# Performance Analysis of VON MISES' Motor Calculus within Modelica

Tobias Zaiczek Olaf Enge-Rosenblatt
Fraunhofer Institute for Integrated Circuits
Design Automation Division
Dresden, Germany
{Tobias.Zaiczek,Olaf.Enge}@eas.iis.fraunhofer.de

## Abstract

This paper presents an alternative concept of modelling multibody systems within Modelica, the so-called motor calculus. This approach was introduced by R. VON MISES in 1924 and can be used to describe the dynamical behaviour of spatial multibody systems in a very efficient way. While the equations clearly take a very simple form in terms of motor algebra, the numerical efficiency is still an open question.

In the paper, first some fundamentals of motor calculus are summarized. An experimental implementation of motor algebra is used to measure and analyse the numerical efficiency and performance regarding the simulation time of VON MISES' approach. Therefore, some components of the Modelica Multibody Standard Library were modified in order to compare both implementations. Finally, some examples are given to prove the applicability and correctness of the concept but also to serve as a basis for a discussion of the numerical performance. The chosen approach utilizes all object-oriented features provided by the modelling language. Besides, it gives reason for the present endeavours to introduce the possibility of operator overloading within Modelica.

*Keywords: motor calculus, screw theory, rigid multibody system, Modelica, performance*

## 1 Introduction

The motion of mechanical systems in three-dimensional space has been examined for hundred of years. In 1924 R. VON MISES suggested an approach, the so-called motor calculus, to describe rigid body motion in 3D mechanics in a very clear and efficient way [6, 7]. Inspired by previous contributions (e.g. [2, 3, 12]), he introduced the motor as a six-tuple of scalar quantities and developed a special algebra for these mathematical objects, called the motor calculus. Though his approach is not well known throughout all branches of mechanical engineering, in the field of robotics VON MISES' ideas were rediscovered during the last decades [1, 5, 10, 11, 13], since they seem to be well suited to investigate the behaviour of spatial multibody systems. However, in the context of the modelling language Modelica (see e.g. [4, 14]), the motor calculus has not been taken into account up to now.

Meanwhile, many researchers apply the Modelica Multibody Standard Library ([9]) to model different kinds of – partially very complex – multibody systems (see proceedings of the Modelica conferences [8]). Hence, this library has proven to be a well suited resource to modelling such systems. Nevertheless, applying the motor calculus, the equations of motion for a rigid body become more concise and clearer, e.g.

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

($\mathfrak{p}$ – momentum motor, $\mathfrak{f}$ – 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.

In our first publication [15], we were already able to show the possible simplifications of the resulting equations within some components of the Modelica Multibody Standard Library using the motor calculus. Furthermore, we compared both approaches e. g. with respect to numerical correctness. So, the motivation to follow further up the motor calculus in the Modelica context is now to investigate the performance of the simulation of mechanical systems with regard to simulation time.

An extended test realization within the Modelica Multibody Standard Library has been carried out by changing 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 effectiveness.

In the following section, some fundamentals of motor calculus are shortly sketched. Some of the most important mathematical operations are defined. The test implementation is presented in section 3. The performance of the motor calculus approach will be evaluated and compared to the performance of the Modelica Multibody Standard Library using some examples in section 4.

# 2 Fundamentals of motor calculus

A *motor*

$$\mathfrak{h} = \begin{pmatrix} \boldsymbol{g} \\ \boldsymbol{h}_o \end{pmatrix}$$

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

$$\boldsymbol{h}(\boldsymbol{r}) = \boldsymbol{h}_o + \boldsymbol{g} \times \boldsymbol{r} \tag{1}$$

in the three-dimensional Euclidean space. In this definition, $\boldsymbol{r}$ is the position vector of any point in space, while the vectors $\boldsymbol{h}$ and $\boldsymbol{g}$ are called the *moment* and the *resultant vector* of the motor, respectively. Accordingly, $\boldsymbol{h}_o$ 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 $\boldsymbol{h}$ is parallel to the resultant vector $\boldsymbol{g}$. All these points exhibit the same moment $\boldsymbol{h}_n$ and lie on a straight line $\mathcal{N}$ given by

$$\boldsymbol{r}_n(\lambda) = \frac{\boldsymbol{g} \times \boldsymbol{h}_o}{|\boldsymbol{g}|^2} + \lambda \boldsymbol{g} \,, \qquad \lambda \in \mathbb{R}.$$

**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 constructions. Hence, all motors can be seen as abstract objects that do not depend on the choice of a reference frame. Details can be found in [7, 15].

## 2.1 Motor calculus

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

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

$$\mathfrak{h} = \begin{pmatrix} \boldsymbol{g} \\ \boldsymbol{h}_o \end{pmatrix}, \quad \mathfrak{h}_1 = \begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{h}_{o1} \end{pmatrix}, \quad \mathfrak{h}_2 = \begin{pmatrix} \boldsymbol{g}_2 \\ \boldsymbol{h}_{o2} \end{pmatrix}.$$

Then, according to VON MISES, the following mathematical operations are defined:

$$\mathfrak{h}_1 + \mathfrak{h}_2 = \begin{pmatrix} \boldsymbol{g}_1 + \boldsymbol{g}_2 \\ \boldsymbol{h}_{o1} + \boldsymbol{h}_{o2} \end{pmatrix} \qquad \text{(addition)}$$

$$\alpha\mathfrak{h} = \begin{pmatrix} \alpha\boldsymbol{g} \\ \alpha\boldsymbol{h}_o \end{pmatrix} \qquad \text{(multiplication with a scalar } \alpha \in \mathbb{R})$$

$$(\mathfrak{h}_1, \mathfrak{h}_2) = (\boldsymbol{g}_1, \boldsymbol{h}_{o2}) + (\boldsymbol{g}_2, \boldsymbol{h}_{o1}) \qquad \text{(inner product)}$$

$$\mathfrak{h}_1 \times \mathfrak{h}_2 = \begin{pmatrix} \boldsymbol{g}_1 \times \boldsymbol{g}_2 \\ \boldsymbol{g}_1 \times \boldsymbol{h}_{o2} + \boldsymbol{h}_{o1} \times \boldsymbol{g}_2 \end{pmatrix} \qquad \text{(outer product)}$$

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 $(6 \times 6)$ matrix.

The mapping can be described in the following manner:

$$\begin{aligned} \mathfrak{T} \circ \mathfrak{h}_1 &= \begin{pmatrix} \boldsymbol{T}_{11} & \boldsymbol{T}_{12} \\ \boldsymbol{T}_{21} & \boldsymbol{T}_{22} \end{pmatrix} \circ \begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{h}_{o1} \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{T}_{11}\boldsymbol{h}_{o1} + \boldsymbol{T}_{12}\boldsymbol{g}_1 \\ \boldsymbol{T}_{21}\boldsymbol{h}_{o1} + \boldsymbol{T}_{22}\boldsymbol{g}_1 \end{pmatrix}. \end{aligned} \tag{2}$$

Now, all calculation rules for the motor calculus can be derived readily. For details we refer to [7, 15]. In [15], it is also shown that, due to the definition of addition and scalar multiplication, motors span a vector space over the field of real numbers. Additionally, by the introduction of the outer product, motors form a *Lie-Algebra*[1].

### 2.1.1 Differentiation with respect to real-valued parameters

Consider a motor $\mathfrak{h}$ that depends on a real parameter $t$ (e. g. the time) with differentiable components $\boldsymbol{g}$ and $\boldsymbol{h}_o$ with respect to $t$. Then, the first derivative of this motor with respect to $t$ can be computed componentwise:

$$\frac{d\mathfrak{h}}{dt} = \begin{pmatrix} \frac{d\boldsymbol{g}}{dt} \\ \frac{d\boldsymbol{h}_o}{dt} \end{pmatrix}.$$

### 2.1.2 Differentiation in moving frames

If frame $\mathcal{F}_1$ moves relatively to a reference frame $\mathcal{F}_0$, the observed temporal change of a motor is then in general different in the two frames. The relative motion of the origin of frame $\mathcal{F}_1$ measured in frame $\mathcal{F}_0$

[1] Named after the mathematician SOPHUS LIE (*1842, †1899).

shall be given by the velocity vector $\underline{\boldsymbol{v}}_o$, while the angular velocity vector of frame $\mathcal{F}_1$ with respect to frame $\mathcal{F}_0$ is denoted by $\underline{\boldsymbol{\omega}}$. Then, the equation

$$\dot{\mathfrak{h}} = \overset{\circ}{\mathfrak{h}} + \begin{pmatrix} \underline{\boldsymbol{\omega}} \\ \underline{\boldsymbol{v}}_o \end{pmatrix} \times \mathfrak{h} \tag{3}$$

holds for the derivation with respect to time observed in frame $\mathcal{F}_0$. In Equ. (3), $\overset{\circ}{\mathfrak{h}}$ denotes the derivation w. r. t. time of the motor $\mathfrak{h}$ observed in frame $\mathcal{F}_1$.

## 2.2 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.

All forces and torques acting on a rigid body can be combined to one single force vector $\boldsymbol{f}$ and one torque vector $\boldsymbol{d}_o$. Similarly, the movement of a rigid body can be fully described by the movement of a special reference point $O$ on the body (i. e. by its velocity vector $\underline{\boldsymbol{v}}_o$) and the angular velocity vector $\underline{\boldsymbol{\omega}}$, the body is turning with (see Fig. 1).

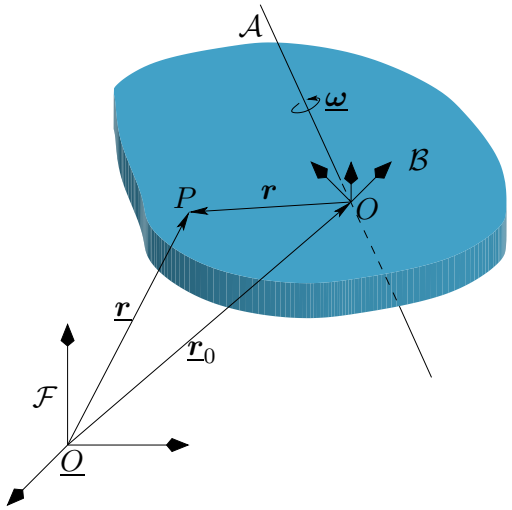


Figure 1: Definition of vectors at the rigid body

The following paragraphs aim to show that, by introducing physically motivated motors, the motor calculus is well suited to describe rigid body movements.

### 2.2.1 Definition of physically motivated motors

Here, 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 $\boldsymbol{f}$ and torque $\boldsymbol{d}_o$ (referred to the reference point $O$) acting on the rigid body, i. e.

$$\mathfrak{f} = \begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{d}_o \end{pmatrix}.$$

Hence, the torque referred to any other point with the position vector $\boldsymbol{r}$ is calculated by

$$\boldsymbol{d}(\boldsymbol{r}) = \boldsymbol{d}_o + \boldsymbol{f} \times \boldsymbol{r}\ .$$

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 $\underline{\boldsymbol{v}}_o$ of the chosen reference point $O$ and the angular velocity vector $\underline{\boldsymbol{\omega}}$ representing the rotation of the body w. r. t. an inertial frame:

$$\mathfrak{v} = \begin{pmatrix} \underline{\boldsymbol{\omega}} \\ \underline{\boldsymbol{v}}_o \end{pmatrix}.$$

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

$$\underline{\boldsymbol{v}}(\boldsymbol{r}) = \underline{\boldsymbol{v}}_o + \underline{\boldsymbol{\omega}} \times \boldsymbol{r}\ .$$

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

$$\mathfrak{p} = \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{l}_o \end{pmatrix}.$$

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

$$\boldsymbol{l}(\boldsymbol{r}) = \boldsymbol{l}_o + \boldsymbol{p} \times \boldsymbol{r}$$

holds. The proof of this statement can be found in [15].

### 2.2.2 Some fundamental laws of mechanics in terms of motor calculus

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

$$\mathfrak{p} = \underbrace{\begin{pmatrix} m\boldsymbol{I} & -m\boldsymbol{R}_{\boldsymbol{s}} \\ m\boldsymbol{R}_{\boldsymbol{s}} & \boldsymbol{\Theta}_o \end{pmatrix}}_{\mathfrak{M}} \circ \mathfrak{v}\ . \tag{4}$$

The new symbol $\boldsymbol{R_s}$ describes the cross product dyad of the vector $\boldsymbol{r}_s$ pointing to the centre of mass.

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

$$\boxed{\dot{\mathfrak{p}} = \mathfrak{f}}\,.$$

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

A much more applicable form for concrete calculations can be derived using (3) to express the time derivation w. r. t. the body frame

$$\boxed{\mathring{\mathfrak{p}} + \mathfrak{v} \times \mathfrak{p} = \mathfrak{f}}\,, \tag{5}$$

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

Replacement of the momentum motor with the help of Equ. (4) yields the following relationship

$$\boxed{\mathfrak{M} \circ \mathring{\mathfrak{v}} + \mathfrak{v} \times (\mathfrak{M} \circ \mathfrak{v}) = \mathfrak{f}}$$

if all components are given in the body frame.

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

$$\boxed{T = \frac{1}{2}(\mathfrak{v}, \mathfrak{p}) \qquad \text{with} \qquad \mathfrak{p} = \mathfrak{M} \circ \mathfrak{v}}\,.$$

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 $\mathfrak{M}$ 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

$$\boxed{P = (\mathfrak{f}, \mathfrak{v})}\,.$$

#### 2.2.3 Applications to multibody systems

The use of the motor calculus introduced above can also be very beneficial when describing multibody systems. These systems are often modelled as an interconnection structure of rigid bodies and ideal joints.

Exemplarily, two types of ideal joints, the revolute and the prismatic joint, will be analysed in this paper. Therefore, the necessary equations will be derived in this paragraph.

Both joints set up one constraint equation on the relative motion of the rigid bodies interconnected. This constraint can easily be expressed in terms of motor algebra. By defining the unit vector $\boldsymbol{e}$ as the joint axis, one can write the velocity motor of the relative motion for the prismatic and the revolute joint as

$$\mathfrak{v}_r = \dot{x}\mathfrak{e}_P = \begin{pmatrix} \mathbf{0} \\ \dot{x}\boldsymbol{e} \end{pmatrix} \quad \text{and} \quad \mathfrak{v}_r = \dot{x}\mathfrak{e}_R = \begin{pmatrix} \dot{x}\boldsymbol{e} \\ \mathbf{0} \end{pmatrix}. \tag{6}$$

Here $\dot{x}$ denotes the translational velocity along or the rotational velocity around the joint axis $\boldsymbol{e}$. The cut forces and torques within the joint are merged in the force motor $\mathfrak{f}$. Since friction is neglected, the dissipated power of the joint vanishes and hence the applied power reads

$$P = (\mathfrak{f}, \mathfrak{v}_r)\,.$$

## 3 Object-oriented implementation

The test implementation presented here is based on the Modelica Multibody Standard Library. Due to some still existing limitations of the Modelica language in terms of operator overloading, compromises had to be made during implementation of the motor calculus.

### 3.1 Motor library

The first step of the implementation towards a description of rigid body motion by means of motor calculus is the realization of a general motor class. From the view of data structure, motors are nothing more than a combination of six scalars.

A clear structured class `motor` with two vectors, the resultant vector and the moment vector, would have been desirable. Due to the missing possiblity of operator overloading in our simulation tool (Dymola 7.1), an alternative implementation has been chosen. All six scalars are stored within one vector which is called `Motor`:

```
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. One drawback is that, 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";
type DerMomMotor  = Motor "Time Derivative
                      of Momentum motor";
```

All the other calculation rules introduced in section 2.1 had to be implemented using Modelica functions.

The first function has been written to perform the inner product between two motors. It is denoted by `dot`:

```
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 using the function `'x'`:

```
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';
```

The preceding 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 $(6 \times 6)$ matrix:

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

To apply a motor dyad to a motor, another function has been created. Referring to Equ. (2), the function has been defined by:

```
function times "Application of a Motor Dyad
                on a 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 enable to transform the components of a motor from one frame to another and vice versa.

```
function coordChange1 "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 2, resolved in
     frame 1";
  input F.Orientation R "Orientation object
    of frame 2 resolved in frame 1";
  input Motor m1 "Motor res. in frame 2";
  output Motor m2 "Motor res. in frame 1";
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 coordChange1;

function coordChange2 "Transforms motor
  from frame 1 to frame 2"
  import F = Modelica.Mechanics.MultiBody.
    Frames;
  input Modelica.SIunits.Position[3] r_0
    "Vector pointing from origin of frame a
     to origin of frame 2, resolved in
     frame 1";
  input F.Orientation R "Orientation object
    of frame 2 resolved in frame 1";
  input Motor m1 "Motor res. in frame 1";
  output Motor m2 "Motor res. in frame 2";
algorithm
  m2 := vector([R.T*m1[1:3];
                R.T*mom(m1,r_0)]);
end coordChange2;
```

## 3.2 Multibody implementation

The existing implementations of several parts of the Modelica Multibody Standard Library were adapted to the motor algebra. First of all, the connectors `frame_a` and `frame_b` were changed by substituting the vectors `force` and `torque` by the force motor `force` in the connector class `frame`. Hence, all other classes of the Multibody library used in this paper had to be adjusted as well. In the following, some important changes to the most relevant classes will be explained in detail.

### 3.2.1 Changes to the `body` class

The first changes were the replacements of important motion variables by some physically motivated motors. While the angular velocity vector as well as the velocity vector of `frame_a` were removed, the velocity motor and the momentum motor were introduced. Also, all acceleration vectors and all inertia dyades have been replaced by the time derivative of the momentum motor `dmom` and the motor inertia dyad, respectively.

```
// Motors
// ------
VelocityMotor velB  (start=[\dots])
  "Velocity motor wrt. frame a";
MomentumMotor mom   (start=[\dots])
  "Momentum motor wrt. frame a";
DerMomMotor dmom    (start=[\dots])
  "Time Derivative of Momentum motor wrt.
   frame a";
ForceMotor f_g
  "Force Motor due to gravitation";
// Motor Dyads
// -----------
final parameter MotorDyad I_mot =
  [diagonal({m, m, m}),-skew(m*r_CM);
   skew(m*r_CM), [I_11, I_21, I_31;
                  I_21, I_22, I_32;
                  I_31, I_32, I_33]
      + m*(diagonal(r_CM*r_CM*ones(3))
           -[r_CM]*transpose([r_CM]))]
  "Motorial Inertia Tensor";
```

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

```
// Motors
// ------
velB  = vector([frame_a.R.w;
           frame_a.R.T*der(frame_a.r_0)]);
mom   = times(I_mot, velB);
dmom  = der(mom);
f_g   = vector([ m*frame_a.R.T*g_0;
          cross(r_CM, m*frame_a.R.T*g_0)]);
```

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 Equ. (5):

```
frame_a.f = der(mom) + 'x'(velB,mom) - f_g;
```

Because of the object-oriented structure of the Modelica Standard Library, the changes had to be implemented only once. All subclasses of the `Body` class, like `BodyShape`, `BodyBox`, or `BodyCylinder` inherit the changes automatically.

### 3.2.2 Changes to the `revolute` class

Also, in the class `revolute` some changes had to be carried out. Firstly,

```
frame_a.f = -Frames.resolve1(R_rel,
                             frame_b.f);
frame_a.t = -Frames.resolve1(R_rel,
                             frame_b.t);
```

had to be replaced by

```
frame_a.f = -coordChange1(zeros(3),R_rel,
                                  frame_b.f);
```

and

```
frame_b.f = -Frames.resolve1(R_rel,
                                  frame_a.f);
frame_b.t = -Frames.resolve1(R_rel,
                                  frame_a.t);
```

was substituted by

```
frame_b.f = -coordChange1(zeros(3),R_rel,
                                  frame_a.f);
```

Last of all, the constraint equation was reformulated according to (6) as

```
  tau=-dot(frame_b.f,vector([e;zeros(3)]));
```

### 3.2.3 Changes to the `prismatic` class

In the class `prismatic` the following lines

```
zeros(3) = frame_a.f + frame_b.f;
zeros(3) = frame_a.t + frame_b.t
     + cross(e*(s_offset + s), frame_b.f);
// d'Alemberts principle
f = -e*frame_b.f;
```

were replaced by the two lines

```
zeros(6) = frame_a.f
        + coordChange1(e*(s_offset + s),
        Frames.nullRotation(),frame_b.f);
// d'Alemberts principle
f = -dot(frame_b.f,vector([zeros(3);e]));
```

# 4 Examples and performance analysis

On the basis of the following examples, different performance tests were carried out in order to evaluate the numerical effectiveness of the two different modelling approaches. Two of the examples can also be found in [15] where the authors already showed the applicability and correctness of some of the implementations.

The following first three subsections describe the chosen examples and show some simulation results. The last subsection introduces a performance criterion and evaluates the performance of the simulations.

## 4.1 Movable double pendulum

As a first example, the movable double pendulum (Fig. 2) was chosen to compare the simulation time of the implemented body classes based on motor calculus to the implementation of the Modelica Standard

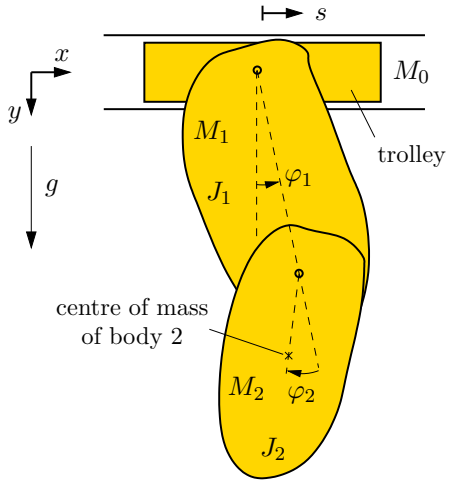


Figure 2: Sketch of double pendulum

Library. 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 Fig. 2). 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 distance between both axis of rotations is denoted by $l_1$.

The pendulum moves from an initial deflection of $\varphi_1(0) = 90\,\text{deg}$ and $\varphi_2(0) = 0\,\text{deg}$ 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. In Fig. 3, the trajectory for the position $s$ of the trolley for both simulations is displayed. Both curves are nearly congruent. Hence, Fig. 3 shows only one curved line.

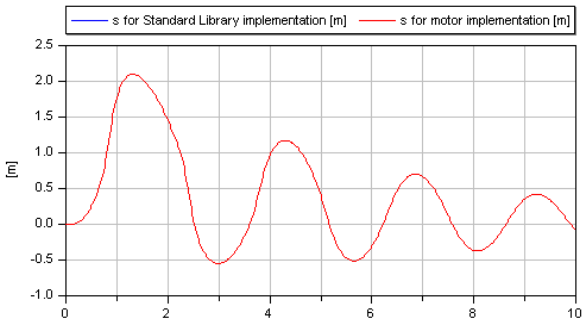


Figure 3: Trajectory of the trolley position $s$

The time histories of the revolute joint angles $\varphi_1$ and $\varphi_2$ are depicted in Fig. 4. The differences between both simulation results for an integration tolerance of $10^{-4}$ are shown in Fig. 5. Apparently, the deviation of the position stays smaller than $6 \cdot 10^{-12}$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, since they depend on the integration tolerance.

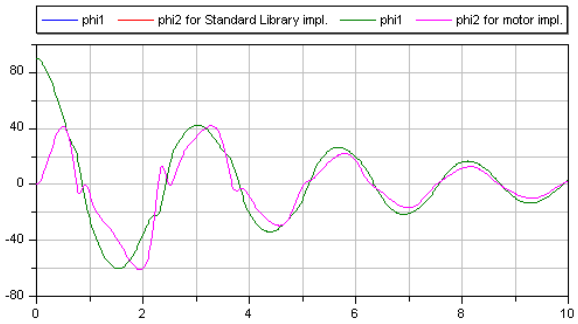


Figure 4: Trajectory of the pendulum angles $\varphi_1$ and $\varphi_2$

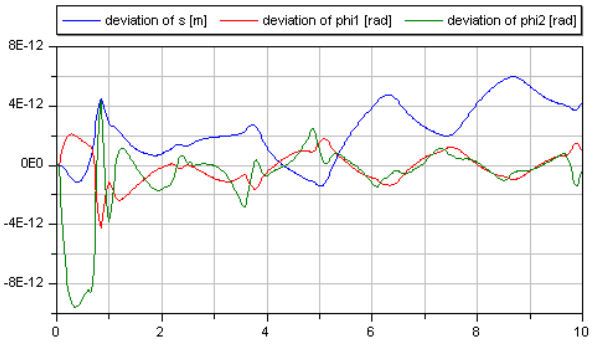


Figure 5: Deviations between both simulations for trolley position $s$ and both angles $\varphi_1$ and $\varphi_2$

## 4.2 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 Fig. 6). Hence, this example contains a closed kinematic loop. Similar to the foregoing example, the pendulum moves due to the gravity field of the earth. The motion starts with an initial deflection (see Fig. 7) and is damped by a viscous friction in ev-

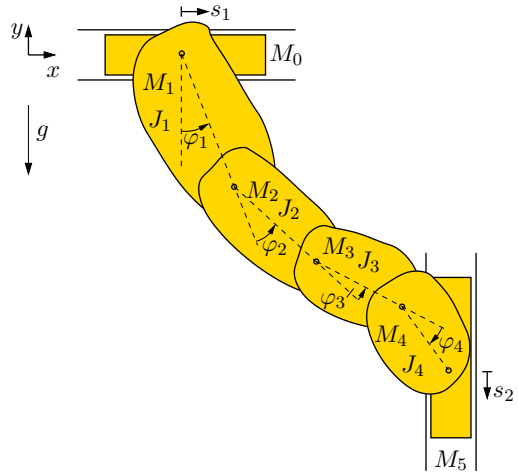


Figure 6: Sketch of fourfold pendulum

ery joint except the last one connecting the bodies with masses $M_4$ and $M_5$. The initial values for the pendulum angles are

$$\varphi_1(0) = 45\,\text{deg}, \qquad \varphi_2(0) = -15\,\text{deg},$$
$$\varphi_3(0) = 30\,\text{deg}, \qquad \varphi_4(0) = -37.5\,\text{deg}.$$

Fig. 7 shows the initial configuration of the pendulum system. Here, the length proportions between the four bodies of the pendulum are illustrated.

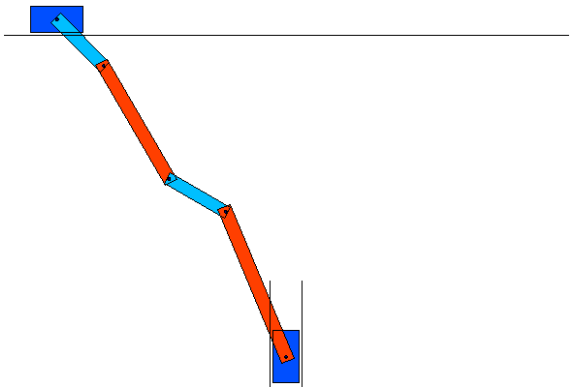

Figure 7: Start configuration of fourfold pendulum

Like before, the pendulum system was implemented twice using two different simulation models. The first implementation is based on the Multibody Standard Library and serves as a reference. The second model uses the modified Multibody Library on the basis of the motor algebra.

In order to compare both approaches concerning their simulation results, one instance of the first model and one instance of the second model were calculated simultaneously. This way, the deviations of both simulations can be calculated. For reasons of compactness only the maximum deviation for all prismatic joints and for all revolute joints are plotted in Fig. 8. They have the same order of magnitude as in the example before and can thus be explained by numerical errors.

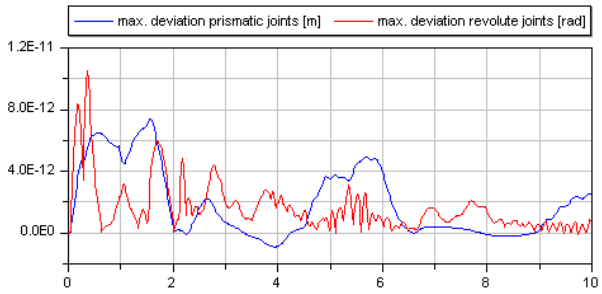


Figure 8: Maximal deviations between both simulations for all prismatic joints and all revolute joints

### 4.3 Rotating wheel on a movable axis

The last example is a rotating wheel that is fixed on a movable axis (see Fig. 9). There are three revolute joints within this mechanism. The first one allows a rotation of the rack (the long cylinder posing upright in Fig. 9) around the $z$-direction. A second revolute joint is the bearing of the wheel that enables the wheel to turn around its axis. Between them, there is a revolute joint enabling a rotation of the axis orthogonal to the rotation of the spinning wheel. Hence, in this example, the rigid bodies do not only perform planar motions.

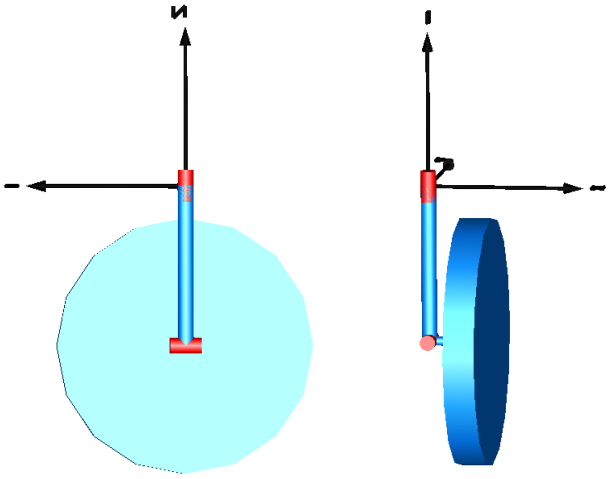

Figure 9: Sketches of a rotating wheel on movable axis

Again, the bodies move under the influence of the earth's gravitational field that acts in the negative $z$-direction.

At the beginning, the wheel turns with a speed of $\omega_3 = 50\,\mathrm{rad/s}$ around its axis while the rack and the wheel's axes stand perpendicular to each other. In opposit to the foregoing examples, this system is completely undamped. Fig. 10 depicts the trajectory of the intersection point between the wheel axis and a unit sphere. Obviously, the resulting motion of the mechanism is a superposition of a precession and a free nutation. In order to illustrate this characteristic motion in more detail, Fig. 11 shows the same trajectory for an initial speed of only $10\,\mathrm{rad/s}$.

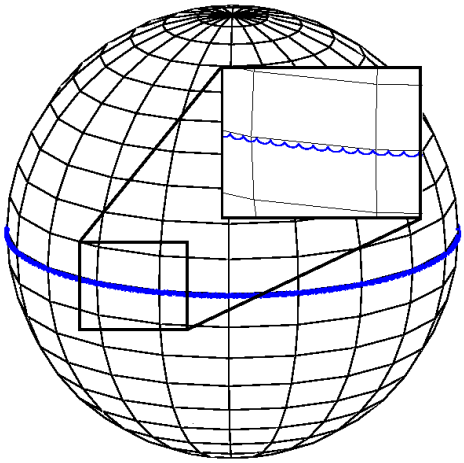

Figure 10: Trajectory of the intersection point between the rotation axis and a unit sphere for $\omega_3 = 50\,\mathrm{rad/s}$

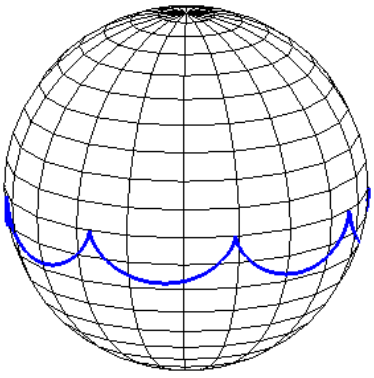

Figure 11: Trajectory of the intersection point between the rotation axis and a unit sphere for $\omega_3 = 10\,\mathrm{rad/s}$

As in the paragraphs before, the example was implemented twice to compare Modelica Standard implementation with modified motor implementation. The deviations for all revolute joint angles between both implementations do not exceed $6 \cdot 10^{-12}$ for a simulation time of 10s. However, since the system is undamped, the deviations of both simulations increase with continuing time.

## 4.4 Performance analysis

All simulation tests were performed using the simulation tool Dymola in the version 7.1. The analysis of the performance requires the definition of a performance indicator. Even though Dymola provides a lot of information on the translation as well as the simulation process, we decided to evaluate the performance by the simulation time, since this might be the most interesting indicator for many users. For all simulations, we used a PC running the operating system Windows XP Professional. All simulations has been carried out ten times for each model while no other application was running on the system. A comparison of the average values of the simulation time for these implementations can be seen in Fig. 12.

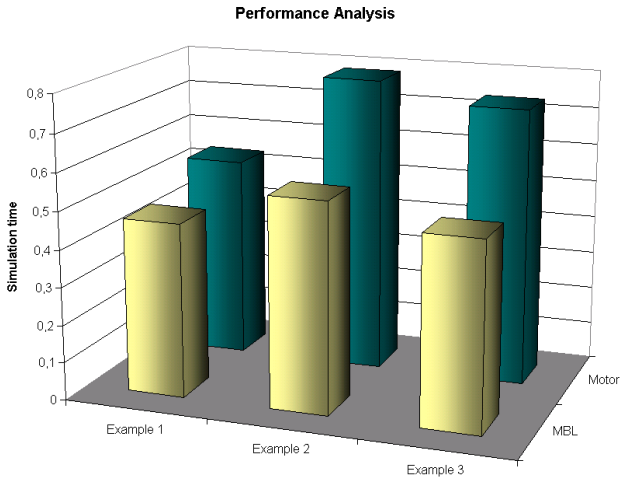


Figure 12: Simulation time of all examples for both implementations

Hence, the performance analysis on the chosen simulation system revealed that the modified multibody library on the basis of the motor algebra shows a performance which is inferior compared to the one of the Modelica Standard Library. According to the translation information of Dymola the reason might be that Dymola was not able to reduce and simplify the system equations of the motor implementation as much as the equations of the Modelica Standard Library. That seems reasonable due to the broad use of functions and vectorised quantities within the motor implemen-

tation. Indicated by this insight, further investigations with different simulation tools seem to be necessary for the future to get results which are clearer and better comparable.

## 5 Summary and outlook

The paper shows an alternative approach to modelling spatial multibody systems in Modelica. This approach is characterized by a clear and concise formulation of the equations of motion.

To get some experiences in terms of numerical efficiency and limits of this approach, an extended test implementation was carried out. Appropriate modifications of the Modelica Multibody Standard Library enabled us to compare the Standard Library implementation and the motor calculus implementation with regard to simulation time.

The results presented here were determined using the Modelica simulator Dymola. These results seem to encourage the idea of testing the motor calculus within other Modelica simulator tools, too.

## References


[1] J. Angeles. *Fundamentals of Robotic Mechanical Systems.*. Second Edition. NewYork, Springer-Verlag, 2003.

[2] R.S. Ball. *A Treatise on the Theory of Skrews*. Cambridge University Press, 1900.

[3] W.K. Clifford. Preliminary sketch of biquaternions. *Proc. London Math. Soc.*, 4:381–395, 1873.

[4] P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley-IEEE Press, 2003.

[5] C. Heinz. Motorrechnung im $X_{1+3+3}$. *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 67(11):537–544, 1987.

[6] R. von Mises. Motorrechnung, ein neues Hilfsmittel der Mechanik. *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 4(2):155–181, 1924.

[7] R. von Mises. Anwendungen der Motorrechnung. *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 4(3):193–213, 1924.

[8] `http://www.modelica.org/events`. seen on August 10th, 2009.

[9] M. Otter, H. Elmqvist, and S. E. Mattsson. The New Modelica MultiBody Library. In *3rd International Modelica Conference, Linköping, Sweden, November 3–4, 2003, Proc.*, pages 311–330. The Modelica Association, 2003.

[10] B. Roth. Screws, motors, and wrenches that cannot be bought in a hardware store. In M. Brady and R. Paul (eds.): *The First Internal Symposium on Robotic Research.*, MIT Press, Cambridge (MA), pp. 679–693,.

[11] K. Sugimoto. Kinematic and Dynamic Analysis of Parallel Manipulators by Means of Motor Algebra. Journal of mechanisms, transmissions, and automation in design, vol. 109(1), pp 3–7, 1987.

[12] E. Study. *Geometrie von Dynamen. Die Zusammensetzung von Kräften und verwandte Gegenstände der Geometrie.* Teubner, Leipzig, 1903,

[13] H. Stumpf and J. Badur. On the non-abelian motor calculus. *Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM)*, 70(12):551–555, 1990.

[14] M.M. Tiller. *Introduction to Physical Modeling with Modelica*. Springer, 2001.

[15] T. Zaiczek, O. Enge-Rosenblatt. Towards an Object-oriented Implementation of VON MISES' Motor Calculus Using Modelica. In *2nd International Workshop on Equation-Based Object-Oriented Languages and Tools, Paphos, Cyprus, July 3, 2008, Proc.,* pages 131–140.