International Symposium on Coupled Methods in Numerical Dynamics
Split, Croatia, September 16th-19th, 2009.

Symbolic Modeling and Analysis of Elastic Multibody Systems

Thomas Kurz and Peter Eberhard

University of Stuttgart, Institute of Engineering and Computational Mechanics,
Pfaffenwaldring 9, 70569 Stuttgart, Germany
`[kurz,eberhard]@itm.uni-stuttgart.de`

Abstract. The algorithms to set up the equations of motion symbolically for both rigid and elastic multibody systems are presented. In the described program Neweul-M^2 which is using the power of Matlab and Maple, the modeling approach with commands and a graphical user interface are discussed as well as an overview of possibilities for system analysis, control design and optimization is given. A double pendulum is modeled both with rigid and elastic bodies to explain the program features.

Key words: *Symbolic Equations of Motion, Elastic Multibody Systems, Research Software, Matlab*

1. Introduction

The generation of equations of motion for large multibody systems is a non-trivial task requiring numerous steps during the evaluation of the fundamental relations. Multibody system formalisms are founded on Lagrange's equations of the first or second kind, or the Newton-Euler equations and D'Alembert's or Jourdain's principle, respectively, see e.g. Kane and Levinson [3], Lu [8] and Schiehlen [10]. Regarding the computational procedure, numerical and symbolical formalisms are distinguished. A numerical formalism provides the numbers in the equations of motion required for each time step of the simulation program. In contrary, symbolical formalisms generate the equations of motion only once with a computer code as it would be done with paper and pencil. The advantage is that different values of the system parameters can be inserted in the symbolical equations of motion, but also the structure of the equations can be further utilized. Symbolical formalisms are especially helpful for real-time simulations, optimizations and control design, where the created equations can be used in a flexible way by other programs, too.

2. Neweul-M^2: The Algorithm

The symbolical formalism Neweul-M^2 is a research software based on the Newton-Euler equations and the principles of d'Alembert and Jourdain. It generates equations of motion in minimal form for open-loop systems, and differential algebraic equations for systems with closed kinematic loops. These can be solved by any integration code for ordinary differential or differential algebraic equations, respectively. Due to the fact that Neweul-M^2 is a research software the user has full access to the complete source code. Thus, it is easier to identify any problems arising in the modeling and simulation process and to adjust and extend the functionality of the code.

The approach used will be briefly presented next, before explaining the software in more detail. The formalism comprises four steps, see also Schiehlen and Eberhard [11] and Popp and Schiehlen [9]. Here, the equations of motion for an elastic multibody system shall be derived, where some bodies are flexible. Flexibility is described in the system by a small elastic deformation overlayed to a large, possibly nonlinear motion of the respective frame of reference, see Schwertassek and Wallrapp [12].

Step 1 (rigid and flexible): System specification and data input. At first, the multibody system is defined and parameters used for the entire system have to be provided. The number of degrees of freedom is specified and the f generalized coordinates y_k are chosen. The inertial frame $ISYS$ and the body fixed frames are defined. For each of the p_r rigid bodies K_i, the corresponding position variables of the center of gravity $\{\mathbf{r}_i, \mathbf{S}_i\}$ and inertia parameters $\{m_i, \mathbf{I}_i\}$ are specified. The p_f flexible bodies can be modeled in a structural analysis software of the user's choice, most often a finite element program. There, e.g., a modal analysis should be performed and the results are stored in the *Standard input data file* (SID-file), see Schwertassek and Wallrapp [12]. This file is a standardized format to store a description of all elastic forces, inertia properties, mode shapes, node positions and all other important information. During the modeling in the finite element program the user has to decide the type of frame of reference for this body. Here only the case of a frame of reference attached to one node of the body shall be investigated, which results in a tangent orientation to the deformed body. In Neweul-M^2 applied forces and torques $\{\mathbf{f}_i^{(a)}, \mathbf{l}_i^{(a)}\}$ can be specified. But they are not defined explicitly by the user but are introduced automatically by the program from the definition of force elements during the modeling. These force elements can be attached to any coordinate system, e.g. a node of an elastic body.

The following steps will be first described for the case of rigid bodies. Then they will be presented for the case of elastic bodies and the differences will be highlighted.

Step 2 (rigid): Element consideration, local equations. First the elements of all vectors and inertia tensors are computed in the inertial frame $ISYS$ by applying appropriate tensor transformations. This is then the only coordinate system further used for the description. The local equations of motion for the center of mass of each rigid body K_i read as

$$m_i \dot{\mathbf{v}}_i = \mathbf{f}_i^{(a)} + \mathbf{f}_i^{(r)} \ , \quad i = 1(1)p_r \ , \tag{1}$$

$$\mathbf{I}_i \cdot \dot{\boldsymbol{\omega}}_i + \tilde{\boldsymbol{\omega}}_i \cdot \mathbf{I}_i \cdot \boldsymbol{\omega}_i = \mathbf{l}_i^{(a)} + \mathbf{l}_i^{(r)} \ , \quad i = 1(1)p_r \ , \tag{2}$$

where $\mathbf{v}_i$ and $\boldsymbol{\omega}_i$ denote the translational and rotational velocity, and the forces and torques are subdivided in the applied forces and torques and the unknown reaction forces and torques $\mathbf{f}_i^{(r)}$, $\mathbf{l}_i^{(r)}$. The reactions are eliminated later, and, therefore, they do not have to be specified.

Step 3 (rigid): Relation between local and global quantities. The relation between the pose (position $\mathbf{r}_i$, rotation matrix $\mathbf{S}_i$) of a single body K_i, and the generalized coordinates, summarized in the vector $\mathbf{y}(t)$, is determined by the constraints. These poses are derived from the input data as

$$\mathbf{r}_i = \mathbf{r}_i(\mathbf{y}, t) \ , \quad \mathbf{S}_i = \mathbf{S}_i(\mathbf{y}, t) \ , \quad i = 1(1)p_r \ , \tag{3}$$

and the corresponding velocities $\mathbf{v}_i$, $\boldsymbol{\omega}_i$ with respect to the body's center of mass are computed as

$$\mathbf{v}_i = \dot{\mathbf{r}}_i = \frac{\partial \mathbf{r}_i}{\partial \mathbf{y}} \cdot \dot{\mathbf{y}} + \frac{\partial \mathbf{r}_i}{\partial t} = \mathbf{J}_{Ti}(\mathbf{y}, t) \cdot \dot{\mathbf{y}} + \overline{\mathbf{v}}_i(\mathbf{y}, t) \ , \tag{4}$$

$$\boldsymbol{\omega}_i = \mathbf{J}_{Ri}(\mathbf{y}, t) \cdot \dot{\mathbf{y}} + \overline{\boldsymbol{\omega}}_i(\mathbf{y}, t) \ . \tag{5}$$

For scleronomic constraints the local velocities $\overline{\mathbf{v}}_i$, $\overline{\boldsymbol{\omega}}_i$ disappear. The $3 \times f$-Jacobian matrices $\mathbf{J}_{Ti}$, $\mathbf{J}_{Ri}$ of translation and rotation, respectively, present the relation between the local and global coordinates. The accelerations can be obtained similarly

$$\mathbf{a}_i = \dot{\mathbf{v}}_i = \mathbf{J}_{Ti}(\mathbf{y}, t) \cdot \ddot{\mathbf{y}} + \overline{\mathbf{a}}_i(\dot{\mathbf{y}}, \mathbf{y}, t) \ , \tag{6}$$

$$\boldsymbol{\alpha}_i = \dot{\boldsymbol{\omega}}_i = \mathbf{J}_{Ri}(\mathbf{y}, t) \cdot \ddot{\mathbf{y}} + \overline{\boldsymbol{\alpha}}_i(\dot{\mathbf{y}}, \mathbf{y}, t) \ . \tag{7}$$

After these preparatory computations the local Newton-Euler equations (1) and (2) for each body K_i are expressed as functions of the generalized coordinates and their derivatives.

Step 4 (rigid): System consideration, global equations. Starting from the equations of motion of the single bodies those of the complete system shall be obtained. For this, the local equations are stacked on each other, which results in a $6p_r$ vector equation. Then e.g. the global vector of generalized applied forces has the form

$$\overline{\mathbf{q}}^{(a)} = \left[\mathbf{f}_1^{(a)\mathrm{T}}, \mathbf{l}_1^{(a)\mathrm{T}}, \dots, \mathbf{f}_{p_r}^{(a)\mathrm{T}}, \mathbf{l}_{p_r}^{(a)\mathrm{T}}\right]^{\mathrm{T}} \ . \tag{8}$$

The global Newton-Euler equations are now represented as

$$\overline{\overline{\mathbf{M}}} \cdot \overline{\mathbf{J}} \cdot \ddot{\mathbf{y}} + \overline{\mathbf{k}} = \overline{\mathbf{q}}^{(a)} + \overline{\mathbf{q}}^{(r)} \ . \tag{9}$$

These $6p_r$ equations are reduced to the minimal number of f ordinary differential equations by pre-multiplication with the $f \times 6p_r$ transposed Jacobian matrix $\overline{\mathbf{J}}^{\mathrm{T}}$,

$$\overline{\mathbf{J}}^{\mathrm{T}} \cdot \overline{\overline{\mathbf{M}}} \cdot \overline{\mathbf{J}} \cdot \ddot{\mathbf{y}} + \overline{\mathbf{J}}^{\mathrm{T}} \cdot \overline{\mathbf{k}} = \overline{\mathbf{J}}^{\mathrm{T}} \cdot \overline{\mathbf{q}}^{(a)} \ , \tag{10}$$

where the term $\overline{\mathbf{J}}^{\mathrm{T}} \cdot \overline{\mathbf{q}}^{(r)}$ is vanishing due to d'Alembert's principle. Summarizing the matrix products, one gets the equations of motion

$$\mathbf{M}(\mathbf{y}, t) \cdot \ddot{\mathbf{y}}(t) + \mathbf{k}(\mathbf{y}, \dot{\mathbf{y}}, t) = \mathbf{q}(\mathbf{y}, \dot{\mathbf{y}}, t) \tag{11}$$

with the symmetric mass matrix $\mathbf{M} = \overline{\mathbf{J}}^{\mathrm{T}} \cdot \overline{\overline{\mathbf{M}}} \cdot \overline{\mathbf{J}}$, the vector of the generalized Coriolis-, centrifugal and gyroscopic forces $\mathbf{k} = \overline{\mathbf{J}}^{\mathrm{T}} \cdot \overline{\mathbf{k}}$ and the vector of the generalized applied forces $\mathbf{q} = \overline{\mathbf{J}}^{\mathrm{T}} \cdot \overline{\mathbf{q}}^{(a)}$.

Step 2 (elastic): Element consideration, local equations. For the frames of reference of each elastic body the velocities and accelerations are set up in the same way as mentioned above. Then the elastic deformations should be added to the expressions. The displacements and its derivatives as well as the expressions defining the inertias are read from the SID-file, which has been created before in a structural analysis software. Therefore, all these expressions are formulated with respect to the frame of reference of this body and described in its coordinate system. To be able to include all values together with forces in one equation, all values have to be expressed in the same coordinate system. Instead of transforming all deformations and forces resulting from the body's description to the inertial system, everything will be described in the frame of reference of each respective body. This requires position vectors, applied forces like gravity and so on to be transformed to the local coordinate system.

Because the frame of reference is located in one node and not in the center of gravity anymore, the translational and rotational equations of motion, see Eqs. (1) and (2), are not decoupled anymore but have to be written like

$$\overline{\overline{\mathbf{M}}}_k \cdot \begin{bmatrix} \mathbf{a}_{abs,ref,k} \\ \boldsymbol{\alpha}_{abs,ref,k} \\ \ddot{\mathbf{y}}_{elastic,k} \end{bmatrix} = \mathbf{h}_{g,k} + \mathbf{h}_{d,k} - \mathbf{h}_{e,k} - \mathbf{h}_{\omega,k} + \begin{bmatrix} \mathbf{f}_k^{(r)} \\ \mathbf{l}_k^{(r)} \\ \mathbf{0} \end{bmatrix} \ . \tag{12}$$

This equation contains the mass matrix $\overline{\overline{\mathbf{M}}}_k$, the vectors of generalized gravitational $\mathbf{h}_{g,k}$, surface $\mathbf{h}_{d,k}$, elastic $\mathbf{h}_{e,k}$ and volume forces $\mathbf{h}_{\omega,k}$. The volume forces, called $\mathbf{h}_{\omega,k}$, result from the fact that the equations of motion are not set up with respect to the center of gravity but with respect to the frame of reference. The reaction forces $\mathbf{f}_k^{(r)}$ and torques $\mathbf{l}_k^{(r)}$ vanish again after the multiplication with the global Jacobian matrix as explained for the rigid bodies.

Step 3 (elastic): Relation between local and global quantities. The absolute position vector of node i on elastic body k $\mathbf{r}_{abs,k,i}$ can be composed from the large nonlinear motion of the frame of reference $\mathbf{r}_{abs,ref,k}$, the relative nodal position in the undeformed configuration $\mathbf{r}_{const,k,i}$ and the elastic displacement $\mathbf{u}_{k,i}$ as

$$\mathbf{r}_{abs,k,i} = \mathbf{r}_{abs,ref,k} + \mathbf{r}_{const,k,i} + \mathbf{u}_{k,i} \ . \tag{13}$$

To approximate the elastic displacement a Ritz-approach is used to distinguish between the time- and position-depedent part

$$\mathbf{u}_{k,i}(\mathbf{R},t) = \mathbf{\Phi}_{k,i}(\mathbf{R}) \cdot \mathbf{y}_{elastic,k}(t)\ . \tag{14}$$

Here, the matrix $\mathbf{\Phi}_{k,i}(\mathbf{R})$ represents the matrix of the Ansatz-functions, whereas the vector $\mathbf{y}_{elastic,k}(t)$ represents the elastic degrees of freedom of this body. All elastic coordinates can be combined to form the global vector of elastic generalized coordinates $\mathbf{y}_{elastic}$. Therefore, the vector of the generalized coordinates of the complete system has to be extended to

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_{rigid}^T & \mathbf{y}_{elastic}^T \end{bmatrix}^T\ . \tag{15}$$

After introducing a similar matrix of Ansatz-functions for the rotations of each node, the velocities and accelerations can be obtained as time derivatives. Similar to the rigid bodies, the translational and rotational accelerations can be written to contain the Jacobian matrix $\mathbf{J}_k$ of body k. As Eq. (12) contains all motions and deformations coupled to each other it is reasonable to also stack the Jacobian matrices of translation, rotation and elastic deformations

$$\mathbf{J}_k = \begin{bmatrix} \mathbf{J}_{T,k,ref}^T & \mathbf{J}_{R,k,ref}^T & \mathbf{J}_{E,k,ref}^T \end{bmatrix}^T\ . \tag{16}$$

Step 4 (elastic): System consideration, global equations. For rigid bodies the equations of all bodies have been stacked in order to gain one set of equations for the complete multibody system. This and the multiplications with the transposed global Jacobian matrix $\overline{\mathbf{J}}^T$, see Eq. (10), can also be performed to get the equations of motion in minimal form, see Eq. (11),

$$\mathbf{M}(\mathbf{y},t) \cdot \ddot{\mathbf{y}}(t) + \mathbf{k}(\mathbf{y},\dot{\mathbf{y}},t) = \mathbf{q}(\mathbf{y},\dot{\mathbf{y}},t)\ . \tag{17}$$

The force vectors of elastic $\mathbf{h}_{e,k}$ and volume forces $\mathbf{h}_{\omega,k}$ as well as inertia forces resulting from the local accelerations, compare Eq. (6) and (7), are sorted in the $\mathbf{k}$ vector. This leaves all applied forces for the $\mathbf{q}$ vector.

The greatest formal difference in the presented steps for rigid and elastic bodies is the coordinate system used to represent the vectors and matrices. For rigid bodies the inertial system is used, whereas all expressions for elastic bodies are described in their respective frame of reference. At first it sounds like a strange concept to use different describing coordinate systems, but due to the stacking of the equations for each body, this does not cause any problems but results in the lowest number of transformations.

The presented four steps show that this symbolical formalism is based on the Newton-Euler equations. However, they are supplemented by typical features of the analytical approach like generalized coordinates. All necessary computations were performed symbolically in Neweul-M^2 using Maple which is called via Matlab's *Symbolic Math Toolbox*.

3. Software Tools

The program Neweul-M^2 is running in the Matlab environment and using Maple for the symbolic manipulations. Because these two powerful software packages are used for the implementation, combining the advantages of symbolical and numerical approaches, the program is called *Neweul-M^2*.

Matlab offers a wide range of efficient tools for numeric simulations. It also contains the Simulink environment used for signal flow simulations and control. This makes it one of the most widely used scientific and engineering softwares. To be able to calculate the governing equations of the multibody system symbolically, a connection to a computer algebra software is required. Here, Maple is used for the symbolic manipulations. For Neweul-M^2 the interface by The Mathworks, called *Symbolic Math Toolbox* is used, which internally connects to a Maple kernel, although the user does hardly notice that he uses not 'pure' Matlab. One advantage of keeping the user in the Matlab environment is the recent change of the *Symbolic Math Toolbox* to use MuPad instead of Maple as the computer algebra system which requires hardly any changes in the program.

Another goal of Neweul-M^2 is to enable the user to use the models and results for other simulations and applications. Due to their complexity, the simulation of mechanical multibody systems, in general, has to be performed numerically, where Matlab is the tool at hand. All kinematic values, like position or acceleration vectors, are calculated fully symbolic. All applied forces, equations of motion and other necessary expressions are formulated fully symbolic as well. When these calculations are finished, the results are stored in a Matlab data structure for further symbolic manipulations. For the numerical evaluation of the equations of motion and all kinematic quantities like position, velocities and acceleration, automatically created source code in Matlab language is provided. Moreover, the equations of motion can be exported as C-code to be used, e.g., as a Simulink S-function, or directly on dSpace hardware for realtime applications. For the evaluation of a numerical value then only these automatically created functions are called. This dual system of having all expressions available symbolically and numerically is a key feature of Neweul-M^2 and allows both symbolic manipulations and fast evaluations of numerical values.

In Matlab a tool to create graphical user interfaces is included. It offers the possibility to easily design interfaces with programmable control elements, from which arbitrary Matlab functions can be called. Using this, a graphical user interface has been created acting as a front end to the routines of Neweul-M^2. As Neweul-M^2 uses only standard software, which can be installed on different operating systems, e.g. Windows or Linux, it is platform independent.

4. Multibody Dynamics by Neweul-M^2

There are two ways in Neweul-M^2 to model systems and perform simulations, using the command-based mode or the graphical user interface, respectively. The change between the two methods can be easily achieved by saving the model data structure created with one interface and opening it with the other interface. To demonstrate the modeling process step by step, a simple double pendulum consisting of rigid bodies will be considered first, see Fig. 1. In the next step the two rigid bodies will be replaced by flexible ones.

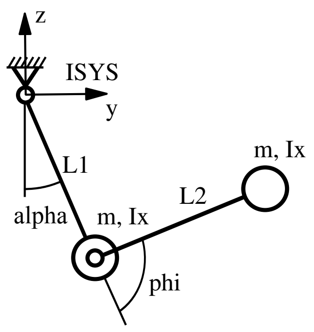


Figure 1. Sketch of the double pendulum model.

4.1. Modeling rigid bodies with commands

In this part, the modeling of a multibody system using modeling commands is described. These modeling commands can be entered directly into the command prompt of Matlab, or can be collected in an input file. A detailed documentation is maintained as a wiki, see Kurz and Henninger [6], and is available in a browser or as an export to a pdf-file.

To define the model of a rigid double pendulum, the following commands are necessary and will be explained in the following.

```
newSys('Id','my_MBS','Name','Double Pendulum');
newUserVarKonst('m',2,'L1',1,'L2',1,'Ix',0.01);
newGenCoord('alpha','phi');
newBody('Id','P1','Name','Pendulum 1','RefSys','ISYS', ...
        'RelPos','[0; 0; 0]','RelRot','[alpha; 0; 0]', ...
        'CgPos','[0; 0; -L1]','CgRot','[0; 0; 0]', ...
        'Mass','m','Inertia','[Ix,0,0; 0,1,0; 0,0,1]');
newBody('Id','P2','Name','Pendulum 2','RefSys','P1_cg', ...
        'RelPos','[0; 0; 0]','RelRot','[phi; 0; 0]', ...
        'CgPos','[0; 0; -L2]','CgRot','[0; 0; 0]', ...
        'Mass','m','Inertia','[Ix,0,0; 0,1,0; 0,0,1]');
calcEqMotNonLin;
writeMbsNonLin;
```

Three points at the end of a line are used, when the command is continued in the next line. When starting the modeling process in Neweul-M^2, a new mechanical system has to be created first by

```
newSys('Id','my_MBS','Name','Double Pendulum');
```

In most cases the parameters of the modeling commands are given pairwise in the Matlab-usual manner. The first parameter of the pair defines the parameter type, the second is the parameter itself. By this convention the syntax is human-readable and comprehensible, and the parameters can be specified in an arbitrary order. For all parameters not specified, default values are taken. Here, we just created a new data structure for a multibody system with the identifier `my_MBS` and the name `Double Pendulum`. For all modeling elements, substructures are entered in this system data structure, where the respective identifier is used as a fieldname.

As the modeling is done symbolically, four types of parameters are available, constants, time- and state-dependent parameters and generalized coordinates. The generalized coordinates are the unknowns in the equations of motion. When creating a time dependent parameter, a function template to evaluate its value and one for each of the first two derivatives with respect to time are created. This allows the user to implement any suitable function for these values. The state dependent parameters can be used in force elements or for the definition of coordinate systems. For the state dependent parameters such function templates are created as well which then should contain some additional information on its derivatives, e.g. for the calculation of Jacobian matrices.

In our example, we first define the constant parameters. These are the pendulum lengths `L1`, `L2`, the mass `m` and the moment of inertia `Ix`.

```
newUserVarKonst('m',2,'L1',1,'L2',1,'Ix',0.01);
```

The numerical values are necessary in later steps, but can be set here already for simplicity. To define a double pendulum, we then have to specify the generalized coordinates with

```
newGenCoord('alpha','phi');
```

The program Neweul-M^2 offers all common modeling elements of multibody systems. The most basic ones are frames and bodies. When defining a new multibody system, the inertial system $ISYS$ is created automatically. The position of each frame is described with respect to an already existing frame, denoted as reference system. Each rigid body is defined by at least two body-fixed frames, which are created automatically. One of them is called the primary system of this body and is used to describe the motion of the body, the other system defines the location of the center of gravity, denoted by `_cg`, and provides the coordinate system for the description of the inertia tensor. Flexible bodies are described by their frame of reference and the nodes. To describe the first pendulum some informations are necessary:

- name and id to identify the body,
- position vector and orientation of the primary frame, often located in the support,

- relative position vector to the frame in the center of gravity and its relative orientation, and
- mass and inertia properties.

As presented above, in Neweul-M^2 these definitions read as

```
newBody('Id','P1','Name','Pendulum 1','RefSys','ISYS', ...
        'RelPos','[0; 0; 0]','RelRot','[alpha; 0; 0]', ...
        'CgPos','[0; 0; -L1]','CgRot','[0; 0; 0]', ...
        'Mass','m','Inertia',[Ix,0,0; 0,1,0; 0,0,1]);
```

The way a mechanical system is modeled in Neweul-M^2 is different to most commercial programs due to the symbolic approach. First one has to declare variables of different types. Then the definition of bodies and frames is achieved by the symbolic definition of the position vectors and rotation angles. The generalized coordinates $\mathbf{y}$ specify the degrees of freedom, see Eq. (3). A list of joints or constraints is not required. The reason for this is the way the system is stored and the equations of motion are generated.

At this point in the simulation process, the user can specify other modeling elements, like force elements or can close kinematic loops. The command syntax is of the same style as when creating a new body. There are different types of force elements available, spring-damper combinations as well as time dependent force excitations. If kinematic loops are defined, the algebraic loop closing conditions are calculated and used as algebraic constraints for the time integration. The actual modeling is finished, if the user has entered all necessary information to create the symbolic expressions, which can be done with

```
calcEqMotNonLin;
```

For all kinematic values and the equations of motion, the files containing source code for their numerical evaluation are created by

```
writeMbsNonLin;
```

These files can then be used in external simulation programs or, as described in the following, within Matlab. An animation window can be initialized to display the positions of all frames, because numerical values for the constants have been defined already

```
createAnimationWindow;
```

To set a certain initial time (t=0) and state (alpha=0.4, phi=1.5) for the positions in the animation window the following command can be used

```
updateGeo(0,[0.4;1.5]);
```

Simple geometrical shapes, e.g. spheres, cuboids or rotational solids can be attached to the frames, see Legland [7]. In an animation these shapes move in the same way their governing frames do, allowing representations of bodies, see

Fig. 2. For more complicated shapes, it is also possible to import the geometry by STL files, created e.g. in CAD applications. These shapes are only for the animation, e.g. a calculation of moments of inertia is not included. Furthermore, line elements connecting two frames or displaying the trajectory of a frame, e.g. from a time integration, are available.

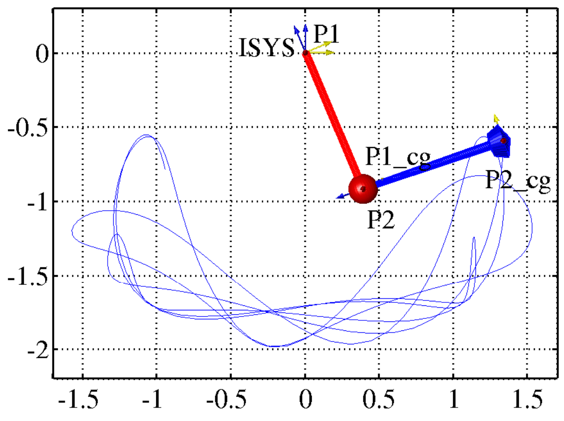


Figure 2. Model of a double pendulum including geometrical shapes of bodies and a trajectory.

Shapes to represent bodies can be easily created, see Fig. 2. The vectors and matrices of the equations of motion, compare Eq. (11), are stored in the data structure under `sys.eqm`. For this example e.g. the mass matrix can be shown using Matlab's 'display()' command

```
display(sys.eqm.M)
 =
[   (2*L1^2+L2^2+2*L1*L2*cos(phi))*m+2*Ix, ...
               m*L2^2+m*L1*L2*cos(phi)+Ix]
[              m*L2^2+m*L1*L2*cos(phi)+Ix, ...
                                 m*L2^2+Ix]
```

Derivatives with respect to time of any variable are denoted in the symbols of Maple by a leading `D` or `D2`, respectively, for the first or second derivative. Using the equations of motion, several methods for simulation and system analysis are available in Neweul-M^2. Some of them are listed in the following. A time integration can be easily performed after setting the initial conditions and the time interval.

```
mytime = [0,10];  % Simulation time interval
y0   = [0.4;1.5];  % Initial values for positions
```

```
Dy0 = [0;0];       % Initial values for velocities

timeInt(y0,Dy0,'time',mytime); % Numerical time integration
fitCanvas;                     % Adjust visible area
plot_trajectories('P2_cg');    % Plot trajectory of a frame
animTimeInt;                   % Start an animation
```

The percent sign '%' in Matlab is used to denote that the rest of this line contains comments and explanations. Even though Neweul-M^2 offers many settings and possibilities to specify optional values, the user can just use the default values at first. Additional options and specifications can be passed in the pair-wise manner explained at the definition of new bodies. In order to investigate the corresponding linear model for small angles only, the equations of motion can be linearized by

```
calcEqMotLin; % Linearize the equations of motion
writeMbsLin;  % Create files for numerical evaluation
```

For the linearization, the expressions in the equations of motion are approximated by a Taylor-series expansion where all second and higher order terms in the generalized coordinates are neglected. Then the mass, damping and stiffness matrices are obtained by appropriate sorting. As a default symbolical constants are used for the set values of the linearization, but time dependent functions are available as well. To perform a modal analysis at evaluation time `t=0` type

```
modalAnalysis(0); % Calculate eigenvectors /-values
showModeShape;    % Display mode shapes
```

Having the symbolic equations of motion, some tasks require very little work by the user. So it takes only the two commands `calcEqMotLin` and `writeMbsLin` to calculate the linearized equations of motion, once the model is described.

4.2. Modeling elastic bodies with commands

In the previous part it was shown how to model a rigid body double pendulum with commands. Now the bodies shall be modelled considering their elasticity. Therefore, it is necessary to model them, e.g. using an FEM program. From this an SID file, compare Schwertassek and Wallrapp [12], containing all information of the elastic body can be created, let's say this file is called `nodalfixed_yz_4n_3m.SID_FEM`.

The structure modeled here is a beam extending from the origin of the coordinate system $1\,m$ in negative z-direction. Before creating the SID file it is possible to select which nodes and which eigenmodes shall be included in the file. Therefore, the user can use a higher number of elements for the modal analysis than necessary for an animation later in Neweul-M^2, also a modal reduction may be performed at this step. For this example 10 elements have been used for the modal analysis, whereas only 4 nodes and 3 mode shapes have been exported to the SID file. Because all properties of the elastic bodies are stored as numerical data, the approach used is not fully symbolic anymore. It would be possible but is not useful to introduce a symbolic parameter for each value from the SID data.

The first commands are known from above to create a new multibody system and define the generalized coordinates, as described before

```
newSys('Id','my_MBS','Name','Flexible Double Pendulum');
newGenCoord('alpha','phi');
```

The definition of the bodies changes when using flexible bodies. The position of the center of gravity, all inertia properties, elastic degrees of freedom and the positions of all nodes are stored in the SID file. In the structural analysis program, the nodes can be given indices, which are included in the SID file. Then the nodal coordinate systems are called by the `Id` of the elastic body with the trailing nodal index, e.g. `P1_2`. Here the nodes at each end of the beam have the indices 1 and 2, where nodal frame of node 1 is used as the frame of reference for the body.

```
newBody('Id','P1','Name','Pendulum 1','RefSys','ISYS', ...
    'RelPos','[0; 0; 0]','RelRot','[alpha; 0; 0]', ...
    'SIDfile','nodalfixed_yz_4n_3m.SID_FEM');
newBody('Id','P2','Name','Pendulum 2','RefSys','P1_2', ...
    'RelPos','[0; 0; 0]','RelRot','[phi; 0; 0]', ...
    'SIDfile','nodalfixed_yz_4n_3m.SID_FEM');
```

The elastic degrees of freedom are generated automatically when read from the SID file and called here `q01` up to `q06`. For better readability the vector of generalized coordinates is sorted to have the generalized coordinates of rigid body motions on top of the elastic degrees of freedom, as in Eq. (15). At this point of the simulation the user can call the same commands as above, noting only that e.g. there is no coordinate system in the center of gravity available and the number of degrees of freedom has changed. To include graphical representations for flexible bodies the most useful representation is a line connecting all coordinate systems, see Fig. 3.

The presented double pendulum has been compared to simulations performed by Simpack. Very good accordance between both results has been observed.

4.3. Modeling with the graphical user interface

A graphical user interface available for Neweul-M^2 was created with the tools provided by Matlab. It offers a different way of accessing the modeling commands, but does not offer additional capabilities.

As only feasible choices are offered in the menus, the possibility of input errors is reduced. Also, this kind of modeling offers more guidance, e.g. by help buttons in each window, see Fig. 4, where the properties of the rigid body `P2` of our example are shown. The options at the lower left corner concerning a so called Joint System are in an experimental state.

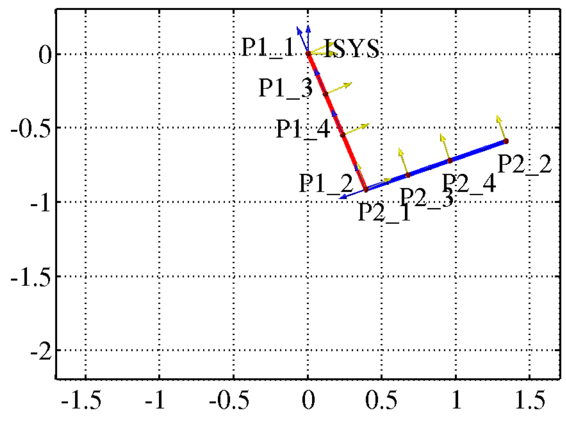


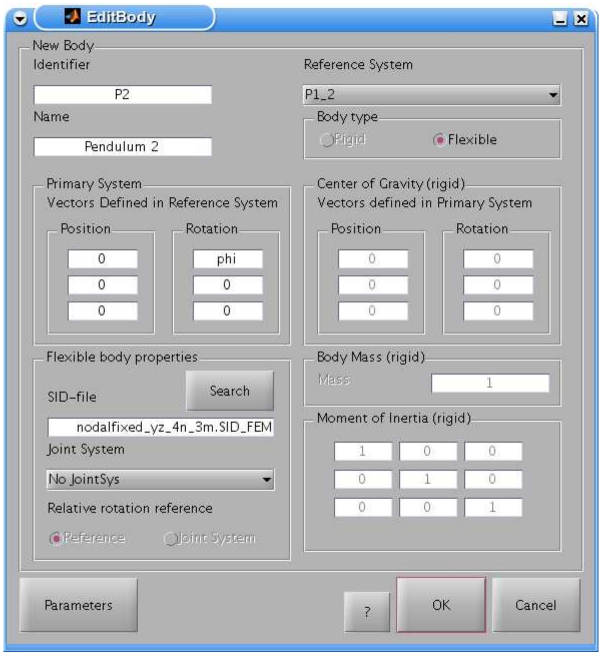
Figure 3. Model of an elastic double pendulum including lines representing the bodies.

4.4. Some features of the program Neweul-M^2

As all expressions are derived fully symbolic and are stored as such and in files for numerical evaluation, they offer a wide range of uses like the system analysis options of Matlab. The equations of motion can, e.g., be numerically integrated to animate the motion. By prescribing given functions to generalized coordinates, a kinematic analysis is easily possible. This is especially important to investigate reachable work spaces or special configurations. For any given set of values for the generalized coordinates and their derivatives, the prescribed functions can be used to calculate kinematic values like positions, orientations, or accelerations.

The nonlinear equations of motion can be linearized for symbolic set values. This enables, e.g., a linear analysis of the small motion around a prescribed trajectory. After the definition of system input and output variables, transfer functions can be determined. In the context of system theory with input and output variables, the possibility to export the equations of motion to Simulink is another interesting feature. Both, the linear and the nonlinear equations of motion can be exported to C-code that can be used as S-functions in Simulink. The linearized system can also be used to perform a modal analysis. For the linearized system, the state space formulation is also available, providing the ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$)-matrices, as required in control engineering.

For parameter optimization of design variables the symbolic expressions offer even more advantages. To improve the performance of deterministic optimization algorithms, the calculation of gradients is crucial. Especially for multibody systems, which show usually a highly nonlinear behavior, the fast calculation of precise gradients is very important. The symbolic expressions can be used to apply semi-analytical methods, namely the direct method and the adjoint variable method, see Bestle [1], Eberhard [2] and Kurz [5]. These methods use


EditBody

New Body

Identifier: P2

Name: Pendulum 2

Reference System: P1_2

Body type: Rigid [✓ Flexible]

Primary System – Vectors Defined in Reference System

Position	Rotation
0	phi
0	0
0	0

Center of Gravity (rigid) – Vectors defined in Primary System

Position	Rotation
0	0
0	0
0	0

Flexible body properties

SID-file: nodalfixed_yz_4n_3m.SID_FEM

Joint System: No JointSys

Relative rotation reference: [✓ Reference] Joint System

Body Mass (rigid) – Mass: 1

Moment of Inertia (rigid)

1	0	0
0	1	0
0	0	1

Figure 4. Graphical user interface to define a body, here editing the flexible body 'Pendulum 2'.

symbolic expressions wherever possible, only when the calculation depends on the results of the time integration of the equations of motion, a numerical solution is used. They are often not only faster than numerical methods, like the finite difference method, their precision is much higher as well. For simple problems, where an analytical calculation of the gradient is possible, these methods reach precisions in the same order as the corresponding numerical time integration. These gradients can also be calculated directly in Neweul-M^2 for a sensitivity analysis. Apart from the standard modeling presented, Neweul-M^2 offers a broad spectrum of analysis and synthesis tools based on Matlab.

5. Conclusions

The symbolical multibody software Neweul-M^2 offers a convenient description of the dynamical properties of a multibody system. The equations of motion in minimal form are obtained by considering the constraints in the assembled system depending on the generalized coordinates. The system modeling may be performed from the command line and/or using a graphical user interface. Using the Maple based Symbolic Math Toolbox of Matlab, the symbolical equations of motion are obtained as well as the corresponding files for numerical computations using the vast mathematical features of Matlab. In particular the equations of motion may be linearized and used in Simulink or for control design within the Matlab environment. Some features are presented for both a rigid body and an elastic double pendulum. The implementation of flexible bodies in a multibody system combining symbolic and numeric approaches will be used for further research work in the areas of optimization and model reduction.

Acknowledgments

We want to thank Dr.-Ing. Christoph Henninger who started this project and made very valuable contributions. Our thank also goes to Prof. Werner Schiehlen for his advice and the development of the preceding version of Neweul, which provided the basis for this program. This research was partially done within the framework of SimTech, the Stuttgart Research Centre for Simulation Technology. All this support is gratefully appreciated.

References

[1] Bestle, D.: *Analyse und Optimierung von Mehrkörpersystemen.* [in German], Springer, Berlin, 1994

[2] Eberhard, P.: Zur Optimierung von Mehrkörpersystemen. [in German], Dissertation, *VDI Fortschritt-Berichte*, Reihe 11, Nr. 227. VDI Verlag, Düsseldorf, 1996

[3] Kane, T.R. and Levinson, D.A.: *Dynamics: Theory and Applications.* McGraw-Hill, New York, 1985

[4] Kreuzer, E.: Symbolische Berechnung der Bewegungsgleichungen von Mehrkörpersystemen. [in German], Dissertation, *VDI Fortschritt-Berichte*, Reihe 11, Nr. 32, VDI-Verlag, Düsseldorf, 1979

[5] Kurz, T.: *Entwicklung eines Optimierungsmoduls mit Sensitivitätsanalyse für die symbolische Mehrkörpersimulationsumgebung SYMBS.* [in German], Diplomarbeit DIPL-122. Institute of Engineering and Computational Mechanics, University of Stuttgart, 2007

[6] Kurz, T. and Henninger, C.: *Program documentation to Neweul-M^2.* Institute of Engineering and Computational Mechanics, University of Stuttgart, 2009

[7] Legland, D.: Graphics Library geom3d. http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=8002 &objectType=file, 2005

[8] Lu, X.: A Lie Group Formulation of Kane's equations for multibody systems. *Multibody System Dynamics* 20, 29-50, 2008

[9] Popp, K. and Schiehlen, W.: *Fahrzeugdynamik.* [in German], B.G. Teubner, Stuttgart, 1993

[10] Schiehlen, W. (ed.): *Multibody Systems Handbook.* Springer, Berlin, 1990

[11] Schiehlen, W. and Eberhard, P.: *Technische Dynamik.* [in German], B.G. Teubner, Wiesbaden, 2004

[12] Schwertassek, R. and Wallrapp, O.: *Dynamik flexibler Mehrkörpersysteme.* [in German], Friedr. Vieweg & Sohn, Braunschweig, 1999