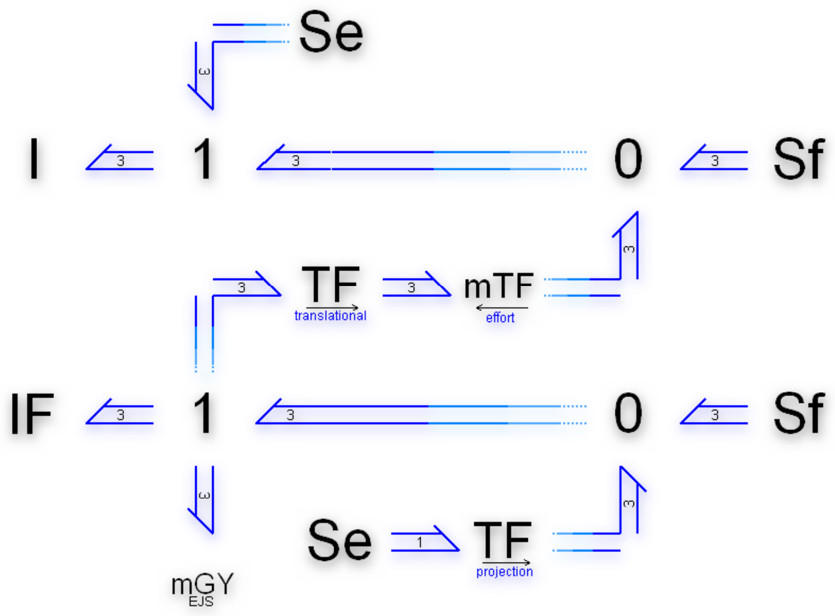
Master Thesis

A Modelica Library for MultiBond Graphs and its Application in 3D-Mechanics

Dirk Zimmer

February 2006



ETH Zürich
Department of Computer Science
Institute of Computational Science

Adviser:
Prof. François E. Cellier

Responsible:
Prof. Walter Gander

Abstract

Bond graphs have established themselves as a reliable tool for modeling physical systems. Multibonds are a bondgraphic extension that provides a general approach to modeling all kinds of multidimensional processes in continuous physical systems. This MS thesis presents a Modelica library for modeling multi-bond graphs and their application to three-dimensional mechanical systems. A comprehensive set of bondgraphic models for ideal mechanical components is provided that enables a fully object-oriented modeling of mechanical systems. The wrapping of the bondgraphic models and their representation by meaningful icons gives the mechanical models an intuitive appeal and makes them easy to use. The resulting mechanical systems can be efficiently simulated. Additionally, the continuous mechanical models were extended to hybrid models that allow discrete changes to be modeled that occur in mechanical system as a result of hard impacts.

Acknowledgments

I’d like to thank my adviser Prof. François E. Cellier for his great support and guidance throughout this thesis. I wish to express my gratitude to Prof. Martin Otter who gave me the opportunity to do an internship at the German Aerospace Center DLR (Deutsches Zentrum für Luft und Raumfahrt). This offered a great opportunity for me to learn about the modeling of mechanical systems and to improve my Modelica skills. Of course I am gratefully indebted to my family and friends for all their patience and support.

Contents

Chapter 1

Introduction

1.1 Motivation

The topic of this thesis is the object-oriented modeling of mechanical systems. In contrast to many other software or methods for the same purpose, the modeling within this work is completely done in terms of bond graphs.

The bondgraphic methodology serves as a powerful modeling tool in the field of physical systems. A lot of models (e.g. for electrical circuits, heat distribution, convectional mass flows, etc.) have been developed by the usage of bond graphs since their invention in 1961 by Paynter[19]. Besides, bondgraphic models also turned out to be of great aid for teaching purposes, because these models allow to get a closer insight into the underlying processes and are often a lot easier to understand than their classic counterparts expressed by pure equations.

The simulation program Dymola[10] provides an object-oriented graphical modeling environment that perfectly suits the demands of the bondgraphic modeling. Dymola is based upon Modelica[21], a general language for object-oriented modeling. Hence, a Modelica library called BondLib[6], [7] has been developed by F. E. Cellier and his students to comfortably model with bond graphs. Using this library, various kinds of bond graphs can be created in an object-oriented fashion. The library uses the graphical support of Dymola, which makes it possible to easily compose bond graphs on screen by drag and drop.

1.2 Approach

Mechanical processes are multidimensional processes. A special bondgraphic enhancement exists, that has been created to handle multidimensional modeling. These are the so called multibond graphs sometimes also to as “vector bond graphs”. Unfortunately, this feature is not supported in the classic BondLib. Therefore the creation of a Modelica library for multibondgraphic modeling was a necessary prerequisite to this work. Although mechanical models are the main application for multibond graphs, the library provides a general solution.

Multibond graphs of simple ideal mechanical elements in 2D and 3D were created and wrapped inside a domain-specific frame. These wrapped models were organized in Modelica libraries for mechanical systems. All models are represented by meaningful icons and give the library an intuitive appeal. Any modeler can use these libraries even by complete ignorance of the bond graph theory.

Two important Modelica libraries were of exemplary nature for this project: One of them is the already mentioned BondLib. It served as a template for the new MultiBondLib. The other one is the MultiBody library designed by M. Otter. His work at the research center DLR resulted in the well known MultiBody library [24], which is part of the standard Modelica package. The decomposition of mechanical systems into ideal elements and the structure of the library were the goal to reach. Also certain methods to solve specific problems (e. g. concerning kinematic loops) had to be developed during the creation process of the Multibody library. In this thesis’ work I could take full advantage of these methods.

1.3 Structure

This thesis is separated into three parts:

Part I introduces the reader to the methods and applications of bondgraphic modeling. This is followed by a presentation of a library for multibond graphs in chapter 3. Chapter 4 is then leading over to part II and presents a survey of applications of bondgraphic modeling to mechanical systems existing so far.

Part II starts with an essay that presents the fundamental laws of motion in a historical context. This is followed by the three chapters 6, 7 and 8 that each present a Modelica library for mechanical systems. These libraries are all built upon models that consist of multibond graphs. The first library is designed to simulate planar mechanical systems. The step into the three dimensional world is then taken in chapter 7, where the “Mechanics3D” library is presented. Chapter 8 provides an extension of this library that is able to handle force impulses as they appear in hard collisions.

Part III is concluding this thesis by a final discussion and presents an outlook on future work. This part also contains the appendix and the bibliography.

1.4 Notation

Vectors and matrices are written in bold face throughout the whole text to allow a better distinction between vectorial and scalar variables.

The vector cross product can also be written in matrix form. The corresponding skew-symmetric matrix is composed out of the vector elements and is denoted with a tilde.

$$\mathbf{a} \times \mathbf{b} = \tilde{\mathbf{a}}\mathbf{b}$$

where

$$\tilde{\mathbf{a}} = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$$

The derivative with respect to time is mostly denoted by a dot.

$$\dot{x} = \frac{d}{dt}x$$

$$\ddot{x} = \frac{d^2}{dt^2}x$$

Part I

Bondgraphic Modeling

Chapter 2

Introduction to Bondgraphic Modeling

This introduction offers the reader a short presentation of the bondgraphic terminology and its usage for modeling purposes. It starts with an easy example of an electric circuit and presents the most important bondgraphic elements. Further on, the issues of domain independence and causality are discussed, and a Modelica library for bondgraphic modeling is presented. Nevertheless, this is only a brief introductory text. Readers who are looking for a more thorough discussion of this topic are referred to the books “Continuous System Modeling”[5] by F. E. Cellier and “System Dynamics”[16] by D. C. Karnopp.

2.1 Classical bondgraphic modeling

Bond graphs are a domain neutral modeling tool for continuous system modeling in the field of physics. A bondgraphic model is a directed graph, where the edges are the bonds themselves. A bond is drawn as a half arrow (or “harpoon”) as in figure 2.1 and represents a power flow between the two vertex elements. We observe that in many systems power can be expressed as the product of two physical variables. One of them is called the effort variable *e* and it is written on the harpoon side of the bond. The other variable is called flow variable *f* and it is written on the plain side of the bond.

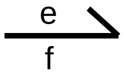


Figure 2.1: A single bond.

In an electric system the power is the product of the voltage *u* and the current *i*. It is stated by definition that *u* is the effort variable and in consequence *i* is the flow variable. Therefore figure

2.2 presents a bond of an electric system. However, this assignment of effort and flow is arbitrary and a converse assignment would involve no disadvantages.

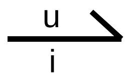


Figure 2.2: An electric Bond carrying voltage u and current i.

The power of this modeling approach becomes evident, if we divide a physical system into separate elements. We then observe that each of these elements has a certain specific behavior with respect to power and energy. An electric circuit as the one in figure 2.3 exemplifies this pretty well.[1]

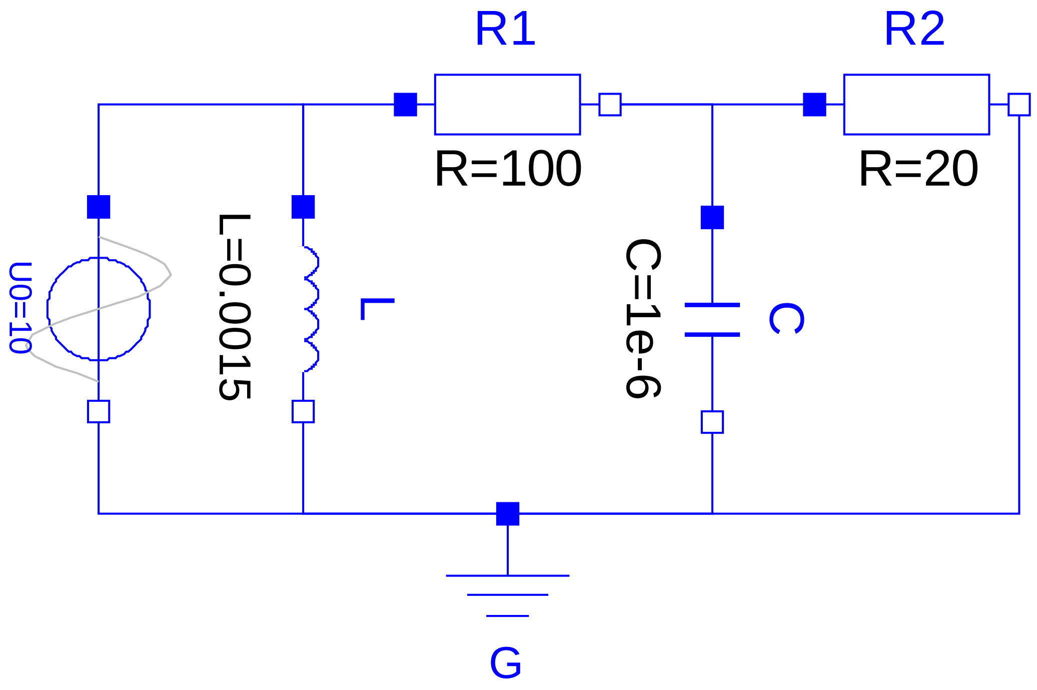


Figure 2.3: An electric circuit.

Certain elements are sources of energy as the battery is one. Other elements like the resistor dissipate energy or, to be more precise, transduce it into thermal energy. Energy can also be stored in capacitive or inductive elements like the capacitor or the coil. These single elements are coupled with each other by parallel or serial connections.

These elements form the vertices of the bond graph and are represented by a mnemonic code according to their behavior with respect to energy and power. Figure 2.4 shows the most important bondgraphic elements together with the equations they generate and their electric analogon.

[1] This example is taken out of [7]

Symbol (mnemonic code)	**Name**	**Electric Analogon**
Se	Source of effort	Voltage source
$e = e_0$		$u = u_0$
Sf	Source of flow	Current source
$f = f_0$		$i = i_0$
R	Resistor	Linear resistor
$e = R \cdot f$		$u = R \cdot i$
G	Conductance	Linear conductance
$f = G \cdot e$		$i = G \cdot u$
C	Capacitance	Capacitor
$\frac{de}{dt} \cdot C = f$		$\dot{u} \cdot C = i$
I	Inductance	Coil
$\frac{df}{dt} \cdot I = e$		$\dot{i} \cdot I = u$
0	0-Junction	Parallel connection
$\sum f = 0$ All effort variables are equal		$\sum i = 0$ All voltages are equal
1	1-Junction	Serial connection
$\sum e = 0$ All flow variables are equal		$\sum u = 0$ All currents are equal

Figure 2.4: Basic bondgraphic vertex elements and their corresponding equations.

The elements of the electric circuit can now be replaced with their bondgraphic counterparts and the circuit is modeled by a bond graph.

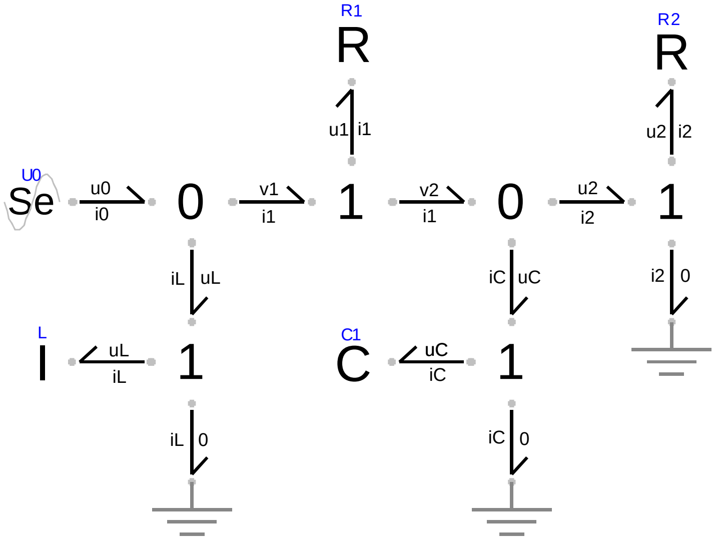


Figure 2.5: Bond graph of the electric circuit.

The bond graph can be simplified by removing ground elements and 2-Port[2] Junctions.

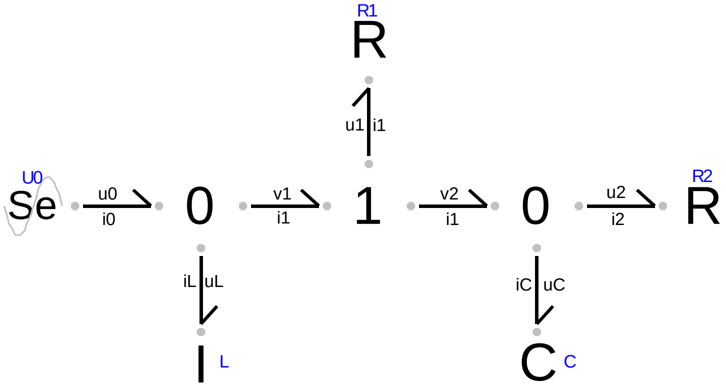


Figure 2.6: Simplified bond graph of the electric circuit.

[2] The vertex elements can also be called according to their number of connections (vertex degree) as 1-Ports, 2-Ports or in general n-Ports. A source of effort is usually connected with only a single bond and is therefore a 1-Port. 0- and 1-Junctions are often 3- or 4-Ports.

The acausal equations can now simply be read out of the bondgraphic model:

$$
\begin{array}{rcl}
u_0 & = & f(t) \\
i_0 - i_L - i_1 & = & 0 \\
u_L & = & u_0 \\
di_L/dt \cdot L & = & u_L \\
v_1 & = & u_0 \\
v_1 - v_2 - u_1 & = & 0 \\
u_1 & = & R_1 \cdot i_1 \\
v_2 & = & u_C \\
i_1 - i_2 - i_C & = & 0 \\
du_C/dt \cdot C & = & i_C \\
u_2 & = & u_C \\
u_2 & = & R_2 \cdot i_2
\end{array}
$$

This set of equations can be causalized, i.e. transformed to a set of assignment statements that forms the computational description of the system. The system can now be simulated by integration of the differential equations.

$$
\begin{array}{rcl}
u_0 & := & f(t) \\
u_L & := & u_0 \\
v_1 & := & u_0 \\
u_2 & := & u_C \\
i_2 & := & u_2/R_2 \\
v_2 & := & u_C \\
u_1 & := & v_1 - v_2 \\
i_1 & := & u_1/R_1 \\
i_0 & := & i_L + i_1 \\
i_C & := & i_1 - i_2 \\
 & & \\
di_L/dt & := & u_L/L \\
du_C/dt & := & i_C/C
\end{array}
$$

2.2 Interdisciplinary modeling

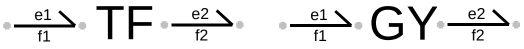
Using other variable pairs for effort and flow makes bondgraphic modeling an extremely valuable tool for a wide range of physical applications. Table 2.1 shows the domain dependent choice of effort and flow according to the general bond graph (GBG) concept.

physical domain	**effort**		**flow**	
translational mech.	force	f $[N]$	velocity	v $[ms^{-1}]$
rotational mechanics	torque	t $[Nm]$	angular velocity	ω $[rad\ s^{-1}]$
hydraulic/acoustic	pressure	p $[Nm^{-2}]$	volume flow	ϕ_v $[m^3 s^{-1}]$
electric	voltage	u $[V]$	current	i $[A]$
magnetic	current	i $[A]$	$\frac{d}{dt}$ magn. flux	$\dot{\Phi}$ $[V]$
thermodynamical	temperature	T $[K]$	entropy flow	$\dot{S}$ $[Js^{-1}K^{-1}]$
chemical	chemical potential	μ $[J \cdot mol^{-1}]$	molar flow	$\dot{N}$ $[mol \cdot s^{-1}]$
material	total chem. pot.	μ_{tot} $[J \cdot mol^{-1}]$	molar flow	$\dot{N}$ $[mol \cdot s^{-1}]$

Table 2.1: Domain dependent choice of effort and flow.

2.2.1 Transformers and gyrators

The single domains can be easily coupled together by the use of transformer and gyrator elements:



$$f_2 = m \cdot f_1 \qquad e_2 = m \cdot f_1$$
$$e_1 = m \cdot e_2 \qquad e_1 = m \cdot f_2$$

Figure 2.7: Presentation of the bondgraphic transformer and gyrator.

A possible coupling of two domains by the usage of a gyrator is an electric engine, where the current i of the electric domain is proportional to the torque t of the rotational domain. A prismatic pump couples the translational domain and the hydraulic domain. It is modeled as transformer, where the pressure p (effort) is proportional to the force f (effort) by the factor of the piston area A.

In figure 2.8 the electric inductance of the last section's model is transferred to the magnetic domain by a gyrator and the storage of magnetic energy in the coil is now modeled as magnetic capacitance. It is important to see that this is not a change in the model, it is just another way of writing.

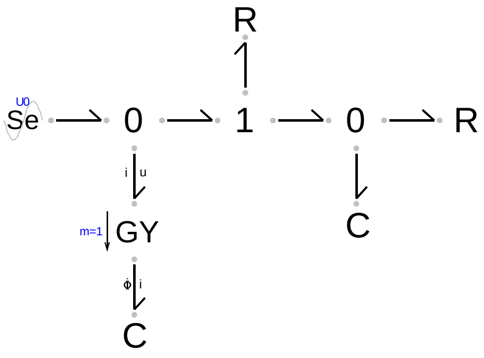


Figure 2.8: Modified bond graph of the electric circuit.

2.2.2 Dual bond graphs

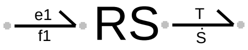
Above model's gyrator with $m = 1$ is a special case. Such a gyrator is called symplectic gyrator and actually denoted with SGY. As mentioned before, the choice which variable of the pair to take as effort is arbitrary and can be modified. A bond graph using i as effort and u as flow variable is the dualized variant of the standard electric bond graph and also capable to model electric circuits. In such a dual bond graph inductances change to capacitances, resistors change to conductors, 0- junction change to 1-junctions and everything vice versa. A symplectic gyrator is the tool to couple dualized variants and therefore partially dualized bond graphs are possible. You may now conclude from the example in figure 2.8 that a magnetic bond graph is nothing else than a dualized electric bond graph.

2.2.3 Intradomain transformers

Of course the usage of transformer elements is not limited to interdomain modeling. Transformers can as well model processes within a single domain. A simple voltage transformer or a mechanic lever are classical examples for this. Gyrators appear mostly in interdomain modeling, but there are also intradomain gyrators like the ones in the eulerian junction structure that will be presented in chapter 7.2.3.

2.2.4 Coupling to the thermal domain

Dissipative elements like resistors or conductors are sources of entropy and therefore also active thermal elements. In bond graphs this can be explicitly modeled using the thermal transducer element, which is a resistor for the domain free side and a source of entropy flow for the thermal side. E.g. it can be used to design a thermal model of the electric resistor R1 in figure 2.6.



$$e = R \cdot f$$
$$\dot{S} := (e \cdot f)/T$$

Figure 2.9: Presentation of thermal transducer element.

Figure 2.10 shows a modification of the model in figure 2.6, where the Resistor R1 is made temperature dependent. The dissipated energy becomes entropy and generates heat in the thermal capacitance of the resistor. This heat is influencing the behavior of the resistance. In this model we also perceive that bondgraphic elements can be modulated by signals from outside. The modulated elements are denoted by the prefix "m" (here: **mRS**). The modulating signals are usually state and/or time dependent. In this case the signal is the temperature and its origin is a sensor element: **De**. This sensor element is measuring the bondgraphic effort and generates the output signal.

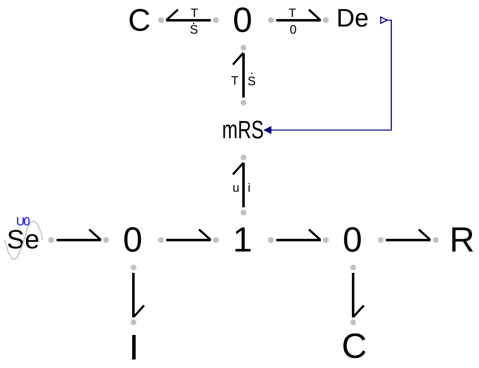
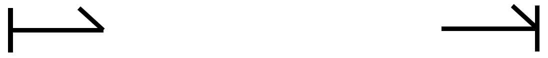


Figure 2.10: Bond graph of the electric circuit with a temperature dependent resistance.

2.3 Causality

A normal bond graph results in an acausal set of equations as we have seen in the first section of this chapter. This set has then to be causalized by a translator program. However, it is possible to express the causal relations also in the bond graph.

We always need two equations to compute the effort and flow variables of a bond. It turns out that each end of a bond computes one of the two variables. Therefore causal variants of the bond have been designed, where a stroke is marking on which side the flow is computed.



Flow variable computed at tail   Flow variable computed at head.

Figure 2.11: The two causal bonds.

The bondgraphic elements have also certain characteristics concerning causality. The causality of sources is fixed, inductances and capacitances have preferred causality, because we'd like to use them as integrators: An inductance prefers to compute the flow and a capacitance prefers to compute the effort. Resistors and conductors are free in their choice of causality. Since a 0-Junction defines just one flow equation, there must be exactly one causality stroke at a 0-junction and for corresponding reasons exactly $n-1$ causality strokes at a 1-junction (where n is the junction degree). Using these rules it is possible to causalize the bond graph of Fig 2.6 as shown in Fig 2.12.

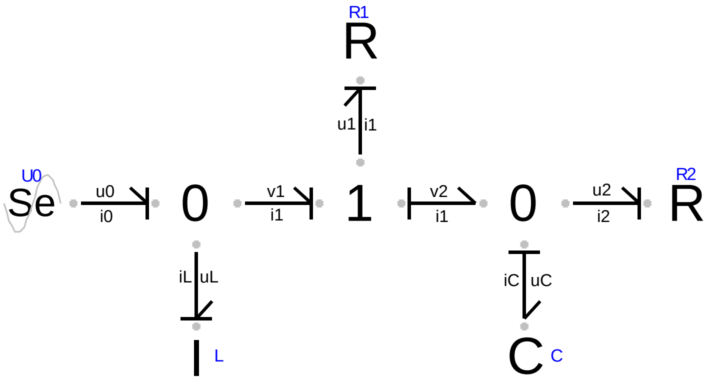


Figure 2.12: Causal bond graph of the electric circuit.

The application of causal bonds gives the modeler a closer insight to possible computational problems. It is not always possible to find a unique causalization, sometimes there are several solutions. Such a lack of uniqueness implies the existence of an algebraic loop. A violation of an element's causality preference reveals a structural singularity. The causalization of Fig 2.12 is unique and all preferences are fulfilled. Therefore there is neither an algebraic loop nor a structural singularity.

It is remarkable that the causal bonds also have a physical interpretation: It is impossible to control both variables, effort and flow at any point of a physical system.

2.4 The Modelica bond graph library

To comfortably model with bond graphs a Modelica library called BondLib has been developed. Using this library, bond graphs can be created in an object-oriented fashion. All important bondgraphic elements are implemented as single models and just wait to be composed together. This library uses the graphical support of the simulation program Dymola, which makes it possible to easily compose bond graphs on screen by drag and drop. The basic bondgraphic elements and the bonds themselves can be placed on the screen and easily be connected with each other. The single elements can then be specified via parameter menus.

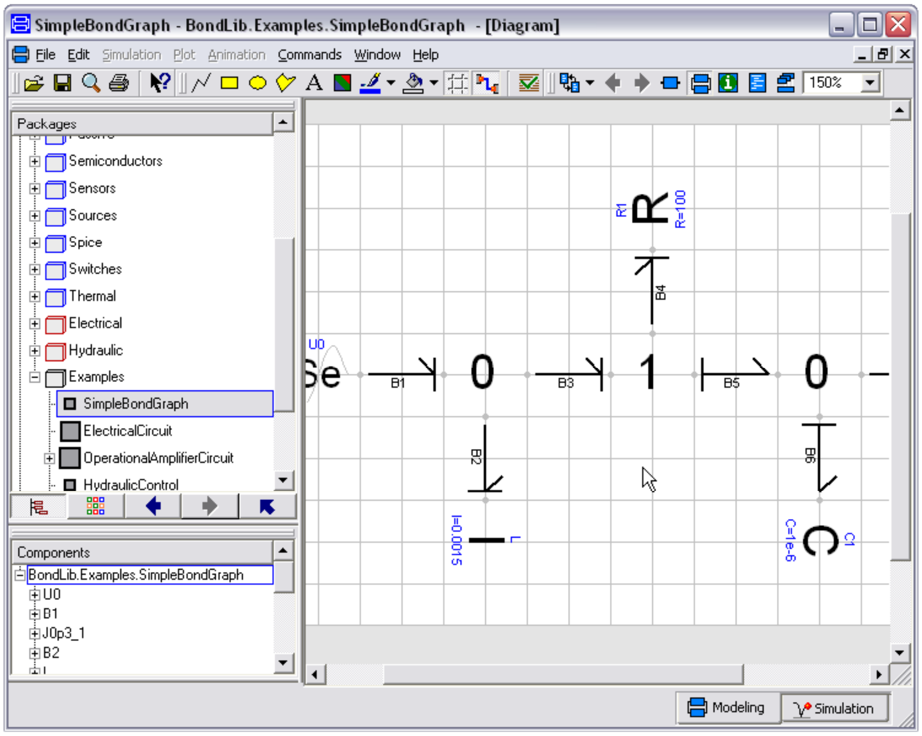


Figure 2.13: A screenshot of the Dymola modeling environment.

The models of the bondgraphic library do not restrict themselves to domain independent bondgraphic elements. Also domain specific, application-near solutions have been developed, e. g., basic electric circuit elements like the ones, which where used to compose the model of figure 2.3. The bond graphs are wrapped inside these models, so that the user can model electric circuits without any knowledge of bond graphs. Also the mechanical models of this thesis will be wrapped in similar fashion.

2.5 Advantages of bondgraphic modeling

Bond graphs are a modeling tool like any other, but compared to equations bond graphs are less general, because there are certain intrinsic assumptions that the bondgraphic concept is based on:

- The systems can be described by a finite set of elements.
- The state of the system can be described by a finite set of continuous state variables.
- There is an energy function for the system and all its subparts that satisfies the first law of thermodynamics
- There are no relativistic effects and time is a global variable.
- The second law of thermodynamics holds.
- The exchange of power happens continuously.

These intrinsic assumptions limit the applicability to physical systems, but give bond graphs also their strength. Concerning the modeling of complex physical systems they find the right balance between specificity and generality.

The interdisciplinary concept of energy and power flows creates a semantic level for each bond graph, even by unawareness of the modeling domain. Thus strong beliefs in physics like the first rule of thermodynamics can always be verified in a bond graph, independent of its application. This semantic level helps the modeler to avoid errors and to find an appropriate solution for his task. Modeling by equations is far more general and therefore leaves more room for mistakes or errors.

The usage of causal bonds helps to understand the computational side of the model and to foresee structural problems. Another advantage of bond graphs is the graphical way of modeling. Relations can be expressed more naturally by two dimensional drawings than in 1-dimensional code. Also the limitations of the screen (or drawing board) forces the modeler to split his model into simple, understandable elements. “Spaghetti code”-like models are therefore avoided.

Chapter 3

Multibond Graphs

3.1 Introduction

For certain modeling tasks, special variants of bond graphs have been established, due to their usefulness. One such task is the modeling of convectional mass flows, where entropy, volume, and mass flows have to be regarded in parallel. Bus bonds are a bondgraphic extension that satisfy these requirements [12], [13]. A red bus bond carries always a specific number of bonds of different domains. In the case of convectional mass flows these are three parallel bonds with the effort - flow pairs:

- (temperature, flow of entropy)
- (pressure, volume flow)
- (free enthalpy, mass flow).

Fig 3.1 shows these three parallel bonds and the abbreviation by a bus bond.

Another important and helpful bondgraphic extension are the multibonds, sometimes also called vector bonds. Contrary to the bus bonds, multibonds carry a variable number of bonds of the same domain or at least of closely related domains (like the translational and the rotational domain).

Figure 3.2 shows three parallel bonds and the abbreviation by a multibond. The number placed in the middle of the bond denotes its cardinality.

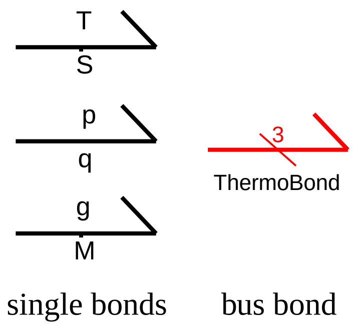


Figure 3.1: Three separate bonds abbreviated by a bus bond.

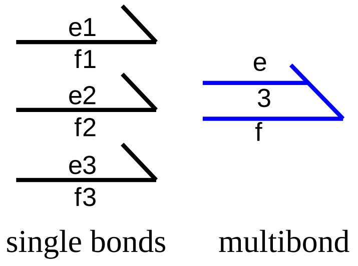


Figure 3.2: Three separate bonds abbreviated by a multibond.

3.2 Applications of multibond graphs

Multibond graphs are mainly used to model multidimensional processes. These can be diffusion processes like a heat distribution in a planar electric circuit. In this thesis multibond graphs are used to model multidimensional mechanics. Another interesting field of application is the modeling of chemical reaction dynamics. Multibonds may also be applicable in the field of general relativity[1], although this violates some of the intrinsic assumption of bond graphs.

3.3 The MultiBondLib

The BondLib only contains models for single bond graphs. To comfortably create models of multibond graphs, an additional library is needed. This thesis presents a Modelica library for multibondgraphic modeling. It's called MultiBondLib and was designed to bear a strong resemblance to the existing BondLib for modeling single bond graphs. A structural overview of the MultiBondLib is presented in appendix C.

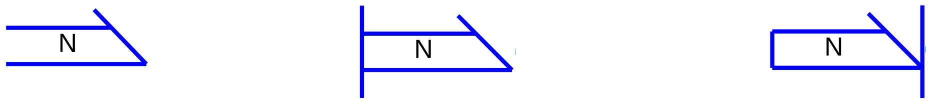
[1] see [14]

3.4 Classic bondgraphic elements and their multibondgraphic counterparts

Each vertex element of a single bond graph has its multibondgraphic counterpart. The elementary equations of the basic bondgraphic elements remain the same. A transformation to the multibondgraphic terminology just extends the equations to be written in vectorial form. Because the two libraries are very similar in structure and design, most of the important components of the MultiBondLib are presented in the following pages together with their classic counterparts of the BondLib. All users of the BondLib should be able to quickly adapt to the new MultiBondLib.

3.4.1 Bonds

The multibond has already been introduced in Fig 3.2. Effort and flow are now vectors of the size *n* where *n* is the cardinality of the multibond. There are also multibond counterparts for the causal bonds, but it has to be mentioned that these two causal multibonds do not cover all possible causalizations, because mixed causality is possible. However, a notation for mixed causality is missing and would not be very meaningful. If we want to explicitly show it, we have to decompose the multibond into single causal bonds.

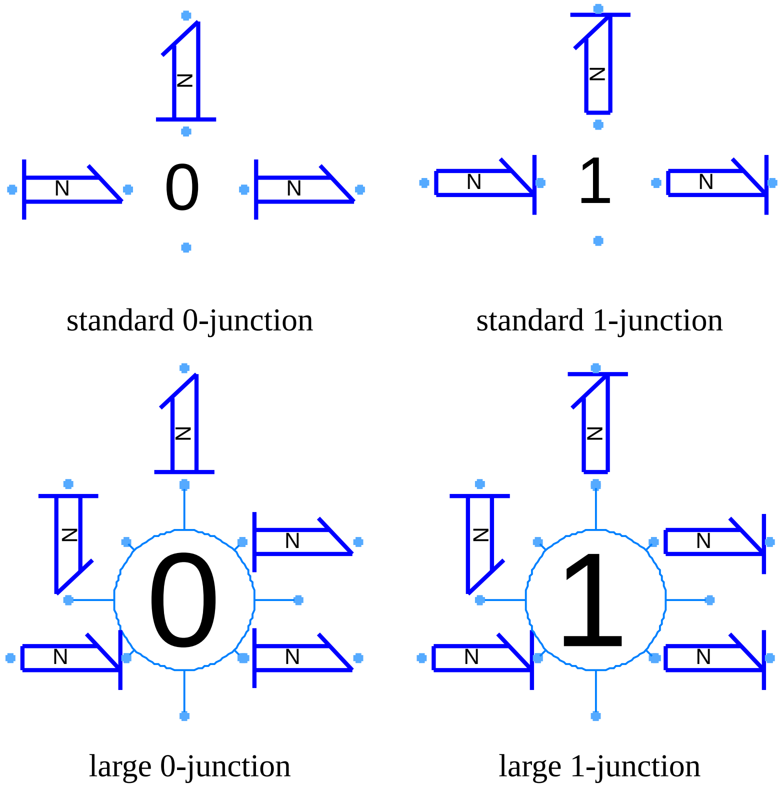


non-causal multibond flow vector computed at tail flow vector computed at head

Figure 3.3: Causal variants of the multibond.

3.4.2 Junctions

The 0- and 1-Junctions of multibond graphs connect the corresponding dimensions. In the MultiBondLib these elements are variable multiports and can therefore handle a variable number of connected bonds. There are standard junctions that handle up to four ports and there are large junctions that handle up to eight ports. Even bigger junctions can be created by coupling two junctions of the same sex with a bond. Contrary to the junctions used in BondLib, it is here possible to leave unused junction connectors open.



standard 0-junction

standard 1-junction

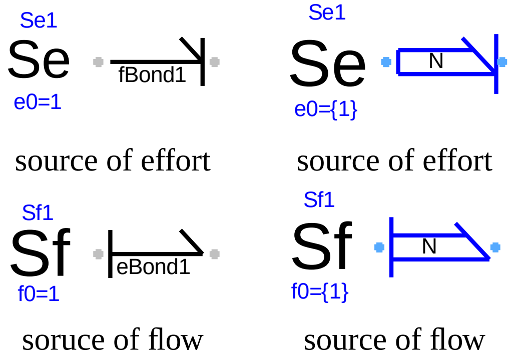
large 0-junction

large 1-junction

Figure 3.4: Multibondgraphic junctions. The causal bonds present a possible causality.

3.4.3 Sources

The source elements are nearly unchanged except that they now have to be specified by a vector. Sources are always of fixed causality.



source of effort

source of effort

soruce of flow

source of flow

Figure 3.5: Sources of flow and effort.

3.4.4 Passive elements

In classic bond notation, passive field elements (this means: resistive, conductive, capacitive or inductive fields) have to be modeled always as multiport elements. An ambiguity in the notation of field elements arises by the introduction of multibonds. An example for this ambiguity is given in figure 3.6, which shows the same resistive field as 2-port or as 1-port element.

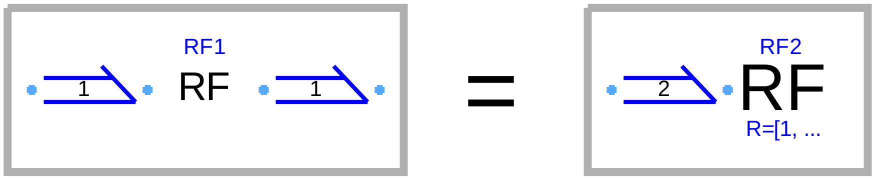


Figure 3.6: Two variants of the same resistive field.

Passive field elements are modeled, if not stated otherwise, as 1-port elements in the MultiBondLib because this is their most common and most comfortable kind of usage.

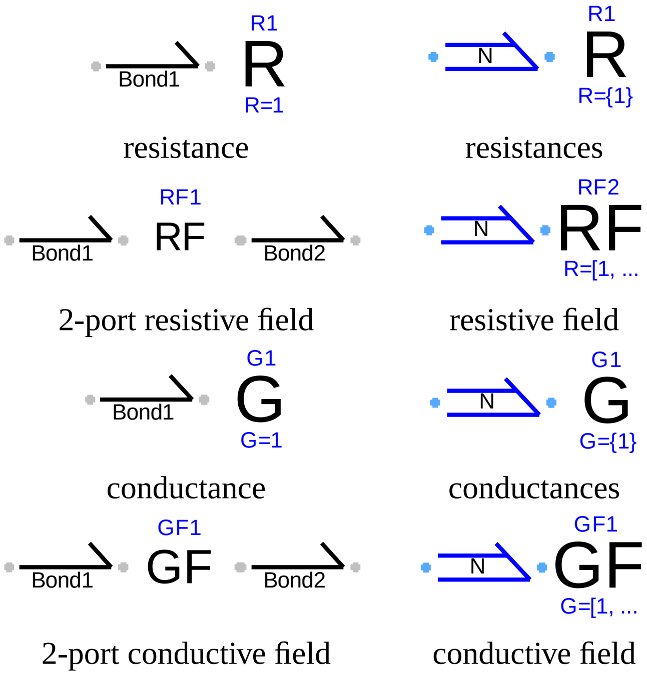
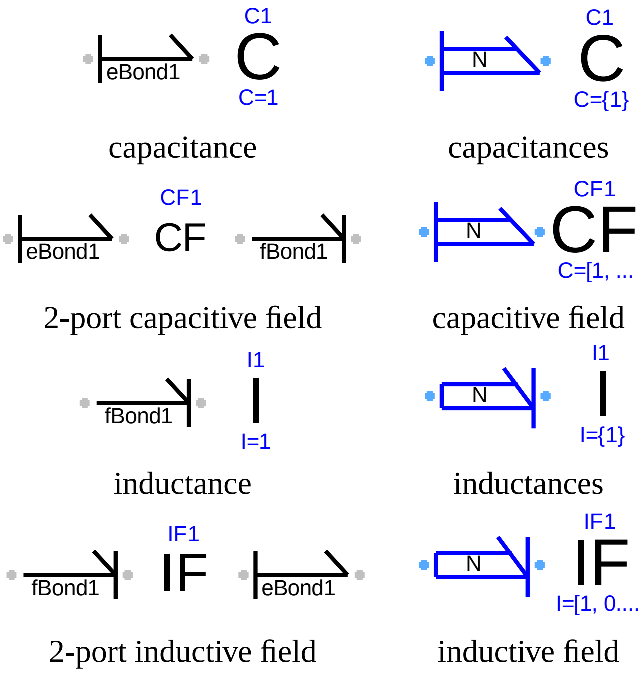


Figure 3.7: Dissipative elements. These elements are free in their choice of causality.

As consequence we observe that in our multibond graphs simple decomposable resistances and resistive fields are modelled as one ports and that a distinction in the notation becomes helpful. Therefore we denote a simple resistance with R and a resistive field with RF. Conductive, capacitive and inductive fields are treated in the same way.[2] Of course one could argue that a simple resistance is nothing else than a special case of a resistive field and that there is therefore no need for a special treatment, but this point of view is not always recommendable, especially concerning the computational aspects.



capacitance

capacitances

2-port capacitive field

capacitive field

inductance

inductances

2-port inductive field

inductive field

Figure 3.8: Storage elements. A causal bond emphasizes the element's preferred causality.

Transformers and gyrators are modeled as 2-port elements to allow a most convenient way of usage. They both are neutral elements and therefore the inflowing power $\mathbf{e}_1^T \cdot \mathbf{f}_1$ has to equal the outflowing power $\mathbf{e}_2^T \cdot \mathbf{f}_2$. Hence, the transformation of an effort or flow variable by a matrix determines also the transformation of its dual variable. If the flow variable is transformed by the matrix **M**:

$$\mathbf{f}_2 = \mathbf{M}\,\mathbf{f}_1$$

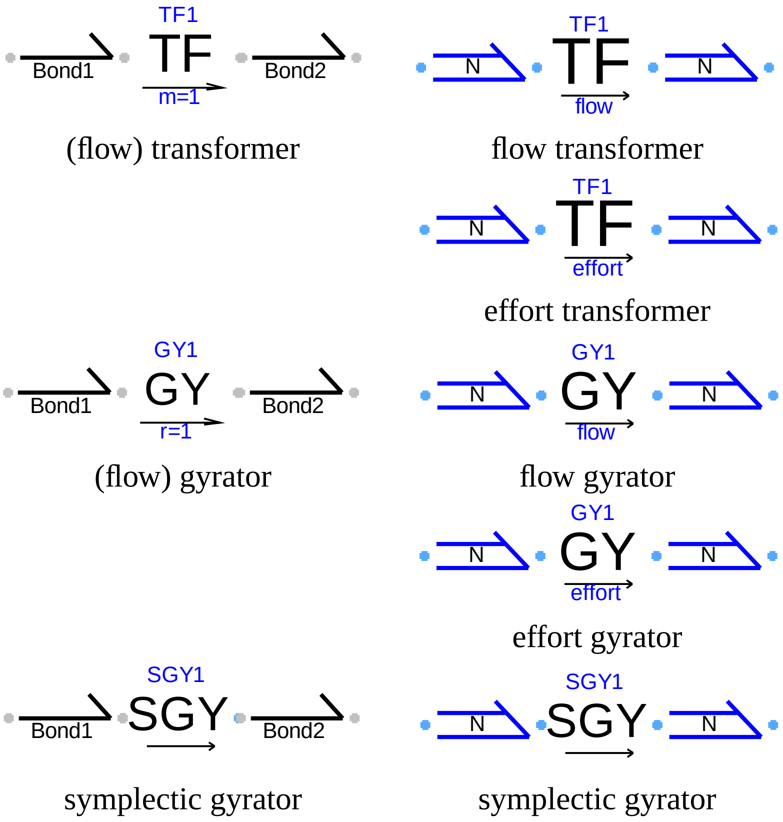
The transformation of the effort is:

$$\mathbf{e}_1 = \mathbf{M}^T\,\mathbf{e}_2$$

There is an inconsistency in the bondgraphic literature about what to transform: the effort variable or the flow variable. In fact, both possible variants turn out be useful and are provided in this library. So each bondgraphic (flow) transformer is confronted with two multibondgraphic counterparts. The same holds for gyrators. A flow gyrator transforms the flow at the inflow to

[2] This distinction between R and RF elements is not part of the original bondgraphic notation.

the effort at the outflow. An effort gyrator does the opposite. What is inflow and what is outflow is denoted by an arrow beneath the mnemonic symbol. Multibondgraphic transformers can also transform between multibonds of different cardinality. Such transformers are called projective transformers.



(flow) transformer

flow transformer

effort transformer

(flow) gyrator

flow gyrator

effort gyrator

symplectic gyrator

symplectic gyrator

Figure 3.9: Transformers and gyrators.

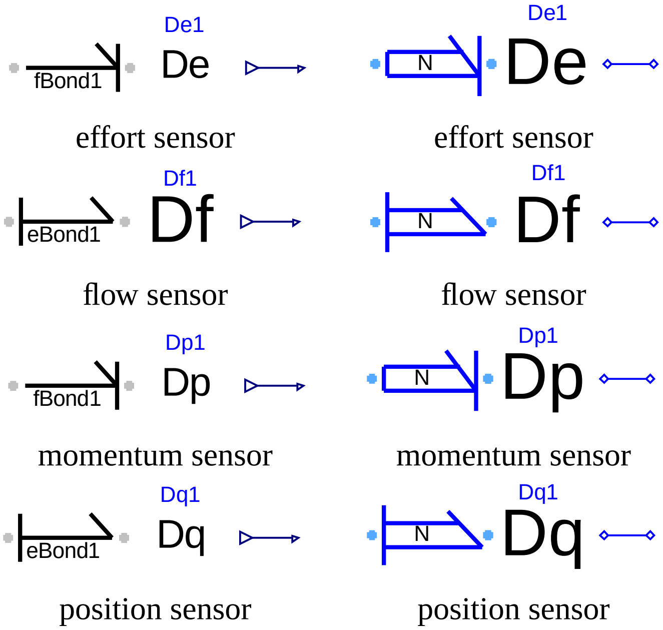
3.4.5 Modulated elements

There exist modulated forms of all sources and passive elements that allow these elements to change their behavior during the simulation. These elements are denoted with the prefix “m”. E.g. a modulated resistance is denoted by the mnemonic symbol mR. The modulating signals are mostly dependent on state and/or time and are acausal.

The modulation of multibond elements can happen in a lot of different ways and it is impossible to foresee all helpful variants. Therefore the development of specific modulated elements for certain applications has to be done in the corresponding libraries. The modulated passive fields (mRF, mGF, mCF, mIF) should be avoided and be (whenever possible) replaced by a composition of a modulated transformer and a corresponding simple passive element.[4]

3.4.6 Sensors

Sensor elements are used to drag information out of the bond graph. Effort, flow, position (integrated effort) and charge (integrated flow) can be measured by the bondgraphic sensors presented below. As it is suggested by the causal multibonds, effort and position should be measured at a 0-junction whereas flow and charge at 1-junctions. The measurement is returned via a signal.



effort sensor effort sensor

flow sensor flow sensor

momentum sensor momentum sensor

position sensor position sensor

Figure 3.10: Sensor elements.

Often these signals are used to modulate another bondgraphic element (as it was shown in figure 2.9). An important fact is that the signals of the MultiBondLib are acausal.[3] Therefore the usage of these sensors is not restricted to measurements and precausalized modulations. They can as well be used to establish algebraic relationships between the bondgraphic elements.

This is important to note, because causal output signals of a charge- or positional sensor would explicitly define integrators. This leads to serious problems if two charges are coupled with each other by an equation relating the corresponding two charge sensors with each other. Such a system has now a non-removable structural singularity. Acausal signals, in contrast, are nothing else than equations and therefore allow the proper handling of structural singularities by the Pantelides algorithm.

[3] The signals of the BondLib are causal signals

3.4.7 Switches

An ideal bondgraphic switch is a 1-port element that is modulated by a boolean signal. If the signal is true, the switch represents a source of zero effort, otherwise it represents a source of zero flow. A non-ideal switch is a similar element that can switch between a small resistance or a small conductance.

Ideal switches change their computational causality. Therefore, the causality of an ideal switch must not be predetermined and it has to be placed in an algebraic loop[18]. Non-ideal switches avoid the causality flip, and can be placed everywhere. The drawback is that these elements always have a small conductance and resistance. This introduces a small error and can make a model needlessly stiff.

Ideal and non-ideal switch elements are provided in the classic BondLib. Also multibondgraphic implementations of them are provided in the MultiBondLib, but their application is barely meaningful. One usually wishes to switch certain variables, not vectors of them. If it is about switching, a solution using single bonds is often better and a decomposition of the multibonds is recommended.

3.5 Additional elements for multibond graphs

3.5.1 Permutation of multibonds

A permutation bond is changing the order of the single bonds in the multibond. It is specified by a permutation vector. A permutation vector is a vector of length n that contains all integers from $1..n$, where n is the cardinality of the multibond. The single bonds are then reordered according to the order of appearance in the permutation vector (e. g. the order of a multibond with cardinality 4 is reversed by the permutation vector: $[4,3,2,1]$).

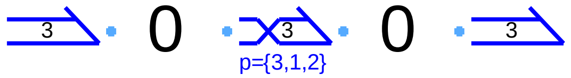


Figure 3.11: The permutation bond is applied between two 0-junctions.

3.5.2 Composition and decomposition

A composition element has been developed to allow the composition of multibonds with different cardinality. It simply merges two multibond graphs so that the multibond at port “a” defines the first elements and the multibond at port “b” the remaining elements of the resulting multibond. In combination with the permutation bond, two multibonds can be composed in any arbitrary fashion. Please note that this is an acausal element and that it can also be used to decompose multibonds.

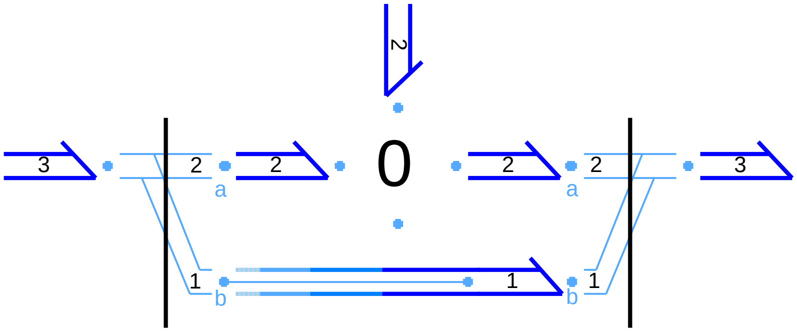


Figure 3.12: Two composition elements are used to decompose a multibond. The model describes a partial 0-junction.

3.5.3 Partial 0- and 1-junctions

Sometimes one wishes to connect only certain components of a multibond without having to do a decomposition of it as in figure 3.12. This can comfortably be achieved by the usage of partial 0- and 1-junctions. These elements are 3-port junctions, where the “intruding” multibond has to be of lower cardinality than the major multibonds. Let *n* be the major cardinality of the junction and let *m* be the smaller intruding cardinality. A specification is needed that defines to which of the *n* major bonds the *m* intruding bonds ought to be connected. This specification is done again by a permutation vector of size *n*. The *m* intruding bonds are then connected via the junction to the first *m* major bonds of the permutation vector. The order of the major bonds remains unaffected.

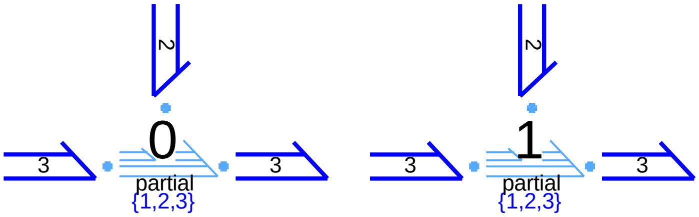


Figure 3.13: Presentation of partial junctions for multi bond graphs.

3.5.4 Conversions between single bonds and multibonds

The multibondgraphic elements define their own connectors, which differ from those of the classic BondLib. For a combination of both notations a conversion element is needed. Because a single bond is simply a multibond with cardinality 1, the idea of a conversion bond is pretty obvious. One side of the conversion bond is seen as classic bond while the other one as multibond of cardinality 1.[4]

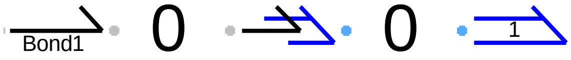


Figure 3.14: A conversion bond is applied between two 0-junctions.

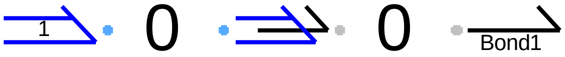


Figure 3.15: The reverse conversion is applied between two 0-junctions.

3.5.5 The default model

The cardinality of a multibond and its connected elements can be defined for each single element. Therefore it is possible to create models containing multibonds of different cardinality without any additional effort. But it would become tedious, if the cardinality would have to be defined for each single element. Hence, a default model has been developed that defines the default cardinality. This model has to be part of all multibondgraphic models, because all basic elements expect an instantiation named “MBG_defaults” as an outer model. [5]

[4] Some of the the second part’s multibondgraphic models for mechanics contain 1-dimensional subparts that could be modeled by classic bond graphs. But to avoid unnecessary and irritating conversion, these models are consistently drawn in the multibondgraphic terminology and the appearance of multibonds with cardinality 1 is tolerated.

[5] more about outer models in the Modelica language specification or in the Modelica tutorial

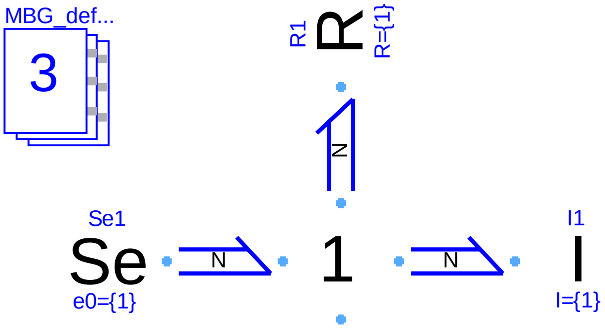


Figure 3.16: Application of the default model. The default cardinality is here chosen to be 3, and its usage is denoted by a large N in the middle of the multibonds.

3.6 Conclusions

The MultiBondLib provides a general solution for multibondgraphic modeling. The graphical modeling environment of Dymola enables the user to comfortably compose multibond graphs on screen. Applications and examples will be presented in the next part of this thesis.

Chapter 4

Bondgraphic Modeling of Mechanical Systems

4.1 Mechanic bond graphs

Already at the introduction of the bondgraphic methodology 1961 by Paynter[19], its application to mechanical systems had been pointed out. After all, Paynter himself was a professor of mechanical engineering at M.I.T.

For translational mechanics, the effort variable is defined to be the force and the flow variable is defined to be the velocity.[1] This choice is maybe not the best. Usually the effort variable expresses an intensive characteristic of an object that is independent of its quantity and the flow represents an extensive characteristic that is proportional to the object's quantity (e.g. two cars can't drive faster than one car but they can pull with twice the force). Therefore it might have been a better choice to take the velocity as effort and the force as flow variable. Unfortunately the standard definition of the bondgraphic community is a different one, and we don't want to violate it.

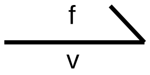


Figure 4.1: A mechanic bond.

The meaning of the basic bondgraphic elements is now defined accordingly. The bondgraphic inductance is a storage of kinematic energy and models a mass element. The storage of potential energy is accordingly the capacitance, which can be used to model a spring. A linear bondgraphic

[1] see table 2.1 at page 22

resistor models an ideal damper. These elements can be rigidly connected using a 1-junction, whereas a force interaction between two elements is modeled with a 0-junction.

Fig 4.2 shows a simple one dimensional mass spring system and its corresponding bond graph.

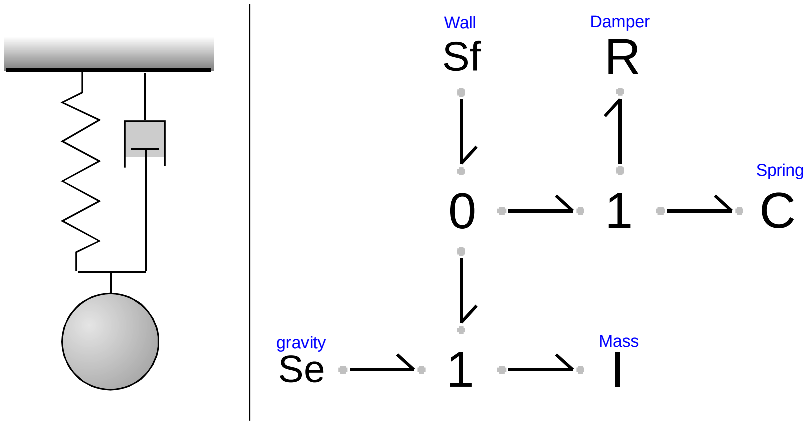


Figure 4.2: A mass hanging on a spring-damper in a uniform gravity field.

Bond graphs for rotational mechanics are defined accordingly. The torque t is the effort variable and the angular velocity ω is the flow variable. A bondgraphic inductance represents the inertia, a capacitance represents a twirl spring and a resistance represents a rotational damper.

4.2 Previous achievements

The extension to a multibondgraphic methodology happened quite early and the results are nicely discussed in the Ph.D. dissertation[4] of P. C. Breedveld. Although this dissertation does hardly mention mechanics (actually not at all) it is extremely valuable and easy to read.

Based upon the concept of multibond graphs A. M. Bos[3] developed basic mechanical models and applied his knowledge to model a motorcycle. He used separate multibonds for the translational and rotational domains. Also the complex model is composed of simpler models with the old-fashioned method of word bond graph[17], which is a predecessor of todays wrapping technique[7].

In the well known book “System Dynamics”[16] the authors propose a different approach by summarizing the rotational and translational variables into one multibond of cardinality 6 and they present this shortly at some basic models.

The paper[28] of A. Zeid offers a comprehensive set of models. It contains a library of three-dimensional joints for multibody systems. Unfortunately, these models are completely based upon single bond graphs. A multibondgraphic library of models for planar and three-dimensional multibody systems is provided in the book "Mechatronics by Bond Graphs" [8]. The book also presents models of impacts and friction. Bondgraphic switches are used to model the transition from slippage to adhesion. The impact models contain spring-damper systems. These are so-called weak solutions. Similar models for impacts are presented in the conference proceeding "Bond Graph Simulation of Vehicle Collisions"[1].

4.3 Contributions of this work

Most of the models and solutions mentioned above have the major drawback that they were developed for outdated simulation systems. These programs were often limited in their abilities and did not support object-oriented modeling. Especially for mechanical models, which tend to become very large, we do miss the concept of object orientation.

The simulation program Dymola and the modeling language Modelica feature object-oriented modeling techniques in a fully sufficient way. The MultiBondLib is a powerful bondgraphic modeling tool and enables a convenient modeling of physical processes. In addition, the MultiBondLib enables a combination with other non-bondgraphic modeling tools via the sensor elements. This is of major importance, because many mechanical systems contain equations that establish relations between the positional variables. These are so-called holonomic constraint equations and they result out of a process of idealization. It is important to note that there is no physical law that states such constraint equations. Hence, mechanical systems are not pure physical systems and cannot be sufficiently modeled by pure bond graphs. Further (graphical) modeling tools are needed.

This thesis presents solutions for a fully object-oriented modeling of mechanical systems in two or three dimensions based upon multibond graphs. It introduces additional bondgraphic and non-bondgraphic elements that are specific for mechanical systems. The resulting set of models is easy to understand and can be used in an intuitive way. The resulting mechanical systems can be efficiently simulated.

The weak modeling of impacts also proves to be insufficient for certain applications. Chapter 8 presents hybrid models that allow discrete changes of motion to be modeled that occur as a result of hard ideal impacts.

Part II

Mechanics

Chapter 5

The Principles of Mechanics in their Development

5.1 Introduction

This chapter is a small essay about the development of the principles of mechanics. Because the fundamental equations of mechanics are expected to be familiar to most of the readers, a textbook-like presentation of them seemed to be of no value to me. An inspection from the historical viewpoint reveals some interesting facts, makes us learn about our greatest scientists and lets us distinguish more clearly their contributions. Going back to the early stages of development is also a good opportunity to become clear about the underlying assumptions and important axioms.

The writing of this chapter offered the chance to me to have a look at some of the old texts, but of course a professional interpretation and examination of them is necessary. Hence this chapter refers mainly to two books: “Die Mechanik in ihrer Entwicklung”[20] from Ernst Mach and “Die Geschichte der mechanischen Prinzipien”[27] from Istvan Szabo. There is not a lot that has to be told about the famous book of Ernst Mach. It has become an important document by now. The book of Szabo is a great book for all those, who learned mechanics in school or university without its historical background and want to learn about it now.

5.2 Theories on impacts

Theories on impacts were made, before the principals of mechanics were developed. So for a long time, they formed a separate concept. Starting from the early work of Galileo Galilei the most important contributors to this field of science were: Marcus Marci(1595-1667), John Wallis (1616-1703), Christopher Wren (1632-1723) and Christiaan Huygens (1629-1695). Marcus Marci noticed the importance of the object's material and stated equations to compute the velocity of two bodies after an fully elastic impact, but not yet in full generality.

This work was then partially redone by Wallis and Wren. It was finally Christiaan Huygens who summarized all the previous and his own results most impressively by recognizing that the movement of the total barycenter isn't influenced by impacts. He stated the law of conservation of momentum.

$$\sum_i m_i v_i = const. \tag{5.1}$$

He also observed the equation for what is today expressed as conservation of kinetic energy by an elastic impulse, but of course without fully recognizing its relevance. So Huygens law to find the velocity after an elastic impulse formed an additional law: The difference of velocity between two bodies is reversed after an elastic impulse.

$$v_{1,post} - v_{2,post} = -(v_{1,pre} - v_{2,pre}) \tag{5.2}$$

Also Newton contributed to this field. He cared about non-perfect elastic impacts and introduced the elasticity coefficient for semi-elastic materials. Therefore Huygens equation 5.2 is extended to:

$$v_{1,post} - v_{2,post} = -\epsilon(v_{1,pre} - v_{2,pre}) \tag{5.3}$$

where ϵ is the elasticity coefficient between 1 (elastic) and 0 (totally non-elastic).

Today, all these laws are derived out of the basic principles of mechanics.

5.3 Newton's principia

Sir Isaac Newton (1643-1727) finished in 1686 his first edition of his “PhilosophiæNaturalis Principia Mathematica”, shorter known as “Principia Mathematica”. This text is world famous, often “quoted” and barely read. It contains the well known three axioms of motion, that are presented here in today's most common form of writing.

First Axiom: The principle of inertia. A body at rest remains at rest and a body in motion remains in motion at a constant velocity as long as outside forces are not involved.

$$v = 0 \texttt{ iff } f = 0 \tag{5.4}$$

Second Axiom: Motion defined in terms of mass and acceleration:

$$f = m \cdot a \tag{5.5}$$

Third Axiom: Action and reaction. This famous law states that for every action there is an opposite and equal reaction.

$$f_{12} = -f_{21} \tag{5.6}$$

A contemporary reader will be surprised not to find the famous second axiom: $f = m \cdot a$ anywhere in the principia, not in words and not in terms of equations. Nevertheless, it is neither a wrong nor a polite insinuation to call Newton the founder of this law. Mostly (but incorrectly) Newton's law is directly derived out of Lex II in the Principia:

"A change in motion is proportional to the motive force impressed and takes place along the straight line in which that force is impressed."[23]

One possible translation of this law is: $f \cdot t = m \cdot v$, which is the integrated form of $f = m \cdot a$. Unfortunately other equations like: $f \cdot t^n = m \cdot a$ $(\forall n \neq 1)$ do satisfy Lex II as well and therefore Newton's law can't be derived out of Lex II, because it does not state that a constant force leads to a constant acceleration. This is obviously a lack of precision in Lex II.

Fortunately Newton adds the necessary precision later on in his text by stating:" *The spaces which a body describes when urged by any finite force whether that force is determinate and immutable or is continually increased or continually decreased, are at the very beginning of the motion the squared ratio of the times.*"[23]

This important statement is strangely hidden in lemma X and it implies the fact that a constant force is leading to a constant acceleration and we can now derive: $f = m \cdot a$. In fact this lemma X is even more powerful than Lex II, because it also includes a description for the case that non-constant forces are acting on a body.

5.4 The physics of rigid bodies in space: Euler's equation

Leonhard Euler (1707-1783) made essential contributions in many mathematical fields such as: analysis, algebra, geometry, graph theory and number theory. Beside of this already tremendous work he contributed as well in the field of mechanical problems. Indeed one of his first works concerned the ideal placement of masts in a sailing ship.[9]

Euler extended the laws of motion to further dimensions and to the rotational domain. This work was stretched over decades and finally concluded 1775 in the fundamental equations for the motion of rigid bodies:

$$P = \int dm \cdot \ddot{x}\,;\;\; Q = \int dm \cdot \ddot{y}\,;\;\; R = \int dm \cdot \ddot{z}\,; \tag{5.7}$$

$$S = \int \left(r_y \cdot \ddot{r}_z - r_z \cdot \ddot{r}_y\right) dm\,;\;\; T = \int (r_z \cdot \ddot{r}_x - r_x \cdot \ddot{r}_z) dm\,;\;\; U = \int \left(r_x \cdot \ddot{r}_y - r_y \cdot \ddot{r}_x\right) dm\,; \tag{5.8}$$

where $[P,Q,R]^T$ forms the force vector **f** and $[S,T,U]^T$ represents the torque vector $\mathbf{t} = \mathbf{r} \times \mathbf{f}$.

It is important to note that these equations can't be derived directly out of Newton's principles.[1] Euler's work is also by no means only a summation and presentation of Newton's principles in a proper mathematical form as it is implicitly insinuated by Ernst Mach:

"Newtons principles are sufficient to understand every mechanical problem that is arising in practice without the addition of further principles, no matter if the problem is of static or dynamic nature. If there are any difficulties, these are only of mathematic (formal) nature and not of principal nature"[2]

In fact equations 5.7 and 5.8 form a separate law in physics and do necessarily imply, as Ludwig Boltzman noticed, the symmetry of the volume element's stress tensor.[3]

[1] as this is incorrectly stated in many text-books and lecture scripts

[2] Author's translation of the original text: "Die Newtonschen Prinzipien sind genügend, um ohne Hinzuziehung eines neuen Prinzips jeden praktisch vorkommenden mechanischen Fall, ob derselbe nun der Statik oder der Dynamik angehört, zu durchschauen. Wenn sich hierbei Schwierigkeiten ergeben, so sind dieselben immer nur mathematischer (formeller) und keineswegs mehr prinzipieller Natur"[20]

[3] A proper explanation of this issue can be found in: [27]

5.5 D'Alembert's principle

D'Alembert's principle describes a supporting method to obtain the correct equations of a dynamical system where the movement of the single elements underlies certain constraints, like in the case of a pendulum or in a system of rigidly connected bodies. Often this principle is "explained" at the example of a freely moving body, where it is completely meaningless.

The original principle of Jean Baptiste le Rond d'Alembert (1717-1783) decomposed a body's motion into two parts as illustrated in figure 5.1 at the example of a pendulum. The first part of the motion is the motion the body "wants" to make according to its own (gravitational) forces. The second part is the motion that is necessary to bring the body back on its constrained trajectory. The virtual forces necessary to do this have then to cancel out each other over the whole system.

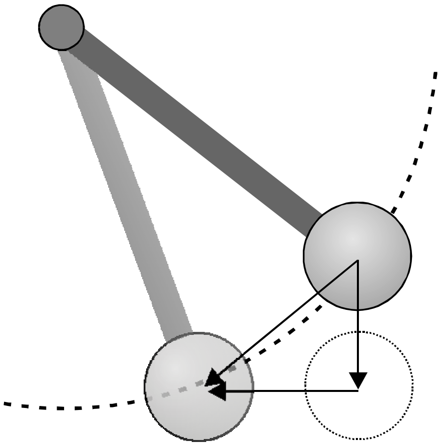


Figure 5.1: Decomposition of a pendulum's constrained motion.

Today's form of usage results of a modification of the original principle by J. L. Lagrange, which definitely simplifies the affair. He defined a reduced force that results out of the difference between the own (gravitational, directly applied) force $\mathbf{f}_e$ and the finally resulting acceleration $\mathbf{a}$.

$$\mathbf{f}_{red} = \mathbf{f}_e - m * \mathbf{a}$$

He then proposed the equilibrium statement, so that all reduced forces have to cancel out each other:

$$\sum \mathbf{f}_{red} = 0$$

This method provides a good way to find a proper set of equations for a dynamical system. In fact, d'Alembert's law is the foundation of any object-oriented formulation for mechanical systems. E.g. in a bond graph, d'Alembert's law is implicitly modeled by 1-junctions.

5.6 Fundamental equations of motion

Todays form of usage can be derived out of Euler's fundamental form, if we introduce the vector notations $\mathbf{x} = [x, y, z]^T$ for the translational part and $\mathbf{r} = \left[r_x, r_y, r_z\right]^T$ for the rotational part. The equations 5.7 and 5.8 can now be written in vectorized form:

$$\mathbf{f} = \int dm \cdot \ddot{\mathbf{x}} = m \cdot \dot{\mathbf{v}} = m \cdot \mathbf{a} \tag{5.9}$$

$$\mathbf{t} = \int \mathbf{r} \times \ddot{\mathbf{r}} dm = \int \mathbf{r} \times \dot{\mathbf{v}}_{\mathbf{r}} dm \tag{5.10}$$

In a rigid body $\mathbf{v_r}$ is defined to be $\boldsymbol{\omega} \times \mathbf{r}$. The angular velocity is a constant for all volume-mass elements and can therefore be extracted out of the integral:

$$\mathbf{t} = \int \mathbf{r} \times \dot{\mathbf{v}}_{\mathbf{r}}\, dm = \int \mathbf{r} \times (\dot{\boldsymbol{\omega}} \times \mathbf{r})\, dm = \int diag(\mathbf{r})^2 - \mathbf{r}\mathbf{r}^T dm \cdot \dot{\boldsymbol{\omega}} \tag{5.11}$$

$\mathbf{z}$ is defined to be the angular acceleration $\dot{\boldsymbol{\omega}}$ and the inertia tensor $\mathbf{J}$ is also by definition:

$$\mathbf{J} = \int diag(\mathbf{r})^2 - \mathbf{r}\mathbf{r}^T dm \tag{5.12}$$

Now the motion of a rigid body can easily be described by the following resulting equations:

$$\mathbf{f} = m \cdot \mathbf{a} \tag{5.13}$$

$$\mathbf{t} = \mathbf{J} \cdot \mathbf{z} \tag{5.14}$$

All these equations hold true for an unaccelerated inertial system. As shall be shown later on, it is not always convenient to compute everything in the inertial system. Therefore a further development of the equations will have to be made.

Chapter 6

Planar Mechanics

6.1 Introduction to planar mechanics

In planar mechanics the world is restricted to two dimensions and only a subpart of the fundamental equations for movement is relevant. Models for planar mechanic systems are a lot simpler than three dimensional models and do have mostly the characteristics of toy models. Still, these models can be interesting and are definitely a good introduction to multidimensional mechanics as they offer an easily understandable approach to more complex problems. Hence, this chapter has been mainly inserted for didactical reasons. The structure of the library and important concepts like wrapping, and the correct closing of kinematic loops can be introduced without having to struggle with the specific problems of 3D-mechanics.

6.1.1 Notations

A body's position and orientation is completely defined by 3 variables:

$$\mathbf{x} = [x, y, \varphi]^T$$

where x and y denote the translational position and φ is the current rotational position. The derivatives of the positional variables are then written as follows:

$$\dot{\mathbf{x}} = [v_x, v_y, \omega]^T$$

$$\ddot{\mathbf{x}} = [a_x, a_y, z]^T$$

And accordingly we denote the forces elements:

$$\mathbf{f} = [f_x, f_y, t]^T$$

The relevant fundamental equations can be drawn out of chapter 5.6 page 52. They are:

$$f_x = m \cdot a_x$$

$$f_y = m \cdot a_y$$

$$t = J \cdot z$$

One major facilitation in planar mechanics is the fact that the rotational inertance J is a constant scalar for all possible orientations because the axis of rotation is fixed. Therefore everything can be comfortably computed using the coordinate vectors of the inertial system. There is no need to transform between different coordinate systems as in 3D-mechanics.

6.2 Basic multibond graphs for planar mechanics

Because all potential effort and flow variables of a planar bond graph can be comfortably resolved with respect to the inertial system, we summarize them all into one multibond. A multibond's composition is consequently defined: The first two bonds belong to the translational domain whereas the third one is of the rotational domain. A multibond's effort vector is then $[f_x, f_y, t]^T$ and the corresponding flow vector: $[v_x, v_y, \omega]^T$. The composition is illustrated in figure 6.1.

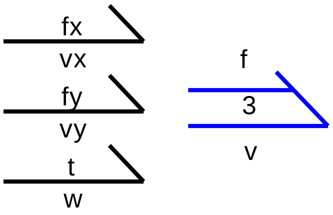


Figure 6.1: The composition of a mechanical multibond.

If the multibond couples elements of two domains, so do the bondgraphic elements. For example, the bondgraphic inductance models now both mass and inertia with the diagonal matrix, defined by its diagonal vector $[m, m, J]^T$ The mechanical interpretation of the bondgraphic elements will now be explained by going through some basic examples:

6.2.1 A freely moving body in a uniform gravity field

This is a very basic example. The storage of kinetic energy by mass and inertia is modeled by a single bondgraphic inductance, as stated above. The uniform gravity is applied by a source of effort, where the third effort element (which represents the torque) is zero. The position of the object is then measured by a bondgraphic position sensor. All these elements are interconnected by a 1-junction that denotes equality of velocity.

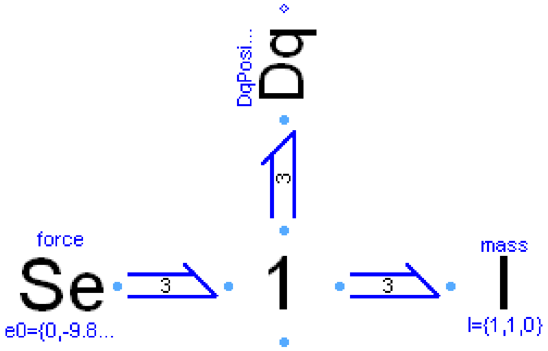


Figure 6.2: The multibond graph of a body moving in space.

6.2.2 A planar pendulum

Even though the rotational domain was correctly modeled, it was superfluous in the last example. Figure 6.3 shows a more complex example, where the rotational domain is relevant. This is a bondgraphic model of a planar pendulum. The translational position is fixed at the revolute joint by a source of zero flow. The rotational movement is free and therefore modeled by a source of zero effort, because no torque can be applied to a revolute joint. The angular position φ is measured by a sensor.

The body is modeled like in the first example and is connected to the revolute joint by a massless rod describing a positional translation between the body element and the revolute joint. This

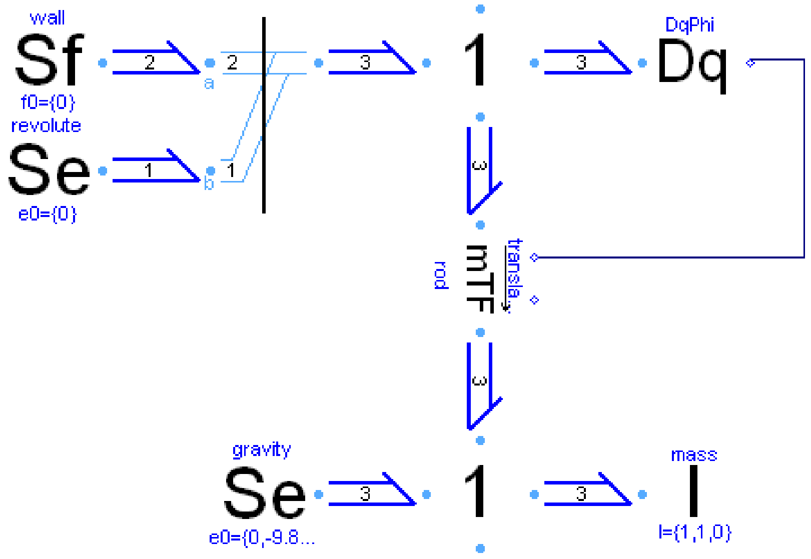


Figure 6.3: Multibond graph of a planar pendulum.

rod is modeled by a transformer that transforms the angular velocity into translational velocity in dependence on the translation vector **r**. This transformation is according to the balance equations of a lever:

$$\begin{pmatrix} v'_x \\ v'_y \\ \omega' \end{pmatrix} = \begin{bmatrix} 1 & 0 & r_y \\ 0 & 1 & -r_x \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} v_x \\ v_y \\ \omega \end{pmatrix}$$

and in consequence:

$$\begin{pmatrix} f_x \\ f_y \\ t \end{pmatrix} = \begin{bmatrix} 1 & 0 & r_y \\ 0 & 1 & -r_x \\ 0 & 0 & 1 \end{bmatrix}^T \begin{pmatrix} f'_x \\ f'_y \\ t' \end{pmatrix}$$

The transformer in the bond graph is a modulated by the revolute angle φ because **r** is dependent on this variable:

$$\mathbf{r} = \begin{pmatrix} r_x \\ r_y \end{pmatrix} = \begin{bmatrix} cos(\varphi) & sin(\varphi) \\ -sin(\varphi) & cos(\varphi) \end{bmatrix} \begin{pmatrix} r_{0x} \\ r_{0y} \end{pmatrix}$$

where r_0 is the original translation at $\varphi = 0$.

The result of the simulation can be seen in the next figure.

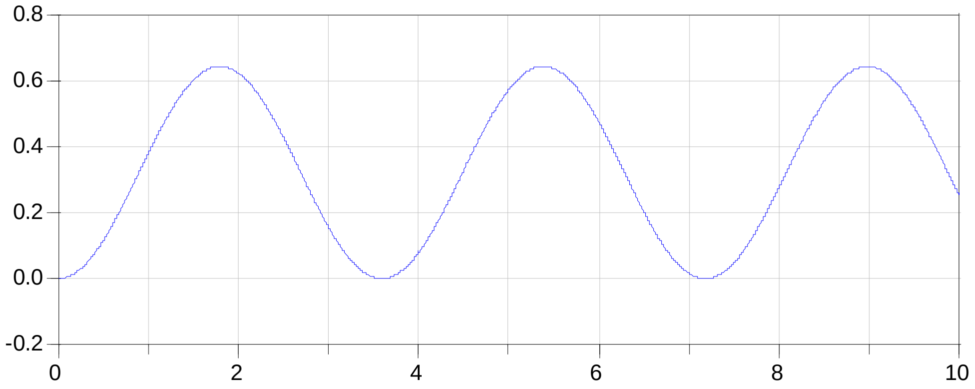


Figure 6.4: The angle φ of the pendulum for a period of 10 seconds.

6.2.3 Additional multibondgraphic elements

The transformer of the last example is a bondgraphic element that has been developed especially for the usage in the planar mechanics library. It is called translational transformer and implements the balance equations of a lever and can be modulated by the orientation angle and an optional length amplification. It is very helpful to model massless rods or prismatic joints.

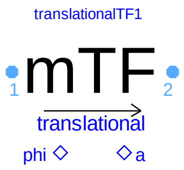
Another additional multibondgraphic element is the prismatic transformer. It projects the one dimensional flow at port a onto two dimensions at port b. The projection is specified by the projection vector and its current orientation.

In addition there is the need for an extra element that handles the positional signals. The translation element allows to shift a position and an orientation. The usage of this element can be seen in another example, at page 60.

These three additional elements are sufficient for the most important models in planar mechanics. In 3D-mechanics a larger number of additional elements will be needed.

6.2.4 The hidden capacitance

In a swinging pendulum there is a continuous interchange between potential and kinematic energy. Anyhow it is not possible to drag this interchange directly out of the bond graph of figure 6.3, because there is only one storage element: the inductance that stores the kinematic energy. We miss the second storage element for the potential energy: the capacitance.



Name: translationalTF

Location: MultiBondLib.planarMechanics.AdditionalMBG

Parameters: translation vector **d**.

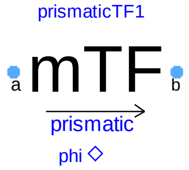
Modulating signals: orientation angle φ.
optional amplification signal a.

Equations:

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & a \cdot [-sin(\varphi), +cos(\varphi)] \cdot \mathbf{d}^T \\ 0 & 1 & a \cdot [-cos(\varphi), -sin(\varphi)] \cdot \mathbf{d}^T \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{f}_2 = \mathbf{M} \cdot \mathbf{f}_1$$

$$\mathbf{e}_1 = \mathbf{M}^T \cdot \mathbf{e}_2$$



Name: prismaticTF

Location: MultiBondLib.planarMechanics.AdditionalMBG

Parameters: projection vector **d**.

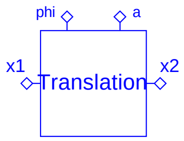
Modulating signals: orientation angle φ.

Equations:

$$\mathbf{M} = \begin{pmatrix} cos(\varphi) & sin(\varphi) \\ -sin(\varphi) & cos(\varphi) \end{pmatrix} \cdot \mathbf{d}$$

$$\mathbf{f}_b = \mathbf{M} \cdot \mathbf{f}_a$$

$$\mathbf{e}_a = \mathbf{M}^T \cdot \mathbf{e}_b$$



Name: Translation

Location: MultiBondLib.planarMechanics.AdditionalMBG

Modified signal: position signal.

Modulating signals: optional amplification signal a .
optional relative rotation angle ψ.

Equations:

$$\mathbf{r} = a \cdot \begin{pmatrix} cos(\varphi_1) & sin(\varphi_1) \\ -sin(\varphi_1) & cos(\varphi_1) \end{pmatrix} \cdot \mathbf{d}$$

$$x_2 = x_1 + r_x$$

$$y_2 = y_1 + r_y$$

$$\varphi_2 = \varphi_1 + \psi$$

Figure 6.5: Special multibondgraphic elements for planar mechanics.

The capacitance is there, but it is hidden. P. C. Breedveld has pointed out rightly that it is always possible to model any storage element as state modulated effort- or flow source.[4] This is exactly what is done here. It does not matter whether the modulation is applied directly on the source element or through a transformer. Figure 6.6 reveals the capacitance of the pendulum bond graph. For small deviation angles, it can also be replaced by a linear capacitance. A position modulated effort source is mostly nothing else then a (non-linear) capacitance. Actually all capacitive influences in this thesis' mechanical models are modeled in this way, because it is the velocity of an object that we would like to integrate not its force.

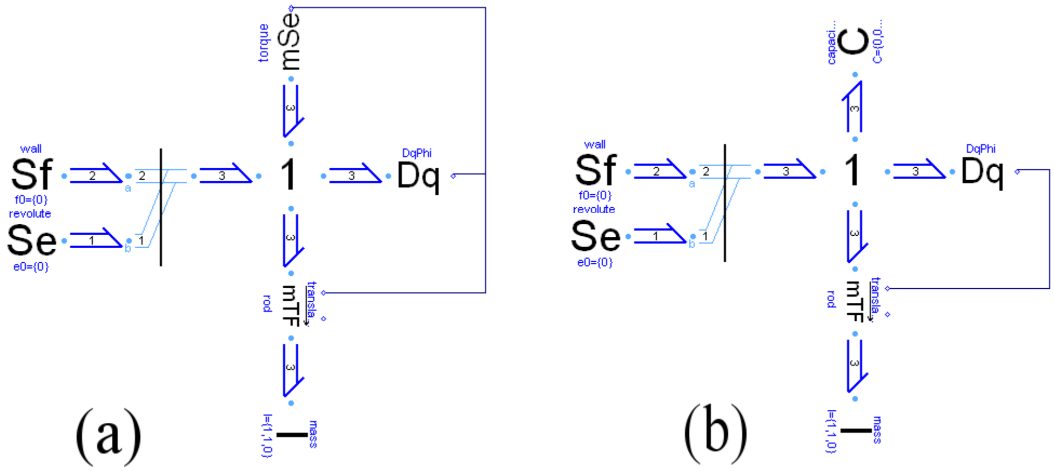


Figure 6.6: Revealing the hidden capacitance. In (a) the modulation of the effort source is made visible. In (b) the modulated effort source is replaced by a linear capacitance, which is a valid approximation for small angles of φ.

6.2.5 The free crane crab

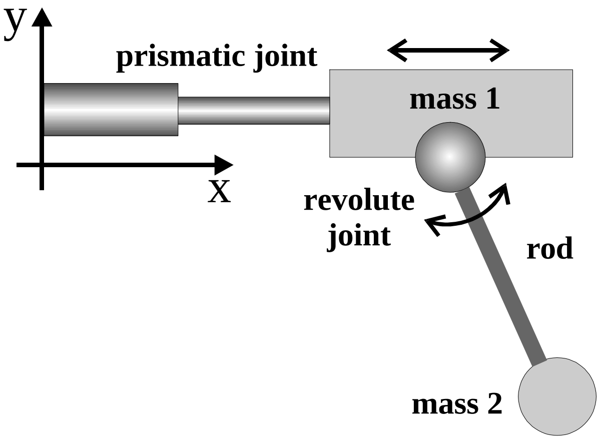


Figure 6.7: Basic drawing of a free crane crab.

We are now ready to enter a more complex example: A free crane crab. The crane crab has a mass m_1 and can move in only one dimension. The load attached to the crane crab is modeled as a simple pendulum with length l and mass m_2. Such a system has two degrees of freedom on positional and velocity level and can be modeled by the bond graph of figure 6.8.

The restriction on the crab's translational movement is modeled by a prismatic joint. This is a rod

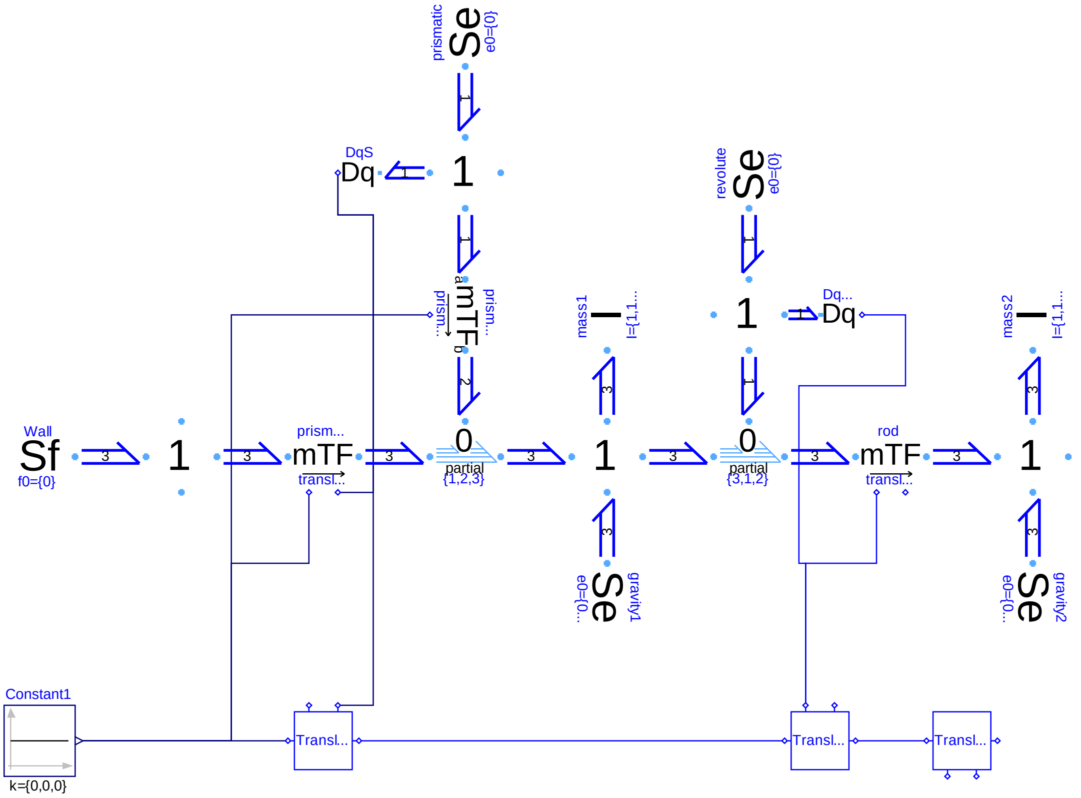


Figure 6.8: Bond graph of a free crane crab.

with variable length s. The transformation due to the translation is dependent on the rod length s and the the rod's orientation φ (which is constant in this example). A more detailed explanation of the prismatic element is given in section 6.3.3, page 68. Anyhow the reader is not expected to fully understand this model at this point.

6.2.6 Wrapping

As observable, bondgraphic models for mechanical systems tend to become quite big for even small examples. Hence, we'd like to replace meaningful subparts of the bond graph by single objects, so that in consequence the graphical model is more readable. This is what actually is called wrapping.

The wrapping technique is now demonstrated at the example of the free crane crab. The meaningful subparts are torn apart by the insertion of additional 1-junctions and are then replaced by the corresponding objects. This procedure is illustrated in figure 6.9 and the resulting model is easy to understand. An additional drawing of the model becomes almost unnecessary. How the wrapping is precisely done and how to model the wrapped elements is the subject of the next section, where a library for planar mechanics is presented.

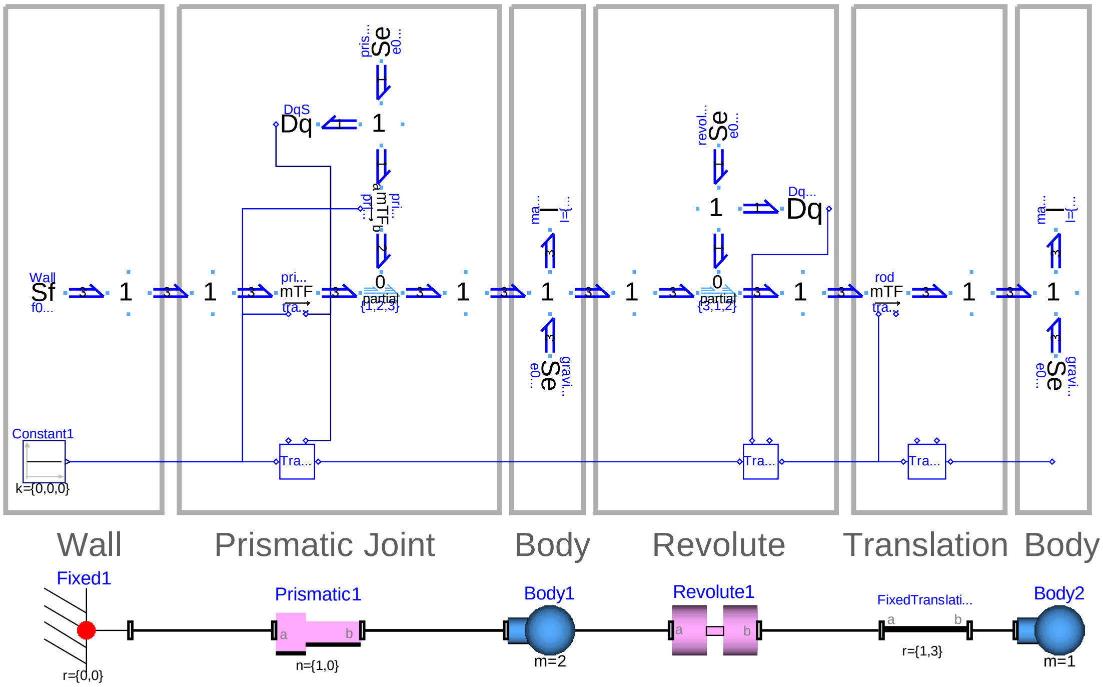


Figure 6.9: Illustration of the wrapping process. The subparts are marked by rectangular frames and are then replaced by single objects that are represented by meaningful icons. These objects contain the corresponding bondgraphic models on a lower level.

6.3 The PlanarMechanics library

6.3.1 Introduction

This library contains models of basic and ideal planar mechanical elements and enables the user to model complex mechanical systems in a object-oriented way. The basic elements are body elements that define mass and inertia, but also massless elements like revolute joints or prismatic joints. Other massless elements are a massless rod or fixation elements. Models of force elements that create non-rigid connections like a spring or a damper are also included in this library. These single elements are represented by single meaningful icons and organized into different subpackages. A structural overview of the PlanarMechanics library is presented in appendix D.

The structure and functionality of this library is very similar to the standard MultiBody library of M. Otter. But in contrast to the MultiBody library the basic elements here are not based upon equations, they are based upon wrapped multibondgraphic models. A closer look at most of these models will give you a bondgraphic explanation of the element's behavior.

All basic elements feature a suitable visualization to enable an animation of the whole mechanical system. The properties of the visual objects can be changed by parameters. The default coloring slightly differs from the coloring in the MultiBody to allow a better distinction of the two libraries.

6.3.2 Wrapping

To successfully wrap bondgraphic models, a new connector type has to be developed that transmits all necessary variables between the single elements. To determine which variables have to be part of the object's frame connector, we can simply look at the variables crossing the borderlines of figure 6.9. These are the bondgraphic effort and flow variables as well as the positional signals.

Because the subparts are always connected by a 1-junction, the variables:

- $[x, y, \varphi]^T$
- $[v_x, v_y, \omega]^T$

are potential variables of the frame connector whereas the variables:

- $[f_x, f_y, t]^T$

are flow variables of the frame connector.

The new connector type is called "Frame" and is used in two variants "frame_a" and "frame_b" to allow a proper distinction between the two connectors of a 2-port element[1]. Please note that there is a redundancy in the potential variables of the connector. The position and the velocity of an object are transferred, even though the velocity can be computed out of the position because it is its derivative. This redundancy will be of importance in the case of kinematic loops. But it doesn't matter as long as the rigid connections form a tree structure.

Together with the new connector, a converter element has been designed that transforms the connector variables into bondgraphic connectors and positional signals and vice-versa. Now the

[1] This is helpful for the parameter definition (e.g. by the definition of parameter vector pointing from frame_a to frame_b)

multibondgraphic models can be connected via the converter and the new frame connector. The wrapping technique is now comfortably applicable.

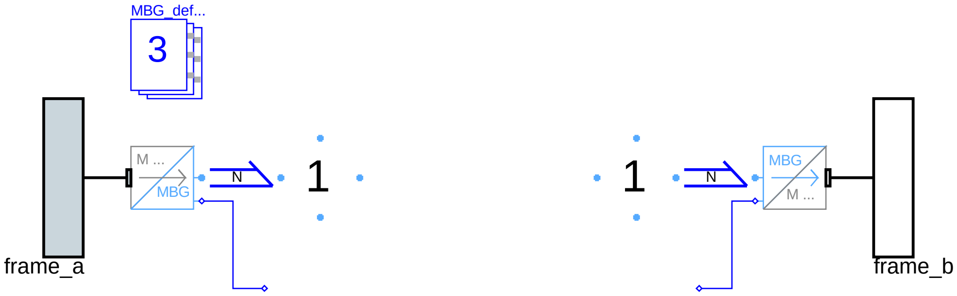


Figure 6.10: The frame connectors in combination with the converter elements form a shell for the bondgraphic model. The bondgraphic model is indicated by 1-junctions. This is the shell of an element with two frame connectors, e.g., a joint.

6.3.3 Basic elements

In this subsection, the bondgraphic models for the most basic planar mechanical elements will be presented.

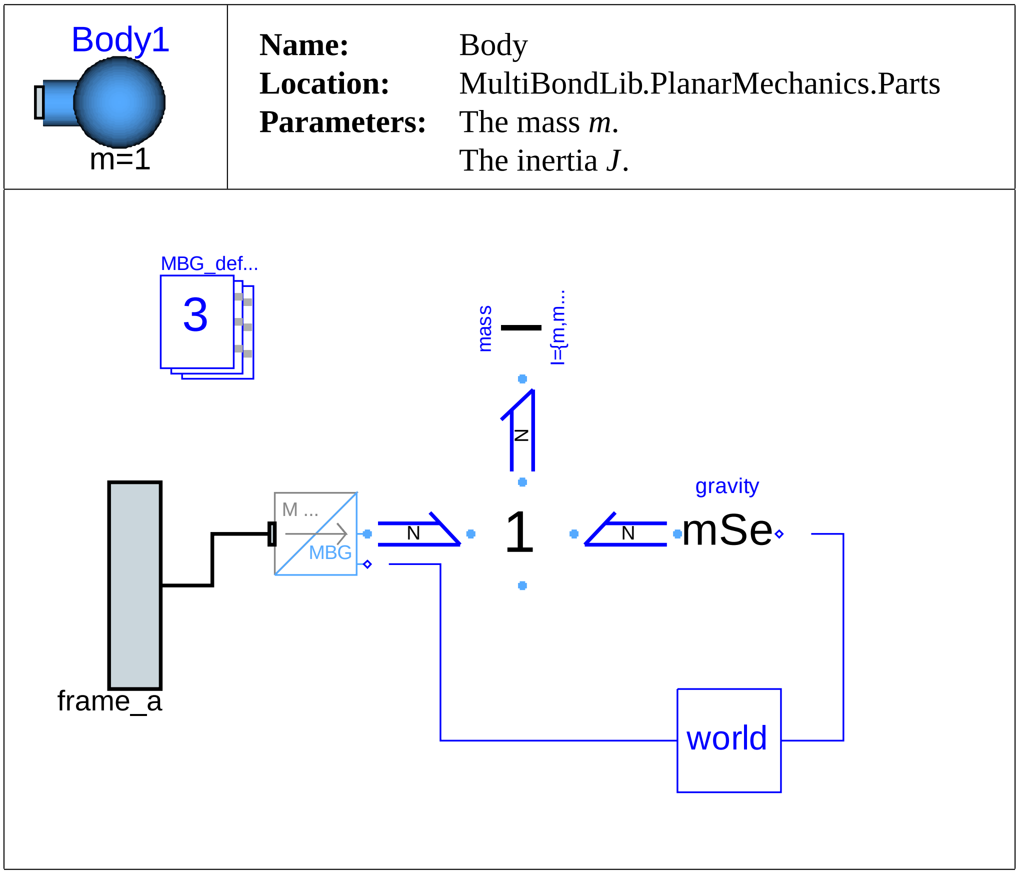
The PlanarWorld model
There is a general outer model for planar mechanical systems called PlanarWorld. This element is used in all mechanical models and must therefore be present in all planar models. It provides several functionalities:

- It models the gravity acceleration field .
- It visualizes the coordinate system.
- It sets default parameter values (e.g. default animation parameters).

This element is an almost identical copy of the “world” model in the MultiBody library.

Body

The model of a body is very similar to that presented in figure 6.2. The main difference is that the source of effort for the gravitational force isn't constant anymore. As suggested by the drawings it is dependent on its position in the world's gravity acceleration field. This dependence is modeled by the planarWorld model.

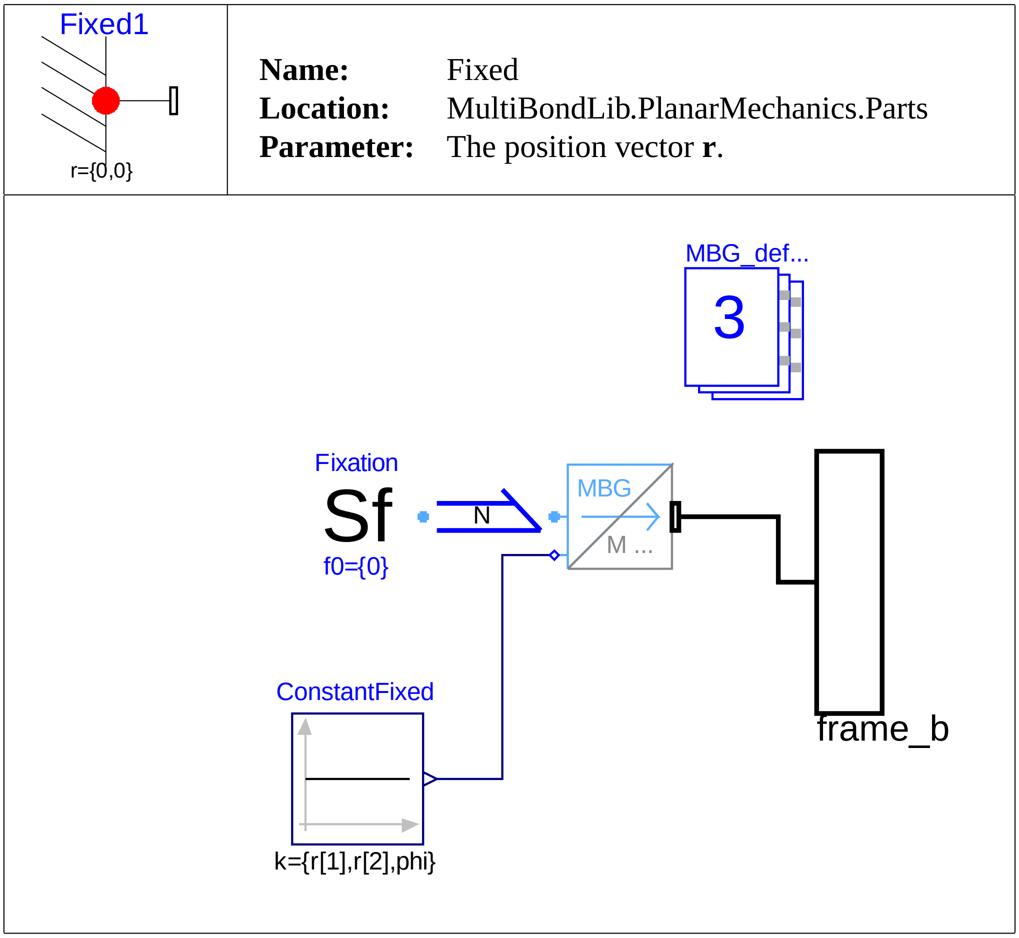


Name:	Body
Location:	MultiBondLib.PlanarMechanics.Parts
Parameters:	The mass *m*. The inertia *J*.

Figure 6.11: Presentation of the body model.

Wall

The wall model represents simply a fixation of the connected objects and therefore is nothing but a source of zero flow. The position can be chosen by a parameter.

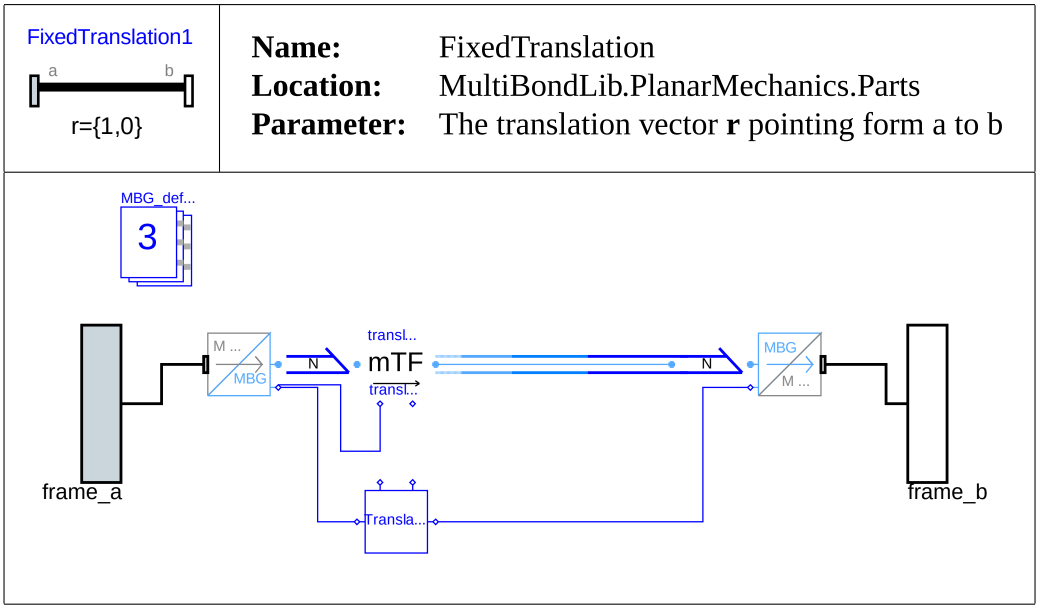


Name:	Fixed
Location:	MultiBondLib.PlanarMechanics.Parts
Parameter:	The position vector **r**.

Figure 6.12: Presentation of the wall model.

Fixed translation

The translation object models a rigid massless rod defined by a vector **r**. The velocity is transformed in dependence of the rotational angular position exactly like in the pendulum example. The difference of the two positional signals is modeled by the special element “Translation”.



Name: FixedTranslation
Location: MultiBondLib.PlanarMechanics.Parts
Parameter: The translation vector **r** pointing form a to b

Figure 6.13: Presentation of the model for a fixed translation.

Revolute joint

A revolute joint adds one degree of freedom to the system and therefore defines one integrator. This is done by the sensor element Dq of the model. The relative angle φ is the integrated angular velocity ω of the revolute joint and added to the rotational position signal. The source of zero effort means that no torque can be transferred through a revolute joint. The translational domain remains unaffected.

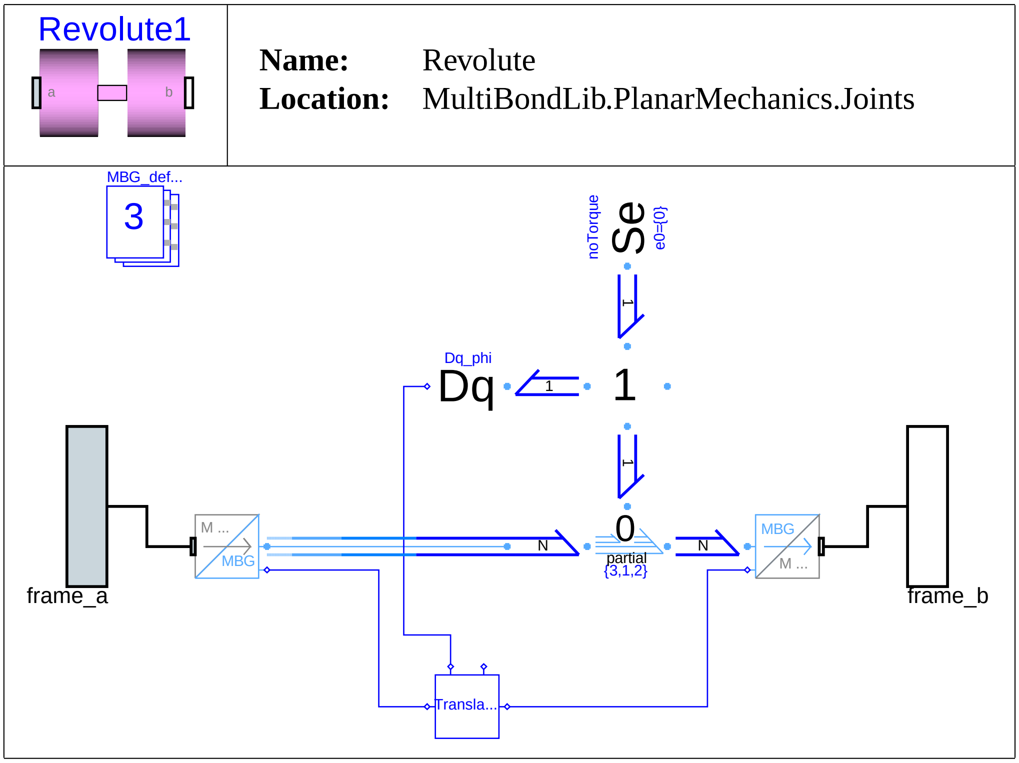

Name: Revolute
Location: MultiBondLib.PlanarMechanics.Joints

Figure 6.14: Presentation of the revloute joint.

Prismatic joint

A prismatic joint is modeled as a massless rod with given orientation and variable length s. Like the revolute joint, the prismatic joint adds one degree of freedom to the system and therefore also defines one integrator. The joint velocity v_s and force f_s in direction of the joint axis are projected on the plane by a transformer. No force is acting in direction of the joint axes, which is modeled by a source of zero effort. The integrated joint velocity v_s determines the rod length s. The transformation of velocities and forces along the massless rod are modeled as in the translation element with the only difference that the transformation is now also modulated by the rod length s.

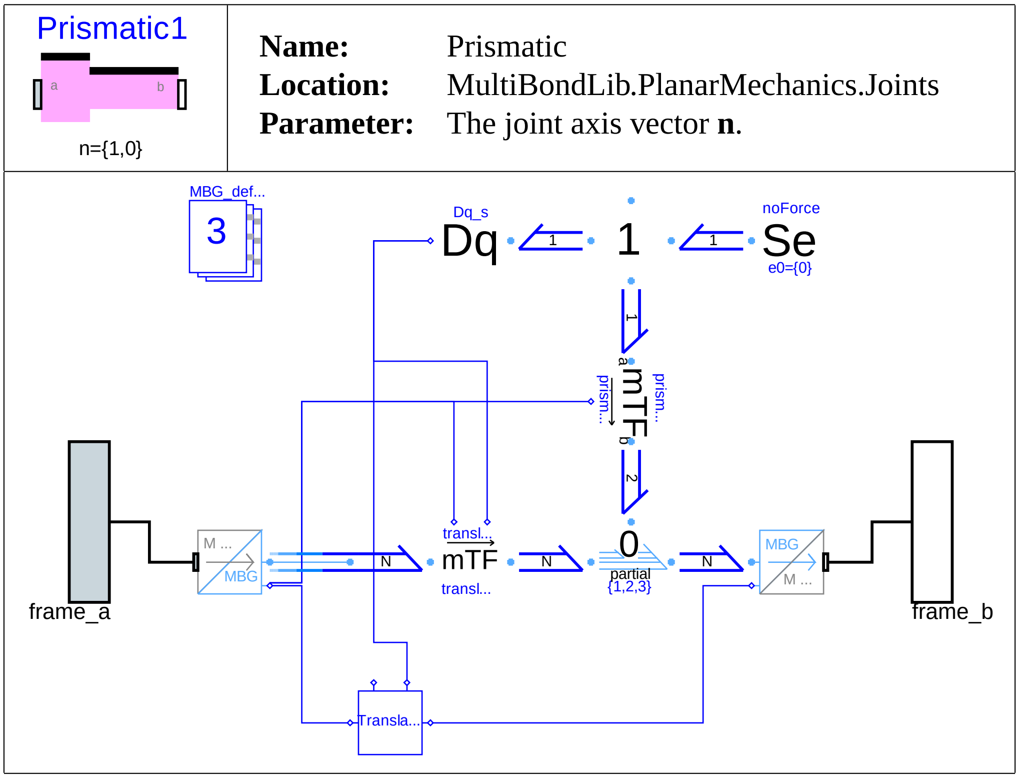
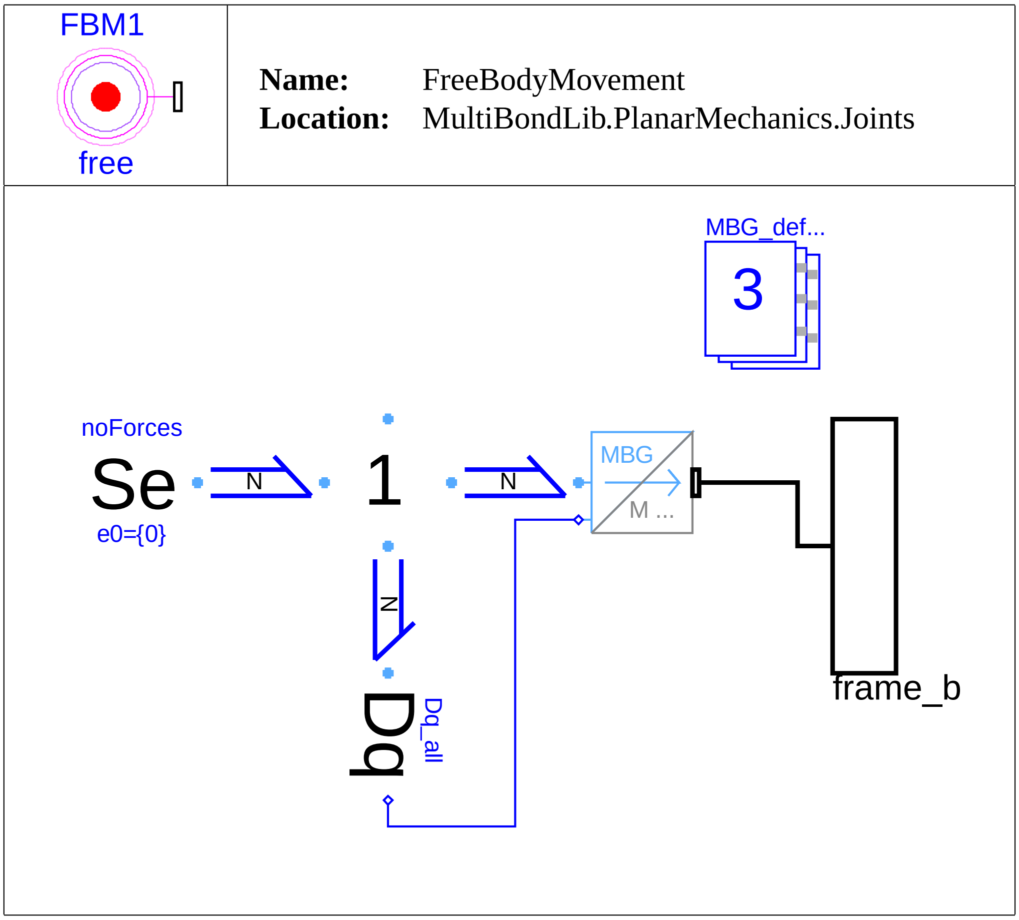

Name: Prismatic
Location: MultiBondLib.PlanarMechanics.Joints
Parameter: The joint axis vector **n**.

Figure 6.15: Presentation of the prismatic joint.

Free-body movement joint

The free-body movement joint is a pseudo joint that allows a body to freely move in space. It adds three degrees of freedom to the model and defines three integrators. The sensor element implements the simple differential equation: $\dot{\mathbf{x}} = \mathbf{v}$.



Name: FreeBodyMovement
Location: MultiBondLib.PlanarMechanics.Joints

Figure 6.16: Presentation of the free-body movement joint.

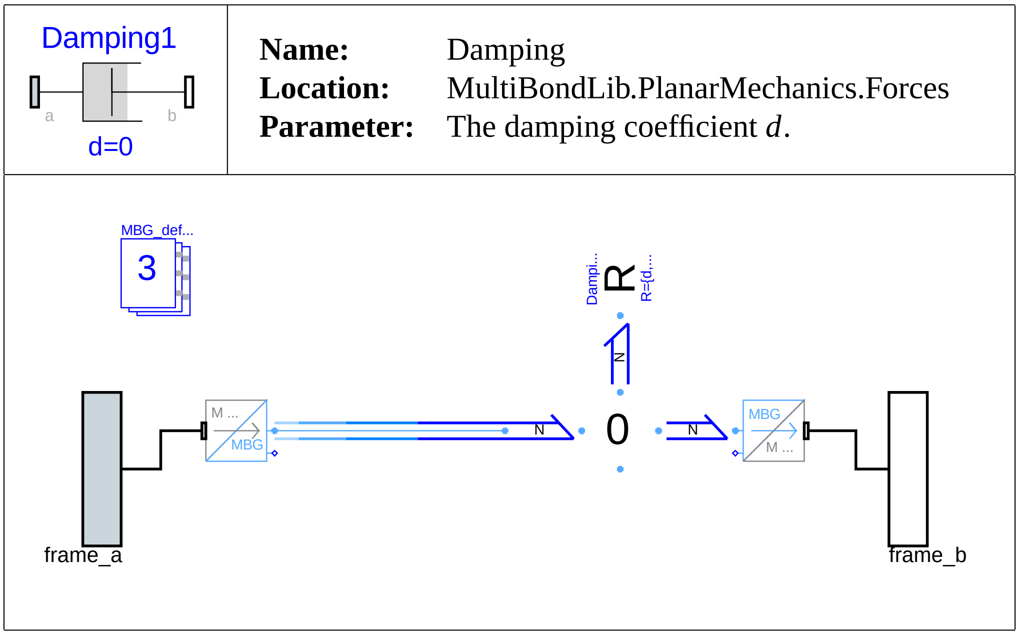
Combinations of joint elements

The combination of ideal joint elements (with no resistance and no momentum) cannot happen in an arbitrary fashion. One has to be careful that the position and movement of a body can correctly be decomposed into the positions and movements of the single joints.

A double pendulum with a massless first pendulum is such a bad example. This systems works well as long the two rods have a different orientation, but if they are placed in one line, the system fails, because it has reached a singular point. The velocity of the second pendulum's body cannot uniquely be decomposed into the two revolute joint velocities. There are infinitely many solutions.

The damper

The linear damper causes a counteracting force that is proportional to the relative velocity. It is modeled as linear resistance for the translational domain.



Name: Damping
Location: MultiBondLib.PlanarMechanics.Forces
Parameter: The damping coefficient d.

Figure 6.17: Presentation of the damper model.

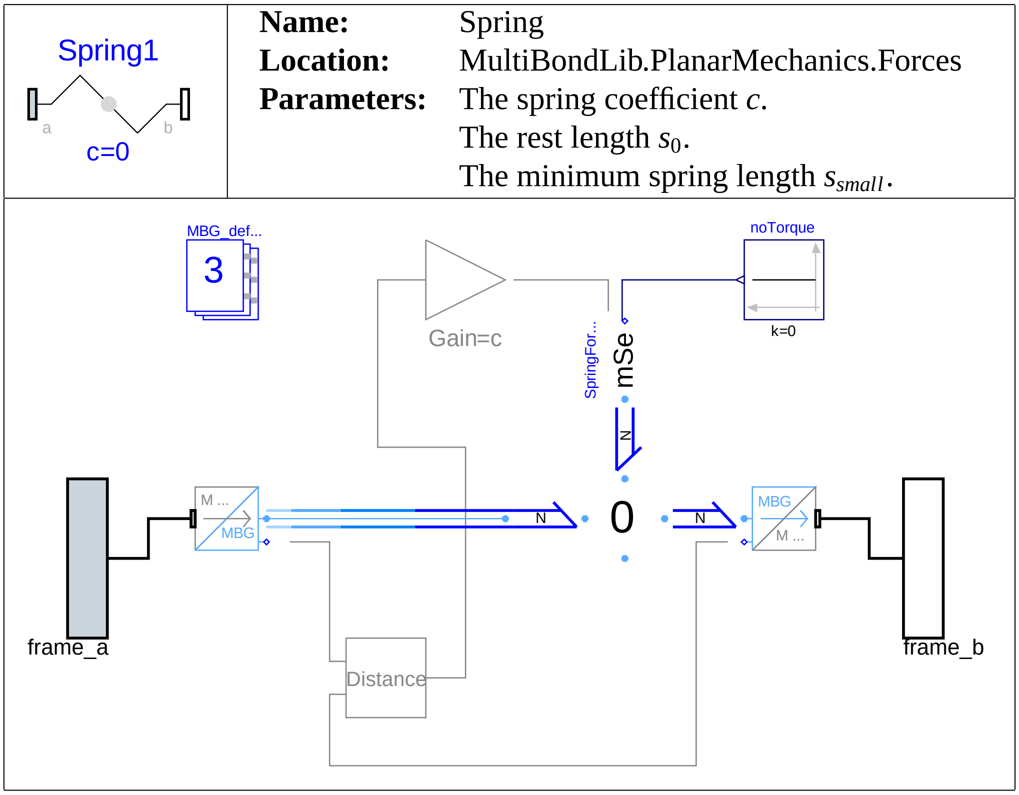
The spring

The linear spring causes a counteracting force that is linearly dependent on the difference in position. The spring is not modeled by a capacitance because we'd like to avoid force elements to define integrators. Therefore we model again the capacitance by a state modulated source of effort.[2] As it is indicated by the gray drawings in the model, the computation of the modulating input is done by equations:

```
  SI.Position sx "positional difference";
  SI.Position sy "positional difference";
  SI.Force fx "spring force";
  SI.Force fy "spring force";

  equation

  s = sqrt((sx^2) + (sy^2));
  fx = if s >= s_small then c*sx - s0*c*(sx/s)
                       else c*sx - s0*c*(sx/s_small);
  fy = if s >= s_small then c*sy - s0*c*(sy/s)
                       else c*sy - s0*c*(sy/s_small);
```

[2] The ambiguity between state modulated sources and energy storage elements has already been pointed out on page 57.

The spring can be parameterized by a spring coefficient c and a rest length s_0. The force acts always in direction of the spring axis. If the spring length approaches zero, this direction is critical to determine and the model becomes stiff. To avoid this stiffness an additional parameter s_{small} is helpful. It is the minimum divisor of the direction normalization. Hence, the computation of the spring force is incorrect for a spring length smaller than s_{small}, but the stiffness has been overcome.



Name: Spring
Location: MultiBondLib.PlanarMechanics.Forces
Parameters: The spring coefficient c.
The rest length s_0.
The minimum spring length s_{small}.

Figure 6.18: Presentation of the model for a linear spring.

6.4 Kinematic loops

6.4.1 Redundant connectors and manual loop closing

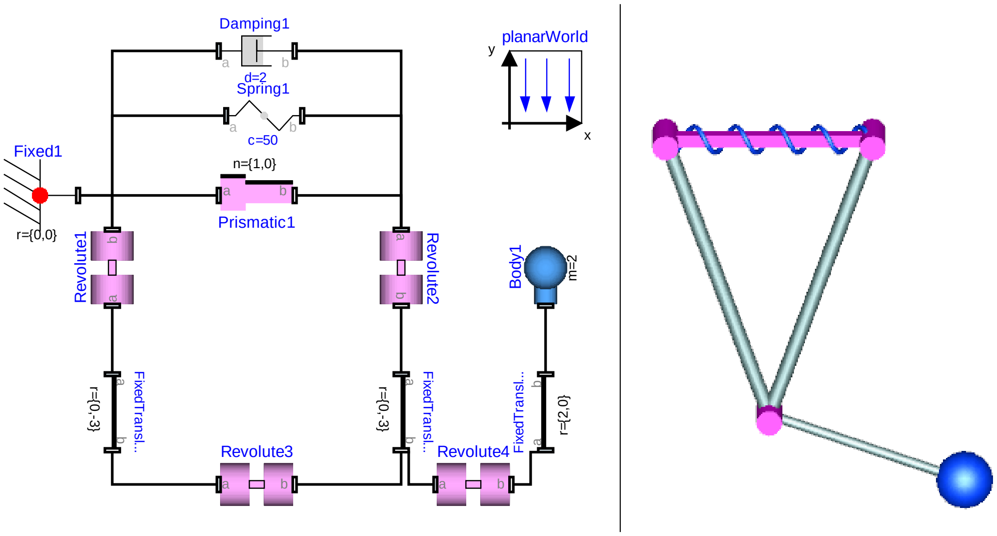


Figure 6.19: A kinematic loop with an attached pendulum.

The system of figure 6.19 demonstrates a kinematic loop. The rigid connections form a circular structure. The closing of the loop by the insertion of “Revolute3” reduces the number of degrees of freedom from 4 to 2. A kinematic loop leads to a structural singularity and integrators have to be removed. For most cases, this can be handled well by the Pantelides algorithm. Anyhow these structural singularities usually lead to a need of solving non-linear equations.

The complexity (here: the number of non-linear equations) of the resulting solution can be reduced by special models for common joint combinations. For example, the joint combination: Revolute-Translation-Revolute can be described by a distance constraint. Anyhow, such special joint elements have not been developed for this library.

It has been already pointed out that the connector’s potential variables are redundant, because the velocities result out of the position’s derivative. This is of no matter as long as the rigid connections form a tree structure. In a tree structure the statement that is connecting velocities and positional variables (by fixation or integration) is only made at one point: at the root of the tree. Two of our basic elements are root elements: The wall (fixation) and the free-body movement joint (integration). Each tree structure has to have exactly one of these root elements. Because there is only one root, the redundant variables are connected at only one point and the overdetermination of the connector’s variables is of no importance.

However, if we now connect two branches of such a rooted tree to form a circle, the resulting system of equations will be overdetermined. There is a root on both sides of the connection and we're in fact stating the same thing twice. By a simple connection we state:

$$x_1 = x_2 \tag{6.1}$$

$$v_1 = v_2 \tag{6.2}$$

Because there is a root on the one hand side of the connection there is an equation:

$$v_1 = f(x_1) \tag{6.3}$$

Because there is a root on the other hand side of the connection there is an equation:

$$v_2 = f(x_2) \tag{6.4}$$

Out of equations 6.1, 6.3 and 6.4 the connection equation 6.2 $v_1 = v_2$ results and this is the redundant statement.

A simple, but not very comfortable and intuitive way of solving this problem is to create an element that enables the user to manually close loops. This element is called CloseLoop and just makes statements for the positional and flow variables:

$$x_1 = x_2$$

$$y_1 = y_2$$

$$\varphi_1 = \varphi_2$$

$$f_{x,1} = f_{x,2}$$

$$f_{y,1} = f_{y,2}$$

$$t_1 = t_2$$

The redundant statements are now removed and the resulting system of equations will be non-singular. Figure 6.20 shows the model for the kinematic loop presented in figure 6.19 and the application of the manual loop closing.

In the next section an automatic solution will be presented. The manual loop closing still remains a helpful technique, if you want to break the circular structure at a specific point.

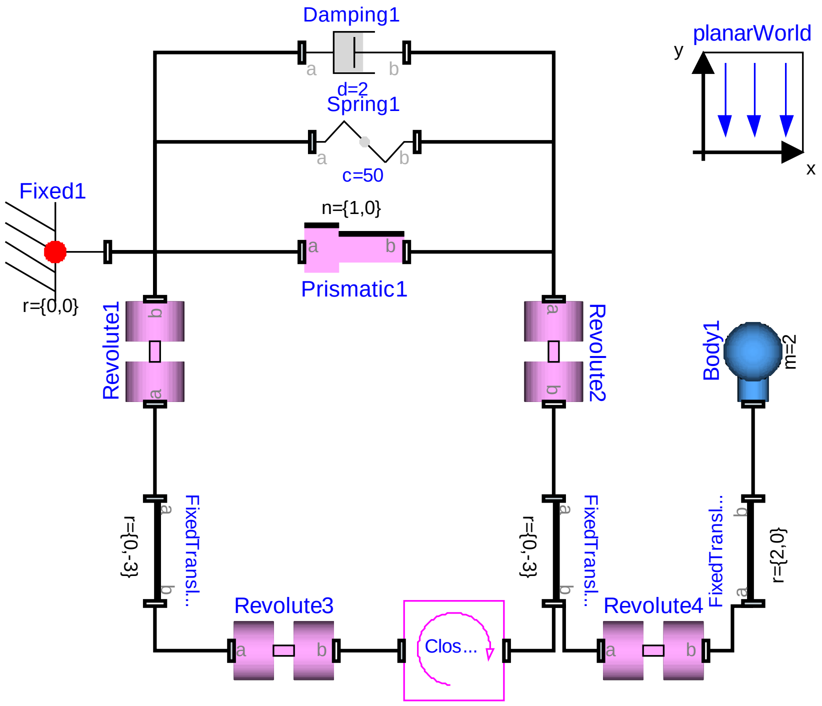


Figure 6.20: The kinematic loop is closed manually in this model.

6.4.2 Automatic loop closing

Methods to handle redundant connectors have been developed for the Modelica language and their application allows an automatic closing of kinematic loops.

The considerations of the previous paragraph led us to certain important conclusions:

- There are certain root elements, which determine the redundant variables completely.
- If an element is connected to a root element it is called a rooted element.
- The equations that we would like to state by connecting two elements are dependent whether only one or both sides of the connection are rooted.

A connection generates a set of equations. It seems to be that there is a need for two alternative sets that are chosen depending on the kind of connection. Modelica supports this in the following (pretty awkward) way:

- The set of redundant variables is summarized in a record
- In the record you can define an encapsulated function with the specific name "equalityConstraint" that takes two of its records as input and returns a vector of reals.
- The length of the output vector corresponds to the number of non-redundant equations.
- The output vector represents a residue that becomes zero if the two input records are equal.

This leads to the following solution for the connector "Frame" in the planar mechanics library: The redundant potential variables are organized as a record and in the record you see the function "equalityConstraints" that generates 3 non-redundant equations.

```
Record Potentials
  "redundant part of the connector for the planar objects"

  import SI = Modelica.SIunits;

  SI.Position x "Position";
  SI.Position y "Position";
  SI.Angle phi;

  SI.Velocity vx "Velocity";
  SI.Velocity vy "Velocity";
  SI.AngularVelocity w;

  encapsulated function equalityConstraint
    "Return the constraint residues to express that two frames have the same orientatio

    import Modelica;
    import MultiBondLib.PlanarMechanics.Interfaces;
    extends Modelica.Icons.Function;
    input Interfaces.Potentials P1;
    input Interfaces.Potentials P2;
    output Real residue[3];
  algorithm
    residue := {P1.x-P2.x,P1.y-P2.y,P1.phi-P2.phi};
  end equalityConstraint;
End Record
```

A Modelica translator can now choose, whether to take the standard connection equations (in this case 6) or the residue equations of the function equalityConstraint (in this case 3). But we have to give the translator additional information for it to be able to make this choice properly.

We have already stated that there are elements that introduce the redundancy. These are the root elements. Some other elements (rigid elements like translations or joints) propagate this redundancy from one connector to another. Yet other elements do not propagate the redundancy and therefore do not close kinematic loops. (E.g: the spring-damper in figure 6.19 does not generate a kinematic loop, although it is connected in a circular way)

It is necessary to mark the elements according to their behavior.

- Let P be the redundant set of variables.
- If c1 is the connector of a root element mark it with the function defineRoot(c1.P).
- If an element is promoting the redundancy between the connectors c1 and c2 mark this with defineBranch(c1.P,c2.P)

Figure 6.21 shows a connection diagram of the kinematic loop. The root connectors are marked by a red circle and all other connectors are marked by an empty circle. Thick lines denote an unbreakable connection between two connectors that has been marked by defineBranch(). The dotted lines denote normal connections between connectors. The structure of normal connections has to form a wood, a set of trees. Therefore it is necessary to replace one normal connection per kinematic loop with its non-redundant alternative set of equations. This is called the cut of the loop. There are many different possibilities, where to cut the loop. The choice of the built-in algorithm is shown marked by stroke.

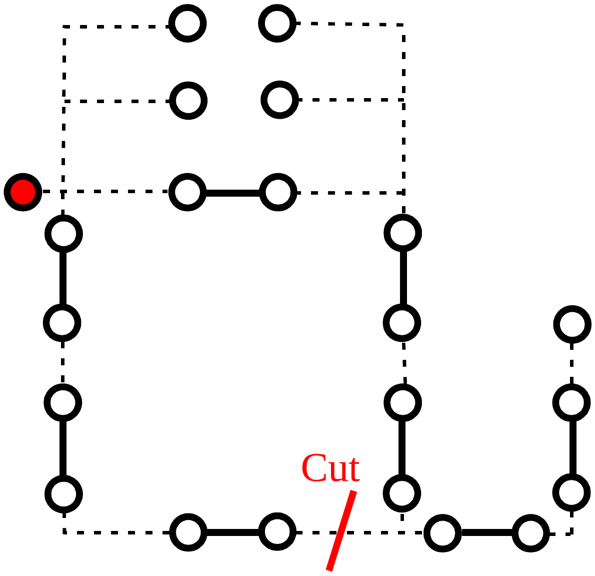
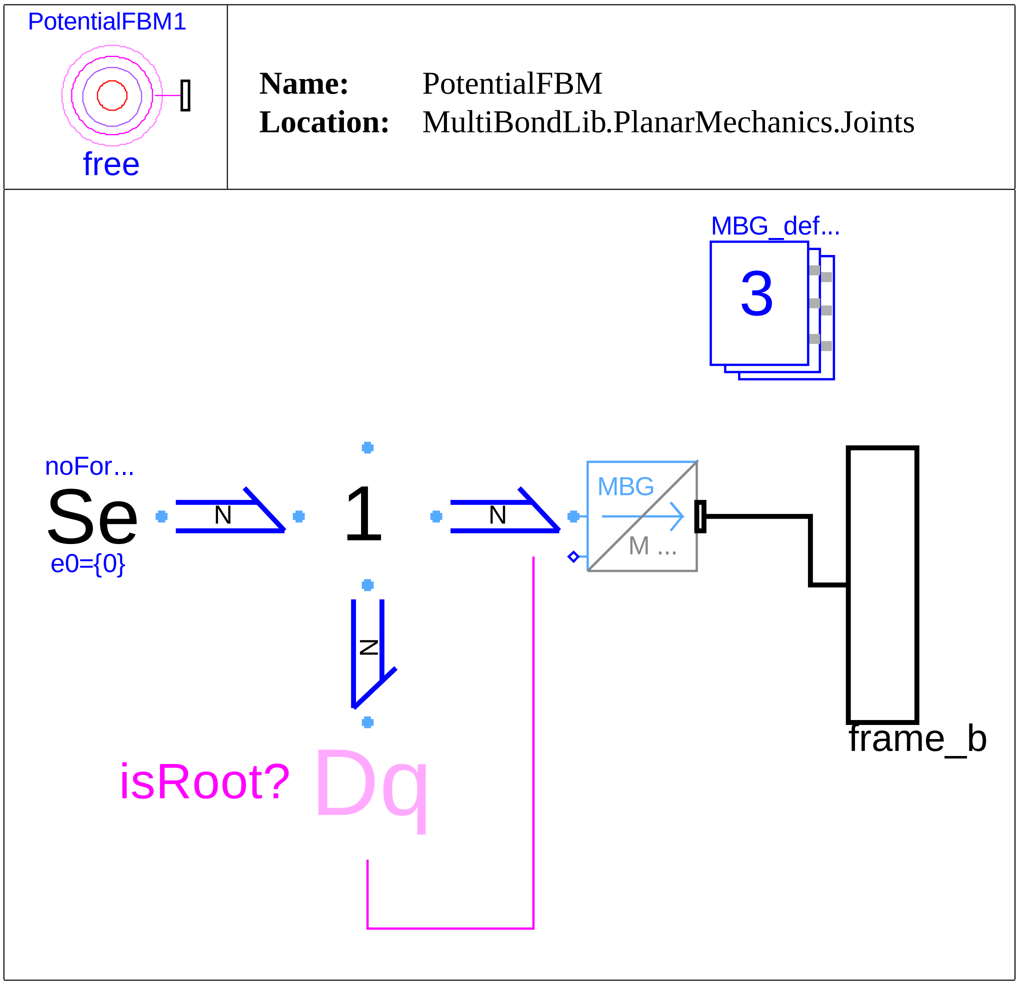


Figure 6.21: Connection diagram of the kinematic loop model.

This technique enables a comfortable modeling of kinematic loops. Loops can be created in a truly object-oriented fashion and the modeler doesn't have to think about any further implications.

6.4.3 Potential rooting

So far a moving body always had to be connected to a joint, which looks pretty strange in the case of a freely moving body. It is not very natural, if a freely moving body has to be connected to a joint. Hence, it appears to be a good idea, to include a free-body movement joint in the body's model. Although this works fine for freely moving bodies, it unfortunately generates unnecessary kinematic loops if the body is restricted in its movement by other joints. Therefore we'd like this included joint only to be active if the body movement can't be derived out of other joints, i. e. if the body's connector is not rooted.



Name: PotentialFBM
Location: MultiBondLib.PlanarMechanics.Joints

Figure 6.22: Presentation of the potential free-body movement joint.

Figure 6.22 shows a potential free-body movement joint (PotentialFBM) that generates exactly this behavior. The model is very similar to the real FBM model, but the integrators are only defined, if the joint's connector was chosen to be a root, as it is indicated by the magenta drawings. This is achieved by the usage of further advanced Modelica commands.

Using the command "definePotentialRoot(frame_a.P)", we can mark this joint's connector as a potential root. The Modelica translator then decides whether frame_a.P should be a root or not, according to the following rules:

- If P is rigidly connected to another root, it won't be chosen to be a root.
- If P is not rigidly connected to any other root, it will be chosen to be a root.
- If there are several potential roots rigidly connected to each other, the Modelica translator chooses one of them to be the real root.

The Modelica command "isRoot(frame_a.P)" enables the modeler to check if "frame_a.P" was chosen to be a root and to adapt the sets of equations. In the root case the integrator equations are defined, otherwise they aren't.

```
if isRoot(frame_b.P) then
  vx = der(x);
  vy = der(y);
  w = der(phi);
end if;
```

The resulting model of the potential free-body movement joint is then included into the body model. Figure 6.23 shows the necessary update.

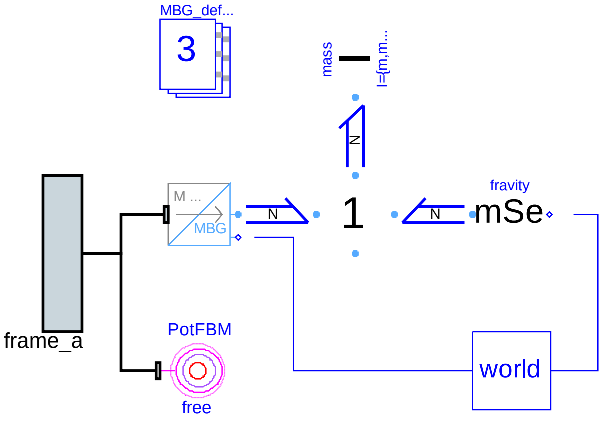


Figure 6.23: Extension of the primal body model.

6.5 Conclusions

The PlanarMechanics library offers a comprehensive set of ideal elements for planar mechanics. Planar systems can be comfortably modeled and efficiently simulated.

A more comprehensive evaluation can be found at the end of the next chapter. Even though this evaluation concerns mainly three dimensional models, the relevant conclusions are valid for the PlanarMechanics library as well.

Chapter 7

3D-Mechanics

7.1 Introduction to 3D-mechanics

The development of the PlanarMechanics library taught us how to use bond graphs to model mechanical systems, how to decompose such systems, how to use the wrapping technology and handle kinematic loops. All this knowledge can be inherited for the creation of a 3D-mechanics library. Unfortunately, two additional difficulties arise by entering the three-dimensional world:

- The computation of all physical variables in the inertial system turns out to be pretty laborious and is not a good approach. The rotational variables will have to be computed in systems attached to the body and we have to handle different coordinate systems and the transformations between them.

- The relative and absolute rotation of an object can't be expressed so easily anymore. There are different techniques to represent an orientation and in consequence there are different methods to integrate the angular velocity.

The first difficulty results out of the fundamental equations. Different from the planar mechanics the rotational inertia isn't independent of the body's orientation any more. A more detailed explanation of this problem is given at the end of this section. To face the second difficulty, we need to ask how to describe the position, orientation and motion of a moving object in space. In planar mechanics this discussion was reduced to a minimum (it was nearly omitted), because the triviality of the presented solution did seemingly not generate the need for further explanations. Anyhow in three-dimensional space this issue is not that trivial any more and there are several good solutions each with its own pros and cons. Also the usage of multiple coordinate systems becomes necessary. Hence, a proper discussion is definitely needed.

7.1.1 The coordinate systems

A body's position can be described by coordinates in a three-dimensional Euclidean coordinate system. Other coordinate systems like polar or cylindrical coordinates are also sufficient, but we shall consistently use Euclidean coordinate systems.

As in planar mechanics, one can arbitrarily choose one possible coordinate system in space and express all positional variables with respect to this system. Such a coordinate system is then said to be the inertial system. Coordinates resolved with respect to the inertial system are called absolute coordinates. An inertial system is supposed to be a non-accelerated system.

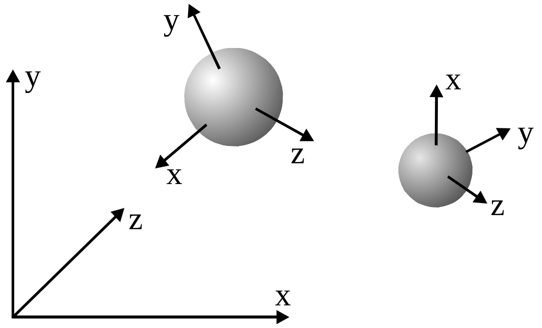


Figure 7.1: Illustration of different coordinate systems.

In 3D-Mechanics there is also the need for coordinate systems that are fixed to a rigid element of interest (e g. a body). Coordinates of objects rigidly fixed to the element remain then constant. Such a coordinate system is then called a body system. A body system is mostly an accelerated system. This has an impact on the laws of physics. Pseudo forces like the gyroscopic force appear. Figure 7.1 shows an inertial coordinate system and two bodies together with their body systems.

Due to the usage of multiple coordinate systems we have to denote in which system the vector variables are resolved. The notation needs to be extended, to allow a clear separation between inertial system vectors and body system vectors. Vectors resolved with respect to the inertial frame are denoted by the suffix 0 (e. g. $\boldsymbol{\omega}_0$), vectors resolved with respect to their body systems are denoted by the suffix "body" (e. g. $\boldsymbol{\omega}_{body}$). Even though several body systems may be used within a model, it is mostly clear which vector belongs to which body system, and hence a further (more precise) notation is omitted to avoid an irritating puzzle of index notations.

7.1.2 Orientation in three-dimensional space

The rotation matrix: R
The orientation of an object is completely defined by the coordinate vectors of its body system.

The relative orientation between two objects can then be described by the transformation between the two orthonormal systems. Such a transformation can be done by a orthonormal matrix: the rotation matrix. The absolute orientation of an object is then its relative orientation to the inertial system. Given the rotational matrix, we can easily transform vectors between different coordinate systems, e. g., $\mathbf{R}\boldsymbol{\omega}_0 = \boldsymbol{\omega}_{body}$.

It is important to note that the rotation matrix is a highly redundant way for describing an object's orientation. Because it is an orthonormal matrix all row and column vectors are of magnitude 1. This results in 6 constraint equations. From the 9 variables of the matrix there remain therefore only 3 unknowns that are are necessary to describe an objects orientation.

The cardan angles: $[\varphi_x, \varphi_y, \varphi_z]^T$
A non-redundant way to describe the orientation is the usage of cardan angles. This technique decomposes the rotation into three separate rotations around given axes, which is possible for every rotation.

Therefore the object is rotated first around the x-axis then around the y-axis and finally around the z-axis. Each of these rotations is defining a rotation matrix:

$$\mathbf{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\varphi_x) & sin(\varphi_x) \\ 0 & -sin(\varphi_x) & cos(\varphi_x) \end{pmatrix} \tag{7.1}$$

$$\mathbf{R}_y = \begin{pmatrix} cos(\varphi_y) & 0 & -sin(\varphi_y) \\ 0 & 1 & 0 \\ \sin(\varphi_y) & 0 & cos(\varphi_y) \end{pmatrix} \tag{7.2}$$

$$\mathbf{R}_z = \begin{pmatrix} cos(\varphi_z) & sin(\varphi_z) & 0 \\ -sin(\varphi_z) & cos(\varphi_z) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{7.3}$$

The absolute rotation $\mathbf{R}$ as a function of φ_x, φ_y, φ_z is then:

$$\mathbf{R} = \mathbf{R_z} \cdot \mathbf{R_y} \cdot \mathbf{R_x} \tag{7.4}$$

Unfortunately the decomposition into separate rotations can't be done in a unique way if the second angle φ_y is $90°$. The other two rotation axes are then aligned and there are infinitely many solutions. So cardan angles are only useful, if one can make sure this case won't appear during simulation time. It has to be pointed out that the sequence of axis rotation was chosen arbitrarily. Other sequences are of course possible as well and each valid sequence has a specific point where the systems becomes singular.

The planar rotation: $[\varphi, \mathbf{n}]^T$
Every rotation can be seen as a planar rotation with the angle φ around a certain axis given by

a unit vector **n**. We therefore have 4 variables and one constraint equation for the unit vector's magnitude.

Such a planar rotation can be expressed by the following rotation matrix: [1]

$$\mathbf{R} = \mathbf{n}\mathbf{n}^T + \left(I - \mathbf{n}\mathbf{n}^T\right) cos(\varphi) - \tilde{\mathbf{n}} sin(\varphi) \tag{7.5}$$

Unfortunately there are infinitely many ways to express a null rotation, because there are infinitely many possible rotation axes. Therefore this way of describing a rotation is only applicable if the rotation axis is known (as in a revolute joint).

The quaternion rotation: Q
Quaternions are an extension of complex numbers and offer a robust way to describe any rotation. The corresponding methods are briefly discussed here. A more thorough discussion of this topic can be found in [15]. A quaternion number consists of one real and three imaginary components, denoted by i, j and k. The imaginary components can be summarized by a vector **u**.

$$Q = c + ui + vj + wk. = c + \mathbf{u}$$

The multiplication rules for the imaginary components are as follows:

$$\begin{array}{lll} ij = k; & ji = -k; & i^2 = -1 \\ jk = i; & kj = -i; & j^2 = -1 \\ ki = j; & ik = -j; & k^2 = -1 \end{array}$$

So the product of two quaternions can be written as:

$$QQ' = (c + \mathbf{u})(c' + \mathbf{u}') = (cc' - \mathbf{u} \cdot \mathbf{u}') + (\mathbf{u} \times \mathbf{u}') + c\mathbf{u}' + c'\mathbf{u}$$

The complement of a quaternion number is defined to be $\bar{Q} = c + \bar{\mathbf{u}} = c - \mathbf{u}$; The product of a quaternion number with its complement results in its norm:

$$|Q| = c^2 + |\mathbf{u}|^2$$

A unit quaternion is a quaternion of norm 1:

$$|Q| = c^2 + |\mathbf{u}|^2 = 1$$

According to a variant of the trigonometric pythagoras: $cos(\varphi/2)^2 + sin(\varphi/2)^2 = 1$, there is an angle φ for every unit quaternion such that: $c = cos(\varphi/2)$ and $|\mathbf{u}| = sin(\varphi/2)$. It now becomes

[1] Please remind the matrix notation of the cross product introduced in chapter 1.4.

evident how a unit quaternion can be used to describe an orientation. The idea is related to the planar rotation. The imaginary component **u** describes the axis, and the length of the axis describes the rotation angle.

The rotation matrix is then defined by:

$$\mathbf{R} = 2\mathbf{u}\mathbf{u}^T + 2(\tilde{\mathbf{u}} \cdot c) + 2c^2\mathbf{I} - \mathbf{I} \tag{7.6}$$

Usage of these techniques
In the multibondgraphic library for 3D-mechanics all of these description techniques are applied. The rotational matrix is the most general way to describe an orientation and it also generates only linear relationships. For these two reasons it is used to transmit the information about an element's orientation and you will find the rotational matrix in an element's connector. Planar rotation is used to describe the rotation of revolute joints and its composites. To describe an unlimited orientation, cardan angles or quaternions are used.

7.1.3 Motion in three-dimensional space

The description of the translational movement is straightforward. The simple equation holds:

$$\mathbf{v}_0 = \dot{\mathbf{x}} \tag{7.7}$$

A rotational movement is usually described by a vector of angular velocities. The vector points in the direction of the rotation axis and its strength denotes the rotational speed. The equations needed to integrate the angular velocity are more complex and dependent on the kind of rotation:

- The rotation matrix itself is the one to integrate:

$$\tilde{\boldsymbol{\omega}}_0\mathbf{R} = \mathbf{R}\tilde{\boldsymbol{\omega}}_{body} = \dot{\mathbf{R}} \tag{7.8}$$

- The rotation matrix results out of a planar rotation dependent on φ with fixed rotation axis

$$\boldsymbol{\omega}_0 = \boldsymbol{\omega}_{body} = \mathbf{n} \cdot \dot{\varphi} \tag{7.9}$$

- The rotation matrix results out of the three cardan angles: (φ_x, φ_y, φ_z):

$$\boldsymbol{\omega}_{body} = \dot{\varphi_z} + \mathbf{R}_z\dot{\varphi_y} + \mathbf{R}_z\mathbf{R}_y\dot{\varphi_x} \tag{7.10}$$

$$\boldsymbol{\omega}_0 = \dot{\varphi_x} + \mathbf{R}_x^T\dot{\varphi_y} + \mathbf{R}_x^T\mathbf{R}_y^T\dot{\varphi_z} \tag{7.11}$$

- The rotation matrix is the result of a quaternion rotation:

$$\boldsymbol{\omega}_{body} = 2 \begin{pmatrix} c & -w & v & u \\ w & c & -u & v \\ -v & u & c & w \end{pmatrix} \cdot \begin{pmatrix} \dot{c} \\ \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} \tag{7.12}$$

$$\boldsymbol{\omega}_{body} = 2 \begin{pmatrix} c & w & -v & u \\ -w & c & u & v \\ v & -u & c & w \end{pmatrix} \cdot \begin{pmatrix} \dot{c} \\ \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} \tag{7.13}$$

7.1.4 Fundamental set of equations

We state the fundamental equations for a rigid body's movement:

$$\mathbf{f}_0 = m \cdot \mathbf{a}_0 \tag{7.14}$$

$$\mathbf{t}_0 = \mathbf{J}_0 \dot{\boldsymbol{\omega}}_0 \tag{7.15}$$

Whereas in planar systems the rotational inertia was a simple constant scalar it is now a matrix called the inertia tensor. Yet the inertia tensor is dependent on the orientation, and therefore the equation for the rotational domain becomes awkward to compute in the inertial system. Fortunately we can transform the law into the body system, where the inertia tensor is of course constant.

$$\mathbf{t}_0 = \frac{d}{dt}\left(\mathbf{R}^T \mathbf{J}_{body} \boldsymbol{\omega}_{body}\right)$$

$$\mathbf{t}_0 = \dot{\mathbf{R}}^T \mathbf{J}_{body} \boldsymbol{\omega}_{body} + \mathbf{R}^T \mathbf{J}_{body} \dot{\boldsymbol{\omega}}_{body}$$

$$\mathbf{R}^T \mathbf{t}_{body} = \mathbf{R}^T \tilde{\boldsymbol{\omega}}_{body} \mathbf{J}_{body} \boldsymbol{\omega}_{body} + \mathbf{R}^T \mathbf{J}_{body} \mathbf{z}_{body}$$

$$\mathbf{t}_{body} = \boldsymbol{\omega}_{body} \times \mathbf{J}_{body} \boldsymbol{\omega}_{body} + \mathbf{J}_{body} \mathbf{z}_{body} \tag{7.16}$$

As one can see, the computation in the accelerated body systems adds a pseudo torque $\boldsymbol{\omega}_{body} \times \mathbf{J}_{body}\boldsymbol{\omega}_{body}$ called the gyroscopic torque. Transforming also the equations of the translational domain into the body system would generate a similar term concerning the force. But there is no need to do this, and therefore we will consistently resolve the rotational variables (torque, angular velocity and angular acceleration) in their body systems and express the translational variables in the inertial system. The positional variables (**x** and **R**) are always resolved with respect to the inertial system.

7.2 The Mechanics3D library

7.2.1 Introduction

The Mechanics3D library is a tool for modeling three-dimensional mechanical systems in an object-oriented way. The structure of this library is very similar to the planar mechanics library of the last chapter and to the MultiBody library. Again the complex models can be composed out of ideal elements for masses, joints, translations and forces. The elements are (if they have a planar counterpart) also symbolized by similar icons. A structural overview of the Mechanics3D library is presented in appendix E.

Of course, also these models are wrapped bondgraphic models. The concept of wrapping has already been explained in the last chapter. Therefore we almost directly jump to the modeling of the basic elements without any introductory examples. But before we do that, we have to take a look at the connector's design and at some special bondgraphic elements for 3D-Mechanics.

7.2.2 The wrapping

The choice of the connector variables follows the same logic as in the PlanarMechanics library. Again the connector models a 1-junction and hence the flow variables are:

- force: **f**
- torque: **t**

In consequence the potential variables are:

- position:$\mathbf{x}$
- orientation:$\mathbf{R}$
- velocity:$\mathbf{v}$
- angular velocity: $\boldsymbol{\omega}$

Notice that the set of potential variables is again redundant. Not only the positional variables $\mathbf{x}$ and $\mathbf{R}$ are related to their derivatives $\mathbf{v}$ and $\boldsymbol{\omega}$, but also the rotational matrix $\mathbf{R}$ itself is a redundant description of the orientation. Unsurprisingly the same method for handling kinematic loops is applied as in the library for planar mechanics. Again a non-redundant alternative set of connection equations has to be stated. This alternative set omits the equality of the velocities and just states the 3 equations that assure the equality of position and another 3 equations that assure the equality of orientation:

- $\mathbf{x}_{[1]} = \mathbf{x}'_{[1]}$
- $\mathbf{x}_{[2]} = \mathbf{x}'_{[2]}$
- $\mathbf{x}_{[3]} = \mathbf{x}'_{[3]}$
- $(\mathbf{R}_{[1,:]} \times \mathbf{R}_{[2,:]}) \cdot \mathbf{R}'_{[2,:]} = 0$
- $(\mathbf{R}_{[1,:]} \times \mathbf{R}_{[2,:]}) \cdot \mathbf{R}'_{[1,:]} = 0$
- $\mathbf{R}_{[2,:]} \cdot \mathbf{R}'_{[1,:]} = 0$

Also the connector summarizes vector variables that are resolved in different systems. Most of these variables are resolved in the inertial system, but the angular velocity $\boldsymbol{\omega}$ and the torque $\mathbf{t}$ are resolved in the body system. The coordinate vectors of the body system are then given by the rotational matrix $\mathbf{R}$.

Because the translational and rotational bondgraphic variables are resolved in different coordinate systems, there are separate multibonds for the translational and for the rotational domain. The interaction of these two domains is then modeled via transformers. In consequence, there are also two separate positional signals $\mathbf{x}$ and $\mathbf{R}$, both of course resolved in the inertial system.

The converter element therefore splits all variables for the rotational and the translational domain. Figure 7.2 shows an empty template for a two-connector element (e.g: a revolute joint). The connectors coupled with the converter elements create the empty shell for the bondgraphic model.

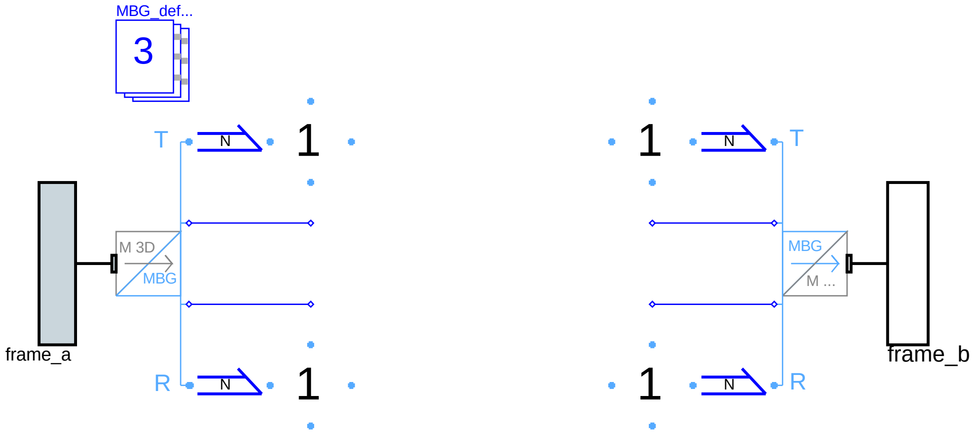


Figure 7.2: The frame connectors in combination with the converter elements form a shell for the bondgraphic model. The bondgraphic model is indicated by 1-junctions.

7.2.3 Special bondgraphic elements

The modulated gyrator: MGY
To model the gyroscopic torque described by the equation $\mathbf{t} = \boldsymbol{\omega}_{body} \times \mathbf{J}_{body}\boldsymbol{\omega}_{body}$ we need a special bondgraphic element. The gyroscopic torque is an internal pseudo torque and behaves neutral with respect to energy and power. It should therefore be modeled by a modulated intradomain gyrator.

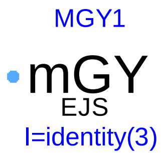
	**Name:** MGY **Location:** MultiBondLib.Mechanics3D.AdditionalMBG **Parameters:** Inertia tensor **J**. **Equations:** $\mathbf{e} = \mathbf{f} \times \mathbf{J} \cdot \mathbf{f}$

Figure 7.3: Special modulated gyrator: mGY.

A decomposition into single bonds reveals an interesting structure and the modulated intradomain gyrators become visible. The modulating signals are the components of the angular momentum. This special junction structure is called Eulerian junction structure.

The translational transformer
Again a special transformer is needed to implement the balance equations of a lever. This transformer is called translational transformer and couples the rotational and the translational domains by converting the angular velocity into translational velocity for a point defined by a vector **r**

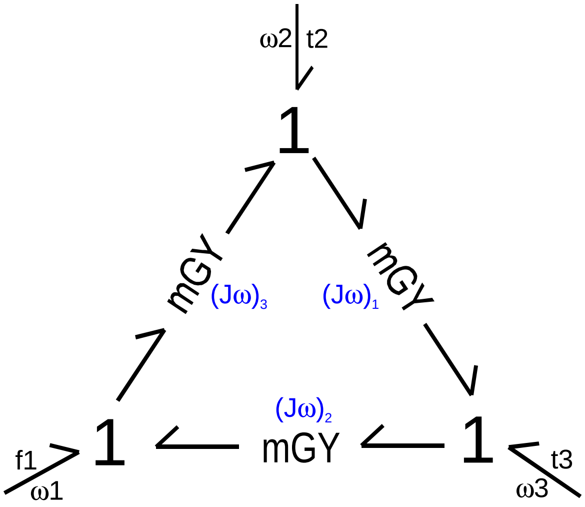


Figure 7.4: The eulerian junction structure.

away from the turning center. The transformation of the force into torque then follows out of conservation of energy.

$$\mathbf{v}_2 = \omega_1 \times \mathbf{r}$$

$$\mathbf{t}_1 = \mathbf{r} \times \mathbf{f}_2$$

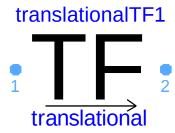
There are two additional modulated variants of this transformer provided. The first variant can be modulated by an amplification signal. This is useful to model the prismatic joint. The second variant is more general: The parameter vector **r** is changed into a modulating signal.

Special modulated transformers

A bondgraphic transformer is often an acausal element. Even though the transformation matrix is usually given for one direction only, the causality can easily be switched by using its transposed variant, if the transformation matrix is orthogonal. Such transformers will appear very often in the models, because transformers are extensively used to rotate the bondgraphic variables. Such a rotation is always defined by an orthogonal matrix. Unfortunately the simulation program Dymola isn't aware of a concept like orthogonality, and therefore will solve (successfully) a linear set of equations to determine the inverse of the transformation matrix, if the causality enforces this. Hence, linear systems of equations are generated, which have to be solved at simulation time. This adds unnecessary computational complexity to the model and this also impairs the state selection mechanism.

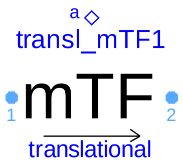
To overcome this problem, a special transformer has been developed, the preferred causality of which can be determined by a boolean signal. I. e.: the choice between transforming the flow or the effort from a to b is now made according to the input signal. If it is true, the flow is transformed in arrow direction, otherwise the effort is transformed in arrow direction. This boolean signal has no semantic (physical) meaning. It only affects the computational aspects of the model. Although the transformer is an acausal element, it appears to be helpful to state the equations in the needed causality.

Often the model structure lets us deduce the most likely causalization and the correct boolean signal value can be determined before the simulation. If this can be done correctly, the unnecessary solving of linear systems can be avoided. Later on, we shall show how to the deduce the correct causalization.



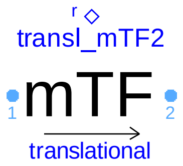
Name: translationalTF
Location: MultiBondLib.Mechanics3D.AdditionalMBG
Parameters: translation vector **r**.
Equations:

$$\mathbf{f}_2 = \mathbf{f}_1 \times \mathbf{r}$$
$$\mathbf{e}_1 = \mathbf{r} \times \mathbf{e}_2$$



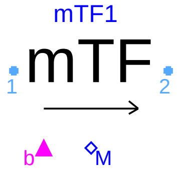
Name: translational_mTF
Location: MultiBondLib.Mechanics3D.AdditionalMBG
Parameters: translation vector **r**.
Modulating signal: amplification factor *a*.
Equations:

$$\mathbf{f}_2 = \mathbf{f}_1 \times (a \cdot \mathbf{r})$$
$$\mathbf{e}_1 = (a \cdot \mathbf{r}) \times \mathbf{e}_2$$



Name: translational_mTF2
Location: MultiBondLib.Mechanics3D.AdditionalMBG
Modulating signal: translation vector **r**.
Equations:

$$\mathbf{f}_2 = \mathbf{f}_1 \times \mathbf{r}$$
$$\mathbf{e}_1 = \mathbf{r} \times \mathbf{e}_2$$



Name: mTF
Location: MultiBondLib.Mechanics3D.AdditionalMBG
Modulating signals: transformation matrix **M**.
boolean control signal *b*.

Equations:

$$\underbrace{\begin{matrix} \mathbf{f}_2 = \mathbf{M}\cdot\mathbf{f}_1 \\ \mathbf{e}_1 = \mathbf{M}^T\cdot\mathbf{e}_2 \end{matrix}}_{b=true} \quad \begin{matrix} \texttt{or} \\ \texttt{or} \end{matrix} \quad \underbrace{\begin{matrix} \mathbf{e}_2 = \mathbf{M}\cdot\mathbf{e}_1 \\ \mathbf{f}_1 = \mathbf{M}^T\cdot\mathbf{f}_2 \end{matrix}}_{b=false}$$

Figure 7.5: Special multibondgraphic elements for 3D mechanics.

Other special elements

Again there is a need for special non-bondgraphic elements to modify the positional signals.

To model a relative translation between two objects, the “Translation” element was designed. It shifts the positional signal by a vector **r** that is resolved in the body system. It is therefore modulated by the current rotation matrix. It can additionally be modulated by an amplification signal which is helpful for modeling a prismatic joint.

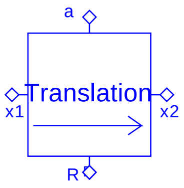
To model a relative rotation between two objects, the “Rotation” element has been designed. The two rotation matrices of the connectors $\mathbf{R}_a$ and $\mathbf{R}_b$ are coupled by a relative rotation matrix $\mathbf{R}_{rel}$ in the following way:

$$\mathbf{R}_2 = \mathbf{R}_{rel}\mathbf{R}_1$$

or

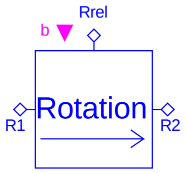
$$\mathbf{R}_1 = \mathbf{R}_{rel}^T\mathbf{R}_2$$

Both statements above are semantically identical, because a rotation matrix is orthogonal. The difference is again only the computational aspect, as it was the case for the special modulated transformer a few lines above. Therefore also this “Rotation” element can be modulated in a similar way for the same reasons.



Name:	Translation
Location:	MultiBondLib.Mechanics3D.Utilities
Parameters:	translation vector **r**.
Modified signal:	positional signal **x**.
Modulating signals:	orientation signal **R** optional amplification signal *a*
Equations:	

$$\mathbf{x}_2 = \mathbf{x}_1 + \mathbf{R}^T(a \cdot \mathbf{r})$$



Name:	Rotation
Location:	MultiBondLib.Mechanics3D.Utilities
Modified signal:	orientation signal.
Modulating signals:	relative rotation matrix $\mathbf{R}_{rel}$ boolean control signal *b*
Equations:	

$$\underbrace{\mathbf{R}_2 = \mathbf{R}_{rel} \cdot \mathbf{R}_1}_{b=true} \quad \texttt{or} \quad \underbrace{\mathbf{R}_1 = \mathbf{R}_{rel}^T \cdot \mathbf{R}_2}_{b=false}$$

Figure 7.6: Special signal elements for 3D mechanics.

Besides, other utility elements are useful to integrate the angular velocity and compute the rotation matrix out of the state variables. The element “toCardanAngles” and “toQuaternions” implement the differential equations for quaternions and cardan angles as they were presented in section 7.1.3 on page 85. The corresponding elements “cardanRotation” and “quaternionRotation” transform these positional variables into a rotation matrix. There is also a need for an object to transform the angle of a planar rotation into a rotation matrix. This is done by the element “planarRotation”. The equation for the necessary transformation was presented in section 7.1.2, page 82.

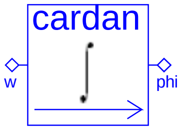
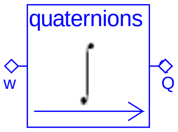
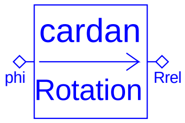
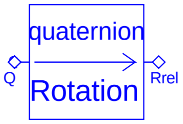
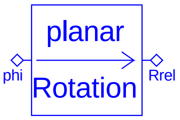
	**Name:**	toCardanAngles
	Location:	MultiBondLib.Mechanics3D.Utilities
	Parameters:	cardan angle sequence
	Signals:	angular velocity $\boldsymbol{\omega}$
		cardan angles φ_1, φ_2, φ_3
	**Name:**	toQuaternions
	Location:	MultiBondLib.Mechanics3D.Utilities
	Signals:	angular velocity $\boldsymbol{\omega}$
		quaternion Q.
	**Name:**	cardanRotation
	Location:	MultiBondLib.Mechanics3D.Utilities
	Parameters:	cardan angle sequence
	Signals:	cardan angles φ_1, φ_2, φ_3.
		rotation matrix $\mathbf{R}_{rel}$.
	**Name:**	quaternionRotation
	Location:	MultiBondLib.Mechanics3D.Utilities
	Signals:	quaternion Q.
		rotation matrix $\mathbf{R}_{rel}$.
	**Name:**	planarRotation
	Location:	MultiBondLib.Mechanics3D.Utilities
	Parameters:	rotation axis $\mathbf{n}$
	Signals:	rotation angle φ.
		rotation matrix $\mathbf{R}_{rel}$.

Figure 7.7: Further special signal elements for 3D mechanics.

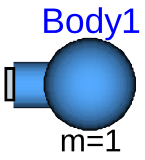
7.2.4 Basic elements

The world model

Unsurprisingly there is also a world model for 3D-Mechanics. There is almost no difference between this model and the corresponding world model in the PlanarMechanics library. The functionality is the same and of course also this world model must be part of all complete models for mechanical systems.

The body
Because of the bondgraphic separation of the translational and rotational parts, also the influence of inertia is modeled by a simple inductance, whereas the inductive field represents the inertia tensor. Both domains stay separate, there is no interaction. Also the gravity force affects only the translational part.

If a body's movement should not be restricted, if it should freely move in space then there is no need to connect to a free-body movement joint. To achieve this, a potential free-body movement joint[2] is attached to the connector of the body model.

	**Name:** **Location:** **Parameters:**	Body MultiBondLib.Mechanics3D.Parts Mass *m*. Inertia tensor elements: I_{11}, I_{12}, I_{13}, I_{22}, I_{23}, I_{33}.

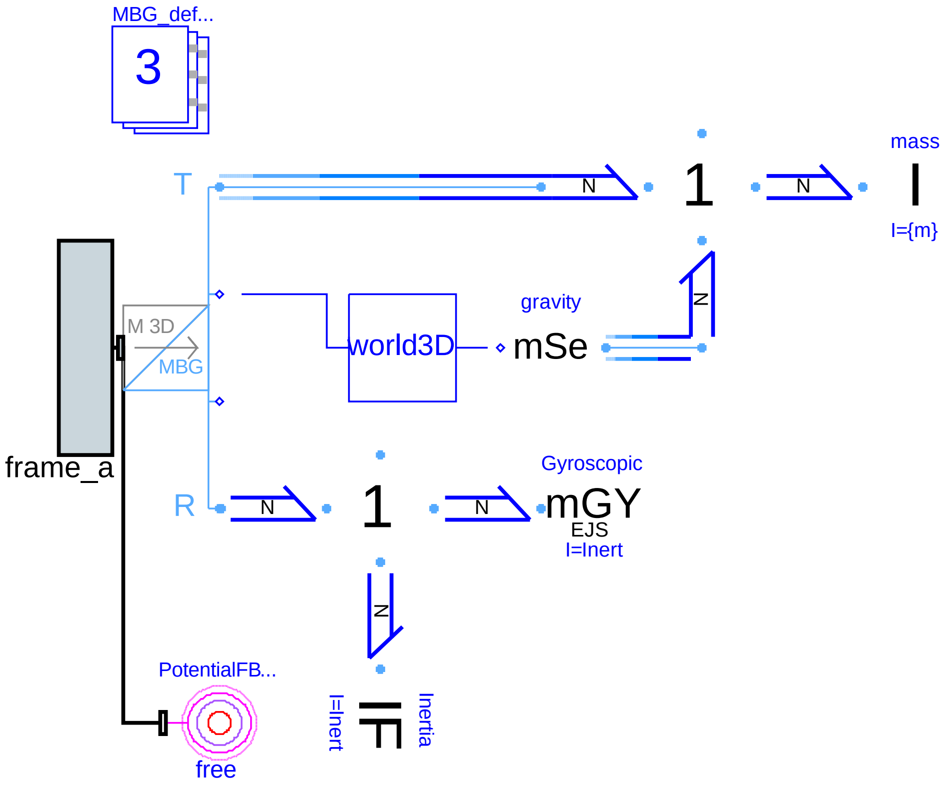
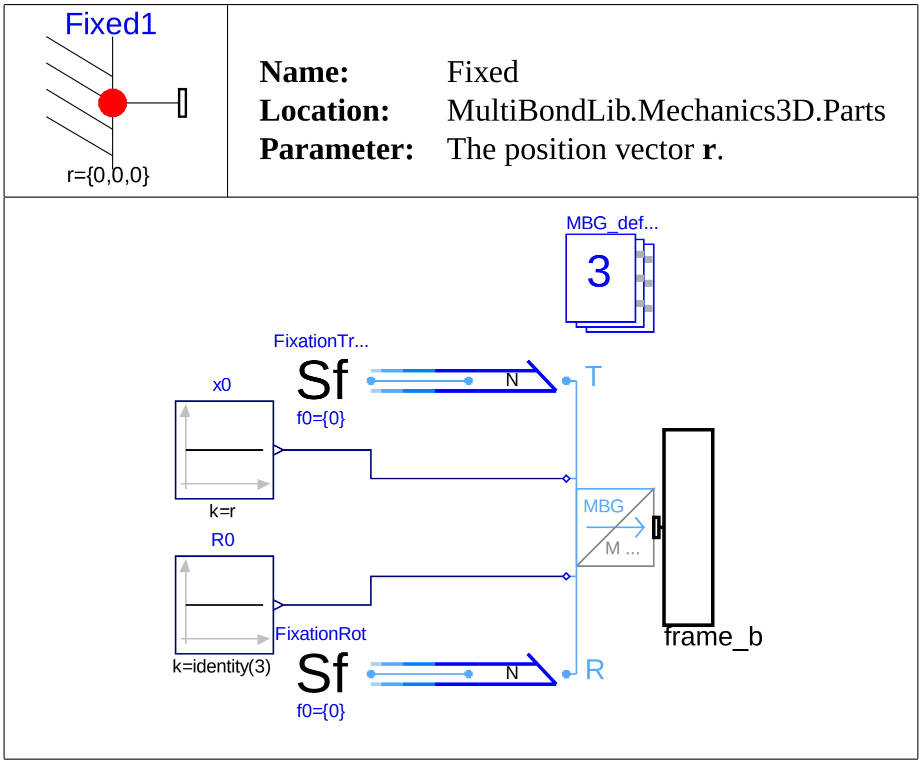


Figure 7.8: Presentation of the body model.

[2] This element will be explained later on, but serves the same functionality as in the PlanarMechanics library

The wall

As in the planar mechanics library the fixation on the wall is modeled by sources of zero flow and constant positional signals. The wall element is a root element.



Name:	Fixed
Location:	MultiBondLib.Mechanics3D.Parts
Parameter:	The position vector **r**.

Figure 7.9: Presentation of the wall model.

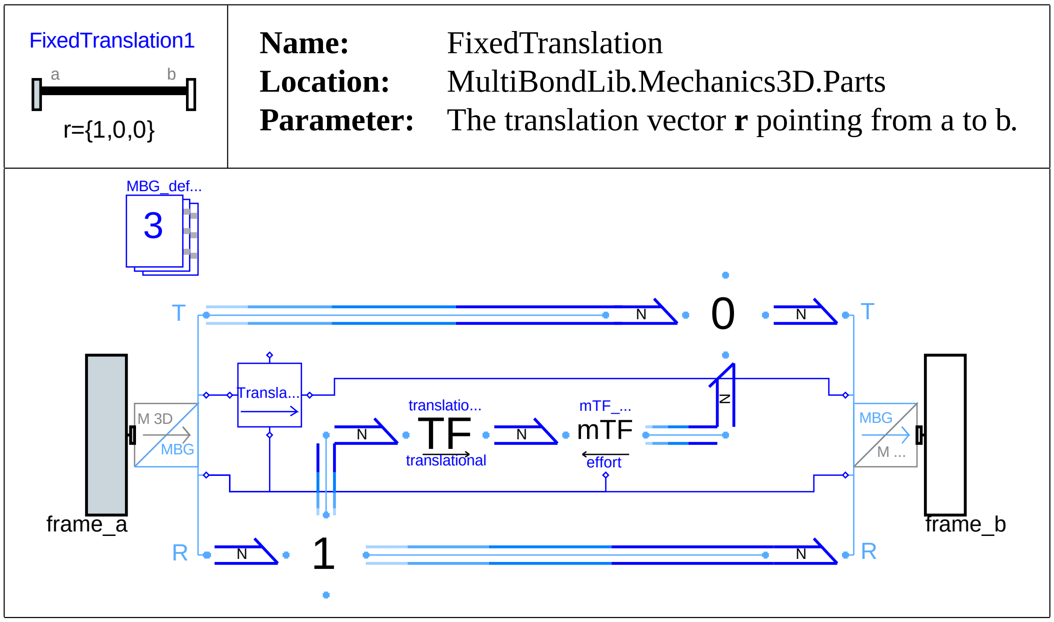
The fixed translation

This model of a massless rod shows beautifully the interaction of the two domains. Because the angular velocity is overall the same, both rotational sides are connected through a 1-junction. The translational sides are coupled through a 0-junction, because there is no difference in force.

The angular velocity is then transformed by the translational transformer and leads to a difference in velocity in the translational domain in dependence of the parameter **r** that is the distance vector between the two connectors. In consequence the force leads to a difference in torque in the rotational domain.

The translational transformer variables are completely resolved in the body system, because **r** is only then a meaningful parameter. Therefore a second transformer is needed that cares about the transition between the two system.

Also the shift of the positional signal needs to be modeled by the predeveloped Translation element.

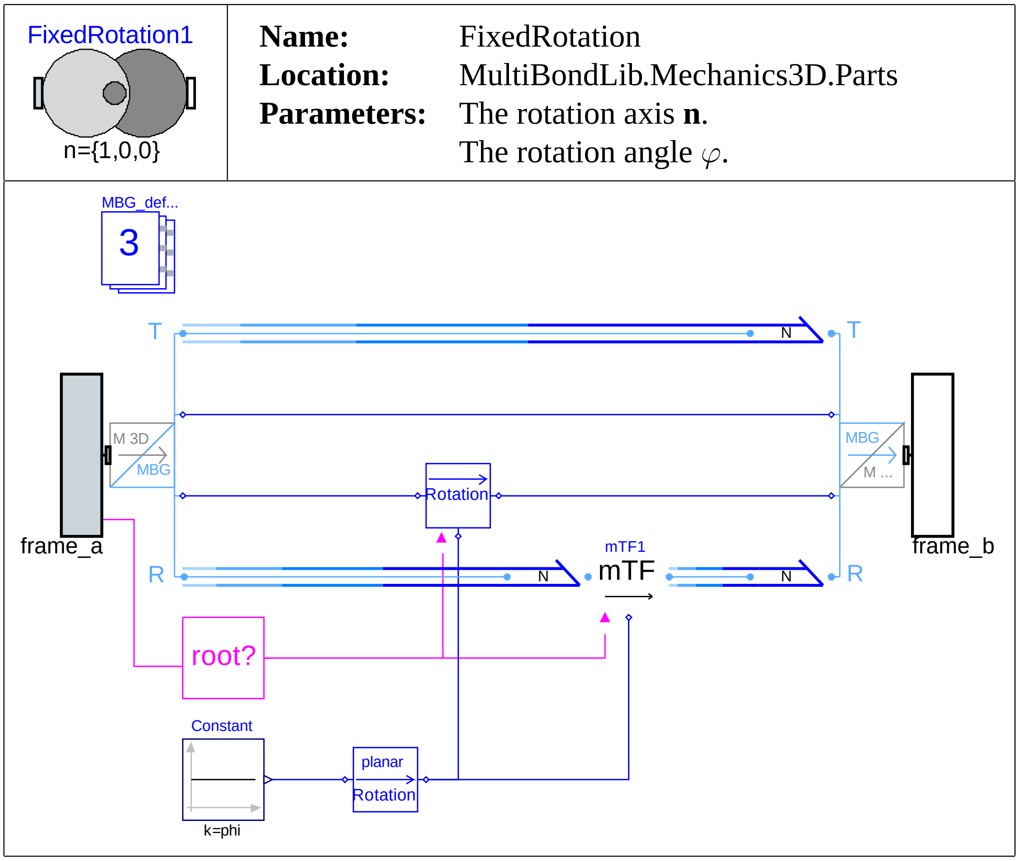


Name: FixedTranslation
Location: MultiBondLib.Mechanics3D.Parts
Parameter: The translation vector **r** pointing from a to b.

Figure 7.10: Presentation of the model for a fixed translation.

The fixed rotation

Unlike the fixed translation, the fixed rotation only affects the rotational domain. It models a fixed relative rotation between the two connected elements that is specified by a planar rotation around an axis **n** with an angle φ.



Name: FixedRotation
Location: MultiBondLib.Mechanics3D.Parts
Parameters: The rotation axis **n**.
The rotation angle φ.

Figure 7.11: Presentation of the model for a fixed rotation.

The constant angle φ is therefore converted to the according planar rotation matrix, which defines the relative rotation between the two connectors. With the aid of this matrix we can transform the bondgraphic variables of the rotational domain by a special modulated transformer and rotate the positional matrix with a “Rotation” element.

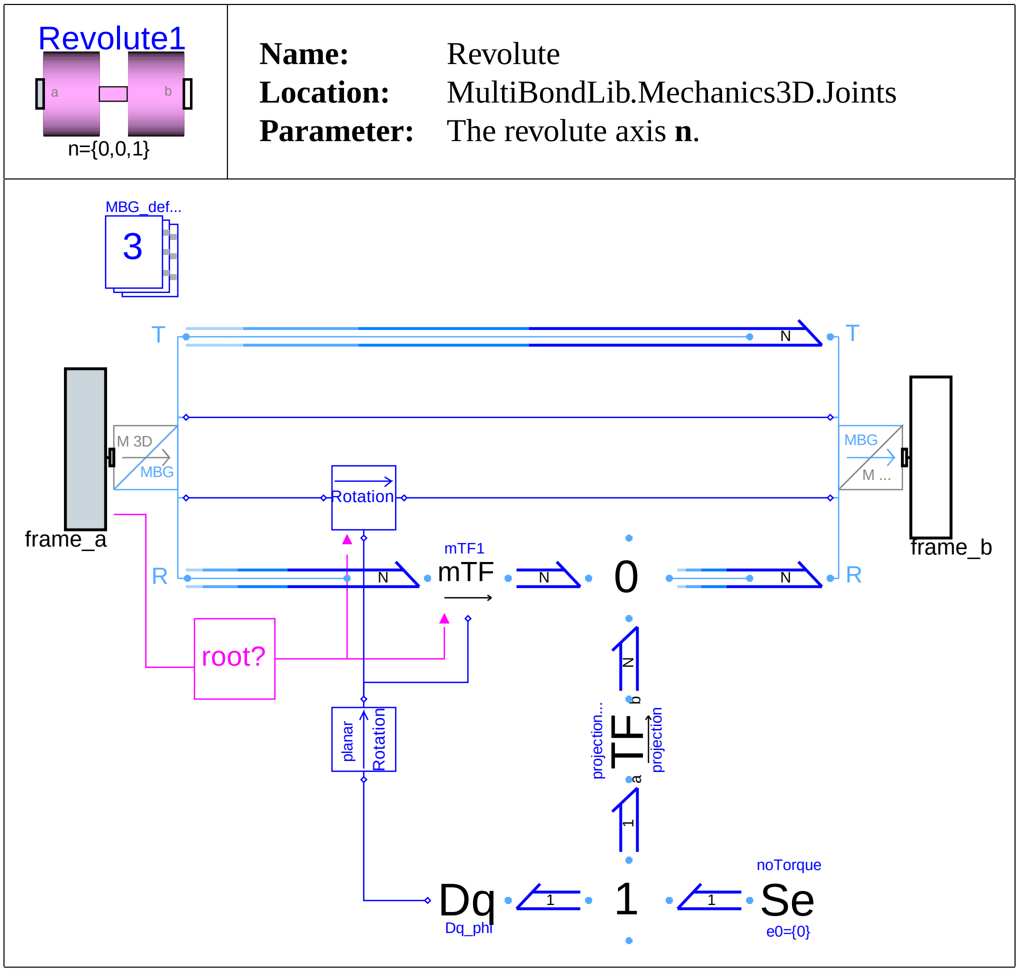
Both of these elements offer two sets of equation, one of which is chosen by a boolean signal. Mostly there is an favorable statement, which lets Dymola avoid the unnecessary solving of linear systems. We can deduce this favorable direction of causality out of the root’s position. In a mechanical system, position and velocities are potential state variables and can be assumed to be known at the place of integration. These places are the root and all successive joints on the tree structure. Hence, it is meaningful to transform the velocity and positional variables from the rooted side to the non-rooted side. This can be achieved by the special Modelica command “isRooted(frame_a.P)”:

```
if rooted(frame_a.P) then
  Rotation1.dirForward = true;
  mTF1.transformFlow = true;
else
  Rotation1.dirForward = false;
  mTF1.transformFlow = false;
end if;
```

This mechanism of deducement is hinted by the violet graphical drawings in the model. This graphical indications will appear also in further models that imply rotations, like the revolute joint.

The revolute joint

There is some similarity between the fixed rotation and the revolute joint, which is a non-fixed planar rotation. Also here, the translational domain remains unaffected.



Name: Revolute
Location: MultiBondLib.Mechanics3D.Joints
Parameter: The revolute axis **n**.

Figure 7.12: Presentation of the revolute joint.

Because there is a difference of angular velocity along the rotation axis, the two connectors are coupled through a 0-junction. Also there acts no torque along the rotation axis. This can be modeled by projecting a single source of zero effort into the 3D-Multibond with the aid of an transformer.

The revolute angle is then measured by a bondgraphic position sensor, and transformed into a rotation matrix via the PlanarRotation element. As in the previous model, we can then transform the variables of the rotational domain by the usage of this matrix. Again these statements are made in dependence of the root position.

Whether this transformation is done before or after the 0-junction doesn't matter for a revolute joint, because the rotation axis itself remains unaffected by the rotation. For convenience, the transformation is done before the 0-junction.

The spherical joint

The spherical joint models a free rotational movement. This movement can then be expressed either by the use of cardan angles or by the use of quaternions. This model implements both variants, which can be selected by the user via the parameter useQuaternions.

In difference to the revolute joint, the angular velocity is not integrated directly by a position sensor. It is first measured by a flow sensor, and then the differential equations are stated by the special elements toCardanAngles or toQuaternion. The resulting positional signals (either in form of three angles or a quaternion) are then transformed to a rotation matrix by the elements cardanRotation or quaternionRotation.

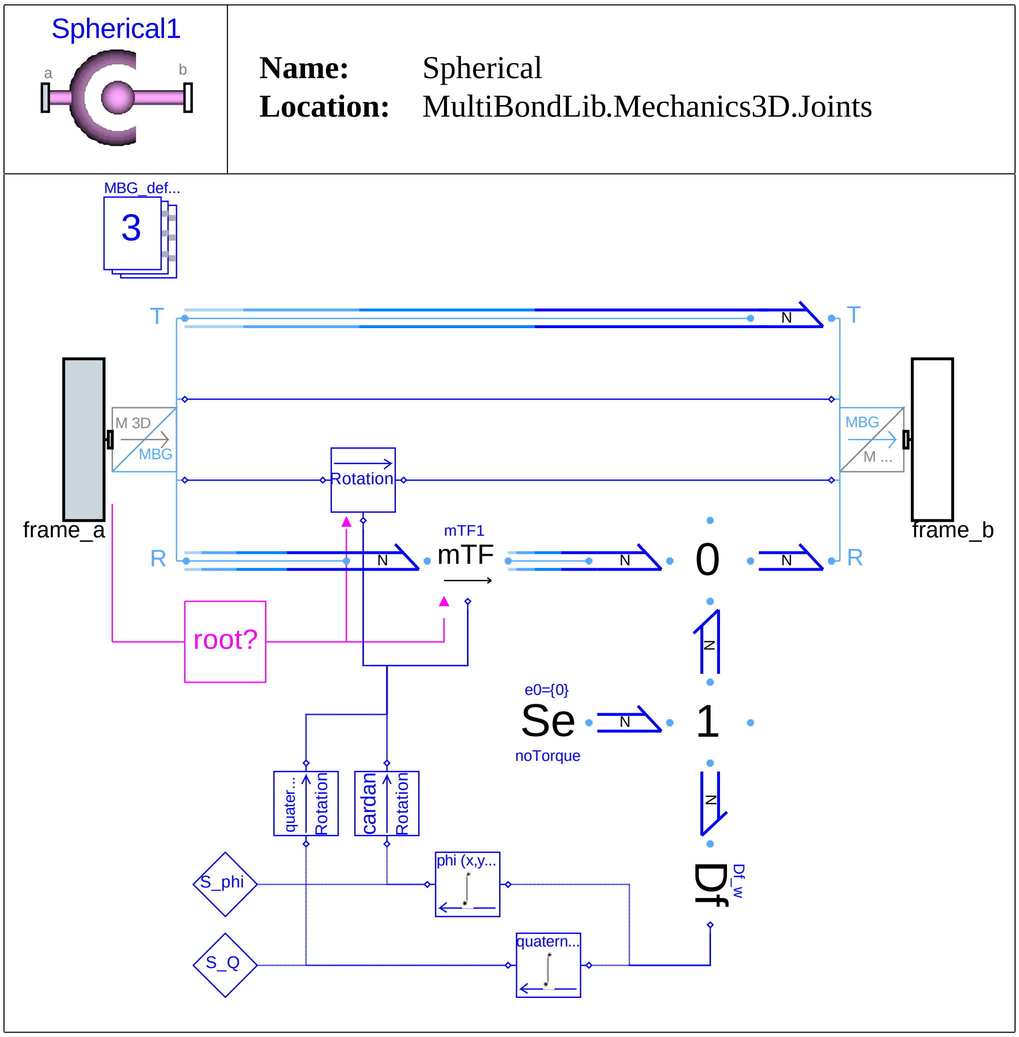


Figure 7.13: Presentation of the spherical joint.

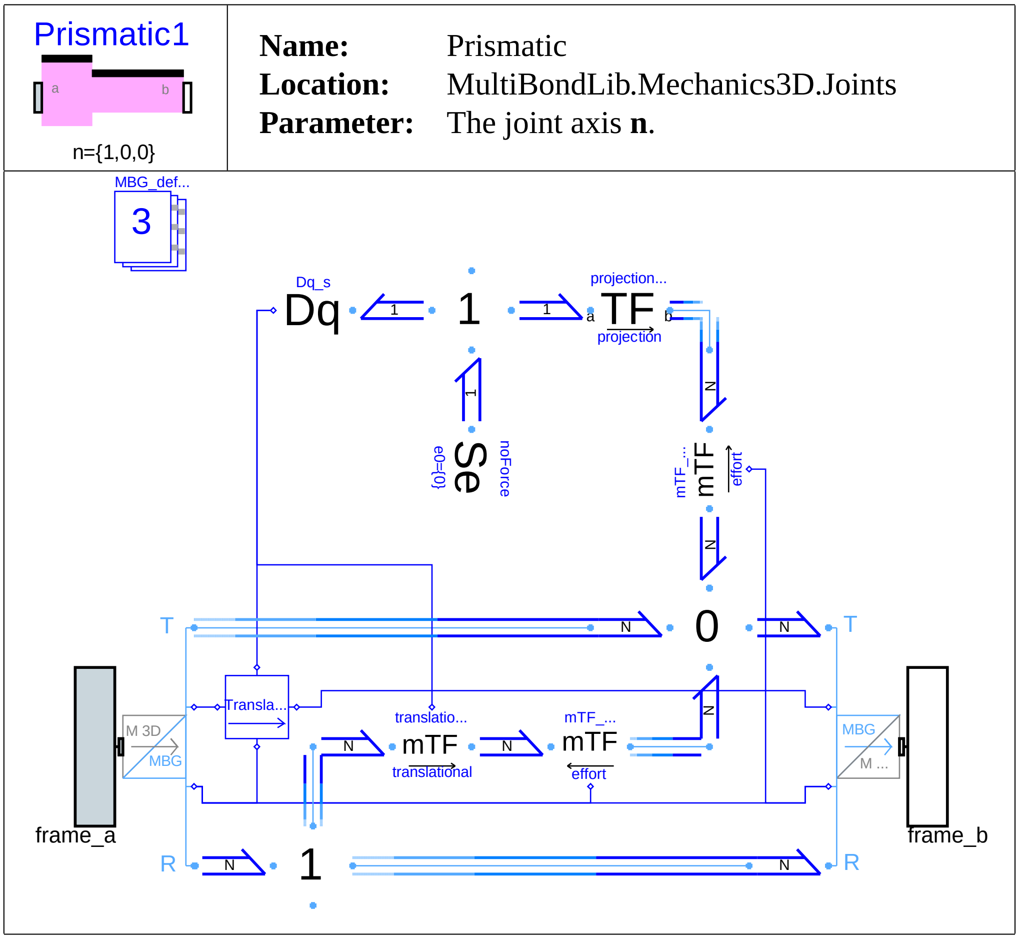
The dotted connections of all these elements denote that they themselves as well as their connections are conditional. Dependent on the user's choice, one of the two possible paths is deactivated. This is implemented by using the Modelica feature of conditional declaration.

The bondgraphic variables can be transformed with the rotation matrix. It does now matter where the transformation happens. Because it was chosen to derive the rotation matrix (and the cardan angles or quaternions respectively) out of the angular velocity of the body system at frame b, the transformation has to happen at the left-hand side of the 0-junction.

The prismatic joint

The prismatic joint is the most complex model. As a subpart it contains the model of the Fixed Translation element. Because this time the length of the translation is variable, there is an additional modulating signal for the corresponding element that emanates from the position sensor representing the current length s of the prismatic joint.

The unit vector **n** denotes the joint axis. The prismatic joint velocity v_s and the corresponding force f_s are projected in the three-dimensional space. An additional transformer becomes necessary, because the joint axis is resolved in the body system and has first to be transformed into the inertial system. The force f_s equals zero, because there acts no force along the joint axis. This is modeled again by a source of zero effort.



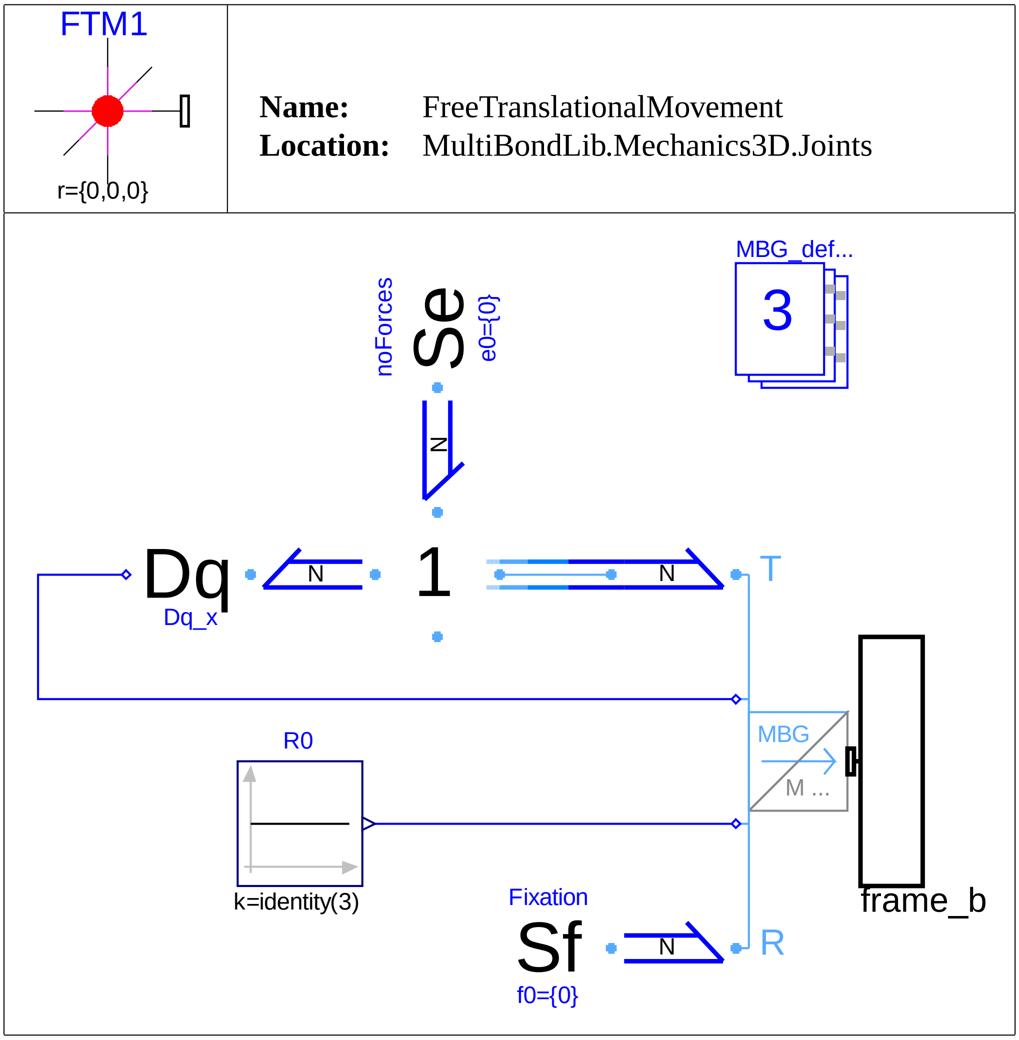
Name: Prismatic
Location: MultiBondLib.Mechanics3D.Joints
Parameter: The joint axis **n**.

Figure 7.14: Presentation of the prismatic joint.

The free translational movement joint

The free translational movement gives all translational degrees of freedom to a body and fixes its orientation. It is a root element.

The model is pretty simple. The rotational domain corresponds to the wall model whereas the translational part links the velocity with the positional signal by the simple differential equation $\dot{\mathbf{x}} = \mathbf{v}$.



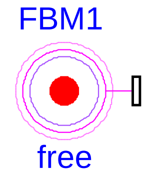
Name: FreeTranslationalMovement

Location: MultiBondLib.Mechanics3D.Joints

Figure 7.15: Presentation of the free translational movement joint.

The free-body movement joint

The free-body movement joint defines all 6 degrees of freedom. It results out of a composition of the free translational movement joint and the spherical joint.



Name: FreeBodyMovement
Location: MultiBondLib.Mechanics3D.Joints

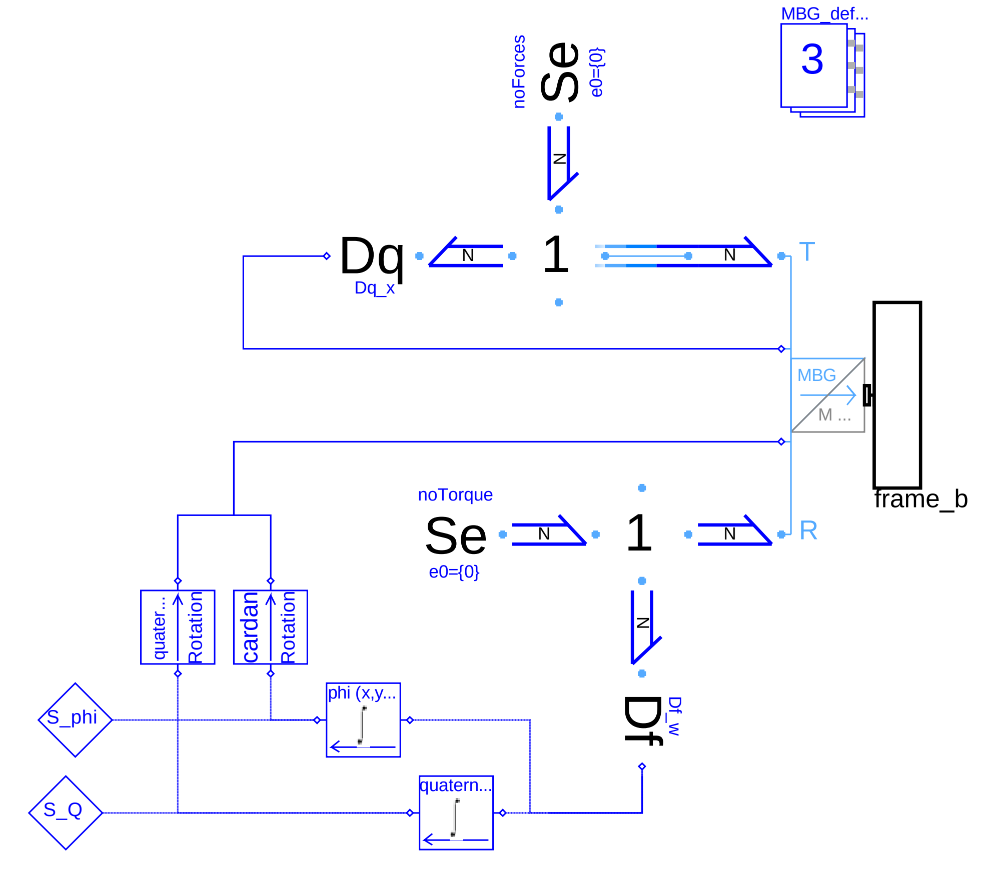


Figure 7.16: Presentation of the free-body movement joint.

The potential free-body movement joint
To enable potential rooting, a potential free-body movement joint has been developed. The integrators of this joint are now created to be optional, and are only defined in the root case. Otherwise the differential equations are replaced with dummy equations. The usage of this joint has already been presented in the body model.

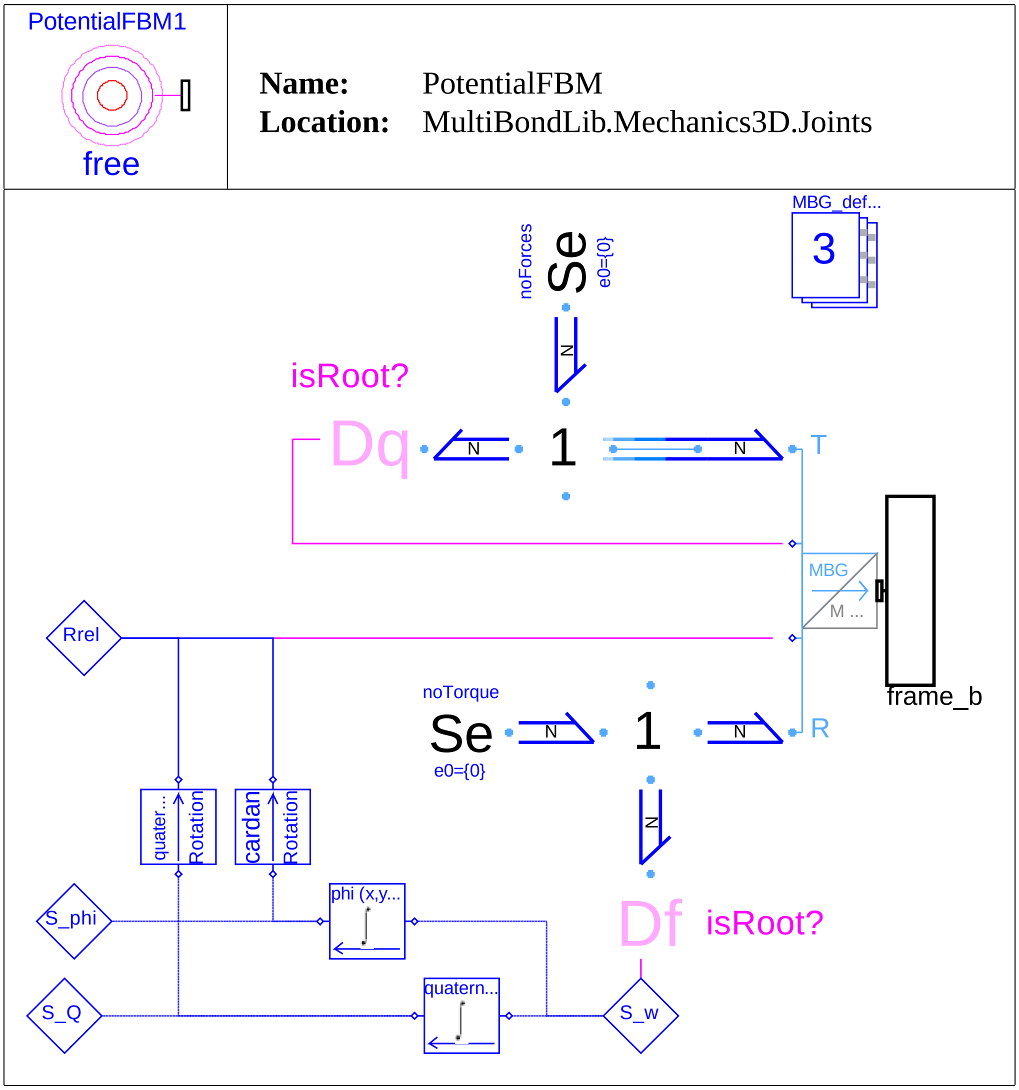
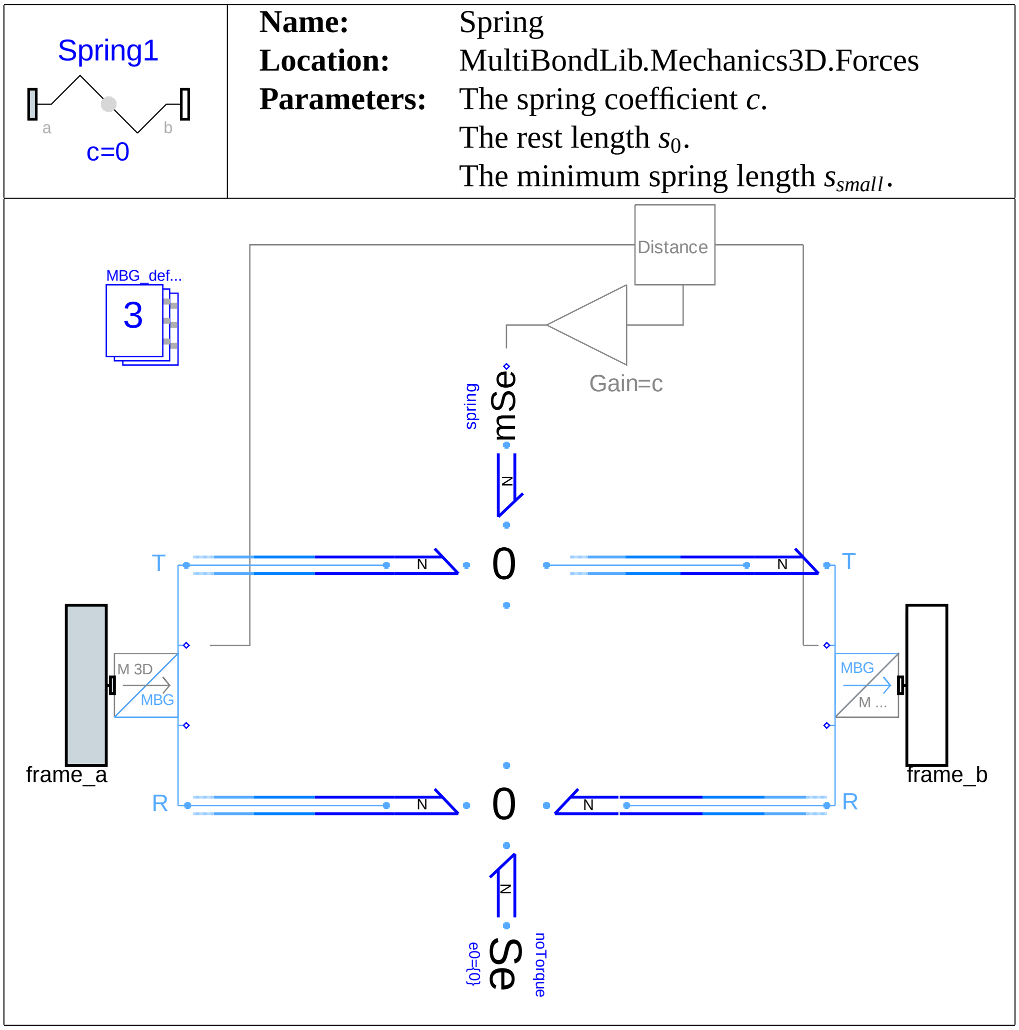


Figure 7.17: Presentation of the potential free-body movement joint.

The spring

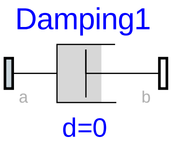
The spring only affects the translational domain. Again it is modeled as state modulated source of effort. This model is simply an extension of the planar model.



Name: Spring
Location: MultiBondLib.Mechanics3D.Forces
Parameters: The spring coefficient c.
The rest length s_0.
The minimum spring length s_{small}.

Figure 7.18: Presentation of the model for a linear spring.

The damper
Also the damper model is nothing more than an extension of the corresponding planar model.It also only affect the translational domain.

	**Name:** Damping **Location:** MultiBondLib.Mechanics3D.Forces **Parameter:** The damping coefficient d.

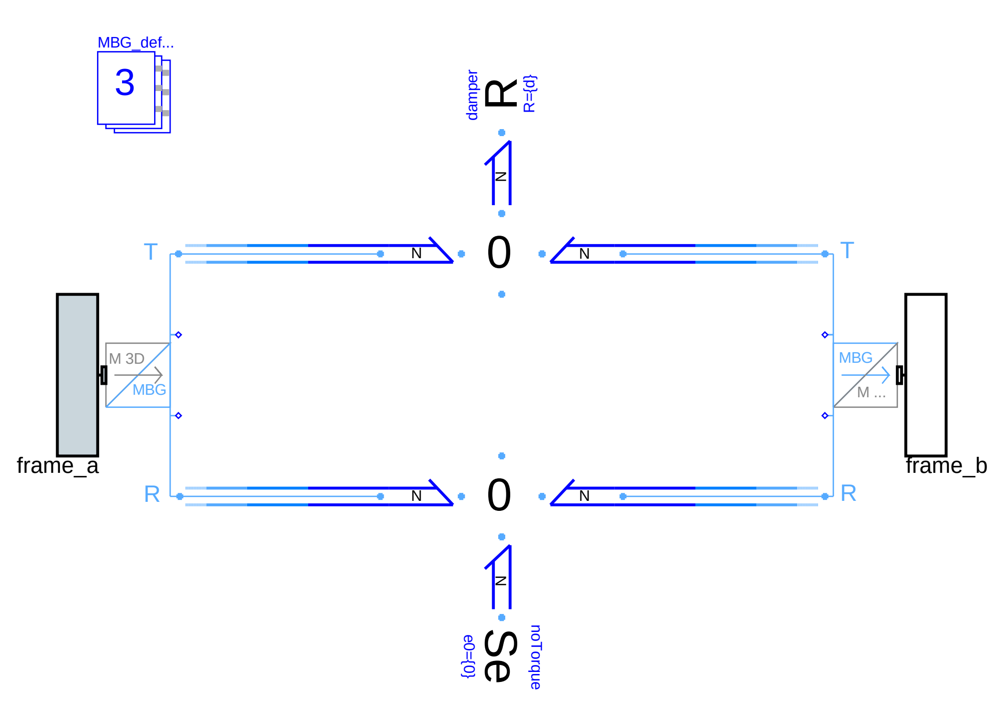


Figure 7.19: Presentation of the damper model.

7.2.5 Closing of kinematic loops

The same problems arise by the closure of kinematic loops as in the planar mechanics library. The methods to handle these problems are then also unsurprisingly the same and they are already implemented in the models presented above. Again the connector's redundant set of potential variables are marked according to the model characteristic using the commands: "defineRoot()", "definePotentialRoot()" and "defineBranch()".

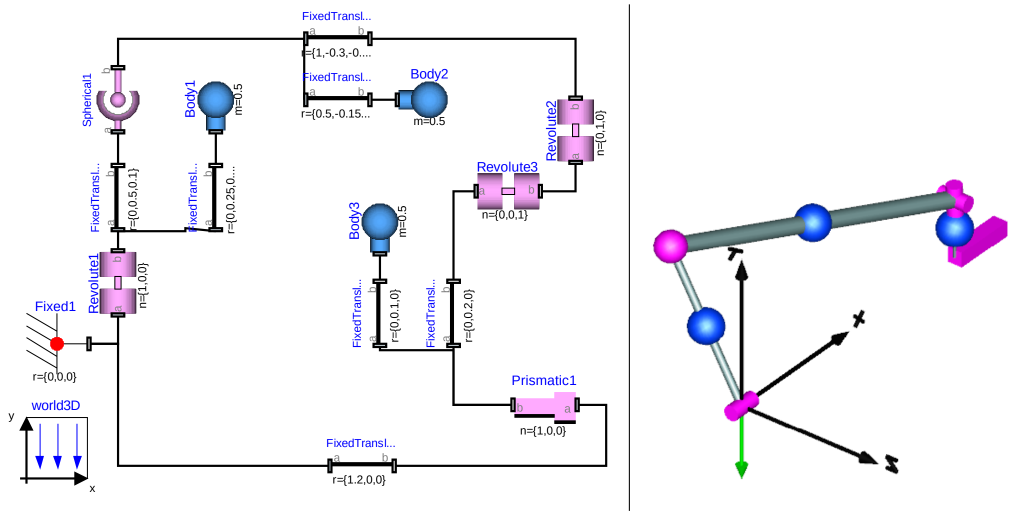


Figure 7.20: A kinematic loop in 3D. The loop in this model is closed automatically.

Planar loops in 3D-Mechanics

An additional difficulty is the closure of planar loops in 3D-Mechanics. Figure 7.21 presents an example of a symmetric planar loop that demonstrates the effect of the centrifugal force. An upright standing spring damper unit is squeezed by two rotating masses.

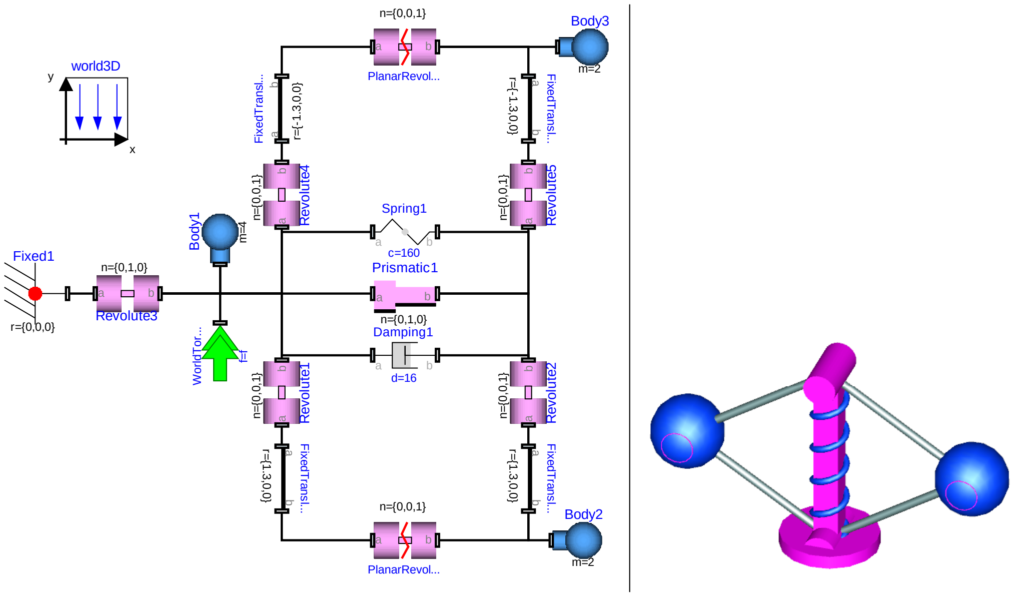


Figure 7.21: Two planar kinematic loops. The loops are closed with a special cut joint.

In the model diagram, one can see that the two parts of the loop are closed each with a special kind of revolute joint: The planar cut revolute joint. The usage of this joint becomes necessary, because a classic revolute joint would cause a model singularity. To understand this, let's take a look at the planar loop: The loop is only formed by revolute joints with the same rotation axis. The translation movement is then restricted to a single plane by only two of the revolute joints and the restriction of the third revolute joint becomes superfluous and creates a singularity of the model. Therefore a planar cut revolute joint was developed that omits these superfluous restrictions and the differential equations that define the integrators. The resulting model then only consists of holonomic constraints. That is: Equations that constraint the positional variables.

7.3 Holonomic and non-holonomic constraints

The closure of kinematic loops confronted us with the concept of holonomic constraints. These are constraints on the positional level, at the top level of integration.

Non-holonomic constraints are constraint at the level of velocity or acceleration. A typical example is $v = 0$. Sometimes it is possible to transform such a constraint into a holonomic constraint by using its integrated form (e.g: $x = const$), but this is not always achievable. Typically these non-holonomic constraints decrease the number of degrees of freedom on the level of velocity, but not necessarily on the level of position.

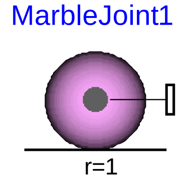
In the following, the behavior of ideal rolling objects on a simple plane will be examined. The rolling plane is defined by its unit normal vector **n** pointing upwards and the plane's distance d from the inertial system's origin. Ideal rolling objects do have the property that the difference of velocity between object and ground at the virtual point of current contact is zero i.e. there is no slippage. If we assume that the ground is not moving, the statement above can be simplified to:

The object's velocity at the current virtual contact point is zero.

This leads to the statement of non-holonomic constraints. Because the virtual contact point is constantly changing, it is difficult (or even impossible) to find an integrated form of this constraint. Anyhow the statement above also implicitly implies that there is a contact between object and plane. This will lead to a holonomic constraint.

7.3.1 Ideal rolling marble

An easy example of an ideal rolling marble will give us a closer insight into this subject. To state the non-holonomic constraints, it is necessary to determine the point of current contact. This is an easy task for a spherical marble.

	**Name:**	MarbleJoint
	Location:	MultiBondLib.Mechanics3D.RollingObjects
	Parameters:	The radius r. The normal vector of rolling plane $\mathbf{n}$. The distance d of rolling plane from origin.

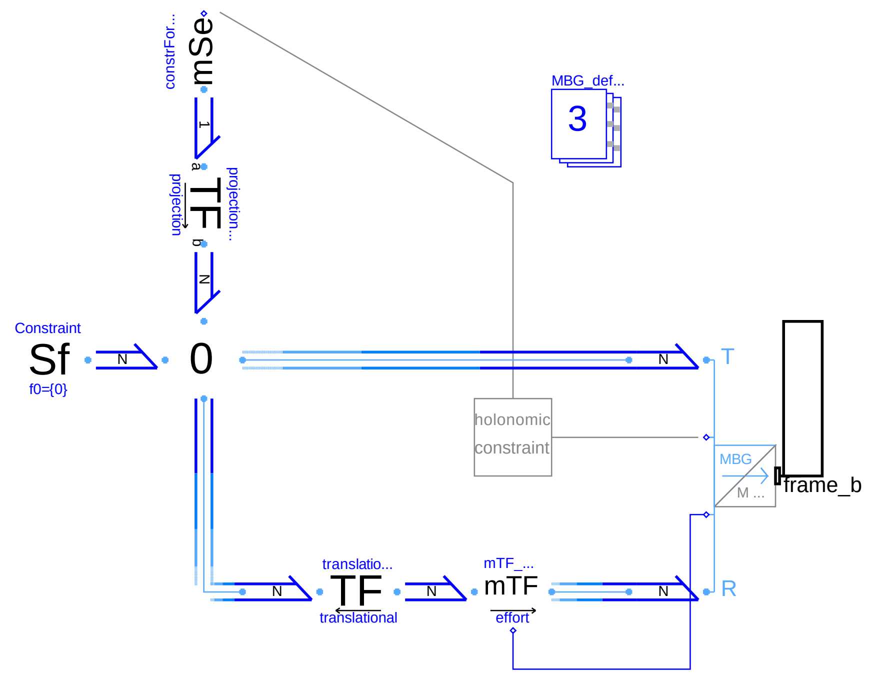


Figure 7.22: Presentation of the model for the movement constraints of an ideal rolling marble.

The geometry of an ideal marble can be simply defined by its radius r. The vector $\mathbf{r}_{cp}$ points from the marble's center to the point of contact and is resolved in the inertial system. This vector is pointing in the negative direction of $\mathbf{n}$ and is of length r.

$$\mathbf{r}_{cp} = -\mathbf{n} * r$$

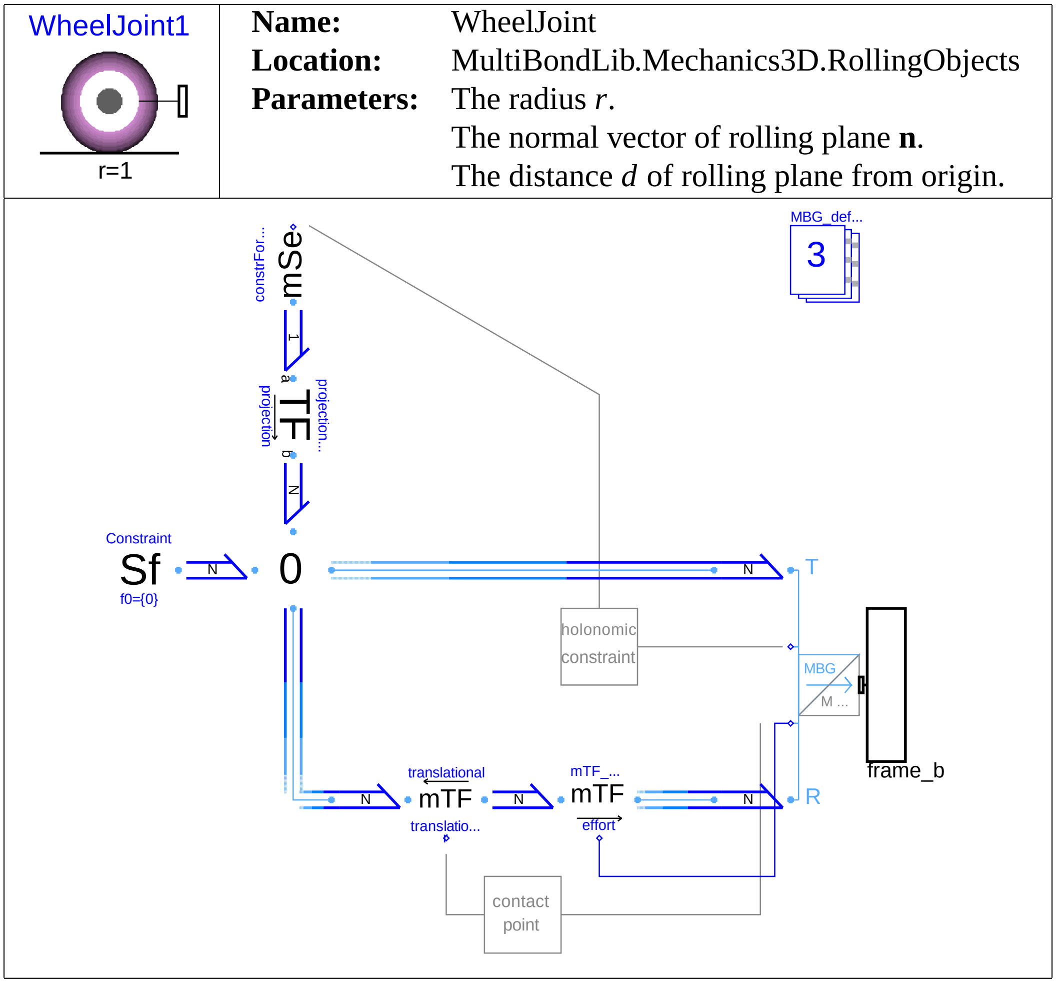
The position of the marble's center is then constrained in direction of $\mathbf{n}$. The velocity at the contact point is then the summation of the translational velocity of the marble's center and the along the vector $\mathbf{r}_c$ transformed angular velocity. This sum has then to be zero.

Figure 7.22 shows the bondgraphic implementation of the contact between marble and ground. The translational transformer that transforms the angular velocity is defined on variables resolved in the inertial system and therefore preceded by a classic transformer. The flow of zero effort implements the actual constraint equation. Unfortunately this is generating one constraint equation too many. Also the velocity in direction of the normal vector $\mathbf{n}$ is stated to be zero. But this statement is already implied by the holonomic constraint:

$$\mathbf{x} \cdot \mathbf{n} = r$$

Therefore we have to reduce the set of constraint equations by the introduction of a single source of arbitrary effort. The value of this effort results out of the holonomic constraint equation presented above. This is graphically indicated by a gray box.

7.3.2 Ideal rolling wheel



Name: WheelJoint
Location: MultiBondLib.Mechanics3D.RollingObjects
Parameters: The radius r.
The normal vector of rolling plane $\mathbf{n}$.
The distance d of rolling plane from origin.

Figure 7.23: Presentation of the model for the movement constraints of an ideal rolling wheel.

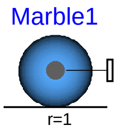
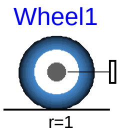
The model of an ideal rolling wheel doesn't differ much from that of an ideal rolling marble. The only change is the different geometry and in consequence a different, more complex computation of the contact point, because the vector $\mathbf{r}_{cp}$ isn't constant any more. It depends on the wheel's orientation.

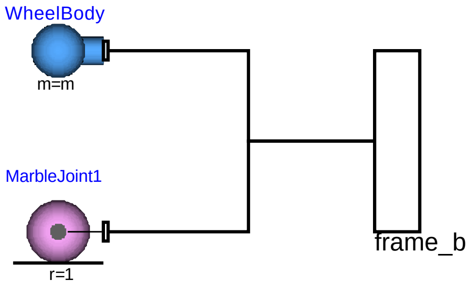
The wheel geometry is defined to be a flat disc with radius r. The wheel axis is denoted by the unit vector $\mathbf{e}_{Axis}$ and defines the wheel plane. The vector $\mathbf{r}_{cp}$ is then located on this plane, has length r, and is aligned to $-\mathbf{n}$.

$$r_{cp} = -r \cdot normalize((\mathbf{n} - (\mathbf{n} \cdot \mathbf{e}_{axis})\mathbf{e}_{axis})$$

Figure 7.23 shows the bondgraphic implementation of the contact between wheel and ground. This model does differ little from the marble model. Only the transformer is exchanged by a modulated variant of it. The input signal of this transformer is the vector $\mathbf{r}_{cp}$. The computation of this variable is modeled by equations according to the above statement. This is indicated by a gray box in the graphic model. Also this model needs to be enhanced by a body model to get a complete model of a rolling wheel.

To get a complete model of an ideal rolling marble or an ideal rolling wheel, we have to connect the contact model with the body model that also includes a potential free-body movement joint. See figure 7.24. The resulting model has then 5 degrees of freedom on the positional level and 3 degrees of freedom on the velocity level.

		**Name:** Marble, Wheel **Location:** MultiBondLib.Mechanics3D.RollingObjects **Parameters:** The radius r. The normal vector of rolling plane $\mathbf{n}$. The distance d of rolling plane from origin.



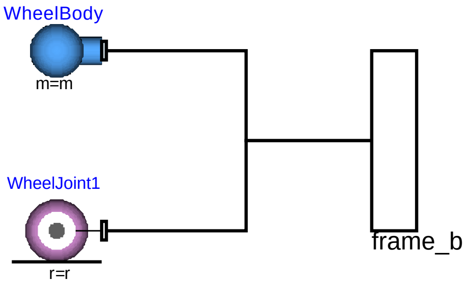


Figure 7.24: Complete models of a marble and a wheel. The potential free-body movement joint is included in the corresponding body model.

7.4 An example: the bicycle

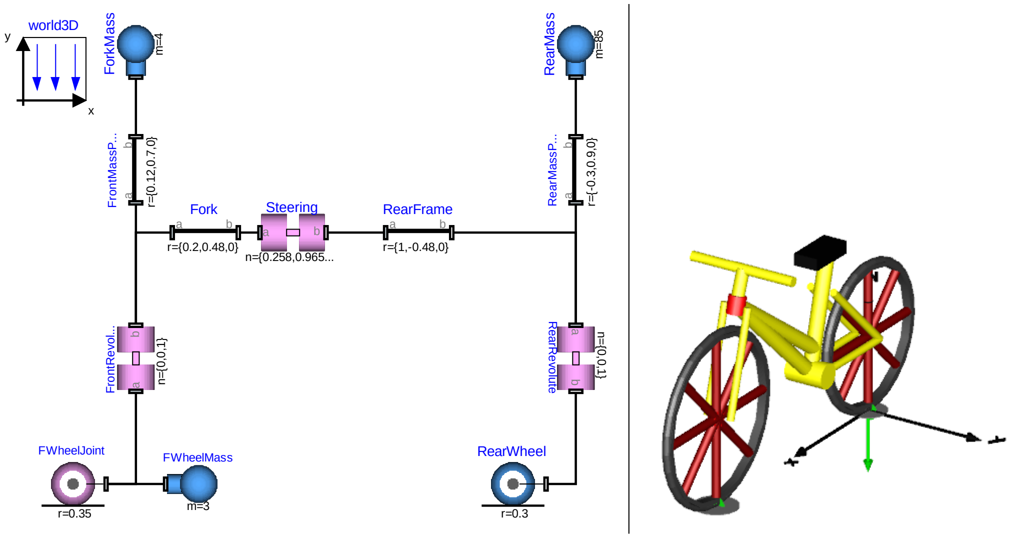


Figure 7.25: Model of a bicycle.

This chapter draws to a close with an example in reference to A. M. Bos[3]. An uncontrolled ideal rolling bicycle is modeled. Such a bicycle has 7 degrees of freedom on the level of position and 3 degrees of freedom on the level of velocity. It is a partially stable system for a specific range of driving velocities. A nice description of the dynamic behavior of a bicycle can be found in a paper of K. Aström[2]. The relevant parameters for mass and geometry for this specific model are taken out of a paper of A. L. Schwab, J. P. Meijaard and J. M. Papadopoulos[26]. This bicycle is self-stabilizing for driving velocities from $4.3ms^{-1}$ to $6.1ms^{-1}$.

The selected state variables are the cardan angles of the rear wheel and their derivatives. These three cardan angles represent the orientation on the plane, the lean of the rear frame and the roll angle of the rear wheel. Additional state variables are the position of the rear wheel on the plane, the angle of the steering joint and the angle of the front revolute joint. Each of the two wheels defines one holonomic constraint equation. Hence, there is a partial kinematic loop that results in one non-linear equation for the angle of the rear revolute joint.

The model of figure 7.25 can be unwrapped, that is, all the objects are replaced by their underlying bond graphs. It results in a huge bond graph that can then be simplified by removing superfluous junctions and signal flows. The final bondgraphic model is presented in the appendix B.

7.5 Evaluation

7.5.1 Selection of state variables

The selection of the state variables is of major importance for the efficiency of the resulting simulation. Usually this selection is automatically done by Dymola. Nevertheless certain variables are suggested to Dymola via the Modelica attribute: “StateSelect”. These variables are always the states of a joint’s relative position and motion. So each joint is declaring state variables, unless there is a kinematic loop.

In the latter case, there are often many possible sets of state variables. Dymola uses then the feature of dynamic state selection, where the state variables are chosen at run time. This leads to a slower simulation and to many unnecessary equations. Hence, it is strongly recommended to determine the state variables manually in such a case. This can be achieved by the activation of the boolean parameter “enforceStates” of a specific joint. E.g. in the kinematic loop of figure 7.20 the first revolute joint is enforced to declare the state variables. All models that contain potential state variables do have the option of a manual enforcement of their state variables via the parameter “enforceStates”.

7.5.2 State variables for orientation

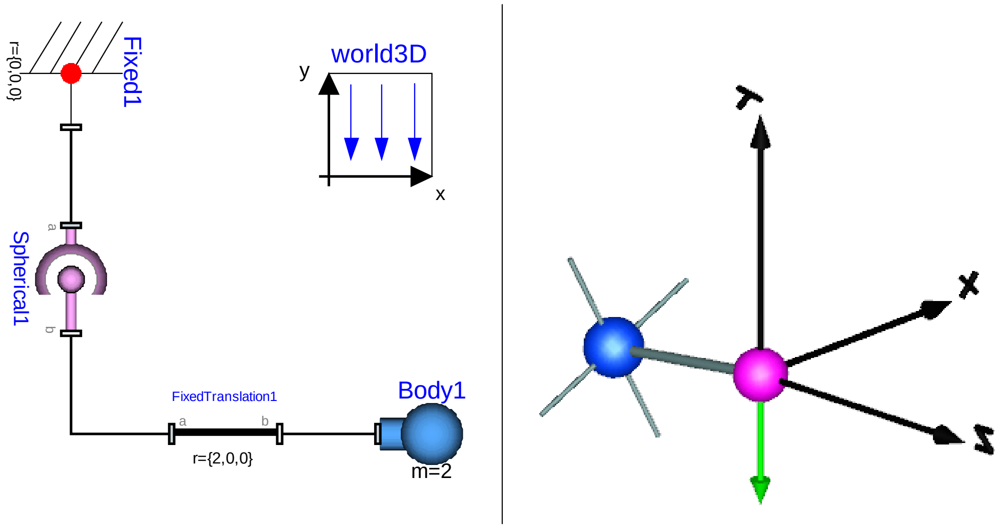


Figure 7.26: This model demonstrates the gyroscopic effect.

The following example demonstrates the gyroscopic effect and shows the importance of a proper choice of the state variables for a general rotation. The setup of the experiment is well known and shown in figure 7.26: A mass, typically a bicycle wheel is rotating with high speed. It is attached with a rod of negligible weight to a rope that is fixed at the ceiling. The initially fixed turning axis is then released. One might intuitively expect the wheel to fall down like a pendulum, but due to the gyroscopic effect it starts to turn around in circles. The spherical joint offers several possible state selections. The parameter “useQuaternions” determines if the relative orientation should be expressed by quaternions or by cardan angles. In the latter case the sequence of rotation axes can be chosen by an additional parameter “sequence_angles”.

Three possible selections are examined:

1. The state variables are the three cardan angles and their derivatives. The cardan angle sequence is chosen wisely to be $\{y,z,x\}$, so the fast rotation of the wheel can solely be expressed by the last cardan angle.

2. A badly chosen sequence $\{x,y,z\}$ of cardan angles and their derivatives form the sequence.

3. Quaternions and the vector of angular velocity are selected to be the state variables.

The results are presented by the curves of the plot 7.27. The plot shows the height of the wheel over a period of 3 seconds. The simulation method was Dassl with an error tolerance of $1 \cdot 10^{-4}$. The simulation with the wisely chosen cardan angles and their derivatives led to the most correct result with a minimum of 294 integration steps for 10 seconds. The integration of the angular velocity onto quaternions caused a slight error and demanded a huge computational effort of 25574 integration steps. A badly chosen sequence of cardan angles led to a catastrophic result. The computational effort was enormous: 53618 integration steps. This last example shows that simulating an ill-defined model can be quite dangerous. Dassl produced wrong results but no error message. A smaller error tolerance of the integration algorithm prevents such huge errors but increases the computational effort. This dependency is presented in table 7.1.

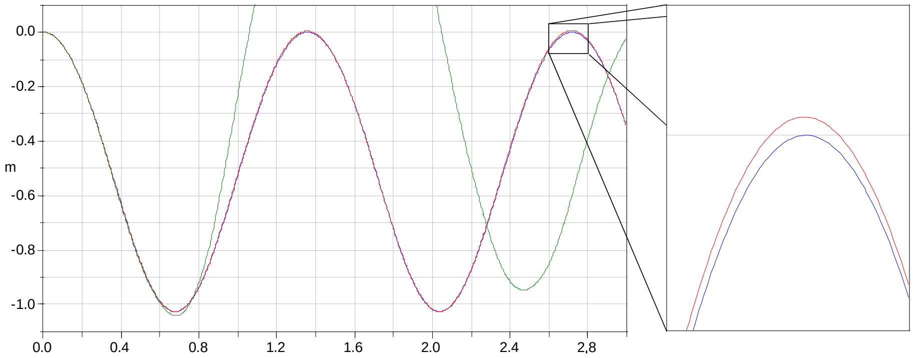


Figure 7.27: The height of the rotating disc for a period of 3 seconds. The different colors represent different state selections: Wisely chosen cardan angles (blue), quaternions(red) and badly chosen cardan angles (green).

	good cardan angle seq.		quaternions		bad cardan angle seq.	
tolerance	error	steps	error	steps	error	steps
$1.0 \cdot 10^{-4}$	$4.9 \cdot 10^{-4}$	$2.9 \cdot 10^{3}$	$5.0 \cdot 10^{-3}$	$2.6 \cdot 10^{4}$	$1.8 \cdot 10^{-0}$	$5.4 \cdot 10^{4}$
$1.0 \cdot 10^{-6}$	$9.7 \cdot 10^{-6}$	$6.2 \cdot 10^{3}$	$3.1 \cdot 10^{-4}$	$4.8 \cdot 10^{4}$	$2.9 \cdot 10^{-4}$	$9.5 \cdot 10^{4}$
$1.0 \cdot 10^{-8}$	$1.2 \cdot 10^{-7}$	$1.4 \cdot 10^{4}$	$1.1 \cdot 10^{-5}$	$8.4 \cdot 10^{4}$	$3.5 \cdot 10^{-5}$	$2.0 \cdot 10^{5}$
$1.0 \cdot 10^{-10}$	$1.2 \cdot 10^{-7}$	$2.3 \cdot 10^{4}$	$1.1 \cdot 10^{-6}$	$1.4 \cdot 10^{5}$	$3.0 \cdot 10^{-6}$	$4.4 \cdot 10^{5}$

Table 7.1: This table presents the dependency of the integration error and the computational effort on the error tolerance of the integration algorithm Dassl. The number of integration steps is counted for a simulation period of 10 seconds. The error of the disc height is measured after 2.7 seconds.

7.5.3 Comparison with the MultiBody library

The Mechanics3D library and the MultiBody library contain elements for the same purpose in a very similar structure. It is therefore easily possible to translate the model for a mechanical system from one library to the other.

Both libraries are examined with respect to efficiency. The complexity of the resulting systems of equation and the computational effort of the simulation are compared for a given set of examples. Table 7.2 contains the result of this examination. There is hardly any significant difference.

	MultiBody			Mechanics3D		
experiment	linear equ.	non-lin. equ.	steps	linear equ.	non-lin. equ.	steps
Pendulum	0	0	207	0	0	207
Double pendulum	2	0	549	2	0	549
Crane crab.	2	0	205	4	0	205
Gyroscopic exp. with Cardans	2, 2	0	294	3, 2	0	294
Gyroscopic exp. with Quaternions	4, 3	4	24438	4, 2	4	25574
Planar Loop	8, 2	2	372	6, 2, 2	2	372
Centrifugal exp.	10, 2, 2	2, 2	70	16, 2, 2	2, 2	70
Four bar loop*	10, 5, 2	5	446	9, 5, 2	5	625
Bicycle*	15, 5, 3, 2	1	97	15, 3	1	84

Table 7.2: This table lists the sets of linear equations and the sets of non-linear equations for a given set of experiments. The number of integration steps is counted for a simulation period of 10 seconds. The simulation method was Dassl with a tolerance of $1 \cdot 10^{-4}$*. A * denotes that the parameters of the experiment setup differ slightly between the two libraries.*

In fact, the generated equations of both libraries are very similar. There is only one important difference: In contrast to the bondgraphic library, the translational velocity is not part of the

MultiBody's connector variables.[3] The elements are connected by the translational position. The equations for the translational velocities are then derived (if necessary) by differentiation. In the bondgraphic models of Mechanics3D these equations are explicitly stated.

The handling of kinematic loops can be improved by the creation of special elements where the non-linear equations are analytically solved. The MultiBody library provides a comprehensive set of such elements and allows a more efficient simulation of kinematic loops. The Mechanics3D abstains from such a solution because a meaningful translation into the bondgraphic framework has not yet been found.

Table 7.3 presents a comparison between the two bondgraphic libraries for mechanics: PlanarMechanics and Mechanics3D. The raising of planar models into the 3D-space is not associated with a loss of efficiency.

	PlanarMechanics			Mechanics3D		
model	linear equ.	non-lin. equ.	steps	linear equ.	non-lin. equ.	steps
Pendulum	0	0	207	0	0	207
Double pendulum	2	0	549	2	0	549
Crane crab.	3	0	205	4	0	205
Planar Loop	$4,2,2$	2	372	$6,2,2$	2	372

Table 7.3: This table lists the sets of linear equations and the sets of non-linear equations for a given set of experiments. The number of integration steps is counted for a simulation period of 10 seconds. The simulation method was Dassl with a tolerance of $1 \cdot 10^{-4}$.

[3] Please note: this inequality of the two connector types prevents a direct combination of the two libraries and their components.

Chapter 8

Impacts

8.1 Motivation

If two hard objects collide, it appears to cause a discrete event, because there seems to be a sudden change of the objects' velocities. Of course, there is still a continuous process underneath that just goes on so fast that it is perceived as a single event. For example, the timespan for a collision between two metallic bodies is usually in the range of a few milliseconds or even less.

Special models that include extremely stiff springs make it possible to model these hard impacts with continuous models only, but such stiff systems are highly problematic to integrate. A high computational effort and clever algorithms are demanded and still the result's precision is often unacceptably poor. Continuous models of impact processes are therefore only appropriate for weak collisions as they might appear in car accidents.

To overcome these problems, impacts should be modeled as they appear to us: as single, discrete events. In such a discrete event model, the velocity is changed at a single point of time. This leads to a fast and precise computation. The knowledge of the underlying continuous process still remains helpful to understand the laws of impacts.

The resulting models are now of a hybrid nature. Whereas the usual behavior is still modeled by continuous processes, the impact behavior is modeled by discrete events. The simulation program has to be able to handle events, solve algebraic equation systems at an event and reset some of the state variables. Fortunately, Modelica and Dymola support all these necessary tasks.

8.2 Theory on impacts

Figure 8.1(b) shows a possible curve of the reaction force between two colliding bodies during the time of impact. The resulting change of velocity is then proportional to the integral of this curve, and this is the only relevant characteristic of this curve to us. So the force impulse is then defined to be:

$$\mathbf{F} = \int \mathbf{f}dt$$

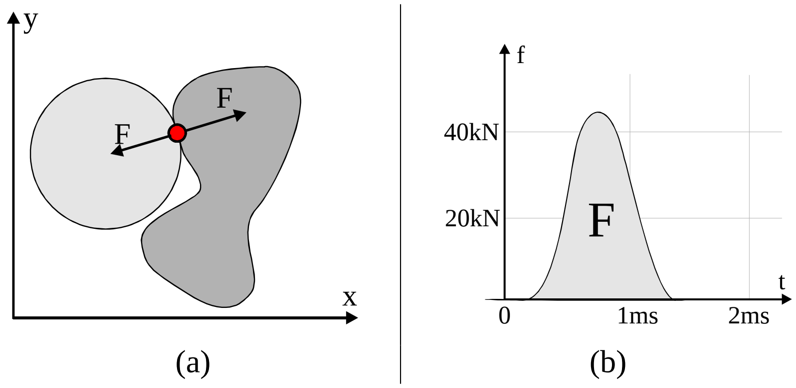


Figure 8.1: (a): Two colliding, planar objects. (b): A hypothetical curve for the reaction force during an impact.

The collision duration of impacts modeled by discrete events is stated to be zero. The interacting force is then infinitely high and the change of the velocity results out of the force impulse acting on a body.

$$\Delta\mathbf{v} \cdot m = \mathbf{F}$$

If all forces of a closed system cancel each other out, the impulses do as well:

$$\sum \mathbf{F} = \mathbf{0}$$

This leads to the law of the conservation of momentum:

$$\sum \mathbf{v} \cdot m = \mathbf{const}$$

8.2.1 Geometry

An impact is decisively influenced by the shapes of the two colliding objects. The colliding objects are supposed to have a well defined shape that is twice derivable, so that the vector perpendicular to the plane is continuously defined at all points of the object's surface. We then suppose that the two objects contact each other at exactly one point. The two normal vectors are then aligned at this point of collision and define the contact direction. The force impulse is acting along this contact direction. For certain shapes the computation of the contact points and the contact direction is very easy. If two spheres collide, the contact point is on the line between the two centers and the contact direction is parallel to this line.

In the following subsections various reflection laws will be discussed. These laws do refer to the relative velocity between the two shapes at the point of contact in direction of the contact direction.

8.2.2 Elastic impacts

The total amount of kinetic energy is conserved if two objects collide with a fully elastic impact. The conservation laws for kinetic energy and momentum lead then to a reversal of the velocity difference between the objects.

$$v_{1,post} - v_{2,post} = -(v_{1,pre} - v_{2,pre})$$

where $v_{1,pre}$ and $v_{2,pre}$ denote the velocity components in contact direction right before the impact. The corresponding velocities right after the impact are denoted by $v_{1,post}$ and $v_{2,post}$.

8.2.3 Non-elastic impacts

If the impact between two bodies leads to some sort of adhesion (i.e. there remains no difference in velocity after the impact) it is called by definition a fully non-elastic impact. Some amount of kinetic energy is dissipated.

$$v_{1,post} - v_{2,post} = 0$$

8.2.4 Mixed cases: the elasticity coefficient

Most of the impacts we observe, are neither fully elastic impacts nor fully non-elastic impacts, but somewhere in between. A possible continuous model of such collisions would be a stiff, open spring-damper element. Hence, the well-known equations for a linear spring-damper oscillation can be applied.

During the contact, the spring-damper element passes through half an oscillation period the duration of which is defined to be $T = \pi\sqrt(M/c)$, where c is the spring constant and M results of the two body masses. This period is independent of the entrance velocity. The oscillation is reduced in its strength by an exponentially decaying function e^{-Ct}, where C is again independent of the entrance velocity. Therefore the exit velocity at time T is smaller than the entrance velocity by roughly the factor $e^{-C\pi\sqrt{M/c}}$. This factor is denoted as elasticity coefficient and the resulting reflection law is identical with Newton's proposal:

$$v_{1,post} - v_{2,post} = -\epsilon \cdot (v_{1,pre} - v_{2,pre})$$

The elasticity coefficient ϵ is usually between 0 and 1. If the objects are penetrating each other, the coefficient is negative. Its absolute value never exceeds one.

8.2.5 Impacts with friction

So far we discussed only force impulses that act along the contact direction, but there is another type of force impulse that goes along with a collision. This is the friction impulse. A heavily rotating cue ball on a billiard table transmits some of its rotation to the object ball. This effect is known to pool players as gearwheel effect. The transmission of the rotation happens due to friction.

The strength of friction impulse is proportional to the strength of impact impulse. This results out of an integration of coulombs law for the friction force:

$$F_R = \int f_R \, dt = \int \mu_R \cdot f_N \, dt = \mu_R \cdot F_N$$

The friction impulse is pointing along the negative velocity difference on the contact plane (or perpendicular to the contact direction). A friction impulse reduces or eliminates the lateral velocity difference, but it can't reverse it. Therefore the friction law above is only partially valid, and F_R is limited so that the friction force impulse causes maximally a non-elastic impulse in the lateral domain. An alternative statement for this restriction is: The resulting elasticity coefficient of a friction impulse has always to be equal or smaller than zero.

8.2.6 Impacts in the rotational domain

Non-central impacts or friction impulses affect also the rotational domain. However, this is not causing any further problems.

The impulse of torque can be defined in correspondence to the force impulse:

$$\mathbf{T} = \int \mathbf{t}dt$$

An impulse of torque is causing a sudden change of the angular velocity:

$$\mathbf{J} \cdot \Delta\boldsymbol{\omega} = \mathbf{T}$$

And the conservation law of angular momentum is:

$$\sum \mathbf{J}\boldsymbol{\omega} = \mathbf{const}$$

Again, all rotational variables are resolved in the body system.

8.3 Hybrid modeling: an example

Modelica and Dymola support the modeling and simulation of hybrid models. These models contain continuous equations and discrete event system. The fundamental methods of hybrid modeling in Modelica are explained in [25]. The technique of hybrid modeling is demonstrated here at a simple example of three colliding objects.

The experiment consists of three objects placed on a line. So the whole geometry is one dimensional. The three objects represent three dimensionless points of mass. There is an invisible spring attached to each object that pulls the objects near to the center. The spring force is proportional to the distance from the center.

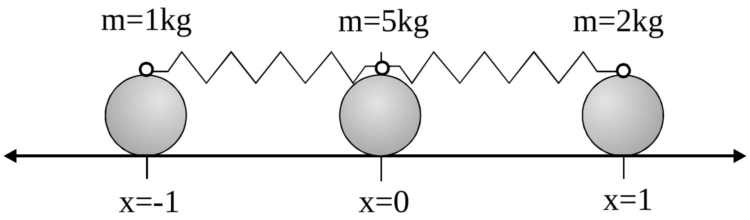


Figure 8.2: Experiment setup.

These objects collide if the positional difference becomes zero. These collisions are defined to be fully elastic and let the object move apart. But the spring forces pull the objects back to the center and cause further collisions. Figure 8.2 sketches the experiment setup and the initial placement of the three objects. The experiment is packed into a single model that contains all variables and equations:

```
model ImpulseTest

 import SI = Modelica.SIunits;

 //masses of the three objects
 parameter SI.Mass m1 = 1;
 parameter SI.Mass m2 = 5;
 parameter SI.Mass m3 = 2;

//start values
 parameter SI.Velocity v1_start = 0;
 parameter SI.Velocity v2_start = 0;
 parameter SI.Velocity v3_start = 0;
 parameter SI.Position x1_start = -1;
 parameter SI.Position x2_start = 0;
 parameter SI.Position x3_start = 1;

 //spring constant
 parameter SI.Acceleration c = 0.3 "spring constant";
```

Continuous variables

```
//position of objects 1-3
SI.Position x1;
SI.Position x2;
SI.Position x3;

//velocities of objects 1-3
SI.Velocity v1;
SI.Velocity v2;
SI.Velocity v3;
```

Discrete variables

```
//velocities of objects 1-3 right after an impact
SI.Velocity v1_p;
SI.Velocity v2_p;
SI.Velocity v3_p;

//velocities of objects 1-3 right before an impact
SI.Velocity v1_a;
SI.Velocity v2_a;
SI.Velocity v3_a;

//overall force impulse on object 1-3
SI.Impulse F1;
SI.Impulse F2;
SI.Impulse F3;

SI.Velocity deltaV12; //velocity difference between object 1 and 2
SI.Velocity deltaV23; //velocity difference between object 2 and 3

SI.Impulse F12; //force impulse from object 1 on object 2
SI.Impulse F23; //force impulse from object 2 on object 3

Boolean contact12; //is true when objects 1 and 2 collide
Boolean contact23; //is true when objects 2 and 3 collide
```

Check for collision and trigger the events

```
algorithm
  when (x1 >= x2) then
      contact12 := true;
   elsewhen (x1 < x2) then
      contact12 := false;
  end when;

  when (x2 >= x3) then
      contact23 := true;
   elsewhen (x2 < x3) then
      contact23 := false;
  end when;
```

State the initial conditions for the three objects

```
initial equation
x1 = x1_start;
x2 = x2_start;
x3 = x3_start;
v1 = v1_start;
v2 = v2_start;
v3 = v3_start;
contact12 = false;
contact23 = false;

equation
```

These are the equations of the continuous system

```
der(x1) = v1;
  der(x2) = v2;
  der(x3) = v3;
  der(v1)*m1 = -c*x1;  //continuous change of momentum of object 1
  der(v2)*m2 = -c*x2;  //continuous change of momentum of object 2
  der(v3)*m3 = -c*x3;  //continuous change of momentum of object 3
```

These equations implement the law of conservation of momentum

```
F1 = F12;
  F2 = -F12+F23;
  F3 = -F23;
  m1*(v1_p-v1_a) = F1;  //discrete change of momentum of object 1
  m2*(v2_p-v2_a) = F2;  //discrete change of momentum of object 2
  m3*(v3_p-v3_a) = F3;  //discrete change of momentum of object 3
```

In case of an impact compute the velocities right before the impact and reinitialize the continuous velocity variables with the velocities right after the impact

```
when contact12 then
      v1_a = v1;  //compute velocities before impulse
      reinit(v1, v1_p);  //reinit velocities
  end when;

  when {contact12,  contact23} then
      v2_a = v2; //compute velocities before impulse
      reinit(v2, v2_p);  //reinit velocities
  end when;

  when contact23 then
      v3_a = v3; //compute velocities before impulse
      reinit(v3, v3_p);  //reinit velocities
   end when;
```

These equations model the elastic impact between object 1 and 2

```
when contact12 then
      deltaV12 = -(v1_a-v2_a);  //state impulse equation
  end when;
  if contact12 then
     v1_p = deltaV12 + v2_p;
  else
     F12 = 0;
  end if;
```

These equations model the elastic impact between object 2 and 3

```
when contact23 then
      deltaV23 = -(v2_a-v3_a);  //state impulse equation
  end when;
  if contact23 then
    v2_p = deltaV23 + v3_p;
  else
    F23 = 0;
  end if;

end ImpulseTest;
```

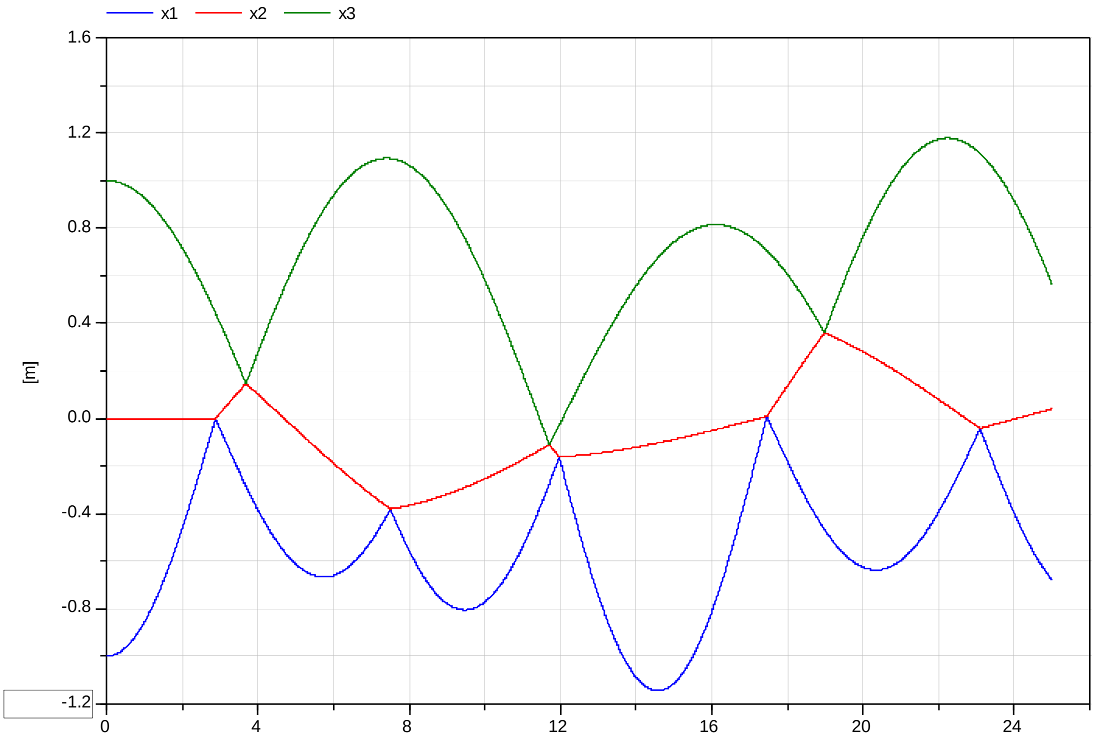


Figure 8.3: Positions of the three objects for a timespan of 25 seconds.

Figure 8.3 shows the result of the simulation. Whereas the continuous part of the model is pretty intuitive and easy to understand,the discrete event modeling needs further explanation. The set of discrete variables form a system of algebraic equations. All equations in the when clauses have the character of an assignment and are supposed to be known. Therefore the variables v1_a, v2_a, v3_a, deltaV12 and deltaV23 are known.

Suppose objects 1 and 2 collide. The boolean variable contact12 then becomes true and triggers further events. These events compute the velocities right before the impact by the usage of the Modelica operator pre(). Also the rule for an elastic impulse is applied by the computation of deltaV12: The velocity difference right after the impact is the negative difference of velocity right before the impact.

The resulting set of equations then contains 8 equations and 8 unknowns:

```
F1 = F12;
F2 = -F12+F23;
F3 = -F23;
m1*(v1_p-v1_a) = F1;
m2*(v2_p-v2_a) = F2;
m2*(v3_p-v3_a) = F3;
v1_p = deltaV12 + v2_p;
F23 = 0;
```

It can easily be reduced to a set of 3 equations and 3 unknowns.

```
m1*(v1_p-v1_a) = F12
m2*(v2_p-v2_a) = -F12;
v1_p = deltaV12 + v2_p;
```

Once the solution for v1_p and v2_p is found, these variables are used to reinitialize the corresponding continuous velocities and the continuous part of the simulation can proceed.

8.4 Object-oriented formulation of the laws

The reflection laws of impacts care only about the behavior at the contact point and lead to certain algebraic equations relating the velocity difference of the colliding objects before and after the impact. To compute the resulting velocities, these equations are not sufficient. The conservation law of momentum has to be taken into account.

However impacts are not restricted to happen between single bodies. They may occur also between (or within) systems of rigidly connected bodies. The force impulse is then directly transmitted from one element to another. Impulses may also act on closed kinematic loops (e. g: a car suspension) and the change of momentum then affects all elements in the loop. Therefore the computation of the resulting velocities can't be done at a unique point. All rigidly connected elements have to be taken into account. This generates the need for an object-oriented solution.

The continuous mechanical systems presented in chapter 6 and chapter 7 were composed out of basic, ideal mechanical elements. These libraries contain models for joints, bodies, fixation elements and translation elements. All these basic elements are represented bondgraphic models.

The laws of force impulse could all be derived out of the laws of continuous mechanics. Therefore it is an evident approach, to inherit the structure and elements of the continuous mechanic libraries and derive the necessary extensions for impulses out of the existing continuous models. But how can the behavior of force impulses be drawn out of the continuous bondgraphic models?

8.4.1 Impulse bonds

Obviously standard bond graphs are not able to model discrete events. A bond graph describes the continuous flow of power, but a force impulse goes along with sudden exchange of energy. Still, this transmissions of energy underlies the rules of thermodynamics and describes a physical process. To handle these energy transmission, a new variant of bond graphs is introduced: The impulse bond graph.

An impulse bond is represented by a double-headed harpoon as in figure 8.4. It carries an effort and a flow variable, but the product of them represents not power anymore, it represents an amount of work. This work is then transferred at a single point of time

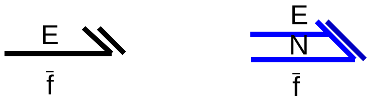


Figure 8.4: The single impulse bond and its multibondgraphic representative. These elements are not part of the standard bond notation.

To find meaningful effort and flow variables, the work is decomposed into a product of variables closely related to the classic effort and flow variables:

$$W = \lim_{a \to 0} \int_0^a p\ dt = \lim_{a \to 0} \int_0^a e \cdot f\ dt \tag{8.1}$$

The effort is supposed to be constant during the integration time. e can therefore be expressed by its integral value $E = \int_0^T e\ dt \Rightarrow e = E/T$:

$$W = \lim_{a \to 0} (E/a) \int_0^a f\ dt \tag{8.2}$$

The flow is supposed to be linearly changing from a starting value f_{pre} to an ending value f_{post}.

$$W = \lim_{a \to 0} (E/a) \int_0^a f_{pre} + \frac{f_{post} - f_{pre}}{a}\ dt \tag{8.3}$$

$$W = \lim_{a \to 0} (E/a) \cdot a \cdot \left(f_{pre} + \frac{1}{2}(f_{post} - f_{pre}) \right) \tag{8.4}$$

$$W = E \cdot \bar{f} \tag{8.5}$$

where $\bar{f}$ is denoting average value of f: $\bar{f} = \frac{f_{pre} + f_{post}}{2}$.

A mechanical impulse bond then carries the force impulse $F = \int f\ dt$ as effort variable and the average velocity $\bar{v} = \frac{v_{pre} + v_{post}}{2}$ as flow variable.

The advantage of impulse bond graphs is that they can be directly drawn out of a classic continuous bond graph. Every bond can be replaced by an impulse bond and for every vertex element there is an impulse-bondgraphic representative. Certain vertex elements can also be neglected and the resulting impulse bond graph can be simplified. The example of the well known planar pendulum will show this in more detail:

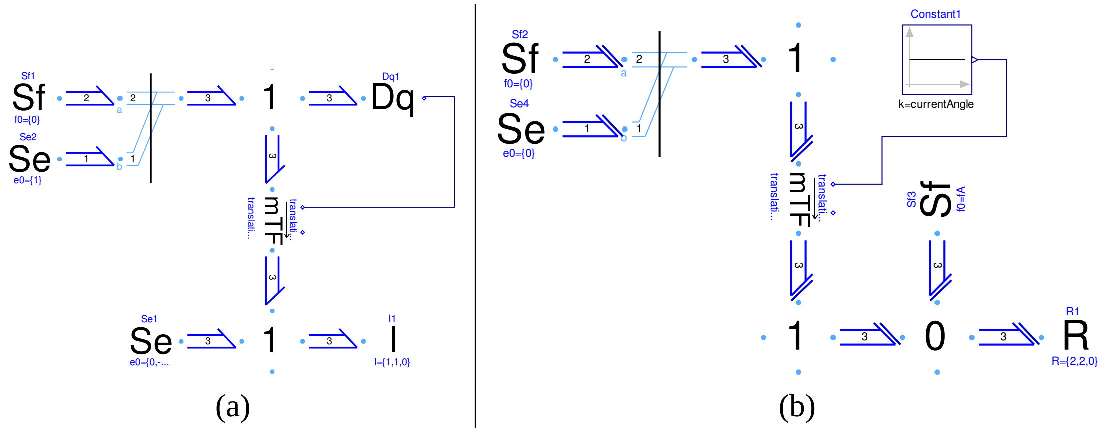


Figure 8.5: The bond graph of a planar pendulum (a) and its corresponding impulse bond graph (b).

Figure 8.5 shows: there is an impulse bond graph "hidden" in the classic bond graph. To reveal this underlying impulse bond graph the following replacement rules have to be applied.

- Capacitive and resistive elements can be neglected or replaced by a source of zero effort.[1]
- Sources of effort can also be neglected or replaced by sources of zero effort.
- A source of flow remains unchanged.
- Transformers and gyrators also remain unchanged.
- Also all junctions remain.
- All modulating signals must be replaced by a constant signal for the time of the impulse. If the value of a signal isn't constant during the impulse, there is no valid impulse bond graph.
- All sensors are removed.
- Inductances or inductive fields are replaced by a corresponding resistance or resistive field of double value. The resistance has a horizontal (velocity) offset v_{pre}, where v_{pre} denotes the corresponding velocity right before the impulse. The offset can be modeled by a 0-junction and a source of flow v_{pre}.

The resulting impulse bond graph is describing a linear system of equations and contains no differential equations. All non-linearities of the original continuous bond graph disappear: Non-linear dampers or effort sources do not matter anymore, and the influence of the modulating

[1] also the multiport gyrator MGY can be neglected

signals is constant during the impulse. The linearity of the resulting model is very important because the remaining impulse bond must be linear, otherwise it is invalid. The assumptions of the derivation in equation 8.2 and 8.3 require linearity.

Fortunately, the underlying impulse bond graph of all mechanical models is linear. This is because the inertia is always linear and all potentially non-linear elements like dampers, springs and other force elements (gravitational force) disappear. The remaining non-linear transformers change into linear ones, because the modulation by the bondgraphic position is constant during the time of the impact.

So impulse bond graphs are a method for deriving the equations of impulse behavior out of a continuous bondgraphic model.

8.5 Extension of the 3D-Mechanics library

A third library for mechanics has been developed. It is called “Mechanics3DwithImpulses” and is an extension of the already known library presented in the previous chapter. The existing continuous 3d mechanical models are extended by their corresponding impulse equations and can be used in models that contain hard collisions. A structural overview of this library is presented in appendix F.

8.5.1 Redeclaration of the interfaces

To afford an extension of the continuous models, it is necessary to redeclare the existing connectors. The new connectors are an extended variant of the old ones and contain 5 additional variables:

Additional potential variables:

- average velocity: $\bar{\mathbf{v}}$ also denoted as $\mathbf{V}m$
- average angular velocity: $\bar{\omega}$ also denoted as $\mathbf{W}m$
- boolean contact signal: contact

Additional flow variables:

- force impulse **F**
- torque impulse **T**

Four of these variables are the connector variables of the corresponding impulse bond graph. The fifth one is the contact signal. This signal is set to true at the time of a force impulse and is transmitted between all rigid connections. Such a signal is necessary, because certain processes like the reinitialization of state variables need to be done at many different points in a system of rigidly connected elements.

8.5.2 Extension of the existing models

Those models with an impulse relevant behavior are extended to hybrid models. The discrete event part could have been modeled by the presented impulse bond graphs, but, although principally possible, this idea was finally discarded. The graphical models of the continuous bond graph obstruct partially further graphical models. For a proper graphical extension, there would be a need for a new drawing layer, but such an overlay technique is not supported by Dymola.

Instead the impulse bond graphs were used to derive the equations for the impulse behavior and these equations were then added to the model in a textual form. The additional equations for two components are of exemplary nature and are presented here:

These are the necessary extensions to the body model:

```
  Real Va[3]; //velocity right before the impulse
  Real Wa[3]; //angular velocity right before the impulse

equation

  //compute velocities right before the impulse
  when frame_a.C.contact then
    Va = frame_a.P.v;
    Wa = frame_a.P.w;
  end when;

  //state the law of the change of momentum
  frame_a.F = m*2*(frame_a.Vm-Va);
  frame_a.T = Inert*2*(frame_a.Wm-Wa);
```

This is the extension of the revolute joint:

```
  Real Rrel[3,3]; //relative orientation between frame a and frame b right before the impulse
  Real Wm; //average relative angular velocity
  Real Wa; //relative angular velocity right before the impulse

equation

  //compute joint velocity and relative orientation right before the impulse
  when frame_a.C.contact then
    Wa = w;
    Rrel = planarRotation1.Rrel;
  end when;

  //reinit potential state variables
  if reinitByImpulse then
    when frame_b.C.contact then
      reinit(w,Wa+2*(Wm-Wa));
    end when;
  end if;

  //propagate contact signal
  frame_a.C.contact = frame_b.C.contact;

  //state equations for the translational domain
  frame_a.F + frame_b.F  = zeros(3);
  frame_a.Vm = frame_b.Vm;

  //state equations for the rotational domain
  Rrel*frame_a.T + frame_b.T  = zeros(3);
  frame_b.T*eN = 0;
  Rrel*frame_a.Wm = -Wm*eN+frame_b.Wm;
```

Please note that the reinitialization can be toggled by the parameter reinitByImpulse. This is a collective property of all joints. All variables that are reinitialized are supposed to be state variables. Not all joints in a kinematic loop define state variables and one has to disable the reinitialization for certain joints. This is the use of this parameter.

All other relevant objects of the continuous Mechanics library were extend in correspondence with this scheme and the resulting library is unsurprisingly very similar to the Mechanics3D library. Again the complex models can be composed out of ideal elements for masses, joints, translations and forces. The only difference is that these basic models are now of a hybrid nature. Still, another class of fundamental components is needed: The collision elements that model the impact between two bodies:

8.5.3 The collision elements

The collision elements model the impact between two rigid elements of different shapes. There are currently two variants: One is modeling the impact of two spheres, the other one is modeling the impact between a sphere and an infinite plane.

The impact characteristics can be specified by the elasticity coefficient ϵ and the friction coefficient μ_R. The impact laws are applied at the time of collision and a contact signals is triggered. The contact signal flip-flops at the moment of collision. It becomes true and is then reset to false at the same moment of time. This signal has then to be connected to the colliding mechanical subparts, which can be done via a special interface element. If a subpart receives several contact signals, the signals have to be merged by a boolean "or" function.

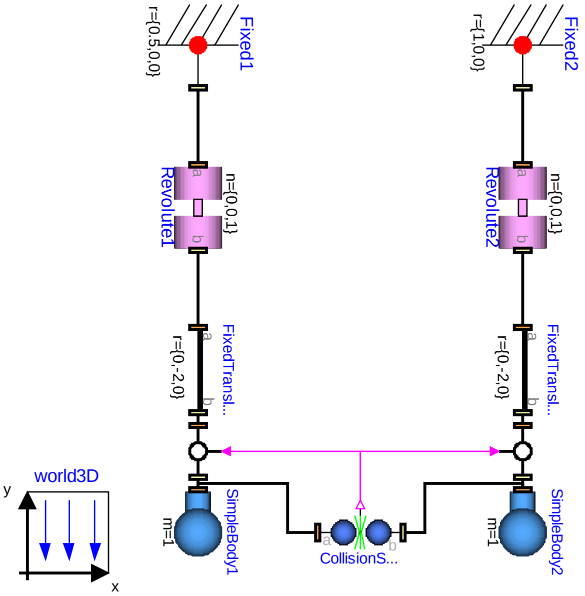


Figure 8.6: Application of the collision element to model the collision of two pendulums.

To model a general impact, three consecutive impulses are needed: The first one is the central impact impulse. The second impulse is the friction impulse. It is proportional to the strength of the central impact impulse. However, the friction impulse is limited and can't cause cause more than an elastic lateral contact. If this limitation is exceeded by the second impulse there's a need for a third impulse: the correction impulse. This impulse counteracts the second impulse and causes a fully non-elastic lateral contact.

These three impulses actually happen at the same moment of time, which makes things even more difficult. The contact signal would have to flip-flop three times within the same moment of time and cause further events in the connected components at each flip-flop. Unfortunately the behavior of events causing further events at the same moment of time is not well specified in Modelica. Usually, the second flip-flop within the same moment of time is not triggering any further events.

To cause three consecutive events at the same moment of time, a more complex contact signal is needed. A simple boolean signal would be insufficient. Anyhow, a change of the contact signal definition would cause further problems (e.g: in the case of several possible impact points). It is principally possible to solve all these problems in Modelica, but the resulting solution would be completely unhandy and very bulky.

The concept of a simple boolean contact signal is therefore maintained. To make this possible, it is necessary to separate the three impulses by a very small amount of time. It can be set by the parameter contactDuration and is usually in the range of a few nano-seconds. It is important to note that this leads to an incorrect results if the impact time of two collisions on the same element differs by an amount that is larger then zero but smaller then contactDuration. Another drawback of this solution is, that the continuous variables are unnecessarily reinitialized several times. Still, for frictionless impacts the presented solution is correct in all cases.

The collision elements are pretty complicated models. The implementation of a collision element between two spherical objects is presented in appendix A.

8.6 Examples

Three prototypical examples are presented in this section.

8.6.1 Newton's cradle

This is the model of Newton's cradle. Five pendulums are placed in a row and the deflection of one of them leads to a sequential series of impacts. The three pendulums in the center undergo two possible collisions. The two corresponding contact signals are therefore merged by an or block.

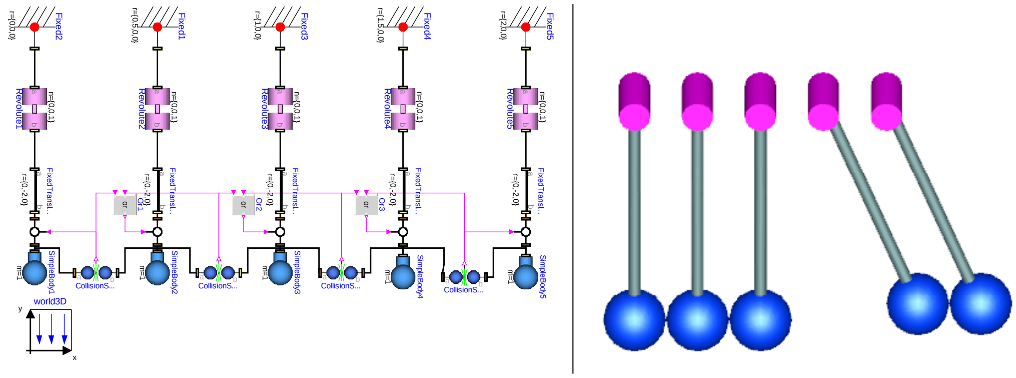


Figure 8.7: An implementation of Newton's cradle.

8.6.2 Impacts on a kinematic loop

This examples demonstrates impacts on a closed kinematic loop. The spherical joint is colliding with the ground plane and is reflected by a fully elastic impact.

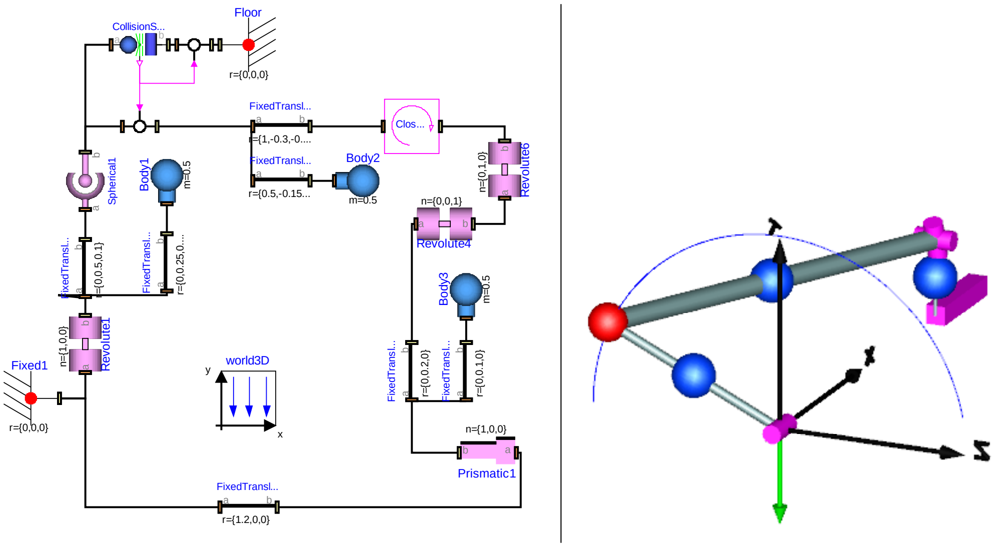


Figure 8.8: The movement of the kinematic loop is restricted to one side of the floor plane.

Impacts on kinematic loops cause no further problems. Only the loop needs to be closed manually by a special CloseLoop elements of the package “Joints”. The CloseLoop element is an extension of the manual loop closer in the continuous mechanics library. It is needed, because a further loop needs to be cut: the one of the boolean contact signal. The built-in mechanism for automatic loop closure are restricted to real variables. Therefore it is not possible to cut a boolean loops automatically[2] and we got to do the closure manually. Also the state selection has to be done manually and only the state holding joint can be reinitialized by an impulse.

8.6.3 Impacts with friction

In this model a heavily rotating ball is falling on the floor. It starts to bounce in a certain direction, because due to friction, the rotation is successively transformed into lateral velocity.

[2] of course one could emulate a boolean signal with a real signal, but this would be again a pretty awkward hack.

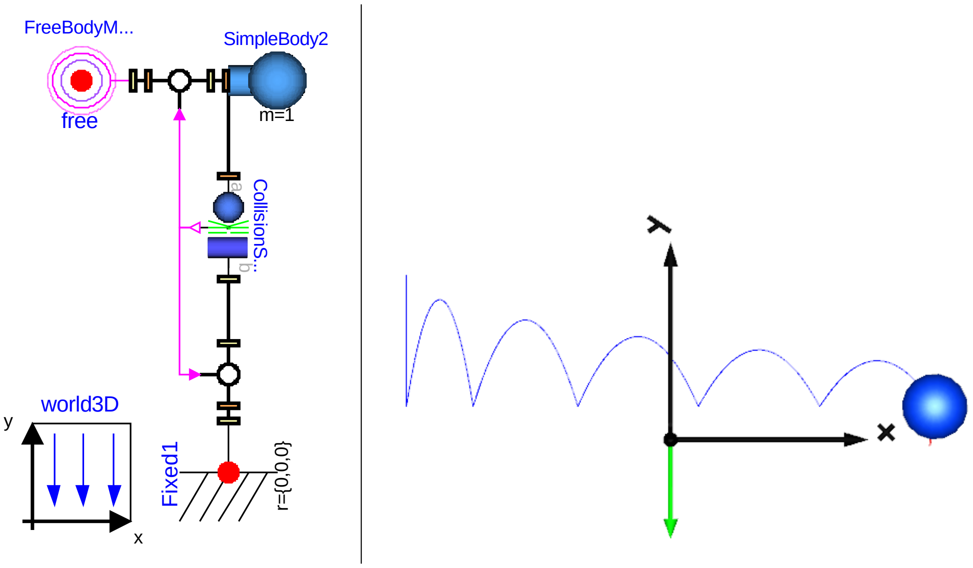


Figure 8.9: Model of a ball bouncing on the floor.

8.7 Evaluation

The continuous models for 3D-Mechanics could successfully be extended to hybrid models that allow a discrete change in velocity. The efficiency of the continuous models is thereby not impaired in any way. The examples of the previous section demonstrate that all kinds of impacts can be accurately modeled.

However, the currently offered support of hybrid models in Modelica and Dymola is not fully sufficient. The usage of the resulting hybrid models is therefore somewhat involved. The state selection has to be done manually in all systems to determine the points of reinitialization. Also the contact signal needs to be handled manually to allow several impacts on the same system of rigidly connected elements. An additional drawback is that kinematic loops need to be closed manually by a special element. Further improvements are needed to get a really user-friendly solution that solves these tasks automatically.

The efficiency of the computation at a discrete event is sufficient but could be improved. Dymola provides a solution so that all possible events could occur simultaneously. This consequently leads to huge systems of linear equations. Although correct, this solution is an overkill, because simultaneous impacts could be handled sequentially in an arbitrary order. Hence, these huge systems of linear equations are unnecessary.

The size of these linear systems was examined in an experiment. A given number of freely moving bodies (each has 6 degrees of freedom) collide with each other due to mutual attraction.

All possible collisions were modeled. The size of the largest appearing system of linear equations was measured. Table 8.1 presents the results.

number of bodies	number of possible collisions	size of linear system
2	1	12
3	3	21
4	6	30
5	10	45
6	15	66
7	21	83
8	28	110

Table 8.1: Appearance of large systems of linear equations.

The efficiency of the contact model is additionally impaired by the triggering of three consecutive events, which are separated by a very small amount of time. A solution where the corresponding three force impulses act simultaneously is highly desirable and would be more efficient.

One may finally conclude that the provided solution works fine for small scale models. The modeling of larger systems becomes laborious and the efficiency of the simulation becomes critical. There is definitely a lot of room for potential improvements, but some of them require an extension of Modelica's and Dymola's abilities.

Part III

Conclusions

Chapter 9

Discussion

A Modelica library for convenient, multibondgraphic modeling has been developed. It provides a general solution to model all kinds of multidimensional processes in continuous physical systems. Multibond graphs offer a very good framework for modeling mechanical systems. The bondgraphic methodology proved to be powerful enough for modeling all important ideal subparts of mechanical systems accurately and efficiently.

The resulting libraries for mechanical systems PlanarMechanics and Mechanics3D provide a comprehensive set of models. These domain specific models have an intuitive appeal and are easy to use. They consist in wrapped bondgraphic models, and a closer look reveals a bondgraphic explanation of their behavior. Also quality and efficiency of the resulting solution are not impaired by the bondgraphic modeling technique. The selected state variables are highly meaningful, and the resulting systems of equations can be solved fast and accurately.

The continuous mechanical models were extended to hybrid models that allow a discrete change in motion. These hybrid models form a new library: Mechanics3DwithImpulses. It was designed to model the behavior of mechanical system in situations of hard impacts. Sadly, the Modelica support for discrete event modeling is not fully sufficient. The resulting solution leads partially to models that are unnecessarily complicated and is only applicable to small scale problems.

9.1 Additional work

Not all achievements could be covered by this document. Further tasks have been approached. The most important of them are briefly presented here.

An alternative approach to the bondgraphic modeling of 3D-Mechanics has been taken. This time the translational and rotational bonds were all covered in a multibond of cardinality 6. In consequence, all bondgraphic variables are resolved in their body system. Also this approach leads to a feasible solution, but it turns out that separate multibonds for the translational and rotational domains generate slightly better equations and are more powerful for explaining small scale models. 6-multibonds are better suited for large scale modeling, but due to the wrapping technology there is hardly any demand for large multibond graphs.

A gravity pool has been created to model the mutual gravitational attraction between body elements. The gravity pool is an outer model and was included with the standard world model. Each mass element can be entered into the gravity pool by the assignment of a unique index number.

A similar task was the modeling of mutual collisions. Again a pool model has been designed: The collision pool. It is not included in the world model and forms a separate element. The geometry of the colliding objects is currently restricted to be spherical.

9.2 Outlook on future tasks

There still remains a vast number of further modeling tasks. A few of them should be mentioned here.

The modeling of friction was hardly looked at. Especially the transition between friction and adhesion is an important issue. Also the transition between slippage and ideal rolling is a very related problem. Another topic of interest concerns joints that are constrained in their range of movement. E.g. the revolute joint in a robot arm may work only in a range of 180°. Another example are the stands of a tripod, which can be elongated but not be retracted.

Often, force impulses go along with such processes. Also holonomic or non-holonomic constraints may be introduced at simulation time, and these constraints may lead to structural changes within the model. The number of degrees of freedom would then be changing during the simulation. Unfortunately, the modeling and especially the simulation of such systems is currently not supported by Dymola in a satisfactorily and usable way.

There is a need to enhance both the modeling language and the simulation program in their abilities. Such enhancements could also lead to a better and more comfortable solution for hybrid models of Mechanics3DwithImpulses.

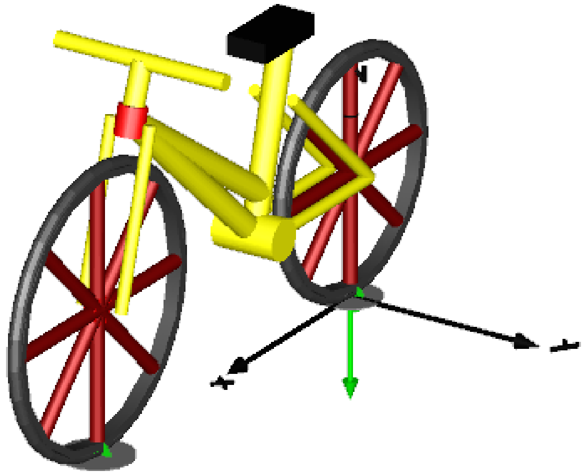
Appendix A

Modelica Code of the Collision Element

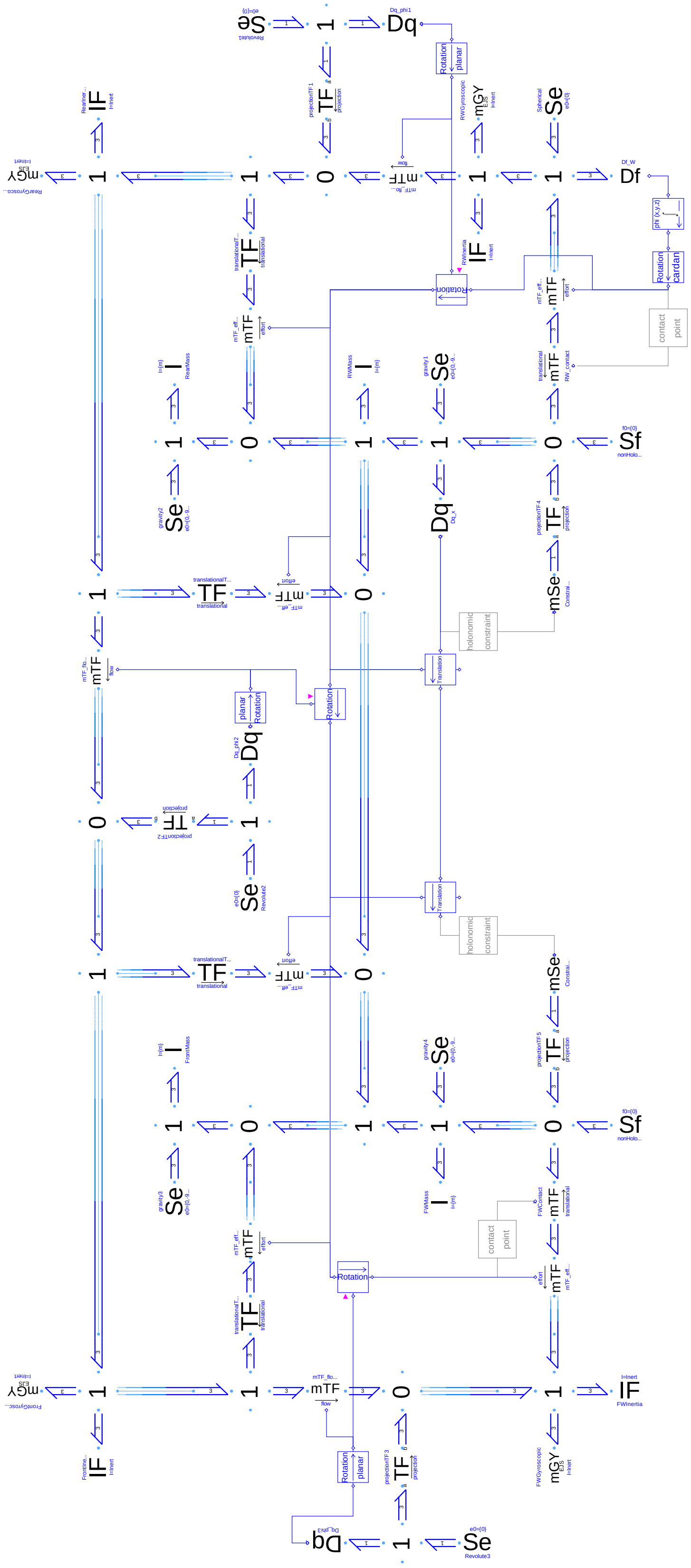
```
model CollisionSphereSphere
  import SI = Modelica.SIunits;

  Interfaces.IFrame_a frame_a; "frame at sphere a"
  Interfaces.IFrame_b frame_b; "frame at sphere b"
  Modelica.Blocks.Interfaces.BooleanOutput y; "boolean contact signal"


  parameter SI.Radius ra = 1 "radius of sphere at frame a";
  parameter SI.Radius rb = 1 "radius of sphere at frame b";

  parameter SI.Position s_small = 1e-4 "critical length";

  parameter Real elasticity = 1.0 "Elasticity of impulse";
  parameter Real muR = 0.0 "friction coefficient";

  parameter Real contactDuration =   1e-9  "time to handle the impulse";

  parameter Boolean useParameters = true "use the parameter values.";

protected
  SI.Distance r[3]; //vector pointing from center of sphere a to center of sphere b
  Real eR[3];       //normalized variant of r
  Real noteR[3];    //a vector that is anything but r
  Real eA[3];       //unit vector that is normal to r
  Real eB[3];       //unit vector that is normal to r and eA.
  //eR, eA, eB build up an orthonormal coordinate system, where eR is pointing in contact direction.

  Boolean contact;  //contact signal
  Integer seqState;  //state in the impluse sequence (0=impact, 1=friction, 2=correction)
  Real frictionImpulseTime;  //time of the succeeding friction Impulse
  Real correctionImpulseTime; //time of the succeeding correction Impulse

  SI.Impulse F[3];   //force impulse between the two frames
  SI.Velocity vA[3]; //rel. velocity of the two frames right before the impulse
  SI.Velocity Vm[3]; //average rel. velocity during the impulse

  //variables of the impact impulse
  SI.Impulse FR;   //force impulse in direction of eR
  SI.Impulse FA0;   //force impulse in direction of eA
  SI.Impulse FB0;   //force impulse in direction of eB
  SI.Velocity VmR0; //average velocity in direction of eR
  //variables of the friction impulse
  SI.Impulse FA1;  //force impulse in direction of eA
  SI.Impulse FB1;  //force impulse in direction of eB
  SI.Impulse FR1;  //force impulse in direction of eR
  SI.Velocity VFricElast[3];

  //eC, eD, eE build up an orthonormal coordinate system where eC points into the direction
  of the lateral velocity difference (slippage)
  Real eC[3];
  Real noteC[3];
  Real eD[3];
  Real eE[3];

  //variables of the correction impulse
  SI.Impulse FD2;   //force impulse in direction of eD
  SI.Impulse FE2;   //force impulse in direction of eE
  SI.Velocity VmC2; //average velocity in direction of eC

  Real[3,3] R_a;  //orientation of frame a at the impulse time
  Real[3,3] R_b;  //orientation of frame b at the impulse time


initial equation
  contact = false;
  seqState = -1;
  frictionImpulseTime = 0;
  correctionImpulseTime = 0;

equation

algorithm
  //reset contact signal to false
  when contact then
    contact :=false;
  end when;

  //detect collision and trigger event
  when sqrt(r*r) <= (ra_+rb_) then
    contact :=true;
    seqState := 0;
   frictionImpulseTime :=time + contactDuration/2;
  end when;

  //trigger event for friction impulse
  when (time >= frictionImpulseTime) and (seqState == 0) then
    contact :=true;
    seqState := 1;
    frictionImpulseTime := 0;
    correctionImpulseTime := time + contactDuration/2;
  end when;

  //trigger event for correction impulse
  when (time >= correctionImpulseTime) and (seqState == 1) then
    contact :=true;
    seqState := 2;
    correctionImpulseTime := 0;
  end when;

equation

//build up coordinate system eR, eA, eB at the moment of collision.
  r = frame_b.P.x- frame_a.P.x;
  when contact and (seqState == 0) then
    eR = r/sqrt(r*r);
    noteR = if abs(eR[1]) > 0.1 then {0,1,0} else (if abs(eR[2])
       > 0.1 then {0,0,1} else {1,0,0});
    eA = cross(eR, noteR)/sqrt(cross(eR, noteR)*cross(eR, noteR));
  eB = cross(eA, eR);
end when;

//state situation right before the impulse
when contact then
  vA = (pre(frame_b.P.v)+cross(transpose(R_b)*pre(frame_b.P.w),-eR*rb_)) -
       (pre(frame_a.P.v)+cross(transpose(R_a)*pre(frame_a.P.w),eR*ra_));
  R_a = frame_a.P.R;
  R_b = frame_b.P.R;
end when;

//store the impact impulse in FR
when contact then
  FR = F*eR;
end when;

//cause impact impulse
when contact and (seqState == 0) then
  VmR0 = (1-elasticity_)*(vA*eR)/2;
  FA0 = 0;
  FB0 = 0;
end when;

//cause friction impulse
when contact and (seqState == 1) then
  FR1 = 0;
  FA1 = -muR_*abs(pre(FR))*((vA/sqrt(vA*vA))*eA);
  FB1 = -muR_*abs(pre(FR))*((vA/sqrt(vA*vA))*eB);
  VFricElast[1] = if sign(vA[1])*Vm[1] >= abs(vA[1]/2) then 0 else Vm[1]- (vA[1]/2);
  VFricElast[2] = if sign(vA[2])*Vm[2] >= abs(vA[2]/2) then 0 else Vm[2]- (vA[2]/2);
  VFricElast[3] = if sign(vA[3])*Vm[3] >= abs(vA[3]/2) then 0 else Vm[3]- (vA[3]/2);
end when;

//cause correction impulse
when contact and (seqState == 2) and (pre(VFricElast)*pre(VFricElast) > 0) then
  eC = if pre(VFricElast)*pre(VFricElast) < s_small^2 then {1,0,0} else
        pre(VFricElast)/sqrt(pre(VFricElast)*pre(VFricElast));
  noteC = if abs(eC[1]) > 0.1 then {0,1,0} else (if abs(eC[2])
     > 0.1 then {0,0,1} else {1,0,0});
  eD = cross(eC, noteC)/sqrt(cross(eC, noteC)*cross(eC, noteC));
  eE = cross(eD, eC);

  VmC2 = sqrt(pre(VFricElast)*pre(VFricElast));
  FD2 = 0;
  FE2 = 0;
end when;

//state the impulse dependent equations
if contact then
  if (seqState == 0) then //further equations for the impact impulse
    Vm*eR = VmR0;
    F*eA = FA0;
    F*eB = FB0;
 else
     if (seqState == 1) then  //further equations fot the friction impulse
       F*eR = FR1;
       F*eA = FA1;
       F*eB = FB1;
   else
      //further equations for the correction impulse (if necessary)
      if (seqState == 2) and (pre(VFricElast)*pre(VFricElast) > 0) then
        Vm*eC = VmC2;
        F*eD = FD2;
        F*eE = FE2;
      else
        F = zeros(3);  //dummy equations
      end if;
     end if;
   end if;
  else
    F = zeros(3);
  end if;

  //propagate contact signal
  y = contact;

  //state equations for the translational domain
  Vm = (frame_b.Vm+cross(transpose(R_b)*frame_b.Wm,-eR*rb_)) -
       (frame_a.Vm+cross(transpose(R_a)*frame_a.Wm,eR*ra_));
  F = -frame_b.F;
  F = frame_a.F;

  //state equations for the translational domain
  R_b*cross(-eR*rb_,F) = -frame_b.T; //actually incorrect but ok because T=0;
  R_a*cross(eR*ra_,F) = frame_a.T; //actually incorrect but ok because T=0;

  //define continuous connector variables
  frame_a.f = zeros(3);
  frame_a.t = zeros(3);
  frame_b.f = zeros(3);
  frame_b.t = zeros(3);

end CollisionSphereSphere;
```

Appendix B

The Unwrapped Model of a Bicycle

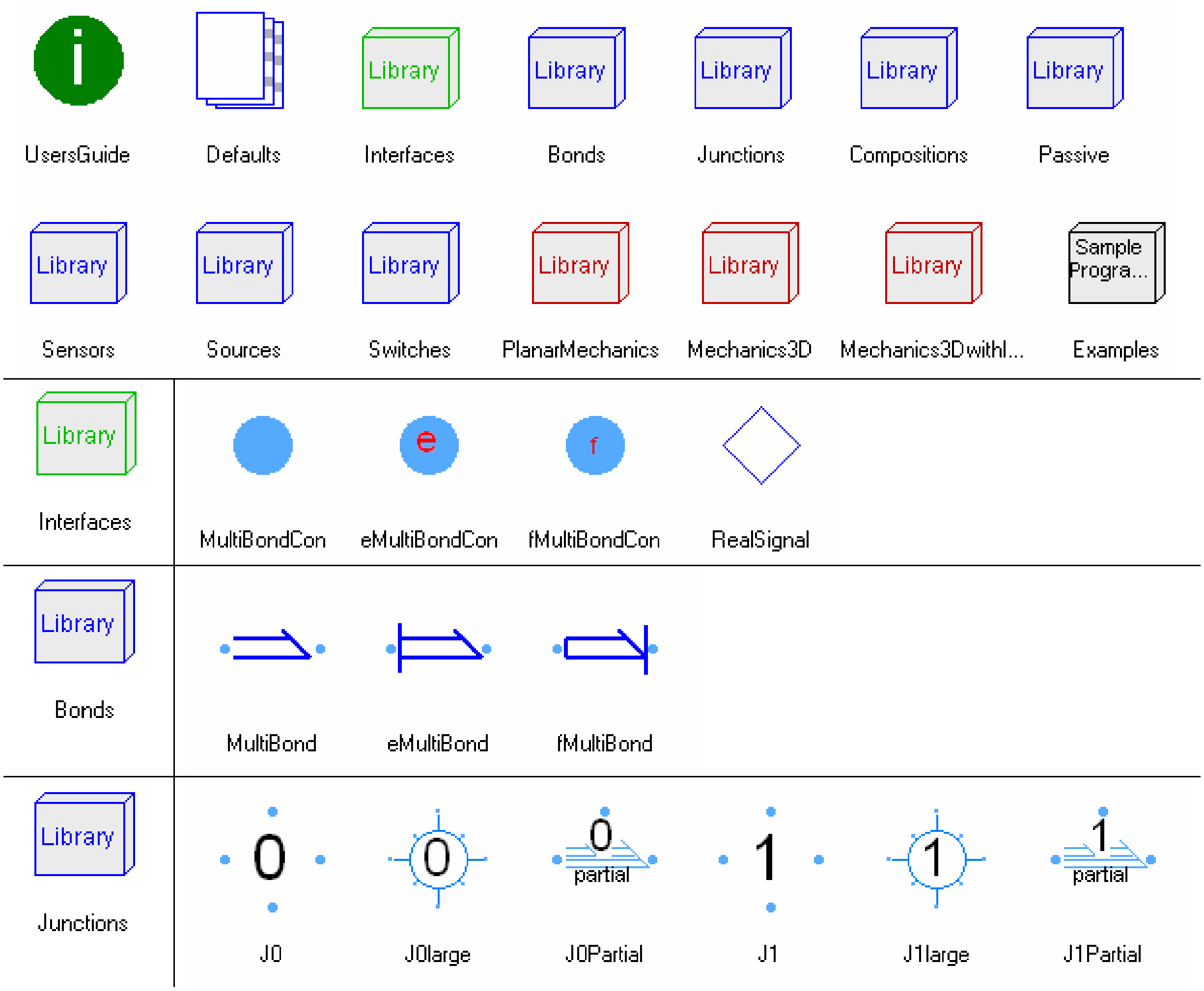


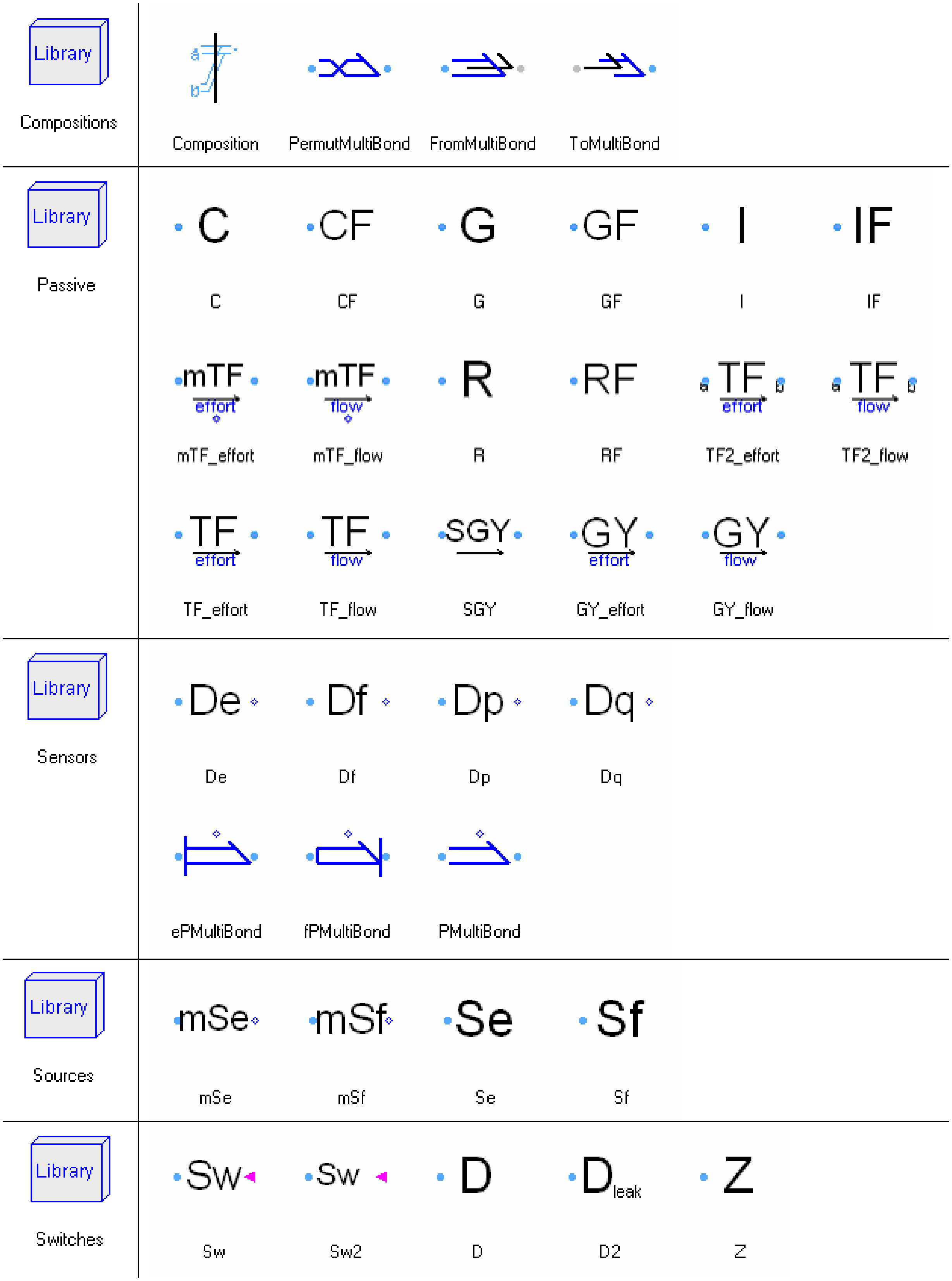
The pure bondgraphic model of bicycle is presented on the next page.


Se
Revolute1
e0={0}
Dq
Dq_phi1
Rotation
planar
TF
projectionTF1
projection
mGY
RWGyroscopic
Inirtert
Se
Spherical
e0={0}
IF
Rearlner...
Inirtert
mGY
RearGyrosco...
Inirtert
mTF
mTF_flo...
flow
Df
Df_W
phi (x,y,z)
Rotation
cardan
IF
RWInertia
Inirtert
Rotation
TF
translationalT...
translational
mTF
mTF_eff...
effort
mTF
mTF_eff...
effort
contact
point
mTF
translational
RW_contact
I
RearMass
I=(m)
I
RWMass
I=(m)
Se
gravity1
e0={0,-9...
Sf
nonHolo...
f0={0}
Se
gravity2
e0={0,-9...
Dq
Dq_x
TF
projectionTF4
projection
TF
translationalT...
translational
mTF
mTF_eff...
effort
mSe
Constrai...
holonomic
constraint
Translation
mTF
mTF_flo...
flow
Rotation
planar
Dq
Dq_phi2
TF
projectionTF2
projection
Se
Revolute2
e0={0}
Translation
holonomic
constraint
TF
translationalT...
translational
mTF
mTF_eff...
effort
mSe
Constrai...
I
FrontMass
I=(m)
Se
gravity4
e0={0,-9...
TF
projectionTF5
projection
Sf
nonHolo...
f0={0}
Se
gravity3
e0={0,-9...
I
FWMass
I=(m)
contact
point
mTF
FWContact
translational
mTF
mTF_eff...
effort
mTF
mTF_eff...
effort
Rotation
TF
translationalT...
translational
mGY
FrontGyrosc...
Inirtert
mTF
mTF_flo...
flow
IF
FWInertia
Inirtert
IF
Frontine...
Inirtert
Rotation
planar
TF
projectionTF3
projection
mGY
FWGyroscopic
Inirtert
Dq
Dq_phi3
Se
Revolute3
e0={0}

Appendix C

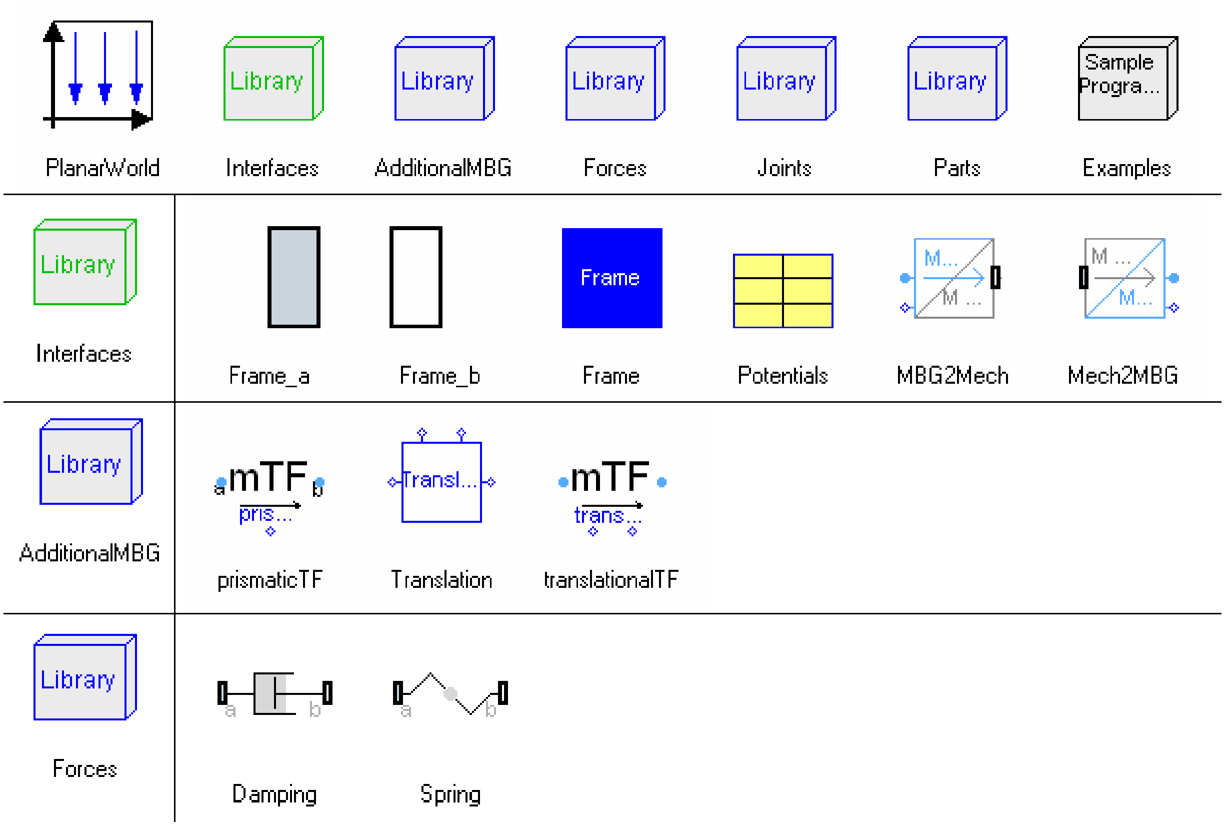
Overview of the MultiBondLib

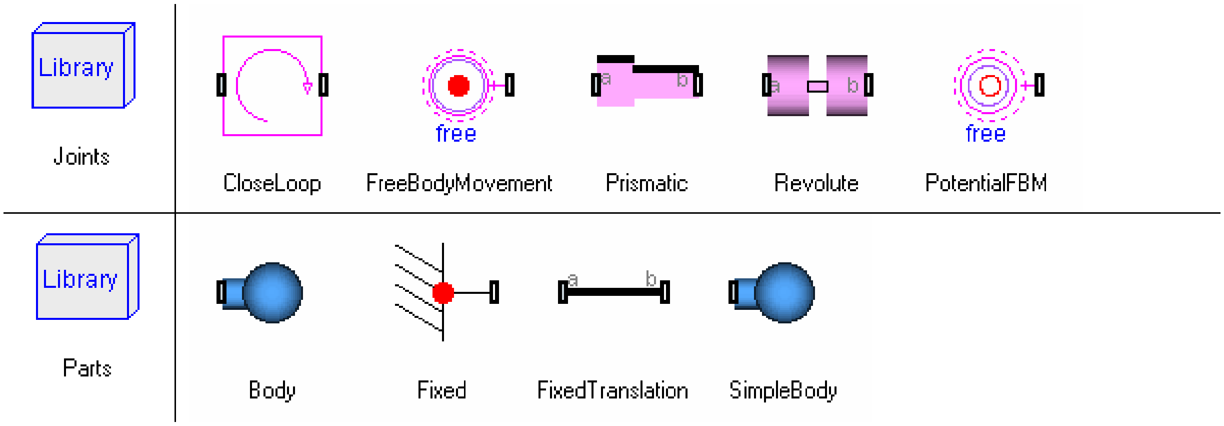



Library
Compositions
a
b
Composition
PermutMultiBond
FromMultiBond
ToMultiBond
Library
Passive
C
C
CF
CF
G
G
GF
GF
I
I
IF
IF
mTF
effort
mTF_effort
mTF
flow
mTF_flow
R
R
RF
RF
a TF b
effort
TF2_effort
a TF b
flow
TF2_flow
TF
effort
TF_effort
TF
flow
TF_flow
SGY
SGY
GY
effort
GY_effort
GY
flow
GY_flow
Library
Sensors
De
De
Df
Df
Dp
Dp
Dq
Dq
ePMultiBond
fPMultiBond
PMultiBond
Library
Sources
mSe
mSe
mSf
mSf
Se
Se
Sf
Sf
Library
Switches
Sw
Sw
Sw
Sw2
D
D
D
leak
D2
Z
Z

Appendix D

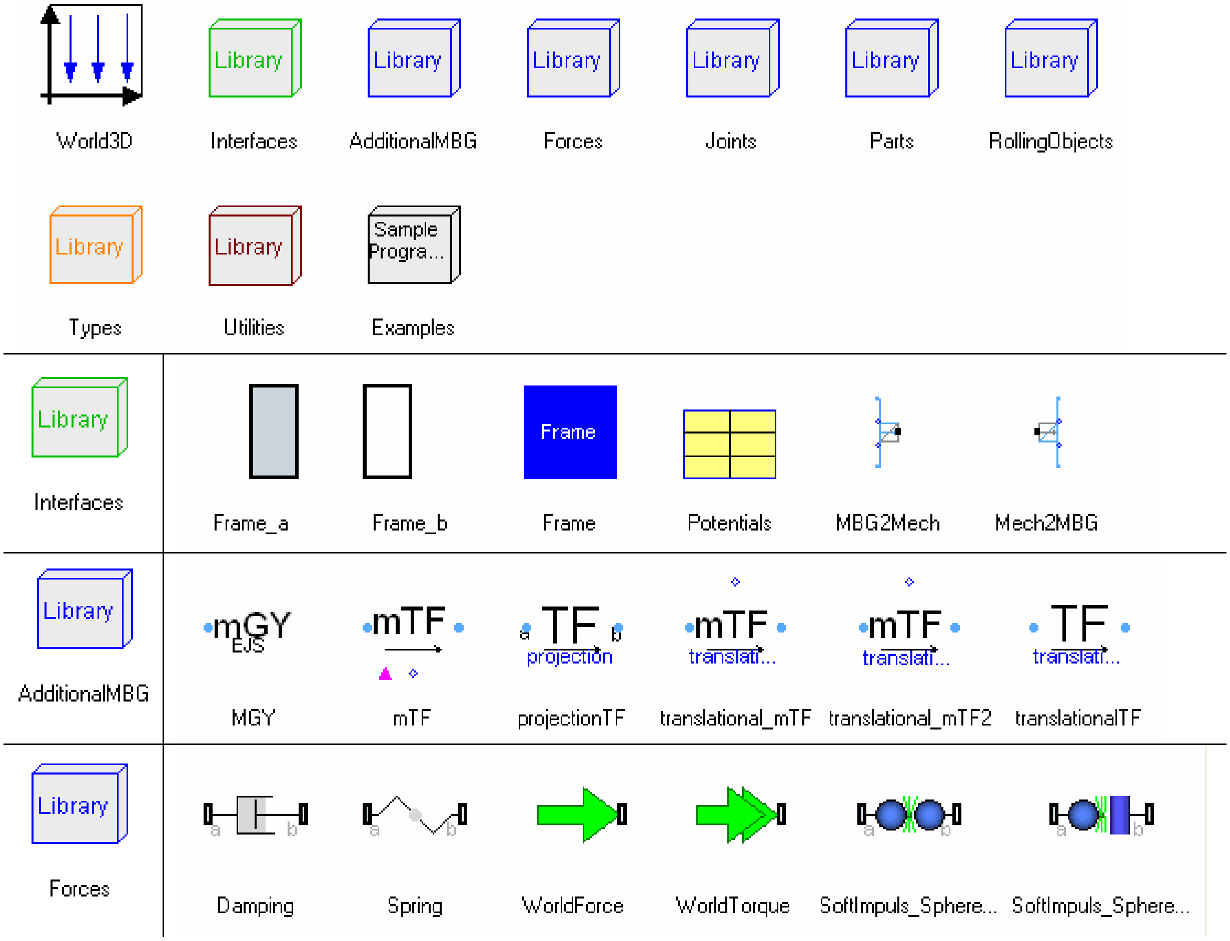
Overview of the PlanarMechanics library

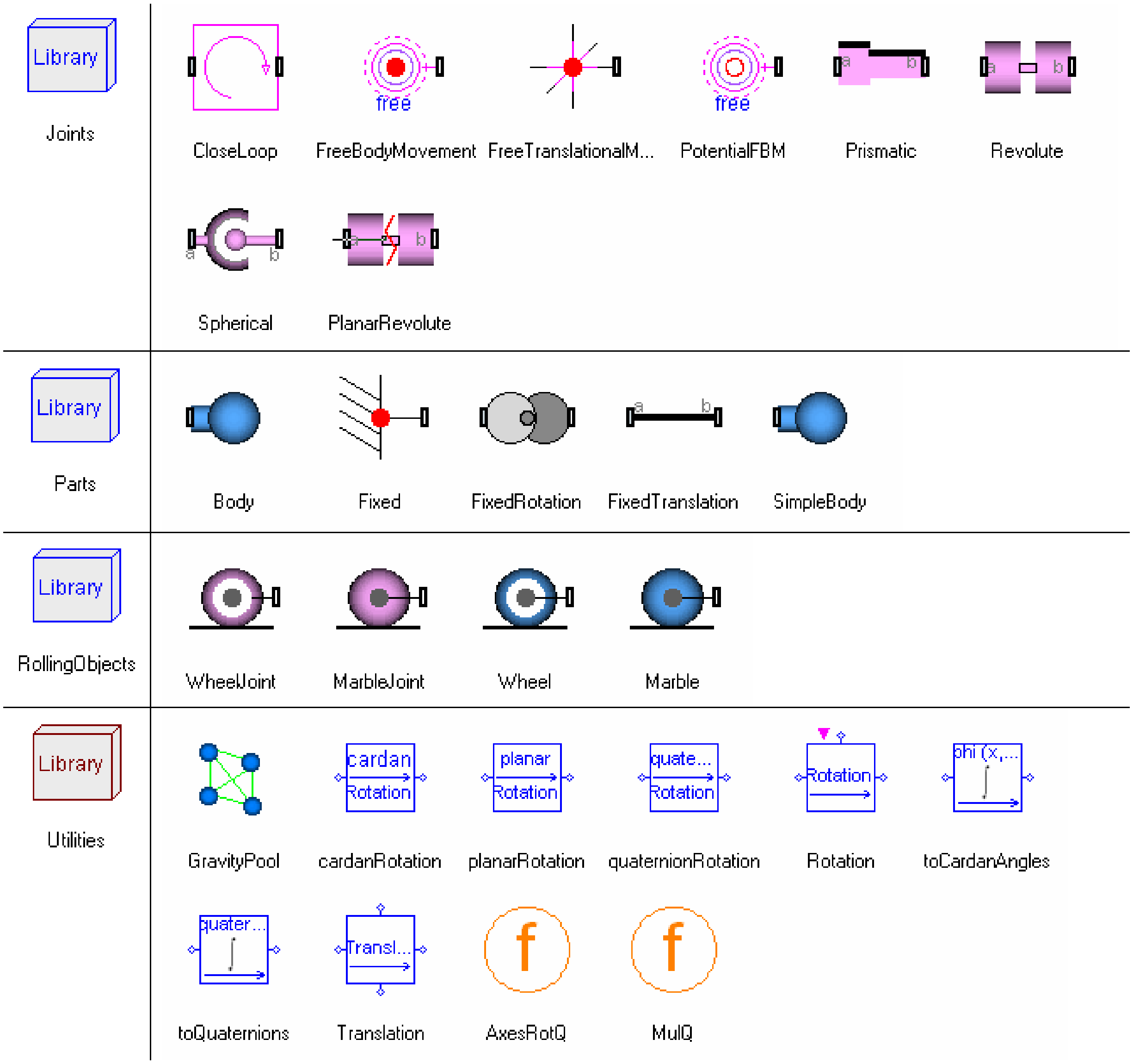



Library
Joints
CloseLoop
free
FreeBodyMovement
a
b
Prismatic
a
b
Revolute
free
PotentialFBM
Library
Parts
Body
Fixed
a
b
FixedTranslation
SimpleBody

Appendix E

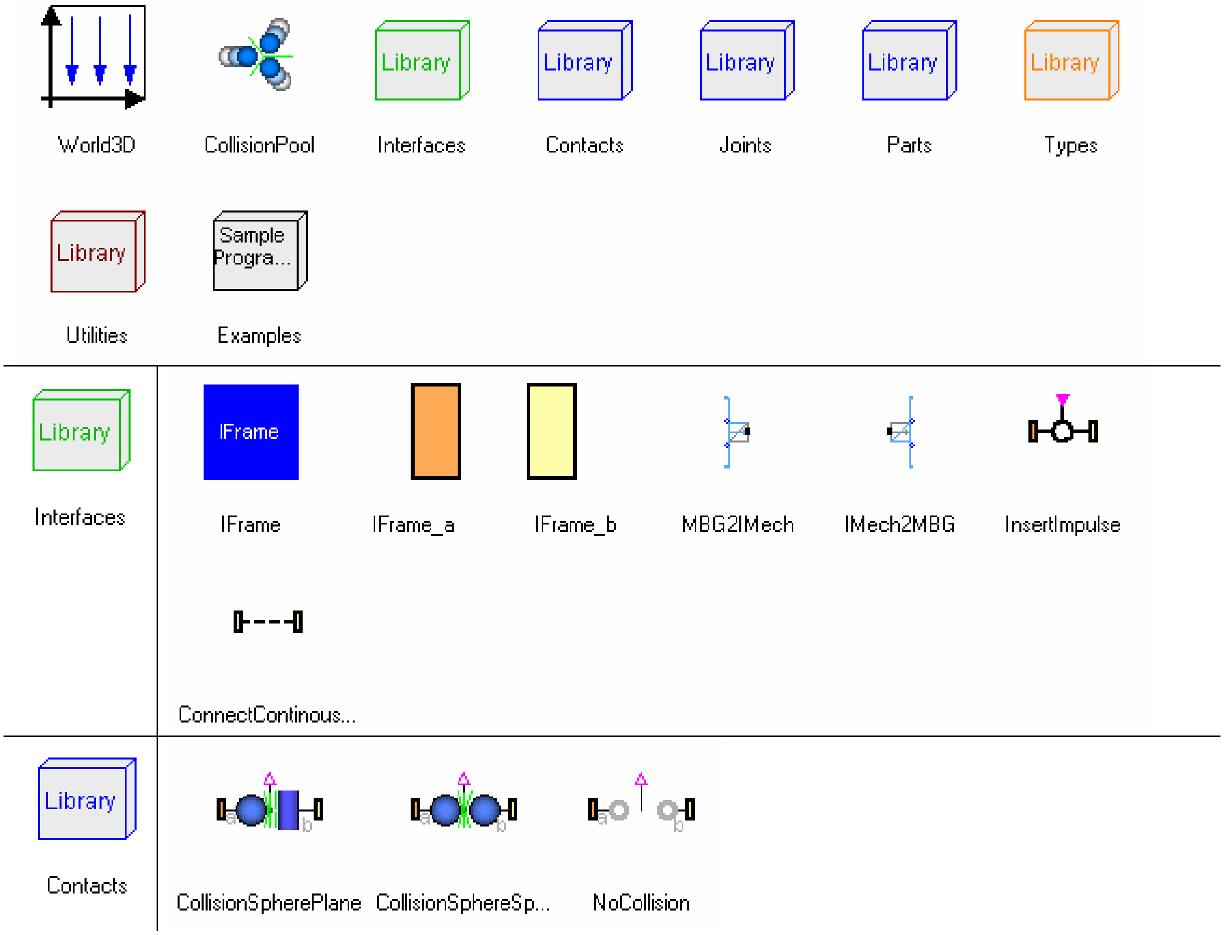
Overview of the Mechanics3D library

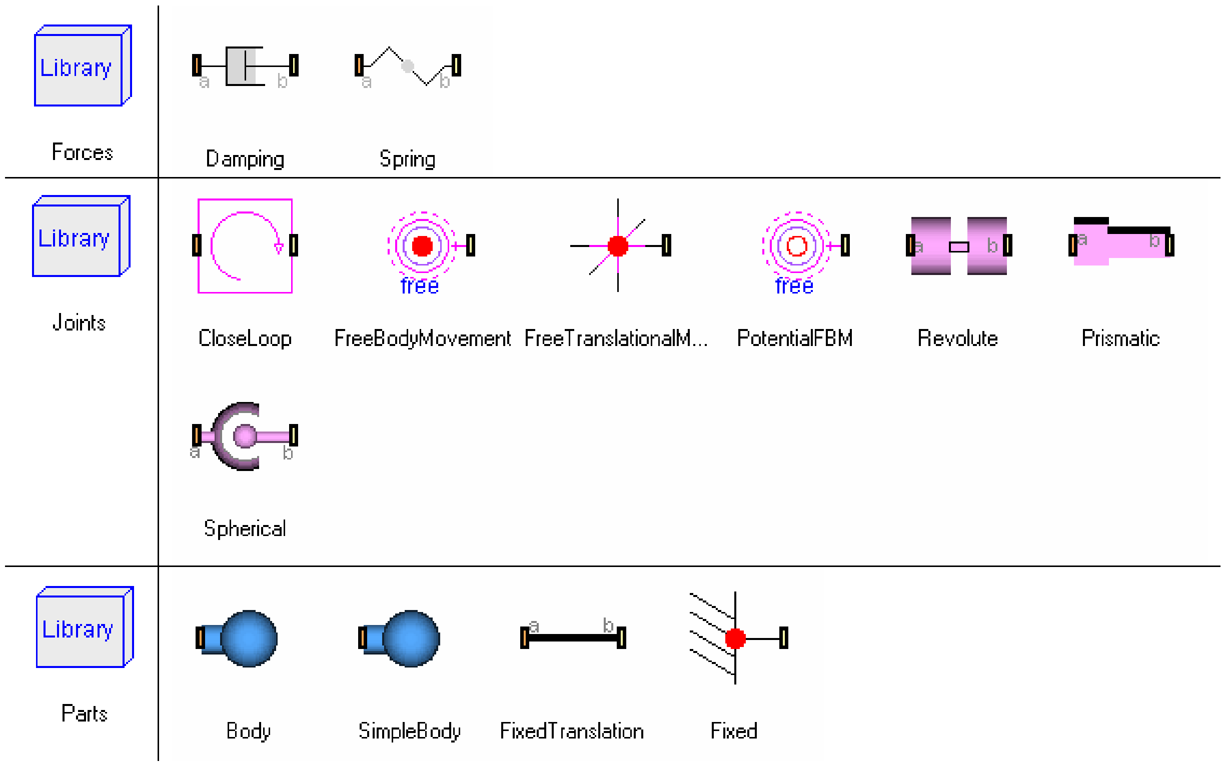



Library
Joints
CloseLoop
free
FreeBodyMovement
FreeTranslationalM...
free
PotentialFBM
a
b
Prismatic
a
b
Revolute
a
b
Spherical
a
b
PlanarRevolute
Library
Parts
Body
Fixed
FixedRotation
a
b
FixedTranslation
SimpleBody
Library
RollingObjects
WheelJoint
MarbleJoint
Wheel
Marble
Library
Utilities
GravityPool
cardan
Rotation
cardanRotation
planar
Rotation
planarRotation
quate...
Rotation
quaternionRotation
Rotation
Rotation
phi (x,...
toCardanAngles
quater...
toQuaternions
Transl...
Translation
f
AxesRotQ
f
MulQ

Appendix F

Overview of the Mechanics3DwithImpulses library




Library
Forces
a
b
Damping
a
b
Spring
Library
Joints
CloseLoop
free
FreeBodyMovement
FreeTranslationalM...
free
PotentialFBM
a
b
Revolute
a
b
Prismatic
a
b
Spherical
Library
Parts
Body
SimpleBody
a
b
FixedTranslation
Fixed

Bibliography

[1] F. Aparicio, C. Vera, J. Félez: **Bond Graph Simulation of Vehicle Collision**. Proc. ICBGM'93 pp. 229-239, Tijuana California 1993.

[2] K. J. Aström, R. E. Klein, A. Lennartsson: **Bicycle Dynamics and Control**. IEEE Control Systems Magazine, August 2005.

[3] A. M. Bos: **Modelling MultiBody Systems in Terms of MultiBond Graphs with application to a motorcycle**. CIP-DATA Koninkljke Bibliotheek Den Haag, 1986.

[4] P. C. Breedveld: **Physical Systems Theory in Terms of Bond Graphs**. CIP-Data Koninkljke Bibliotheek Den Haag, 1984.

[5] F. E. Cellier: **Continuous System Modeling**. Springer-Verlag New York, 1991.

[6] F. E. Cellier, R. T. McBride: **Object-oriented Modeling of Complex Physical Systems Using the Dymola Bond-graph Library**. Proc. ICBGM'03 pp. 157-162, Orlando Florida, 2003.

[7] F. E. Cellier, A. Nebot: **The Modelica Bond Graph Library**. Proc. 4th International Modelica Conference, Hamburg Germany, 2005.

[8] V. Damic and J. Montgomery: **Mechatronics by Bond Graphs**. Springer-Verlag, 2003.

[9] W. Dunham: **Euler, The Master of Us All**. The mathematical association of America, 1999.

[10] **Dymola Homepage**: www.dynasim.se

[11] H. Elmquist M. Otter, J. Díaz López: **Collision Handling for the Modelica MultiBody Library**. Proceedings of the 4th International Modelica Conference, 2005.

[12] J. Greifeneder, F. E. Cellier: **Modeling convective flows using bond graphs**. Proc. ICBMG 2001 pp. 276-284, Phoenix, Arizona, 2001

[13] J. Greifeneder: **Modellierung thermodynamischer Phänomene mittels Bondgraphen**. Master Thesis, Universität Stuttgart, Deutschland, 2001

[14] N. M. N. Gussn: **On the Bondgraphic Power Postulate and its Role in Interpreting the 1905 and 1915 Relativity Theories**. Master Thesis, Dept. of Electr. & Comp. Engr., University of Arizona, Tucson, AZ, 1994

[15] S. G. Hoggar: **Mathematics for Computer Graphics**. Cambridge University Press, 1994.

[16] D. C. Karnopp, D. L. Margolis, R. C. Rosenberg: **System Dynamics, Third Edition**. Wiley-Interscience, 2000.

[17] D. C. Karnopp, R. C. Rosenberg: **System Dynamics: a unified approach**. John Wiley, New York, 1975.

[18] M. Krebs: **Modeling of Conditional Index Changes**. Master Thesis, Dept. of Electr. & Comp. Engr., University of Arizona, Tucson, AZ, 1997.

[19] H. M. Paynter, **Analysis and Design of Engineering Systems**. M.I.T. Press, Cambridge, Mass., 1961

[20] Ernst Mach: **Die Mechanik in ihrer Entwicklung, 7. Auflage, 1912** . Akademie-Verlag Berlin, 1988.

[21] **Modelica Homepage**: www.modelica.org

[22] Isaac Newton: **Mathematische Grundlagen der Naturphilosophie (Pricipia Mathematica) 3. Ausgabe, 1726** . Felix Meiner Verlag, 1988.

[23] Isaac Newton **The Principia. A new translation by I. Bernhard Cohen and Anne Whitman** University of California Press, 1999

[24] M. Otter, H. Elmquist and S. E. Mattsson: **The New Modelica MultiBody Library**. Proceedings of the 3rd International Modelica Conference, 2003.

[25] M. Otter, H. Elmquist, S. E. Mattsson: **Hybrid Modeling in Modelica based on the Synchronous Data Flow Principle**. CACSD'99, Hawaii, USA, 1999.

[26] A. L. Schwab, J. P. Meijaard, J. M. Papadopoulos: **Benchmark results on the linearized equation of motion of an uncontrolled bicycle**. Proceedings of ACMD 2004.

[27] Istvan Szabo: **Geschichte der mechanischen Prinzipien, 3. Auflage 1986**. Birkhäuser Verlag Basel, 1987.

[28] A. Zeid, C.-H. Chung: **Bond Graph Modeling of MultiBody Systems: A Library od Three Dimensional Joints**. J. Franklin Inst. 329, pp. 605-636, 1992