



# Modelica Introduction

*Modeling of Multi-Body Systems*

Hubertus Tummescheit

UTRC

With Material from Martin Otter

## Outline

- Dynamics of a free rigid body
- Direct object-oriented modeling
- Object-oriented modeling with relative variables
- Demo of vehicle model in Dymola
- Modeling and numerical details in Appendix

## Methods of modeling MBS

- Current tools use different methods for object-based modeling of MBS
- Direct modeling of MBS
  - fewer basic models
  - less reliable numerics
  - large, sparse DAE-systems (slower)
- Modeling of MBS using relative variables
  - more basic models
  - standard, high quality solvers can be used
  - forward and inverse problem based on same code
  - less convenient when modeling kinematic loops (only old MBS, not Modelica.MultiBody)

## Dynamic equations of a free body

No contact to the environment or other bodies

Forces and torques are acting on the body

Equations of Newton and Euler:

$${}^0\dot{\mathbf{p}} = \Sigma {}^0\mathbf{f}_i$$

$${}^0\dot{\mathbf{L}} = \Sigma {}^0\boldsymbol{\tau}_i$$

Center of mass  
(body fixed frame, index b)

Equations are expressed in inertial frame

0

Inertial system (frame index 0)

## Dynamic equations of a free body

$\mathbf{p} = m\mathbf{v}$  linear momentum ( ${}^0\mathbf{v} = {}^0\dot{\mathbf{r}}$ )

$\mathbf{L} = \mathbf{I}\boldsymbol{\omega}$  angular momentum ( ${}^0\boldsymbol{\omega} = \mathbf{T}vec(\mathbf{T}^T\mathbf{T})$ )

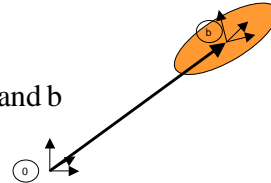
$\mathbf{T}$  transformation matrix between frames 0 and b

Differentiating the angular momentum results in:

$$\dot{\mathbf{L}} = \mathbf{I}\dot{\boldsymbol{\omega}} + \dot{\mathbf{I}}\boldsymbol{\omega}$$

Instead of differentiating the inertia tensor, apply Euler's differentiation rule to the angular momentum vector

$$\begin{aligned} {}^0\dot{\mathbf{L}} &= \mathbf{T}({}^b\dot{\mathbf{L}} + {}^b\boldsymbol{\omega} \times {}^b\mathbf{L}) \\ &= \mathbf{T}({}^b\mathbf{I}{}^b\dot{\boldsymbol{\omega}} + {}^b\boldsymbol{\omega} \times {}^b\mathbf{L}) \\ &= {}^0\mathbf{I}{}^0\dot{\boldsymbol{\omega}} + {}^0\boldsymbol{\omega} \times {}^0\mathbf{I}{}^0\boldsymbol{\omega} \end{aligned}$$



## Dynamic equations of a free body

Variant of Newton's and Euler's equations for practical calculations:

$$\begin{aligned} m\dot{\mathbf{v}} &= \sum {}^0\mathbf{f}_i \\ {}^0\mathbf{I}{}^0\dot{\boldsymbol{\omega}} + {}^0\boldsymbol{\omega} \times {}^0\mathbf{I}{}^0\boldsymbol{\omega} &= \sum {}^0\boldsymbol{\tau}_i \end{aligned}$$

To get a complete set of differential equations, the kinematic equations have to be added. Motion of a body can be described by 6 position and 6 velocity coordinates. The difficult part is the description of the rotation.

## Cardan angles for rotation description

Selection of state variables:

- r** position vector to center of mass
- $\varphi$**  cardan angles of body fixed frame w.r.t.inertial frame  $\varphi=[\alpha, \beta, \gamma]$
- v** velocity of center of mass
- $\omega$**  angular velocity of body fixed frame w.r.t.inertial frame

Differential equations of body:

$$\mathbf{r} = \dot{\mathbf{v}}$$

$$\mathbf{A}(\varphi)\dot{\varphi} = \omega$$

$$m\dot{\mathbf{v}} = \Sigma \mathbf{f}_i$$

$$\mathbf{I}\dot{\omega} + \omega \times \mathbf{I}\omega = \Sigma \tau_i$$

## Cardan angles for rotation description

12 differential equations for 12 unknowns, but:

matrix **A** is singular for  $\beta=90^\circ$

Numerically sound way: use two different sets of

3-parameterizations with different singularities and switch  
between these two sets of state variables during integration  
 $\Rightarrow$  handling of the switching.

Method is one of the choices implemented in Dymola/new MultiBody  
library

## Euler parameters for rotation description

- $\mathbf{r}$  position vector to center of mass
- $\mathbf{q}, q_0$  Euler parameters of body fixed frame w.r.t. inertial frame
- $\mathbf{v}$  velocity of center of mass
- $\boldsymbol{\omega}$  angular velocity of body fixed frame w.r.t. inertial frame

Differential equations of body:

$$\begin{aligned} \mathbf{r} &= \dot{\mathbf{v}} \\ \mathbf{A}(\mathbf{q}, q_0) \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{q}_0 \end{bmatrix} &= \boldsymbol{\omega} \\ q_0^2 + \mathbf{q}\mathbf{q}^T &= 1 \\ m\dot{\mathbf{v}} &= \Sigma \mathbf{f}_i \\ \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} &= \Sigma \boldsymbol{\tau}_i \end{aligned}$$

13 differential equations in 13 unknowns. This is a high index system. It is not possible to solve for  $\dot{\mathbf{q}}, \dot{q}_0$  (A is a 3 x 4 matrix)

## Euler parameters for rotation description

Differentiating the constraint equation leads to

$$2 \begin{bmatrix} \mathbf{q}^T q_0 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{q}_0 \end{bmatrix} = 0$$

Adding this to the set of equations for the body leads to

$$\begin{aligned} \mathbf{r} &= \dot{\mathbf{v}} \\ \tilde{\mathbf{A}}(\mathbf{q}, q_0) \begin{bmatrix} \dot{\mathbf{q}} \\ \dot{q}_0 \end{bmatrix} &= \begin{bmatrix} \boldsymbol{\omega} \\ 0 \end{bmatrix} \\ q_0^2 + \mathbf{q}\mathbf{q}^T &= 1 \\ m\dot{\mathbf{v}} &= \Sigma \mathbf{f}_i \\ \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} &= \Sigma \boldsymbol{\tau}_i \end{aligned}$$

Overdetermined set of 14 equations in 13 unknowns. Matrix  $\tilde{\mathbf{A}}$  is always regular. This problem can not be eliminated using the dummy derivative method, since every reduction of the 4 rotation variables to 3 will re-introduce a singularity.

## Euler parameters for rotation description

Two numerically sound ways of solving the resulting equations:

- Use a special index 2 DAE solver to solve the original set of 13 equations in 13 unknowns (e.g. MEXX).
- Use a special solver for overdetermined DAEs to solve the set of 14 equations in 13 unknowns (e.g. ODASSL).

## Other possibility:

### Euler parameters for rotation description

Euler parameters or quaternions are another (overdetermined) possibility to parameterize the body rotation. Analysis reveals that the resulting numerics lead to either an index 2 DAE or overdetermined equations.

Two numerically sound ways of solving the resulting equations:

- Use a special index 2 DAE solver to solve the original set of 13 equations in 13 unknowns (e.g. MEXX).
- Use a special solver for overdetermined DAEs to solve the set of 14 equations in 13 unknowns (e.g. ODASSL).

## Numerical solution of high index DAE

Consider DAEs of the form

$$0 = f(\dot{x}(t), x(t), w(t), t)$$

where  $x$  are variables that appear differentiated and  $w$  are purely algebraic variables with  $\dim(f) = \dim(x) + \dim(w)$ .

The numerical solution of the DAE above can be analyzed by examining the exact solution of the slightly perturbed DAE

$$f(\hat{y}(t), \hat{y}(t), t) = \varepsilon(t)$$

where  $\varepsilon(t)$  is small. The difference in the solutions of these two DAEs can be bounded by

$$\max |y(t) - \hat{y}(t)| \leq C(y(0) - \hat{y}(0) + \max \left| \int \varepsilon(t) dt \right| + \max |\varepsilon(t)| + \max |\varepsilon(t)'| + \dots + \max |\varepsilon(t)^{j-1}|)$$

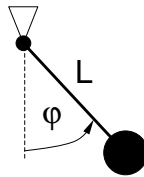
## Numerical solution of high index DAE

Loosely speaking, the numerical approximation of the solution  $\hat{y}(t)$  is a function of the  $j-1^{\text{th}}$  derivative of the rounding and truncation error. If this error is small, its derivatives may still be big and therefore a big overall approximation error can be expected in the solution whenever the DAE index is greater than one.

Standard DAE solvers, like DASSL and its many derivatives, can only solve index 1 DAEs reliably.

Other methods to deal with high index DAE, e.g. projection methods, are slower and suffer from other shortcomings.

## Example: Simple pendulum



Derivation by hand:

$$\dot{\varphi} = \omega$$
$$\dot{\omega} = -\frac{g}{L} \sin(\varphi)$$

Direct modeling using Euler angles:

Switching between 2 different sets of 3-parameterizations:

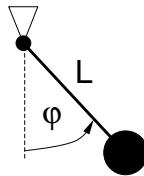
12 equations for body

5 equations for joint

i.e. event-dependent index 3 DAE with 17 equations

(special purpose integrator needed)

## Example: Simple pendulum



Over-determined DAE system:

13 equations for body

5 equations for joint

10 equations by differentiation

i.e. overdetermined DAE with 28 equations

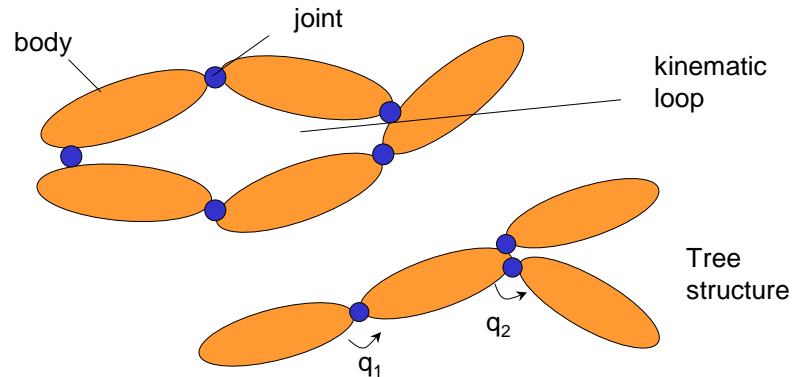
(special purpose integrator needed).

This method is used by ADAMS (market leader for MBS-simulation)



## Better alternative for object-oriented modeling

Two different kinematic structures:



For tree-structured systems, the position of an mbs is uniquely defined when the relative joint variables  $q$  are given, i.e., when these variables are used as state variables.

## Modeling with relative variables

Central idea:

Derive equations for inverse problem, i.e., given the relative joint variables and their first and second derivatives ( $q$   $q'$   $q''$ ), compute the driving forces/torques in the joints

For a revolute joint, the relative joint variable is the relative angle; the driving torque is a possible torque around the axis of rotation.

For a prismatic joint, the relative joint variable is the relative distance; the driving force is a possible force along the axis of translation.

Transport not only  $r$  and  $T$  in a mechanical cut, but also

$v$	velocity of origin of cut frame
$\omega$	angular velocity of cut frame
$a$	acceleration of origin of cut frame
$\alpha$	angular acceleration of cut frame

## Modeling with relative variables

- Compute position vectors, transformation matrices, velocities and accelerations of all cut frames in a recursive way, starting at the inertial system and moving in the direction of the outboard bodies. This is possible because the **relative variables**  $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$  are assumed to be known.
- Compute the driving forces/torques  $\mathbf{f}$  in a recursive way, starting at the outward bodies and moving in the direction of the inertial system.

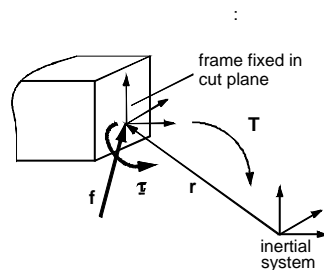
Since  $\dim(\mathbf{f}) = \dim(\ddot{\mathbf{q}})$ , the generated equations can also be used for the **simulation problem**, where

- $\mathbf{q}, \dot{\mathbf{q}}$  are **state variables**,
- $\mathbf{f}$  are **known applied forces** in the joints (e.g. zero),
- $\ddot{\mathbf{q}}$  are the **unknown highest derivatives**.  
All other variables are treated as purely algebraic variables.

## Definition of a 3D mechanical cut

At a mechanical 3D cut, components can be attached rigidly together. The cut-plane is uniquely identified by a frame fixed in the cut-plane, called cut frame, which is described by the following variables:

- |                       |  |
|-----------------------|--|
| $\mathbf{r}$          | position vector of origin of cut frame                         |
| $\mathbf{T}$          | transformation matrix from the cut frame to the inertial frame |
| $\mathbf{f}$          | cut force at the origin of the cut frame                       |
| $\boldsymbol{\tau}$   | cut torque at the origin of the cut frame                      |
| $\mathbf{v}$          | velocity of origin of cut frame                                |
| $\boldsymbol{\omega}$ | angular velocity of cut frame                                  |
| $\mathbf{a}$          | acceleration of origin of cut frame                            |
| $\boldsymbol{\alpha}$ | angular acceleration of cut frame                              |



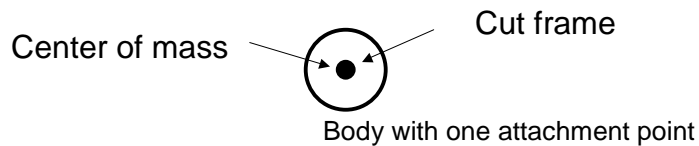
## Definition of a 3D mechanical cut

```

record Frame "Frame record of a mechanical element"
  SI.Position r0[3] "Position vector from inertial system to frame origin";
  Real S[3, 3] "Transformation matrix from frame_a to inertial system";
  SI.Velocity v[3] "Absolute velocity of frame origin";
  SI.AngularVelocity w[3] "Absolute angular velocity of frame_a";
  SI.Acceleration a[3] "Absolute acceleration of frame origin";
  SI.AngularAcceleration z[3] "Absolute angular acceleration of frame_a";
  flow SI.Force f[3];
  flow SI.Torque t[3];
end Frame;

```

## Model body



No Cardan angles or Euler parameters are needed,  
because the velocity and acceleration of the body are already  
provided in the cut frame!

$$m\mathbf{a} = \mathbf{f}$$

$$\mathbf{I}\boldsymbol{\alpha} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} = \boldsymbol{\tau}$$

No Cardan angles or Euler parameters are needed,  
because the velocity and acceleration of the body are already  
provided in the cut frame

## Model bar

The following variables are defined for cut A and correspondingly for cut B:

$$\mathbf{r}_a, {}^0\mathbf{T}^a, \mathbf{v}_a, \boldsymbol{\omega}_a, \mathbf{a}_a, \boldsymbol{\alpha}_a, \mathbf{f}_a, \boldsymbol{\tau}_a$$

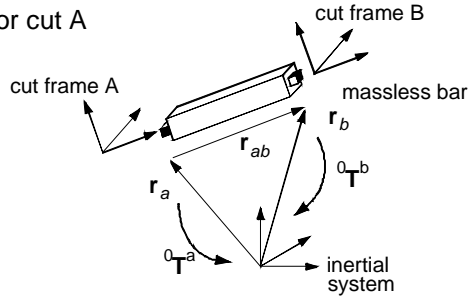
The bar contains exactly the same equations as before and additionally the relationships between the velocities and accelerations at the two cut frames ( $\mathbf{r}_{ab} = {}^0\mathbf{T}^a \mathbf{r}_{ab}$ ):

$$\mathbf{v}_b = \mathbf{v}_a + \boldsymbol{\omega}_a \times \mathbf{r}_{ab}$$

$$\mathbf{a}_b = \mathbf{a}_a + \boldsymbol{\alpha}_a \times \mathbf{r}_{ab} + \boldsymbol{\omega}_a \times (\boldsymbol{\omega}_a \times \mathbf{r}_{ab})$$

$$\boldsymbol{\omega}_a = \boldsymbol{\omega}_b$$

$$\boldsymbol{\alpha}_a = \boldsymbol{\alpha}_b$$

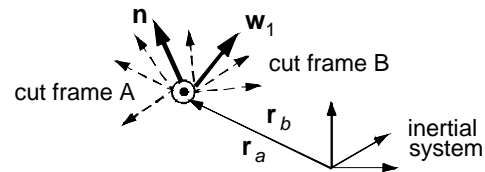


## Model revolute joint

The following variables are defined for cut A and correspondingly for cut B:

$$\mathbf{r}_a, {}^0\mathbf{T}^a, \mathbf{v}_a, \boldsymbol{\omega}_a, \mathbf{a}_a, \boldsymbol{\alpha}_a, \mathbf{f}_a, \boldsymbol{\tau}_a$$

The relative angle  $\varphi$  is introduced as additional variable as well as its first and second derivative. The equations in the revolute joint express how the variables of cut frame B are computed, given the variables of cut frame A, the axis of rotation  $\mathbf{n} = {}^a\mathbf{n}$  and the relative variables  $\varphi, \dot{\varphi}, \ddot{\varphi}$



$${}^b\mathbf{T}^a = \mathbf{n}\mathbf{n} + (\mathbf{E} - \mathbf{n}\mathbf{n}^T)\cos(\varphi) + \text{skew}(\mathbf{n})\sin(\varphi)$$

$${}^0\mathbf{T}^b = {}^0\mathbf{T}^a {}^a\mathbf{T}^b$$

$$\boldsymbol{\omega}_b = \boldsymbol{\omega}_a + {}^0\mathbf{n}\dot{\varphi}$$

$$\mathbf{a}_b = \mathbf{a}_a + {}^0\mathbf{n}\ddot{\varphi} + \boldsymbol{\omega}_a \times {}^0\mathbf{n}\dot{\varphi}$$

$$\mathbf{r}_b = \mathbf{r}_a$$

$$\mathbf{v}_b = \mathbf{v}_a$$

$$\mathbf{a}_b = \mathbf{a}_a$$

## Model revolute joint

As before, the force/torque balance at the revolute joint leads to the following two equations:

$$\mathbf{f}_a + \mathbf{f}_b = 0$$

$$\boldsymbol{\tau}_a + \boldsymbol{\tau}_b = 0$$

Introduce additionally the driving torque  $\tau$  acting along the axis of rotation.

For the inverse problem,  $\varphi, \dot{\varphi}, \ddot{\varphi}$  are assumed to be known and  $\tau$  should be computed. This is done by projecting the cut-torque at one of the two cuts onto the axis of rotation:

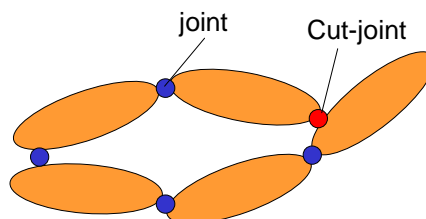
$$\tau^0 + {}^0\mathbf{n}^T \boldsymbol{\tau}_b = 0$$

## Problems with using relative variables:

Kinematic loops can not use the same joint models as tree-structured systems. The cut variables are not minimal,  $\Rightarrow$  the overall system of equations is over-determined. Remedy in the (old) MBS-library:

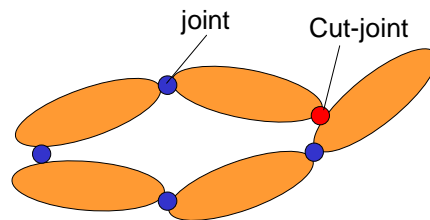
Use special **cut-joints** with a smaller number of equations and special constraint equations

Disadvantage: selection of cut-joints in complex kinematic situations, e.g. car suspension systems, can be difficult and requires user know-how and experience.

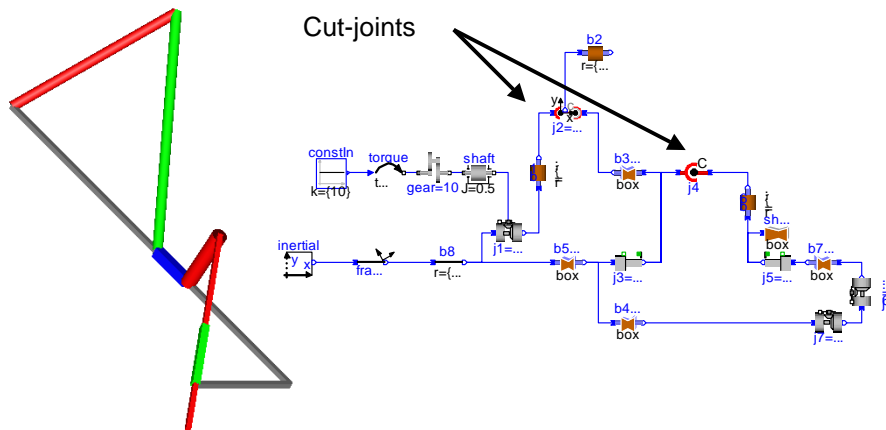


## Solution in new MultiBody library:

Connection graph is used to detect possible cut-joints automatically  
For details of new MBS-library see paper by Otter/Elmqvist/Mattsson  
At course home page.

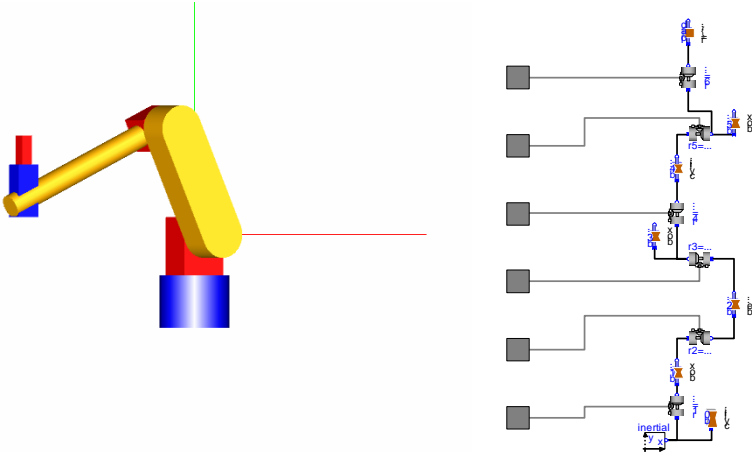


## Examples from MBS-library



Two kinematic loops  $\Rightarrow$  two cut-joint models needed

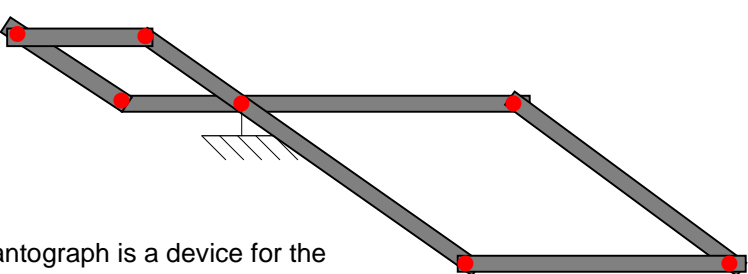
# Examples from MBS-library



No kinematic loops  $\Rightarrow$  only standard revolute joint models

## Exercise:

Using the old & new MBS-libraries, build a model of a pantograph:



A diagram of a pantograph mechanism. It consists of seven links connected by seven revolute joints (red dots). The mechanism is shown in a specific configuration. A fixed pivot is at the top left. A horizontal link of length 1m extends to the right from this pivot. A second link of length 1m connects the end of the first link to a third link of length 1m, which is angled downwards. This third link is connected to a fourth link of length 1m, which is horizontal and passes through a fixed pivot point. A fifth link of length 1m connects the end of the fourth link to a sixth link of length 1m, which is angled downwards. The sixth link is connected to a seventh link of length 1m, which is horizontal and ends at a fixed pivot point. The diagram illustrates the geometry and constraints of the pantograph mechanism.

A Pantograph is a device for the mechanical copying of maps etc. Make the long bars 1 m from joint to joint and make this pantograph useful for shrinking its right-hand side movements by a ratio of 3:1

## Summary

- Basic Equations for MBS systems
- Short comparison of different modeling options (details in Appendix)
- Modeling of MBS using relative variables
  - often much faster: capable of real-time simulation of complex models
  - same model can be used for simulation problem and inverse dynamics
  - less user friendly for kinematic loops
  - Problem has been solved in new MultiBody library (which is not yet used for the VehicleDynamics lib).

## Appendix:

- Tearing for inverse solution of MBS problems
- Detailed description of model equations for direct Modeling of MBS systems:
  - fewer different models, more user friendly
  - less efficient and less reliable numerics
  - not suitable for inverse dynamics



## Tearing of sparse equation systems

Tearing is a technique to reduce large, sparse systems of equations into small dense ones by choosing a subset of the variables, e.g.  $z_i$  in a way that, if  $z_i$  are known, all other variables can be computed in an explicit way. In other words: the remaining system can be transformed into a BLT-partitioned system where all blocks have dimension 1.

In general, finding tearing variables and residue equations such that the resulting set of equations is minimal, is an NP-complete problem. It can only be carried out by trying all possible combinations. Algorithms for this purpose are therefore always heuristic and don't guarantee to find the minimum number of equations. An additional constraint is that tearing has to keep the regularity of the original matrix.

Remedy:

By knowing the structure of the equations, make the selection based on physical insight and guarantee regularity by physical reasoning.

## Tearing

Tearing is a technique to reduce large, sparse systems of equations to small dense ones by choosing a subset of the variables, e.g.  $z_i$  in a way that, if  $z_i$  are known, all other variables can be computed in an explicit way. In other words: the remaining system can be transformed into a BLT-partitioned system where all blocks have dimension 1.

In general, finding tearing variables and residue equations such that the resulting set of equations is minimal, is an NP-complete problem. It can only be carried out by trying all possible combinations. Algorithms for this purpose are therefore always heuristic and don't guarantee to find the minimum number of equations. An additional constraint is that tearing has to keep the regularity of the original matrix.

Remedy:

By knowing the structure of the equations, make the selection based on physical insight and guarantee regularity by physical reasoning.

## Tearing

Tearing is a technique to reduce large, sparse systems of equations to small dense ones by choosing a subset of the variables, e.g.  $z_t$  in a way that, if  $z_t$  are known, all other variables can be computed in an explicit way. In other words: the remaining system can be transformed into a BLT-partitioned system where all blocks have dimension 1.

BLT-partitioning finds minimal algebraic systems of equations of the form " $h(z) = 0$ " in the overall DAE. When these are large and sparse, tearing means to select tearing variables  $z_t$  and residue equations  $h_t$  such that  $h$  is separated into two parts:

$$\begin{aligned} z_r &= h(z_t) \\ h(z_t, z_r) &= \text{residue}(z_t) \end{aligned}$$

The residue operator is used to characterize the residue equations and tearing variables. The above transformation reduces the dimension of the equations: When " $x = z_t$ " is provided from the solver,  $z_t$  can be computed from the first equation. The second equation gives the residue.

## Tearing, linear example

If the original equation is linear:

$$\begin{bmatrix} L & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} z_r \\ z_t \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 + \text{residue}(z_t) \end{bmatrix}$$

The dimension of the equation system can be reduced to:

$$(A_{22} - A_{21}L^{-1}A_{12})z_t = b_2 - A_{21}L^{-1}b_1$$

This transformation can be done fully symbolically (without knowing the actual numerical values), provided matrix  $L$  is lower triangular (and regular).

## Using physical insight for tearing

When dynamics is present at all essential places, no algebraic loops occur. Algebraic loops are only present, if dynamics is neglected, e.g. in the following way:

$$\varepsilon \dot{x} = f(x)$$

If  $\varepsilon$  goes to zero, the derivative is cancelled and the equation reduces to an algebraic equation. The residue operator can be seen as:

$$\text{residue}(x) = \lim_{\varepsilon \rightarrow 0} \varepsilon \dot{x}$$

In other words, the residue operator can be seen as an element with infinite fast dynamics. It should therefore be placed in such a way that newly introduced dynamics would brake an algebraic loop.

## Using physical insight for tearing

From the solution of the inverse dynamics problem it is known that, if  $\ddot{\mathbf{q}}$  would be known, all other quantities and especially the driving forces/torques  $\mathbf{f}$  could be computed. Therefore,  $\ddot{\mathbf{q}}$  are good candidates for tearing variables. In the Modelica language, it is easy to find such variables from the symbolic equation system.

Since  $\ddot{\mathbf{q}}$  are the tearing variables, the dimension of the resulting system of equations is equal to the number of degrees of freedom of the multibody system.

This is the same result as it is derived in traditional mechanics using a mechanical principle, like d'Alembert's principle, Jourdain's principle, Kane's equations or Lagrange's equations of the second kind.

## Direct object-oriented modeling of MBS

Build up a multi body system using the basic elements:

1. Inertial system.
2. Body with mass and inertia having one attachment point at the center of mass.
3. Massless bar with two attachment points.
4. Revolute joint.
5. Prismatic joint.
6. General force element.

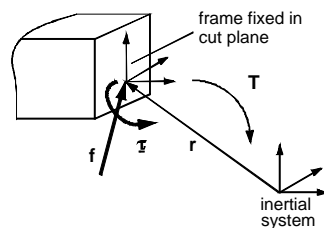
All other elements are built up by composition of the basic ones.

## Definition of a 3D mechanical cut

At a mechanical 3D cut, components can be attached rigidly together. The cut-plane is uniquely identified by a frame fixed in the cut-plane, called cut frame.

It is described by the following variables:

- $r$  position vector of origin of cut frame
- $T$  transformation matrix from the cut frame to the inertial frame
- $f$  cut force at the origin of the cut frame
- $\tau$  cut torque at the origin of the cut frame



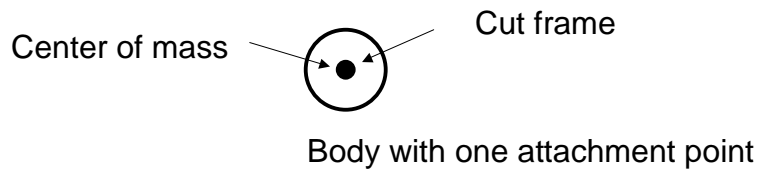
## Definition of a 3D mechanical cut

```

record Cut "Cut record of a mechanical element"
  SI.Position r0[3] "Position vector from inertial system to Cut origin";
  Real S[3, 3] "Transformation matrix from Cut_a to inertial system";
  flow SI.Force f[3] "force is a flow variable";
  flow SI.Torque t[3] "torque is a flow variable";
end Cut;

```

## Model body



Use either 3 Cardan angles or Euler parameters to describe the body. Here: Cardan angles:

$$\varphi = \varphi(\mathbf{T})$$

$$\mathbf{r} = \dot{\mathbf{v}}$$

$$\mathbf{A}(\varphi)\dot{\varphi} = \boldsymbol{\omega}$$

$$m\dot{\mathbf{v}} = \mathbf{f}$$

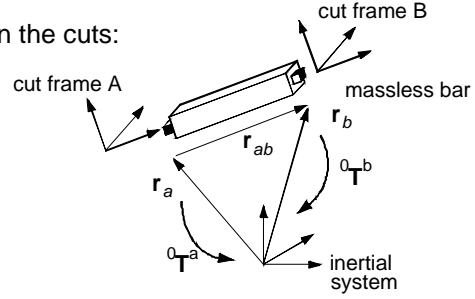
$$\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} = \boldsymbol{\tau}$$

## Model bar

The following variables are defined in the cuts:

$$\mathbf{r}_a, {}^0\mathbf{T}^a, \mathbf{f}_a, \boldsymbol{\tau}_a, \mathbf{r}_b, {}^0\mathbf{T}^b, \mathbf{f}_b, \boldsymbol{\tau}_b$$

The equations of the bar have to state the relationship between all cut variables. The bar itself is defined by the relative position vector  ${}^a\mathbf{r}_{ab}$ , resolved in cut frame A. This leads to the following equations ( $\mathbf{r}_{ab} = {}^0\mathbf{r}_{ab} = {}^0\mathbf{T}^a {}^a\mathbf{r}_{ab}$ )



$$\mathbf{r}_a + \mathbf{r}_{ab} = \mathbf{r}_b$$

$${}^0\mathbf{T}^a = {}^0\mathbf{T}^b$$

$$\mathbf{f}_a + \mathbf{f}_b = 0$$

$$\boldsymbol{\tau}_a + \boldsymbol{\tau}_b + \mathbf{r}_{ab} \times \mathbf{f}_b = 0$$

## Model revolute joint

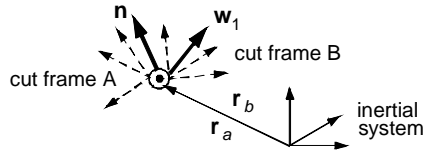
The following variables are defined in the cuts:

$$\mathbf{r}_a, {}^0\mathbf{T}^a, \mathbf{f}_a, \boldsymbol{\tau}_a, \mathbf{r}_b, {}^0\mathbf{T}^b, \mathbf{f}_b, \boldsymbol{\tau}_b$$

The equations of the revolute joint state the relationship between all cut variables. The joint is defined by the axis of rotation  ${}^a\mathbf{n}$  and two axes  ${}^b\mathbf{w}_1, {}^b\mathbf{w}_2$ , which are orthogonal to  $\mathbf{n}$ . Using the auxiliary variables

$${}^b\mathbf{T}^a = {}^0\mathbf{T}^{bT} {}^0\mathbf{T}^a, \quad {}^b\mathbf{n} = {}^b\mathbf{T}^a {}^a\mathbf{n}$$

A revolute joint can be described by:



$$\mathbf{r}_a = \mathbf{r}_b$$

$${}^b\mathbf{n}^T {}^b\mathbf{w}_1 = 0$$

$${}^b\mathbf{n}^T {}^b\mathbf{w}_2 = 0$$

$$\mathbf{f}_a + \mathbf{f}_b = 0$$

$$\boldsymbol{\tau}_a + \boldsymbol{\tau}_b = 0$$

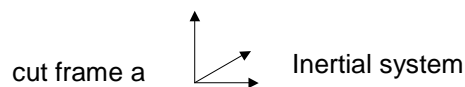
$$\boldsymbol{\tau}_b = {}^0\mathbf{T}^b \begin{bmatrix} {}^b\mathbf{w}_1 & {}^b\mathbf{w}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\tau}_{w1} \\ \boldsymbol{\tau}_{w2} \end{bmatrix}$$

## Model revolute joint

- Contrary to the bar object, there is the condition that the cut torque in direction of the axis of rotation is zero. In order to utilize this condition, the two constraint torques  $\tau_{w1}$ ,  $\tau_{w2}$  are introduced as new unknown variables in the direction of the given vectors  ${}^b w_1$ ,  ${}^b w_2$ .  
The cut torques are then expressed as functions of  $\tau_{w1}$ ,  $\tau_{w2}$
- The revolute joint introduces 5 constraint equations:  
Especially, the axis of rotation in cut frame A must always be orthogonal to the two vectors  ${}^b w_1$ ,  ${}^b w_2$  which are by construction orthogonal to the axis of rotation resolved in frame B.

A model for a prismatic joint is defined in a similar way.

## Model inertial system



The inertial system has only one cut frame and the equations simply state that the kinematic variables are zero:

$$\mathbf{r}_a = 0$$

$${}^0\mathbf{T}_a = 0$$

Therefore, the kinematic variables of every object connected to the inertial frame are set to zero.

## Defining the cut frames

Several vectors, like the relative position vector of a bar or the axis of rotation of a revolute joint, have to be defined in specific cut frames. The multibody system is defined in a special, user defined, position which is called home position.

By definition, all cut frames are parallel to the inertial frame in the home position! All cut frames are thus implicitly defined.

All the special vectors, like the axis of rotation vector of a revolute joint, are defined in the home position with respect to a specific cut frame. Since all cut frames are parallel to the inertial frame this means that all these vectors are defined with respect to the inertial frame (in the home position).

A user interaction is not needed. All vectors are just defined in the home position in the inertial frame.

## Advantages of the direct modeling

- Only few basic components.
- The description is very general. Nearly every kind of multi body system can be build up. The user does not have to handle special cases, depending on the connection structure of the multi body system (kinematic loops).



## Problems with the direct description

Every body introduces 12 state variables. Revolute or prismatic joints restrict the motion between bodies by defining 5 algebraic constraint equations. This means that there are algebraic conditions between state variables, which in turn means that the overall system has a DAE index  $> 1$ .

The constraint equations of a joint are given on position level. Differentiating these equations 2 times produces constraint equations on acceleration level. It turns out that these constraint equations can be solved for, i.e., the index of the multi body system DAE is 3.

The direct solution of the DAE requires a special purpose integrator despite the fact that the direct numerical solution of index 3 DAEs is questionable. Why?