Weierstraß-Institut
für Angewandte Analysis und Stochastik

im Forschungsverbund Berlin e.V.

Preprint ISSN 0946 – 8633

GEOMS: A software package for the numerical integration of general model equations of multibody systems

Andreas Steinbrecher

submitted: September 25, 2007

Weierstrass Institute for Applied
Analysis and Stochastics
Mohrenstrasse 39
10117 Berlin
Germany
E-mail: steinbrecher@wias-berlin.de

No. 1259
Berlin 2007

2000 *Mathematics Subject Classification.* 70E55, 65L80.

Key words and phrases. differential-algebraic equations, equations of motion, multibody system, numerical integration, simulation.

This work has been supported by DFG research grant no. Me790/13 during the stay of the author at TU Berlin, Fachgebiet Numerische Mathematik. The software package GEOMS is part of the PhD thesis [38] of the author developed at the TU Berlin.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Mohrenstraße 39
10117 Berlin
Germany

Fax: + 49 30 2044975
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

Abstract

In this paper we present the new numerical algorithm GEOMS for the numerical integration of the most general form of the equations of motion of multibody systems, including nonholonomic constraints and possible redundancies in the constraints, as they may appear in industrial applications. Besides the numerical integration it offers some additional features like stabilization of the model equations, use of different decomposition strategies, or checking and correction of the initial values with respect to their consistency. Furthermore, GEOMS preserves hidden constraints and (possibly) existing solution invariants if they are provided as equations.
We will also demonstrate the performance and the applicability of GEOMS for two mechanical examples of different degrees of complexity.

1 Introduction

The *multibody system* (MBS) approach is frequently used in industrial simulation packages in robotics, vehicle system dynamics, and biomechanics. A multibody system model consists of a finite number of rigid or elastic bodies and their interconnections like, e.g., joints, springs, dampers, and actuators. The *equations of motion* may be generated in a systematic way by *multibody formalisms* that are based on the principles of classical mechanics [35].
The efficient and robust numerical integration of these equations is a challenging problem in the development of simulation packages, since dynamical simulation is frequently used and one of the most time consuming analysis methods for MBS models. The equations of motion with nonredundant constraints form a nonlinear system of differential-algebraic equations (DAEs) of differentiation index (d-index) 3, see [7, 10, 13, 18]. It is well known that the numerical treatment of DAEs of high index or higher index, i.e., d-index 2 or larger than 2, respectively, is nontrivial in general. Effects arising in the numerical treatment are, for example, drift, instabilities, convergence problems, or inconsistencies. These difficulties in the numerical solution of such high index problems are discussed in [4, 12, 14, 15, 17, 18, 24, 29, 30]. However, the equations of motion are DAEs with a very special structure that should be exploited in the numerical solution [7, 18].
In this report we will present the new software package GEOMS for the numerical integration of general equations of motion of multibody systems in descriptor form. In contrast to standard textbook presentations like [18], we do not restrict ourselves to classical constrained mechanical systems but consider the more complex model equations that are actually used in state-of-the-art MBS simulation packages [7, 34].

The software package GEOMS is suited for general equations of motion involving dynamical force elements, contact conditions, and (possibly) redundant holonomic as well as nonholonomic constraints. Furthermore, the package takes into account possibly existing information concerning solution invariants, e.g., energy conservation. The code is based on residual evaluations, i.e., the system need not be given completely in explicit form. It is sufficient that the right-hand side of the equations of motion and the mass matrix are specified.
Although, the package GEOMS is able to treat also redundant constraints, in this paper we will restrict our considerations to regular equations of motion, i.e., the constraints are assumed to be nonredundant. For more details on equations of motion with redundant constraints we refer to [26, 38].
As base of the integration method GEOMS we will propose a remodeling of the equations of motions. The aim of this remodeling is to determine an equivalent formulation, the so called *projected-strangeness free form*, which has d-index 1 but has the same set of solutions as the original equations of motion. Because of the reduced d-index, the numerical treatment of the projected-strangeness free form by use of implicit ODE methods is not affected by instabilities arising from the higher index. Furthermore, all (hidden) constraints are preserved such that no drift-off effects arise in the numerical treatment. The proposed remodeling can be seen as regularization of the equations of motion. For more details on the regularization of equations of motion we refer to [38]. The integration method implemented in GEOMS combines an implicit Runge-Kutta-Method of order 5 with this regularization technique.
The report is organized as follows. In Section 2 we introduce the equations of motion which we want to treat numerically and we discuss the remodeling to the projected-strangeness-free form which will be used for the discretization in GEOMS. In Section 3 we introduce the code GEOMS and we discuss its features and its applicability in detail. In Section 4 we demonstrate the properties of the software package GEOMS by two numerical examples. For the usage and implementation of GEOMS the manual is presented in Appendix A.

2 The Equations of Motion and their Remodeling

Here and in the following we will use the following notation.

Notation 2.1 Let f be a differentiable function $f : \mathbb{X} \to \mathbb{R}^m$, $\mathbb{X} \subset \mathbb{R}^n$, and let x be a differentiable function $x : \mathbb{I} \to \mathbb{X}$, where $\mathbb{I}$ is an open interval in $\mathbb{R}$. The ith (total) derivative of $x(t)$ with respect to t is denoted by $x^{(i)}(t) = \mathrm{d}^i x(t)/\mathrm{d}t^i$ for $i \in \mathbb{N}_0$. Note the convention $x^{(0)}(t) = x(t)$, $x^{(1)}(t) = \dot{x}(t)$, and $x^{(2)}(t) = \ddot{x}(t)$. The (partial) derivative of $f(x)$ with respect to x is denoted by $f_{,x}(x) = \frac{\partial}{\partial x} f(x)$. The same notation is used for differentiable vector and matrix functions. The set of l-times continuously differentiable functions from $\mathbb{X}$ to $\mathbb{Y}$ is denoted by $\mathcal{C}^l(\mathbb{X}, \mathbb{Y})$. $\triangleleft$

In the following we investigate a spatial multibody system with holonomic as well as nonholonomic constraints [19, 33]. More precisely we consider the following initial

value problem on the domain $\mathbb{I} = [t_0, t_f]$ consisting of the equations of motion in the form

$$\begin{aligned}
\dot{p} &= Z(p)v, && \text{(1a)}\\
M(p,t)\dot{v} &= f(p,v,r,w,s,\lambda,\mu,t) - Z^T(p)G^T(p,s,t)\lambda - Z^T(p)H^T(p,s,t)\mu, && \text{(1b)}\\
\dot{r} &= b(p,v,r,w,s,\lambda,\mu,t), && \text{(1c)}\\
0 &= d(p,v,r,w,s,\lambda,\mu,t), && \text{(1d)}\\
0 &= c(p,s,t), && \text{(1e)}\\
0 &= H(p,s,t)Z(p)v + h(p,s,t) \qquad (= \breve{h}(p,v,s,t)), && \text{(1f)}\\
0 &= g(p,s,t) && \text{(1g)}
\end{aligned}$$

with the initial values

$$\begin{gathered}
p(t_0) = p_0 \in \mathbb{R}^{n_p},\ v(t_0) = v_0 \in \mathbb{R}^{n_v},\ r(t_0) = r_0 \in \mathbb{R}^{n_r},\ w(t_0) = w_0 \in \mathbb{R}^{n_w},\\
s(t_0) = s_0 \in \mathbb{R}^{n_s},\ \lambda(t_0) = \lambda_0 \in \mathbb{R}^{n_\lambda},\ \mu(t_0) = \mu_0 \in \mathbb{R}^{n_\mu}.
\end{gathered} \tag{2}$$

Here, the *position vector* p contains arbitrary position coordinates of the multibody system. The Euler-Lagrange formalism for modeling multibody systems yields the equations of motion in second order form. In order to transform the second order system to an equivalent first order system we introduce a *velocity vector* v and get the relation (1a) between the *generalized velocities* $\dot{p}$ and the velocities v with a matrix $Z(p)$, that determines the angular velocities. The equations (1a) are called *kinematic equations*. The transformation matrix $Z(p)$ occurs only if there are rotations in three dimensional space, it may be determined by Poisson's kinematical equations [1, 7]. In the two dimensional case we have $Z(p) = I,\ \dot{p} = v$.
The equations (1b) are called *dynamic equations of motion*. They follow from the equilibrium of forces and momenta and include the *mass matrix* $M(p)$, the vector of the applied and gyroscopic forces $f(p,v,r,w,s,\lambda,\mu,t)$, the *constraint matrices* $G(p,s,t)$ and $H(p,s,t)$ of the holonomic and nonholonomic constraints, respectively, which contain the inaccessible directions of motion column-wise, the associated *constraint forces* $G^T(p,s,t)\lambda$ and $H^T(p,s,t)\mu$, and the *Lagrange multipliers* λ and μ. The holonomic constraint matrix is defined as $G(p,s,t) = \frac{\mathrm{d}}{\mathrm{d}p}g(p,s(p,t),t)$. The mass matrix $M(p)$ is positive semi-definite, since the kinetic energy is a nonnegative quadratic form, and includes the inertia properties of the multibody system.
In a real multibody system, there are often *dynamic force elements* which are described by the vector r and determined by equations (1c), see [7].
Furthermore, not all constraints of a multibody system are directly described by the position variables p or the velocity variables v, but depend on certain *contact points* with coordinates s on the surface of some bodies. The relationship between these contact point coordinates s and the position variables p are given by (1e). Furthermore, the equations of motion are affected by the n_λ *holonomic constraints* (1g) and n_μ *nonholonomic constraints* (1f). These constraints are also called the *holonomic constraints on position level* and the *nonholonomic constraints on velocity level*, respectively. Sometimes, force laws and constraints may be formulated more conveniently using *auxiliary variables* w that are implicitly defined by the n_w

possibly nonlinear equation (1d).
Here, $n = n_p + n_v + n_r + n_w + n_s + n_\lambda + n_\mu$ denotes the number of unknown variables. Furthermore, many motions of mechanical systems have known solution invariants, i.e., relations which are satisfied along any motion of the mechanical system, like the invariance of the total energy, momentum, or impulse. Let us denote the m_e equations describing such solution invariants by

$$0 = e(p, v, s, t). \tag{3}$$

In particular, conservative multibody systems are energy conserving. In this case the total energy is constant along every motion of the system. For more details on solution invariants we refer to [38].
The theoretical basis of the code `GEOMS` is based on the following assumptions.

Assumption 2.2 *Consider the equations of motion (1). Then the matrices*

$$\text{a)} \quad d_{,w}, \tag{4a}$$
$$\text{b)} \quad c_{,s}, \tag{4b}$$
$$\text{c)} \quad M \tag{4c}$$
$$\text{d)} \quad \begin{bmatrix} GZM^{-1}G_\lambda & GZM^{-1}H_\mu \\ HZM^{-1}G_\lambda & HZM^{-1}H_\mu \end{bmatrix} \tag{4d}$$

are assumed to be nonsingular with a bounded inverse for all $(p, v, r, w, s, \lambda, \mu, t) \in \mathbb{M}$*, see (10), where*

$$G_\lambda = Z^T G^T - f_{,\lambda} + f_{,w} d_{,w}^{-1} d_{,\lambda}, \tag{5}$$
$$H_\mu = Z^T H^T - f_{,\mu} + f_{,w} d_{,w}^{-1} d_{,\mu}. \tag{6}$$

Furthermore, it is assumed that

$$d \in \mathcal{C}^1(\mathbb{M}, \mathbb{R}^{n_w}),\ c \in \mathcal{C}^1(\mathbb{M}, \mathbb{R}^{n_s}),\ \breve{h} \in \mathcal{C}^2(\mathbb{M}, \mathbb{R}^{n_\mu}),\ g \in \mathcal{C}^3(\mathbb{M}, \mathbb{R}^{n_\lambda}).$$

Remark 2.3 a) The nonsingularity of the mass matrix M is assumed only for reasons of simplicity. It is not necessary for the successful numerical integration with `GEOMS`.
b) Furthermore, note that in Assumption 2.2 redundant constraints are excluded. Redundant constraints may result in a nonuniqueness of the Lagrange multipliers. Nevertheless, `GEOMS` is able to deal with certain types of redundant constraints. For more details on redundant constraints see [26, 38]. ◁

Using the equations of motion (1), the first and second derivatives with respect to t of the holonomic constraints (1g) are given by

$$0 = g^I(p, v, s, t) = \frac{\mathrm{d}}{\mathrm{d}t} g(p, s, t) \tag{7a}$$
$$= GZv + g_{,t} - g_{,s} c_{,s}^{-1} c_{,t} \tag{7b}$$

and

$$0 = g^{II}(p,v,r,w,s,\lambda,\mu,t) = \frac{\mathrm{d}^2}{\mathrm{d}t^2} g(p,s,t) \tag{8a}$$

$$= (g^I_{,p} - g^I_{,s} c^{-1}_{,s} c_{,p}) Zv + GZM^{-1}(f - Z^T G^T \lambda - Z^T H^T \mu) + g^I_{,t} - g^I_{,s} c^{-1}_{,s} c_{,t}. \tag{8b}$$

They are called *holonomic constraints on velocity level* (7a) and *holonomic constraints on acceleration level* (8a), respectively. The first derivative with respect to t of the nonholonomic constraints (1f) is given by

$$0 = h^I(p,v,r,w,s,\lambda,\mu,t) = \frac{\mathrm{d}}{\mathrm{d}t}(H(p,s,t)Z(p)v + h(p,s,t)) \tag{9a}$$

$$= (\breve{h}_{,p} - \breve{h}_{,s} c^{-1}_{,s} c_{,p}) Zv + HZM^{-1}(f - Z^T G^T \lambda - Z^T H^T \mu) + (\breve{h}_{,t} - \breve{h}_{,s} c^{-1}_{,s} c_{,t}) \tag{9b}$$

which are called *nonholonomic constraints on acceleration level* (9a).
The holonomic constraints on velocity level and on acceleration level in form (7b) and (8b), respectively, as well as the nonholonomic constraints on acceleration level in form (9b) turn out to be the hidden constraints of the equations of motion, see [38]. The choice of values $(p,v,r,w,s,\lambda,\mu,t) \in \mathbb{R}^n \times \mathbb{I}$ is restricted by all constraints including the hidden constraints, i.e., (1d)-(1g), (7b), (8b), and (9b). Values which satisfy all of these constraints are called *consistent* and we get the *set of consistency*

$$\begin{aligned} \mathbb{M} = \{(p,v,r,w,s,\lambda,\mu,t) \in \mathbb{R}^n \times \mathbb{I} \ : \ 0 &= d(p,v,r,w,s,\lambda,\mu,t), \\ 0 &= c(p,s,t), \\ 0 &= H(p,s,t)Z(p)v + h(p,s,t), \\ 0 &= g(p,s,t), \\ 0 &= h^I(p,v,r,w,s,\lambda,\mu,t), \\ 0 &= g^I(p,v,s,t), \\ 0 &= g^{II}(p,v,r,w,s,\lambda,\mu,t)\}. \end{aligned} \tag{10}$$

Theorem 2.4 *Let the equations of motion (1) satisfy Assumptions 2.2. Then there exist matrix functions* $S_p \in \mathcal{C}^0(\mathbb{M}, \mathbb{R}^{n_{f_p}, n_p})$ *and* $S_v \in \mathcal{C}^0(\mathbb{M}, \mathbb{R}^{n_{f_v}, n_v})$ *with* $n_{f_p} = n_p - n_\lambda$ *and* $n_{f_v} = n_v - n_\lambda - n_\mu$ *such that the matrix functions*

$$\begin{bmatrix} S_p(p,t) \\ G(p,t) \end{bmatrix} \quad \textit{and} \quad \begin{bmatrix} S_v(p,t)M(p,t) \\ G(p,t)Z(p) \\ H(p,t)Z(p) \end{bmatrix} \quad \textit{are nonsingular} \tag{11}$$

for all $(p, v, r, w, s, \lambda, \mu, t) \in \mathbb{M}$. *Then the differential-algebraic system*

$$
\begin{aligned}
S_p(p,t)\dot{p} &= S_p(p,t)Z(p)v, && \text{(12a)}\\
S_v(p,t)M(p,t)\dot{v} &= S_v(p,t)f(p,v,r,w,s,\lambda,\mu,t) && \text{(12b)}\\
&\quad -S_v(p,t)Z^T(p)G^T(p,s,t)\lambda - S_v(p,t)Z^T(p)H^T(p,s,t)\mu,\\
\dot{r} &= b(p,v,r,w,s,\lambda,\mu,t), && \text{(12c)}\\
0 &= d(p,v,r,w,s,\lambda,\mu,t), && \text{(12d)}\\
0 &= c(p,s,t), && \text{(12e)}\\
0 &= H(p,s,t)Z(p)v + h(p,s,t), && \text{(12f)}\\
0 &= g(p,s,t), && \text{(12g)}\\
0 &= h^I(p,v,r,w,s,\lambda,\mu,t), && \text{(12h)}\\
0 &= g^I(p,v,s,t), && \text{(12i)}\\
0 &= g^{II}(p,v,r,w,s,\lambda,\mu,t) && \text{(12j)}
\end{aligned}
$$

has d-index 1 and the same set of solutions as the equations of motion (1).

Proof. The proof can be found in [38]. □

Remark 2.5 a) The matrix functions S_p and S_v are called *kinematic selector* and *dynamic selector*, respectively.
b) We will call the DAE (12) the *projected-strangeness-free formulation of the equations of motion*. In [21, 22, 23, 24] the *strangeness-concept* is introduced as tool for the classification of general nonlinear DAEs including over- and underdetermined DAEs. In particular, so called *strangeness-free* DAEs are introduced. Apart from the over- or underdeterminedness strangeness-free DAEs behave like DAEs with d-index 1 while nonstrangeness-free DAEs behave like DAEs with d-index 2 or larger. Strangeness-free DAEs do not contain hidden constraints. In particular, in [21, 22, 23, 24] it is pointed out that strangeness-free DAEs and, therefore, the projected-strangeness-free formulation of the equations of motion (12), are suited and preferable for the numerical treatment using stiff ODE solvers like implicit Runge-Kutta-Methods or BDF methods.
c) The algorithm `GEOMS` is based on a projected-strangeness-free from (12) of the equations of motion but it is not necessary that this form is provided by the user, i.e., the user does not have to perform the regularization to the projected-strangeness-free form. It is sufficient, if the user provides the constraints on velocity level (7b) and on acceleration level (8b) and (9b) in addition to the original equations of motion (1) and, if available, (3). By use of so called *order-n-formalisms* for the evaluation of the equations of motion the constraints on velocity level and on acceleration level are computed automatically, see [8, 34]. ◁

With these preparations we have presented all the tools to perform the consistency preserving index reduction of the equations of motion (1) as follows.

Algorithm 2.6 (Consistency preserving index reduction)
The equations of motion (1) are assumed to satisfy Assumptions 2.2. Furthermore, let $M \in \mathcal{C}^0(\mathbb{M}_p, \mathbb{R}^{n_v,n_v})$ and $Z \in \mathcal{C}^0(\mathbb{M}_p, \mathbb{R}^{n_p,n_v})$, where $\mathbb{M}_p = \mathbb{M} \cap (\mathbb{R}^{n_p} \times \mathbb{I})$ is the set of consistent (p,t).
Then the regularization via consistency preserving index reduction is done by choosing a selector $S_p \in \mathcal{C}^0(\mathbb{M}_p, \mathbb{R}^{n_{f_p},n_p})$ and a selector $S_v \in \mathcal{C}^0(\mathbb{M}_p, \mathbb{R}^{n_{f_v},n_v})$ depending on (p,u) with $n_{f_p} = n_p - n_\lambda$ and $n_{f_v} = n_v - n_\lambda - n_\mu$, in the following way.

1. **Determination of selector S_p**
   (a) Determine $K_p \in \mathcal{C}^0(\mathbb{M}_p, \mathbb{R}^{n_p,n_{f_p}})$ depending on (p,t) such that the columns of $K_p(p,t)$ span $\ker(G(p,s(p,t),t))$ for all $(p,t) \in \mathbb{M}_p$.
   (b) Determine the selector $S_p \in \mathcal{C}^0(\mathbb{M}_p, \mathbb{R}^{n_{f_p},n_p})$ depending on (p,t) such that $S_p(p,t)K_p(p,t)$ is nonsingular for all $(p,t) \in \mathbb{M}_p$.
2. **Determination of selector S_v**
   (a) Determine $K_v \in \mathcal{C}^0(\mathbb{M}_p, \mathbb{R}^{n_v,n_{f_v}})$ depending on (p,t) such that the columns of $K_v(p,t)$ span
   $$\ker\left(\begin{bmatrix} G(p,s(p,t),t)Z(p) \\ H(p,s(p,t),t)Z(p) \end{bmatrix} \right)$$
   for all $(p,t) \in \mathbb{M}_p$.
   (b) Determine the selector $S_v \in \mathcal{C}^0(\mathbb{M}_p, \mathbb{R}^{n_{f_v},n_v})$ depending on (p,t) such that $S_v(p,t)M(p,t)K_v(p,t)$ is nonsingular for all $(p,t) \in \mathbb{M}_p$.
3. **Projected strangeness-free form of the equations of motion**
   By appending the constraints on velocity level (7b) and the constraints on acceleration level (8b) and (9b), the projected-strangeness-free form of the equations of motion is given by (12).

With this algorithm we are able to determine a projected-strangeness-free form (12) of the equations of motion which contains all information of the set of consistency (10). The projected-strangeness-free form (12) that is created in this way is analytically equivalent to the original equations of motion in the sense that both have the same set of solutions. Therefore and because of Remark 2.5, the projected-strangeness-free form (12) can be seen as a regularization technique. In particular, the semi-implicit form of the projected-strangeness-free form (12) is of great advantage, since all constraints are stated as purely algebraic equations, and there are no redundancies among the algebraic constraints and the differential equations.

Remark 2.7 Note that Selectors S_p and S_v satisfying the rank conditions (11) are not uniquely determined. Rather it is possible to choose the selectors in a piecewise constant fashion. In principle, the selectors may be kept constant as long as the Newton iteration matrix $\mathfrak{N}$ (see Page 15) remains nonsingular. But the choice of

the selectors influences the conditioning of the projected-strangeness-free formulation. Therefore, with respect to the conditioning of the linear systems which have to be solved during the Newton iteration, the selectors should be recomputed early enough and not just shortly before reaching a state, where the Newton iteration matrix becomes singular. This fact is treated in `GEOMS` by the recomputation of the selectors if the column pivoting with respect to the algebraic constraints changes or convergence problems of the Newton iteration occur. This is demonstrated in two simulation scenarios which are depicted in Tables 3 and 4.
Note that the piecewise constant choice of the selectors is of great advantage and importance for the numerical integration, because it offers the possibility to reduce the amount of computational work for the computation of the selectors. In particular, this means, that the condition number of the Newton iteration matrix $\mathfrak{N}$ depends directly on the choice of the selectors. ◁

Example 2.8 The mathematical pendulum: Let us consider a mathematical pendulum, of length $L > 0$ which represents a point mass moving without friction along a vertical circle of radius L under gravity denoted by the gravity acceleration g. For the description of the configuration of the pendulum we choose Cartesian coordinates $p = \begin{bmatrix} x & y \end{bmatrix}^T$ denoting the position of the mass m in the two dimensional space $\mathbb{R}^2$. The equations of motion of first order have the form

$$\begin{bmatrix} \dot{p}_1 \\ \dot{p}_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \tag{13a}$$

$$\begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix} \begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -mg \end{bmatrix} - \begin{bmatrix} 2p_1 \\ 2p_2 \end{bmatrix} \begin{bmatrix} \lambda_1 \end{bmatrix}, \tag{13b}$$

$$0 = \begin{bmatrix} p_1^2 + p_2^2 - L^2 \end{bmatrix}. \tag{13c}$$

The holonomic constraints on velocity level and on acceleration level are given by

$$0 = \begin{bmatrix} 2p_1v_1 + 2p_2v_2 \end{bmatrix}, \tag{13d}$$

$$0 = \begin{bmatrix} 2v_1^2 + 2v_2^2 - 2p_2g - \frac{4}{m}(p_1^2 + p_2^2)\lambda_1 \end{bmatrix}, \tag{13e}$$

respectively. Following Algorithm 2.6 we have to consider $G = \begin{bmatrix} 2p_1 & 2p_2 \end{bmatrix}$. The matrix function K_p can be determined as

$$K_p = \begin{bmatrix} -p_2 \\ p_1 \end{bmatrix}$$

and, therefore, the selector S_p can be chosen as

$$S_p = \begin{bmatrix} -p_2 & p_1 \end{bmatrix}$$

such that

$$S_pK_p = \begin{bmatrix} -p_2 & p_1 \end{bmatrix} \begin{bmatrix} -p_2 \\ p_1 \end{bmatrix} = \begin{bmatrix} p_2^2 + p_1^2 \end{bmatrix} = \begin{bmatrix} L^2 \end{bmatrix},$$

see the constraints (13c). Since the mass matrix is given by $M = mI$, we can use $S_v = S_p$ and we get the projected-strangeness-free formulation

$$\begin{aligned} -p_2\dot{p}_1 + p_1\dot{p}_2 &= -p_2 v_1 + p_1 v_2, && \text{(14a)} \\ -mp_2\dot{v}_1 + mp_1\dot{v}_2 &= -mgp_1, && \text{(14b)} \\ 0 &= p_1^2 + p_2^2 - L^2, && \text{(14c)} \\ 0 &= 2p_1 v_1 + 2p_2 v_2, && \text{(14d)} \\ 0 &= 2v_1^2 + 2v_2^2 - 2p_2 g - \frac{4}{m}(p_1^2 + p_2^2)\lambda_1. && \text{(14e)} \end{aligned}$$

As mentioned in Remark 2.7, the selectors S_p and S_v are not uniquely determined by the conditions (11) or the Algorithm 2.6. In particular, the selectors can be chosen to be piecewise constant.
Let us consider this fact for the pendulum with the initial state $p_1 = 0$ and $p_2 = -L$, i.e., the pendulum is hanging downwards. In this position the selectors can be determined as

$$S_p(p,u) = S_v(p,u) = \begin{bmatrix} L & 0 \end{bmatrix}. \tag{15}$$

Keeping these selectors constant, the leading matrix of the left-hand side of the underlying ordinary differential equations, (obtained by substituting the algebraic equations in (14) by their derivatives with respect to t) is

$$\begin{bmatrix} L & 0 & 0 & 0 & 0 \\ 0 & 0 & mL & 0 & 0 \\ 2p_1 & 2p_2 & 0 & 0 & 0 \\ \times & \times & 2p_1 & 2p_2 & 0 \\ \times & \times & \times & \times & \frac{4}{m}(p_1^2 + p_2^2) \end{bmatrix}. \tag{16}$$

Obviously, the rank conditions (11) are fulfilled and the leading matrix (16) is non-singular, as long as p_2 does not become zero. In particular, this means that as long as the pendulum does not reach one of the horizontal positions, i.e., $p_1 = \pm L$ and $p_2 = 0$, the selectors can be chosen constant as in (15). Otherwise, if the pendulum reaches or passes the horizontal position, the matrix (16) becomes singular and the first and third as well as the second and fourth equations are redundant such that the solution is not uniquely defined. Furthermore, the condition number of matrix (16) goes to infinity as p_2 goes to zero.
For these reasons, in the neighborhood of the horizontal position of the pendulum new selectors have to be determined. See also the Example 4.1 for numerical results. ◁

3 GEOMS

The code `GEOMS` is implemented in FORTRAN77 and furthermore, there exists a MATLAB [20] interface via MEX files for the direct usage of `GEOMS` in MATLAB.

However, in the following we only discuss the in FORTRAN77 implementation of `GEOMS`.
In `GEOMS` the 3-stage implicit Runge-Kutta Method Radau IIa of order 5, see [18], as discretization of the projected-strangeness-free formulation (12) of the equations of motion is implemented. Although, `GEOMS` bases on the presented stabilization technique developed in [38] and presented in Theorem 2.4, i.e., `GEOMS` uses the projected-strangeness-free formulation (12) for the discretization, the user does not have to provide the projected-strangeness-free formulation. Instead the user has to provide all necessary information, i.e., in particular, the hidden constraints in addition to the original equations of motion (1) and , if available, (3).
The Runge-Kutta matrix A, the weight vector b, and the node vector c are given by the Butcher tableau

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} \quad \Leftrightarrow \quad \begin{array}{c|ccc} \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array}, \tag{17}$$

see [17, 18]. The algorithm `GEOMS` is designed to handle equations of motion of the form (1) with possible redundant constraints as well as with possibly known solution invariants (3) which may be provided as additional equations. If the mass matrix M is nonsingular and the constraints are nonredundant then the equations of motion have to satisfy Assumption 2.2. If this is not the case some further rank assumptions have to be satisfied. For more details see [38].
Here and in the following we will use the typewriter style for objects which are part of the source codes of the implemented numerical algorithms. In particular, this involves names of subroutines like `GEOMS`, `GEERREST`, `IVCOND`, and variables like `T`, `X`, `NWTMAT`, `CALSEL`.
In the following we will discuss the features of `GEOMS` in detail. For the use and implementation of `GEOMS` see the manual in Appendix A.

The information of the equations of motion needed from the integration algorithm has to be provided in the following form.
The *vector of unknown variables* has to be in the form

$$x^T = \mathtt{X}^T = \left[\begin{array}{ccc|c|c|cc} w^T & \lambda^T & \mu^T & r^T & v^T & s^T & p^T \end{array} \right]$$

and the *right-hand side* in (1) and (3) of the hidden constraints has to be specified in a user-supplied subroutine with a name given by the user. The different parts

have to be given in the following order provided they occur.

$$\texttt{RDA} = \left[\begin{array}{c} d(p,v,r,w,s,\lambda,\mu,t) \\ g^{II}(p,v,r,w,s,\lambda,\mu,t) \\ h^{I}(p,v,r,w,s,\lambda,\mu,t) \\ \hline g^{I}(p,v,s,t) \\ H(p,s,t)Z(p)v + h(p,s,t) \\ e(p,v,s,t) \\ \hline c(p,s,t) \\ g(p,s,t) \\ \hline\hline b(p,v,r,w,s,\lambda,\mu,t) \\ \hline f(p,v,r,w,s,\lambda,\mu,t) - Z^T(p)G^T(p,s,t)\lambda - Z^T(p)H^T(p,s,t)\mu \\ \hline Z(p)v \end{array}\right] \begin{array}{l} \left.\begin{array}{l} \\ \\ \\ \hline \\ \\ \\ \hline \\ \\ \end{array}\right\} \begin{array}{l}(a)\\(b)\\(c)\end{array} \left.\begin{array}{l}\\\\\\\\\\\\\\\\\end{array}\right\}(A) \\ \hline\hline \left.\begin{array}{l} \\ \\ \\ \end{array}\right\} \begin{array}{l}(d)\\(e)\\(f)\end{array} \left.\begin{array}{l}\\\\\\\end{array}\right\}(D) \end{array} \qquad (18)$$

In particular, the right-hand side has to be ordered such that the *algebraic part*, i.e., the upper part (18A), contains the algebraic constraints ordered with respect to their dependencies, i.e., first (18a) , the constraints which restrict the additional variables w as well as the Lagrange multipliers λ and μ, second (18b), the constraints on velocity level and the information concerning solution invariants which restrict the velocities v, and third (18c), the constraints on position level, which restrict the position p and the contact variables s. The specified order leads to a Jacobian of the algebraic part with respect to x which has already block upper triangular structure that will be exploited in `GEOMS`.
The differential part, i.e., the second part (18D), contains the right-hand side of the differential equations also ordered in the same way as the algebraic part. We first (18d) have the equations that describe the behavior of the dynamical force elements followed by (18e) the dynamical equations of motion and, finally, (18f) the kinematical equations of motion.

In some cases the constraints of acceleration level (8) and (9), i.e., $0 = g^{II}$ and $0 = h^I$, are not explicitly available or difficult to evaluate. In this case `GEOMS` is also applicable. But one should note that only if all algebraic information, including $0 = g^I$, $0 = g^{II}$, and $0 = h^I$ are provided, instabilities and drift can be avoided by `GEOMS`. It is preferable to provide as much information as possible. In the case that the constraints on acceleration level are missing, the provided information is similar to a DAE that behaves like a DAE with d-index 2.
This fact has to be communicated by the user to the code `GEOMS` with help of the option `IOPT(5)`=`FORM`. If `IOPT(5)`=0 then the projected-strangeness-free form (12) of the equations of motion will be expected as basis for the discretization. Thus, the user has to specify all information of the hidden constraints, i.e., up to acceleration level. If `IOPT(5)`=1, then the discretization will be done without specifying the constraints on acceleration level $0 = g^{II}$ and $0 = h^I$. In the latter case the used formulation of the equations of motion behaves like a system of d-index 2, i.e., it is not strangeness-free. Because of the fact that the used formulation is not strangeness-free, the success of the numerical integration depends highly sensitively on the problem and on the consistency of the initial values, in particular, on the

Option	Name	Feature	Page
		preserving invariant solutions	4
		preserving hidden constraints	5
		preserving nonholonomic constraints	2
		taking into account of redundancies in the constraints	19
IOPT(2)	LUN	optional output for integration information	
IOPT(3)	NIT	maximal number of Newton iterations	17
IOPT(4)	STARTN	starting values for the internal stages in the Newton iteration	15
IOPT(5)	FORM	incomplete regularization	11
IOPT(6)	NMAX	maximal number of integration steps	19
IOPT(8)	PRED	step size control	18
IOPT(9)	NWTMAT	approximation of the Newton matrix at x_0 or one of the extrapolated stages possible	15
IOPT(10)	NWTUPD	update of the Newton matrix	17
IOPT(11)	DECOMPC	LU, QR, or SV decomposition for the algebraic part	17
IOPT(12)	DECOMPD	LU or QR decomposition for the differential part	17
IOPT(13)	SELCOMP	selector control	18
IOPT(14)	AUTONOM	exploitation of autonomous equations of motion	19
IOPT(15)	MASSTRCT	exploitation of the structure of the mass matrix	19
IOPT(17)	IVCNSST	check and correction of the initial values with respect to its consistency	14

Table 1: Options and features of GEOMS

consistency of the Lagrange multipliers λ and μ.
An overview over the features of GEOMS is given in Table 1. Furthermore, in Table 2 the subroutines belonging to GEOMS and their task are listed.

The initial values are of great importance for the existence and the uniqueness of the solution. For the existence of a continuous solution the consistency of the initial values is necessary. In particular, admissible initial values are restricted by the (hidden) constraints. On the other hand consistent initial values, in particular, consistent initial Lagrange multipliers, are not automatically given by the modeling process and their determination by solving a system of nonlinear algebraic equations is difficult for complex multibody systems with a large number of constraints. Therefore, the algorithm GEOMS provides the possibility to determine consistent initial values.
In addition to the algebraic equations determining the set of consistency $\mathbb{M}$, see (10), the user has to define in a subroutine IVCOND additional conditions to determine consistent initial values. Such conditions offer the possibility to determine some of the freely choosable variables or to give further relations which allows a unique determination of consistent initial values.

Subroutines contained in the code GEOMS	
GEBSUBST	backward substitution of the algebraic part
GECORE	core routine
GEDECCLU	decomposition of the algebraic part with LU decomposition
GEDECCQR	decomposition of the algebraic part with QR decomposition
GEDECCSV	decomposition of the algebraic part with SV decomposition
GEDECDLU	LU decomposition of the differential part
GEELIMFXQ	elimination in the differential part according to QR decomposition of the algebraic part
GEELIMFXS	elimination in the differential part according to SV decomposition of the algebraic part
GEELIMMIQ	elimination in the mass matrix and the identity of the kinematical equations of motion according to QR decomposition of the algebraic part
GEELIMMIS	elimination in the mass matrix and the identity of the kinematical equations of motion according to SV decomposition of the algebraic part
GEERREST	error estimation, see Page 18
GEFXNUM	numerical approximation of the Jacobian of the right-hand side of the equations of motion
GEGREPEQ	picking relevant columns of the differential part according to QR decomposition
GEGREPES	picking relevant columns of the differential part according to LU and SV decomposition
GEINIVAL	determination of consistent initial values, see Page 14
GEOMS	main routine
GESOLDLU	solving the differential part by use of LU decomposition
GESOLDQR	solving the differential part by use of QR decomposition
GETRFRHSC	transformation of the right-hand side according to the algebraic part
User-supplied subroutines	
EOM	provides the reduced derivative array RDA (18)
IVCOND	provides additional initial conditions needed for the consistent initialization, see Page 12
JAC	provides the Jacobian of the reduced derivative array
MAS	provides the mass matrix
SOLOUT	output of the numerical solution and additional information during integration

Table 2: Subroutines of GEOMS

Example 3.1 The mathematical pendulum: In Example 2.8 we have introduced the mathematical pendulum. The position variables p are restricted to the circle with radius L, i.e., the constraint on position level is given by $0 = p_1^2 + p_2^2 - L^2$. If one of the position variables is given, the other is uniquely determined up to the

sign.
By defining additional conditions via the subroutine `IVCOND` the user can force the pendulum into a deviation of $\pi/4$ by setting $p_1 = L/\sqrt{2}$ or by $0 = p_1 + p_2$, for instance. Furthermore, a certain angular velocity ω can be prescribed by $0 = \sqrt{v_1^2 + v_2^2}/L - \omega$. ◁

The determination of consistent initial values is done in the subroutine `GEINIVAL` and is based on the collection of all algebraic constraints (1d)-(1g) and (7), (8), and (9) together with the conditions defined in the subroutine `IVCOND`.
The user has to decide if the given initial values are assumed to be consistent or not. By setting `IOPT(17)`=`IVCNSST`=1, the initial values are assumed to be consistent and no check of consistency or correction of the initial values is done during the run of `GEOMS`. Note that nonconsistent initial values could lead to convergence problems in the integration process which leads to an abort of the run of `GEOMS`. Otherwise, by setting `IOPT(17)`=0, the initial values are considered to be possibly inconsistent. Thus, consistency will be checked and the initial values will be corrected during the run of `GEOMS`, if necessary. If the user does not provide sufficiently many additional conditions, only the consistency is checked. If the initial values are consistent, then the integration will be continued, otherwise the run of `GEOMS` will be stopped. If the user provides more additional conditions than necessary, then the correction (if necessary) is done regarding the overdetermined nonlinear system. If all conditions together are noncontradictory, then consistent initial values will be determined. Otherwise, the Newton iteration used in this process will diverge and the run of `GEOMS` will be stopped.
The solution of the nonlinear system of equations is obtained via a simplified Newton method with the possibility of a certain number of updates of the iteration matrix, as described at Page 17. The stopping criterion is the same as that for the simplified Newton method during the integration process described at Page 16.

Remark 3.2 Note the fact that the conditions provided to `IVCOND` by the user dominate the given initial guess, i.e., if the given initial guess is consistent but does not satisfy the (possibly wrong) conditions provided by `IVCOND`, then the initial guess will be corrected in such a way that both, the constraints (1d)-(1g) and (7), (8), and (9) and the initial conditions provided to `IVCOND` are satisfied.
In case of an initial guess which is consistent to the constraints, the option `IOPT(17)` can be set to one to avoid such a correction. Otherwise, the conditions provided to `IVCOND` should be adapted. ◁

If there is only interest in the computation of consistent initial values, the user has to set `T`=`TEND` and `IOPT(17)`=0. Then the code `GEOMS` determines consistent initial values, will call the user-supplied subroutine `SOLOUT`, and finally will return to the calling subroutine.

In the following we will discuss the approach which is used in the algorithm `GEOMS` for the numerical integration of the equations of motion (1) and, if available, (3) by

use of the three stage Runge-Kutta method of type Radau IIa of order 5. Let $s = 3$ denote the number of stages.
As mentioned above, the code GEOMS combines the discretization method with the regularization technique presented in Theorem 2.4. Therefore, the algorithm uses the projected-strangeness-free form (12) as basis for the discretization. For more details on the discretization we refer to [38]. This discretization leads to a nonlinear stage equation for the determination of the three stages $X_{ki} \in \mathbb{R}^n$, $i = 1, 2, 3$ on the current integration interval $[t_k, t_{k+1}]$ with $t_{k+1} = t_k + h_k$. Here h_k denotes the current step size. The stages $X_{ki} \in \mathbb{R}^n$, $i = 1, 2, 3$ approximate the solution at the points $t_{ki} = t_k + c_i h_k$ The nonlinear stage equation has to be solved by use of a (simplified) Newton method.

A good choice of *starting values* X_{ki}^0, $i = 1, 2, 3$ is very important for the convergence of the Newton iteration. In the code GEOMS two different possibilities for the determination of starting values for the integration step from t_k to t_{k+1} are implemented. The user has to define in advance which of both shall be used during the integration process.
By setting IOPT(4)=STARTN=1 the starting values for the internal stages are chosen by $X_{ki}^0 = x_k$, $i = 1, 2, 3$, where x_k denotes the already known value which approximates the solution at the point t_k. This x_k corresponds either to the initial value in the first integration step, i.e., $k = 0$, or it corresponds to the value determined at the end of the preceding integration step.
On the other hand setting IOPT(4)=0 (which is the default) the starting values X_{ki}^0, $i = 1, 2, 3$ for the Newton iteration are obtained by evaluating the interpolation polynomial $q(t)$ of degree s over the already passed integration interval $[t_{k-1}, t_k]$ with $t_{k-1} = t_k - h_{k-1}$ and with $q(t_{k-1}) = x_{k-1}$, $q(t_{k-1} + c_i h_{k-1}) = X_{k-1i}$, $i = 1, 2, 3$. In this way we obtain the starting values for the Newton iteration as $X_{ki}^0 = q(t_k + c_i h_k)$, $i = 1, 2, 3$, where x_{k-1} denotes the numerical solution at the point t_{k-1}. In particular, this means that the new starting values in the integration step from t_k to t_{k+1} are obtained by extrapolation to the points $t_k + c_i h_k$, $i = 1, 2, 3$ based on the internal stages of the earlier integration step from t_{k-1} to t_k. Of course, this is not possible in the first step. For more details see [18].

In GEOMS a simplified Newton method is implemented. For more details on Newton methods we refer to [6]. In particular, this means that a constant Newton iteration matrix $\mathfrak{N}$ is used during the whole or several parts of the Newton iteration inside the current integration step $[t_k, t_{k+1}]$. We use the simplified Newton method, since a constant Newton iteration matrix reduces the amount of computation because of the saved evaluation of Jacobians and saved decompositions of the Newton iteration matrix in every except the first Newton iteration step. But the particular choice of the Newton iteration matrix influences the convergence of the Newton iteration. For this reason, the code GEOMS offers the possibility to choose between several reference points (X^*, t^*) for the determination of the Newton matrix. The choice has to be determined by the user by setting the option IOPT(9)=NWTMAT. The range of possible choices is related to the stages during the integration step. As discussed previously,

there are two possibilities for the choice of initial values for the Newton iteration for the determination of the internal stages. In case of `IOPT(4)`=0 the initial values are obtained by extrapolation of the solution computed so far in the points $t_k + c_i h_k$, $i = 1, 2, 3$. This offers the possibility to approximate the Newton iteration matrix at four different reference points $(X^*, t^*) = (X^0_{ki}, t_k + c_i h_k)$ for $i = 0, ..., 3$, where $c_0 = 0$ and c_i, $i = 1, 2, 3$ correspond to the node vector of the Runge-Kutta method, see Table 17. Furthermore, X^0_{ki} corresponds to the extrapolated starting values for the internal stages at the times $t_k + c_i h_k$, $i = 0, ..., 3$, and, in particular, $X^0_{k0} = x_k$ corresponds to the initial state of the current integration interval. Note that this possibility is only given if the initial values for the Newton iteration are extrapolated, i.e., if `IOPT(4)`=0. In the case of initial values chosen such that $X^0_{ki} = x_k$ for all $i = 1, 2, 3$ this possibility is not given and the Newton iteration matrix will be approximated at the initial point (x_k, t_k) with the initial state of the current integration step $[t_k, t_{k+1}]$.
Several numerical experiments have shown that the convergence of the Newton iteration can be improved by use of extrapolated initial values, i.e., `IOPT(4)`=0 in connection with an approximation of the Newton iteration matrix at the second internal stage, i.e., $(X^*, U^*) = (X^0_{k2}, t_k + c_2 h_k)$ with `IOPT(9)`=2. But, if the Newton iteration detects convergence problems, and the integration step has to be repeated with a smaller step size, then the Newton iteration matrix has to be recomputed such that the overall computation time may increase if the number of times a convergence problems is detected is large. This number is reflected in the counter `NCRJCT=IWORK(11)` which corresponds to the number of step rejections caused by convergence test failures.
It should be noted that the choice of different Newton iteration matrices within the Newton iteration is not available in the code `RADAU5`. Furthermore, the code `GEOMS` offers the possibility of a certain number of updates of the Newton iteration matrix during the Newton iteration inside of one integration step, see the following.

The convergence rate of the simplified Newton method is investigated in detail in [6], see also [17, 27]. One important question in the use of an iterative method for solving nonlinear systems inside an integration process is when to stop the iteration such that the obtained accuracy of the computed solution of the nonlinear system is within the prescribed tolerance without performing too many Newton iteration steps.
The convergence estimation and the stopping criterion implemented in `GEOMS` is described in [18] and adopted from the code `RADAU5` [17, 18]. The estimation of the convergence is based on the *weighted root square norm* $||\cdot||_{sc}$ which is defined for a $\zeta \in \mathbb{R}^n$ by

$$||\zeta||_{sc} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(\frac{\zeta_i}{sc_i} \right)^2} \tag{19}$$

with $sc_i = \texttt{ATOL}(i) + \max(|x_{ki}|, |x_{k+1i}|)\texttt{RTOL}(i)$, see [18]. This norm allows to prescribe that some solution components have to be more precisely approximated than

other. This can be specified in the vectors ATOL and RTOL prescribing the absolute and relative tolerance, respectively. For more details on the error estimation and the stopping criterion of the Newton iteration we refer to [18, 38].
In the case of a very slow convergence or, in particular, in the case of divergence, the number of Newton iteration steps has to be restricted by a maximal number k_{max} = NIT = IOPT(3). Thus, the Newton iteration will stop unsuccessfully if a) the stopping criterion is not satisfied within the maximal number k_{max} of allowed Newton iteration steps, or if b) the iteration diverges.
In case a) the user has to decide whether the whole integration step has to be rejected because of convergence failures and to be repeated with a reduced step size, or if the Newton iteration should be continued with an updated Newton iteration matrix. In GEOMS this decision is made by defining the maximal number of updates in the option IOPT(10)=NWTUPD. However, several numerical results suggest that the number of allowed updates should not exceed 1.
It should be noted that the possibility of an update of the Newton iteration matrix within the Newton iteration is not available in the code RADAU5.

During the Newton iteration a linear system has to be solved in each step. This has to be done in an efficient but stable way. The code GEOMS offers the possibility to decompose the differential part and the algebraic part via different decomposition methods. The user has to specify in the option IOPT(11)=DECOMPC if the algebraic part, i.e., the Jacobian of the constraints, should be decomposed by use of the LU decomposition with full pivoting (IOPT(11)=1), by a QR decomposition with pivoting (IOPT(11)=2), or by a SV decomposition (IOPT(11)=3). Heuristically seen, the LU decomposition with (partial) pivoting is a good compromise concerning efficiency and stability. Therefore, it is the default in GEOMS, although, the SV decomposition offers excellent stability properties but is more expensive.
Furthermore, with the option IOPT(12)=DECOMPD, the user can specify how to decompose the differential part. By setting IOPT(12)=0 the LU decomposition with partial pivoting is used and by setting IOPT(12)=1 the QR decomposition is used.

Remark 3.3 a) The separate decomposition implemented in GEOMS has the advantage that the decomposition of the algebraic part can be done independently of the step size h. Only the decomposition of the differential part has to be done separately depending on h. In particular, if the Newton iteration has convergence problems and the algorithm interrupts the Newton for to reduce the step size, then the information with respect to the algebraic part may be recycled which saves computational work.
b) For the linear algebra computations like QR decompositions and SV decompositions we use BLAS[1] (Basic Linear Algebra Subprograms) [25] and LAPACK[2] (Linear Algebra PACKage) [2] subroutines. ◁

For strangeness-free differential-algebraic systems in semi-implicit form like the projected-strangeness-free form (12) the scaling of the algebraic constraints with $1/h$ is

[1] BLAS - http://www.netlib.org/blas/
[2] LAPACK - http://www.netlib.org/lapack/

recommended in [32], where h is the current step size. Since the numerical integration of the equations of motion in GEOMS is based on the projected-strangeness-free formulation of the equations of motion, the constraints are scaled by $1/h$.

The *step size control* of the integration process is a very sensitive topic in the implementation of numerical algorithms for the integration of ODEs as well as for DAEs. An overview over several step size control strategies is given in [37], see also [4, 5, 11, 18]. The code GEOMS works with two different step size control strategies as used in the code RADAU5, but adapted to the structure of the equations of motion (1). The basis for a step size control mechanism is a local error estimation. For more details we refer to [18]. The error estimation is implemented in the subroutine GEERREST. For the choice of a new step size for the next integration step or a repeated integration step two possibilities are implemented in GEOMS which have to be selected by use of the option IOPT(8)=PRED. With IOPT(8)=2 the *classical step size controller* developed in [11] is used and with IOPT(8)=1 the *predictive step size controller*, developed by Gustafsson in [16], is used. The predictive step size control is not possible in the first step, so, the classical step size controller will be used instead. The predictive step size controller needs slightly more work and storage than the classical step size controller but is more flexible in adaptating the step size. By use of the predictive step size controller a faster reduction of the step size without step rejections is possible than by use of the classical step size controller. This leads to a possible reduction of the overall amount of computation by use of the predictive step size controller. Experiments suggest that the predictive step size controller seems to produce safer results for simple problems. On the other hand, the choice of the classical controller often produces slightly faster runs, see also [18]. The predictive step size controller will be used in GEOMS by default.

Since the code GEOMS is based on the combination of discretization and regularization to the projected-strangeness-free formulation of the equations of motion which is influenced by the choice of the selectors S_p and S_v, see Theorem 2.4, an efficient computation of these selectors is also important and will be discussed in the following.
In general, it is not necessary to recompute selectors in every integration step, see Remark 2.7.
If the LU decomposition is used for the differential part then it is possible to decide whether the determination of the selectors is done in each integration step (IOPT(13)=SELCOMP=1) or the selectors are kept constant for those integration steps where the pivoting in the algebraic part does not change (IOPT(13)=0). The latter case is the default.

The code GEOMS offers the possibility to integrate the equations of motion of form (1) with possibly redundant constraints. As discussed in the literature [26], see also Remark 2.3, the solution may not be unique in this case, but under certain conditions the nonuniqueness is only restricted to the Lagrange multipliers λ, μ, and w. For more details see [38].

Very important for the integration of equations of motion with redundant constraints is the detection of the degree of redundancy, i.e., the determination of the rank of the Jacobian associated with the constraints. The reliable numerical determination of the rank of a matrix is a delicate task and the SV decomposition is a commonly used tool for doing this. Therefore, the numerical integration of equations of motion with redundant constraints is only allowed via the SV decomposition for the constraints, i.e., `IOPT(11)`=`DECOMPC`=3.
The rank of the constraints will be determined in every integration step. If it is detected in the first step that the constraints are redundant, a reliable numerical integration requires the use of the SV decomposition at least for the decomposition of the constraints. Furthermore, if a possibly change of the rank from one step to another is detected, then the integration possibly has reached a singular point and will be stopped with an error message.

If the equations of motion have solution invariants (3), then it is often desirable to preserve these solution invariants explicitly. `GEOMS` is able to preserve solution invariants if they are provided by the user as equations (3) in the `RDA` (18). See the Example 4.1.

The user may restrict the maximal number of allowed integration steps by setting the option `IOPT(6)`=`NMAX`. The default value of `NMAX` is 100000. Furthermore, the user may force the code to exploit some special structures of the problem. If the problem is autonomous the amount of computational work for the numerical integration may be reduced. By setting `IOPT(14)`=`AUTONOM`=1 the user tells the code that the problem is autonomous and the code `GEOMS` exploits this in the integration process. The default is `IOPT(14)`=0, i.e., the problem is not autonomous. In particular, if the mass matrix is constant and/or diagonal a large amount of computational work can be saved. Therefore, the user can specify by use of `IOPT(15)`=`MASSTRKT` if the mass matrix is diagonal and constant `IOPT(15)`=4, full and constant `IOPT(15)`=3, diagonal and time and/or state dependent `IOPT(15)`=2, or full and time and/or state dependent `IOPT(15)`=1 (default).

4 Numerical experiments

In the following we will demonstrate the applicability and the performance of the new solver `GEOMS`. The integration with `GEOMS` will be performed for three different formulations of the regularized equations of motion. First, the numerical results obtained with `GEOMS` using the projected-strangeness-free form (12) of the equations of motion will be abbreviated by GEOMS(psfEoM). Second, the numerical results obtained with `GEOMS` without providing the constraints on acceleration level, see option `IOPT(5)` on Page 11, will be abbreviated by GEOMS(pEoM1). Furthermore, if the solution of the considered example satisfies some solution invariants we will use the projected-strangeness-free form of the equations of motion with explicit forcing of the solution invariants in addition to the two formulations above, see Page 4. The

numerical results in this case are denoted by GEOMS(psfEoM+I).
The numerical integrations are done on an AMD Athlon XP 1800+, 1533 MHz.
Let us note that we will abstain from the use of physical units like meters or seconds.

Example 4.1 The mathematical pendulum: In Example 2.8 we introduced the equations of motion of the mathematical pendulum and we did regularize them to the projected-strangeness-free form (14) which is used for the numerical integration via `GEOMS`.
For the numerical simulations of the movement we used the mass $m = 1$, the length $L = 1$, and the gravitational acceleration $g = 13.75$. Let us note that we did modify the gravitational acceleration to approximately $g = 13.75$ such that the exact solution has a period of 2 which allows the comparison of the accuracy every period.

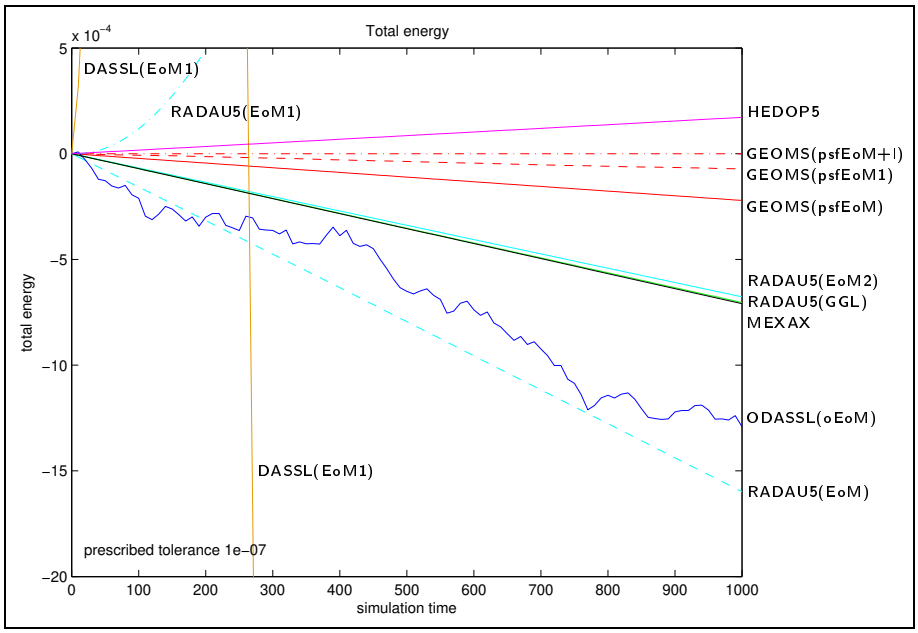

Figure 1: Mathematical Pendulum: Conservation of the total energy by the numerical solutions for prescribed `RTOL`=`ATOL`=10^{-7} on the time domain $\mathbb{I} = [0, 1000]$

The mathematical pendulum modeled as in 13 represents a mechanical system which conserves the total energy. This total energy is given by

$$E(p, v) = \frac{1}{2}m(v_1^2 + v_2^2) + mgp_2 \tag{20}$$

and is conserved such that

$$0 = E(p(t), v(t)) - E_0 = e(p, v) \quad \text{with} \quad E_0 = E(p_0, v_0) \tag{21}$$

for $t \in \mathbb{I}$ and every solution of the equations of motion (13).
Let us consider the holonomic constraints (13c) and their derivatives, which restrict

the motion of the pendulum in a nonredundant way, in comparison to the conservation of the total energy (21). We have

$$\begin{aligned} 0 &= p_1^2 + p_2^2 - L^2, && \text{(22a)} \\ 0 &= 2p_1v_1 + 2p_2v_2, && \text{(22b)} \\ 0 &= 2v_1^2 + 2v_2^2 - 2p_2g - \frac{4}{m}(p_1^2 + p_2^2)\lambda_1, && \text{(22c)} \\ 0 &= \frac{1}{2}m(v_1^2 + v_2^2) + mgp_2 - E_0. && \text{(22d)} \end{aligned}$$

The constraints (22) are nonredundant for all p, v, and λ satisfying (22). In particular, in addition to the holonomic constraints and their derivatives the energy conservation restricts the solution as well. The dimension of the solution manifold with the energy conservation is therefore smaller than without the energy conservation.

For comparison, in Figure 1 the total energy in the numerical solution is depicted. In addition to `GEOMS`, the numerical solution is computed with `RADAU5` [17, 18] for different formulations, i.e., (EoM) the equations of motion (1) of d-index 3, (EoM2) the d-index 2 formulation (using the constraints on velocity level instead of the holonomic constraints), (EoM1) the d-index 1 formulation (using the constraints on acceleration level instead of the holonomic constraints), and (GGL) the *Gear-Gupta-Leimkuhler formulation*, see [13]. Furthermore, the solution is computed with `ODASSL` [9, 10], `DASSL` [4, 31], `MEXAX` [28], and `HEDOP5` [3]. Expecting GEOMS(psfEoM+I) the numerically computed total energy is far from being constant. This can be expected because the energy conservation is contained as an equation in the used formulation and is therefore explicitely forced during the numerical integration. However, even the other numerical results obtained with `GEOMS` satisfy the conservation of total energy very accurately. The preserving of the total energy yields a stabilization of the solution.

In the Figures 2 and 3 the efficiency is depicted, i.e., the relation between the obtained accuracy and the consumed computation time of the different used formulations. Obviously, the integration with use of `GEOMS` based on the projected-strangeness-free formulation (14) plus solution invariants GEOMS(psfEoM+I) offers the best performance for this example. Note that the approximation of the Lagrange multipliers by GEOMS(psfEOM+I) is much better than of the other results.

A very important fact for the numerical integration and the stability of the numerical algorithms regarding the integration of DAEs is the satisfaction of the constraints, including the hidden constraints. In Figure 4 the residual of the constraints of position level, of velocity level, and of acceleration level depending on the simulation time is depicted. As one can see, `GEOMS` satisfies all constraints well.

Above we discussed the strategy for the determination of appropriate selectors, concerning `IOPT(13)`, see Page 18. Furthermore, the projected-strangeness-free formulation (14) of the pendulum has been developed in Example 2.8 and the choice of the selectors S_p and S_v has been considered. We have stated above that, in principle the selectors may be kept constant as long as the deviation of the pendulum does

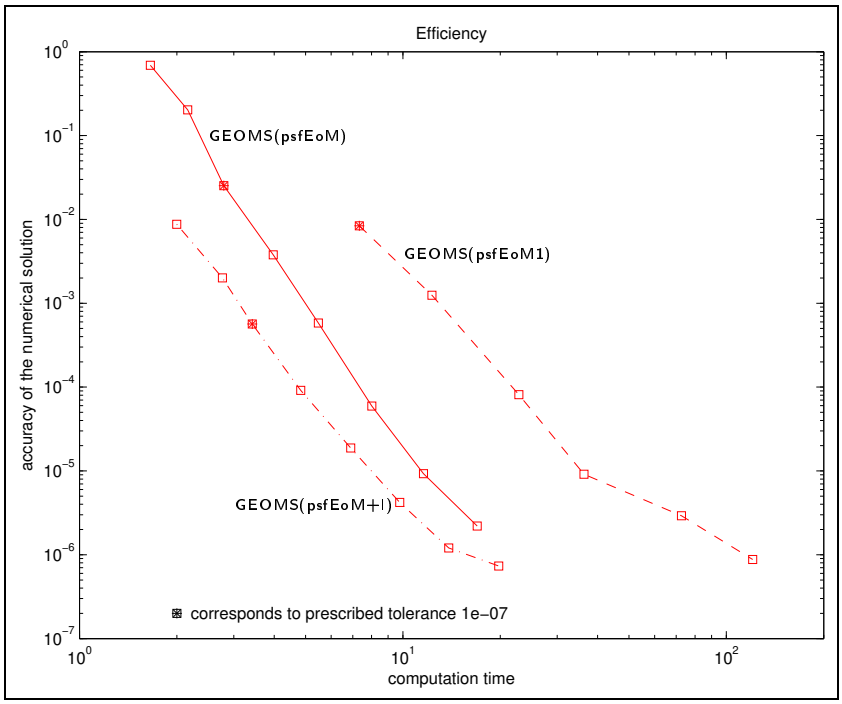

Figure 2: Mathematical Pendulum: Efficiency of the solvers based on residual evaluations. Simulations are done on the time domain $\mathbb{I} = [0, 1000]$.

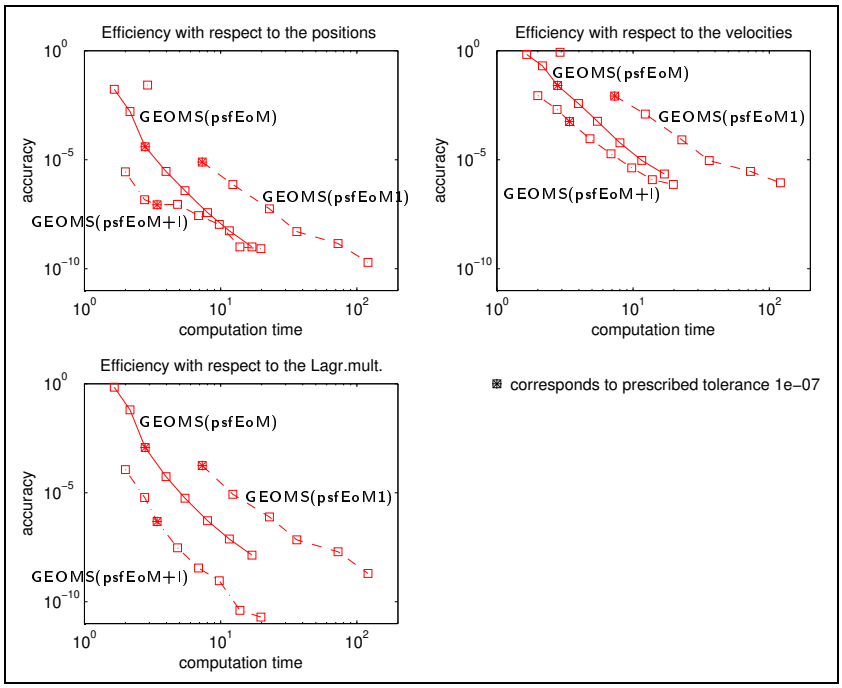

Figure 3: Mathematical Pendulum: Efficiency of the solvers based on residual evaluations. Simulations are done on the time domain $\mathbb{I} = [0, 1000]$.

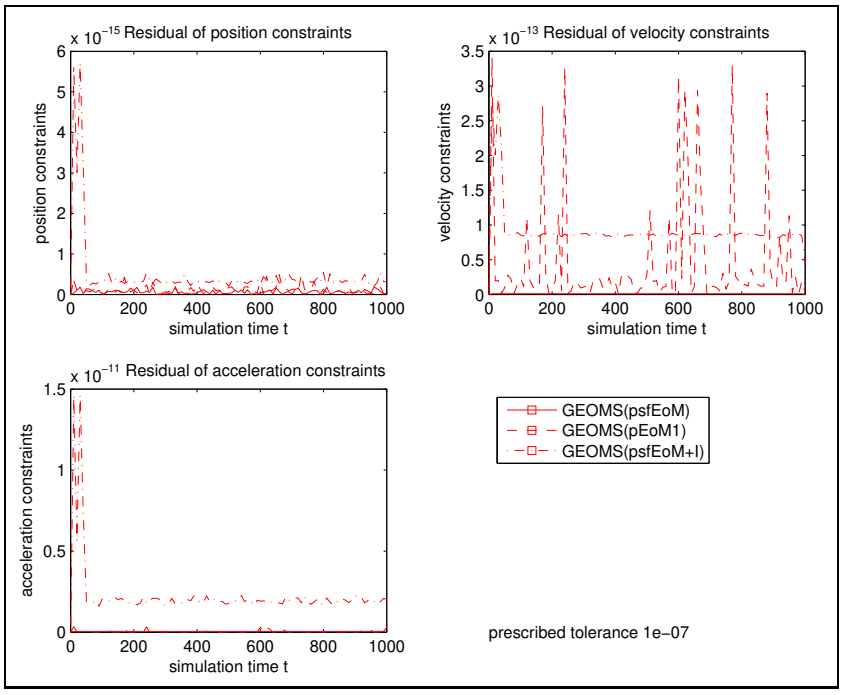

Figure 4: Mathematical Pendulum: Residual of the constraints depending on $t \in [0, 1000]$ for prescribed RTOL=ATOL=10^{-7}

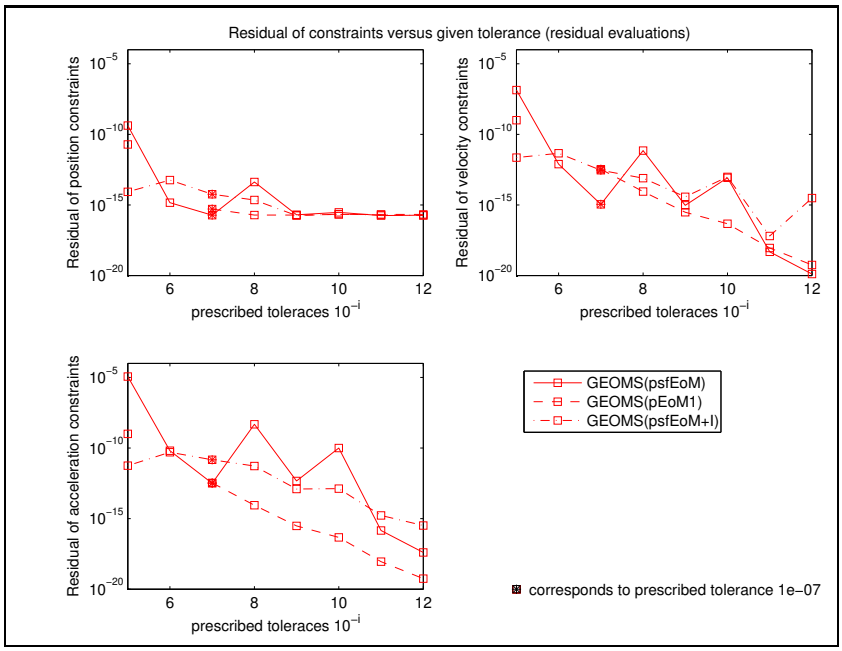

Figure 5: Mathematical Pendulum: Residual of the constraints depending on the prescribed tolerance. Simulations are done on the time domain $\mathbb{I} = [0, 1000]$ with solvers based on residual evaluations.

```
Example 01_SimpPend
 Integration with GEOMS(psfEoM)

 TSTART =  0.00     TEND   =  5.00     H0 = 0.100E-01
 TOLMIN =  1.0D- 7  TOLMAX =  1.0D- 9
 Initial velocity   2.80 rad

[SimpPend] GEOMS(psfEoM) starts   with IDID=    0 H= 0.100E-01 TOL = 0.100E-06
[SimpPend] GEOMS(psfEoM) finished with IDID=    0 H= 0.254E-01 at T= 0.500E+01
   NACCPT =    187    | NEOM=   2167 | NPDEC =    187
   NERJCT =     16    | NJAC=    187 | NEDEC =    204
   NCRJCT =      1    | NMAS=      1 | NBSUB =    660
   CPUTIME=   0.060s  |              | NSEL  =      2

[SimpPend] GEOMS(psfEoM) starts   with IDID=    0 H= 0.100E-01 TOL = 0.100E-07
[SimpPend] GEOMS(psfEoM) finished with IDID=    0 H= 0.176E-01 at T= 0.500E+01
   NACCPT =    270    | NEOM=   2961 | NPDEC =    270
   NERJCT =     13    | NJAC=    270 | NEDEC =    284
   NCRJCT =      1    | NMAS=      1 | NBSUB =    897
   CPUTIME=   0.060s  |              | NSEL  =      2

[SimpPend] GEOMS(psfEoM) starts   with IDID=    0 H= 0.100E-01 TOL = 0.100E-08
[SimpPend] GEOMS(psfEoM) finished with IDID=    0 H= 0.118E-01 at T= 0.500E+01
   NACCPT =    391    | NEOM=   4141 | NPDEC =    391
   NERJCT =      9    | NJAC=    391 | NEDEC =    401
   NCRJCT =      1    | NMAS=      1 | NBSUB =   1250
   CPUTIME=   0.080s  |              | NSEL  =      2
```

Table 3: Mathematical Pendulum: Statistical results for the numerical simulation with `GEOMS` using the psfEoM with initial velocity $v_{10} = 2.8$

not reach 90 degrees with respect to the initial state. The strategy for choosing the selectors is demonstrated in two simulation scenarios which are depicted in Tables 3 and 4.

Both scenarios simulate the motion of the pendulum starting with the downward hanging initial position $p_0 = \begin{bmatrix} 0 & -1 \end{bmatrix}^T$ and an initial velocity $v_0 = \begin{bmatrix} v_{10} & 0 \end{bmatrix}^T$ over the time domain $\mathbb{I} = [0, 5]$. In Table 3 the simulation starts with an initial velocity of $v_{10} = 2.8$. This initial velocity leads to the highest deviation of $p = \begin{bmatrix} \pm 0.699 & -0.715 \end{bmatrix}$ which does not reach the deviation of 45 degrees. Because the constraint matrix has the form $G = \begin{bmatrix} 2p_1 & 2p_2 \end{bmatrix}$ and because of $|2p_1| < |2p_2|$ for all $t \in \mathbb{I}$, a change of the pivoting is not necessary such that a (re-)computation of the selector is only necessary at the beginning of the integration process and after every detected convergence failure. Therefore, the number `NSEL` of (re-)computations of the selector equals the number `NCRJCT` of rejections because of convergence failures plus one initial computation. The situation changes completely if the pendulum passes the deviation of 45 degrees with respect to the initial state. This happens if the initial velocity is increased to $v_{10} = 2.9$. The numerical results are depicted in Table 4. Obviously, the (re-)computations of the selector `NSEL` happened 13 times

```
Example 01_SimpPend
 Integration with GEOMS(psfEoM)

 TSTART =  0.00     TEND   =  5.00     H0 = 0.100E-01
 TOLMIN =  1.0D- 7  TOLMAX =  1.0D- 9
 Initial velocity   2.90 rad

[SimpPend] GEOMS(psfEoM) starts   with IDID=    0 H= 0.100E-01 TOL = 0.100E-06
[SimpPend] GEOMS(psfEoM) finished with IDID=    0 H= 0.236E-01 at T= 0.500E+01
   NACCPT =    207    | NEOM=   2730 | NPDEC =    207
   NERJCT =      7    | NJAC=    207 | NEDEC =    227
   NCRJCT =     13    | NMAS=      1 | NBSUB =    841
   CPUTIME=   0.060s  |              | NSEL  =     26

[SimpPend] GEOMS(psfEoM) starts   with IDID=    0 H= 0.100E-01 TOL = 0.100E-07
[SimpPend] GEOMS(psfEoM) finished with IDID=    0 H= 0.121E-02 at T= 0.500E+01
   NACCPT =    303    | NEOM=   3729 | NPDEC =    303
   NERJCT =     12    | NJAC=    303 | NEDEC =    321
   NCRJCT =      6    | NMAS=      1 | NBSUB =   1142
   CPUTIME=   0.080s  |              | NSEL  =     19

[SimpPend] GEOMS(psfEoM) starts   with IDID=    0 H= 0.100E-01 TOL = 0.100E-08
[SimpPend] GEOMS(psfEoM) finished with IDID=    0 H= 0.109E-01 at T= 0.500E+01
   NACCPT =    441    | NEOM=   5208 | NPDEC =    441
   NERJCT =     12    | NJAC=    441 | NEDEC =    459
   NCRJCT =      6    | NMAS=      1 | NBSUB =   1589
   CPUTIME=   0.090s  |              | NSEL  =     19
```

Table 4: Mathematical Pendulum: Statistical results for the numerical simulation with `GEOMS` using the psfEoM with initial velocity $v_{10} = 2.9$

more often than convergence problems `NCRJCT` are detected. In Figure 6 the motion of the pendulum is depicted. One can see that the altitude of the pendulum passes 12 times the altitude of a deviation of 45 degrees. Therefore, the number `NSEL` of (re-)computations of the selectors exceeds the number `NCRJCT` of convergence problems by 13, i.e., 12 plus one initial computations of the selectors. ◁

Example 4.2 The truck model: In [36] a planar nonlinear model of a truck is introduced as benchmark example. In Figure 7 the topology as well as the coordinates, bodies, joints, and force elements are depicted. The model consists of eleven coordinates p_i, $i = 1, ..., 11$ describing the motion of seven rigid bodies and one Lagrange multiplier λ_1, see Table 5.

We omit to specify the equations of motion in detail and refer to [36] instead. Note that the equations of motion of the truck model are badly scaled, since the solution of the Lagrange multiplier λ_1 is of magnitude 10^4 but the solution of the other independent variables p and v are of magnitude 10^{-2}.

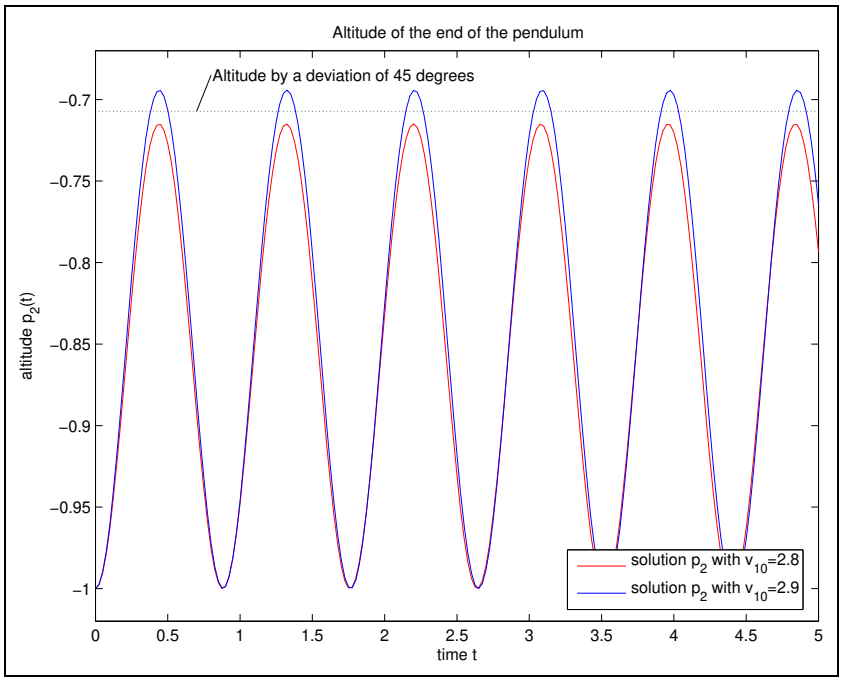

Figure 6: Mathematical Pendulum: Solution p_2 for initial velocity $v_{10} = 2.8$ and $v_{10} = 2.9$ on the time domain $\mathbb{I} = [0, 5]$.

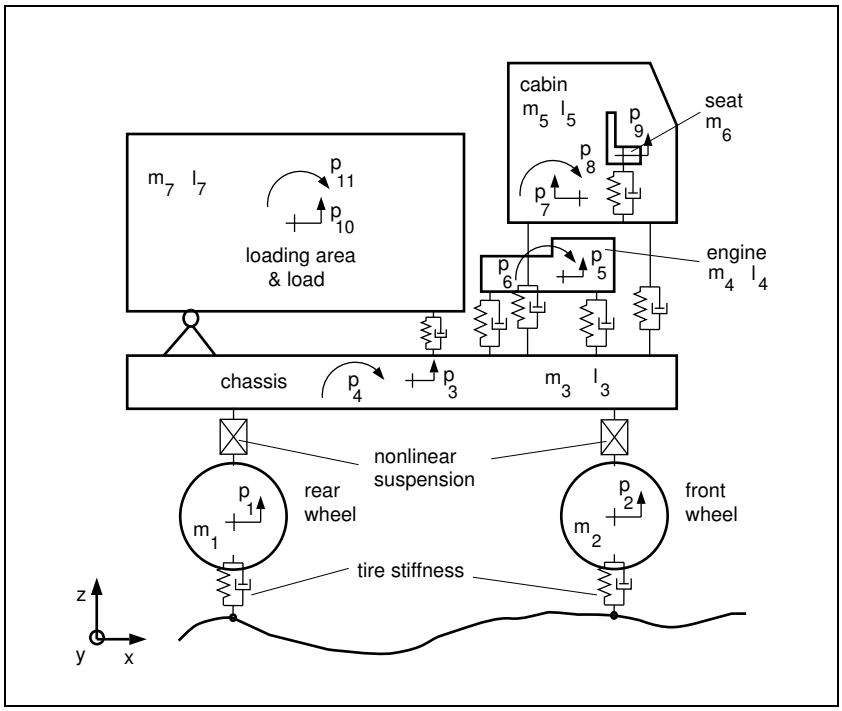

Figure 7: Topology of the truck

Body		Coordinate	
1	rear wheel	p_1	vertical motion
2	front wheel	p_2	vertical motion
3	truck chassis	p_3	vertical motion
		p_4	rotation about y-axis
4	engine	p_5	vertical motion
		p_6	rotation about y-axis
5	driver cabin	p_7	vertical motion
		p_8	rotation about y-axis
6	driver seat	p_9	vertical motion
7	loading area	p_{10}	vertical motion
		p_{11}	rotation about y-axis
		λ_1	Lagrange multiplier with respect to the joint between loading area and truck chassis

Table 5: Nonlinear truck model

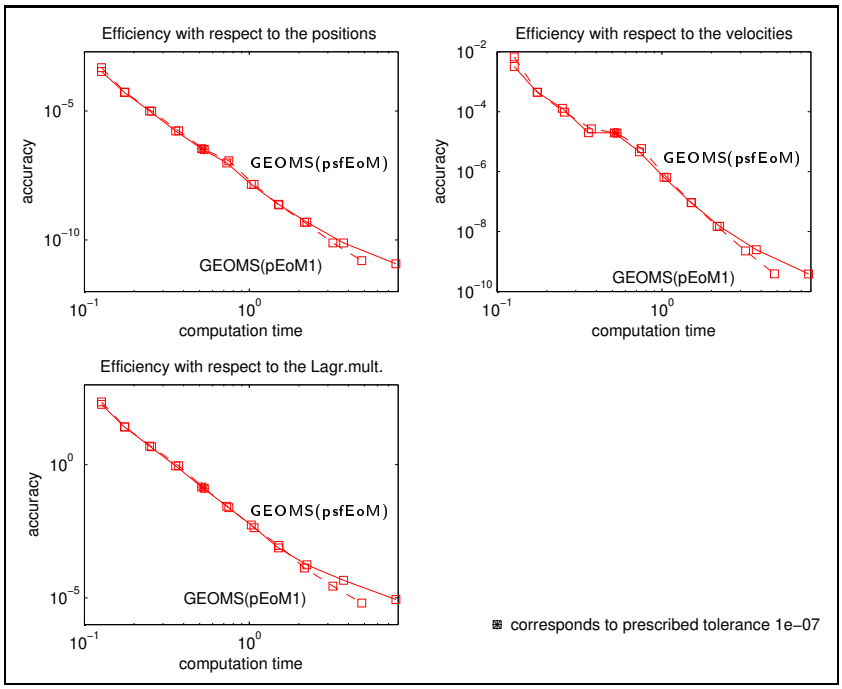

Figure 8: Truck: Efficiency of the solvers based on residual evaluations. Simulations are done on the time domain $\mathbb{I} = [0, 20]$.

The obtained accuracy of the numerical solutions is compared with the numerical solution RADAU5(GGL) obtained with a prescribed tolerance RTOL=ATOL=10^{-15}. The precision of all results obtained by a prescribed tolerance are of similar accuracy but the consumed computation time differs, as seen in the Figure 8. By the use of the code GEOMS no problem occurred in the numerical integrations for any prescribed

tolerances RTOL=ATOL$\geq 10^{-15}$. ◁

5 Summary

In this paper we have presented the new numerical algorithm GEOMS for the numerical integration of general equations of motion.
In particular, the algorithm GEOMS has been developed to carry out the numerical integration of the most general form of the equations of motion, including nonholonomic constraints and possible redundancies in the constraints, as they may appear in industrial applications. Besides the numerical integration it offers some additional features like preservation of invariant solutions, preservation of hidden constraints, use of different decomposition strategies, use of an incomplete regularization, and also checking and correction of the initial values with respect to their consistency.
Subsequently, we have demonstrated the performance and the applicability of the algorithm for two mechanical examples of different degrees of complexity. The experience with these numerical examples and several other numerical tests suggest that the code GEOMS is an efficient and robust method for the numerical integration of the equations of motion.

A Manual of GEOMS

```
      SUBROUTINE GEOMS(
     #   NP,NV,NR,NW,NS,NL,NM,NI,M,N,NIVCOND,
     #   X,T,TEND,H,RTOL,ATOL,ITOL,IOPT,ROPT,
     #   IVCOND,EOM,MAS,JAC,IJAC,
     #   SOLOUT,IOUT,
     #   LIWORK,IWORK,LRWORK,RWORK,
     #   RPAR,IPAR,IERR,
     #   IDID)
C -----------------------------------------------------------------------------
C
C  NAME      : (G)eneral (E)quations (O)f (M)otion (S)olver
C
C  PURPOSE   : This subroutine performs the numerical simulation
C              of a multibody system whose state is described by
C
C              p - position variables                  of dimension NP,
C              v - velocity variables                  of dimension NV,
C              r - dynamical force element variables   of dimension NR,
C              w - auxiliary variables                 of dimension NW,
C              s - contact point variables             of dimension NS,
C              l - holonomic Lagrange multipliers      of dimension NL,
C              m - nonholonomic Lagrange multipliers   of dimension NM
C
C              by numerical integration of the equations of motion
C              of the form
C
```

```
C                        p'= Z(p)*v,                                  (1) (f_kin)
C                 M(p,t)*v'= f(p,v,r,w,s,l,m,t)-ZT(p)*GT(p,s,t)*l
C                                              -ZT(p)*HT(p,s,t)*m, (2) (f_dyn)
C                        r'= b(p,v,r,w,s,l,m,t),                      (3)
C                        0 = d(p,v,r,w,s,l,m,t),                      (4)
C                        0 = c(p,s,t),                                (5)
C                        0 = H(p,s,t)Z(p)v+h(p,s,t)                   (6)
C                        0 = g(p,s,t),                                (7)
C                        0 = e(p,v,s,t)                               (8)
C
C              on the domain [t_0,t_f]=[T,TEND].
C
C              The prime denotes the time derivative, e.g., p'=dp/dt, and the
C              'T' following a matrix or vector denotes the transpose of this
C              matrix or vector, e.g.,GT is the  transpose of G and ZT is the
C              transpose of Z. Furthermore, the equations correspond to
C
C              (1) Kinematical equations of motion      of dimension NP,
C              (2) Dynamical equations of motion        of dimension NV,
C              (3) Dynamical force element equations    of dimension NR,
C              (4) Additional equations for variables w of dimension NW,
C              (5) Contact equations                    of dimension NS,
C              (6) Nonholonomic constraints             of dimension NM,
C                    Notation: h~(p,v,s,t)=H(p,s,t)Z(p)v+h(p,s,t)
C              (7) Holonomic constraints                of dimension NL,
C              (8) Solution invariants                  of dimension NI.
C
C              The System (1)-(8) has to satisfy the following.
C               a) G = dg/dp - dg/ds*(dc/ds)^{-1}*dc/dp.
C               b)      [ GZM^{-1}Gl   GZM^{-1}Hm]
C                  rank([                        ])=rank(G)+rank(H)=constant
C                       [ HZM^{-1}Gl   HZM^{-1}Hm]
C                  with Gl=ZT*GT-df/dl+df/dw*(dd/dw)^{-1}*dd/dl
C                  and  Hm=ZT*HT-df/dm+df/dw*(dd/dw)^{-1}*dd/dm
C                  for all t in [T,TEND].
C                  Alternatively,
C                       [ M  Gl Gm ]
C                  rank([ GZ 0  0  ])=NV+rank(G)+rank(H)
C                       [ HZ 0  0  ]
C                  has to be satisfied for all t in [T,TEND].
C               c) dc/ds has to be nonsingular for all times t in [T,TEND].
C               d) dd/dw has to be nonsingular for all times t in [T,TEND].
C               e) de/dv has to have full rank for all times t in [T,TEND].
C
C              The integration method used is the implicit Runge-Kutta method
C              (Radau IIa) of order 5 with step size control, continuous
C              output, and consistent initialization.
C
C  METHOD   :  The equations of motion are integrated by the implicit
C              Runge-Kutta method of type RADAU IIa of order 5 and using the
C              projected-strangeness-free formulation or the
C              projected-strangeness-index-1 formulation of the equations of
C              motion.
C
```

```
C  VERSION   : April 12, 2006
C
C  REVISIONS : -
C
C  AUTHORS   : Address: A. Steinbrecher
C                       Weierstrass Institute for Applied Analysis
C                           and Stochastics
C                       Forschungsverbund Berlin e.V.
C                       Mohrenstr. 39
C                       10117 Berlin
C              e-mail:  steinbrecher@wias-berlin.de
C
C  REFERENCES: This code is part of the PhD thesis:
C              A.Steinbrecher. Numerical Solution of Quasi-Linear Differential-
C              Algebraic Equations and Industrial Simulation of Multibody
C              Systems. PhD thesis, TU Berlin, Institut fuer Mathematik, 2006
C
C  KEYWORDS  : numerical simulation of mechanical systems, equations of motion,
C              differential-algebraic equations, projected-strangeness-free
C              formulation, projected-strangeness-index-1 formulation
C
C  NOTE      : The (basic) linear algebra routines are provided by the
C              libraries BLAS and LAPACK
C
C  DISCLAIMER: Warranty disclaimer: The software is supplied "as is" without
C              warranty of any kind. The copyright holder:
C              (1) disclaim any warranties, express or implied, including but
C                  not limited to any implied warranties of merchantability,
C                  fitness for a particular purpose, title or non-infringement,
C              (2) do not assume any legal liability or responsibility for the
C                  accuracy, completeness, or usefulness of the software,
C              (3) do not represent that use of the software would not
C                  infringe privately owned rights,
C              (4) do not warrant that the software will function
C                  uninterrupted, that it is error-free or that any errors
C                  will be corrected.
C              Limitation of liability: In no event will the copyright holder:
C              be liable for any indirect, incidental, consequential, special
C              or punitive damages of any kind or nature, including but not
C              limited to loss of profits or loss of data, for any reason
C              whatsoever, whether such liability is asserted on the basis
C              of contract, tort (including negligence or strict liability),
C              or otherwise, even if any of said parties has been warned of
C              the possibility of such loss or damages.
C
C -----------------------------------------------------------------------------
C
C  CALL
C  --------------------------
C
C      SUBROUTINE GEOMS(
C     #   NP,NV,NR,NW,NS,NL,NM,NI,M,N,NIVCOND,
C     #   X,T,TEND,H,RTOL,ATOL,ITOL,IOPT,ROPT,
C     #   IVCOND,EOM,MAS,JAC,IJAC,
```

```
C     #   SOLOUT,IOUT,
C     #   LIWORK,IWORK,LRWORK,RWORK,
C     #   RPAR,IPAR,IERR,
C     #   IDID)
C      IMPLICIT NONE
C      INTEGER          NP,NV,NR,NW,NS,NL,NM,NI,M,N,NIVCOND,
C     #                 ITOL,IJAC,IOUT,LIWORK,LRWORK,IERR,IDID,
C     #                 IOPT(40),IWORK(LIWORK),IPAR(*)
C      DOUBLE PRECISION T,TEND,H,
C     #                 X(N),RTOL(*),ATOL(*),ROPT(40),RWORK(LRWORK),
C                      RPAR(*)
C      EXTERNAL         IVCOND,EOM,MAS,JAC,SOLOUT
C
C  INPUT- AND OUTPUT-ARGUMENTS
C  ---------------------------
C
C     NP       Input  : integer
C              Number of position variables p.
C
C     NV       Input  : integer
C              Number of velocity variables v.
C
C     NR       Input  : integer
C              Number of dynamical force element variables r.
C
C     NW       Input  : integer
C              Number of auxiliary variables w.
C
C     NS       Input  : integer
C              Number of contact point variables s.
C
C     NL       Input  : integer
C              Number of Lagrange multipliers l=lambda for holonomic
C                constraints.
C
C     NM       Input  : integer
C              Number of Lagrange multipliers m=mu for nonholonomic
C                constraints.
C
C     NI       Input  : integer
C              Number of invariants, e.g., energy conservation.
C
C     M        Input  : integer
C              Total number of provided equations (M.GE.N), i.e., dimension of
C              RDA, see subroutine EOM. In the case of the use of the
C              * projected-strangeness-free formulation we have
C                  M=NP+NV+NR+NW+NS+3*NL+2*NM+NI,
C              * projected-strangeness-index-1 formulation we have
C                  M=NP+NV+NR+NW+NS+2*NL+NM+NI.
C
C     N        Input  : integer
C              Number of unknowns (M.GE.N), i.e., dimension of X. We have
C              N=NP+NV+NR+NW+NS+NL+NM.
C
```

```
C     NIVCOND Input  : integer
C             Number of initial  value conditions, which have to be satisfied
C             in addition to the constraints obtained from the provided equa-
C             tions of motion. See subroutine IVCOND.
C
C     X       Input  : double precision array X(N)
C             Initial values for X. The array X contains  the (initial) state
C             of the mechanical system in the following order
C
C                 X(1:NW)                                        =w
C                 X(NW+1:NW+NL)                                  =l (=lambda)
C                 X(NW+NL+1:NW+NL+NM)                            =m (=mu)
C                 -------------------------------------------------------
C                 X(NL+NM+NW+1:NL+NM+NW+NR)                      =r
C                 -------------------------------------------------------
C                 X(NL+NM+NW+NR+1:NL+NM+NW+NR+NV)                =v
C                 -------------------------------------------------------
C                 X(NL+NM+NW+NR+NV+1:NL+NM+NW+NR+NV+NS)          =s
C                 X(NL+NM+NW+NR+NV+NS+1:NL+NM+NW+NR+NV+NS+NP) =p
C
C             Output :
C             Numerical approximation of the solution at the last successfully
C             reached time T.
C
C     T       Input  : double precision
C             Initial time.
C             Output :
C             Last successfully reached time. If the whole integration was
C               successful then T=TEND.
C
C     TEND    Input  : double precision
C             Final time.
C
C     H       Input  : double precision
C             Initial step size.
C             Output :
C             Last used step size.
C
C     RTOL    Input  : double precision RTOL (or array RTOL(N))
C     ATOL    Input  : double precision ATOL (or array ATOL(N))
C             Relative and absolute error tolerances. They can be both
C               scalars or else both vectors of length N.
C             In the case of a scalar the prescribed relative and absolute
C               tolerances are valid for every component of the vector of
C               unknowns X. The code keeps, roughly, the local error of X(I)
C               below RTOL*ABS(X(I))+ATOL.
C             In the case of a vector of dimension N the prescribed relative
C               tolerances RTOL(I) and absolute tolerances ATOL(I) are valid
C               for the I-th component X(I) of the vector of unknowns X.
C               The code keeps, roughly, the local error of X(I) below
C               RTOL(I)*ABS(Y(I))+ATOL(I).
C
C     ITOL    Input  : integer
C             Switch for RTOL and ATOL:
```

```
C             ITOL=0  Both RTOL and ATOL are scalars.
C             ITOL=1  Both RTOL and ATOL are vectors.
C
C     IOPT    Input : integer array IOPT(40)
C             Serve as parameters for the code. For standard use of the code
C             IOPT(2),..,IOPT(17) must be set to zero before calling.
C             See below for a more sophisticated use.
C
C             IOPT( 2)=LUN output device
C                      0  - no output (default)
C                      6  - output to the screen
C                     >10 - other output devices (to define)
C                      In the case that the output of several messages is de-
C                      sired, the user has to define an  output device and to
C                      associate this device with IOPT(2), e.g.,
C                        IOPT(2)=13
C                        OPEN(UNIT=13,FILE='geoms.log')
C                      Finally, the output device has to be closed, e.g.,
C                        CLOSE(13)
C                      In the case of an unsuccessful  run of GEOMS it is re-
C                      commended  to set IOPT(2) > 0 such  that GEOMS is able
C                      to provide more detailed informations to the user.
C                      Furthermore, it is recommended to set
C                        IOPT(2)=0, 6, or >10.
C
C             IOPT( 3)=NIT maximum number of  Newton iterations for the solu-
C                      tion of the implicit system in each step.
C                      The default value (for IOPT(3)=0) is 10.
C
C             IOPT( 4)=STARTN defines the choice  of starting  values for the
C                      Newton method solving the nonlinear stage equations
C                      0 - The extrapolated collocation solution is  taken as
C                          starting value for Newton method. (default)
C                      1 - Zero  starting values are  used as  starting value
C                          forNewton method.
C                      IOPT(4)=1 is recommended if the Newton method has con-
C                      verging difficulties (this is the case  when IWORK(11)
C                      is very large  in comparison to  IWORK(1),  see output
C                      parameters).
C
C             IOPT( 5)=FORM    Used  formulation  as basis  of the  numerical
C                      integration
C                      0 - projected-strangeness-free  formulation, i.e., the
C                            user has to provide  the equations (1)-(7) toge-
C                            ther with  the first and  second time derivative
C                            of the holonomic constraints, i.e.,
C                            gI(p,v,t)           = d/dt g(p,t),
C                            gII(p,v,r,w,s,l,m,t)= d^2/dt^2 g(p,t),
C                            and the first time derivative of the nonholonomic
C                            constraints, i.e.,
C                            hI(p,v,r,w,s,l,m,t)= d/dt(H(p,s,t)Z(p)v+h(p,s,t)).
C                            If there exist some solution invariants (8) the
C                            user should also provide them and set NI equal
C                            to the number of the solution invariants. All
```

```
C                         provided equations have to be defined in the sub-
C                         routine EOM and the subroutine MAS in the correct
C                         order, see the subroutines EOM and MAS for more
C                         details.
C                   1 - projected-strangeness-index-1 formulation , i.e.,
C                         the user has to provide the equations  (1)-(7)
C                         together with the first time derivative of the
C                         holonomic constraints, i.e.,
C                         gI(p,v,t)            = d/dt g(p,t).
C                         If there exist some solution invariants (8) the
C                         user should also provide them and set NI equal
C                         to the number of the solution invariants. All
C                         provided equations have to be defined in the sub-
C                         routine EOM and the subroutine MAS in the correct
C                         order, see the subroutines EOM and MAS for more
C                         detail.
C
C             IOPT( 6)=NMAX    Maximal number of allowed steps.
C                      The default value (for IOPT(6)=0) is 100000.
C                      If the code stops with the error message IDID=-1117,
C                      IOPT(6) has to be increase or
C                      the integration can be continued by use of the obtained
C                      X and T as initial values for the continued integration.
C
C             IOPT( 8)=PRED    Step size strategy
C                       1 -  predictive controller (Gustafsson)
C                       2 -  classical step size control
C                      The default value (for IOPT(8)=0) is 1.
C                      The choice IOPT(8)=1 seems to produce safer results;
C                      for simple problems, the choice IOPT(8)=2 produces
C                      often slightly faster runs.
C
C             IOPT( 9)=NWTMAT   Approximation of the Newton iteration matrix
C                       0 - approximation at the initial point x_i of the
C                             current integration interval [t_{i},t_{i+1}]
C                             i.e., at (t_{i},x_{i}) (default)
C                       1 - approximation at the first extrapolated stage of
C                             the current integration interval [t_{i},t_{i+1}]
C                             i.e., at (t_{i}+c_{1}*h,X_{i1})
C                       2 - approximation at the second extrapolated stage of
C                             the current integration interval [t_{i},t_{i+1}]
C                             i.e., at (t_{i}+c_{2}*h,X_{i2})
C                       3 - approximation at the third extrapolated stage of
C                             the current integration interval [t_{i},t_{i+1}]
C                             i.e., at (t_{i}+c_{3}*h,X_{i3})
C                      Several numerical experiments turned out that the
C                      choice IOPT(9)=2 is the fastest while the
C                      choice IOPT(9)=0 is theoretically the safest.
C                      IOPT(9).NE.0 is only possible if IOPT(4)=STARTN=0,
C                      i.e., the extrapolated collocation solution is taken
C                      as starting value for Newton method.
C
C             IOPT(10)=NWTUPD   Update of the Newton iteration matrix
C                       0 - for the whole Newton iteration process in one
```

```
C                         integration step the same Newton iteration matrix
C                         is used, i.e., no update is allowed. (default)
C                    >0 - during the Newton iteration process in the current
C                         integration step IOPT(10) updates of the Newton
C                         iteration matrix are allowed.
C                    If convergence problems during the Newton iteration
C                    process occur, often the Newton matrix is not suitable.
C                    Therefore, in the case of IOPT(10)=0 the
C                    current integration step is rejected, the counter NCRJCT
C                    will be increased by one and the current integration
C                    step will be repeated with reduced step size.
C                    The option IOPT(10)>0 allows the update of the Newton
C                    iteration matrix IOPT(10) times. The Newton iteration
C                    matrix will be updated by use of the current iterates
C                    and the Newton iteration will be continued.
C                    Several numerical experiments have shown that IOPT(10)
C                    should not exceed 1.
C
C               IOPT(11)=DECOMPC  Decomposition of the algebraic part
C                     0 - LU decomposition with full pivoting (default)
C                     1 - QR decomposition with pivoting
C                     2 - SV decomposition
C                    By use of IOPT(11)=1 the integration becomes fastest
C                    but the stability of the decomposition can not be
C                    guaranteed. In situations with isolated singularities
C                    it may happen that the integrator does not detect
C                    the singularity if the tolerances RTOL or ATOL are
C                    too large.
C                    By use of IOPT(11)=2 or 3 the stability of the
C                    decomposition is guaranteed but the integration
C                    becomes slower.
C                    In case of redundant constraints only IOPT(11)=3 is
C                    possible.
C
C               IOPT(12)=DECOMPD  Decomposition of differential part
C                    0 - LU decomposition with partial pivoting (default)
C                    1 - QR decomposition
C                    By use of IOPT(12)=0 the integration becomes fastest.
C
C               IOPT(13)=SELCOMP  Recomputation strategy for the selectors
C                     0 - situation adapted
C                         the recomputation of the selector will be done
C                         only if the row pivoting of the constraints is
C                         changing or convergence problems occur during
C                         the Newton iteration process (default)
C                     1 - in every integration step
C                    In case of IOPT(13)=0 the amount of computations
C                    is reduced and the integration becomes faster.
C                    This speed-up is only possible if DECOMPD=0.
C
C               IOPT(14)=AUTONOM  Autonomy of the equations of motion
C                     0 - the equations of motion are not autonomous
C                           (default)
C                     1 - the equations of motion are autonomous
```

```
C                    If the equations of motion are autonomous the amount
C                    of computations can be reduced and the integration
C                    becomes faster.
C
C            IOPT(15)=MASSTRKT Structure of the mass matrix
C                     1 - full and time or/and state dependent
C                     2 - diagonal and time or/and state dependent
C                     3 - full and constant
C                     4 - diagonal and constant
C                    The default value (for IOPT(15)=0) is 1.
C
C            IOPT(17)=IVCNSST are the initial values consistent
C                     0 - No, the initial values are assumed to be not
C                           consistent. A check of consistency will be
C                           done and if necessary a correction will be
C                           computed. (default)
C                     1 - Yes, the initial values are assumed to be
C                           consistent. No check of consistency will be
C                           done.
C
C     ROPT    Input : double precision array ROPT(40)
C             Serve as parameters for the code. For standard use of the code
C             ROPT(1),..,ROPT(40) must be set to zero before calling.
C             See below for a more sophisticated use.
C
C             ROPT( 1)=UROUND   The rounding unit
C                      The default value (for ROPT(1)=0.0) is 1.D-16.
C
C             ROPT( 2)=SAFE     Safety factor in step size prediction
C                      The default value (for ROPT(2)=0.0) is 0.9.
C
C             ROPT( 3)=THET     Recomputation of the Jacobian
C                      Decides whether the Jacobian should be recomputed.
C                      Increase ROPT(3), to 0.1 say, when Jacobian evaluations
C                      are costly. for small systems ROPT(3) should be smaller
C                      (say 0.001D0). Negative ROPT(3) forces the code to
C                      compute the Jacobian after every accepted step.
C                      The default value (for ROPT(3)=0.0) is 0.001D0.
C
C             ROPT( 4)=FNEWT    Stopping criterion for Newton's method
C                      Smaller values of ROPT(4) make the code slower, but
C                      safer.
C                      The default value (for ROPT(4)=0.0) is
C                      MIN(0.03D0,RTOL(1)**0.5D0)
C
C
C             ROPT( 5)=QUOT1    Change of the step size
C                      See ROPT(6).
C                      The default value (for ROPT(5)=0.0) is 1.0D0
C
C             ROPT( 6)=QUOT2    Change of the step size
C                      If QUOT1 < HNEW/HOLD < QUOT2, then the step size is not
C                      changed. This saves, together with a large ROPT(3),
C                      decompositions and the amount of computations for
```

```
C                     large systems. For small systems one may have
C                     ROPT(5)=1.00D0, ROPT(6)=1.2D0, for large full systems
C                     ROPT(5)=0.99D0, ROPT(6)=2.0D0  might be good choices.
C                     The default value (for ROPT(6)=0.0) is 1.2D0
C
C             ROPT( 7)=HMAX     Maximal step size
C                     The default value (for ROPT(7)=0.0) is TEND-T
C
C             ROPT( 8)=FACL     PARAMETER FOR STEP SIZE SELECTION
C                     See ROPT(9).
C                     The default value (for ROPT(9)=0.0) is 8.0D0
C
C             ROPT( 9)=FACR     Step size selection
C                     The new step size is chosen subject to the restriction
C                     FACR <= HNEW/HOLD <= FACL
C                     The default value (for ROPT(8)=0.0) is 0.2D0
C
C     IVCOND   User supplied subroutine which provides initial conditions in
C                     addition to the constraints contained in the equations
C                     of motion (including hidden constraints)
C
C                 SUBROUTINE IVCOND(N,T,X,NCOND,COND,IPAR,RPAR,IERR)
C                 IMPLICIT NONE
C                 INTEGER          N,NCOND,IPAR(*),IERR
C                 DOUBLE PRECISION T,X(N),COND(NCOND),RPAR(*)
C
C                 N     Input  : integer
C                       Number of unknowns, i.e., dimension of X
C                       X has to remain unchanged.
C
C                 T     Input  : double precision
C                       Initial time t_0.
C                       T has to remain unchanged.
C
C                 X     Input  : double precision array X(N)
C                       Unknown variables, see above.
C                       X has to remain unchanged.
C
C                 NCOND Input  : integer
C                       Number of additional initial conditions provided in the
C                       subroutine IVCOND.
C                       NCOND has to remain unchanged.
C
C                 COND  Output : double precision array COND(NCOND)
C                       Residual of initial conditions, e.g. the condition
C                       COND(1)=X(4)-.5 forces the initial state of X(4) to
C                       be 0.5, i.e. X(4)=0.5D0.
C                       Note the fact, that the conditions given in IVCOND
C                       override the given initial values, i.e., if the given
C                       initial values are consistent but do not satisfy the
C                       (possibly wrong) conditions given in IVCOND the
C                       initial values will be corrected such that both,
C                       the constraints and the initial conditions are
C                       satisfied.
```

```
C                     In case of initial values which are consistent to
C                     the constraints the option IOPT(17) could be set to 1
C                     to avoid such a correction.
C
C                IPAR  Input/Output: integer array IPAR(*)
C                      Integer parameters which are only used by the user.
C                      They are unused and unchanged by GEOMS.
C
C                RPAR  Input/Output: double precision array RPAR(*)
C                      Double precision parameters which are only used by the
C                      user. RPAR is unused and unchanged by GEOMS.
C
C                IERR  Output : integer
C                      Indicator of success. IERR is only used by
C                      user supplied subroutines. After every call of a user
C                      supplied  subroutine the status of IERR is checked. If
C                      IERR is negative  the run of GEOMS will be interrupted
C                      and GEOMS returns to the  calling program. IERR  is
C                      unchanged by GEOMS.
C
C     EOM      Name (EXTERNAL) of the user supplied subroutine which provides
C              the right-hand side (RHS) of EoM (1)-(8) together with the first
C              and second time derivative of the holonomic constraints, i.e.,
C                gI(p,v,t)            = d/dt g(p,t),
C                gII(p,v,r,w,s,l,m,t) = d^2/dt^2 g(p,t),
C              and the first time derivative of the nonholonomic constraints,
C              i.e.,
C                hI(p,v,r,w,s,l,m,t)  = d/dt (H(p,s,t)Z(p)v+h(p,s,t)).
C              The order and the number of the provided right-hand sides
C              depends on the used formulation, see IOPT(5) and above for more
C              detail.
C
C              SUBROUTINE EOM(M,N,T,X,RDA,IOPT,ROPT,IPAR,RPAR,IERR)
C              IMPLICIT NONE
C              INTEGER          M,N,IOPT(*),IPAR(*),IERR
C              DOUBLE PRECISION T,X(N),RDA(M),ROPT(*),RPAR(*)
C
C                 M     Input  : integer
C                       Total umber of provided equations (M.GE.N),
C                       i.e., dimension of RDA, see below.
C                       In the case of use of
C                       * projected-strangeness-free formulation we have
C                         M=NP+NV+NR+NW+NS+3*NL+2*NM+NI,
C                       * projected-strangeness-index-1 formulation we have
C                         M=NP+NV+NR+NW+NS+2*NL+NM+NI.
C                       M has to remain unchanged.
C
C                 N     Input  : integer
C                       Number of unknowns (M.GE.N), i.e., dimension of X.
C                       We have N=NP+NV+NR+NW+NS+NL+NM.
C                       N has to remain unchanged.
C
C                 T     Input  : double precision
C                       Evaluation of the right-hand side of the provided
```

```
C                       equations at time T.
C                       T has to remain unchanged.
C
C                 X     Input  : double precision array X(N)
C                       Vector of unknowns, see above.
C                       X has to remain unchanged.
C
C                 RDA   Output : double precision array RDA(M)
C                       Right-hand side of the provided reduced derivative
C                       array. The order and the number of the provided
C                       right-hand sides depends on the used formulation, see
C                       IOPT(5).
C                       If IOPT( 5)=0 the numerical integration is based on the
C                         projected-strangeness-free formulation, i.e., the
C                         user  has to provide the equations (1)-(7) together
C                         with the first and second time derivative of the
C                         holonomic constraints, i.e.,
C                           gI(p,v,t)           =d/dt g(p,t),
C                           gII(p,v,r,w,s,l,m,t)=d^2/dt^2 g(p,t),
C                         and the first time derivative of the nonholonomic
C                         constraints, i.e.,
C                           hI(p,v,r,w,s,l,m,t) =d/dt(H(p,s,t)Z(p)v+h(p,s,t)).
C                         If there exist some solution invariants (8) the user
C                         should also provide them and set NI equal to the
C                         number of the solution invariants. The order is given
C                         by
C
C                         RDA(1:NW)                                       =d
C                         RDA(NW+1:NW+NL)                                 =gII
C                         RDA(NW+NL+1:NW+NL+NM)                           =hI
C                         -----------------------------------------------------
C                         RDA(NW+NL+NM+1:NW+NL+NM+NL)                     =gI
C                         RDA(NW+NL+NM+NL+1:NW+NL+NM+NL+NM)               =h
C                         RDA(NW+NL+NM+NL+NM+1:NW+NL+NM+NL+NM+NI)         =e
C                         -----------------------------------------------------
C                         RDA(NW+NL+NM+NL+NM+NI+1:NW+NL+NM+NL+NM+NI+NS) =c
C                         RDA(NW+NL+NM+NL+NM+NI+NS+1:...
C                                               NW+NL+NM+NL+NM+NI+NS+NL) =g
C                         -----------------------------------------------------
C                         RDA(NW+NL+NM+NL+NM+NI+NS+NL+1:...
C                                           NW+NL+NM+NL+NM+NI+NS+NL+NR) =b
C                         -----------------------------------------------------
C                         RDA(NW+NL+NM+NL+NM+NI+NS+NL+NR+1:...
C                                        NW+NL+NM+NL+NM+NI+NS+NL+NR+NV) =f_dyn
C                         -----------------------------------------------------
C                         RDA(NW+NL+NM+NL+NM+NI+NS+NL+NR+NV+1:...
C                                     NW+NL+NM+NL+NM+NI+NS+NL+NR+NV+NP) =f_kin
C
C                       If IOPT( 5)=1 the numerical integration is based on the
C                         projected-strangeness-index-1 formulation , i.e., the
C                         user has to provide the equations (1)-(7) together
C                         with the first time derivative of the holonomic
C                         constraints, i.e.,
C                           gI(p,v,t)            = d/dt g(p,t).
```

```
C                       If there exist some solution invariants (8) the user
C                       should also provide them and set NI equal to the
C                       number of the solution invariants. The order is given
C                       by
C
C                       RDA(1:NW)                                          =d
C                       ---------------------------------------------------
C                       RDA(NW+1:NW+NL)                                    =gI
C                       RDA(NW+NL+1:NW+NL+NM)                              =h
C                       RDA(NW+NL+NM+1:NW+NL+NM+NI)                        =e
C                       ---------------------------------------------------
C                       RDA(NW+NL+NM+NI+1:NW+NL+NM+NI+NS)                  =c
C                       RDA(NW+NL+NM+NI+NS+1:NW+NL+NM+NI+NS+NL)            =g
C                       ---------------------------------------------------
C                       RDA(NW+NL+NM+NI+NS+NL+1:NW+NL+NM+NI+NS+NL+NR) =b
C                       ---------------------------------------------------
C                       RDA(NW+NL+NM+NI+NS+NL+NR+1:...
C                                            NW+NL+NM+NI+NS+NL+NR+NV) =f_dyn
C                       ---------------------------------------------------
C                       RDA(NW+NL+NM+NI+NS+NL+NR+NV+1:...
C                                         NW+NL+NM+NI+NS+NL+NR+NV+NP) =f_kin
C
C
C                  IOPT  Input  : integer array IOPT(40)
C                        Serve as parameters for the code.
C                        IOPT has to remain unchanged.
C
C                  ROPT  Input  : double precision array ROPT(40)
C                        Serve as parameters for the code.
C                        ROPT has to remain unchanged.
C
C                  IPAR  Input/Output: integer array IPAR(*)
C                        Integer parameters which are only used by the user.
C                        They are unused and unchanged by GEOMS.
C
C                  RPAR  Input/Output: double precision array RPAR(*)
C                        Double precision parameters which are only used by the
C                        user. They are unused and unchanged by GEOMS.
C
C                  IERR  Output : integer
C                        Indicator of success. IERR is only used by
C                        user supplied subroutines. After every call of a user
C                        supplied  subroutine the status of IERR is checked. If
C                        IERR is negative  the run of GEOMS will be interrupted
C                        and GEOMS returns to the  calling program. IERR  is
C                        unchanged by GEOMS.
C
C     MAS       Name (EXTERNAL) of the user supplied subroutine which provides
C               the mass matrix M(p,t) in equation (2) of the EoM
C
C               SUBROUTINE MAS(T,NX,X,M,N,MA,IOPT,ROPT,IPAR,RPAR,IERR)
C               IMPLICIT NONE
C               INTEGER          NX,M,N,IOPT(*),IPAR(*),IERR
C               DOUBLE PRECISION T,X(NX),MA(M,N),ROPT(*),RPAR(*)
```

```
C
C                T     Input  : double precision
C                      Evaluation of the mass matrix MA at time T.
C                      T has to remain unchanged.
C
C                NX    Input  : integer
C                      Number of unknowns, i.e., dimension of X. We have
C                      NX=NP+NV+NR+NW+NS+NL+NM.
C                      NX has to remain unchanged.
C
C                M     Input  : integer
C                      Number of rows of the mass matrix MA. We have M=NV.
C                      M has to remain unchanged.
C
C                N     Input  : integer
C                      Number of rows of the mass matrix MA. We have N=NV.
C                      N has to remain unchanged.
C
C                X     Input  : double precision array X(NX)
C                      Vector of unknowns, see above.
C                      X has to remain unchanged.
C
C                MA    Output : double precision array MA(M,N)
C                      Mass matrix of the equations of motion. The mass matrix
C                      has to be provided as a full M x N array,
C                      also in the case of diagonal structure. Because of the
C                      used regularization technique a sparse storage is not
C                      possible and does not save time or memory.
C
C                IOPT  Input  : integer array IOPT(40)
C                      Serve as parameters for the code.
C                      IOPT has to remain unchanged.
C
C                ROPT  Input  : double precision array ROPT(40)
C                      Serve as parameters for the code.
C                      IOPT has to remain unchanged.
C
C                IPAR  Input/Output: integer array IPAR(*)
C                      Integer parameters which are only used by the user.
C                      They are unused and unchanged by GEOMS.
C
C                RPAR  Input/Output: double precision array RPAR(*)
C                      Double precision parameters which are only used by the
C                      user. They are unused and unchanged by GEOMS.
C
C                IERR  Output : integer
C                      Indicator of success. IERR is only used by
C                      user supplied subroutines. After every call of a user
C                      supplied  subroutine the status of IERR is checked. If
C                      IERR is negative  the run of GEOMS will be interrupted
C                      and GEOMS returns to the  calling program. IERR  is
C                      unchanged by GEOMS.
C
C     JAC      Name (EXTERNAL) of the user supplied subroutine which computes
```

```
C              the NEGATIVE partial derivatives of the right-hand side of the
C              equations of motion. (This routine is only called if IJAC=1.
C              Supply a dummy subroutine in the case IJAC=0).
C
C              SUBROUTINE JAC(M1,M2,M3,M4,M5,M6,N1,N2,N3,N4,M,N,
C             #               T,X,FX1,FX2,FX3,FX4,FX5,FX6,
C             #               IOPT,ROPT,RPAR,IPAR,IERR)
C              IMPLICIT NONE
C              INTEGER          M1,M2,M3,M4,M5,M6,N1,N2,N3,N4,M,N,
C             #                 IOPT(*),IPAR(*),IERR
C              DOUBLE PRECISION T,X(N),FX1,FX2,FX3,FX4,FX5,FX6,ROPT(*),RPAR(*)
C
C                 M1    Input  : integer
C                       Number of constraints depending on all unknown
C                       variables and restricting the Lagrange multipliers l
C                       and m and the auxiliary variables w, i.e., 0=d,
C                       0=gII, 0=hI. IF IOPT(5)=0 we have M1=NW+NL+NM and if
C                       IOPT(5)=1 we have M1=NW (note that M1=0 is possible).
C                       Compare with the block row structure of RDA in the
C                       subroutine EOM.
C                       M1 has to remain unchanged.
C
C                 M2    Input  : integer
C                       Number of constraints only depending on the unknown
C                       variables p, v, and s and restricting the velocity
C                       variables v, i.e., 0=gI, 0=h, 0=e. We have M2=NL+NM+NI.
C                       Compare with the block row structure of RDA in the
C                       subroutine EOM.
C                       M2 has to remain unchanged.
C
C                 M3    Input  : integer
C                       Number of constraints only depending on the unknown
C                       variables p and s and restricting the position p and
C                       the contact variables s, i.e., 0=c, 0=g. We have
C                       M3=NS+NL.
C                       Compare with the block row structure of RDA in the
C                       subroutine EOM.
C                       M3 has to remain unchanged.
C
C                 M4    Input  : integer
C                       Number of dynamical force element equations (3), i.e.,
C                       we have M4=NR.
C                       Compare with the block row structure of RDA in the
C                       subroutine EOM.
C                       M4 has to remain unchanged.
C
C                 M5    Input  : integer
C                       Number of dynamical equations of motion (2), i.e.,
C                       we have M5=NV.
C                       Compare with the block row structure of RDA in the
C                       subroutine EOM.
C                       M5 has to remain unchanged.
C
C                 M6    Input  : integer
```

```
C                       Number of kinematical equations of motion (1), i.e.,
C                       we have M6=NP.
C                       Compare with the block row structure of RDA in the
C                       subroutine EOM.
C                       M6 has to remain unchanged.
C
C               N1      Input  : integer
C                       Number of auxiliary variables plus the number of
C                       Lagrange multipliers, i.e., we have N1=NW+NL+NM.
C                       Compare with the block row structure of X above.
C                       N1 has to remain unchanged.
C
C               N2      Input  : integer
C                       Number of dynamical force element variables, i.e.,
C                       we have N2=NR.
C                       Compare with the block row structure of X above.
C                       N2 has to remain unchanged.
C
C               N3      Input  : integer
C                       Number of velocity variables, i.e., we have N3=NV.
C                       Compare with the block row structure of X above.
C                       N3 has to remain unchanged.
C
C               N4      Input  : integer
C                       Number of contact point variables plus the number of
C                       position variables, i.e., we have N1=NS+NP.
C                       Compare with the block row structure of X above.
C                       N4 has to remain unchanged.
C
C               M       Input  : integer
C                       Total number of provided equations,
C                       i.e., dimension of RDA, see subroutine EOM and the
C                       number of rows of the partial derivatives. We have
C                       M=M1+M2+M3+M4+M5+M6.
C                       M has to remain unchanged.
C
C               N       Input  : integer
C                       Number of unknowns, i.e., dimension of X.
C                       We have N=NP+NV+NR+NW+NS+NL+NM=N1+N2+N3+N4.
C                       N has to remain unchanged.
C
C               T       Input  : double precision
C                       Evaluation of the partial derivatives at time T.
C                       T has to remain unchanged.
C
C               X       Input  : double precision array X(NX)
C                       Vector of unknowns, see above.
C                       X has to remain unchanged.
C
C               FX1     Output : double precision array FX1(M1,N)
C                       NEGATIVE partial derivatives of d, gII, hI with
C                       respect to [w l m | r | v | s p ]. We have
C                             [ d d  /d[w l m | r | v | s p ] ]
C                       FX1=[ d gII/d[w l m | r | v | s p ] ] in R^(M1,N)
```

```
C                       [ d hI /d[w l m | r | v | s p ] ]
C                 Compare with the block row structure of RDA in the
C                 subroutine EOM and with the block row structure of X
C                 above.
C
C           FX2   Output : double precision array FX2(M2,N3+N4)
C                 NEGATIVE partial derivatives of gI, h~, e with
C                 respect to [v s p]. We have
C                       [ d gI/d[v s p] ]
C                   FX2=[ d h~/d[v s p] ] in R^(M2,N3+N4)
C                       [ d e /d[v s p] ]
C                 Compare with the block row structure of RDA in the
C                 subroutine EOM and with the block row structure of X
C                 above.
C
C           FX3   Output : double precision array FX3(M3,N4)
C                 NEGATIVE partial derivatives of c and g with
C                 respect to [s p]. We have
C                       [ d c/d[s p] ]
C                   FX3=[            ] in R^(M3,N4)
C                       [ d g/d[s p] ]
C                 Compare with the block row structure of RDA in the
C                 subroutine EOM and with the block row structure of X
C                 above.
C
C           FX4   Output : double precision array FX4(M4,N)
C                 NEGATIVE partial derivatives of the right-hand side of
C                 the dynamical force element equations, i.e., of b with
C                 respect to [w l m | r | v | s p ]. We have
C                   FX4=d b/d[w l m | r | v | s p ] in R^(M4,N)
C                 Compare with the block row structure of RDA in the
C                 subroutine EOM and with the block row structure of X
C                 above.
C
C           FX5   Output : double precision array FX5(M5,N)
C                 NEGATIVE partial derivatives of the right-hand side of
C                 the dynamical equations of motion, i.e., of f_dyn with
C                 respect to [w l m | r | v | s p ]. We have
C                   FX5=d f_dyn/d[w l m | r | v | s p ] in R^(M5,N)
C                 Compare with the block row structure of RDA in the
C                 subroutine EOM and with the block row structure of X
C                 above.C
C
C           FX6   Output : double precision array FX6(M6,N3+N4)
C                 NEGATIVE partial derivatives of the right-hand side of
C                 the kinematical equations of motion, i.e., of f_kin
C                 with respect to [v | s p ]. We have
C                   FX6=d f_kin/d[ v | s p ] in R^(M6,N3+N4)
C                 Compare with the block row structure of RDA in the
C                 subroutine EOM and with the block row structure of X
C                 above.
C
C           IOPT  Input  : integer array IOPT(40)
C                 Serve as parameters for the code.
```

```
C                     IOPT has to remain unchanged.
C
C               ROPT  Input  : double precision array ROPT(40)
C                     Serve as parameters for the code.
C                     IOPT has to remain unchanged.
C
C               IPAR  Input/Output: integer array IPAR(*)
C                     Integer parameters which are only used by the user.
C                     They are unused and unchanged by GEOMS.
C
C               RPAR  Input/Output: double precision array RPAR(*)
C                     Double precision parameters which are only used by the
C                     user. They are unused and unchanged by GEOMS.
C
C               IERR  Output : integer
C                     Indicator of success. IERR is only used by
C                     user supplied subroutines. After every call of a user
C                     supplied  subroutine the status of IERR is checked. If
C                     IERR is negative  the run of GEOMS will be interrupted
C                     and GEOMS returns to the  calling program. IERR  is
C                     unchanged by GEOMS.
C
C     IJAC    Input  : integer
C             Switch for the computation of the partial derivatives of the
C             right-hand side of the equations of motion
C             IJAC=0 Partial derivatives are computed internally by finite
c                    differences, subroutine JAC is never called.
C             IJAC=1 Partial derivatives are supplied by subroutine JAC.
C
C     SOLOUT  Name (EXTERNAL) of subroutine providing the numerical solution
C             during integration.
C             If IOUT=1, it is called after every successful step. Supply a
C             dummy subroutine if IOUT=0.
C             SOLOUT furnishes the solution X at the nr-th grid-point T
C             (Thereby the initial value is the first grid-point).
C
C             SUBROUTINE SOLOUT(NACCPT,TOLD,T,X,N,NN2,NN3,NN4,CONTX,H,C1M1,
C            #                   C2M1,RPAR,IPAR,IERR)
C             IMPLICIT NONE
C             INTEGER          NACCPT,N,NN2,NN3,NN4,IPAR(*),IERR
C             DOUBLE PRECISION TOLD,T,H,X(N),CONTX(NN4),RPAR(*),C1M1,C2M1
C             DOUBLE PRECISION GEDENSOUT
C             EXTERNAL         GEDENSOUT
C
C               NACCPT Input  : integer
C                     Number of accepted steps so far.
C                     NACCPT has to remain unchanged.
C
C               TOLD  Input  : double precision
C                     The preceeding grid-point.
C                     TOLD has to remain unchanged.
C
C               T     Input  : double precision
C                     Current simulation time T.
```

```
C                       T has to remain unchanged.
C
C                 X     Input  : double precision array X(NX)
C                       Vector of unknowns, see above.
C                       X has to remain unchanged.
C
C                 N     Input  : integer
C                       Number of unknowns, i.e., dimension of X.
C                       We have N=NP+NV+NR+NW+NS+NL+NM=N1+N2+N3+N4.
C                       N has to remain unchanged.
C
C                 NN2,NN3,NN4,CONTX,H,C1M1,C2M1 Input: integer/double precision
C                       Internal communication for the use by the subroutine
C                       GEDENSOUT for dense output.
C                       NN2,NN3,NN4,CONTX,H,C1M1,C2M1 have to remain unchanged.
C
C                 IPAR  Input/Output: integer array IPAR(*)
C                       Integer parameters which are only used by the user.
C                       They are unused and unchanged by GEOMS.
C
C                 RPAR  Input/Output: double precision array RPAR(*)
C                       Double precision parameters which are only used by the
C                       user. They are unused and unchanged by GEOMS.
C
C                 IERR  Output : integer
C                       Indicator of success. IERR is only used by
C                       user supplied subroutines. After every call of a user
C                       supplied  subroutine the status of IERR is checked. If
C                       IERR is negative  the run of GEOMS will be interrupted
C                       and GEOMS returns to the  calling program. IERR  is
C                       unchanged by GEOMS.
C
C                 -----  Continuous output -----
C                       During calls to "SOLOUT", a continuous solution
C                       for the interval [TOLD,T] is available through
C                       the function
C                           GEDENSOUT(I,TOUT,N,NN2,NN3,NN4,T,H,CONTX,C1M1,C2M1)
C                       which provides an approximation to the I-th
C                       component of the solution at the point TOUT, e.g.,
C                          DO I=1,N
C                          XOUT(I)=GEDENSOUT(I,TOUT,N,NN2,NN3,NN4,T,H,CONTX,
C                         #                  C1M1,C2M1)
C                          END DO
C                       The value TOUT should lie in the interval [TOLD,T].
C                       Do not change the entries of N, NN2, NN3, NN4, T, H,
C                       CONTX, C1M1, C2M1.
C                       The function GEDENSOUT is adopted from the code RADAU5,
C                       see the book:
C                       E. Hairer and G. Wanner, Solving Ordinary Differential
C                       Equations II. Stiff and Differential-Algebraic Problems
C                       Springer Series in Computational Mathematics 14,
C                       Springer-Verlag 1991, Second edition 1996.
C                       The former name was CONTR5.
C
```

```
C     IOUT    Input  : integer
C             Switch for the calling of subroutine SOLOUT.
C             IOUT=0 Subroutine is never called.
C             IOUT=1 Subroutine is available for output.
C
C     LIWORK  Input  : integer
C             Declares the length of the array IWORK. LIWORK has to be at least
C             20.
C
C     IWORK   Output: integer array IWORK(LIWORK)
C             Statistical information
C             IWORK( 1) NACCPT - Number of accepted integration steps
C             IWORK( 2) NEOM   - Number of evaluations of the right-hand side
C                                of the equations of motion
C             IWORK( 3) NMAS   - Number of evaluations of the mass matrix
C             IWORK( 4) NJAC   - Number of evaluations of the Jacobian of the
C                                right-hand side of the equations of motion
C             IWORK( 5) NSEL   - Number of determinations of suitable selectors
C             IWORK( 6) NPDEC  - Number of predecompositions, i.e., of FX, M,
C                                and IKIN
C             IWORK( 7) NEDEC  - Number of E-decompositions, i.e., of E1 and E2
C             IWORK( 8) NBSUB  - Number of backward substitutions
C             IWORK( 9) NSTEP  - Number of steps
C             IWORK(10) NERJCT - Number of rejections caused by error test
C                                failures
C             IWORK(11) NCRJCT - Number of rejections caused by convergence
C                                problems of the Newton process
C
C     LRWORK  Input  : integer
C             Declares the length of the array RWORK.
C             A safe choice for all possible setting in IOPT is
C               LRWORK at least 5*N
C             Depending on IOPT it is sufficient ...
C               If IOPT(17)=IVCNSST=0 then LRWORK has to be at least 5*N
C               If IOPT(11)=DECOMPC=3 then LRWORK has to be at least
C                 5*MAX(M1,M2,M3,N1,N3,N4), see comments to subroutine JAC.
C               If IOPT(11)=DECOMPC=2 then LRWORK has to be at least N
C               If IOPT(12)=DECOMPD=1 then LRWORK has to be at least 2*N
C             For good performance, LRWORK should generally be larger.
C
C     RWORK   Intern : integer array IWORK(LIWORK)
C
C     IPAR    Input/Output  : integer array IPAR(*)
C             Integer parameters which are only used by the user. They are
C             unused and unchanged by GEOMS.
C
C     RPAR    Input/Output: double precision array RPAR(*)
C             Double precision parameters which are only used by the user.
C             RPAR is unused and unchanged by GEOMS.
C
C     IERR    Input/Output : integer
C             Indicator of success. IERR is only used by user
C             supplied subroutines. After every call of a user supplied
C             subroutine the status of IERR is checked. If IERR is negative
```

```
C             the run of GEOMS will be interrupted and GEOMS returns to the
C             calling program. IERR  is unchanged by GEOMS.
C
C     IDID    Output : integer
C             Reports success upon return. The first two digits
C             indicate the subroutine which causes trouble.
C
C       IDID=-10.. An error occurred in the subroutine GEOMS
C            -1001 Option array IOPT or ROPT or tolerances RTOL or ATOL
C                  contains wrong data
C                  Check the output in UNIT=IOPT(2) for more information
C                  If the option IOPT(2) equals 0 turn on the output.
C            -1002 Initial IDID lower than 0
C
C       IDID=-11.. An error occurred in the subroutine GECOR
C            -1101 Stop initialized by SOLOUT
C            -1102 Stop initialized by EOM
C            -1103 Stop initialized by MAS
C            -1104 Stop initialized by JAC
C            -1105 Initial conditions not consistent
C            -1106 Final time TEND before initial time T
C            -1111 QR-Decomposition of FX1 not possible
C            -1112 QR-Decomposition of FX2 not possible
C            -1113 QR-Decomposition of FX3 not possible
C            -1114 QR-Decomposition of E1 or E2 not possible
C            -1115 Newton method repeatedly does not converge NSING.GE.5
C            -1116 Newton method repeatedly does not converge NSING.GE.5
C            -1117 More than NMAX steps are needed
C            -1118 Step size too small
C            -1128 An error occurred during use of DORMQR
C            -1129 An error occurred during use of DORMQR
C
C       IDID=-12.. An error occurred in the subroutine GEFXNUM
C            -1201 Stop initialized by EOM
C
C       IDID=-14.. An error occurred in the subroutine GEDECCQR
C            -1401 Constraints redundant or dd/dw singular
C                  (FX1 rank deficient).
C                  Try the integration again with IOPT(11)=3 (SVD).
C            -1402 Constraints or the invariant equations are redundant
C                  (FX2 rank deficient). Try the integration again with
C                  IOPT(11)=3 (SVD).
C            -1403 Constraints redundant or dc/ds singular
C                  (FX3 rank deficient).
C                  Try the integration again with IOPT(11)=3 (SVD).
C
C       IDID=-18.. An error occurred in the subroutine GETRFRHSC
C            -1801 Multiplication with Q1 not possible
C            -1802 Multiplication with Q2 not possible
C            -1803 Multiplication with Q3 not possible
C            -1804 Multiplication with Q4 not possible
C
C       IDID=-20.. An error occurred in the subroutine GEERREST
C            -2004 Multiplication with Q4 not possible
```

```
C
C       IDID=-21.. An error occurred in the subroutine GEINIVAL.
C            -2101 Stop initialized by EOM.
C            -2102 Stop initialized by IVCOND.
C            -2103 An error occurred during SVD.
C            -2104 Divergence during determination of consistent initial
C                  values. The given conditions in IVCOND together with
C                  all constraints of the EoM form an overdetermined system.
C                  Perhaps it is contradictory.
C                  => Check consistency of all constraints of the EoM in
C                     relation to the conditions given in IVCOND!
C                  => If you are sure that the initial values are consistent
C                     (at least variables P and V) you can set IOPT(17)=1.
C            -2105 No Convergence in the given limit of iterations.
C                  (See the source code of GEINIVAL and increase NIT or/and
C                  NNWTUPD.
C            -2106 Given conditions in IVCOND together with constraints in
C                  EoM are not sufficient to uniquely determine consistent
C                  initial values. Perhaps there are not enough conditions
C                  or they are redundant.
C                  => Provide more (nonredundant) conditions in IVCOND!
C                  => Check NIVCOND!
C                  => Check redundancy of all constraints of the EoM in
C                     relation to the conditions given in IVCOND!
C                  => If you are sure that the initial values are consistent
C                     (at least variables P and V) you can set IOPT(17)=1.
C
C       IDID=-24.. An error occurred in the subroutine GEDECCSV
C            -2401 Constraints are not uniformly redundant, i.e., rank of FX1
C                  was changing
C            -2402 Constraints are not uniformly redundant, i.e., rank
C                  deficiency of FX1 not identical to rank deficiency of FX2
C            -2403 Constraints are not uniformly redundant, i.e., rank
C                  deficiency of FX1 not identical to rank deficiency of FX3
C            -2404 An error occurred during SVD of FX1 or FX2 or FX3
C
C       IDID=-26.. An error occurred in the subroutine GEDECCLU
C            -2601 Constraints redundant or dd/dw singular
C                  (FX1 rank deficient).
C                  Try the integration again with IOPT(11)=3 (SVD).
C            -2602 Constraints or the invariant equations are redundant
C                  (FX2 rank deficient). Try the integration again with
C                  IOPT(11)=3 (SVD).
C            -2603 Constraints redundant or dc/ds singular
C                  (FX3 rank deficient).
C                  Try the integration again with IOPT(11)=3 (SVD).
C
C -------------------------------------------------------------------------
```

References

[1] F.M.L. Amirouche. *Computional Methods in Multibody Dynamics.* Prentice Hall, Englewood Cliffs, New Jersey 07632, Chicago, 1992.

[2] E. Anderson and et.al. *LAPACK Users' Guide.* SIAM, 3rd edition, 1999.

[3] M. Arnold. Half-explicit Runge-Kutta methods with explicit stages for differential-algebraic systems of index 2. *BIT*, 38(3):415–438, 1998.

[4] K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential Algebraic Equations*, volume 14 of *Classics in Applied Mathematics.* SIAM, Philadelphia, PA, 1996.

[5] J.J.B. de Swart and G. Söderlind. On the construction of error estimators for implicit Runge-Kutta methods. *Journal of Computional and Applied Mathematics*, 86:347–358, 1998.

[6] P. Deuflhard. *Newton methods for nonlinear problems. Affine invariance and adaptive algorithms*, volume 35 of *Springer Series in Computational Mathematics.* Springer-Verlag, Berlin, 2004.

[7] E. Eich-Soellner and C. Führer. *Numerical Methods in Multibody Dynamics.* B.G.Teubner, Stuttgart, 1998.

[8] A. Eichberger. *Simulation von Mehrkörpersystemen auf parallelen Rechnerarchitekturen.* Number 332 in Fortschritt-Berichte VDI, Reihe 8: Meß-, Steuerungs- und Regelungstechnik. VDI-Verlag Düsseldorf, 1993.

[9] C. Führer. *Differential-algebraische Gleichungssysteme in mechanischen Mehrkörpersystemen - Theorie, numerische Ansätze und Anwendungen.* PhD thesis, Technische Universität München, 1988.

[10] C. Führer and B.J. Leimkuhler. Numerical solution of differential-algebraic equations for constrained mechanical motion. *Numerische Mathematik*, 59:55–69, 1991.

[11] C.W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall Inc. Englewood Cliffs, New Jersey, 1971.

[12] C.W. Gear. Differential-algebraic equation index transformations. *SIAM Journal on Scientific and Statistic Computing*, 9:39–47, 1988.

[13] C.W. Gear, B. Leimkuhler, and G.K. Gupta. Automatic integration of Euler-Lagrange equations with constraints. *Journal of Computional and Applied Mathematics*, 12/13:77–90, 1985.

[14] C.W. Gear and L.R. Petzold. ODE methods for the solution of differential/algebraic systems. *SIAM Journal on Numerical Analysis*, 21:716–728, 1984.

[15] E. Griepentrog and R. März. *Differential-Algebraic Equations and Their Numerical Treatment*, volume 88 of *Teubner-Texte zur Mathematik*. BSB B.G.Teubner Verlagsgesellschaft, Leipzig, 1986.

[16] K. Gustafsson. Control-theoretic techniques for step size selection in implicit Runge-Kutta methods. *Association for Computing Machinery. Transactions on Mathematical Software*, 20:496–517, 1994.

[17] E. Hairer, C. Lubich, and M. Roche. *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*. Springer-Verlag, Berlin, Germany, 1989.

[18] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II - Stiff and Differential-Algebraic Problems*. Springer-Verlag, Berlin, Germany, 2nd edition, 1996.

[19] E.J. Haug. *Computer Aided Kinematics and Dynamics of Mechanical Systems*, volume 1: Basic Methods. Allyn & Bacon, Boston, 1989.

[20] D.J. Higham and N.J. Higham. *MATLAB Guide*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2005.

[21] P. Kunkel and V. Mehrmann. Regular solutions of nonlinear differential-algebraic equations and their numerical determination. *Numerische Mathematik*, 79:581–600, 1998.

[22] P. Kunkel and V. Mehrmann. Analysis of over- and underdetermined nonlinear differential-algebraic systems with application to nonlinear control problems. *Mathematics of Control, Signals and Systems*, 14:233–256, 2001.

[23] P. Kunkel and V. Mehrmann. Index reduction for differential-algebraic equations by minimal extension. *ZAMM. Zeitschrift für Angewandte Mathematik und Mechanik. Journal of Applied Mathematics and Mechanics*, 84(9):579–597, 2004.

[24] P. Kunkel and V. Mehrmann. *Differential-Algebraic Equations. Analysis and Numerical Solution*. EMS Publishing House, Zürich, Switzerland, 2006.

[25] C.L. Lawson, R.J. Hanson, D.R. Kincaid, and F.T. Krogh. Basic Linear Algebra Subprograms for Fortran usage. *ACM Transactions on Mathematical Software*, 5(3):308–323, September 1979.

[26] P. Lötstedt. Mechanical systems of rigid bodies subject to unilateral constraints. *SIAM Journal on Applied Mathematics*, 42:281–296, 1982.

[27] C. Lubich. Integration of stiff mechanical systems by Runge-Kutta methods. *Zeitschrift für Angewandte Mathematik und Physik*, 44:1022–1053, 1993.

[28] C. Lubich, U. Nowak, U. Pöhle, and C. Engstler. MEXX – numerical software for the integration of constrained mechanical multibody systems. Preprint SC 92-12, Konrad-Zuse-Zentrum für Informationstechnik Berlin, dec 1992.

[29] L.R. Petzold. Differential/algebraic equations are not ODEs. *SIAM Journal on Scientific and Statistic Computing*, 3:367–384, 1982.

[30] L.R. Petzold. Order results for implicit Runge-Kutta methods applied to differential/algebraic systems. *SIAM Journal on Numerical Analysis*, pages 837–852, 1986.

[31] L.R. Petzold. A description of DASSL: A differential/algebraic system solver. In R.S. Stepleman and et al., editors, *Scientific Computing*, pages 65–68. North Holland, Amsterdam, 1983.

[32] L.R. Petzold and P. Lötstedt. Numerical solution of nonlinear differential equations with algebraic constraints ii: Practical implications. *SIAM Journal on Scientific and Statistic Computing*, 7(3):720–733, 1986.

[33] R.E. Roberson and R. Schwertassek. *Dynamics of multibody systems*. Springer-Verlag, Berlin, Germany, 1988.

[34] W. Rulka. Effiziente Simulation der Dynamik mechatronischer Systeme für industrielle Anwendungen. Technical Report IB 532–01–06, DLR German Aerospace Center, Institute of Aeroelasticity, Vehicle System Dynamics Group, 2001.

[35] W.O. Schiehlen. *Multibody System Handbook*. Springer-Verlag, Berlin, Germany, 1990.

[36] B. Simeon, F. Grupp, C. Führer, and P. Rentrop. A nonlinear truck model and its treatment as a multibody system. *Journal of Computional and Applied Mathematics*, 50:523–532, 1994.

[37] G. Söderlind. The automatic control of numerical integration. *CWI Quarterly*, 11(1):55–74, 1998.

[38] A. Steinbrecher. *Numerical Solution of Quasi-Linear Differential-Algebraic Equations and Industrial Simulation of Multibody Systems*. PhD thesis, Technische Universität Berlin, 2006.