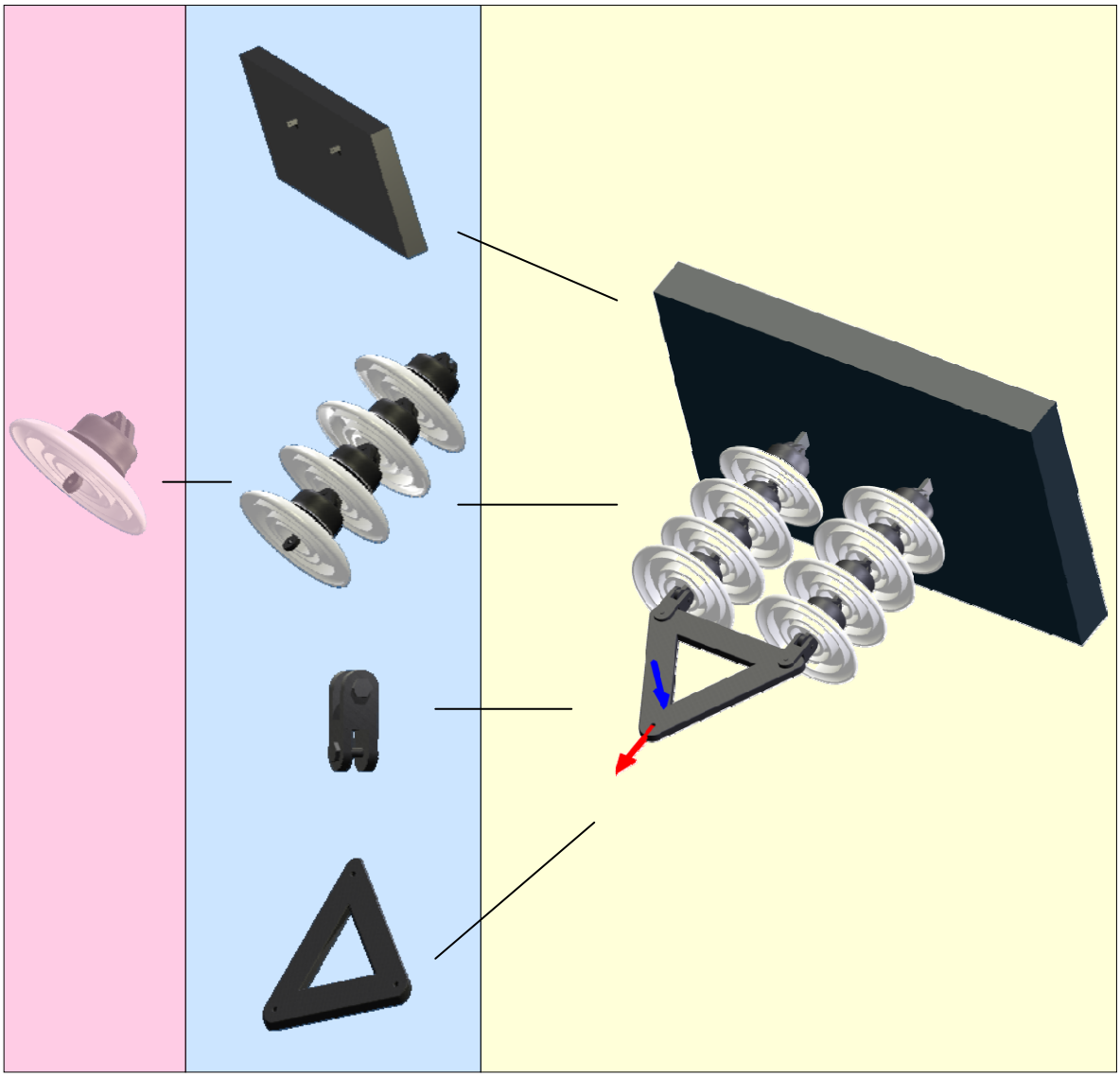

Fig. 5 Double insulator chain model

Fig. 6 shows the changes of the *z*-coordinate of the distance holder in 10 seconds.

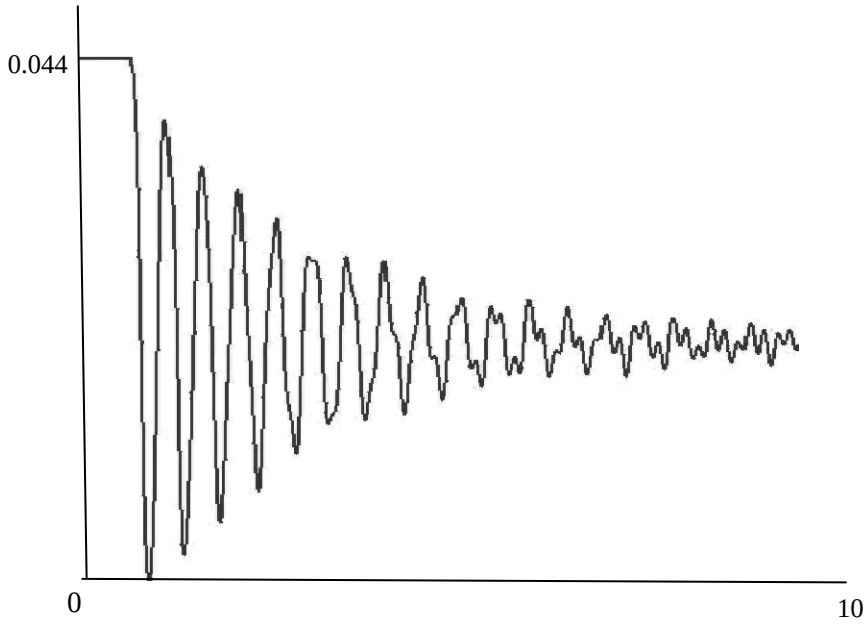

Figure 6: Z-coordinate of the triangular distance holder

Fig. 7 illustrates the structure of the matrix $\mathbf{A}_1$ of the chain submodel and the structure of the matrix $\mathbf{A}_2$ of the system of the highest level of the hierarchy. Here symbolic elements are colored black, zero elements are blank and numerical elements are non-blank.

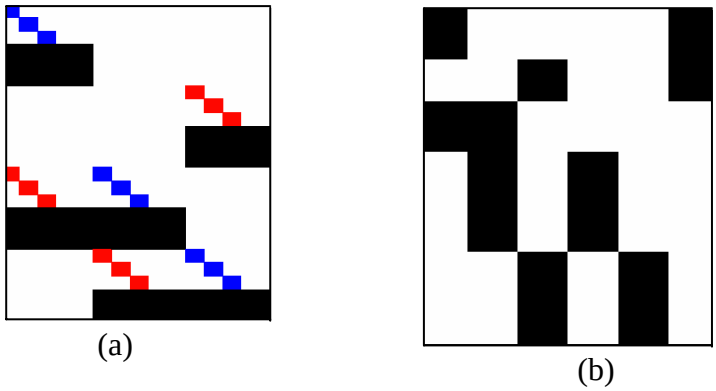

Figure 7: Matrices of the double insulator chain example, (a) $\mathbf{A}_1$ (b) $\mathbf{A}_2$

In Table 1 are shown the densities of the matrices, numerical costs of their QR-decomposition using the preprocessing module, and the numerical costs of the QR-decomposition using the dense solver.

	Density, %	Flops (VSD)	Flops (dense solver)
A_1 (chain submodel)	30	5165	14640
A_2 (highest level of the hierarchy)	33	27301	92190

Table 1 Comparison of simulation effort of double insulator chain model

From Table 1 follows that the distributed calculation of accelerations needs the decomposition only of sparse matrices, which can be efficiently calculated using the preprocessing module.

6 CONCLUSION AND FUTURE WORK

The sparse matrix method, implemented in VSD Software, performs the efficient component-oriented simulation of dynamics of CAD models of multibodies. It was shown that during the distributed calculation of accelerations only the sparse matrices should be decomposed, that makes the method suitable for the implementation of a special preprocessing module, performing the symbolic simplification of the decomposition of matrices. The results of the simulation of CAD model of a double insulator chain show the method's efficiency.

It seems needful to perform the detailed comparison of the efficiency of standard sparse solvers and the preprocessing module in future.

REFERENCES

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

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

[3] Ch. Lubich, U. Nowak, U. Pöhle, and Ch. Engstler. *MEXX- numerical software for the integration of constrained mechanical multibody systems*. Technical Report SC 92-12, Konrad-Zuse-Zentrum Berlin, 1992.

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

[5] A. George and J. W. H. Liu *Computer Solution of Large Sparse Positive Definite Systems*, PrenticeHall, 1981.

[6] http://www.cise.ufl.edu/research/sparse/umfpack/

[7] S. C. Eisenstat, M. H. Schultz and A. H. Sherman *Algorithms and data structures for sparse symmetric Gaussian elimination*, SIAM Journal on Scientific and Statistical Computing, 2(1981), pp 225-237

[8] United Kingdom Atomic Energy Authority Harwell subroutine library *A catalogue of Subroutines*, Tech. Report AERE R 9185, Harwell Laboratory, Oxfordshire OX11 0RA, Great Britain, 1988

[9] Hagedorn, P., Idelberger, H., Mocks, L: Dynamische Vorgänge bei Lastumlagerung in Abspannketten von Freileitungen. etz Archiv 2, p 109-119 (1980).