MULTIBODY DYNAMICS 2011, ECCOMAS Thematic Conference
J.C. Samin, P. Fisette (eds.)
Brussels, Belgium, 4-7 July 2011

SYSTEMS WITH CONSTRAINT EQUATIONS IN THE SYMBOLIC MULTIBODY SIMULATION SOFTWARE NEWEUL-M^2

Thomas Kurz, Markus Burkhardt, Peter Eberhard

Institute of Engineering and Computational Mechanics
University of Stuttgart
Pfaffenwaldring 9, 70569 Stuttgart, Germany
e-mail: thomas.kurz@itm.uni-stuttgart.de,
markus.burkhardt@itm.uni-stuttgart.de,
peter.eberhard@itm.uni-stuttgart.de
www.itm.uni-stuttgart.de

Keywords: Symbolic Multibody Systems, Kinematic Constraints, Differential Algebraic Equations.

Abstract. *The research software Neweul-M^2 can be used to set up the equations of motion for rigid or elastic mechanical multibody systems symbolically. Depending on the structure of the system, this results in ordinary differential equations or systems of equations with additional algebraic constraint equations. These differential algebraic equations can either be solved directly or reformulated to improve the behavior during a time integration. Here, several different formulations are presented and compared.*

ISBN 978-2-8052-0116-5

1 Introduction

Symbolic equations are desirable for a number of engineering applications. They are preferable for real-time applications and uses, where the system shall not only be integrated over the time. This includes optimization, control design, or applications where their easy export to other simulation environments is very useful. On the other hand, symbolic modeling of mechanical systems always has to fight against problems arising from the complexity of systems, which usually can be handled easier by numerical algorithms. Here, possibilities to formulate systems with kinematic loops shall be presented and investigated. Such systems are common in many technical applications, but at first result in differential algebraic equations, and not in ordinary differential equations which are easier to handle. After an overview over the research software Neweul-M^2 the equations are set up and possible formulations are shown, which are then investigated and compared for a slider crank mechanism.

2 Neweul-M^2: A Research Software

The symbolical formalism Neweul-M^2 is a research software for the modeling, analysis and simulation of multibody systems, see [1]. It is based on the Newton-Euler equations and the principles of d'Alembert and Jourdain, see [2] and [3]. Neweul-M^2 is implemented in Matlab calling Maple or MuPad, whichever is present, through the Symbolic Math Toolbox for the symbolic manipulations. The system description is stored in a data structure containing all kinematic values and other necessary symbolic expressions. For the numerical evaluation, files in the Matlab language are written automatically, which then can also be used for other applications. The equations of motion, e.g., can be solved by any integration code for ordinary or differential algebraic equations, respectively. As all expressions are available symbolically, they can easily be exported to another programming language. From the expressions, e.g. Simulink S-functions in C can be created automatically. Most commercial programs use solely numerical algorithms for the modeling and simulation of multibody systems. Then, the modeling can be based on catalogues, e.g., of constraints or force elements, and a time integration is easily possible even for complex systems. However, it is advantageous for many applications to obtain a symbolical formulation, which shall be shortly discussed in the following. When a system is modeled symbolically, the describing kinematic values like velocity and acceleration, as well as the equations of motion are expressed depending on user-defined variables. Those variables are read from the input data and used to obtain all necessary quantities. After the kinematic values and equations of motion have been derived only once, they can be used for fast numerical evaluations. Since the expressions are set up prior to the numerical simulations, those evaluations are very fast, allowing real-time applications. When performing a parameter optimization of nonlinear equations, the fast calculation of gradients with a high accuracy is crucial. The symbolic system formulation can be used in the calculation of these gradients in analytical or semi-analytical formalisms, depending on the optimization criteria. When the criterion function depends on a numerical time integration of the equations of motion, these semi-analytical gradients reach an accuracy in the same order of error as the time integration involved, and can be obtained fully automatic. Also when a controller is to be designed, the symbolical expressions allow more possibilities and strategies than mere numerical algorithms, see [4]. In the following, some recent developments shall be presented after the basics have been summarized.

ISBN 978-2-8052-0116-5

3 Systems with tree structure

The software Neweul-M^2 uses the Newton-Euler equations, which are conveniently set up in the respective frame of reference of each body, see [5]. For systems with tree structure, this can be easily done. A system has a tree structure as long as every coordinate system and body has clearly only one frame of reference with respect to which it has been defined, see Fig. 1.

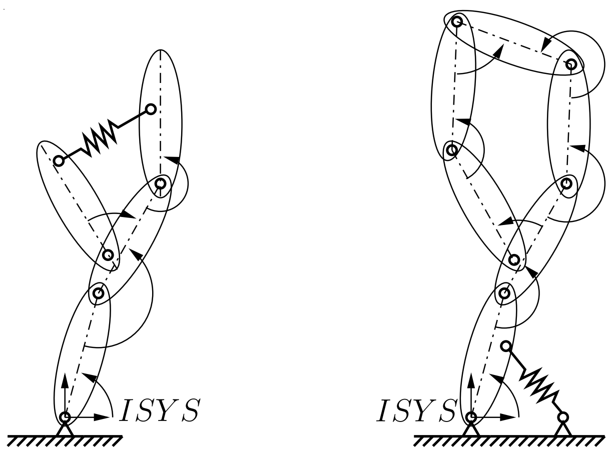

Figure 1: Difference between systems with tree or loop structure.

Here, usually the constraint equations are not explicitly formulated but are contained in the possible motion of the coordinate systems. By a premultiplication with the transposed global Jacobian matrix $\mathbf{J}$, the equations of motion can be obtained

$$\mathbf{J}^T \cdot \overline{\mathbf{M}} \cdot \mathbf{J} \cdot \ddot{\mathbf{y}} + \mathbf{J}^T \cdot \mathbf{q}^c = \mathbf{J}^T \cdot \mathbf{q}^a + \mathbf{J}^T \cdot \mathbf{Q} \cdot \boldsymbol{\lambda} \; . \tag{1}$$

Here the generalized coordinates are denoted by $\mathbf{y}$ and $\overline{\mathbf{M}}$ is the global mass matrix of the Newton-Euler equations. The forces are split up in $\mathbf{q}^c$ which contains local accelerations and will result in generalized Coriolis, centrifugal and gyroscopic forces, $\mathbf{q}^a$ are the applied forces and $\boldsymbol{\lambda}$ are the reaction forces, together with the distribution matrix $\mathbf{Q}$. As this premultiplication eliminates the reaction forces, the following abbreviations can be introduced for the equations in minimal form, where all generalized coordinates are independent

$$\mathbf{M} \cdot \ddot{\mathbf{y}} + \mathbf{k} = \mathbf{q} \; . \tag{2}$$

Here, the generalized applied forces are collected in the vector $\mathbf{q}$, the generalized centrifugal, Coriolis and gyroscopic forces are in the vector $\mathbf{k}$, while $\mathbf{M}$ is the minimal mass matrix.

4 Systems with kinematic loops

When additional algebraic constraint equations are introduced, they are commonly added to the system of equations together with Lagrangian multipliers, resulting in a set of Differential Algebraic Equations (DAE).

4.1 Differential Algebraic Equations

A set of Differential Algebraic Equations (DAE) is obtained by adding the constraint equations $\mathbf{c} = \mathbf{0}$ or one of its time derivatives to the existing Ordinary Differential Equations (ODE),

ISBN 978-2-8052-0116-5

see Eq. (2)

$$\begin{aligned} \mathbf{M}\cdot\ddot{\mathbf{y}}+\mathbf{k} &= \mathbf{q}+\mathbf{C}^T\cdot\boldsymbol{\lambda} & (3)\\ \ddot{\mathbf{c}}(t,\mathbf{y},\dot{\mathbf{y}}) &= \mathbf{0}\;. & (4)\end{aligned}$$

Here the generalized coordinates $\mathbf{y}$ are no longer all independent, but due to the constraint equations they depend on each other. There exist many integration algorithms to solve such kind of systems. However, most of them require the constraint equations to be formulated on acceleration level. This means, that the integration algorithm performs a root search in every step to fulfill the constraint equations within the given tolerances. However, the velocity and position level constraints are not completely satisfied due to integration errors, which is the so called drift error. This means that one criterion to compare the different formulations of systems which started as DAE-systems is the drift behavior. Of course the computational effort necessary to solve the equations is another criterion.

Starting from the constraint equations on position level

$$\mathbf{c}(t,\mathbf{y})=\mathbf{0}\;, \qquad (5)$$

we can obtain the constraint equations on velocity and acceleration level

$$\begin{aligned} \dot{\mathbf{c}}(t,\mathbf{y},\dot{\mathbf{y}}) &= \frac{\partial\mathbf{c}}{\partial\mathbf{y}}\cdot\dot{\mathbf{y}}+\frac{\partial\mathbf{c}}{\partial t}=\mathbf{C}\cdot\dot{\mathbf{y}}+\mathbf{b}'=\mathbf{0} & (6)\\ \ddot{\mathbf{c}}(t,\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}}) &= \mathbf{C}\cdot\ddot{\mathbf{y}}+\dot{\mathbf{C}}\cdot\dot{\mathbf{y}}+\frac{d\mathbf{b}'}{dt}=\mathbf{C}\cdot\ddot{\mathbf{y}}+\mathbf{b}''=\mathbf{0}\;. & (7)\end{aligned}$$

Usually the equations of motion are formulated using the constraint equations on acceleration level $\ddot{\mathbf{c}}$, which then can be written in matrix notation as

$$\begin{bmatrix}\mathbf{M} & \mathbf{C}^T\\ \mathbf{C} & \mathbf{0}\end{bmatrix}\cdot\begin{bmatrix}\ddot{\mathbf{y}}\\ -\boldsymbol{\lambda}\end{bmatrix}=\begin{bmatrix}\mathbf{q}-\mathbf{k}\\ -\mathbf{b}''\end{bmatrix}\;. \qquad (8)$$

The additional constraint equations provide another difficulty. The user has to specify generalized coordinates and corresponding constraint equations. While it is quite easy to formulate them reasonably for the initial configuration, it may be quite difficult in the whole coordinate space. It may happen that two or more of the constraint equations become linearly dependent. This means that the system has a singular configuration in which the equations may not be solvable. The system can reach these configurations, just the choice of generalized coordinates will be not suitable for the description.

Since many integration algorithms require the matrix on the left hand side to be constant, the Eq. (8) is not suitable. Also, many standard integration algorithms require a state-space representation of the equations of motion, two possibilities for this are implemented in Neweul-M^2. The first is an explicit formulation

$$\begin{bmatrix}\mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}\cdot\begin{bmatrix}\dot{\mathbf{y}}\\ \ddot{\mathbf{y}}\\ \dot{\boldsymbol{\lambda}}\end{bmatrix}=\begin{bmatrix}\dot{\mathbf{y}}\\ \mathbf{M}^{-1}\cdot(\mathbf{q}-\mathbf{k}+\mathbf{C}^T\cdot\boldsymbol{\lambda})\\ \ddot{\mathbf{c}}\end{bmatrix}\;. \qquad (9)$$

The second available formulation is an implicit formulation

$$\begin{bmatrix}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}\cdot\begin{bmatrix}\dot{\mathbf{y}}\\ \ddot{\mathbf{y}}\\ \dddot{\mathbf{y}}\\ \dot{\boldsymbol{\lambda}}\end{bmatrix}=\begin{bmatrix}\dot{\mathbf{y}}\\ \ddot{\mathbf{y}}\\ \mathbf{M}\cdot\ddot{\mathbf{y}}+\mathbf{k}-\mathbf{q}-\mathbf{C}^T\cdot\boldsymbol{\lambda}\\ \ddot{\mathbf{c}}\end{bmatrix}\;. \qquad (10)$$

ISBN 978-2-8052-0116-5

The naming comes from the formulation of the actual equations of motion as the part due to the state space formulation is always explicit and the constraint equations always implicit in Neweul-M^2. The main difference is that the first formulation requires either the mass matrix $\mathbf{M}$ to be inverted, or usually the corresponding system of equations to be solved. This is done implicitly in the second formulation by a root search. However, this comes at the expense of a larger state vector.

The DAE formulation is usually a good way to simulate the system as long as the system does not reach singular configurations, see [6]. Then it is quite unpredictable what will happen. Therefore, it is interesting to investigate other possibilities to satisfy the algebraic constraints in the equations.

4.2 Separation of coordinates, basic algorithm

Let us for now assume that the system does not reach a singular position. Then the problem is, to make a good choice of independent coordinates $\mathbf{y}_i$ among all available generalized coordinates $\mathbf{y}$. As soon as this choice is made, it is possible to solve for all dependent coordinates $\mathbf{y}_d$ and use these values in the equations of motion. It is usually impossible to do such a solution symbolically and insert it into the equations of motion because of several possible solutions of the constraint equations and required information to attain such a step. However, we will later refine this algorithm so that it can be applied generally.

Starting from the constraint equations (5-7), let us separate the generalized coordinates $\mathbf{y}$ into independent $\mathbf{y}_i$ and dependent coordinates $\mathbf{y}_d$. When using variational calculus and also separating the Jacobian matrix of the constraints $\mathbf{C}$ we get

$$\begin{bmatrix} \mathbf{C}_i & \mathbf{C}_d \end{bmatrix} \cdot \begin{bmatrix} \delta\mathbf{y}_i \\ \delta\mathbf{y}_d \end{bmatrix} = \mathbf{0} \; . \tag{11}$$

This means that the variation of the generalized coordinates has to respect the constraint equations, but by solving this for the dependent variations we can rewrite this equation and obtain

$$\delta\mathbf{y} = \begin{bmatrix} \delta\mathbf{y}_i \\ \delta\mathbf{y}_d \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{C}_d^{-1} \cdot \mathbf{C}_i \end{bmatrix} \cdot \delta\mathbf{y}_i = \mathbf{J}_i \cdot \delta\mathbf{y}_i \; . \tag{12}$$

With this, we can rewrite the constraint equations on velocity level, Eq. (6), to

$$\mathbf{C}_i \cdot \dot{\mathbf{y}}_i + \mathbf{C}_d \cdot \dot{\mathbf{y}}_d + \mathbf{b}' = \mathbf{0} \; , \tag{13}$$

or

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{\mathbf{y}}_i \\ \dot{\mathbf{y}}_d \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{C}_d^{-1} \cdot \mathbf{C}_i \end{bmatrix} \cdot \dot{\mathbf{y}}_i + \begin{bmatrix} \mathbf{0} \\ -\mathbf{C}_d^{-1} \cdot \mathbf{b}' \end{bmatrix} = \mathbf{J}_i \cdot \dot{\mathbf{y}}_i + \boldsymbol{\theta} \; . \tag{14}$$

Similarly with Eq. (7) on acceleration level it follows

$$\ddot{\mathbf{y}} = \begin{bmatrix} \ddot{\mathbf{y}}_i \\ \ddot{\mathbf{y}}_d \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{C}_d^{-1} \cdot \mathbf{C}_i \end{bmatrix} \cdot \ddot{\mathbf{y}}_i + \begin{bmatrix} \mathbf{0} \\ -\mathbf{C}_d^{-1} \cdot \mathbf{b}'' \end{bmatrix} = \mathbf{J}_i \cdot \ddot{\mathbf{y}}_i + \boldsymbol{\gamma} \; . \tag{15}$$

Here again, the matrix $\mathbf{J}_i$ appears, which can be used to transform the equations of motion into the space of the independent generalized coordinates

$$\mathbf{J}_i^T \cdot \mathbf{M} \cdot \mathbf{J}_i \cdot \ddot{\mathbf{y}}_i + \mathbf{J}_i^T \cdot \mathbf{k} + \mathbf{J}_i^T \cdot \mathbf{M} \cdot \boldsymbol{\gamma} = \mathbf{J}_i^T \cdot \mathbf{q} \; . \tag{16}$$

ISBN 978-2-8052-0116-5

To write these equations of motion in state-space form, the constraint equations on velocity level, Eq. (14), and the part of the dependent accelerations of Eq. (15) can be combined with the equations of motion solved for the independent accelerations, Eq. (16), to obtain

$$\begin{bmatrix} \dot{\mathbf{y}}_i \\ \dot{\mathbf{y}}_d \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{C}_d^{-1} \cdot \mathbf{C}_i \end{bmatrix} \cdot \dot{\mathbf{y}}_i + \begin{bmatrix} \mathbf{0} \\ -\mathbf{C}_d^{-1} \cdot \mathbf{b}' \end{bmatrix} \tag{17}$$

$$\begin{bmatrix} \ddot{\mathbf{y}}_i \\ \ddot{\mathbf{y}}_d \end{bmatrix} = \begin{bmatrix} \left(\mathbf{J}_i^T \cdot \mathbf{M} \cdot \mathbf{J}_i\right)^{-1} \cdot \left(\mathbf{J}_i^T \cdot \mathbf{q} - \mathbf{J}_i^T \cdot \mathbf{k} - \mathbf{J}_i^T \cdot \mathbf{M} \cdot \boldsymbol{\gamma}\right) \\ -\mathbf{C}_d^{-1} \cdot \mathbf{C}_i \cdot \ddot{\mathbf{y}}_i - \mathbf{C}_d^{-1} \cdot \mathbf{b}'' \end{bmatrix} \tag{18}$$

In the last line, the independent acccelerations $\ddot{\mathbf{y}}_i$ are contained. It looks strange to have those variables on the right hand side of the equation, but as the evaluation goes from top to bottom, the just calculated results can be used. This formulation has the advantage to use the constraint equations to obtain the matrix $\mathbf{J}_i$ with which the equations of motion can be transformed to minimal form. As we used the constraint equations on acceleration level, we could not make an improvement concerning the drift effect. As we have a manual selection of the independent coordinates, this does not solve any problem with singularities. Here they appear as problems in the inversion of the dependent part of the constraint Jacobian matrix $\mathbf{C}_d$.

4.3 Improvements to the basic algorithm

As we just saw, there are still some open issues in this formulation. We can address them in several steps. At first, it is very useful to use indexing of vectors and matrices, which can be done very conveniently in Matlab. This allows us to switch from the stacking of independent coordinates on top of the dependent ones to an elementwise identification. It improves this method as it allows us to smoothly exchange the coordinates which we consider as independent if their choice becomes troublesome. This means, we could formulate a transformation matrix $\mathbf{T}$ to choose the independent and dependent vectors

$$\begin{bmatrix} \mathbf{y}_i \\ \mathbf{y}_d \end{bmatrix} = \mathbf{T} \cdot \mathbf{y} \; . \tag{19}$$

Here this matrix would simply contain ones or zeros to select coordinates. However, if we can find a suitable matrix $\mathbf{T}$, we can use the best linear combinations of $\mathbf{y}$ as independent coordinates. Such a formulation was described in [7]. The use of such a transformation matrix has two possible conclusions. Either we consider the matrix $\mathbf{T}$ to be constant and update it only upon violation of some criterion. Then, the choice of coordinates is not optimal, but it is easy to implement. If we want to determine such a matrix in every time step, Leister and Bestle [7] had to rewrite the integration algorithm, which is also no really satisfying solution.

To evaluate the equations of motion we want to transform the generalized coordinates $\mathbf{y}$ on a minimal formulation $\mathbf{y}_i$, which consists of the optimal linear combinations of all generalized coordinates. This happens inside the function called by the integration algorithm, as described in the following scheme.

So we can conclude the following steps

- The integration algorithm calls the m-file function and passes the complete vector of generalized coordinates $\mathbf{y}$
  - Calculate the transformation matrix $\mathbf{T}$, which describes the optimal linear combination of generalized coordinates $\mathbf{y}_i$ and the corresponding dependent coordinates $\mathbf{y}_d$

ISBN 978-2-8052-0116-5

  - Transform the generalized coordinates and all expressions into the space of minimal coordinates $\mathbf{y}_i$
  - Evaluate the equations of motion in minimal form for $\mathbf{y}_i$
  - Evaluate the constraint equations to obtain values for the dependent coordinates $\mathbf{y}_d$
  - Perform back transformation so the generalized coordinates $\mathbf{y}$ can be passed back to the integrator

- Prepare the next integration step

It is the advantage of the algorithm which was just explained, that we always get the optimal linear combination of generalized coordinates to describe the system in this time step. As the integration algorithm is always considering the same, full state vector we don't have any problems with additional derivatives of the transformation. Therefore, we can now investigate how to find this optimal linear combination.

4.4 Optimal linear combination of generalized coordinates

As the generalized coordinates have to fulfill the constraint equations, it is useful to imagine valid directions for the generalized coordinates and orthogonal to them the reaction forces caused by the constraints. This results in a depiction of the constraint manifold and the manifold of possible movement. From Eqs. (5-6) we can see that the generalized coordinates on velocity and acceleration level have to be orthogonal to the constraints. Then, the Jacobian matrix of constraints $\mathbf{C}$ can be interpreted as a collection of vectors in the constrained directions. For system configurations where these vectors become dependent, a solution is no longer easily possible. Therefore, in this context optimal describes the fact that we find one orthonormal basis for each of the two subspaces, the possible movements and the directions of the reaction forces. There are two convenient methods to find all vectors which are orthogonal to a certain matrix, the Singular Value Decomposition (SVD) and the QR decomposition. Both algorithms are available in Matlab, a detailed explanation can be found, e.g., in [8].

4.4.1 QR decomposition

A QR decomposition of the transposed Jacobian matrix of the constraints $\mathbf{C}^T$ results in

$$\mathbf{C}^T = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_1 \cdot \mathbf{R}_1 \ . \tag{20}$$

The column vectors of $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are orthonormal, while the matrix $\mathbf{R}_1$ is an upper triangular matrix. In many applications the matrix to be decomposed has full rank and thus no matrix $\mathbf{Q}_2$. The idea of a QR decomposition is to split a given matrix into an orthogonal matrix and the remaining part which results in an upper triangular matrix. Then the orthogonal matrix is the optimal set of vectors which could be used as a basis to span the given space. Therefore, here the column vectors of matrix $\mathbf{Q}_1$ are an optimal basis of the constraint manifold, and analog the matrix $\mathbf{Q}_2$ for the valid motions. Therefore, a projection on the space described by $\mathbf{Q}_2$ results in an optimal linear combination of generalized coordinates for the current configuration.

By comparison with Eq. (15) we find

$$\ddot{\mathbf{y}} = \mathbf{Q}_2 \cdot \ddot{\mathbf{y}}_i + \mathbf{Q}_1 \cdot \ddot{\mathbf{y}}_d \ , \tag{21}$$

ISBN 978-2-8052-0116-5

and can thus identify $\mathbf{J}_i = \mathbf{Q}_2$. Please note that the independent and dependent generalized coordinates $\mathbf{y}_i$ and $\mathbf{y}_d$ are no longer single elements of the vector $\mathbf{y}$ and a vectorized formulation as in Eq. (15) is no longer possible. If we insert the just received result in the constraint equations on acceleration level, see Eq. (7), we obtain

$$\mathbf{C} \cdot \mathbf{Q}_2 \cdot \ddot{\mathbf{y}}_i + \mathbf{C} \cdot \mathbf{Q}_1 \cdot \ddot{\mathbf{y}}_d + \mathbf{b}'' = \mathbf{0} \,. \tag{22}$$

With Eq. (20) we get

$$\mathbf{R}_1^T \cdot \underbrace{\mathbf{Q}_1^T \cdot \mathbf{Q}_2}_{=\mathbf{0}} \cdot \ddot{\mathbf{y}}_i + \mathbf{R}_1^T \cdot \underbrace{\mathbf{Q}_1^T \cdot \mathbf{Q}_1}_{=\mathbf{I}} \cdot \ddot{\mathbf{y}}_d + \mathbf{b}'' = \mathbf{0} \,. \tag{23}$$

Using Eq. (21), we can summarize for the generalized accelerations

$$\ddot{\mathbf{y}} = \mathbf{J}_i \cdot \ddot{\mathbf{y}}_i - \mathbf{Q}_1 \cdot \mathbf{R}_1^{-T} \cdot \mathbf{b}'' = \mathbf{J}_i \cdot \ddot{\mathbf{y}}_i + \boldsymbol{\gamma} \,. \tag{24}$$

The last equation was set up so the comparison with the method presented next is easier.

4.4.2 Singular value decomposition

Similar to the usage of the QR decomposition, the singular value decomposition can be used

$$\mathbf{C}^T = \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{\Sigma} \\ \mathbf{0} \end{bmatrix} \cdot \mathbf{V}^T = \mathbf{U}_1 \cdot \boldsymbol{\Sigma} \cdot \mathbf{V}^T \,. \tag{25}$$

The singular value decomposition also finds an optimal basis and for non-square matrices vectors to span the remaining directions. Again, most often this method is used to find a good basis to a given set of vectors, however we are more interested in the good basis of the remaining coordinate space, denoting the valid motions. Analog to the just described QR decomposition we can identify the matrix $\mathbf{J}_i = \mathbf{U}_2$ and obtain

$$\mathbf{C} \cdot \mathbf{U}_2 \cdot \ddot{\mathbf{y}}_i + \mathbf{C} \cdot \mathbf{U}_1 \cdot \ddot{\mathbf{y}}_d + \mathbf{b}'' = \mathbf{0} \,. \tag{26}$$

Inserting the relation $\mathbf{C} = \mathbf{V} \cdot \boldsymbol{\Sigma} \cdot \mathbf{U}_1^T$ leads to

$$\ddot{\mathbf{y}}_d = -\boldsymbol{\Sigma}^{-1} \cdot \mathbf{V}^T \cdot \mathbf{b}'' \,, \tag{27}$$

which results in

$$\ddot{\mathbf{y}} = \mathbf{J}_i \cdot \ddot{\mathbf{y}}_i - \mathbf{U}_1 \cdot \boldsymbol{\Sigma}^{-1} \cdot \mathbf{V}_1^{-T} \cdot \mathbf{b}'' = \mathbf{J}_i \cdot \ddot{\mathbf{y}}_i + \boldsymbol{\gamma} \,. \tag{28}$$

4.5 Automatic separation of coordinates

Equations (24) and (28) allow us to choose any of the two presented methods to switch between all generalized coordinates $\mathbf{y}$ and the independent coordinates $\mathbf{y}_i$. We now have to combine the projected velocities as in Eq. (14) with the equations of motion in the independent coordinates and the constraints on acceleration level, see Eq. (18).

Using the abbreviations of Eq. (24) we obtain the equations of motion as

$$\mathbf{J}_i^T \cdot \mathbf{M} \cdot \mathbf{J}_i \cdot \ddot{\mathbf{y}}_i + \mathbf{J}_i^T \cdot \mathbf{k} - \mathbf{J}_i^T \cdot \mathbf{q} + \mathbf{J}_i^T \cdot \mathbf{M} \cdot \boldsymbol{\gamma} = \mathbf{0} \,. \tag{29}$$

These equations of motion are now in minimal form and always use an optimal choice of generalized coordinates. In order to use them for a time integration, they now have to be transformed back to the complete vector of generalized coordinates using Eq. (24).

ISBN 978-2-8052-0116-5

The independent and dependent generalized coordinates $\mathbf{y}_i$ and $\mathbf{y}_d$ are linear combinations of the generalized coordinates $\mathbf{y}$. For the velocity level, we use Eq. (14) to calculate the independent velocities and from them calculate the full vector of generalized velocities and similarly for the acceleration level using Eq. (15)

$$\dot{\mathbf{y}} = \mathbf{J}_i \cdot \mathbf{J}_i^T \cdot (\dot{\mathbf{y}} - \boldsymbol{\theta}) + \boldsymbol{\theta} \tag{30}$$

$$\ddot{\mathbf{y}} = \mathbf{J}_i \cdot \left(\mathbf{J}_i^T \cdot \mathbf{M} \cdot \mathbf{J}_i\right)^{-1} \cdot \left(\mathbf{J}_i^T \cdot \mathbf{k} - \mathbf{J}_i^T \cdot \mathbf{q} + \mathbf{J}_i^T \cdot \mathbf{M} \cdot \boldsymbol{\gamma}\right) + \boldsymbol{\gamma} \; . \tag{31}$$

Now we obtained the equations of motion in the full vector of generalized coordinates which can be used for time integration.

5 Comparison of results

In order to compare the different ways to formulate the equations of motion, a slider crank mechanism is used as an example, see Fig. 1.

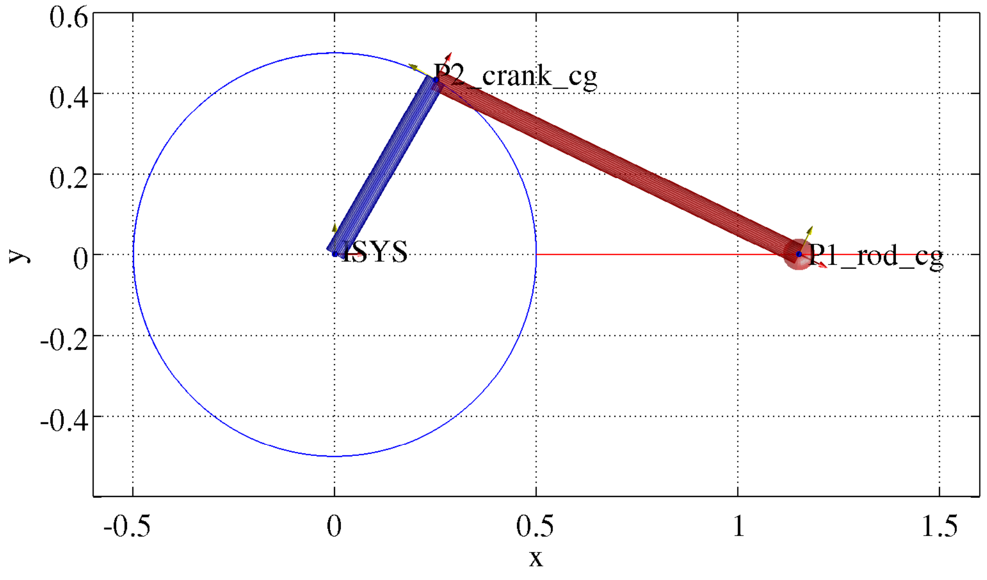

Figure 2: Slider crank mechanism.

Usually when such a mechanism is modeled, one would start at the crank on the left side and use relative angles up to the slider on the right. Then the loop closing condition would be to prevent any motion in y-direction of the slider. However, here the mechanism has been modeled starting from the right. Then two constraint equations are required, the possible coordinates show distinct singular configurations, and the drift behavior is clearly visible. The generalized coordinates are the x-coordinate of the slider, the rotation angle α of the rod, which is depicted red and the relative rotation angle β of the crank, depicted blue. The gravity shall act in negative z-direction, then not appearing in the equations of motion. The shown configuration with an initial velocity is used as initial condition for the time integration. No additional stabilization has been used as the drift effect shall be considered, even though it is available in Neweul-M^2. One advantage of this simple model is that an analytical formulation of the equations of motion can be easily set up as a reference.

This system has been modeled using all the formulations described above over two different time intervals, see Table 1. The time used for all systems to set up the symbolic equations is

ISBN 978-2-8052-0116-5

Table 1: Simulation times of a slider crank mechanism, simulated over different time intervals.

Formulation	Modeling Time	Simulation Time (40s)	Simulation Time (300s)
Analytical	3.29 s	0.67 s	2.57 s
Explicit	2.56 s	2.73 s	–
Partialexplicit	2.36 s	1.86 s	–
Partition, $y_i = x$	2.40 s	0.64 s	4.85 s
Partition, $y_i = \alpha$	2.41 s	0.63 s	4.62 s
Partition, $y_i = \beta$	2.27 s	0.64 s	4.59 s
SVD	2.43 s	0.87 s	6.53 s
QR	2.52 s	0.81 s	6.31 s

roughly the same, which is listed in the column Modeling Time in Table 1. All systems could be integrated over a time interval of 40 s, however, the DAE formulations did not reach the end of the second time interval of 300 s. This comes from the drift effect, which caused a violation of the constraint equations too large to continue the integration, see Fig. 3. The comparison of the simulation times shows that the DAE formulations take much longer than any other simulation, when considering the 40 s time interval which all methods finished. The SVD and QR method take longer as the manual selection of an independent coordinate, which was expected as additional computations are necessary. For the longer time interval, the advantage of the analytical solution comes through, which has to be the fastest possible solution as it has the smallest possible set of ordinary differential equations.

In Fig. 3 the violation of the constraint equation in x-direction is shown for five different system formulations. The QR method shows no difference to the SVD method resulting in congruent lines, which show the least drift effect. The fixed choice of the x coordinate of the slider as the independent coordinate results in the second best drift behavior. The other manual choices would result in similar curves. However, here the restriction has to be made, that by chance the integration algorithm always stepped over the singular configuration, otherwise resulting in bad results. But depending on the system, such a manual selection often reaches the same drift behavior as the QR and SVD method. Therefore, as soon as the user knows enough about the system, the manual selection is a very good choice. The QR and SVD method, on the other hand, have the advantage of making optimal choices independent of the experience of the user. When the system dimension and the number of constraint equations is increased, the additional computational effort for these methods will increase as well. The two DAE formulations show a distinctly nonlinear, progressive drift behavior which caused the integration not to reach the desired 300 s.

6 Conclusions

Different possibilities to formulate a system of differential algebraic equations have been shown. These equations arise in mechanical multibody systems by the implementation of constraint equations as they result from kinematic loops. In Neweul-M^2, such systems can be simulated keeping these algebraic constraint equations offering two formulations of the equations. These results have been compared to an analytical solution and different ways to eliminate the algebraic equations ending up with ordinary differential equations again. In the investigated example, a slider crank mechanism, these ODE formulations were superior in both computational times and observable errors, here the drift behavior has been used as a representative. The manual selection of independent coordinates seems to be a very good choice as long as the

ISBN 978-2-8052-0116-5

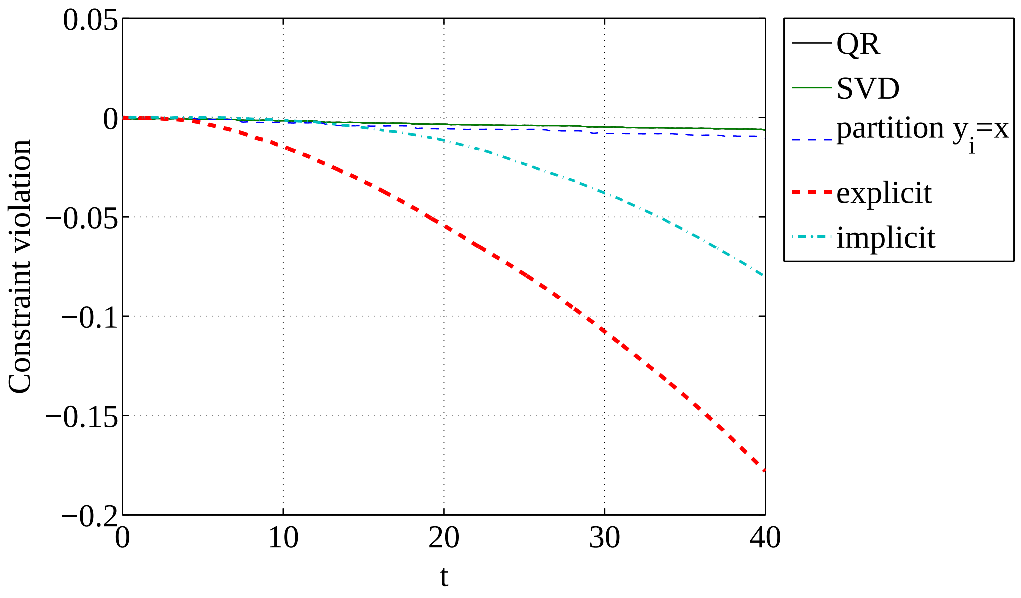

Figure 3: Violation of the constraint on position level in y-direction.

integration algorithm does not reach singular configurations and the user can choose a suitable coordinate. The SVD or QR method provide reliable results with a very small drift effect and reasonable computational effort.

Acknowledgement

This research is partially supported by SimTech, the Cluster of Excellence within the Stuttgart Research Center for Simulation Technology. This support is highly appreciated.

REFERENCES

[1] Kurz, T.; Eberhard, P.; Henninger, C.; Schiehlen, W.: From Neweul to Neweul-M^2: Symbolical Equations of Motion for Multibody System Analysis and Synthesis. Multibody System Dynamics, Vol. 24, No. 1, pp. 25–41, 2010.

[2] Popp, K.; Schiehlen, W.: Ground Vehicle Dynamics. Berlin: Springer, 2010.

[3] Schiehlen, W.; Eberhard, P.: Technische Dynamik – Modelle für Regelung und Simulation (in German). Wiesbaden: Teubner, 2004.

[4] Seifried, R.; Held, A.; Dietmann, F.: Analysis of Feed-Forward Control Designs for Flexible Multibody Systems. Journal of System Design and Dynamics, 2010. Special Issue ACMD.

[5] Schwertassek, R.; Wallrapp, O.: Dynamik flexibler Mehrkörpersysteme (in German). Braunschweig: Vieweg, 1999.

[6] Fleissner, F.: Bewertung numerischer Verfahren zur Lösung differential-algebraischer Systeme im Kontext mechanischer Mehrkörpersysteme. Diplomarbeit, Universität Erlangen-Nürnberg, Lehrstuhl für Technische Mechanik, 2003.

ISBN 978-2-8052-0116-5

[7] Leister, G.; Bestle, D.: Symbolic-Numerical Solution of Multibody Systems with Closed Loops. Vehicle System Dynamics, Vol. 21, pp. 129–142, 1992.

[8] Golub, G.; Van Loan, C.: Matrix Computations. Baltimore: The Johns Hopkins University Press, 3^{rd} Edn., 1996.

ISBN 978-2-8052-0116-5