Efficient corrector iteration for DAE time integration in multibody dynamics

M. Arnold [a,1], A. Fuchs [b], C. Führer [c]

[a] *Martin–Luther–University Halle–Wittenberg, Department of Mathematics and Computer Science, Institute of Numerical Mathematics, D - 06099 Halle (Saale), Germany*

[b] *DLR German Aerospace Center, Institute of Robotics and Mechatronics, P.O. Box 1116, D - 82230 Wessling, Germany*

[c] *Lund University, Numerical Analysis, Center for Mathematical Sciences, P.O. Box 118, SE - 221 00 Lund, Sweden*

Abstract

Efficient time integration is a key issue in computational multibody dynamics. Implicit time integration methods for stiff systems and constrained systems require the solution of a system of nonlinear equations in each time step. The nonlinear equations are solved iteratively by Newton type methods that are tailored to the structure of the equations of motion in multibody dynamics. In the present paper we discuss classical and recent methods for reducing the numerical effort in the application to multibody systems that are modelled in joint coordinates. The methods have been implemented in an industrial multibody system simulation package. Results of numerical tests for two benchmark problems from vehicle dynamics are presented.

Key words: Multibody dynamics, DAE time integration, Corrector iteration

1 Introduction

Complex mechanical systems with components that undergo large motions may be studied efficiently by the methods and software tools of multibody dynamics [1]. Furthermore, multibody system models provide an integration

URLs: `http://www.mathematik.uni-halle.de/~arnold/` (M. Arnold), `http://www.maths.lth.se/na/staff/claus/` (C. Führer).

[1] Corresponding author.

platform for the multi-domain analysis of mechatronic systems in robotics and vehicle system dynamics [2,3]. The backbone of multibody system simulation packages are specialized numerical solution methods that are tailored to the structure of the nonlinear equations of motion [4].

In the present paper we study the time integration of multibody systems that are described by joint coordinates. Each component of the vector of position coordinates $p(t)$ corresponds to one degree of freedom of a joint in the multibody system. For tree structured multibody systems these joint coordinates form a minimal set of coordinates. The equations of motion are [1]

$$M(p, u(t))\, \ddot{p}(t) = \Psi(p, \dot{p}, u(t)) \tag{1}$$

with the time dependent external excitations $u(t)$, the symmetric, positive definite mass matrix $M(p, u(t))$ containing mass and inertia properties of all bodies and the vector of applied and gyroscopic forces $\Psi(p, \dot{p}, u(t))$.

Exploiting the topology of the multibody system the right hand side of (1) may be evaluated with a complexity that grows linearly with the number N of bodies (*multibody formalisms*) [1,5]. *Explicit formalisms*, see, e. g., [5,6], evaluate for given t and for given vectors p, v the expression

$$\varphi(p, v, u(t)) := M^{-1}(p, u(t))\, \Psi(p, \dot{p}, u(t)) \tag{2}$$

with $\mathcal{O}(N)$ complexity. *Residual formalisms* [7] evaluate for given t and given vectors p, v, a the residual

$$\Phi(p, v, a, u(t)) := M(p, u(t))\, a - \Psi(p, \dot{p}, u(t)) \tag{3}$$

with $\mathcal{O}(N)$ complexity.

In industrial applications the numerical effort of the time integration methods is often dominated by the computing time for the evaluations of φ and Φ in (2) and (3). Each evaluation of φ or Φ involves the full multibody formalism and the evaluation of all force elements in the multibody system that define the actual function value of the force vector $\Psi(p, \dot{p}, u(t))$.

If the multibody system has kinematically closed loops then the joint coordinates $p(t)$ are not longer independent of each other. Loop closing joints restrict the configuration of the system to joint coordinates $p(t)$ satisfying n_g *constraints*

$$0 = g(p, u(t))\,. \tag{4}$$

In the equations of motion the constraints (4) are coupled to the dynamical equations (1) by constraint forces $-G^\top(p, u(t))\, \lambda$ with the constraint matrix $G(p, u(t)) := (\partial g/\partial p)(p, u(t))$ and Lagrange multipliers $\lambda(t) \in \mathbb{R}^{n_g}$, see [5]:

$$M(p,u(t))\,\ddot{p}(t) = \Psi(p,\dot{p},u(t)) - G^{\top}(p,u(t))\,\lambda\,, \tag{5a}$$

$$0 = g(p,u(t))\,. \tag{5b}$$

Explicit and residual formalisms have been extended to the differential-algebraic equations of motion (5), see [5,7]. Typical problem dimensions in industrial applications are $p(t) \in \mathbb{R}^{n_p}$, $\lambda(t) \in \mathbb{R}^{n_g}$ with $n_p = 10 \ldots 100$, $n_g \leq 10$ for the simulation of a vehicle component and $n_p = 200 \ldots 1000$, $n_g = 10 \ldots 50$ for a detailed full vehicle model in automotive and railway engineering.

Standard time integration methods for differential-algebraic equations (DAEs) are used for the numerical solution of (5), see Section 2. The methods are implicit and require in each time step the iterative solution of a system of nonlinear equations by Newton's method (*corrector iteration*).

The corrector iteration is the most time consuming part of time integration. If joint coordinates are used then the numerical effort is typically dominated by the evaluation of Jacobian matrices of the right hand sides in (5). In Section 3 we present three novel algorithms for the efficient re-evaluation and update of Jacobian matrices during time integration. They are tailored to large scale applications and to the semi-explicit structure of the equations of motion (5).

All three algorithms have been implemented in an industrial multibody system tool. In Section 4 we report on practical experience in test problems from railway and automotive engineering. The algorithms proved to be reliable and efficient. The overall computing time for the dynamical simulation is typically reduced by more than 50%.

2 Time integration of constrained mechanical systems

The equations of motion (5) are solved numerically by DAE time integration methods. In this section we discuss the application of the DAE integrator DASSL [8] with focus on the corrector iteration and the approximation of Jacobian matrices by standard finite differences.

2.1 DAE time integration methods for constrained mechanical systems

In DAE terminology the equations of motion (5) have index 3, see [8,9]. The constraints $0 = g(p,u(t))$ in (5b) imply additional restrictions on the state variables that have to be considered in time integration for reasons of numerical stability [8,9].

Differentiating (5b) w. r. t. t we obtain the equation

$$0 = \frac{\mathrm{d}}{\mathrm{d}t} g(p(t), u(t)) = \frac{\partial g}{\partial p}(p(t), u(t)) \cdot \frac{\mathrm{d}p}{\mathrm{d}t} + \frac{\partial g}{\partial u}(p(t), u(t)) \cdot \frac{\mathrm{d}u}{\mathrm{d}t}$$

that results in a *hidden* constraint for the velocity coordinates $v(t) := \dot{p}(t)$:

$$0 = G(p, u(t))\, v + g^{(I)}(p, u(t), \dot{u}(t)) \tag{6}$$

with $g^{(I)}(p, u, \dot{u}) := (\partial g/\partial u)(p, u)\, \dot{u}$. The right hand side of (6) is evaluated by the multibody formalism, see [10].

There are various stabilization and projection methods to exploit the hidden constraint (6) for a numerically stable time integration of the equations of motion [4,9,11]. In the framework of an industrial multibody system tool the approach of Gear et al. [12] is especially useful because it may be implemented without any modifications of the multibody formalism.

In the *Gear–Gupta–Leimkuhler formulation* (or *stabilized index-2 formulation*) of the equations of motion the DAE (5) is rewritten as first order system and the constraints (5b) and (6) on position and velocity level are considered simultaneously:

$$\dot{p}(t) = v - G^{\top}(p, u(t))\, \eta\,, \tag{7a}$$

$$M(p, u(t))\, \dot{v}(t) = \Psi(p, v, u(t)) - G^{\top}(p, u(t))\, \lambda\,, \tag{7b}$$

$$0 = G(p, u(t))\, v + g^{(I)}(p, u(t), \dot{u}(t))\,, \tag{7c}$$

$$0 = g(p, u(t))\,. \tag{7d}$$

The increasing number of equations is compensated by a correction term $-G^{\top}(p, u)\, \eta$ with auxiliary variables $\eta(t) \in \mathbb{R}^{n_g}$. The correction term vanishes identically for the analytical solution ($\eta(t) \equiv 0$) and remains in the size of the user-defined error tolerances `TOL` for the numerical solution.

For the application of general purpose DAE time integration methods the Gear–Gupta–Leimkuhler formulation (7) is written in residual form

$$F(y(t), \dot{y}(t), u(t)) = 0 \tag{8}$$

with the vector of unknowns $y := (p, v, \lambda, \eta)^{\top} \in \mathbb{R}^{n_y}$ and the residual

$$F(y, \dot{y}, u(t)) := \begin{pmatrix} \dot{p} - v + G^{\top}(p, u(t))\, \eta \\ M(p, u(t))\, \dot{v} - \Psi(p, v, u(t)) + G^{\top}(p, u(t))\, \lambda \\ G(p, u(t))\, v + g^{(I)}(p, u(t), \dot{u}(t)) \\ g(p, u(t)) \end{pmatrix} \tag{9}$$

that may be evaluated with $\mathcal{O}(N)$ complexity by residual formalisms [7], see also (3).

For explicit multibody formalisms [5,6,10] the residual F in (8) has the form

$$F(y, \dot{y}, u(t)) := \begin{pmatrix} \dot{p} - v + G^\top(p, u(t))\, \eta \\ \dot{v} - (M(p, u(t)))^{-1}\Big(\Psi(p, v, u(t)) - G^\top(p, u(t))\, \lambda\Big) \\ G(p, u(t))\, v + g^{(I)}(p, u(t), \dot{u}(t)) \\ g(p, u(t)) \end{pmatrix}. \quad (10)$$

One of the simplest methods for the time integration of general DAEs in residual form (8) is the backward Euler method

$$F(y_{n+1}, \frac{y_{n+1} - y_n}{h_n}, u(t_{n+1})) = 0 \quad (11)$$

that defines for a given $y_n \approx y(t_n)$ the numerical solution $y_{n+1} \approx y(t_{n+1})$ in time step $t_n \to t_{n+1}$ as solution of a system of n_y nonlinear equations. In (11) the time stepsize is denoted by $h_n := t_{n+1} - t_n$.

In practical applications DAE (8) is solved by k-step backward differentiation formulae (BDF, also: Gear's method) that generalize the first order backward Euler method (11) to higher order, see [8]:

$$F(y_{n+1}, \frac{1}{h_n} \sum_{j=0}^{k_n} \alpha_{n,j} y_{n+1-j}, u(t_{n+1})) = 0\,. \quad (12)$$

The parameter k_n defines the order of the method and the BDF coefficients $\alpha_{n,j}$, ($1 \le k_n \le 6$; $j = 0, 1, \ldots, k_n$).

In (12) the numerical solution $y_{n+1} \approx y(t_{n+1})$ depends on the k_n previous values y_{n+1-j}, ($j = 1, \ldots, k_n$) and has to be computed as solution y of the system of n_y nonlinear equations

$$F(y, \alpha y + \beta, u(t_{n+1})) = 0 \quad (13)$$

with $\alpha := \alpha_{n,0}/h_n$ and the vector

$$\beta := \frac{1}{h_n} \sum_{j=1}^{k_n} \alpha_{n,j} y_{n+1-j}$$

that is defined by the k_n previous solution vectors $y_{n-(k_n-1)}, \ldots, y_n$. In the special case $k_n = 1$ we have the backward Euler method (11) with $\alpha = 1/h_n$ and $\beta = -y_n/h_n$.

The BDF integrator DASSL [8] is one of the most frequently used DAE integrators in technical simulation. It is free software that can be downloaded at `http://www.netlib.org/ode`.

DASSL implements BDF (12) with order and stepsize control. The systems of nonlinear equations (13) are solved iteratively starting with a *predictor* $y_{n+1}^{(0)}$ that is obtained by polynomial extrapolation of $y_{n-(k_n-1)}, \ldots, y_n$. In the *corrector iteration* a simplified Newton method is used to improve the numerical solution $y_{n+1}^{(m)}$ iteratively:

$$y_{n+1}^{(m+1)} = y_{n+1}^{(m)} - \bar{J}^{-1} F(y_{n+1}^{(m)}, \alpha y_{n+1}^{(m)} + \beta, u(t_{n+1}))\,, \quad (\, m \geq 0\,)\,, \tag{14}$$

see [8,13]. The matrix $\bar{J}$ approximates the Jacobian

$$J(t,y) := \frac{\mathrm{d}}{\mathrm{d}y} F(y, \alpha y + \beta, u(t)) = \alpha \frac{\partial F}{\partial \dot{y}}(y, \alpha y + \beta, u(t)) + \frac{\partial F}{\partial y}(y, \alpha y + \beta, u(t)) \tag{15}$$

of the nonlinear system (13).

In DASSL, the approximation $\bar{J}$ of the Jacobian J is selected carefully because of its dominating influence on the convergence speed of the corrector iteration (14). In the application to multibody systems that are modelled in joint coordinates the computation of $\bar{J}$ is often the most time consuming part of time integration.

2.2 Jacobian evaluation by finite differences

If the Jacobian $J(t,y)$ in (15) is not supplied by the user then DASSL applies a standard finite difference approximation $\bar{J} = (\bar{j}_{ir})$ to approximate J columnwise by difference quotients [13]. This classical approach requires n_y+1 function evaluations of F: the nominal value $F^0 := F(y, \alpha y + \beta, u(t))$ and n_y perturbed function values

$$F^r := F(y + \delta_r e_r, \alpha(y + \delta_r e_r) + \beta, u(t))\,, \quad (\, r = 1, \ldots, n_y\,)$$

with the rth unit vector e_r and small increments $\delta_r \in \mathbb{R} \setminus \{0\}$. The difference approximation $\bar{J}(t,y)$ of $J(t,y)$ is given by

$$\bar{j}_{ir} := \frac{F_i^r - F_i^0}{\delta_r}\,, \quad (\, i, r = 1, \ldots\, n_y\,). \tag{16}$$

The increments δ_r must be selected such that the difference between the approximated and the exact Jacobian can be neglected. This error consists of an approximation error $\mathcal{O}(|\delta_r|)$, and roundoff errors in the evaluation of F^0

and F^r that are of size $\mathcal{O}(\varepsilon/|\delta_r|)$ with the machine precision ε. A typical value of δ_r is

$$\delta_r = \max(|y_r|, \sqrt[4]{\varepsilon})\sqrt{\varepsilon}\,, \tag{17}$$

see [13].

The standard finite difference approximation (16) is robust. Its main drawback is, however, the large computational effort for n_y+1 function evaluations of F.

2.3 Efficient Jacobian updates in DASSL

Often, the difference approximations of the Jacobian cause a dominating part of the numerical effort in time integration. Therefore, the time consuming re-evaluations of $\bar{J}$ are avoided in the corrector iteration of DASSL as long as possible keeping one and the same approximation $\bar{J}$ over several time steps.

Typically, the re-use of old Jacobian approximations will slow down the convergence of the corrector iteration. This undesired effect is compensated by the modified corrector iteration

$$y_{n+1}^{(m+1)} = y_{n+1}^{(m)} - c\bar{J}^{-1}F(y_{n+1}^{(m)}, \alpha y_{n+1}^{(m)} + \beta, u(t_{n+1})), \tag{18}$$

with a scalar scaling factor c that is selected such that an optimal rate of convergence is achieved [14], see also (14).

By this implementation of Newton's method DASSL avoids frequent re-evaluations of the iteration matrix without slowing down the convergence of corrector iteration too much. A new difference approximation of J is computed only if

a) the corrector iteration fails to converge,
b) the speed of convergence is too slow, or
c) the parameter α changes drastically, see (15).

3 Adapted approximation of the Jacobian

The efficiency of time integration in dynamical simulation is improved by several methods for reducing the computational effort in the approximation of the Jacobian J. It is due to the special structure of J that a remarkable saving of computing time can be attained.

3.1 Difference approximation of sparse Jacobians

Difference approximations of sparse Jacobians require typically substantially less function evaluations than the standard finite differences (16). The structure of sparse Jacobians is exploited, e. g., by "Algorithm TOMS 618" of the **T**ransactions **o**n **M**athematical **S**oftware that has originally been developed for the difference approximation of sparse Jacobians and Hessians in optimization [15].

Algorithm TOMS 618 makes explicit use of the sparsity structure Σ of the Jacobian $J = (j_{ir})$ that has to be provided as input parameter of the algorithm:

$$\Sigma(J) := \{\, (i,r) \;:\; j_{ir} \neq 0 \,\}\,.$$

The important observation is that one perturbed function value $F^r \in \mathbb{R}^{n_y}$ in (16) gives n_y elements of the Jacobian. If the Jacobian is dense then the n_y elements $\bar{j}_{ir}$ of column r are obtained. For sparse Jacobians the increment $\delta_r e_r$ is slightly modified to approximate all non-zero elements of a whole group of columns using just one perturbed function value $F^{[k]} \in \mathbb{R}^{n_y}$, see (20) below.

To explain this procedure in more detail, let us look at the following example:

$$J = \begin{pmatrix} \star & 0 & 0 & \star & 0 & 0 \\ \star & \star & \star & 0 & 0 & 0 \\ \star & 0 & \star & 0 & 0 & \star \\ 0 & 0 & 0 & \star & \star & 0 \\ 0 & \star & \star & 0 & \star & 0 \\ \star & \star & 0 & 0 & 0 & 0 \end{pmatrix} \quad \Rightarrow \quad \underbrace{\begin{matrix} \star\ 0 \\ \star\ 0 \\ \star\ 0 \\ 0\ \star \\ 0\ \star \\ \star\ 0 \end{matrix}}_{\text{group1}} \quad \underbrace{\begin{matrix} 0\ \star\ 0 \\ \star\ 0\ 0 \\ 0\ 0\ \star \\ 0\ \star\ 0 \\ \star\ 0\ 0 \\ \star\ 0\ 0 \end{matrix}}_{\text{group2}} \quad \underbrace{\begin{matrix} 0 \\ \star \\ \star \\ 0 \\ \star \\ 0 \end{matrix}}_{\text{group3}} \tag{19}$$

The sparsity structure $\Sigma(J)$ is indicated by stars '$\star$' for the non-zero elements. With the nominal function value $F^0 = F(y, \alpha y + \beta, u(t))$ and only *one* perturbed function value

$$F^{\{2,4,6\}} = F(y+\delta_y^{\{2,4,6\}}, \alpha(y+\delta_y^{\{2,4,6\}})+\beta, u(t)) \;\text{ with }\; \delta_y^{\{2,4,6\}} := \delta_2 e_2 + \delta_4 e_4 + \delta_6 e_6$$

all three non-zero elements of the 2nd column ($j_{22} \approx (F_2^{\{2,4,6\}} - F_2^0)/\delta_2$, $j_{52} \approx (F_5^{\{2,4,6\}} - F_5^0)/\delta_2$, $j_{62} \approx (F_6^{\{2,4,6\}} - F_6^0)/\delta_2$), both non-zeros of the 4th column ($j_{14} \approx (F_1^{\{2,4,6\}} - F_1^0)/\delta_4$, $j_{44} \approx (F_4^{\{2,4,6\}} - F_4^0)/\delta_4$) and the non-zero element of the 6th column ($j_{36} \approx (F_3^{\{2,4,6\}} - F_3^0)/\delta_6$) are approximated simultaneously.

Algorithm TOMS 618 uses methods of graph theory to assign the n_y columns of J to $n_k \leq n_y$ groups such that each column belongs to one and only one group and in each group there is at most one non-zero entry per row. A partitioning of the columns of J with this property is called *consistent* with the difference approximation of J. In (19) we have $n_k = 3$ groups with columns 1 and 5 in group 1, columns 2, 4 and 6 in group 2 and column 3 in group 3.

Formally, the consistent partitioning is described by a function

$$\gamma \,:\, \{\, 1, \ldots, n_y \,\} \to \{\, 1, \ldots, n_k \,\}$$

with $\gamma(r) = k$ if the r-th column of J belongs to the k-th group. The partitioning is consistent if for all column indices $r_1, r_2 \in \{\, 1, \ldots, n_y \,\}$ with $r_1 \neq r_2$ the condition

$$(i, r_1) \in \Sigma(J),\ (i, r_2) \in \Sigma(J) \quad \Rightarrow \quad \gamma(r_1) \neq \gamma(r_2)$$

is satisfied for all row indices $i = 1, \ldots, n_y$. In (19) we have $\gamma(1) = \gamma(5) = 1$, $\gamma(2) = \gamma(4) = \gamma(6) = 2$ and $\gamma(3) = 3$.

Algorithm TOMS 618 computes a difference approximation $\bar{J}(t, y; \Sigma)$ of the Jacobian $J(t, y)$ using $n_k + 1$ function evaluations of F instead of the $n_y + 1$ function evaluations in the standard approach (16). The $n_k + 1$ function evaluations are $F^0 := F(y, \alpha y + \beta, u(t))$ and

$$F^{[k]} := F(y + \delta_y^{[k]}, \alpha(y + \delta_y^{[k]}) + \beta, u(t)) \quad \text{with} \quad \delta_y^{[k]} := \sum_{\{\, r \,:\, 1 \leq r \leq n_y,\, \gamma(r) = k \,\}} \delta_r e_r \,,$$

($k = 1, \ldots, n_k$). Here, e_r denotes again the rth unit vector and δ_r is chosen according to (17). We obtain

$$\bar{J}(t, y; \Sigma) = \left(\bar{j}_{ir}(\Sigma) \right)_{i,r=1}^{n_y} \quad \text{with} \quad \bar{j}_{ir}(\Sigma) := \begin{cases} \dfrac{F_i^{[\gamma(r)]} - F_i^0}{\delta_r} & \text{if } (i, r) \in \Sigma \,, \\ 0 & \text{if } (i, r) \notin \Sigma \,. \end{cases} \tag{20}$$

The sparsity structure $\Sigma(J)$ is completely defined by the topology of the multibody system and by the input–output relations of the force elements. In industrial multibody system simulation both the topology of the system and the input–output relations of all library elements are known. Practical experience shows, however, that it is fairly complicated to trace the non-zero elements of the Jacobian resulting from user-defined force elements.

A more fail-safe alternative is the use of estimates Ω of the sparsity pattern $\Sigma(J)$. At the beginning of time integration the Jacobian $J(t_0, y_0)$ is approximated once by standard finite differences (16) and Ω is initialized with the

sparsity structure of this difference approximation: $\Omega := \Sigma\Big(\bar{J}(t_0, y_0)\Big)$. In the following, the difference approximations of the Jacobian are computed applying Algorithm TOMS 618 with the *estimated* sparsity structure Ω, see (20):

$$J(t, y) \approx \bar{J}(t, y; \Omega)\,.$$

Force elements, that are not active from the very beginning of dynamical simulation make it necessary to update the estimated sparsity pattern Ω during time integration whenever the convergence of the corrector iteration is too slow and the difference approximation $\bar{J}(t, y; \Omega)$ itself has been updated recently.

In that case one full difference approximation $\bar{J}(t, y)$ is evaluated by standard finite differences (16) involving $n_y + 1$ function evaluations. The difference approximation $\bar{J}(t, y)$ is scanned for additional potential non-zero elements:

$$\Omega := \Omega \cup \Sigma\Big(\bar{J}(t, y)\Big)\,.$$

Afterwards, the time integration is continued with sparse difference approximations according to Algorithm TOMS 618 with the updated estimate Ω of the sparsity structure.

It is important to note that Algorithm TOMS 618 with the *exact* sparsity pattern $\Sigma(J)$ results in a difference approximation $\bar{J}(t, y; \Sigma(J))$ in (20) that is exactly equal to the standard finite difference approximation $\bar{J}(t, y)$ in (16). The application of Algorithm TOMS 618 with an *estimated* sparsity pattern Ω may, however, result in a different difference approximation $\bar{J}(t, y; \Omega)$. Practical experience has shown that the corrector iteration will nevertheless converge sufficiently fast if the estimated sparsity pattern Ω is regularly updated as described above, see also the results of numerical tests in Section 4 below.

3.2 Partitioned evaluation of Jacobians

There is an algorithmic detail of general purpose DAE integrators like DASSL that can make them substantially slower than classical integrators for ordinary differential equations (ODEs). Rewriting an ODE in residual form (8) we get

$$\dot{y}(t) = \varphi(y, u(t)) \quad \Leftrightarrow \quad F(y, \dot{y}, u(t)) = 0 \;\text{ with }\; F(y, \dot{y}, u) := \dot{y} - \varphi(y, u) \quad (21)$$

and the Jacobian J in (15) has the simpler form $J = \alpha I_{n_y} - (\partial\varphi/\partial y)$. Strong changes of α that enforce in the general DAE case an update of the full Jacobian J, see (15), may be handled much more conveniently in the ODE case as long as $\partial\varphi/\partial y$ varies only slightly:

$$J(t_{\text{new}}, y_{\text{new}}) = \alpha_{\text{new}} I_{n_y} - \frac{\partial \varphi}{\partial y}(y_{\text{new}}, u(t_{\text{new}}))\,,$$
$$\approx \alpha_{\text{new}} I_{n_y} - \frac{\partial \varphi}{\partial y}(y_{\text{old}}, u(t_{\text{old}})) = (\alpha_{\text{new}} - \alpha_{\text{old}}) I_{n_y} + J(t_{\text{old}}, y_{\text{old}}).$$

If the Jacobian $J(t_{\text{old}}, y_{\text{old}})$ at some previous time $t = t_{\text{old}}$ was approximated by $\bar{J}_{\text{old}}$ then the actual Jacobian $J(t_{\text{new}}, y_{\text{new}})$ may be approximated by

$$\bar{J}_{\text{new}} := \bar{J}_{\text{old}} + (\alpha_{\text{new}} - \alpha_{\text{old}}) I_{n_y}\,, \tag{22}$$

i. e., the Jacobian approximation $\bar{J}$ is updated by the cheap matrix addition (22) avoiding completely the time consuming evaluation of the Jacobian $\partial\varphi/\partial y$ of the right hand side φ in ODE (21). This Jacobian update strategy is implemented in all standard stiff ODE solvers, see, e. g., [9].

Generalizing (22) to DAEs (8) we get the *partitioned* Jacobian update

$$\bar{J}_{\text{new}} := \bar{J}_{\text{old}} + \alpha_{\text{new}} \frac{\partial F}{\partial \dot{y}}(y_{\text{new}}, \alpha y_{\text{new}} + \beta, u(t_{\text{new}})) - \alpha_{\text{old}} \frac{\partial F}{\partial \dot{y}}(y_{\text{old}}, \alpha y_{\text{old}} + \beta, u(t_{\text{old}})) \tag{23}$$

for DAEs. This partitioned update handles the entries $\partial F/\partial \dot{y}$ and $\partial F/\partial y$ in J separately, see (15), and avoids the time consuming re-evaluations of $\partial F/\partial y$. In the DAE case the partitioned update is attractive only, if $\partial F/\partial \dot{y}$ has a simple structure.

If the equations of motion of a constrained mechanical system are evaluated by explicit multibody formalisms, the update formula (23) simplifies to

$$\bar{J}_{\text{new}} := \bar{J}_{\text{old}} + (\alpha_{\text{new}} - \alpha_{\text{old}}) \begin{pmatrix} I_{2n_p} & 0_{2n_p \times 2n_g} \\ 0_{2n_g \times 2n_p} & 0_{2n_g \times 2n_g} \end{pmatrix}, \tag{24}$$

see (10).

For linear equations of motion the Jacobian $\partial F/\partial y$, that involves the partial derivatives of all force elements, has to be evaluated only once at the beginning of time integration. Later, the Jacobian approximation $\bar{J}$ is updated by the cheap matrix addition (24).

For nonlinear equations of motion the partitioned Jacobian evaluation (24) is attractive whenever α varies strongly and $\partial F/\partial y$ remains nearly unchanged. Strong changes of α result typically from strongly varying time stepsizes in the beginning of time integration and at discontinuities. The partitioned evaluation (24) is not applicable if $\partial F/\partial y$ changed substantially. In that case a full difference approximation of J has to be computed by the methods of Sections 2.2 and 3.1, see (16) and (20).

The partitioned Jacobian update (23) may be combined as well with the residual formalism. Here, the multibody formalism has to be modified to evaluate the mass matrix $M(p, u(t))$ as additional output parameter. For the equations of motion (9) the Jacobian update (23) has the form

$$\bar{J}_{\text{new}} := \bar{J}_{\text{old}} + \alpha_{\text{new}} \begin{pmatrix} I_{n_p} & 0 & 0 \\ 0 & M(p_{\text{new}}, u(t_{\text{new}})) & 0 \\ 0 & 0 & 0 \end{pmatrix} - \alpha_{\text{old}} \begin{pmatrix} I_{n_p} & 0 & 0 \\ 0 & M(p_{\text{old}}, u(t_{\text{old}})) & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{25}$$

3.3 Multibody system models with dominating external excitations

Recently, the partitioned evaluation (24) was extended to mildly nonlinear models with dominating external time excitations $u(t)$, see [16]. This important problem class involves many models with rheonomic joints.

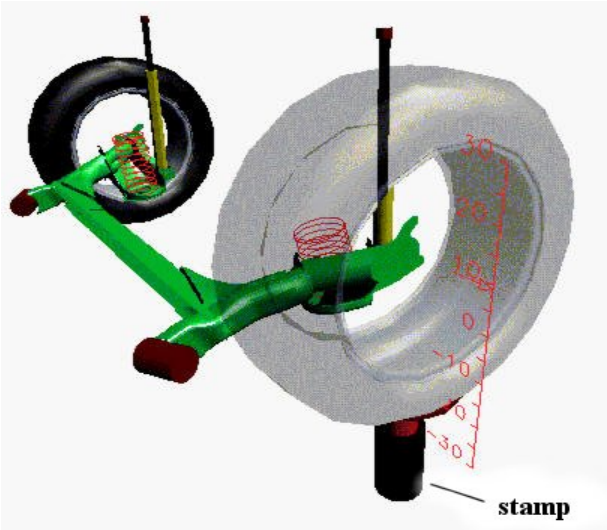

Fig. 1. Rear axle on top of a stamp [17].

A typical example is illustrated by Fig. 1 showing a test rig for rear axles. One of the wheels is on top of a stamp that moves up and down to study the frequency response of the rear axle. This model has an approximately linear behaviour. With the partitioned update formula (24) of Section 3.2 one expects only a few difference approximations (16) or (20) during time integration.

A more detailed analysis shows, however, that the partitioned evaluation (24) is not successful here since some entries of $(\partial F/\partial y)(y, \dot{y}, u(t))$ vary strongly because of the periodically moving stamp that is modelled as a rheonomic joint with periodic time excitation $u(t)$. The time dependent entries of the

Jacobian J make the time integration inefficient because periodic changes of $u(t)$ require frequent difference approximations of J. Therefore the partitioned evaluation (24) has to be extended by an additional update of time-dependent entries of $\bar{J}$.

The new Jacobian $J(t_{\text{new}}, y_{\text{new}})$ is rewritten as

$$\begin{aligned} J(t_{\text{new}}, y_{\text{new}}) &= \alpha_{\text{new}} \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} + \frac{\partial F}{\partial y}(y_{\text{new}}, \dot{y}_{\text{new}}, u(t_{\text{new}})) \\ &= J(t_{\text{old}}, y_{\text{old}}) + (\alpha_{\text{new}} - \alpha_{\text{old}}) \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} + \\ &\quad + \frac{\partial F}{\partial y}(y_{\text{new}}, \dot{y}_{\text{new}}, u(t_{\text{new}})) - \frac{\partial F}{\partial y}(y_{\text{old}}, \dot{y}_{\text{old}}, u(t_{\text{old}})). \end{aligned}$$

Taylor expansion up to some second order derivatives gives

$$\begin{aligned} \frac{\partial F}{\partial y}(y_{\text{new}}, \dot{y}_{\text{new}}, u(t_{\text{new}})) &= \frac{\partial F}{\partial y}(y_{\text{old}}, \dot{y}_{\text{old}}, u(t_{\text{old}})) + \\ &\quad + \sum_{i=1}^{n_u} \frac{\partial}{\partial u_i} \frac{\partial F}{\partial y}(y_{\text{old}}, \dot{y}_{\text{old}}, u(t_{\text{old}}))(u_i(t_{\text{new}}) - u_i(t_{\text{old}})) + R \end{aligned}$$

with a remainder term R of second order which can be neglected in the mildly nonlinear case. The new update formula for the approximation $\bar{J}$ is

$$\begin{aligned} \bar{J}_{\text{new}} := \bar{J}_{\text{old}} &+ (\alpha_{\text{new}} - \alpha_{\text{old}}) \begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} + \\ &+ \sum_{i=1}^{n_u} \frac{\partial}{\partial u_i} \frac{\partial F}{\partial y}(y_0, \dot{y}_0, u(t_0)) \cdot (u_i(t_{\text{new}}) - u_i(t_{\text{old}})). \end{aligned} \tag{26}$$

The first line in (26) is equivalent to the partitioned evaluation (24) in Section 3.2. The additional update of time dependent entries in J requires the approximation of n_u second order derivatives $\partial^2 F / \partial u_i \partial y$, $(i = 1, \ldots, n_u)$ by finite differences. These partial derivatives $\partial^2 F / \partial u_i \partial y$ are not calculated at every update of the Jacobian. It is sufficient to evaluate them once at the beginning of integration.

Note, that (26) is tailored to explicit multibody formalisms with equations of motion (10). The extension to equations of motion (9) resulting from residual formalisms would require substantial additional numerical effort for the evaluation of $\partial M / \partial u_i$.

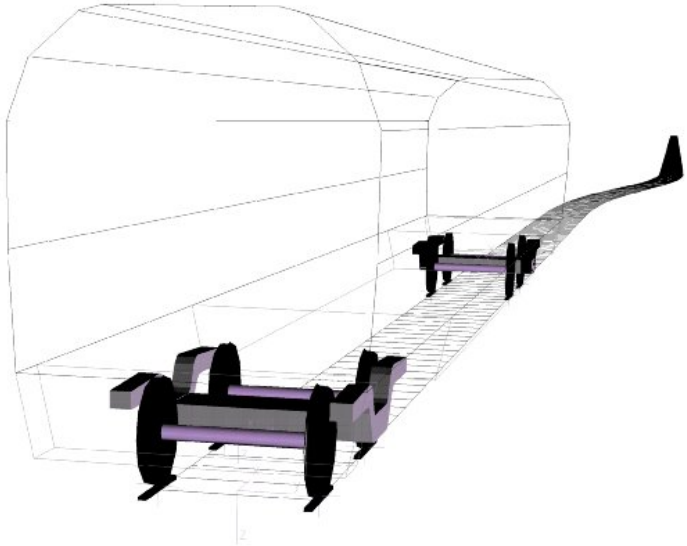

Fig. 2. Train carriage passing an S-shaped line change.

4 Numerical experiments

The industrial multibody system simulation package SIMPACK [2] offers finite difference approximations (16) and (20) for dense and sparse Jacobians, respectively, and the cheap Jacobian updates (24) and (25) for the partitioned evaluation of $\bar{J}$, see [10]. In Section 4.1 these algorithms will be compared in the application to a railway model.

In Section 4.2 we discuss the extended partitioned evaluation method (26) applied to a simplified benchmark problem and to an approximately linear automotive model in a SIMPACK developer version [16].

4.1 Dynamical simulation of a train carriage passing an S-shaped line change

Fig. 2 shows the multibody system model of a train carriage passing an S-shaped line change. It will be used as test problem throughout the present section. The model has $n_p = 41$ position coordinates describing the car body, the two bogies and the four wheelsets. A quasi-elastic model for wheel-rail contact [18] results in $n_g = 8$ constraints corresponding to eight wheels of four wheelsets in two bogies, see Fig. 2. The carriage crosses the line change with constant speed, the integration ranges from 0 s to 15 s.

In the dynamical simulation of the train carriage the numerical effort is dom-

[2] SIMPACK is a trademark of INTEC GmbH, see `http://www.simpack.com`.

inated by the evaluation of Jacobians J. Because of $2n_p + 2n_g = 98$ the Jacobian is of size 98×98. The maximum number of non-zero elements for $t \in [0\,\mathrm{s}, 15\,\mathrm{s}]$ is $n_{nz} = 1620 \ll 98^2$, the Jacobian is sparse.

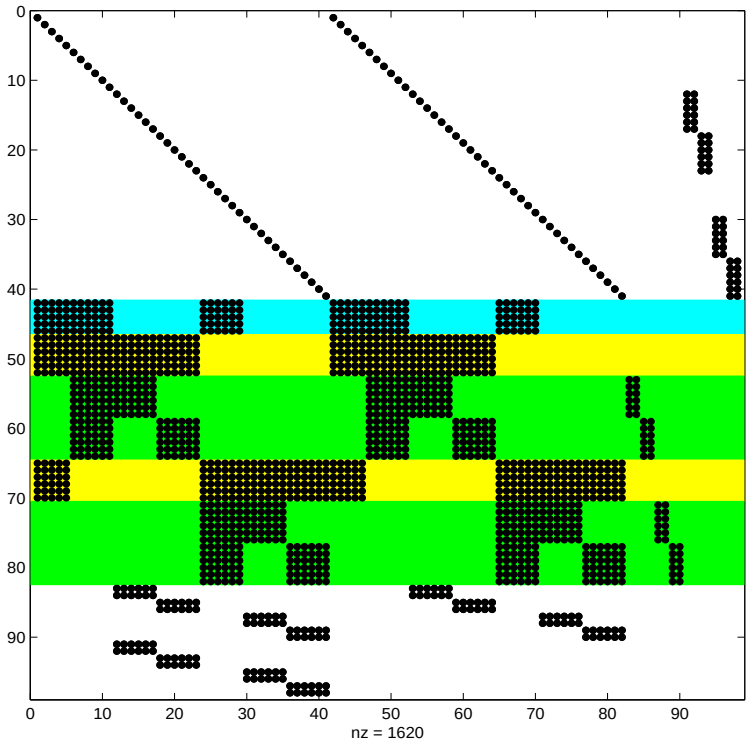

Fig. 3. Sparsity structure of J.

Fig. 3 shows the sparsity structure of J with rows 1 ... 41 corresponding to the kinematic equations (7a). Below we will discuss in more detail the sparsity structure of rows 42 ... 82 that correspond to the dynamical equations (7b). The last 16 rows result from the constraints (7c) and (7d) on velocity and position level.

The dynamic equations (7b) reflect the topology of the model. The present example has a clearly visible modular structure since there is no direct coupling between front and rear bogie. Rows 42 ... 82 of Fig. 3 illustrate this structure in more detail. There are variables describing

a) the car body (rows 42 ... 46),
b) the front bogie (rows 47 ... 52) and the rear bogie (rows 65 ... 70)
c) and the front and rear wheelsets (rows 53 ... 64 and 71 ... 82).

In rows 42 ... 46 the non-zero entries result from the dynamics of the car body itself (columns 1 ... 5 and 42 ... 46) and from the dynamical interaction of car body and bogies (columns 6 ... 11, 24 ... 29, 47 ... 52 and 65 ... 70).

There is no direct connection between car body and wheelsets. Hence, the corresponding elements of the Jacobian vanish identically (columns 12 ... 23, 30 ... 41, 53 ... 64 and 71 ... 82).

A comparison of (19) and Fig. 3 shows that this sparse Jacobian J is a perfect candidate for a groupwise finite difference approximation. A typical group is given by columns 54 and 72, i. e., by the combination of a column from a front wheelset with a column from a rear wheelset, since there is no direct dynamical interaction between these two bodies. On the other hand a column from the front bogie cannot build a group with a column from the rear bogie because both are directly connected to the car body.

The number n_k of groups in a consistent partitioning of the columns of J is minimized by Algorithm TOMS 618 applying methods from graph theory to the estimated sparsity pattern Ω, see Section 3.1. For the present example the number of groups does not exceed $n_k = 46$ during time integration. Instead of $n_y + 1 = 99$ function evaluations for the standard finite difference approximation (16) the adapted difference approximation (20) needs only $n_k + 1 = 47$ function evaluations to approximate the sparse Jacobian J.

Figs. 4 and 5 illustrate the consistent partitioning for the train carriage example. Algorithm TOMS 618 combines columns 8, 32 and 89 to group 1. Group 2 consists of columns 12, 31 and 86, group 3 is defined by column 5 and column 90, group 4 contains columns 19, 78, 83 and 87, and so on.

Substantial savings of computing time are also achieved by the partitioned Jacobian evaluations (24) and (25). Furthermore, the special structure of the Gear–Gupta–Leimkuhler formulation (7) may be exploited to avoid the difference approximation of the last $n_g = 8$ columns of J that correspond to the correction term $-G^\top \eta$, see [11,19,20]. Therefore the standard finite difference approximation (16) requires only $n_y + 1 - 8 = 91$ function evaluations of F in the present example.

The results for the test problem are shown in Table 1 for the explicit formalism (10) (`explicit`, see [10]) and for the residual formalism (9) (`residual`, see [7]). In both cases the tolerances are set to $\texttt{RTOL} = 10^{-4}$ (relative error) and $\texttt{ATOL} = 10^{-4}$ (absolute error). The abbreviations used in the table are defined as follows:

```
dense     standard finite differences (16)
sparse    finite differences (20) for sparse Jacobians
          (Algorithm TOMS 618)
part.     partitioned evaluation (24) and (25)

cpu       cpu-time
#fc       number of function calls (total)
```

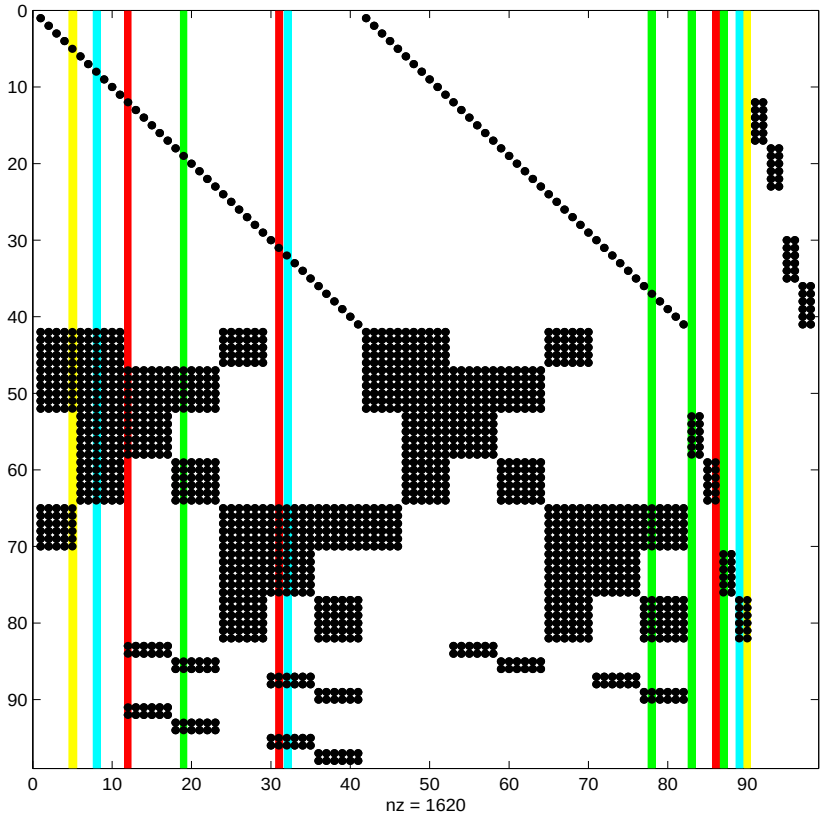

Fig. 4. Consistent partitioning of the columns of J in Algorithm TOMS 618.

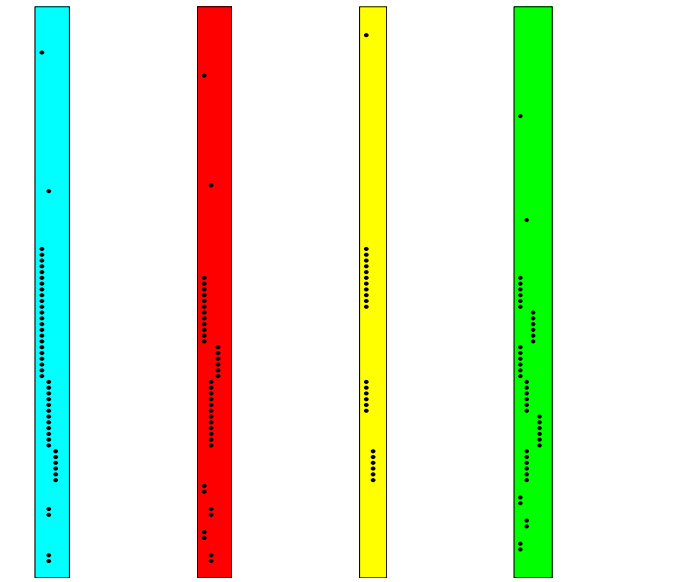

Fig. 5. Groups of the columns of Jacobian J in Algorithm TOMS 618.

	dense	sparse	part.	cpu	#fc	#fcw	#Je	percent
explicit	⋆			18.86	9739	2179	84	100.0%
		⋆		14.78	7178	2166	103	78.4%
	⋆		⋆	11.03	4759	2509	25	58.5%
		⋆	⋆	10.59	4605	2288	43	56.2%
residual	⋆			18.59	10473	2193	92	100.0%
		⋆		13.55	6992	2152	100	72.9%
	⋆		⋆	10.33	4739	2386	24	55.6%
		⋆	⋆	9.63	4415	2226	40	51.8%

Table 1
Numerical tests for a train carriage passing an S-shaped line change.

```
#fcw      number of function calls without counting those
          for the evaluation of the Jacobian
#Je       number of difference approximations of the Jacobian
percent   percentage of cpu-time compared to standard DASSL
```

The number `#Je` of finite difference approximations of the Jacobian J and the computing time `cpu` are of special interest.

Comparing the two columns `#fc` and `#fcw`, i. e., the number of function calls with and without the ones that are needed for evaluating the Jacobian, demonstrates the large computational effort for the finite difference approximations of J. The table shows that the adapted algorithms for Jacobian approximation and Jacobian update reduce the cpu-time significantly. The best results are achieved by combining the finite difference approximation (20) for sparse matrices with the partitioned Jacobian updates (24) and (25). Total savings of up to 50% are obtained. The differences between explicit multibody formalism and residual formalism are negligible.

At first glance it might be surprising that the total number of Jacobian evaluations is typically slightly increased if Algorithm TOMS 618 is applied to evaluate the finite difference approximation (20) of sparse Jacobians. The reason is the use of estimated sparsity patterns Ω instead of the exact sparsity patterns $\Sigma(J)$, see Section 3.1. In this example the estimated sparsity pattern Ω has to be updated several times during time integration. Nevertheless the overall cpu-time is decreased substantially by Algorithm TOMS 618 since the sparse difference approximations (20) are much cheaper than the standard difference approximations (16).

4.2 Multibody systems with dominating external time excitations

In this section we consider in detail the use of the extended partitioned Jacobian update (26) for a simple benchmark problem and for a test problem from automotive engineering. In the application to automotive engineering the overall computing time is reduced by more than 50% combining adapted finite difference approximations and (extended) partitioned updates of the Jacobian.

4.2.1 Benchmark: Chain of mathematical pendulums

Fig. 6 shows a chain of mathematical pendulums consisting of $N+1$ point masses $m_i = 1.0\,\mathrm{kg}$ that are connected by massless rods of length $l_i = 1.0\,\mathrm{m}$ and move under the influence of gravity. The state of the chain is described by the angles α_i between rod "i" and the $y-$axis, see Fig. 6. From the viewpoint of multibody dynamics the angle α_i may be considered as a coordinate that characterizes the degree of freedom in "joint" i.

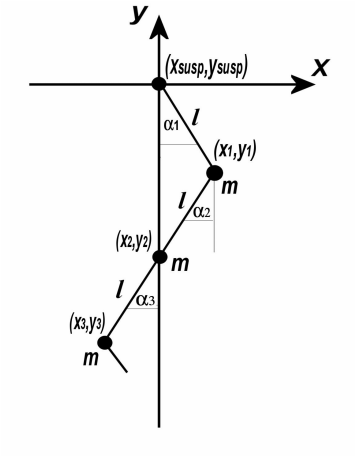

Fig. 6. Chain of pendulums.

The first mass point is attached to the suspension point $(x_{\mathrm{susp}}, y_{\mathrm{susp}})$ which is excited periodically:

$$x_{\mathrm{susp}}(t) = 2.0\,\mathrm{m} + \sin\omega t \cdot 0.3\,\mathrm{m}\,, \qquad y_{\mathrm{susp}}(t) = \sin\omega t \cdot 0.2\,\mathrm{m}$$

with $\omega = 0.05\,\mathrm{Hz}$. Frequency and amplitudes of the time excitation were chosen to guarantee an approximately linear system behaviour that is characterized by small deviations $|\alpha_i(t)|$ from the nominal position and by small velocities $|\dot{\alpha}_i(t)|$ over the whole time span $t \in [0\,\mathrm{s}, 200\,\mathrm{s}]$.

For the numerical tests the extended partitioned Jacobian update (26) was implemented in DASSL. The error bounds are set to $\texttt{ATOL} = 10^{-6}$ for the

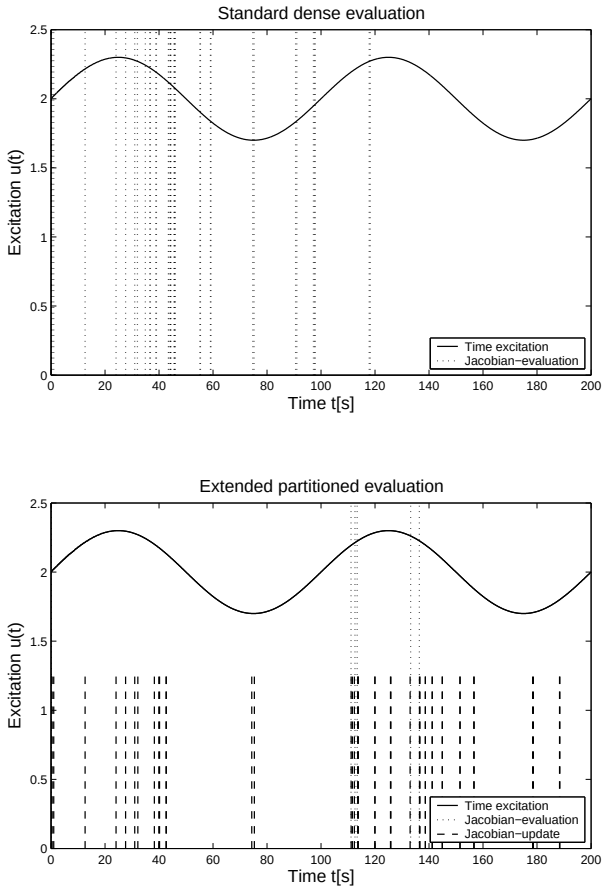

Fig. 7. Jacobian evaluations by finite differences (16) and updates (26).

absolute error and `RTOL` $= 10^{-4}$ for the relative error. The tests are performed for different values of N resulting in problem dimensions $n_p = 2N$, $n_g = 0$.

Fig. 7 shows for $N = 16$ the time instances of Jacobian evaluations by finite differences (16) and the time instances of cheap Jacobian updates (26). The sine function corresponds to the time excitation at the suspension point. The dotted lines mark the expensive finite difference approximations of the Jacobian whereas the dashed lines in the lower plot show the cheap updates in the extended partitioned method (26).

The upper plot shows the results for the standard implementation of DASSL with `#Je` $= 43$ finite difference approximations of the Jacobian in the time span $t \in [0\,\mathrm{s}, 200\,\mathrm{s}]$, see also Section 2.2. The lower plot illustrates the benefits of the extended partitioned Jacobian updates that substitute most of the finite difference approximations by cheap matrix additions (26) reducing the total number of finite difference approximations (16) to `#Je` $= 6$.

In this fairly simple benchmark problem without any time consuming force elements the extended partitioned Jacobian update (26) saves between 10% and 30% of the total computing time, see the results for $N = 12$ and $N = 14$

N	dense	sparse	extended partitioned	cpu	#Je	#updates (26)	percent
12	⋆			5.69	59	0	100%
		⋆		5.24	72	0	92.1%
	⋆		⋆	4.68	9	27	82.2%
		⋆	⋆	5.09	10	59	89.5%
14	⋆			9.03	96	0	100%
		⋆		6.76	70	0	74.9%
	⋆		⋆	6.78	12	62	75.1%
		⋆	⋆	6.16	8	16	68.2%

Table 2
Numerical tests for the benchmark "Chain of pendulums".

in Table 2. The savings are larger for higher problem dimensions since the numerical effort of standard finite differences (16) grows linearly with $n_p = 2N$. As in Section 4.1 we observe that the use of estimated sparsity patterns Ω in Algorithm TOMS 618 ("`sparse`") may result in an increased total number of Jacobian evaluations (`#Je` + `#updates`) but the overall computing time is nevertheless reduced in most computations.

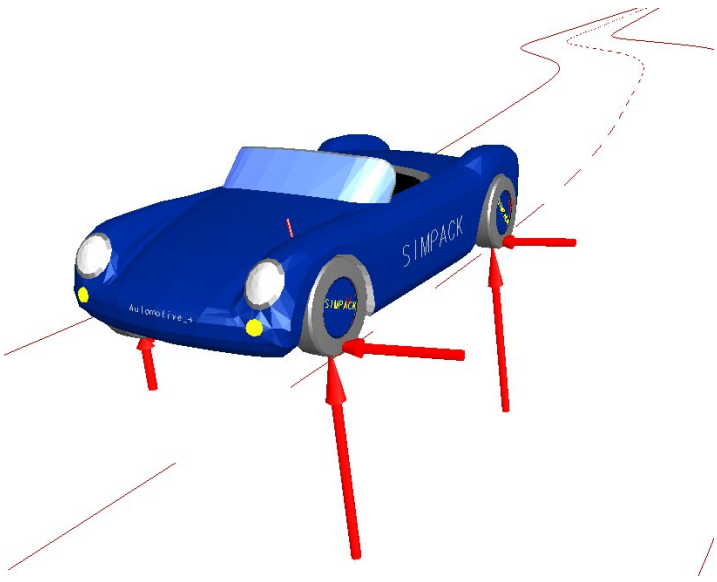

Fig. 8. Passenger car performing periodic line changes.

4.2.2 Dynamical simulation of a motor-car performing periodic line changes

The section on numerical experiments is completed by a test problem from automotive engineering, see Fig. 8. The model describes a passenger car that

drives with constant speed and performs periodic line changes.

The SIMPACK model has a rheonomic joint between car body and steering which is excited by a sinusoidal input $u(t)$. For moderate driving speed the car behaves approximately linearly. Therefore, substantial savings of computing time may be expected if the (extended) partitioned Jacobian updates (24) and (26) are used.

In the time integration a developer version of DASSL is applied with tolerances $\texttt{ATOL} = 10^{-5}$ for the absolute error and $\texttt{RTOL} = 10^{-4}$ for the relative error. The time integration is performed for $t \in [0\,\mathrm{s}, 5\,\mathrm{s}]$.

The results of Table 3 show that the overall computing time is decreased substantially by partitioned Jacobian updates (24). Further savings result from the additional update (26) of time dependent Jacobian entries. In the best case a reduction of up to 85% was obtained combining the extended partitioned update (26) with adapted finite differences (16) for sparse Jacobians.

dense	sparse	partitioned	extended partitioned	cpu	#Je	percent
$\star$				296.86	2229	100%
	$\star$			309.44	2303	103.3%
$\star$		$\star$		141.28	660	29.6%
$\star$			$\star$	145.75	668	30.0%
	$\star$	$\star$		79.66	410	18.4%
	$\star$		$\star$	59.45	317	14.2%

Table 3
Numerical tests for a passenger car performing periodic line changes.

5 Summary

If complex mechanical systems are described by joint coordinates then the computational effort in the dynamical simulation is typically dominated by the evaluation of the Jacobian of the equations of motion. The topology of the mechanical system is reflected by the structure of this Jacobian and should be exploited to save computing time.

The standard algorithm computes a difference approximation of the Jacobian column by column using one difference quotient per column of the Jacobian. If the Jacobian is sparse and its sparsity structure is explicitly known then several columns of the Jacobian may be approximated simultaneously using just one

perturbed function value. This algorithm for the difference approximation of sparse Jacobians reduces the overall computing time substantially.

The semi-explicit structure of the differential-algebraic equations of motion is exploited by partitioned Jacobian approximations that re-use information from previous time steps. If possible, the new Jacobian is calculated by a cheap matrix addition instead of an expensive difference approximation. Additional updates of time dependent entries in the Jacobian proved to be favourable for multibody system models with strong external excitations.

The methods were successfully implemented and tested in an industrial multibody system tool. Numerical experiments show substantial savings of computing time in the dynamical simulation of models from railway and automotive engineering.

Acknowledgements

The authors acknowledge many fruitful discussions on the subject with Wolfgang Rulka (Siemens Transportation Systems, Munich).

References

[1] A. Shabana, Dynamics of Multibody Systems, 2nd Edition, Cambridge University Press, Cambridge, 1998.

[2] B. Simeon, C. Führer, P. Rentrop, Differential-algebraic equations in vehicle system dynamics, Surveys on Mathematics for Industry 1 (1991) 1–37.

[3] W. Kortüm, W. Schiehlen, M. Arnold, Software tools: From multibody system analysis to vehicle system dynamics, in: H. Aref, J. Phillips (Eds.), Mechanics for a New Millennium, Kluwer Academic Publishers, Dordrecht, 2001, pp. 225–238.

[4] E. Eich-Soellner, C. Führer, Numerical Methods in Multibody Dynamics, Teubner–Verlag, Stuttgart, 1998.

[5] R. Roberson, R. Schwertassek, Dynamics of Multibody Systems, Springer–Verlag, Berlin Heidelberg New York, 1988.

[6] H. Brandl, R. Johanni, M. Otter, A very efficient algorithm for the simulation of robots and similar multibody systems without inversion of the mass matrix, in: P. Kopacek, I. Troch, K. Desoyer (Eds.), Theory of Robots, Pergamon Press, Oxford, 1988, pp. 95–100.

[7] A. Eichberger, Simulation von Mehrkörpersystemen auf parallelen Rechnerarchitekturen, Fortschritt-Berichte VDI Reihe 8, Nr. 332, VDI–Verlag, Düsseldorf, 1993.

[8] K. Brenan, S. Campbell, L. Petzold, Numerical solution of initial–value problems in differential–algebraic equations, 2nd Edition, SIAM, Philadelphia, 1996.

[9] E. Hairer, G. Wanner, Solving Ordinary Differential Equations. II. Stiff and Differential-Algebraic Problems, 2nd Edition, Springer–Verlag, Berlin Heidelberg New York, 1996.

[10] W. Rulka, Effiziente Simulation der Dynamik mechatronischer Systeme für industrielle Anwendungen, Ph.D. thesis, Vienna University of Technology, Department of Mechanical Engineering (1998).

[11] M. Arnold, Simulation algorithms and software tools, submitted to: G. Mastinu, M. Plöchl (Eds.), Road and Off-Road Vehicle System Dynamics Handbook, Taylor & Francis, London, to appear in 2005.

[12] C. Gear, B. Leimkuhler, G. Gupta, Automatic integration of Euler–Lagrange equations with constraints, J. Comp. Appl. Math. 12&13 (1985) 77–90.

[13] C. Kelley, Solving Nonlinear Equations with Newton's Method, SIAM, Philadelphia, 2003.

[14] K. Burrage, J. Butcher, F. Chipman, An implementation of singly–implicit Runge–Kutta methods, BIT 20 (1980) 326–340.

[15] T. Coleman, B. Garbow, J. Moré, Algorithm 618: FORTRAN subroutines for estimating sparse Jacobian matrices, ACM Transactions on Mathematical Software 10 (1984) 346–347.

[16] A. Fuchs, Effiziente Korrektoriteration für implizite Zeitintegrationsverfahren in der Mehrkörperdynamik, Master Thesis, Munich University of Technology, Department of Mathematics (2002).

[17] S. Dietz, G. Hippmann, G. Schupp, Interaction of vehicles and flexible tracks by co–simulation of multibody vehicle systems and finite element track models, in: H. True (Ed.), The Dynamics of Vehicles on Roads and on Tracks, Proc. of the 17th IAVSD Symposium, Denmark, 20-24 August 2001, Supplement to Vehicle System Dynamics, Vol. 37, Swets & Zeitlinger B.V., 2003, pp. 372–384.

[18] M. Arnold, H. Netter, Approximation of contact geometry in the dynamical simulation of wheel-rail systems, Mathematical and Computer Modelling of Dynamical Systems 4 (1998) 162–184.

[19] C. Führer, Differential-algebraische Gleichungssysteme in mechanischen Mehrkörpersystemen. Theorie, numerische Ansätze und Anwendungen, PhD Thesis, TU München, Mathematisches Institut und Institut für Informatik (1988).

[20] C. Führer, B. Leimkuhler, Numerical solution of differential–algebraic equations for constrained mechanical motion, Numer. Math. 59 (1991) 55–69.