

TECHNICAL NOTES

Towards an Object-oriented Implementation of VON MISES' Motor Calculus Using Modelica

Tobias Zaiczek, Olaf Enge-Rosenblatt, Fraunhofer Institute for Integrated Circuits, Dresden, Germany

{Tobias.Zaiczek, Olaf.Engel}@eas.iis.fraunhofer.de

SNE Simulation Notes Europe SNE 20(1), 2010, 5-15, doi: 10.11128/sne.20.tn.09961

This paper deals with a first implementation of the so-called motor calculus within Modelica. The motor calculus can be used to describe the dynamical behaviour of spatial multibody systems in an efficient way. This method represents an alternative approach to modelling of multibody systems. In the paper, some fundamentals of motor calculus are summarized. Furthermore, a simple implementation of motor algebra by special additional Modelica code within some components of the Modelica Multibody Standard Library is presented. This approach fully corresponds with the paradigm of object-oriented modelling. However, the present realisation is not equation-based in its full sense because of the missing possibility of operator overloading (at least in the available Modelica simulator environment). Instead of this, some functions are used carrying out the necessary calculations. Using this implementation, some examples are given to prove the applicability and correctness of the implemented approach.

SNE 20/1, April 2010

5

Introduction

The notion of motor, composed of the words *moment* and *rotor*, was coined by Clifford in 1873 in his algebra of biquaternions [4]. But Clifford did not apply his concept neither to the modelling of motion of a single rigid body nor to the modelling of spatial multibody systems. The approach of motor calculus to 3D mechanics was suggested by von Mises in 1924 [11, 12]. In the first part [11], von Mises introduces the dual motor product. He indicates the role of the dual motor product as a measure of the instantaneous change of a motor associated to a rigid body by the action of a second motor. In the second part [12], von Mises applied the motor calculus in the derivation of a general form of the equations of motion of a rigid body. Due to this work, translations and rotations, velocities and angular velocities, forces and torques, etc. can be described by motor calculus (or motor algebra). Hence, this approach is well suited to investigate the behaviour of spatial multibody systems.

One of the authors studied motor calculus in his Diploma thesis [22] initiated and supervised by Prof. K. Reinschke from the Technical University Dresden (one of the former institutes of R. von Mises). Recent publications dealing with this subject can rarely be found (except e.g. for [8, 18]). In the context of the modelling language for heterogeneous systems Modelica (see e.g. [5, 13, 19]), the motor calculus has not been taken into account up to now.

Within the Modelica community, spatial multibody systems are usually modelled using the Modelica Multibody Standard Library (see [14] or [15]). Meanwhile, many researchers apply this library to model different kinds of – partially very complex – multibody systems [2, 9, 10, 16, 20]. This library has proven to be a well suited resource to modelling such systems. However, applying the motor calculus, the equations of motion for a rigid body become more concise and clearer, e.g.

$$\dot{\mathbf{p}} = \mathbf{f} \quad (1)$$

(\mathbf{p} – momentum motor, \mathbf{f} – force motor). Despite the formal equivalence to Newton's Second Law for a point mass, this equation fully describes the three-dimensional mechanics of a rigid body.

The motivation to follow up the motor calculus in the Modelica context is to investigate the possible simplification of handling spatial mechanical systems. A test realisation within the Modelica Multibody Standard Library has been carried out by implementing special additional Modelica code within some components of this library. These modifications take advantage of the built-in feature of inheritance. Hence, it is possible to compare both approaches e. g. with respect to numerical correctness.

In the following section, some fundamentals of motor calculus are shortly sketched. Some of the most important mathematical operations are defined. The test

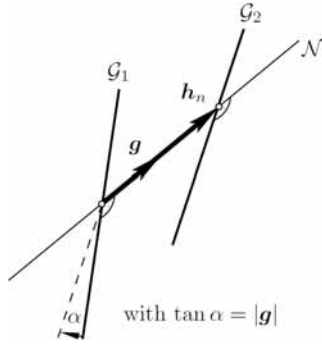


Figure 1. Geometrical interpretation of motors.

implementation is presented in Section 2. It fully corresponds with the paradigm of object-oriented modelling (see e.g. [3]). In modelling and simulation, one usually distinguishes between equations and assignments. In this context, the test implementation is not completely equation-based because some special mathematical operations had to be realised by functions. Some examples in section 3 show the principal applicability of the motor calculus approach.

1 Fundamentals of motor calculus

A motor

$$\mathfrak{h} = \begin{pmatrix} \mathbf{g} \\ \mathbf{h}_o \end{pmatrix} \quad (2)$$

is an ordered pair of vectors, \mathbf{h}_o and \mathbf{g} , that define a vector field

$$\mathbf{h}(\mathbf{r}) = \mathbf{h}_o + \mathbf{g} \times \mathbf{r} \quad (3)$$

in the three-dimensional Euclidean space. In this definition, \mathbf{r} is the position vector of any point in space, while the vectors \mathbf{h} and \mathbf{g} are called the *moment* and the *resultant vector* of the motor, respectively. Accordingly, \mathbf{h}_o stands for the *moment* of the motor at the origin O of the reference coordinate system.

For every motor, an infinite number of points exists, for which the moment of the motor \mathbf{h} is parallel to the resultant vector \mathbf{g} . All these points exhibit the same moment \mathbf{h}_n and lie on a straight line \mathcal{N} given by:

$$\mathbf{r}_n(\lambda) = \frac{\mathbf{g} \times \mathbf{h}_o}{|\mathbf{g}|^2} + \lambda \mathbf{g} \quad (4)$$

Geometrical interpretation A very strong goal of the motor calculus is the fact that motors and all operations with motors (that will be defined later on) can be interpreted as geometrical objects or construc-

tions. Hence, all motors can be seen as abstract objects that do not depend on the choice of a reference frame. R. von Mises emphasises this fact by giving the definition of motors in terms of geometrical objects describing them. Here, just an interpretation of the foregoing definition is given.

For every pair of straight lines (\mathcal{G}_1 and \mathcal{G}_2) defined in Euclidean space, there exists a straight line \mathcal{N} connecting them and being orthogonal to both of them (see Figure 1). For a pair of non-parallel lines, \mathcal{N} is uniquely defined. Otherwise, there exists an infinite number of such connecting lines that are parallel to each other. Now, every ordered pair of straight lines (\mathcal{G}_1 , \mathcal{G}_2) can be mapped to a motor (see Figure 1). In this case, \mathcal{N} is denoted as *motor axis*, according to von Mises. The oriented segment of the axis \mathcal{N} between the intersection with \mathcal{G}_1 and the intersection with \mathcal{G}_2 can be interpreted as the moment \mathbf{h}_n of the motor on its axis. The smaller one of both angles included by the lines \mathcal{G}_1 and \mathcal{G}_2 is understood as a measure for the orientation and is simultaneously interpreted as the length of the resultant vector \mathbf{g} . The tangent of this angle α is equal to the length of the resultant vector, while the direction of the resultant vector is defined in such a manner that \mathcal{G}_1 can be transferred into \mathcal{G}_2 by a mathematically positive screw motion across the resultant vector. The mapping from an ordered pair of straight lines to a motor is not a one-to-one mapping because all ordered pairs of straight lines that can be transferred into each other by a screw motion across \mathcal{N} define the same motor.

1.1 Motor calculus

In the following, some computational rules of motor calculus are recalled.

Let \mathfrak{h} , \mathfrak{h}_1 and \mathfrak{h}_2 be three motors given by

$$\mathfrak{h} = \begin{pmatrix} \mathbf{g} \\ \mathbf{h}_o \end{pmatrix}, \quad \mathfrak{h}_1 = \begin{pmatrix} \mathbf{g}_1 \\ \mathbf{h}_{o1} \end{pmatrix}, \quad \mathfrak{h}_2 = \begin{pmatrix} \mathbf{g}_2 \\ \mathbf{h}_{o2} \end{pmatrix} \quad (5)$$

Then, according to von Mises, the following mathematical operations can be defined:

Addition

The addition of motors is performed component-wise according to:

$$\mathfrak{h}_1 + \mathfrak{h}_2 = \begin{pmatrix} \mathbf{g}_1 + \mathbf{g}_2 \\ \mathbf{h}_{o1} + \mathbf{h}_{o2} \end{pmatrix} \quad (6)$$

The neutral element of the addition is the zero motor

$$\mathbf{o} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix} \quad (7)$$

Multiplication

For the multiplication of motors the following three cases can be distinguished:

Multiplication with a scalar The scalar multiplication is defined component-wise

$$\alpha \mathbf{h} = \begin{pmatrix} \alpha \mathbf{g} \\ \alpha \mathbf{h}_o \end{pmatrix} \quad (8)$$

Inner product The result of the inner product of two motors is a scalar. Thus, the product corresponds to the scalar product of the vector calculus. The definition is

$$(\mathbf{h}_1, \mathbf{h}_2) = (\mathbf{g}_1, \mathbf{h}_{o2}) + (\mathbf{g}_2, \mathbf{h}_{o1}) \quad (9)$$

while (\mathbf{g}, \mathbf{h}) is the scalar product of two vectors. Using matrix notation, the equation

$$(\mathbf{h}_1, \mathbf{h}_2) = \begin{pmatrix} \mathbf{g}_1^T & \mathbf{h}_{o1}^T \end{pmatrix} \Gamma \begin{pmatrix} \mathbf{g}_2 \\ \mathbf{h}_{o2} \end{pmatrix}$$

holds, where Γ is a well-chosen matrix according to:

$$\Gamma = \begin{pmatrix} \mathbf{0} & \mathbf{I}_3 \\ \mathbf{I}_3 & \mathbf{0} \end{pmatrix}$$

and \mathbf{I}_3 denotes the (3×3) identity matrix.

Outer product The outer product of two motors results in another motor, which is composed as follows:

$$\mathbf{h}_1 \times \mathbf{h}_2 = \begin{pmatrix} \mathbf{g}_1 \times \mathbf{g}_2 \\ \mathbf{g}_1 \times \mathbf{h}_{o2} + \mathbf{h}_{o1} \times \mathbf{g}_2 \end{pmatrix} \quad (10)$$

The outer product is also referred to as motorial product or as dual motor product [6]. In terms of vectors and vector dyads, the product can be written as

$$\mathbf{h}_1 \times \mathbf{h}_2 = \begin{pmatrix} \mathbf{0} & \mathbf{G}_1 \\ \mathbf{G}_1 & \mathbf{H}_{o1} \end{pmatrix} \Gamma \begin{pmatrix} \mathbf{g}_2 \\ \mathbf{h}_{o2} \end{pmatrix} \quad (11)$$

where \mathbf{G}_1 and \mathbf{H}_{o1} are the cross product matrices of the vectors \mathbf{g}_1 and \mathbf{h}_{o1} , respectively.

Motor dyads

In analogy to the vector calculus, von Mises declared dyads for the motor calculus by linear vector functions mapping motors to motors. Referred to a concrete coordinate system, such a dyad can be represented as a (6×6) matrix.

The mapping can be described in the following manner:

$$\mathfrak{T} \circ \mathbf{h}_1 = \begin{pmatrix} \mathbf{T}_{11} & \mathbf{T}_{12} \\ \mathbf{T}_{21} & \mathbf{T}_{22} \end{pmatrix} \Gamma \begin{pmatrix} \mathbf{g}_1 \\ \mathbf{h}_{o1} \end{pmatrix} = \begin{pmatrix} \mathbf{T}_{11}\mathbf{h}_{o1} + \mathbf{T}_{12}\mathbf{g}_1 \\ \mathbf{T}_{21}\mathbf{h}_{o1} + \mathbf{T}_{22}\mathbf{g}_1 \end{pmatrix}. \quad (12)$$

The neutral element of the dyadic multiplication in motor calculus is the identity motor dyad \mathfrak{G} that can be represented in every frame as

$$\mathfrak{G} = \begin{pmatrix} \mathbf{0} & \mathbf{I}_3 \\ \mathbf{I}_3 & \mathbf{0} \end{pmatrix} \quad (13)$$

Now, all calculation rules for the motor calculus can be derived readily, some of which are presented here for any arbitrarily chosen motors $\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3$ and $\alpha \in \mathbb{R}$:

$$\begin{aligned} (\mathbf{h}_1, \mathbf{h}_2) &= (\mathbf{h}_2, \mathbf{h}_1) \\ \mathbf{h}_1 \times \mathbf{h}_2 &= -\mathbf{h}_2 \times \mathbf{h}_1 \\ (\mathbf{h}_1, (\mathbf{h}_2 + \mathbf{h}_3)) &= (\mathbf{h}_1, \mathbf{h}_2) + (\mathbf{h}_1, \mathbf{h}_3) \\ \mathbf{h}_1 \times (\mathbf{h}_2 + \mathbf{h}_3) &= \mathbf{h}_1 \times \mathbf{h}_2 + \mathbf{h}_1 \times \mathbf{h}_3 \\ (\alpha \mathbf{h}_1, \mathbf{h}_2) &= \alpha (\mathbf{h}_1, \mathbf{h}_2) \\ \alpha \mathbf{h}_1 \times \mathbf{h}_2 &= \alpha (\mathbf{h}_1 \times \mathbf{h}_2) \\ (\mathbf{h}_1, \mathbf{h}_2 \times \mathbf{h}_3) &= (\mathbf{h}_2, \mathbf{h}_3 \times \mathbf{h}_1) = (\mathbf{h}_3, \mathbf{h}_1 \times \mathbf{h}_2) \end{aligned} \quad (14)$$

Remark: Due to the definition of addition and scalar multiplication, motors span a vector space over the field of real numbers. Moreover, by the introduction of the outer product, motors form a Lie-Algebra (Named after the mathematician Sophus Lie *1842, †1899) since all the following conditions are fulfilled:

1. The bilinearity of the motorial product is given, i. e., for all real α and β , the motors $\mathbf{h}_1, \mathbf{h}_2$, and \mathbf{h}_3 satisfy the equations

$$(\alpha \mathbf{h}_1 + \beta \mathbf{h}_2) \times \mathbf{h}_3 = \alpha \mathbf{h}_1 \times \mathbf{h}_3 + \beta \mathbf{h}_2 \times \mathbf{h}_3$$
 and

$$\mathbf{h}_1 \times (\alpha \mathbf{h}_2 + \beta \mathbf{h}_3) = \alpha \mathbf{h}_1 \times \mathbf{h}_2 + \beta \mathbf{h}_1 \times \mathbf{h}_3$$
2. The motorial multiplication is skew commutative, i. e.,

$$\mathbf{h}_1 \times \mathbf{h}_2 = -\mathbf{h}_2 \times \mathbf{h}_1$$
3. The Jacobian Identity holds, i. e. for arbitrarily chosen motors $\mathbf{h}_1, \mathbf{h}_2$, and \mathbf{h}_3 , the equation:

$$\mathbf{h}_1 \times (\mathbf{h}_2 \times \mathbf{h}_3) + \mathbf{h}_2 \times (\mathbf{h}_3 \times \mathbf{h}_1) + \mathbf{h}_3 \times (\mathbf{h}_1 \times \mathbf{h}_2) = \mathbf{0}$$
 is true.

Coordinate Transformations

For concrete calculations with motors, it is necessary to introduce a coordinate system, also called frame, in which the components of the motor are given. Considering two different frames \mathcal{F}_1 and \mathcal{F}_2 , it may be of interest how to transform the components of a motor

\mathfrak{h} given in frame \mathcal{F}_1 into the components referred to frame \mathcal{F}_2 and vice versa. So let vector r_{12} denote the position vector of the origin of \mathcal{F}_2 declared in frame \mathcal{F}_1 . Furthermore, let the rotation from frame \mathcal{F}_1 to frame \mathcal{F}_2 be given by the direction cosine matrix A . Then, the transformation is performed by the equation

$$[\mathfrak{h}]_{\mathcal{F}_2} = \left[\begin{pmatrix} \mathbf{0} & A \\ A & -A\mathbf{R}_{12} \end{pmatrix} \circ \mathfrak{h} \right]_{\mathcal{F}_1} \quad (15)$$

Here, the matrix \mathbf{R}_{12} is the cross product matrix of the vector r_{12} .

Differentiation with respect to real-valued parameters

Consider a motor \mathfrak{h} that depends on a real parameter t (e. g. the time). Then, the first derivative of this motor with respect to t can be computed component-wise:

$$\frac{d\mathfrak{h}}{dt} = \left(\frac{d\mathbf{g}}{dt} \quad \frac{d\mathbf{h}_o}{dt} \right)^T \quad (16)$$

Differentiation in moving frames

The temporal change of a motor seen from two different frames will, in general, lead to differing results if one frame, say \mathcal{F}_1 , moves relatively to the other frame, say \mathcal{F}_0 . The relative motion of the origin of frame \mathcal{F}_1 measured in frame \mathcal{F}_0 shall be given by the velocity vector \underline{v}_0 , while the angular velocity vector of frame \mathcal{F}_1 with respect to frame \mathcal{F}_0 is denoted by $\underline{\omega}$. Then, the equation

$$\dot{\mathfrak{h}} = \dot{\mathfrak{h}} + \begin{pmatrix} \underline{\omega} \\ \underline{v}_0 \end{pmatrix} \times \mathfrak{h} \quad (17)$$

holds for the derivation with respect to time observed in frame \mathcal{F}_0 . In Eq. (6), $\dot{\mathfrak{h}}$ denotes the derivation w. r. t. time of the motor \mathfrak{h} observed in frame \mathcal{F}_1 .

1.2 Applications of motor calculus

The most important application of motor calculus is the description and analysis of the static and dynamic behaviour of rigid bodies subject to external forces and torques. Following the ideas of von Mises, the next paragraphs will give an overview, how to describe the rigid body movements in the three-dimensional space in a very effective way using the motor calculus.

Before that, some definitions have to be explained that are essential for the succeeding subsections. To describe the motion of a rigid body in three-dimensional space, one chooses a reference point O of the bo-

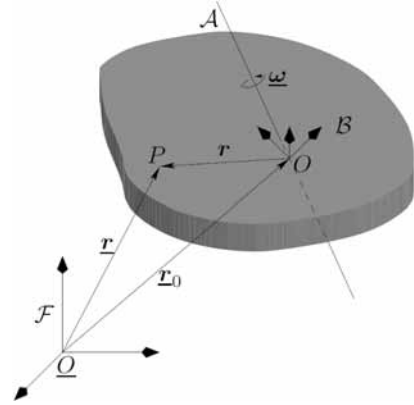


Figure 2. Definition of vectors at the rigid body.

dy. The motion of point O can be expressed w. r. t. a reference frame \mathcal{F} by the position vector \underline{r}_0 (see Figure 2). The origin of frame \mathcal{F} is denoted by \underline{Q} .

For the description of all other points of the rigid body, it is suitable to introduce a body fixed frame, called body frame \mathcal{B} , with the origin located in the reference point O . To distinguish the position vectors of both frames, the position vectors of the inertial frame are underlined. The position of an arbitrarily chosen point P of the body is therefore given by

$$\underline{r} = \underline{r}_0 + \underline{r} \quad (18)$$

The motion of the rigid body is fully described by the velocity of the reference point $\underline{v}_0 = \dot{\underline{r}}_0$ and the angular velocity $\underline{\omega}$ the body frame \mathcal{B} is rotating w. r. t. \mathcal{I} .

Definition of physically motivated motors

The introduction of motor calculus is justified by the comfortable applicability to mechanical rigid body issues in three-dimensional space. As already described before, some physical quantities for the description of rigid body movements can be composed to motors. Hence, the motion laws of rigid body mechanics can be written in a very compact and clear form. This will be shown in the subsequent paragraphs.

We introduce some motors that are able to describe the motion sequence of a rigid body as well as the acting torques and forces in a physically meaningful manner.

The first motor is called the force motor \mathfrak{f} combining the resulting force \underline{f} and torque \underline{d}_o (referred to the reference point O) acting on the rigid body, i. e.

$$\mathfrak{f} = \begin{pmatrix} \underline{f} \\ \underline{d}_o \end{pmatrix} \quad (19)$$

For any rigid body, every single force \mathbf{f}_i and torque \mathbf{d}_j can be assigned to a force motor according to

$$\mathbf{f}_{f,i} = \begin{pmatrix} \mathbf{f}_i \\ \mathbf{r}_i \times \mathbf{f}_i \end{pmatrix} \quad \text{and} \quad \mathbf{f}_{d,j} = \begin{pmatrix} \mathbf{0} \\ \mathbf{d}_j \end{pmatrix} \quad (20)$$

respectively. The resulting force motor can then be simply calculated as the sum of all single force motors

$$\mathbf{f} = \sum_{(i)} \mathbf{f}_{f,i} + \sum_{(j)} \mathbf{f}_{d,j} \quad (21)$$

Please note that the overall torque do as well as the representation of the motor depend upon the chosen reference point O . Hence, the torque referred to any other point with the position vector \mathbf{r} is calculated by

$$\mathbf{d}(\mathbf{r}) = \mathbf{d}_o + \mathbf{f} \times \mathbf{r} \quad (22)$$

That is exactly the relationship stated in Eq. (1). This characteristic can be interpreted as a force screw (see e.g. [1]), since there always exists an instantaneous line on which the force and the torque vectors act parallel. A second motor, the so-called velocity motor, is able to describe the whole motion of a rigid body. It consists of the velocity vector \mathbf{v}_o of the chosen reference point O and the angular velocity vector $\underline{\omega}$ representing the rotation of the body w. r. t. an inertial frame:

$$\mathbf{v} = \begin{pmatrix} \underline{\omega} \\ \mathbf{v}_o \end{pmatrix} \quad (23)$$

This motor is able to describe the velocity \mathbf{v} of any point \mathbf{r} of the rigid body by the equation

$$\mathbf{v}(\mathbf{r}) = \mathbf{v}_o + \underline{\omega} \times \mathbf{r} \quad (24)$$

Two other important vectors in the description of dynamic mechanical systems are the momentum vector \mathbf{p} and the angular momentum vector \mathbf{l}_o . Both are combined in the momentum motor \mathbf{p} with

$$\mathbf{p} = \begin{pmatrix} \mathbf{p} \\ \mathbf{l}_o \end{pmatrix} \quad (25)$$

Similar to the force motor, the representation of momentum motor depends upon the chosen reference point. Between the angular momentum \mathbf{l}_o referred to O and the angular momentum vector $\mathbf{l}(\mathbf{r})$ referred to any other point at position \mathbf{r} , the relationship

$$\mathbf{l}(\mathbf{r}) = \mathbf{l}_o + \mathbf{p} \times \mathbf{r} \quad (26)$$

holds. This statement can be proven by using the definition of the vectors \mathbf{p} and \mathbf{l}_o according to

$$\begin{aligned} \mathbf{p} &= \int \dot{\mathbf{r}} \, dm = \dot{\mathbf{r}}_o \int dm + \underline{\omega} \times \int \mathbf{r} \, dm \\ &= m\dot{\mathbf{r}}_o - m\mathbf{r}_s \times \underline{\omega} = m\mathbf{v}_o - m\mathbf{r}_s \times \underline{\omega} \end{aligned} \quad (27)$$

$$\begin{aligned} \mathbf{l}_o &= \int \mathbf{r} \times \dot{\mathbf{r}} \, dm = \int \mathbf{r} \, dm \times \dot{\mathbf{r}}_o + \int \mathbf{r} \times (\underline{\omega} \times \mathbf{r}) \, dm \\ &= m\mathbf{r}_s \times \mathbf{v}_o + \boldsymbol{\Theta}_o \underline{\omega} \end{aligned} \quad (28)$$

where m denotes the mass of the body and $\boldsymbol{\Theta}_o$ the inertia tensor w. r. t. the reference point O . The vector \mathbf{r}_s is the position vector of the centre of mass referred to the body frame given by $\mathbf{r}_s = \int \mathbf{r} \, dm / \int dm$.

Some fundamental laws of mechanics in terms of motor calculus

With the definitions above, a relationship between the velocity motor \mathbf{v} and the momentum motor \mathbf{p} can be derived by introducing the inertia dyad \mathfrak{M} for the motor calculus:

$$\mathbf{p} = \mathfrak{M} \circ \mathbf{v} = \begin{pmatrix} m\mathbf{I} & -m\mathbf{R}_s \\ m\mathbf{R}_s & \boldsymbol{\Theta}_o \end{pmatrix} \circ \mathbf{v} \quad (29)$$

The new symbol \mathbf{R}_s describes the cross product dyad of the vector \mathbf{r}_s .

Referred to a concrete frame in u , v and w , the dyad can be written as a (6×6) matrix of the following form:

$$\mathfrak{M} = \begin{pmatrix} m & 0 & 0 & 0 & mw_s & -mv_s \\ 0 & m & 0 & -mw_s & 0 & mu_s \\ 0 & 0 & m & mv_s & -mu_s & 0 \\ 0 & -mw_s & mv_s & \Theta_{uu} & \Theta_{uv} & \Theta_{uw} \\ mw_s & 0 & -mu_s & \Theta_{vu} & \Theta_{vv} & \Theta_{vw} \\ -mv_s & mu_s & 0 & \Theta_{wu} & \Theta_{wv} & \Theta_{ww} \end{pmatrix}$$

where u_s , v_s , and w_s are the coordinates of centre of mass. Choosing the body frame parallel to the body's principal axes of inertia and selecting the centre of mass as the reference point, \mathfrak{M} becomes a diagonal matrix. With the help of the foregoing motor relations, the main mechanical laws can be rewritten in terms of motors.

The first law describes the change of momentum and angular momentum in the presence of external forces and torques in a very efficient and short way, namely

$$\dot{\mathbf{p}} = \mathbf{f}$$

Here, $\dot{\mathbf{p}}$ denotes the time derivative of the momentum motor \mathbf{p} observed in an *inertially fixed* reference frame.

The unique simplicity and shortness of this equation is doubtless a goal of this calculus, even more considering that it formally takes exactly the form of Newton's Second Law for mass points. Unfortunately, this formula is not very practical, since the derivation has to be done w. r. t. the inertial frame. However, the

momentum motor is much easier to determine in a body fixed frame, because the inertia dyad \mathcal{M} is therein constant. So, a much more applicable form for concrete calculations can be derived using (6) to express the time derivation w. r. t. the body frame

$$\dot{\mathbf{p}} + \mathbf{v} \times \mathbf{p} = \mathbf{f} \quad (30)$$

where \mathbf{p} , \mathbf{v} , and \mathbf{f} are referred to the origin of the body frame.

Replacement of the momentum motor using Eq. (7) yields the following relationship:

$$\mathcal{M} \circ \dot{\mathbf{v}} + \mathbf{v} \times (\mathcal{M} \circ \mathbf{v}) = \mathbf{f} \quad (31)$$

The kinetic energy of a rigid body can be expressed by means of motor calculus as follows:

$$T = \frac{1}{2}(\mathbf{v}, \mathbf{p}) \quad \text{with} \quad \mathbf{p} = \mathcal{M} \circ \mathbf{v} \quad (32)$$

Again, this expression agrees formally with the equation of the kinetic energy of a mass point, if therein the mass is substituted by the inertia dyad \mathcal{M} and the vectors are substituted by their corresponding motors.

Similarly, the equation for the power performed by the applied forces and torques is given by

$$P = (\mathbf{f}, \mathbf{v}) \quad (33)$$

so that the energy law for a rigid body results in

$$\frac{dT}{dt} = \frac{1}{2} \frac{d}{dt}(\mathbf{v}, \mathcal{M} \circ \mathbf{v}) = (\mathbf{f}, \mathbf{v}) \quad (34)$$

2 Object-oriented implementation

The test implementation presented here is based on the Modelica Multibody Standard Library. Hence, it fully corresponds with the paradigm of object-oriented modelling. Due to some limitations of the Modelica language, compromises had to be made during implementation of the motor calculus. Because of the necessarily used functions, the realisation is not a completely equation-based formulation.

2.1 Motor library

The first step of the implementation towards a description of rigid body motion by means of motor calculus is the realisation of a general motor class. From the view of data structure, motors are nothing more than a combination of six scalars. According to the definition of motors provided above, the first idea of arranging these scalars within the motor class was to group them into two vectors of type *Real*. The first vector would represent the resultant vector and the second vector would be the moment vector of the

motor at the reference point:

```
record Motor "Motor"
  Real[3] res "resultant vector";
  Real[3] mom "moment vector at O";
end Motor;
```

Unfortunately, this approach, similar to the implementation of the complex numbers in [7], prohibits the use of basic mathematical operators on the newly defined data types. A solution would be the overloading of these operators as it is possible in C++ [17]. However, Modelica does still not support this feature. Thus, an alternative implementation has been chosen, where all six scalars are stored within one vector:

```
type Motor = Real[6]
  "Motor: [Resultant; Moment at r0]";
```

The reason for the chosen implementation was the ability to keep at least the operators “+” and “−” as well as the multiplication with scalars for the motor calculus in its original sense. Within the context of inheritance, no real specialization concerning the physical units of the quantities can be made. Hence, the child classes of velocity motor, force motor, and momentum motor have also a quite simple definition, namely:

```
type VelocityMotor = Motor "Velocity motor";
type ForceMotor = Motor "Force motor";
type MomentumMotor = Motor "Momentum motor";
```

All the other calculation rules introduced in section 1.1 had to be implemented using Modelica functions. The first function has been written to perform the inner product between two motors according to Eq. (2):

```
function dot "Inner product of motor calculus"
  input Motor m1 "First motor";
  input Motor m2 "Second motor";
  output Real r3 "Resulting scalar";
algorithm
  r3 := m1[1:3]*m2[4:6] + m1[4:6]*m2[1:3];
end dot;
```

Similarly, the outer product has been implemented as stated in Eq. (3):

```
function 'x' "Outer product of motor calculus"
  input Motor m1 "First motor";
  input Motor m2 "Second motor";
  output Motor m3 "Resulting motor";
algorithm
  m3 := vector([cross(m1[1:3], m2[1:3]);
               cross(m1[1:3], m2[4:6]);
               cross(m1[4:6], m2[1:3])]);
end 'x';
```

A function that returns the moment of the motor for any position vector \mathbf{r} has also been realised to simplify the motor handling:

```

function mom "Moment of the motor referred to position r"
  input Motor m "Motor";
  input Modelica.SIunits.Position[3] r
    "Position vector";
  output Real[3] mom "Moment of the motor";
  algorithm
    mom := m[4:6] + cross(m[1:3], r);
  end mom;

```

The foregoing reasons for the simple implementation of the motor class apply for the implementation of the motor dyads, too. Hence, a motor dyad given w. r. t. a given frame can be expressed as a (6×6) matrix:

```

type MotorDyad = Real[6,6] "Motor Dyad";

```

To apply a motor dyad to a motor, another function has been created. Referring to Eq. (4), the function has been defined by:

```

function times "Application of a Motor Dyad on Motor"
  input MotorDyad m1 "Motor dyad to be applied";
  input Motor m2 "Input motor";
  output Motor m3 "Output motor";
  algorithm
    m3 := m1[:,1:3]*m2[4:6] + m1[:,4:6]*m2[1:3];
  end times;

```

Finally, there exist two functions that are able to transform the components of a motor from one frame to another and vice versa (refer to section 2.1.4):

```

function coordChange1
  "Transforms motor from frame a to frame b"
  import F= Modelica.Mechanics.MultiBody.Frames;
  input Modelica.SIunits.Position[3] r_0
    "Vector pointing from origin of frame a to
    origin of frame b, resolved in frame a";
  input F.Orientation R
    "Orientation object of frame b resolved in frame a";
  input Motor m1 "Motor resolved in frame a";
  output Motor m2 "Motor resolved in frame b";
  algorithm
    m2 := vector([R.T*m1[1:3]; R.T*mom(m1, r_0)]);
  end coordChange1;

function coordChange2
  "Transforms motor from frame b to frame a"
  import F= Modelica.Mechanics.MultiBody.Frames;
  input Modelica.SIunits.Position[3] r_0
    "Vector pointing from origin of frame a to
    origin of frame b, resolved in frame a";
  input F.Orientation R
    "Orientation object of frame b resolved in frame a";
  input Motor m1 "Motor resolved in frame a";
  output Motor m2 "Motor resolved in frame b";
  algorithm
    m2 := vector([transpose(R.T)*m1[1:3];
      transpose(R.T)*m1[4:6]
      + cross(r_0, transpose(R.T)*m1[1:3])]);
  end coordChange2;

```

2.2 Multibody implementation

After implementing the most important operations of the motor calculus, we were able to take advantage of the efficient description of the rigid body motion. Therefore, as a first step, the existing implementation of a rigid body object from the Modelica Multibody Standard Library was adapted to the motor algebra. To simplify the implementation, all interfaces and all existing variables were kept. Only some small changes had to be made within the so-called Body class. The first changes were the declaration of the following physically motivated Motor and MotorDyad objects:

```

// Motor Dyads
Real[3,3] I0 "Inertia dyad wrt. B";
MotorDyad I_mot "Motorial inertia dyad wrt. B";

// Motors
VelocityMotor vel_B "Velocity motor wrt. B";
MomentumMotor mom "Momentum motor wrt. B";
ForceMotor f_g "Gravity force motor wrt. B";
ForceMotor f_a "Cut force motor wrt. B";

```

Afterwards, all declared motors and motor dyads had to be defined using the following statements:

```

// force motors
f_g = vector([ m*frame_a.R.T*g_0;
  cross(r_CM, m*frame_a.R.T*g_0)]);
f_a = vector([frame_a.f; frame_a.t]);

// velocity motor
vel_B = vector([ frame_a.R.w;
  frame_a.R.T*der(frame_a.r_0)]);

// inertia matrices
I0 = I + m*(diagonal(r_CM*r_CM*ones(3))
  - [r_CM]*transpose([r_CM]));
I_mot = [diagonal({m, m, m}), -skew(m*r_CM);
  skew(m*r_CM), I0];

// momentum motor
mom = vector(times(I_mot, vel_B));

```

Finally, the equations of motion originally implemented according to

```

frame_a.f = m*(Frames.resolve2(frame_a.R,
  a_0 - g_0)
  + cross(z_a, r_CM)
  + cross(w_a, cross(w_a, r_CM)));
frame_a.t = I*z_a + cross(w_a, I*w_a)
  + cross(r_CM, frame_a.f);

```

have been replaced by the very clear and short Eq. (8):

```

f_a = der(mom) + 'x'(vel_B, mom) - f_g;

```

Because of the object-oriented structure of the Modelica Standard Library, the changes had to be imple-

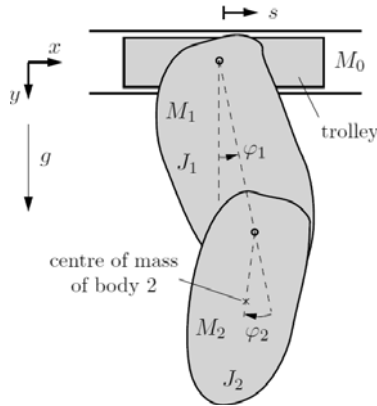


Figure 3. Sketch of double pendulum.

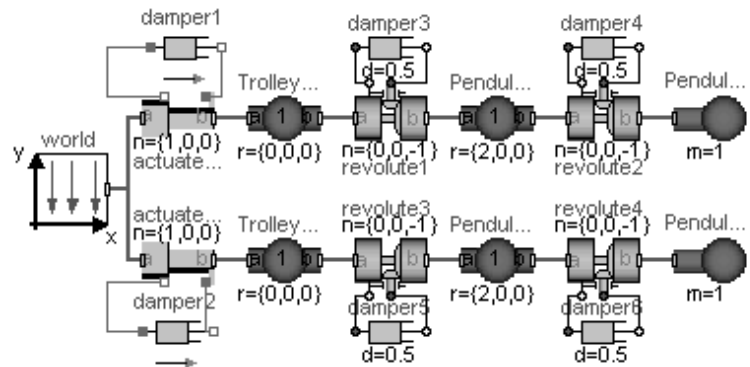


Figure 5. Implementation of the double pendulum.

mented only once. All subclasses of the Body class, like BodyShape, BodyBox, or BodyCylinder inherit the changes automatically.

3 Examples and verification

As a first example, the movable double pendulum (Fig. 3) was chosen to show the correctness of the implemented body classes based on motor calculus. The pendulum consists of a trolley with the mass M_0 and two rigid bodies with masses M_1 and M_2 . The trolley is able to move horizontally. The first body is suspended on the trolley by a revolute joint. The second body is suspended on the first body via a revolute joint, too. Both axes of rotation are parallel to the z -axis which lies perpendicular to the xy -plane (see Figure 3). The moments of inertia of both bodies around the axis of rotation w. r. t. their particular centre of mass are given by J_1 and J_2 .

The pendulum moves from an initial deflection of $\varphi_1(0) = 90^\circ$ and $\varphi_2(0) = 0^\circ$ due to the earth's gravity field. A viscous friction, acting in every joint, damps the motion of the pendulum. As a reference, the same pendulum system has been implemented using the Modelica Standard Library. A sketch of the structure is shown in the lower part of Figure 5. The upper part of this figure shows the pendulum using the modified Body objects adapted to the motor calculus.

Figure 4 shows the trajectory for the positions of the trolley. Figures 6 and 7 depict the time histories of the revolute joint angles φ_1 and φ_2 . In every diagram, the trajectory of both systems, the double pendulum using the motor calculus and the double pendulum using the Modelica Standard Library, were plotted together.

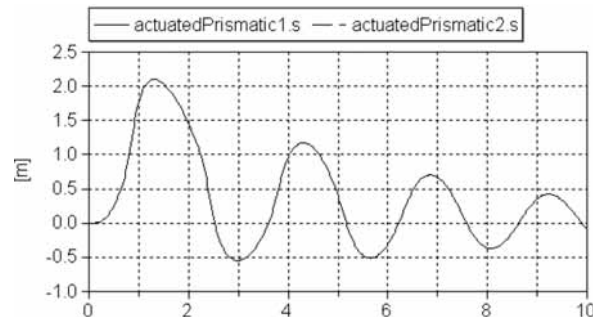


Figure 4. Trajectory of the trolley position s .

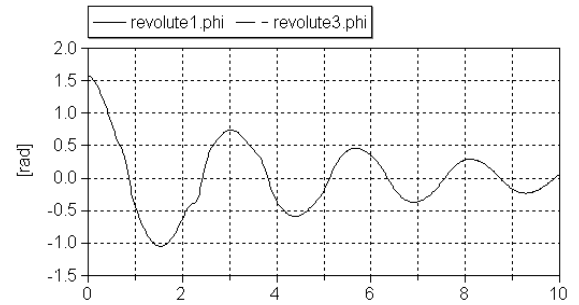


Figure 6. Trajectory of the first pendulum angle φ_1 .

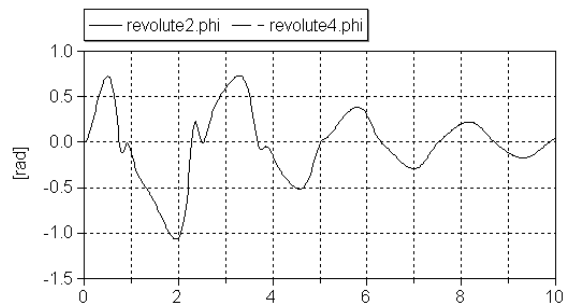


Figure 7. Trajectory of the first pendulum angle φ_2 .

Figure 8 presents the deviation for all corresponding variables (residue_s , $\text{residue_phi1}/2$). Apparently, the deviation of the position stays smaller than $1.2 \cdot 10^{-11} \text{ m}$ for the given simulation time of 10s. The deviations of both pendulum angles are also very small. They do not exceed 10^{-11} rad . Hence, these differences can be interpreted as numerical errors of the simulator, because for simulations with a lower error tolerance, the deviations decrease. For a comparison even a third implementation within the simulation system Matlab (refer to [21]) was consulted that led to very similar results.

3.1 Fourfold pendulum on two movable sliders

The second example is a fourfold pendulum. It consists of two trolleys and a chain of four rigid bodies between them. Both trolleys are guided along straight tracks (see Figure 10). Hence, this example contains a *closed kinematic loop*. Similar to the foregoing example, the pendulum moves from an initial deflection due to the gravity field of the earth and is damped by a viscous friction in every joint. The initial values for the pendulum angles are

$$\begin{aligned} \varphi_1(0) &= 45 \text{ deg} & \varphi_2(0) &= -15 \text{ deg} \\ \varphi_3(0) &= 30 \text{ deg} & \varphi_4(0) &= -37.5 \text{ deg} \end{aligned} \quad (35)$$

As before, the pendulum system was implemented twice. The first pendulum system works on the basis of the modified Mechanical Multibody Library, while the second one uses the Multibody Standard Library and serves as a reference. Hence, the deviations to the modified model can be calculated. They have the same order of magnitude as in the example before and can thus be explained by numerical errors.

For the rough illustration of the simulation results, Figure 9 shows the configuration of the pendulum at ten different time instances (the time interval is 1 s). The dashed lines show the tracks of both trolleys. The

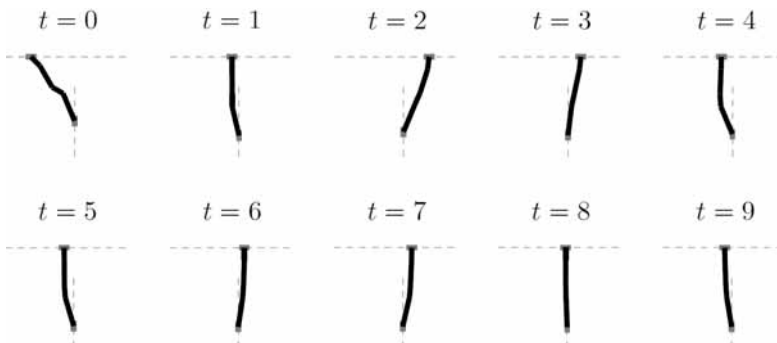


Figure 9. Sequence of configurations of the fourfold pendulum.

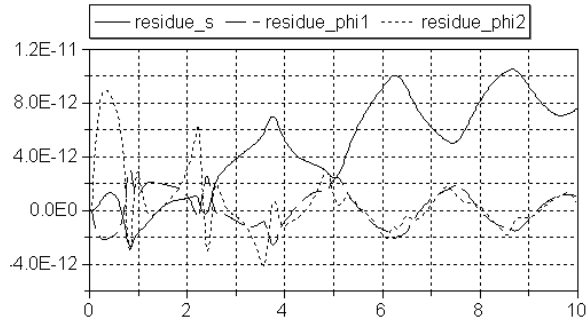


Figure 8. Deviation of the most interesting coordinates between the motor calculus and the Modelica Standard Library implementation.

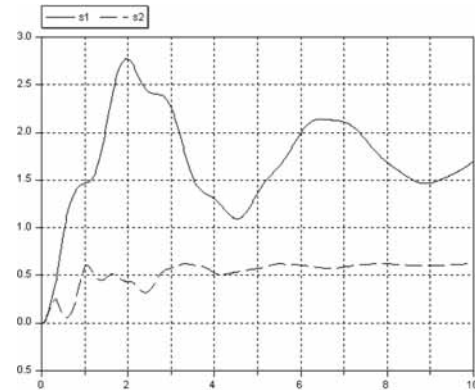


Figure 11. Position of both trolleys for the motor calculus implementation.

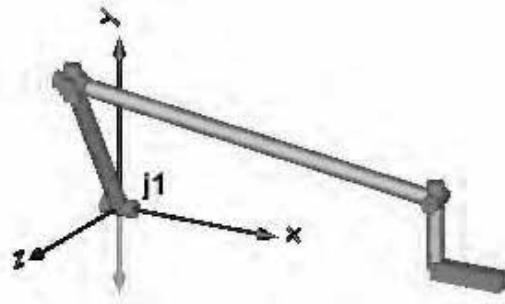


Figure 12. Sketch of the fourbar mechanism.

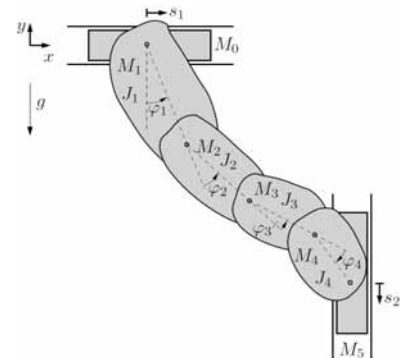


Figure 10. Sketch of the fourfold pendulum.

bold plotted polygon consists of four segments. It represents the idealized shape of the chain. In Figure 11, the position of both trolleys are plotted against the time.

3.2 Fourbar mechanism

The last example is a so-called fourbar mechanism from the Modelica Standard Library, that, again, sets up a closed kinematic loop (see Figure 12). However, in this example, the rigid bodies do not perform planar motions any more and, hence, the whole complexity of the three-dimensional mechanics is necessary.

The fourbar mechanism moves under the influence of the earth's gravitational field. The initial condition of the angular velocity of the first revolute joint (j1) is set to 300 deg/s. In opposite to the foregoing examples, this system is completely undamped. As in the paragraphs before, the example was implemented twice in one model. One system has just been kept in its original form while in the second system, all `BodyCylinder` objects have been replaced by the modified `BodyCylinder` objects. The difference between both implementations is shown in Figure 13. The numerical results of the simulation show an increasing deviation with advancing time. The reason for this fact may be the absence of any damping elements. Indicated by this result, further investigations on numerical accuracy seem to be necessary for the future.

4 Summary and outlook

The paper traces the idea of applying the so-called motor calculus within Modelica modelling language to handle models of spatial multibody systems in an efficient way. This method represents an alternative approach to modelling such systems. This approach is characterized by a clear and concise formulation of the equations of motion.

To get some experiences with possibilities and limits of this approach, a first test implementation was carried out. The Modelica Multibody Standard Library was used to implement appropriate extensions within some selected submodels. This implementation allows a comparison of the standard library implementation and the motor calculus implementation by means of simple simulation tasks. Appropriate results are presented in the paper.

These results seem to encourage the idea of motor calculus usage within Modelica. Nevertheless, there are open challenges to be solved in the future. Oper-

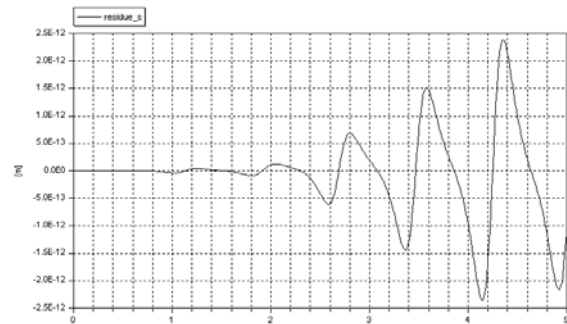


Figure 13. Deviation of the slider position between the motor calculus and the Modelica Standard Library implementation.

and overloading would be a very helpful feature in this context. Furthermore, efficient methods have to be adapted to compute actual position and orientation of a rigid body from its velocity motor. That's why further investigations as well as implementation work will still have to be carried out for a full support of rigid-body motion equation by means of motor calculus in Modelica.

References

- [1] R.S. Ball. *A Treatise on the Theory of Screws*. Cambridge University Press, 1900.
- [2] F. Casella, M. Lovera. *High-accuracy orbital dynamics simulation through Keplerian and equinoctial parameters*. In Proc. 6th Int. Modelica Conference, Bielefeld, Germany, March 3–4, 2008, pages 505–514. The Modelica Association, 2008.
- [3] F.E. Cellier. *Continuous System Modeling*. Springer, 1991.
- [4] W.K. Clifford. *Preliminary sketch of bi-quaternions*. Proc. London Math. Soc., 4:381–395, 1873.
- [5] P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Press, 2003.
- [6] J. Gallardo-Alvarado. *Kinematics of a hybrid manipulator by means of screw theory*. Journal Multibody System Dynamics, 14(3–4):345–366, 2005.
- [7] A. Haumer, C. Kral, J.V. Gragger, H. Kapeller. *Quasistationary modeling and simulation of electrical circuits using complex phasors*. In Proc. 6th Int. Modelica Conference, Bielefeld, Germany, March 3–4, 2008, pages 229–236. The Modelica Association, 2008.
- [8] C. Heinz. *Motorrechnung im X_{1+3+3}* . Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM), 67(11):537–544, 1987.
- [9] C. Knobel, G. Janin, A. Woodruff. *Development and verification of a series car Modelica/Dymola multibody model to investigate vehicle dynamics systems*. In Proc. 5th Int. Modelica Conference, Vienna, Aus-

- tria, September 4–5, 2006, pages 167–173. The Modelica Association, 2006.
- [10] I.I. Kosenko, M.S. Loginova, YA.P. Obratsov, and M.S. Stavrovskaya. *Multibody systems dynamics: Modelica implementation and Bond Graph representation*. In Proc. 5th Int. Modelica Conference, Vienna, Austria, September 4–5, 2006, pages 213–223. The Modelica Association, 2006.
- [11] R. von Mises. *Motorrechnung, ein neues Hilfsmittel der Mechanik*. Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM), 4(2):155–181, 1924.
- [12] R. von Mises. *Anwendungen der Motorrechnung*. Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM), 4(3):193–213, 1924.
- [13] www.modelica.org/documents. 2008-04-22
- [14] www.modelica.org/libraries/Modelica/, 2008-04-22
- [15] M. Otter, H. Elmqvist, and S. E. Mattsson. *The New Modelica MultiBody Library*. In Proc. 3rd Int. Modelica Conference, Linköping, Sweden, November 3–4, 2003, pages 311–330. The Modelica Association, 2003.
- [16] T. Pulecchi, M. Lovera. *Object-oriented modelling of the dynamics of a satellite equipped with single gimbal control moment gyros*. In Proc. 4th Int. Modelica Conference, Hamburg, Germany, March 7–8, 2005, Proc., pages 35–44. The Modelica Association, 2005.
- [17] B. Stroustrup. *The C++ Programming Language – Special Edition*. Addison-Wesley, 2007.
- [18] H. Stumpf, J. Badur. *On the non-abelian motor calculus*. Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM), 70(12):551–555, 1990.
- [19] M.M. Tiller. *Introduction to Physical Modeling with Modelica*. Springer, 2001.
- [20] L. Viganò, G. Magnani. *Acausal modelling of helicopter dynamics for automatic flight control applications*. In Proc. 5th Int. Modelica Conference, Vienna, Austria, September 4–5, 2006, pages 377–384. The Modelica Association, 2006.
- [21] T. Zaiczek. *Modellbildung für mechanische Systeme mit einer endlichen Anzahl von Freiheitsgraden und Steuerungsentwurf mithilfe von $m \geq 1$ Aktuatoren*. Technical report, 2006.
- [22] T. Zaiczek. *Modellierung, Regelung und Simulation mechanischer Starrkörpersysteme im dreidimensionalen Raum*. Diploma thesis, TU Dresden, Germany, 2007.

Corresponding author: Tobias Zaiczek
 Fraunhofer Institute for Integrated Circuits,
 Design Automation Division, Dresden, Germany
Tobias.Zaiczek@eas.iis.fraunhofer.de

Accepted: EOOLT 2007, June 2007

Received: August 10, 2007

Accepted: August 20, 2007