Analysing Dynamical Phenomenons: Introduction to MBSim

Thorsten Schindler*, **Martin Förg***, **Markus Friedrich***, **Markus Schneider***, **Bastian Esefeld***, **Robert Huber***, **Roland Zander**† [1], **Heinz Ulbrich***

* Institute of Applied Mechanics
Technische Universität München
Boltzmannstraße 15, 85748 Garching, Germany
e-mail: [schindler, foerg, friedma, schneidm, esefeld, huber, ulbrich]@amm.mw.tum.de

† SIMPACK AG
Friedrichshafener Strasse 1, 82205 Gilching, Germany
e-mail: roland.zander@simpack.de

ABSTRACT

The current paper gives an overview of the simulation tool *MBSim* (http://mbsim.berlios.de), which is licenced under the GNU Lesser General Public License. Point of departure is the formulation of the mathematical equations of oscillatory dynamical systems with uni- and bilateral frictional constraints and impacts. The algorithmic treatment with event-detecting and timestepping integration is the basis for an efficient solution strategy. From a software architecture point of view, the components and the connections in-between are classes in C++. The object-oriented design and the corresponding modelling possibilities with XML-input schemes as well as the external visualisation tool OpenMBV (http://openmbv.berlios.de) show a modular structure for universal usage. Altogether, the simulation program and the computable results allow the detailed analysis of complex dynamical systems.

Keywords: nonsmooth multibody systems, time integration, C++ program development.

1 NONSMOOTH MULTIBODY SYSTEMS

Nonsmooth multibody systems are special mechanical systems basically including rigid bodies and in space discretised deformable bodies [27] in a hybrid way. They are additionally characterised by rigid unilateral and bilateral contacts as well as impacts, which lead to discrete jumps within the system's velocities. So the degree of freedom is not a constant function, but changes during the simulation process and determines a time-variant topology. A unitary mathematical and numerical formulation based on measure differential equations (MDE) with constraints has been processed in the last decades at different research institutes, summarised e.g. in [20, 26, 23, 6, 14, 1]. It allows for the efficient integration even of industrial systems with large numbers of transitions [18] and avoids both high artificial stiffnesses and additional modelling errors due to regularised interactions.

1.1 Measure differential equation

A measure differential equation [16]

$$\boldsymbol{M}\boldsymbol{\mu}^{\boldsymbol{u}} = \boldsymbol{\mu}^{\boldsymbol{G}} + \sum_k \boldsymbol{\mu}^{\boldsymbol{H}_k} \tag{1}$$

involves measures $\boldsymbol{\mu}$ representing the velocity by superscript $\boldsymbol{u}$, integrable forces by superscript $\boldsymbol{G}$ and impacts at countable points in time t_k, $k \in \mathbb{N}$, by superscript for Heaviside functions $\boldsymbol{H}_k$. The symmetric and positive definite mass matrix $\boldsymbol{M}$ depends on the position $\boldsymbol{q}$ of the system.

[1] The contributions of Roland Zander originate from his work at the Institute of Applied Mechanics.

Equivalent to the MDE (1) it is also possible to distinguish between smooth –non-impulsive– and impact –impulsive– dynamics. Then, the equations of motion satisfy

$$\boldsymbol{M}\dot{\boldsymbol{u}} = \boldsymbol{h} + \boldsymbol{W}\boldsymbol{\lambda}\,, \tag{2}$$

$$\boldsymbol{M}_k\left(\boldsymbol{u}_k^+ - \boldsymbol{u}_k^-\right) = \boldsymbol{W}_k\boldsymbol{\Lambda}_k \quad \forall k \in \mathbb{N} \tag{3}$$

using $\dot{\boldsymbol{u}}$ for denoting the weak time derivative of $\boldsymbol{u}$ and $\boldsymbol{u}_k^+$ as well as $\boldsymbol{u}_k^-$ for describing the velocity after and before an impact time t_k. The generalised velocities depend on the positions via the linear equation $\dot{\boldsymbol{q}} = \boldsymbol{Y}\,\boldsymbol{u}$ with $\boldsymbol{Y} = \boldsymbol{Y}(\boldsymbol{q})$ and the vector $\boldsymbol{h}$ contains all smooth external, internal and gyroscopic forces. It is a function of $\boldsymbol{q}$, $\boldsymbol{u}$ and explicitly of the time t and also holds reactions of single-valued contacts e.g. flexible ones. The directions of set-valued contact reactions are summarised in the wrench matrix $\boldsymbol{W}(\boldsymbol{q})$ as well as $\boldsymbol{\lambda}$ and $\boldsymbol{\Lambda}_k$ refer to smooth and nonsmooth contact reaction values due to persisting contacts as well as discrete impulses, respectively.

1.2 Contact laws

The computation of the accelerations $\dot{\boldsymbol{u}}$ in (2) and the post-impact velocities $\boldsymbol{u}_k^+$ in (3) requires the knowledge of the unknown contact reactions $\boldsymbol{\lambda}$ and $\boldsymbol{\Lambda}_k$ governed by set-valued contact laws $(\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\Lambda}_k, t) \in \mathcal{N}$.

First of all, only smooth motion is considered which means that no impacts occur. Then, a bilateral contact, which is always closed, implies a bilateral constraint of the form

$$g_B = 0, \quad \lambda_B \lesseqgtr 0\,, \tag{4}$$

where g_B denotes the normal distance of the interacting bodies in the contact point. The second type of contact also allows for detachment. The associated unilateral constraint is given by the SIGNORINI-FICHERA-condition

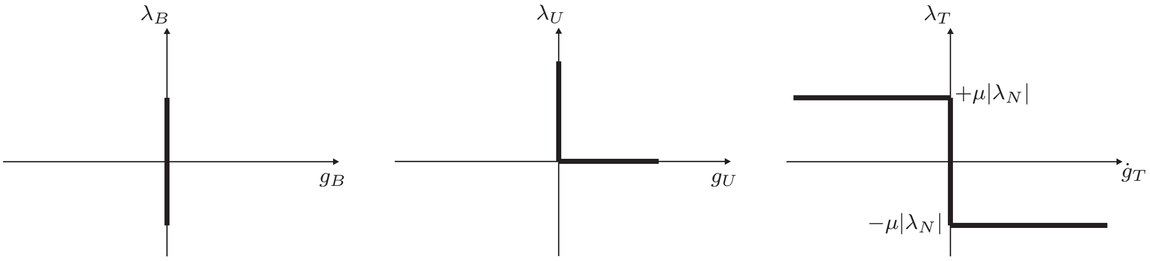
$$g_U \geq 0, \quad \lambda_U \geq 0, \quad g_U\lambda_U = 0 \tag{5}$$

with the normal distance g_U. For both bi- and unilateral constraints e.g. dry friction can be considered. In order to establish COULOMB's law, the force of a single contact is decomposed in a component $\lambda_N \in \{\lambda_B, \lambda_U\}$ normal to the contact plane and – in case of three dimensional dynamics – tangential components $\boldsymbol{\lambda}_T$ in friction direction. Using the relative tangential velocity $\dot{\boldsymbol{g}}_T$ and the friction coefficient μ, COULOMB's friction law is given by

$$\dot{\boldsymbol{g}}_T = \boldsymbol{0} \quad \Rightarrow \quad |\boldsymbol{\lambda}_T| \leq \mu|\lambda_N|\,, \tag{6a}$$

$$\dot{\boldsymbol{g}}_T \neq \boldsymbol{0} \quad \Rightarrow \quad \boldsymbol{\lambda}_T = -\frac{\dot{\boldsymbol{g}}_T}{|\dot{\boldsymbol{g}}_T|}\mu|\lambda_N|\,. \tag{6b}$$

The respective force laws are shown in Fig. 1.



(a) Bilateral constraint. (b) Unilateral constraint. (c) Planar COULOMB friction.

Figure 1. Force laws for bi- and unilateral contacts and friction.

An impact influences all contacts between bodies concerning the post impact velocity. So, the impact laws have to be formulated on velocity level, substituting g by $\dot{g}^+$ and λ by Λ in (4) and (5), subject to the condition that the affected contact is closed. In this context, it is even possible to define special impact laws by replacing $\dot{g}^+$ with adequate physical approximations to regard for example elastic impact behaviour [16].

The formulations of the contact and impact laws (4) to (6) are self-evident from the mechanical point of view [20, 12] but not suitable for the numerical computation. A more appropriate formulation can be attained using convex analysis. With the proximal point to a convex set $C \subset \mathbb{R}^n$, $n \in \mathbb{N}$,

$$\mathbf{prox}_C(\boldsymbol{x}) = \arg \min_{\boldsymbol{x}^* \in C} |\boldsymbol{x} - \boldsymbol{x}^*|, \quad \boldsymbol{x} \in \mathbb{R}^n \tag{7}$$

the relations (4) to (6) have the form [3]

$$\lambda_B = \mathrm{prox}_{C_B}(\lambda_B - r\, g_B)\,, \qquad \Lambda_B = \mathrm{prox}_{C_B}(\Lambda_B - r\, \dot{g}_B^+)\,, \tag{8a}$$

$$\lambda_U = \mathrm{prox}_{C_U}(\lambda_U - r\, g_U)\,, \qquad \Lambda_U = \mathrm{prox}_{C_U}(\Lambda_U - r\, \dot{g}_U^+)\,, \tag{8b}$$

$$\boldsymbol{\lambda}_T = \mathbf{prox}_{C_T(\lambda_N)}(\boldsymbol{\lambda}_T - r\, \dot{\boldsymbol{g}}_T)\,, \qquad \boldsymbol{\Lambda}_T = \mathbf{prox}_{C_T(\Lambda_N)}(\boldsymbol{\Lambda}_T - r\, \dot{\boldsymbol{g}}_T^+)\,. \tag{8c}$$

The corresponding convex sets are specified by:

$$C_B = \mathbb{R}\,, \quad C_U = \{x \in \mathbb{R}\ ;\ x \geq 0\}\,, \quad C_T(y) = \{\boldsymbol{x} \in \mathbb{R}^2\ ;\ |\boldsymbol{x}| \leq \mu|y|\} \tag{9}$$

with $y \in \mathbb{R}$. The independent auxiliary parameter for each contact $r > 0$ is arbitrary from the mathematical but not from the numerical point of view. The optimal choice of r with respect to numerical efficiency and stability of the fixed-point or root-finding solution scheme is discussed in [11].

1.3 Contour description

With the description outlined above, a mechanical system is divided in the motion of bodies and in the interaction between bodies. Only missing is the calculation of the wrench matrix $\boldsymbol{W}$, the gaps g and relative velocities $\dot{g}$. This is done body-per-body by assigning a contour characterised by a position vector $\boldsymbol{r} = \boldsymbol{r}(\boldsymbol{q}, \boldsymbol{s})$, the outward pointing contour normal $\boldsymbol{n} = \boldsymbol{n}(\boldsymbol{q}, \boldsymbol{s})$ and the associated tangents $\boldsymbol{T} = (\boldsymbol{t}_1(\boldsymbol{q}, \boldsymbol{s}), \boldsymbol{t}_2(\boldsymbol{q}, \boldsymbol{s}))$ all depending on the generalised position $\boldsymbol{q}$ of the associated body and on the contour parameters $\boldsymbol{s}$ [27]. Then assuming unique point-to-point contacts, the contact parameters $\boldsymbol{s}_{c_1}$ and $\boldsymbol{s}_{c_2}$ for two contacting bodies necessarily fulfill

$$\boldsymbol{T}_1^T(\boldsymbol{s}_{c_1})\left[\boldsymbol{r}_2(\boldsymbol{s}_{c_2}) - \boldsymbol{r}_1(\boldsymbol{s}_{c_1})\right] = \boldsymbol{0}\,, \tag{10}$$

$$\boldsymbol{T}_2^T(\boldsymbol{s}_{c_2})\left[\boldsymbol{r}_2(\boldsymbol{s}_{c_2}) - \boldsymbol{r}_1(\boldsymbol{s}_{c_1})\right] = \boldsymbol{0}\,. \tag{11}$$

Depending on the structure of these equations either analytical – for geometric primitive contour pairings – or numerical e.g. NEWTON methods have to be applied to get a set of potential contact parameters. Selecting the solution with minimal normal distance

$$g_N = \boldsymbol{n}_1^T(\boldsymbol{s}_{c_1})\left[\boldsymbol{r}_2(\boldsymbol{s}_{c_2}) - \boldsymbol{r}_1(\boldsymbol{s}_{c_1})\right] \tag{12}$$

allows calculating the relative normal and tangential velocities by projection of the relative velocity on the corresponding matrices $\boldsymbol{n}$ and $\boldsymbol{T}$ of the bodies:

$$\begin{pmatrix} \dot{g}_N \\ \dot{g}_{T_1} \\ \dot{g}_{T_2} \end{pmatrix} = (\boldsymbol{n}\ ,\ \boldsymbol{T})^T \left[\boldsymbol{v}_2(\boldsymbol{s}_{c_2}) - \boldsymbol{v}_1(\boldsymbol{s}_{c_1})\right]\ . \tag{13}$$

Each body's portion of the wrench matrix $\boldsymbol{W}$ is the projection of the Cartesian directions $\boldsymbol{n}$ and $\boldsymbol{T}$ of contact reactions in the space of generalised velocities $\boldsymbol{u}$ by appropriate JACOBIAN matrices $(\partial\dot{\boldsymbol{r}}/\partial\boldsymbol{u})^T$.

2 INTEGRATION SCHEMES

Sophisticated computational methods have been established to adopt mechanical models to a wide range of industrial applications [4]. In order to integrate multibody systems with rigid contacts, two different numerical methods can be distinguished, namely event-driven and time-stepping schemes.

2.1 Event-driven integration schemes

Event-driven or event tracking schemes [20] detect changes of the constraints, for example closing of unilateral contacts or stick-slip transitions, and resolve the exact transition times using indicator functions. Between these events the motion of the system is smooth and can be computed by a standard integrator for differential algebraic equations. While the general procedure using event-driven methods is known, the particular implementation depends on the underlying integrator. Especially, the treatment of constraints and the root finding mechanism play a crucial role in this context [11].

While the event-driven integration is very accurate, the detection of events can be time consuming, especially in case of frequent transitions for example for systems with numerous contacts and Zeno phenomenons. In principal, this approach is used for systems with only few configuration changes.

2.2 Time-stepping integration schemes

In contrast to event-driven schemes so-called time-stepping schemes belong to event capturing methods. They are based on the discretisation of the equations of motion including the complementarity conditions not adapting the globally fixed time step size Δt due to closing contacts. So, time-stepping schemes allow to focus on the global averaged physical behaviour of the simulated models. This reduces the number of combinatorial problems and avoids event detections. Therefore, a large number of contact transitions can be handled with increased computational efficiency if single events are not so important. On the other hand, common time-discretisations are of order one and the integrator is very sensible with respect to the time step size influencing numerical stability and accuracy [24, 10, 11, 25].

A robust linear-implicit time-stepping algorithm of first order on velocity level is briefly introduced as an example. In the following a single integration step $l \rightarrow l+1$ is outlined:

1. Compute the distances $\boldsymbol{g}_U^l = \boldsymbol{g}_U(\boldsymbol{q}^l, t^l)$ of all unilateral contacts.
2. Compute the index set $\{i \in \mathbb{N} : g_{U,i}^l \leq 0\}$ of active unilateral contacts and note that bilateral constraints are active by definition.
3. Compute the generalised velocities by solving the discretised equations of motion considering the active constraints (index a) on velocity level:

$$\begin{aligned}
\boldsymbol{u}^{l+1} &= \boldsymbol{u}^l + \left(\boldsymbol{M}_{\text{eff}}^l\right)^{-1} \left(\Delta t\, \boldsymbol{h}_{\text{eff}}^l + \boldsymbol{W}_a^l \boldsymbol{\Lambda}_a^l\right) , && \text{(14a)}\\
\dot{\boldsymbol{g}}_a^{l+1} &= \dot{\boldsymbol{g}}_a(\boldsymbol{u}^{l+1}, \boldsymbol{q}^l, t^{l+1}) , && \text{(14b)}\\
\boldsymbol{\Lambda}_a^{l+1} &= \mathbf{proj}(\dot{\boldsymbol{g}}_a^{l+1}, \boldsymbol{\Lambda}_a^{l+1}) . && \text{(14c)}
\end{aligned}$$

   The effective mass matrix and right hand side are given by

$$\begin{aligned}
\boldsymbol{M}_{\text{eff}}^l &= \boldsymbol{M}^l - \Delta t \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{u}}\right|_l - \Delta t^2 \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\right|_l \boldsymbol{Y}^l , && \text{(15a)}\\
\boldsymbol{h}_{\text{eff}}^l &= \boldsymbol{h}^l + \Delta t \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\right|_l \boldsymbol{Y}^l \boldsymbol{u}^l + \Delta t \left.\frac{\partial \boldsymbol{h}}{\partial t}\right|_l . && \text{(15b)}
\end{aligned}$$

   An implementation of the complementarity conditions according to Sect. 1.2 is denoted by **proj**.

4. Compute the new generalised positions $\boldsymbol{q}^{l+1} = \boldsymbol{q}^l + \Delta t\, \boldsymbol{Y}^l \boldsymbol{u}^{l+1}$.
5. Correct numerical drifts.

3 MBSIM

The software for modelling and simulation of nonsmooth dynamical systems at the Institute of Applied Mechanics of the Technische Universität München is called *MBSim* [15]. Mathematically, it is based on the ideas outlined in Sects. 1 and 2.

From the software development point of view, [21] proposed a standard structure for multibody simulation frameworks distinguishing between bodies and interactions. Also, the programs described in [2] and [5] follow this approach. It is approved and used in *MBSim* using object-oriented C++ programming. Altogether, Fig. 2 shows the embedding of *MBSim* in the global simulation and analysing process. The kernel of *MBSim* was historically devoted to the analysis of multibody systems explaining the program name. Now, it is able to handle nearly arbitrary dynamical systems according to Sect. 1. Though, the simulation of hydraulics, electronics, control and power train systems is included within several modules. *MBSim* is based on the interface *FMatVec* [7] using either LaPack[2] or ATLAS[3] for fast evaluation of linear algebra routines. Further, with *HDF5Serie* it writes simulation result files in the hierarchical HDF5 file format[4] even for large dynamical systems. These files can be read by *H5PlotSerie* for plotting or by *OpenMBV* [17] for visualisation. Thereby, *OpenMBV* is based on the Coin implementation[5] of the Open Inventor Library[6]. A co-simulation with MATLAB/Simulink[7], *HySim* [19] for hydraulic components and *KetSim* [13] for camshaft timing chains is possible as well [9]. *MBSim* is divided in a modelling part using C++ or XML and a simulation part. The simulation part is implemented quite modular distinguishing between the update of bodies and interactions concerning kinematics, kinetics as well as force laws and integration or nonlinear solution schemes. External libraries are used where it is possible for having always a state-of-the-art numerical basis.

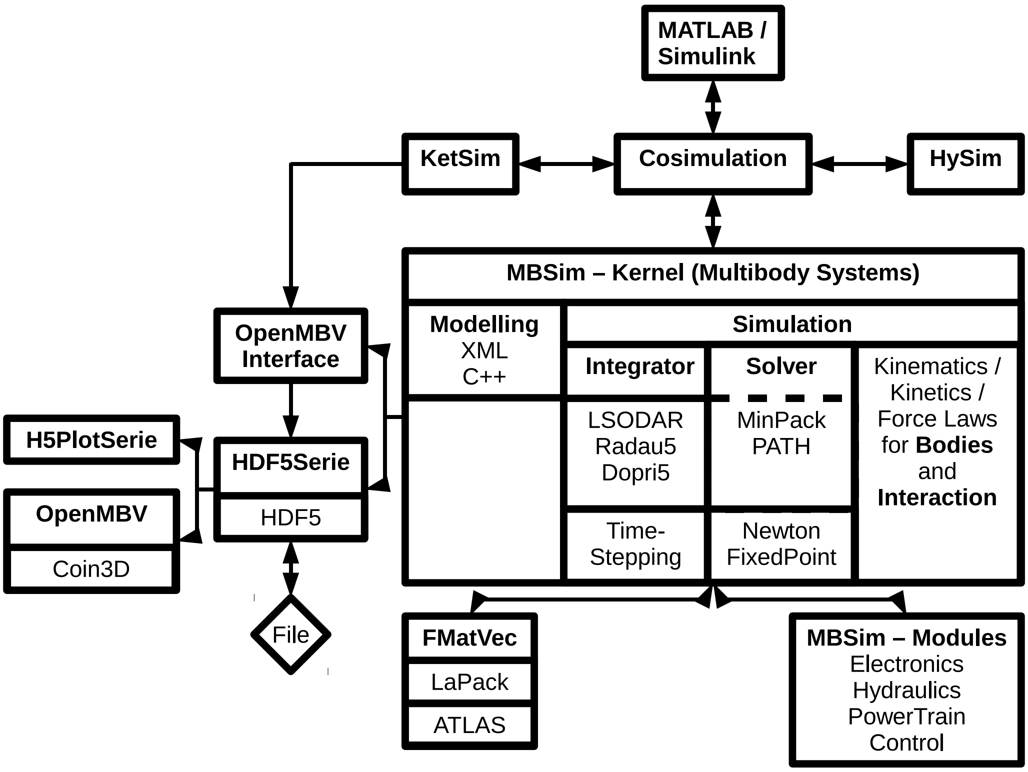


Figure 2. Embedding of *MBSim* in the global simulation and analysing process.

The interface of all *MBSim* classes is documented in the source code using Doxygen[8]. With this class documentation, self-explaining names in the user interface and the tutorial available in [15] it is goal of this section to give an overview about the assembly of *MBSim*.

[2] cf. http://www.netlib.org/lapack/
[3] cf. http://www.netlib.org/atlas/
[4] cf. http://www.hdfgroup.org/HDF5/
[5] cf. http://www.coin3d.org/
[6] http://oss.sgi.com/projects/inventor/
[7] cf. http://www.mathworks.de/
[8] cf. http://www.stack.nl/~dimitri/doxygen/index.html

A typical example of a dynamical system only from mechanics is given by Fig. 3. One has the environment

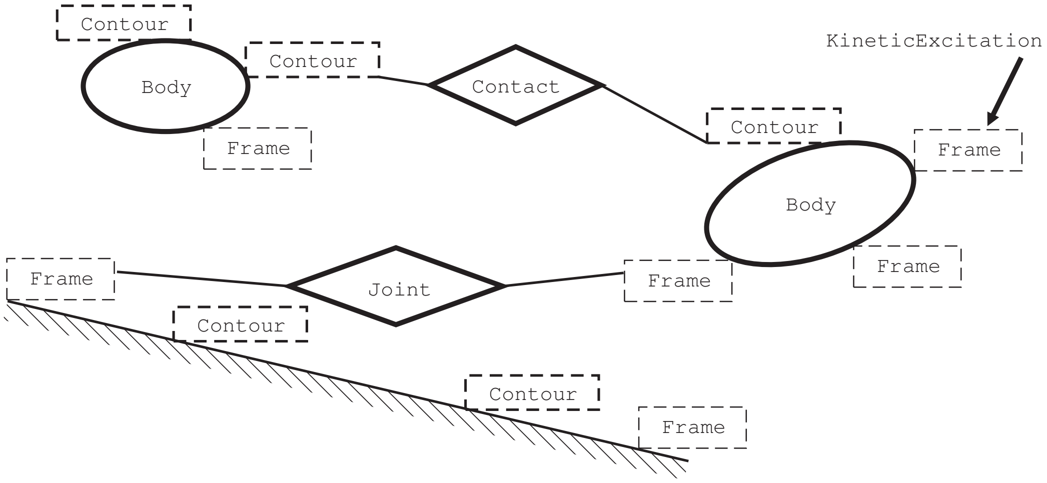


Figure 3. Mechanical object structure in *MBSim*.

and two Objects, in mechanics called Bodys, holding the inertia terms. At a Body both Contours and Frames can be attached. Then, interconnecting Links like frictional Contacts between Contours and like ideal Joints between Body Frames can be defined. External KineticExcitations act on Body Frames. These ingredients are basically explained in the following from the mechanical point of view.

3.1 DynamicSystem and DynamicSystemSolver

Hierarchically Objects and Links belong to DynamicSystems being explained in the following (cf. Fig. 4). Shaded one can see interfaces for the different components of the e.g. smooth modelling equations

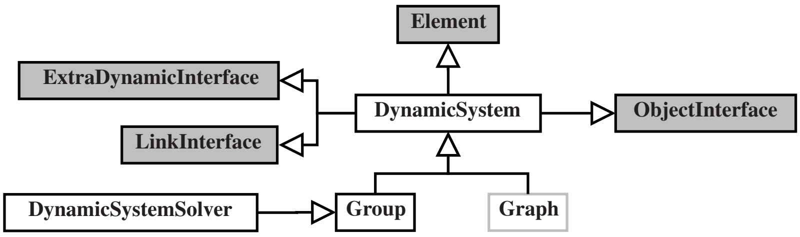


Figure 4. Dynamic system type classes in *MBSim* (UML).

- ExtraDynamicInterface $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x})$,
- ObjectInterface $\dot{\boldsymbol{q}} = \boldsymbol{Y}(\boldsymbol{q})\,\boldsymbol{u}$, $\boldsymbol{M}(\boldsymbol{q})\,\dot{\boldsymbol{u}} = \boldsymbol{h}(\boldsymbol{q},\boldsymbol{u},t)$ with $\boldsymbol{M}$ symmetric positive definite,
- LinkInterface $(\boldsymbol{q},\boldsymbol{u},\boldsymbol{\lambda},t) \in \mathcal{N}$, $\boldsymbol{h}(\boldsymbol{q},\boldsymbol{u},t)$, $\boldsymbol{W}(\boldsymbol{q})\,\boldsymbol{\lambda}$.

and Element for a load/save mechanism as well as plot and data administration. Class Graph is dotted as it is not available for modelling. A graph structure is automatically built during initialisation for an efficient simulation process evaluating the hierarchical modelling structure. Though, DynamicSystems can be hierarchically assembled in Groups. The top-most DynamicSystem is called DynamicSystemSolver. It also represents the interface to the integration schemes and allows the setting of environment variables, like gravitation.

3.2 Frames

`Frames` are a basic concept in the kernel of *MBSim* to define an interface for *MBSim* multibody components concerning kinematic and kinetic expressions. They specify a unique point including position and orientation. New `Frames` can be added arbitrarily and recursively based on a predecessor `Frame`. That is why there has to be a first `Frame` for specific *MBSim* components, e.g. each `DynamicSystem` has got a stationary `Frame "I"` (inertial frame).

Important settings for an arbitrary `Frame` path based on the parent `Body` next to position and Cartesian orientation are defined according to the parametrisation

$$\begin{pmatrix} {}_I\boldsymbol{a} \\ {}_I\boldsymbol{\Psi} \end{pmatrix} = \begin{pmatrix} {}_I\boldsymbol{J}_T \\ {}_I\boldsymbol{J}_R \end{pmatrix} \dot{\boldsymbol{u}} + \begin{pmatrix} {}_I\bar{\boldsymbol{\iota}}_T \\ {}_I\bar{\boldsymbol{\iota}}_R \end{pmatrix} \tag{16}$$

of translational and angular acceleration. This is basically an affine relationship concerning the derivative of generalised velocities $\dot{\boldsymbol{u}}$, JACOBIAN matrices $\boldsymbol{J}$ and an additional summand $\bar{\boldsymbol{\iota}}$ with gyroscopic and explicit time dependent contributions.

3.3 Bodies

Mechanical `Bodys` provide their portion of a positive definite mass matrix, a smooth right hand side, state and energy expressions according to `Object`, which is e.g. similar for some hydraulic objects, like pipes (cf. Fig. 5). One difference between mechanics and hydraulics is the kinematic description which is based on `Frames` in the case of mechanical `Bodys`. Also the connectors to other `Bodys` might follow other structural rules. Depending on the type of linkage, `Frames` or `Contours` occur in the case of `Bodys`. In *MBSim* one distinguishes between rigid and flexible `Bodys`.

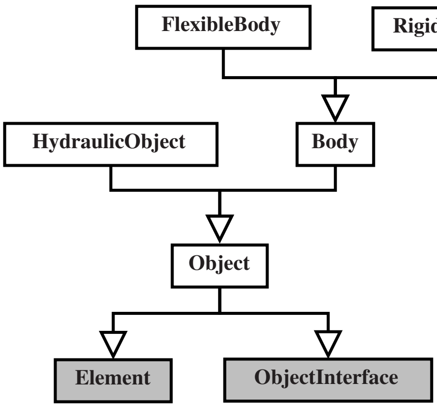


Figure 5. Object type classes in *MBSim* (UML).

3.3.1 Rigid bodies

For each `RigidBody` a `Frame "C"` in the centre of gravity is predefined. One perhaps newly created `Frame` of the `RigidBody` has to be chosen as frame for kinematics `"K"` with centre P and one `Frame` of another `Body` or a `DynamicSystem` has to be chosen as frame of reference `"R"` with centre O. Both absolute –if the frame of reference belongs to a `DynamicSystem`– and relative –if the frame of reference belongs to another `RigidBody`– kinematic structures are canonically given by the `Frame` recursion. The motion of the frame for kinematics and so also of the `RigidBody` with respect to the frame of reference is defined by the individual generalised coordinates $\boldsymbol{q}_{\text{rel}}$ of the `RigidBody` or by a time-dependent path. These settings can be defined individually on position, velocity and acceleration level according to

$$\begin{aligned} {}_R\boldsymbol{r}_{OP} &= {}_R\boldsymbol{r}_{OP}\left(\boldsymbol{q}_{\text{rel}},\, t\right), && (17)\\ \boldsymbol{A}_{RK} &= \boldsymbol{A}_{RK}(\boldsymbol{q}_{\text{rel}},\, t)\,, && (18)\\ {}_R\boldsymbol{v}_{OP,\text{rel}} &= {}_R\boldsymbol{J}_{T,\text{rel}}\;\boldsymbol{u}_{\text{rel}} + {}_R\boldsymbol{\iota}_{T,\text{rel}}\;, && (19)\\ {}_R\boldsymbol{\omega}_{RK} &= {}_R\boldsymbol{J}_{R,\text{rel}}\;\boldsymbol{u}_{\text{rel}} + {}_R\boldsymbol{\iota}_{R,\text{rel}}\;, && (20) \end{aligned}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left({}_R\boldsymbol{v}_{OP,\mathrm{rel}}\right) = {}_R\boldsymbol{J}_{T,\mathrm{rel}}\;\dot{\boldsymbol{u}}_{\mathrm{rel}} + \frac{\mathrm{d}}{\mathrm{d}t}\left({}_R\boldsymbol{J}_{T,\mathrm{rel}}\right)\boldsymbol{u}_{\mathrm{rel}} + \frac{\mathrm{d}}{\mathrm{d}t}\left({}_R\boldsymbol{\iota}_{T,\mathrm{rel}}\right) \;, \tag{21}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\left({}_R\boldsymbol{\omega}_{RK}\right) = {}_R\boldsymbol{J}_{R,\mathrm{rel}}\;\dot{\boldsymbol{u}}_{\mathrm{rel}} + \frac{\mathrm{d}}{\mathrm{d}t}\left({}_R\boldsymbol{J}_{R,\mathrm{rel}}\right)\boldsymbol{u}_{\mathrm{rel}} + \frac{\mathrm{d}}{\mathrm{d}t}\left({}_R\boldsymbol{\iota}_{R,\mathrm{rel}}\right) \;. \tag{22}$$

In the referencing coordinate frame, Eq. (17) describes the translational position of the frame for kinematics and Eq. (18) its orientation. Then, the velocity level is given by an affin relation involving JACOBIAN matrices $\boldsymbol{J}$, generalised velocities $\boldsymbol{u}_{\mathrm{rel}}$ and explicit time-dependent summands $\boldsymbol{\iota}$. The acceleration level can be obtained by differentiation. Figure 6 shows the Frame recursion also including the link relationships of Frames and Contours. In a DynamicSystem two stationary Frames "I" and "Ĩ" are defined as frames of reference of the lower right or left RigidBody in an absolute parametrisation, respectively. For the left RigidBody the frame for kinematics is given by Frame "B", whereas for the lower right RigidBody the canonic Frame "C" in the centre of gravity is used. The lower right RigidBody is also the reference of another RigidBody altogether yielding a relative parametrisation. These two RigidBodys are Linked concerning their "B"-Frames as well. For completing the description, two additional Links between dotted Contours are inserted. The drawback of this general description is a time-dependent mass-matrix also in the absolute kinematics case.

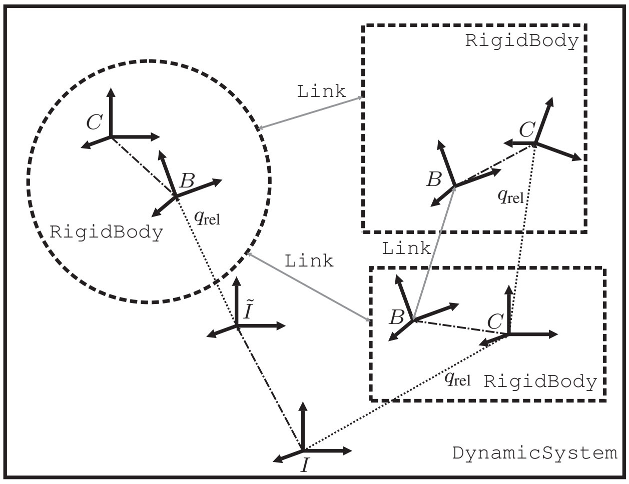


Figure 6. Rigid kinematics in *MBSim*.

3.3.2 Flexible bodies

The equations of motion of a FlexibleBody is at the moment only available with respect to a stationary Frame. So, for FlexibleBodys the frame of reference must belong to a DynamicSystem. Available are different FlexibleBodys for beams and plates as well as linear differential equations based on arbitrary mass and stiffness matrices.

3.4 LinkMechanics

LinkMechanics represents interconnections between mechanical Bodys according to Fig. 7 using the connectors of Bodys. If other connectors are used, other link classes have to be inherited. Links distribute locally to $\boldsymbol{h}$ and $\boldsymbol{W}\boldsymbol{\lambda}$ in the equations of motion. Thereby, it is distinguished between the kinematics and kinetics of links – set-valued and single-valued constitutive laws on acceleration and on velocity level can be selected by a plug-in functionality. Basic types are summarised in the following:

- A SpringDamper connects two Frames using a predefined spring force function.
- A KineticExcitation is connected to one Frame using a predefined excitation function.
- Joints connect two Frames with the force laws depending on the ideal normal relative kinematics. The constitutive law has to be chosen for the calculation of the force parameter.

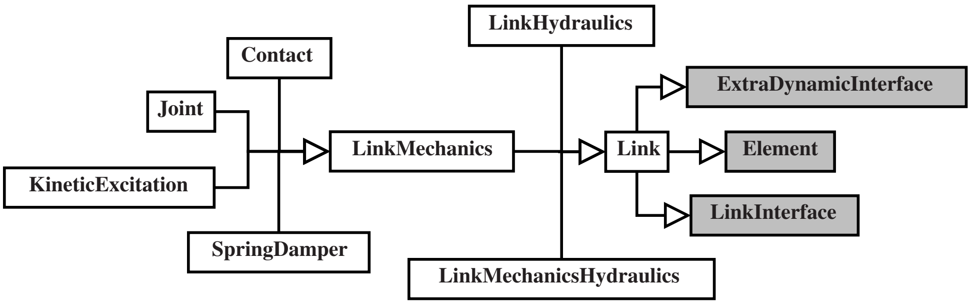


Figure 7. Link type classes in *MBSim* (UML).

- Contacts and impacts are managed by the class `Contact`. Apart from the constitutive law plug-in functionality here also the `ContactKinematics` on position level has to be defined with respect to the used `Contours`.

4 CONCLUSION

The current paper discusses the multibody simulation tool *MBSim*. The underlying equations of motion are defined, advantages and disadvantages of modelling and integrator selection as well as the programming structure are discussed. The representation of rigid and flexible bodies, the incorporation of hydraulic components [22], and modules for nonlinear signal processing, electronics and power train simulation make *MBSim* to a domain independent simulation tool. Co-simulation [8] and parallelisation show that *MBSim* is a unique and efficient basis for the analysis of dynamical questions [28] reducing the number of experiments.

REFERENCES

[1] Acary, V., and Brogliato, B. *Numerical methods for nonsmooth dynamical systems : applications in mechanics and electronics*, 1st edition, vol. 35 of *Lecture notes in applied and computational mechanics*. Springer, Berlin, 2008.

[2] Acary, V., and Perignon, F. An introduction to Siconos. Tech. rep., INRIA, 2007.

[3] Alart, P., and Curnier, A. A mixed formulation for frictional contact problems prone to Newton like solution methods. *Comput Methods Appl Mech Engrg 92* (1991), 353 – 375.

[4] Arnold, M. Numerical methods for simulation in applied dynamics. In *Simulation Techniques for Applied Dynamics*, M. Arnold and W. Schiehlen, Eds., no. 507 in CISM International Centre for Mechanical Sciences. Springer, Wien, 2009, pp. 191–246.

[5] Berard, S., Trinkle, J., Nguyen, B., Roghani, B., Fink, J., and Kumar, V. daVinci Code: A multi-model simulation and analysis tool for multi-body systems. In *2007 IEEE International Conference on Robotics and Automation ; Roma, Italy, 10th-14th April 2007* (2007), IEEE, pp. 2588–2593.

[6] Brogliato, B., ten Dam, T., Paoli, L., Genot, F., and Abadie, M. Numerical simulation of finite dimensional multibody nonsmooth mechanical systems. *Appl Mech Rev 55* (2002), 107–150.

[7] fmatvec - fast matrix-vector library. GNU Lesser General Public License
http://fmatvec.berlios.de.

[8] Friedrich, M., Schneider, M., and Ulbrich, H. A parallel co-simulation for mechatronic systems. In *Proceedings of 1st Joint International Conference on Multibody System Dynamics, Lappeenranta, 25th-27th May 2010* (to appear 2010).

[9] Friedrich, M., and Ulbrich, H. A parallel co-simulation for multi body systems. In *2nd South-East European Conference on Computational Mechanics , Rhodes , 22nd-24th June 2009* (2009).

[10] FUNK, K., AND PFEIFFER, F. A time-stepping algorithm for stiff mechanical systems with unilateral constraints. In *International Conference on Nonsmooth/Nonconvex Mechanics with Applications in Engineering , Thessaloniki , 5th until 6th July 2002* (2002).

[11] FÖRG, M. *Mehrkörpersysteme mit mengenwertigen Kraftgesetzen : Theorie und Numerik*, als Manuskript gedruckt, vol. 411 of *Fortschritt-Berichte VDI : Reihe 20, Rechnerunterstützte Verfahren*. VDI-Verl., Düsseldorf, 2007.

[12] GLOCKER, C. *Set-Valued Force Laws in Rigid Body Dynamics : Dynamics of Non-Smooth Systems*, 1st edition, vol. 1 of *Lecture Notes in Applied and Computational Mechanics*. Springer Verlag, Berlin, 2001.

[13] KELL, T., FRITZ, P., AND PFEIFFER, F. Vibrations in roller chain drives. In *5th International Congress on Sound and Vibration, Adelaide, 15th until 18th December 1997* (1997).

[14] LEINE, R. I., AND VAN DE WOUW, N. *Stability and Convergence of Mechanical Systems with Unilateral Constraints*, vol. 36 of *Lecture Notes in Applied and Computational Mechanics*. Springer, Berlin, 2008.

[15] MBSim - multi-body simulation software. GNU Lesser General Public License
http://mbsim.berlios.de.

[16] MOREAU, J. J. Unilateral contact and dry friction in finite freedom dynamics. In *Nonsmooth Mechanics and Applications*. Springer Verlag, Wien, 1988, pp. 1–82.

[17] OpenMBV - open multi body viewer. GNU General Public License
http://openmbv.berlios.de.

[18] PFEIFFER, F. *Mechanical System Dynamics*, corr. 2. print., vol. 40 of *Lecture notes in applied and computational mechanics*. Springer, Berlin, 2008.

[19] PFEIFFER, F., BORCHSENIUS, F., AND FÖRG, M. New hydraulic system modelling. In *ASME Design Engineering Technical Conference, Chicago, 2nd - 6th September 2003* (2003).

[20] PFEIFFER, F., AND GLOCKER, C. *Multibody Dynamics with Unilateral Contacts*, 1st edition. Wiley series in nonlinear science. John Wiley & Sons Inc., New York, 1996.

[21] SCHIELEN, W. Multibody system dynamics: Roots and perspectives. *Multibody System Dynamics 1* (1997), 149–188.

[22] SCHNEIDER, M., KRÜGER, K., AND ULBRICH, H. Simulation of hydraulic systems with set-valued force laws. In *Proceedings of 1st Joint International Conference on Multibody System Dynamics, Lappeenranta, 25th-27th May 2010* (to appear 2010).

[23] STEWART, D. Rigid-body dynamics with friction and impact. *SIAM Rev 42* (2000), 3–39.

[24] STIEGELMEYR, A. A time stepping algorithm for mechanical systems with unilateral contacts. In *ASME International Design Engineering Technical Conference, Las Vegas, 4th until 7th September 1999* (1999), ASME.

[25] STUDER, C. *Numerics of unilateral contacts and friction : modeling and numerical time integration in non-smooth dynamics*, vol. 47 of *Lecture notes in applied and computational mechanics*. Springer Verlag, Berlin, 2009.

[26] TRINKLE, J., PANG, J.-S., SUDARSKY, S., AND LO, G. On dynamic multi-rigid-body contact problems with Coulomb friction. *J Appl Math Mech 4* (1997), 267–279.

[27] ZANDER, R. *Flexible multi-body systems with set-valued force laws*, als Manuskript gedruckt, vol. 420 of *Fortschritt-Berichte VDI : Reihe 20, Rechnerunterstützte Verfahren*. VDI-Verlag, Düsseldorf, 2009. http://mediatum2.ub.tum.de/node?id=654788.

[28] ZANDER, R., RETTIG, F., AND SCHINDLER, T. Concepts for the simulation of belt drives - industrial and academic approaches. In *Proceedings of 1st Joint International Conference on Multibody System Dynamics, Lappeenranta, 25th-27th May 2010* (to appear 2010).