



# **Modelling and Simulation of Rigid and Flexible Multibody Systems in Modelica**





**Tutorial at the SIMS2007 , Göteborg (Särö), October 30<sup>th</sup>, 2007**

**Andreas Heckmann, German Aerospace Center (DLR)**

**Institute of Robotics and Mechatronics**



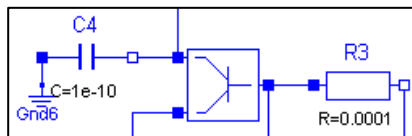
# Contents

- Modelica, very briefly
- Modelica Multibody Basics 
- Exercise 1: Control of an inverse pendulum
- Modelica Multibody Advanced 
- Exercise 2: Hexapod
- FlexibleBodies Library: Beams 
- Exercise 3: Aircraft Fin
- FlexibleBodies Library: General bodies based on finite element data 

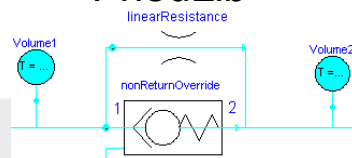
# Modelica, very briefly (see Tutorial by Peter Fritzon for more)

- General goal:
  - modelling and simulation of **complex physical systems**, consisting of components from **different engineering domains**
- Focus:
  - **System Dynamics**: global behavior of complex systems
  - differential, algebraic, discrete equations (low-order PDE only if at all, i.e. no FEM or CFD)
- Platform for collaboration between engineering disciplines
  - all-in-one approach (contrary to e.g. co-simulation)
  - specialists contribute **libraries** (open/commercial) for general use
  - supplier provides executable specification of components for OEMs

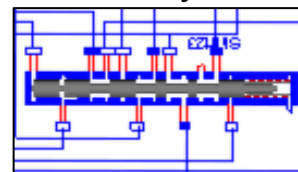
Modelica.Electrical.Analog



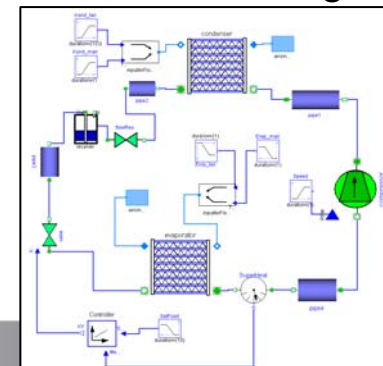
PneuLib



Modelica.Hydraulics

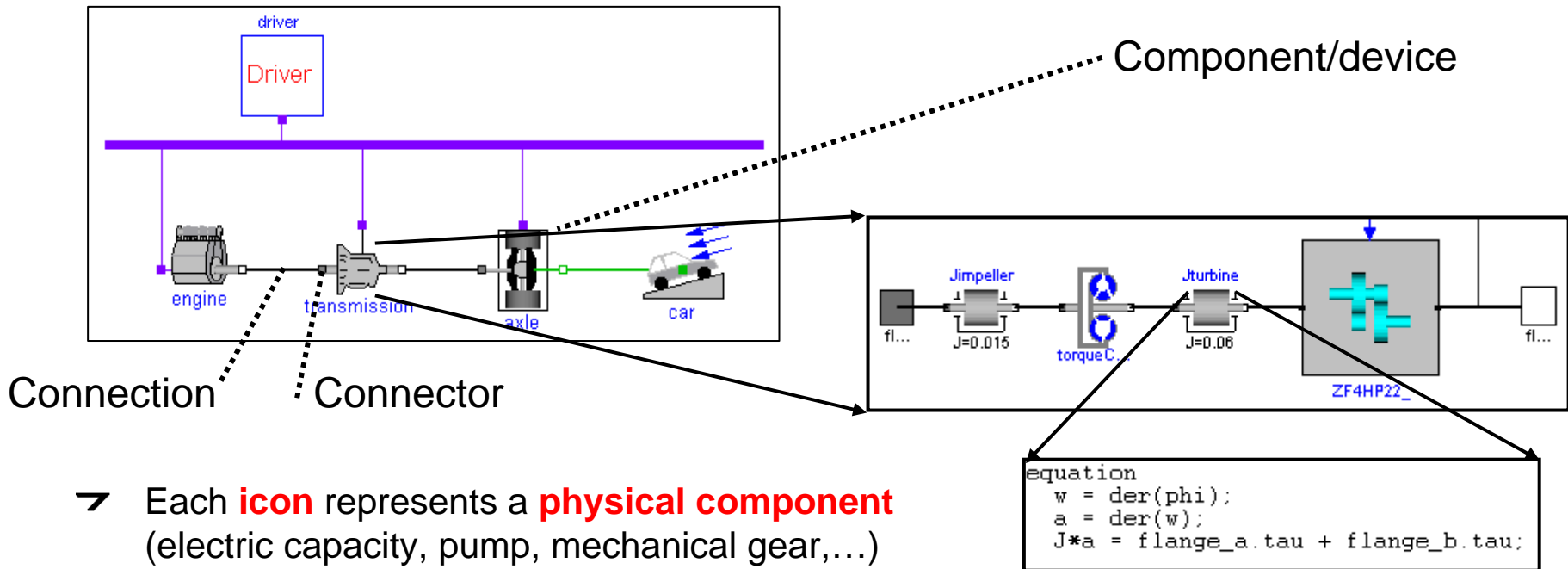


AirConditioning



# Modelica, very briefly (see Tutorial by Peter Fritzon for more)

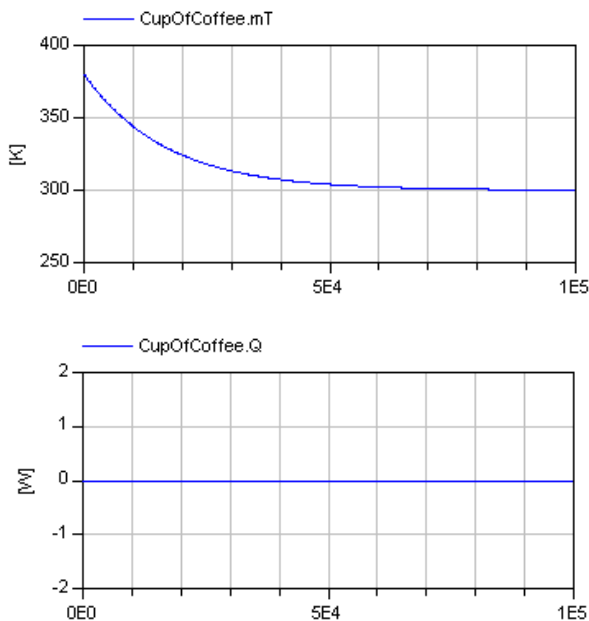
- Object orientation for complex model structuring
  - component-connections, hierarchies, inheritance



- Each **icon** represents a **physical component** (electric capacity, pump, mechanical gear,...)
- a **connection** line represents the actual physical **coupling**  
A component consists of (wire, fluid flow, heat flow,...)
- A **component** consist of **connected sub-components** (hierarchical structure) and/or is described by **equations**

# Modelica, very briefly (see Tutorial by Peter Fritzon for more)

- Deklarative, acausal modelling
  - equations no assignments
  - mathematical description no algorithms



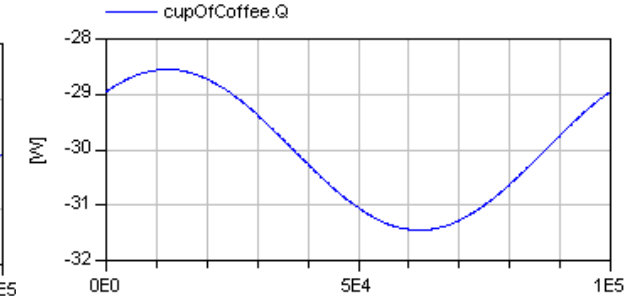
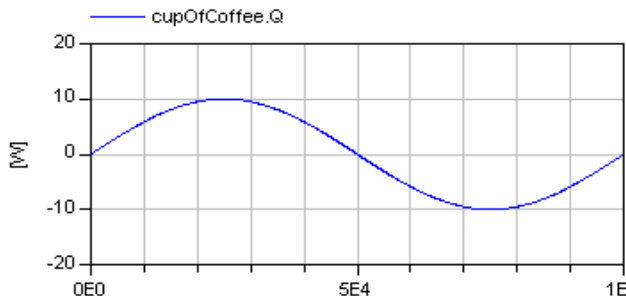
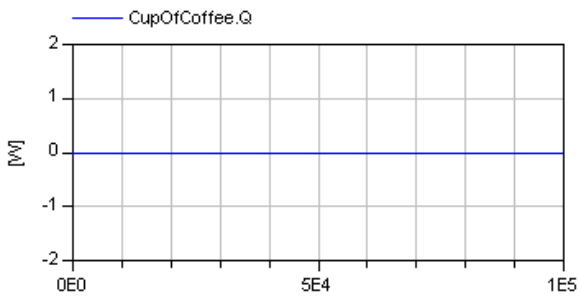
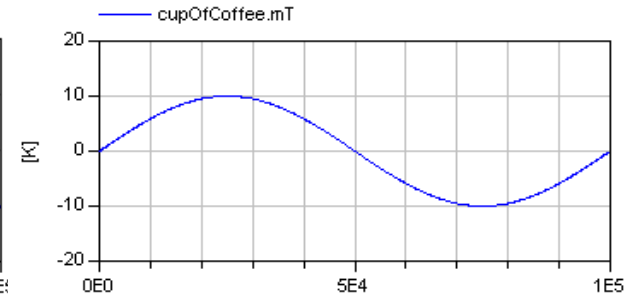
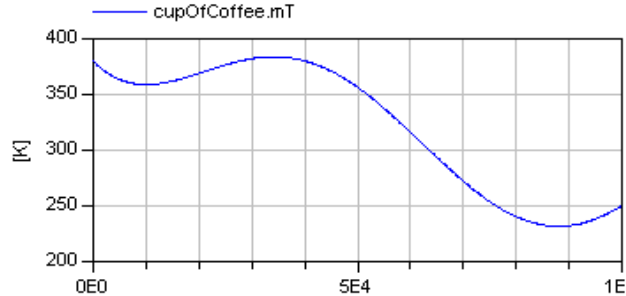
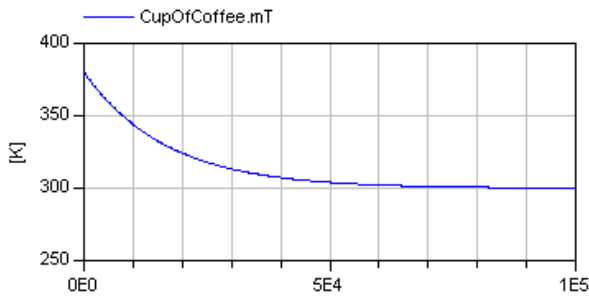
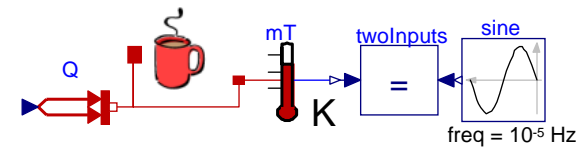
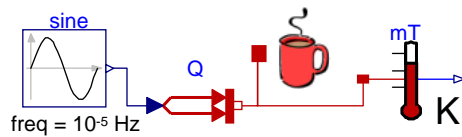
```
model CupOfCoffee
  import Modelica.SIunits.*;
  import Modelica.Thermal.HeatTransfer.Interfaces.*;

  parameter Temperature T0=380 "Initial temp.";
  parameter Temperature Tamb=300 "Ambient temperature";
  parameter Density rho=1000 "Coffee density";
  parameter SpecificHeatCapacity cv=4179 "Coffee specific heat";
  parameter CoefficientOfHeatTransfer h=25 "Convection coefficient";
  parameter Volume V=4e-4 "Volume of coffee";
  parameter Area A=4e-3 "Area of coffee";

  Temperature T(start=T0) "Coffee Temperature";
  HeatPort_a port "interface to external components" a;
  a;
equation
  0 = rho*V*cv*der(T) + h*A*(T - Tamb) - port.Q_flow "First law of thermodynamics";
  port.T = T;
end CupOfCoffee;
```

# Modelica, very briefly (see Tutorial by Peter Fritzon for more)

- Deklarative, acausal modelling
  - equations no assignments
  - mathematical description no algorithms



# Modelica, very briefly (see Tutorial by Peter Fritzon for more)

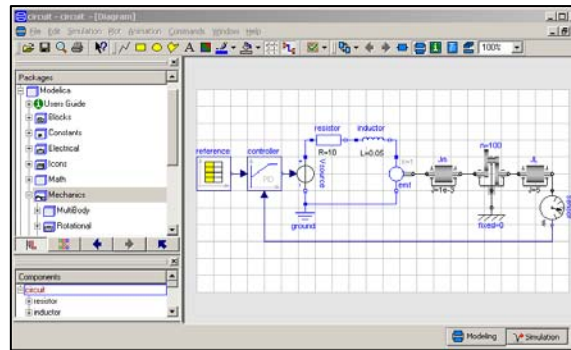
- Deklarative, acausal modelling
  - equations no assignments
  - mathematical description no algorithms
    - strictly speaking: restricted use of algorithms
  - relies on symbolic manipulations
    - on system level
    - determine signal path at compile time
    - results in state space form and explicit ode

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



# Modelica and Simulation Environments

**Graphical editor**  
for Modelica models



Modelica simulation environment  
(Dymola, MathModelica, SimulationX, Mosilab, ...)

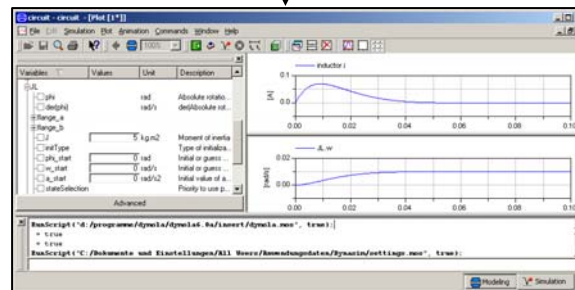
**Textual description**  
on file (equations,  
"schematic", animation)

```
model circuit
  ;
  Modelica.Electrical.Analog.Basic.Resistor resistor(R=10) ;
  Modelica.Electrical.Analog.Basic.Inductor inductor(L=0.05) ;
  Modelica.Electrical.Analog.Basic.EMF emf(k=1) ;
  Modelica.Electrical.Analog.Sources.SignalVoltage Vsource ;
  Modelica.Electrical.Analog.Basic.Ground ground ;
  Modelica.Blocks.Continuous.LimPID controller ;
  Modelica.Mechanics.Rotational.Inertia Jm(J=1e-3) ;
  Modelica.Mechanics.Rotational.IdealGear n(ratio=100) ;
  Modelica.Mechanics.Rotational.Fixed fixed ;
  Modelica.Mechanics.Rotational.Inertia JI(J=5) ;
  Modelica.Mechanics.Rotational.Sensors.SpeedSensor sensor ;
  Modelica.Blocks.Sources.CombiTimeTable reference ;
```

Free Modelica language



**Translation** of Modelica models in C-Code,  
**Simulation**, and  
interactive **scripting**  
(plot, freq. resp., ...)



Modelica Simulation-environment  
(Dymola, MathModelica, SimulationX, Mosilab, ...)



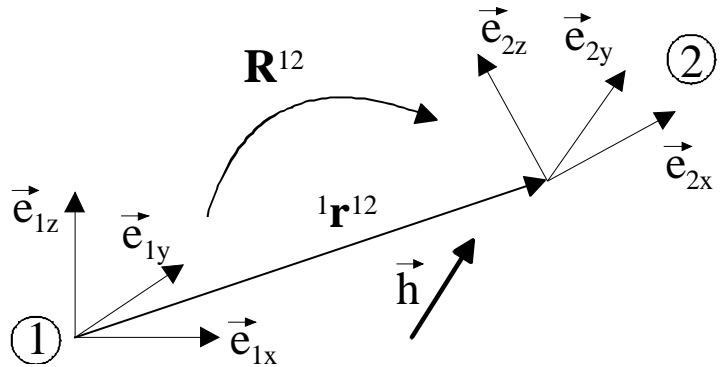
# Contents

- Introduction
  - Modelica, very briefly
- Modelica Multibody Basics
- Exercise 1: Control of an inverse pendulum
- Modelica Multibody Advanced
- Exercise 2: Hexapod
- FlexibleBodies Library: Beams
- Exercise 3: Aircraft Fin
- FlexibleBodies Livbrary: General bodies based on finite element data



# Modelica Multibody Basics: Orientation

## ➤ Coordinate systems and their orientation



```
import MultiBody.Frames;
Frames.Orientation R12;
Real h1[3] "h resolved in frame 1";
Real h2[3] "h resolved in frame 2";
equation
  h2 = Frames.resolve2(R12, h1); //or
  h1 = Frames.resolve1(R12, h2);
```

## ➤ Orientation object $R^{12}$

➤ describes orientation of coordinate system 2 wrt. 1

➤ holds

```
Real T[3, 3] "Transformation matrix from world frame to local frame";
SI.AngularVelocity w[3]
  "Absolute angular velocity of local frame, resolved in local frame";
```

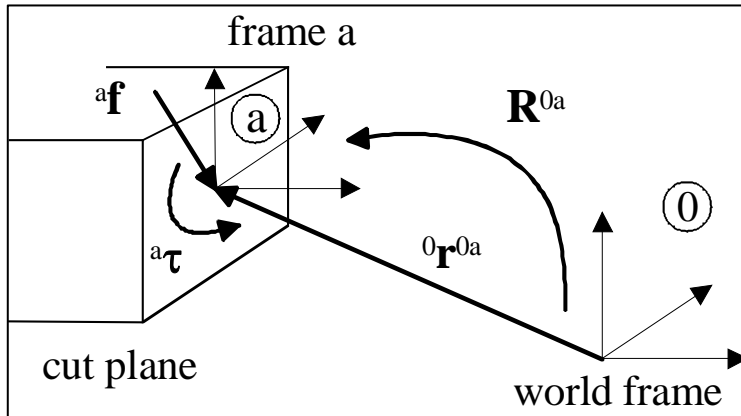
➤ may be computed using rotation angles or quaternions

➤ Multibody Lib. contains over 30 functions to operate on orientation objects

- Frames
- Orientation
  - orientationConstraint
  - angularVelocity1
  - angularVelocity2
  - resolve1
  - resolve2
  - resolveRelative
  - resolveDyade1
  - resolveDyade2
  - nullRotation
  - inverseRotation
  - relativeRotation
  - absoluteRotation
  - planarRotation
  - planarRotationAngle
  - axisRotation
  - axesRotations
  - axesRotationsAngles
  - smallRotation
  - from\_nxy
  - from\_nxz
  - from\_T
  - from\_T2
  - from\_T\_inv
  - from\_Q
  - to\_T
  - to\_T\_inv
  - to\_Q
  - to\_vector
  - to\_exy
  - length
  - normalize
  - axis

# Modelica Multibody Basics: Connectors I

- Connectors: the interface to connect components
  - Position is resolved in world frame
  - Forces and torques are resolved in local frame



```
connector Frame
  "Coordinate system fixed to the component with one cut-force and cut-torque (no icon)"
  import SI = Modelica.SIunits;
  SI.Position r_0[3]
    "Position vector from world frame to the connector frame origin, resolved in world frame";
  Frames.Orientation R
    "Orientation object to rotate the world frame into the connector frame";
  flow SI.Force f[3] "Cut-force resolved in connector frame" a;
  flow SI.Torque t[3] "Cut-torque resolved in connector frame";
end Frame;
```

non-flow !

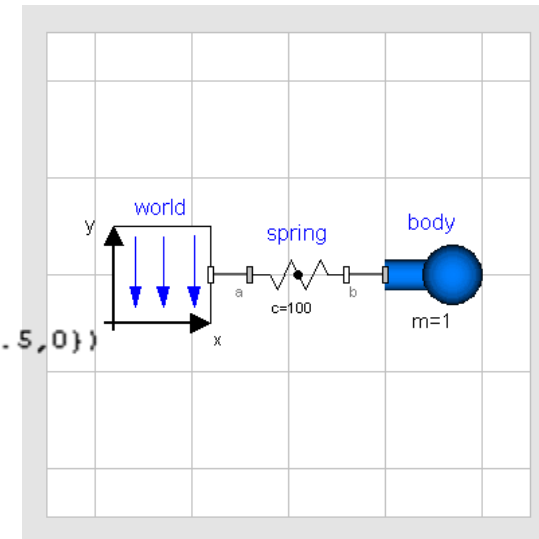
flow !



# Modelica Multibody Basics: Connectors II

## ➤ Connectors: how they work

```
model SpringMass
  inner Modelica.Mechanics.MultiBody.World world
    a;
  Modelica.Mechanics.MultiBody.Forces.Spring spring(c=100)
    a;
  Modelica.Mechanics.MultiBody.Parts.Body body(x_0_start={0,.5,0})
    a;
equation
  connect(world.frame_b, spring.frame_a) a;
  connect(spring.frame_b, body.frame_a) a;
end SpringMass;
```



## ➤ Modelica's general connections rules

- non-flow variables are set to be equal, i.e. frames coincide
  - since they represent „some kind of potential“
- flow variables sum to zero (Kirchhoff's current law)
  - since they represent time derivatives of preserved quantities
  - are consequently set to zero if connector is not connected to anything

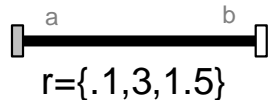
➤ see Modelica.UsersGuide.Connectors for a comparison of connectors in various domains

# Modelica Multibody Basics: Components I

## ➤ Kinematics:

- Component equations provide relations between connector variables on position level
- MultiBody.Parts.FixedTranslation  
i.e. fixed translation of frame\_b with respect to frame\_a
- Tool (e.g. Dymola) differentiates these equations twice for dynamics

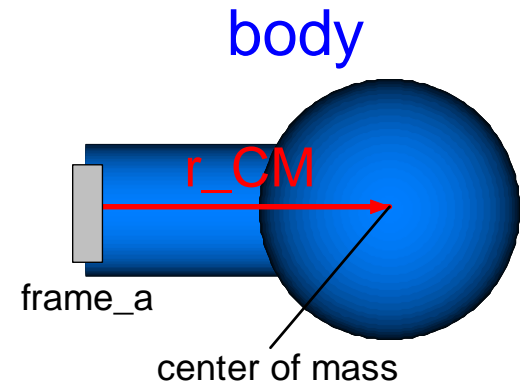
### fixedTranslation



```
frame_b.r_0 = frame_a.r_0 + Frames.resolve1(frame_a.R, r);  
frame_b.R = frame_a.R;  
  
/* Force and torque balance */  
zeros(3) = frame_a.f + frame_b.f;  
zeros(3) = frame_a.t + frame_b.t + cross(r, frame_b.f);
```

# Modelica Multibody Basics: Components II

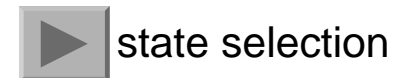
- Dynamics
  - Newton-Euler equations
  - MultiBody.Parts.Body



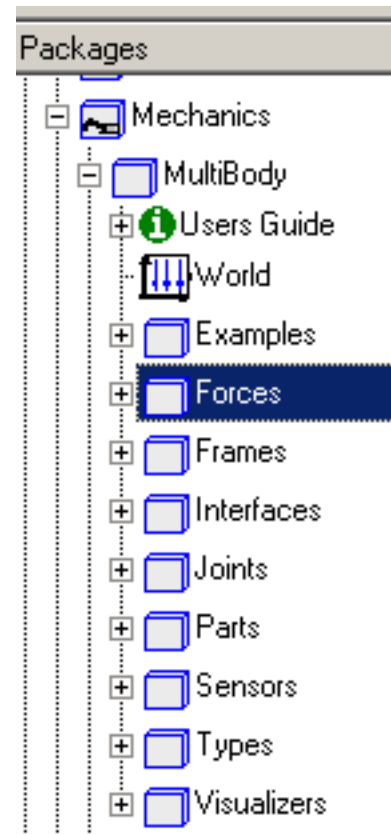
```
// import Modelica.Mechanics.MultiBody.Frames;
// translational kinematic differential equations resolved in local frame_a
v_a = Frames.resolve2(frame_a.R, der(frame_a.r_0));
a_a = der(v_a);

// rotational kinematic differential equations
w_a = Modelica.Mechanics.MultiBody.Frames.angularVelocity2(frame_a.R);
z_a = der(w_a);

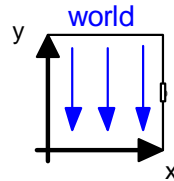
// Newton/Euler equations with respect to center of mass
a_CM = a_a + cross(z_a, r_CM) + cross(w_a, cross(w_a, r_CM));
f_CM = m*a_CM;
t_CM = I*z_a + cross(w_a, I*w_a);
frame_a.f = f_CM;
frame_a.t = t_CM + cross(r_CM, f_CM);
```



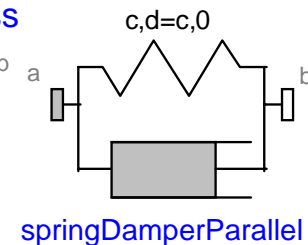
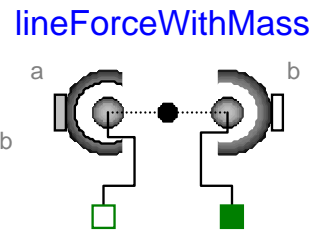
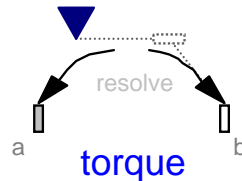
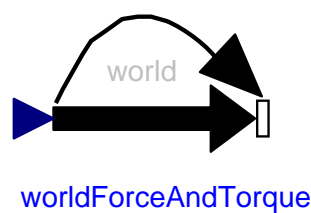
# Modelica Multibody Basics: Elementary Components I



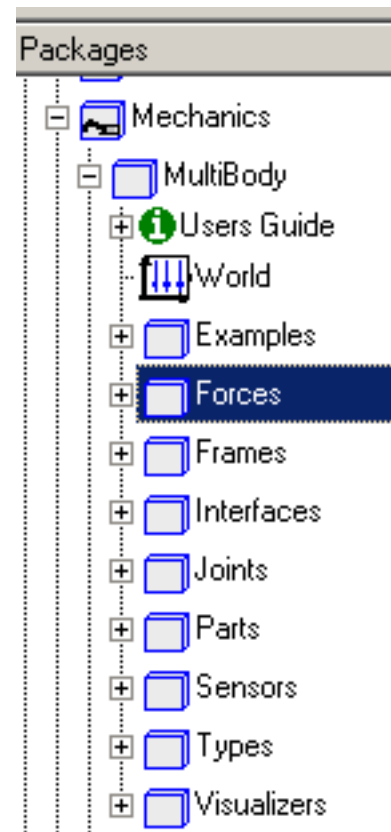
- Modelica.Mechanics.MultiBody.World
  - defines inertial frame, gravity, animation defaults



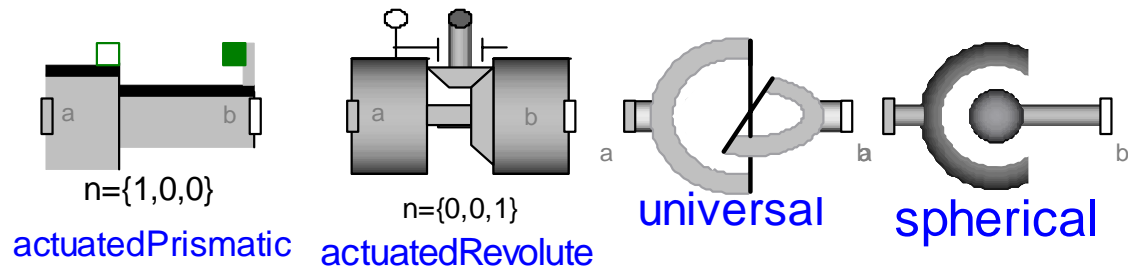
- Modelica.Mechanics.MultiBody.Forces
  - different resolution properties
  - interface to Real input functions and 1D mechanics
  - several spring/damper configurations



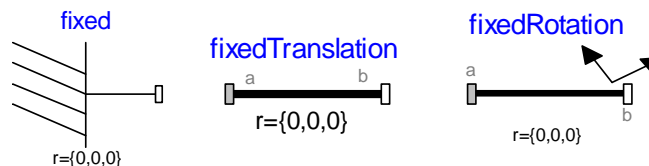
# Modelica Multibody Basics: Elementary Components II



- Modelica.Mechanics.MultiBody.Joints
  - define specific degree of freedom
  - capability to set-up initial configuration
  - interface to/for 1D mechanics and rheonom motion
  - e.g.:

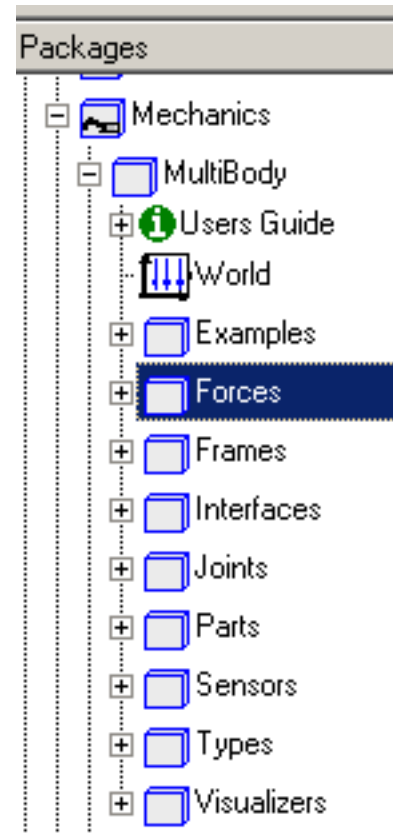


- Modelica.Mechanics.MultiBody.Parts
  - Fixed, FixedTranslation and FixedRotation



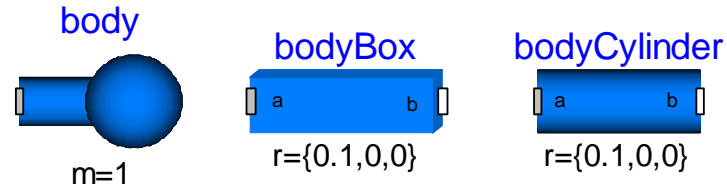


# Modelica Multibody Basics: Elementary Components III



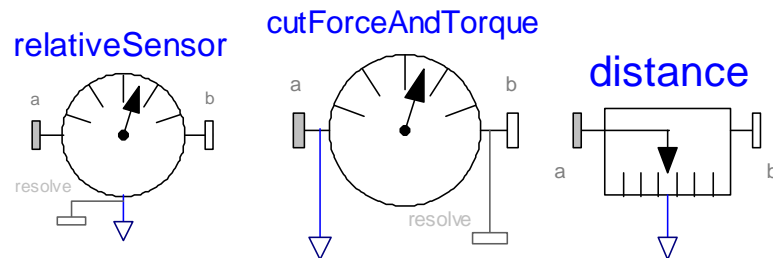
## ➤ Modelica.Mechanics.MultiBody.Parts

- Rigid bodies with predefined geometric shapes

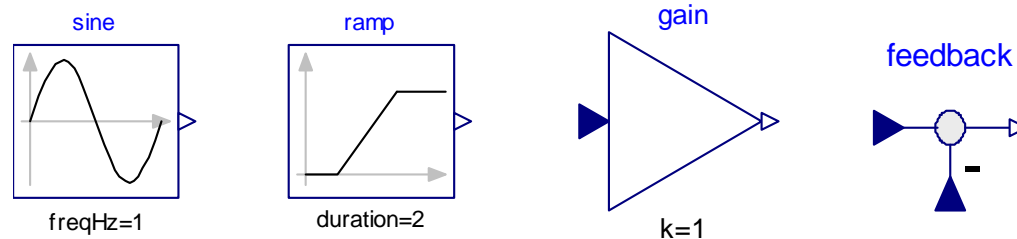


## ➤ Modelica.Mechanics.Multibody.Sensors

- for control and validation purposes



## ➤ Modelica.Blocks.Sources + Modelica.Blocks.Math



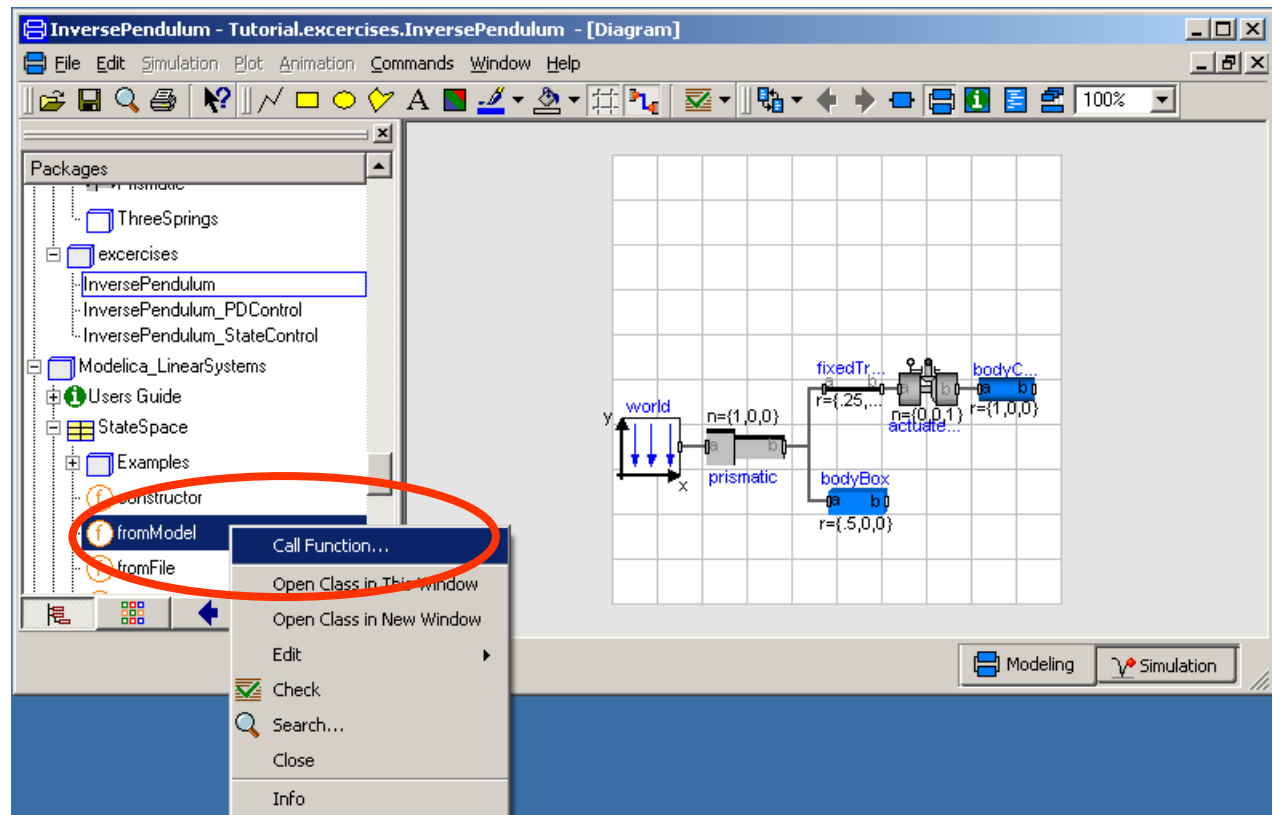
# Modelica Multibody Basics: Analysis Methods

- Model check
- Experiment setup, translation and time simulation

The screenshot displays the Modelica IDE interface for the 'InversePendulum' model. The top window, titled 'InversePendulum - Tutorial.exercises.InversePendulum - [Diagram]', shows the model's structure in the 'Packages' pane and a 'Messages - Dymola' window. The messages window indicates a successful check of the DAE system, reporting 1836 scalar unknowns and 1836 scalar equations. The bottom window, titled 'InversePendulum - Tutorial.exercises.InversePendulum', shows the simulation controls and the 'Experiment Setup' dialog. The simulation controls include a toolbar with a green checkmark icon circled in red, and a status bar showing 'Time: 0' and 'Speed: 1'. The 'Experiment Setup' dialog is open, showing the 'General' tab with the following settings: Experiment Name: 'InversePendulum', Simulation interval: Start time 0, Stop time 5, Output interval: Number of intervals 500, Integration: Algorithm 'Dassl', Tolerance 0.0001, and Fixed Integrator Step 0.

# Modelica Multibody Basics: Analysis Methods

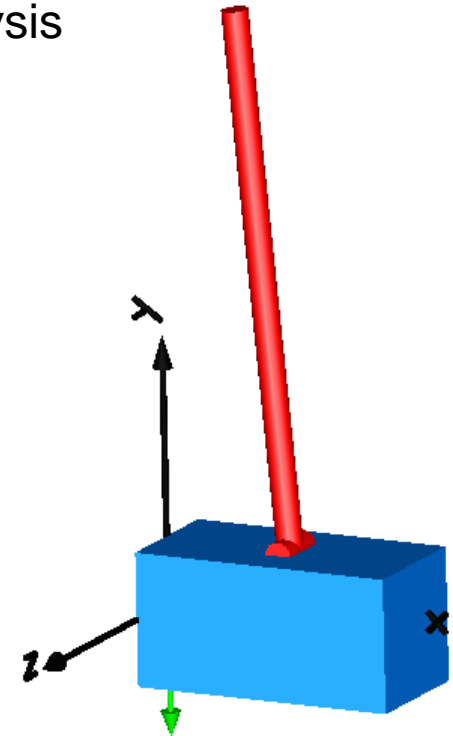
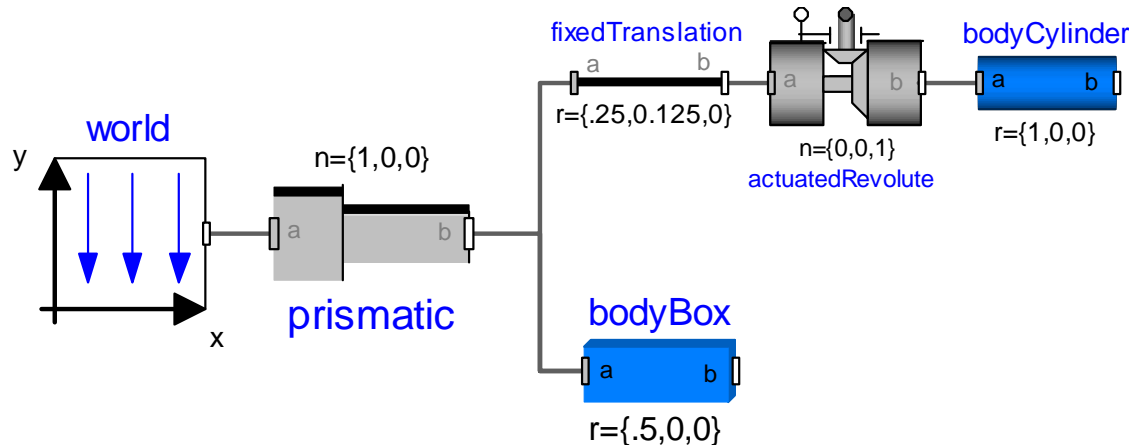
- Model check
- Experiment setup, translation and time simulation
- Eigenvalue analysis
  - Menu: File→Libraries→LinearSystems



# Example 1: Control of an inverse pendulum I

## ➤ Initial model

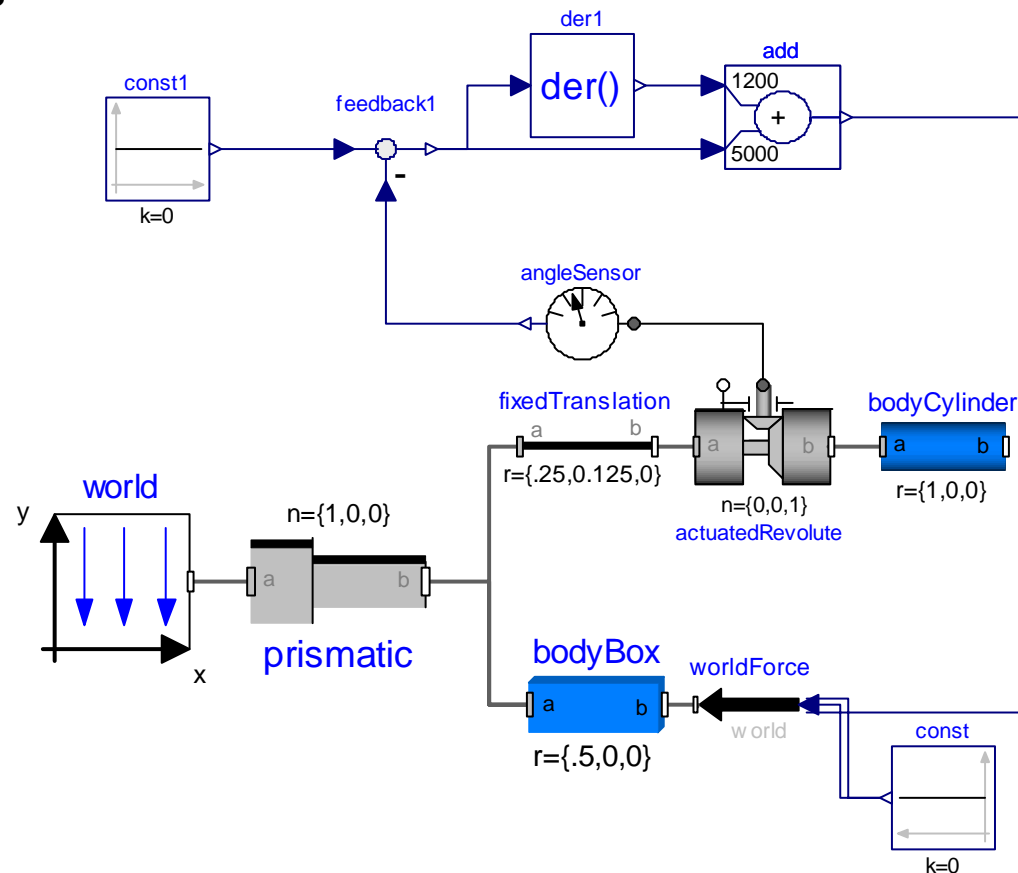
- Box: 0.5 x 0.25 x 0.25 m
- actuatedRevolute: 90° phi\_offset, 5° phi\_start
- perform time simulation and eigenvalue analysis



# Example 1: Control of an inverse pendulum II

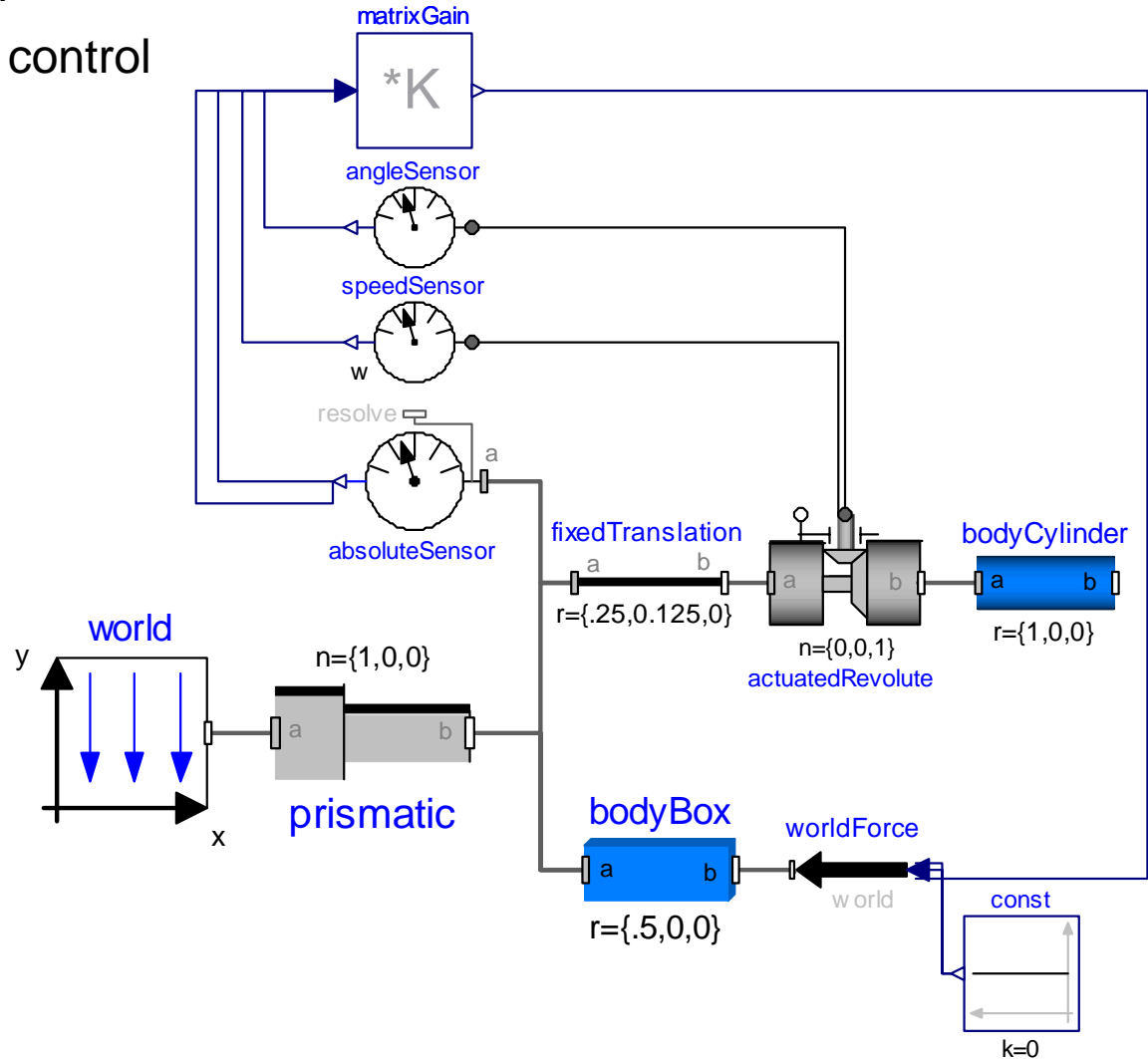
## ➤ PD-Control of angle only

- use extend to inherit initial model
- use transient response and pole placement to set-up controller gains



# Example 1: Control of an inverse pendulum III

- Improve control
  - e.g. state control



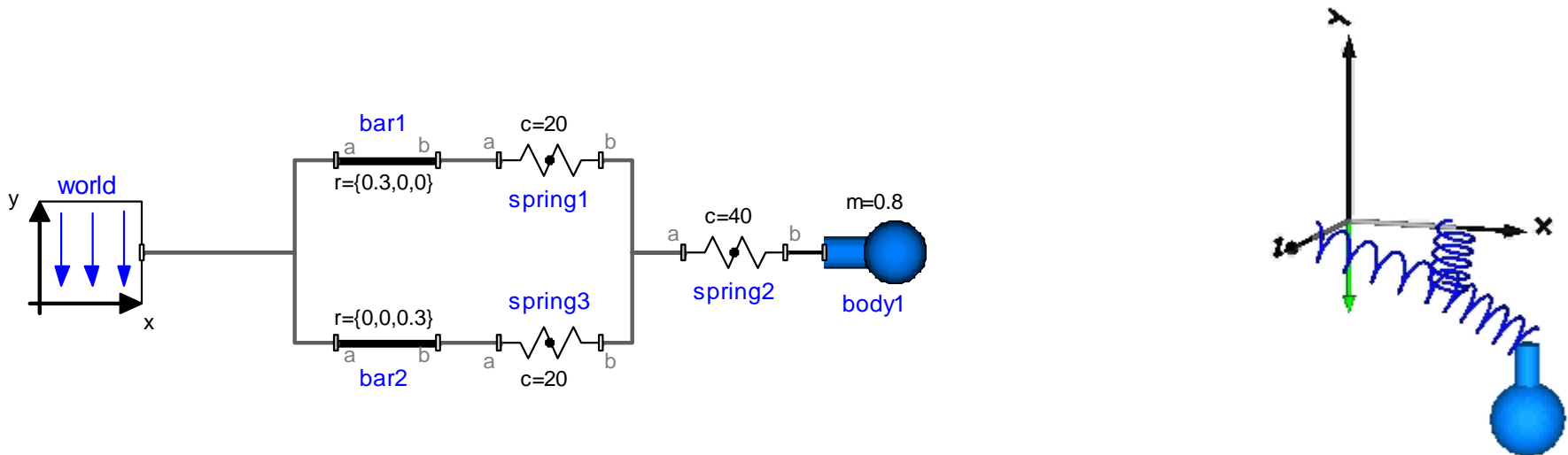
# Contents

- Introduction
  - Modelica, very briefly
- Modelica Multibody Basics
- Exercise 1: Control of an inverse pendulum
- Modelica Multibody Advanced
- Exercise 2: Hexapod
- FlexibleBodies Library: Beams
- Exercise 3: Aircraft Fin
- FlexibleBodies Livbrary: General bodies based on finite element data



# Modelica Multibody Advanced: State selection I

- Joints AND bodies have potential states
  - number of joints is independent from number of bodies
  - an assignment of joints to bodies is not mandatory
  - e.g.:

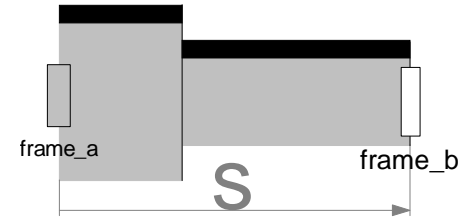


- here: body coordinates: position, quaternions and their derivatives are used as states

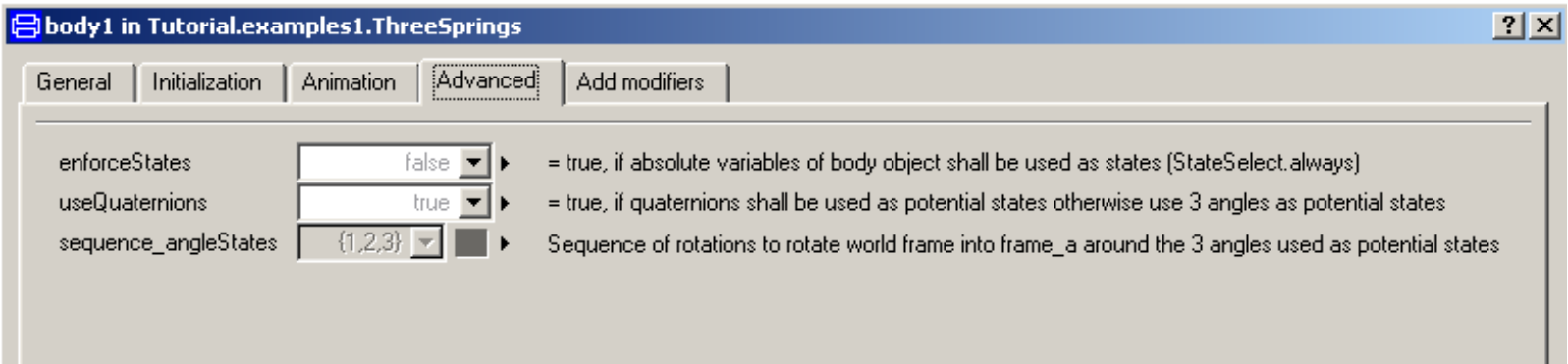


# Modelica Multibody Advanced: State selection II

- relative joint coordinates are used as states if possible
  - default: `stateSelect = StateSelect.prefer`
  - e.g. `Multibody.Joints.Prismatic`



```
final parameter Real e[3]=Modelica.Mechanics.MultiBody.Frames.normalize(n)
  "Unit vector in direction of prismatic axis n";
SI.Position s(stateSelect=if enforceStates then
  StateSelect.always else StateSelect.prefer)
  "Relative distance between frame_a and frame_b";
SI.Velocity v(stateSelect=if enforceStates then StateSelect.always else
  StateSelect.prefer) "First derivative of s (relative velocity)";
```

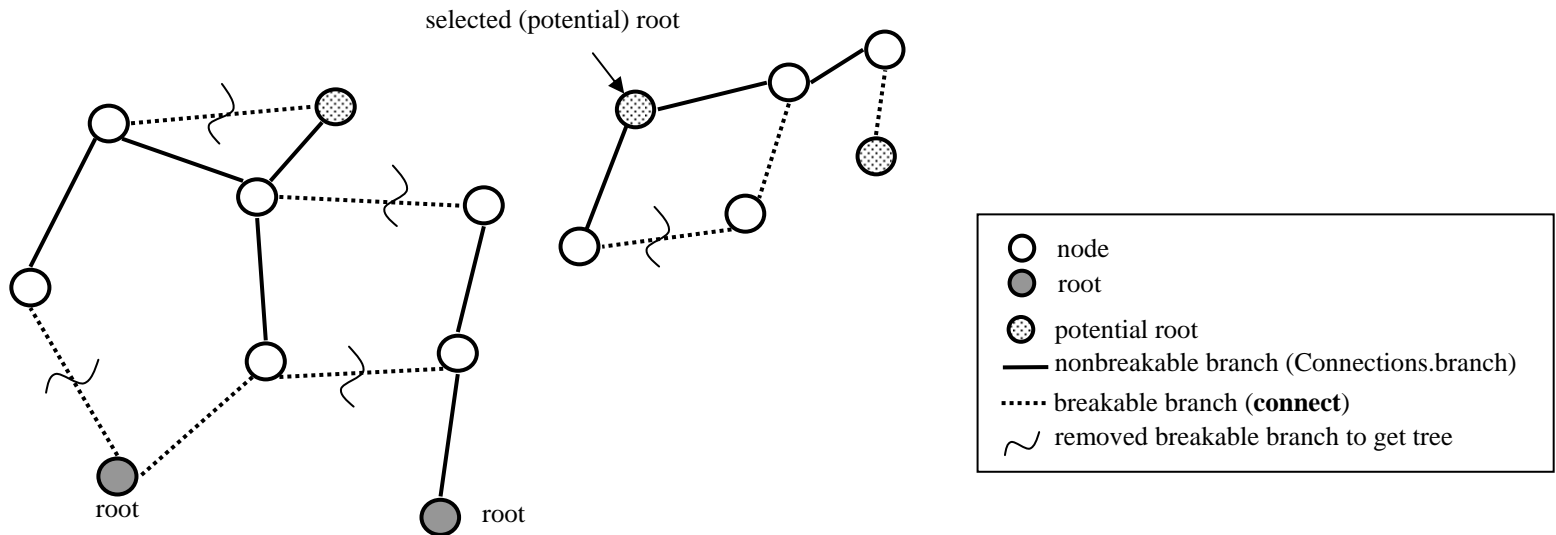


- Advanced user may influence state selection directly

# Modelica Multibody Advanced: Loops I

## ➤ Standard case

- no specific action by the user is required
- every connector is one node in the virtual connection graph
- roots of the virtual connection graph are found, e.g. world.frame\_b
- loops are virtually broken



# Modelica Multibody Advanced: Loops I

## ➤ Standard case

- no specific action by the user is required
- every connector is one node in the virtual connection graph
- roots of the virtual connection graph are found, e.g. world.frame\_b
- loops are virtually broken
- the related constraint equations are provided

⇒ DAE

$$0 = f(\dot{x}, x, y, t, \dots) \quad \dim(f) = \dim(x) + \dim(y)$$

- Equations are rearranged to get a sequence for model evaluation  
(**Block Lower Triangle**-partitioning)

$$\begin{array}{l} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{array} \begin{pmatrix} z_1 & z_2 & z_3 & z_4 & z_5 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} \Rightarrow \begin{array}{l} f_2 \\ f_4 \\ f_3 \\ f_5 \\ f_1 \end{array} \begin{pmatrix} z_1 & z_2 & z_3 & z_4 & z_5 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

# Modelica Multibody Advanced: Loops I

## ➤ Standard case

- no specific action by the user is required
- every connector is one node in the virtual connection graph
- roots of the virtual connection graph are found, e.g. world.frame\_b
- loops are virtually broken
- the related constraint equations are provided

⇒ DAE

$$0 = f(\dot{x}, x, y, t, \dots) \quad \dim(f) = \dim(x) + \dim(y)$$

- Equations are rearranged to get a sequence for model evaluation (**B**lock **L**ower **T**riangle-partitioning)
- Equations to be differentiated are determined (Pantelides algorithm)
- superfluous potential states are deselected dynamically (dummy derivative method) ⇒ ODE:

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

# Modelica Multibody Advanced: Loops II

- review Translation Log in order to streamline simulation performance with model adjustments

```
Messages - Dymola
Syntax Error | Translation | Dialog Error | Simulation | Version Manager
Translation of Tutorial.Exercises.Hexapod_Trajectory:
DAE having 6280 scalar unknowns and 6280 scalar equations.

STATISTICS

Original Model
Number of components: 347
Variables: 3072
  Constants: 13 (13 scalars)
  Parameters: 980 (2126 scalars)
  Unknowns: 2079 (6280 scalars)
  Differentiated variables: 118 scalars
Equations: 1626
Nontrivial: 1060

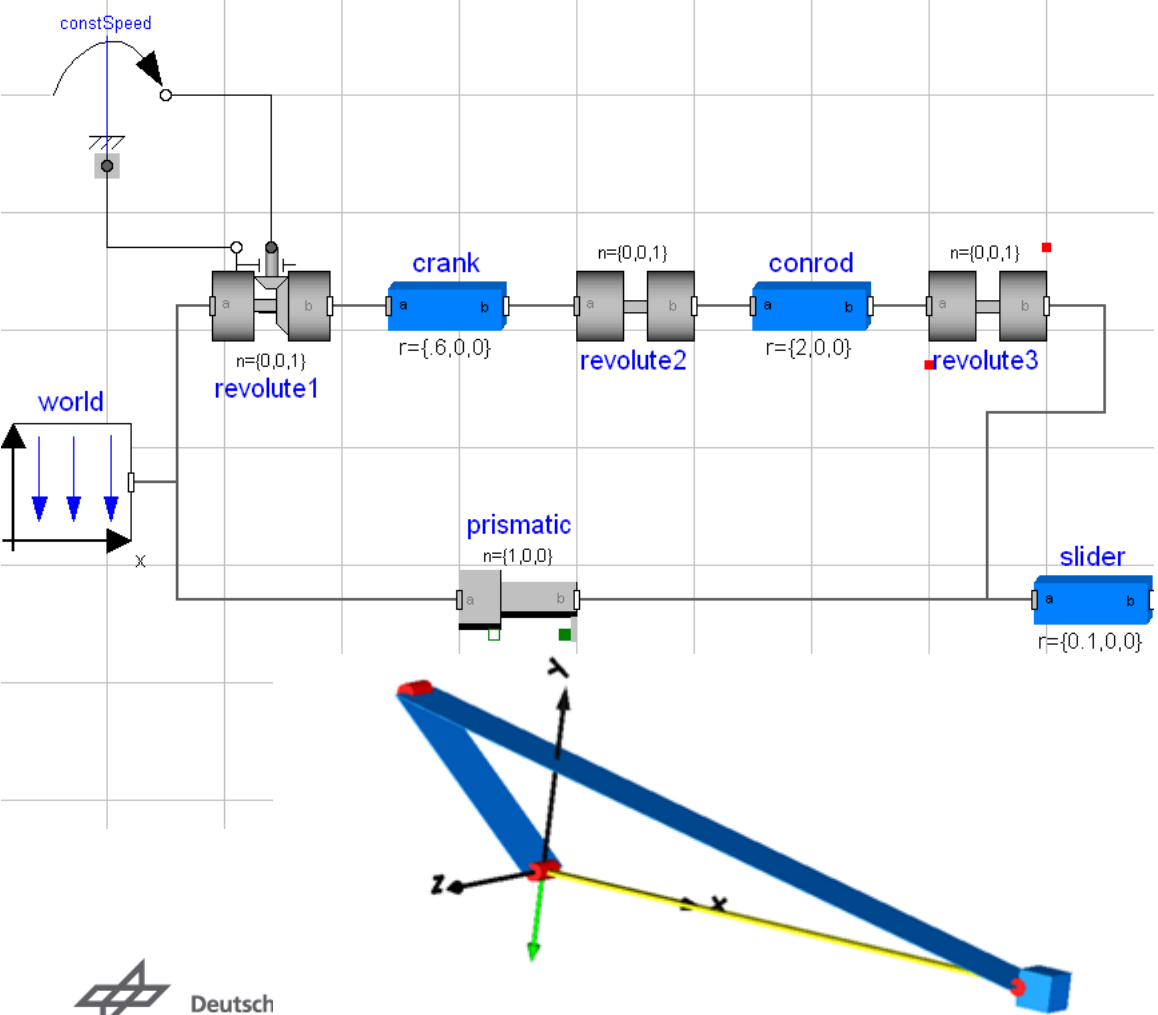
Translated Model
Constants: 4005 scalars
Free parameters: 343 scalars
Parameter depending: 1170 scalars
Inputs: 0
Outputs: 0
Continuous time states: 18 scalars
Time-varying variables: 875 scalars
Alias variables: 2063 scalars
Assumed default initial conditions: 12
  LogDefaultInitialConditions=true; gives more information
Number of mixed real/discrete systems of equations: 0
Sizes of linear systems of equations: {3, 4, 26, 26, 26, 26, 26, 3, 4, 26,
26, 26, 26, 26, 26, 8, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2}
Sizes after manipulation of the linear systems: {3, 4, 11, 11, 11, 11, 11, 11,
3, 4, 5, 5, 5, 5, 5, 5, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13}
Sizes of nonlinear systems of equations: {163}
Sizes after manipulation of the nonlinear systems: {34}
Number of numerical Jacobians: 0

Initialization problem
Sizes of linear systems of equations: {3, 4}
Sizes after manipulation of the linear systems: {3, 4}
Sizes of nonlinear systems of equations: {16, 71, 71, 71, 71, 71, 71}
Sizes after manipulation of the nonlinear systems: {4, 10, 10, 10, 10, 10, 10}
Number of numerical Jacobians: 0
```

# Modelica Multibody Advanced: Loops III

➤ Planar loops

➤ error message



Messages - Dymola

Syntax Error Translation Dialog Error Simulation

Translation of [Tutorial.Examples.SliderCrank](#):  
 Error: The problem is structurally singular:  
 It has 2234 scalar unknowns and 2234 scalar equations.  
 The Real part has 2162 unknowns and 2162 equations.  
 The Integer part has 72 unknowns and 72 equations.  
 The Boolean part has 0 unknowns and 0 equations.  
 The String part has 0 unknowns and 0 equations.

Attempting to further localize singularity.

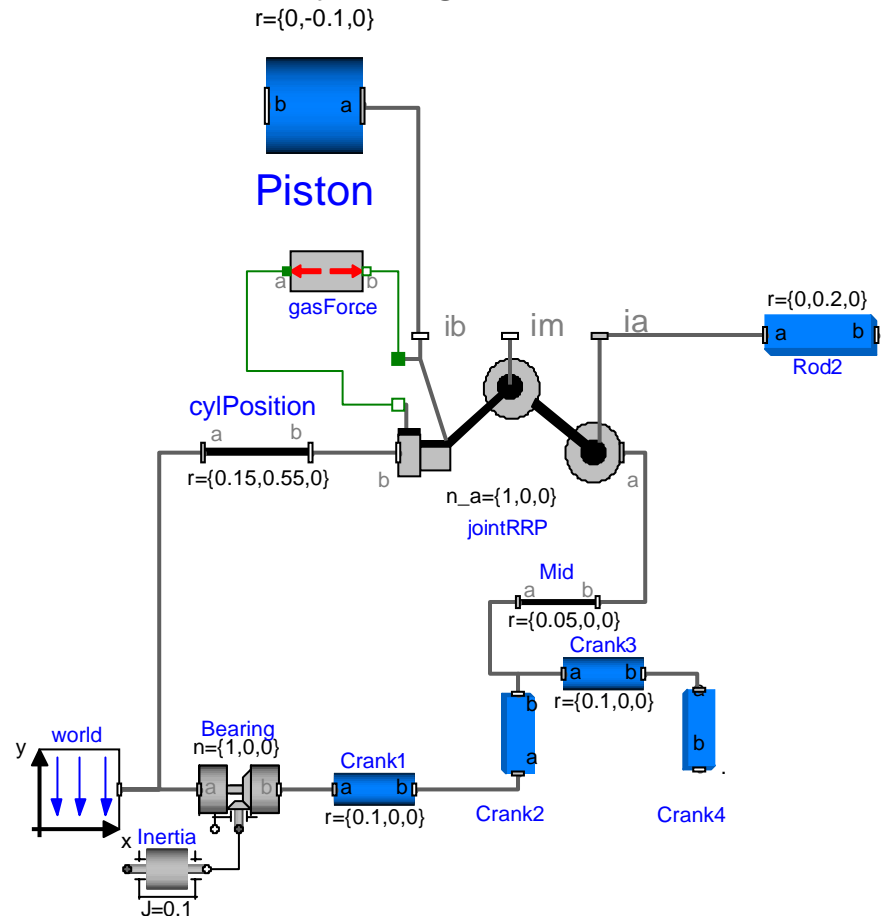
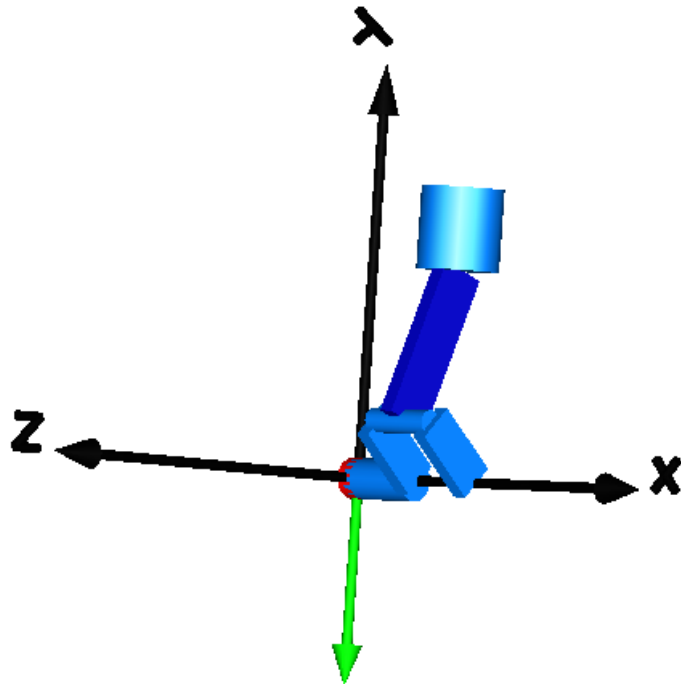
Singularity of [Tutorial.Examples.SliderCrank](#) is at the top level.  
 Error: The model [Tutorial.Examples.SliderCrank](#) is structurally singular.  
 The problem is structurally singular for the element type Real.  
 The number of scalar Real unknown elements are 2162.  
 The number of scalar Real equation elements are 2162.  
 The model includes the following hints:  
 All Forces cannot be uniquely calculated.  
 The reason could be that the mechanism contains a planar loop or that joints constrain the same motion. For planar loops, use in one revolute joint per loop the option `PlanarCutJoint=true` in the Advanced menu.

The problem is structurally regular for the element type Integer.  
 The number of scalar Integer elements are 72.  
 The problem has no elements of type Boolean.  
 The problem has no elements of type String.

Translation aborted.  
 ERROR: 2 errors were found

# Modelica Multibody Advanced: Loops IV

- Use of aggregated joint objects
  - to profit from analytical loop handling according to the „characteristic pair of joints“ method by the group of Prof. Hiller



# Modelica Multibody Advanced: Initialisation

- Initialisation default:
  - every state is assumed to be arbitrary unless otherwise provided
  - Newton solver starts with guess value zero in order to find consistent initial states unless otherwise provided
- If initialisation fails
  - determine, i.e. fix, characteristic variables/states in order to influence the system of equations to solve
  - provide „good“ guesses for initial states
  - be aware of singular positions, e.g. piston at bottom dead center
  - keep system of equations consistent

The screenshot shows a dialog box titled "Initialization" with the following fields and descriptions:

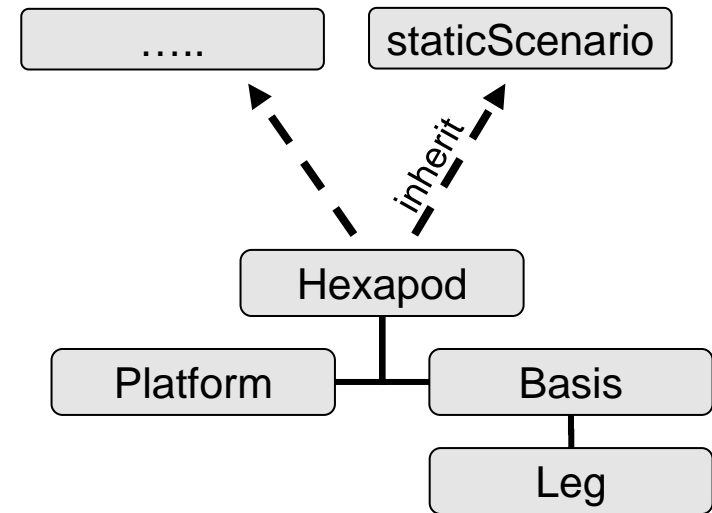
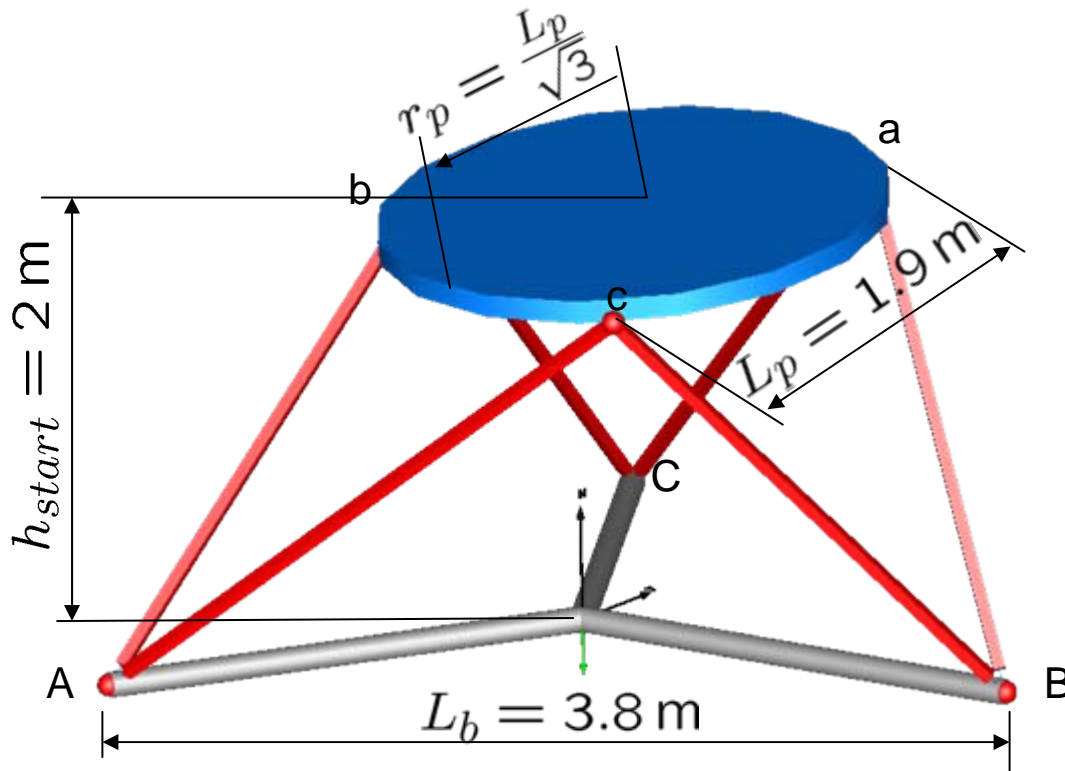
Field	Value	Unit	Description
initType	Modelica.Mechanics.MultiBody.Types.Init.PositionVelocity		Type of initialization (defines usage of start values below)
phi_start	5	deg	Initial value of rotation angle phi (fixed or guess value)
w_start	0	deg/s	Initial value of relative angular velocity $w = \text{der}(\text{phi})$
a_start	0	deg/s <sup>2</sup>	Initial value of relative angular acceleration $a = \text{der}(w)$

Buttons: OK, Info, Cancel



# Exercise 2: Hexapod I

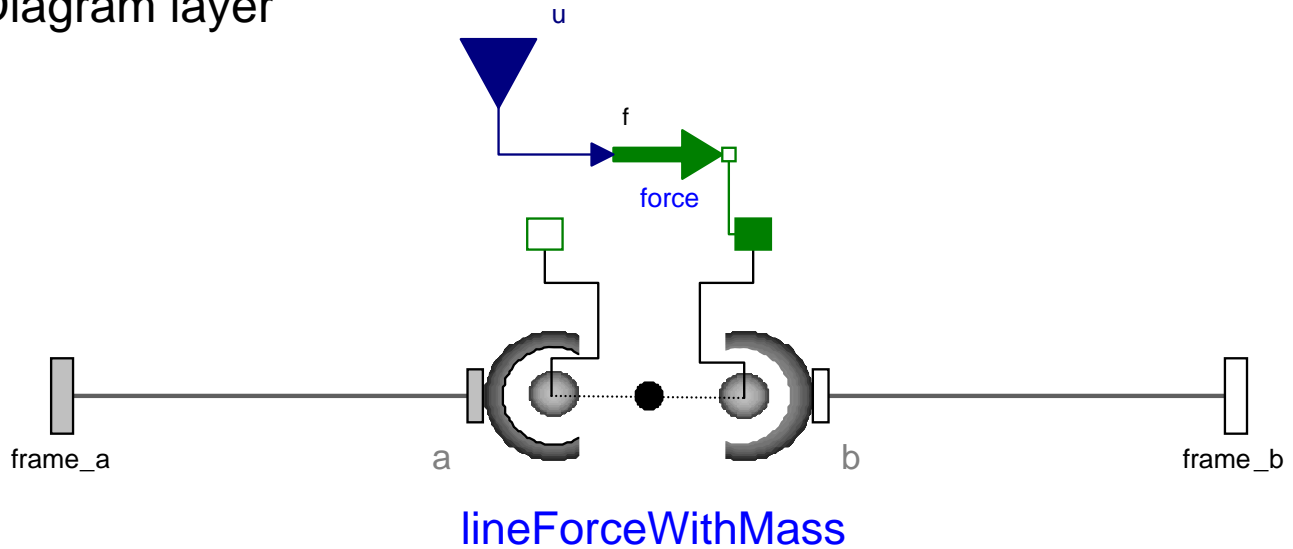
➤ The modelling concept



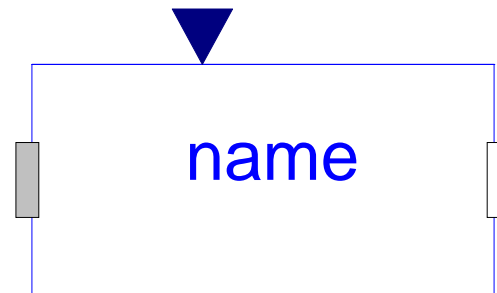
# Exercise 2: Hexapod II

➤ Model Leg

➤ Diagram layer



➤ Icon layer



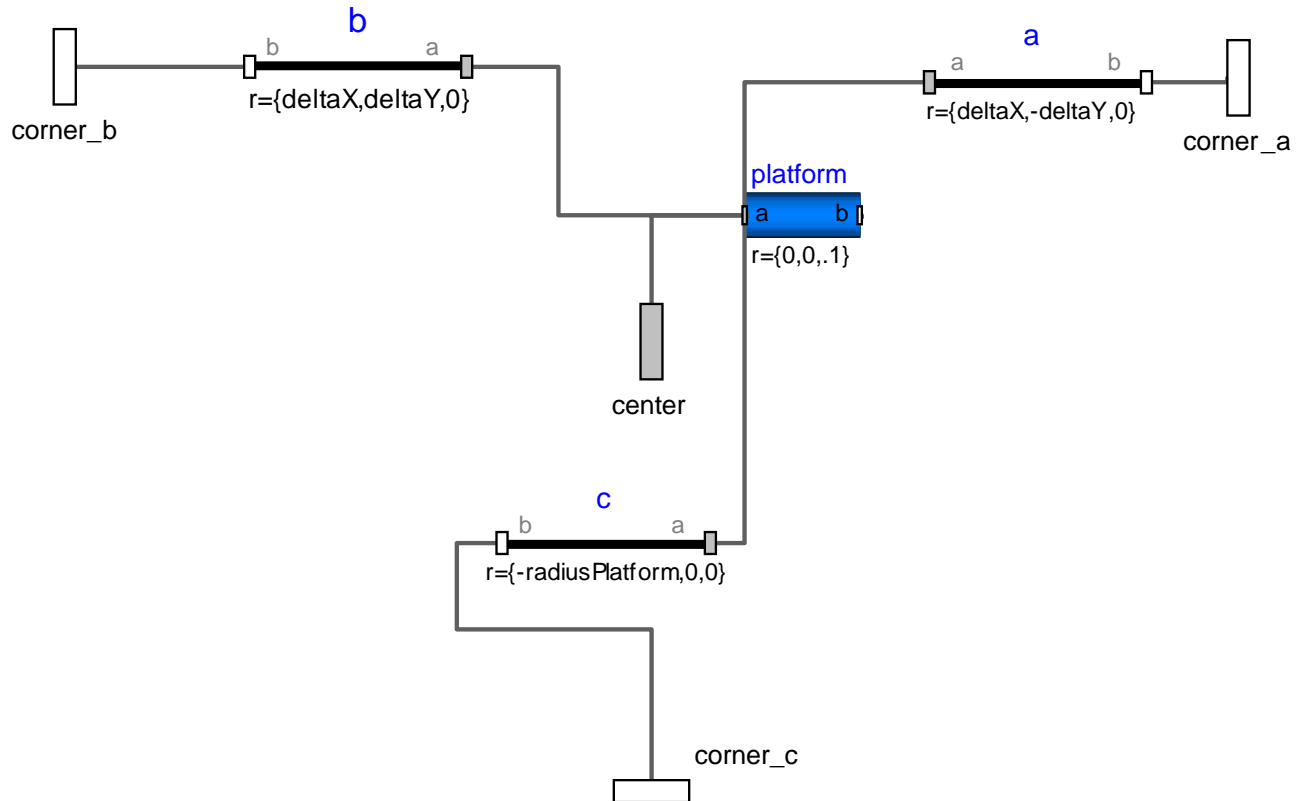
# Exercise 2: Hexapod III

## ➤ Model Platform

```

model Platform
  import SI = Modelica.SIunits;
  parameter SI.Length Lp=1.9 " length of a side of the platform";
  final parameter SI.Length radiusPlatform=Lp/sqrt(3);
  final parameter SI.Length deltaX=radiusPlatform/2;
  final parameter SI.Length deltaY=Lp/2;

```

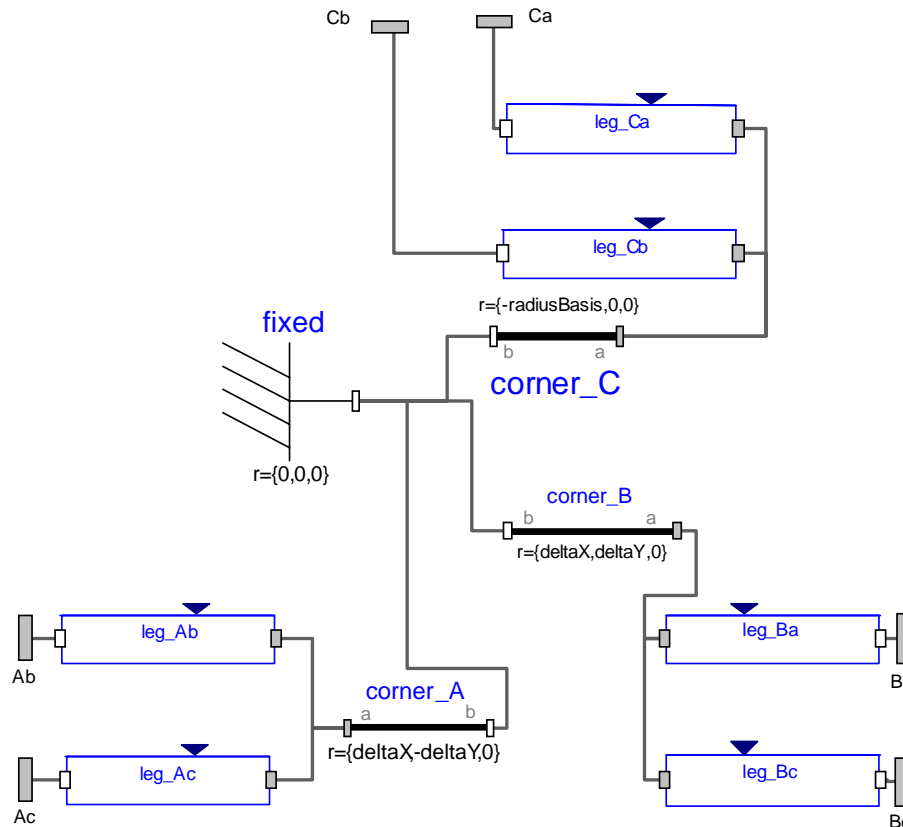


# Exercise 2: Hexapod IV

## ➤ Model Basis

```

model Basis
  import SI = Modelica.SIunits;
  parameter SI.Length Lb=3.8 " length of a side of the basis";
  final parameter SI.Length radiusBasis=Lb/sqrt(3);
  final parameter SI.Length deltaX=radiusBasis/2;
  final parameter SI.Length deltaY=Lb/2;
  
```

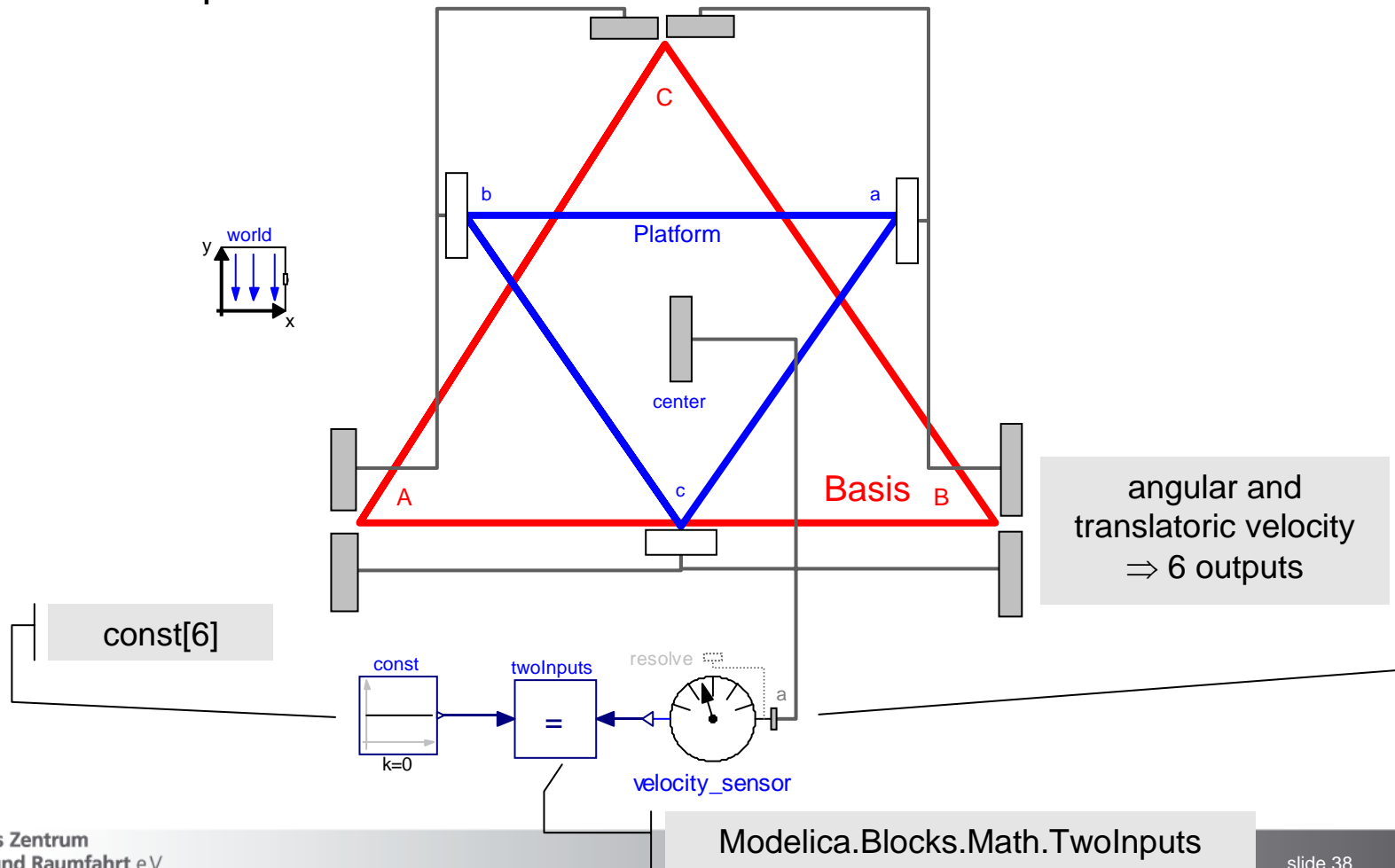


# Exercise 2: Hexapod V

## ➤ Hexapod\_StaticScenario

- actuator force in initial position ?

```
partial model Hexapod
import SI = Modelica.SIunits;
parameter SI.Length h_start=2;
```



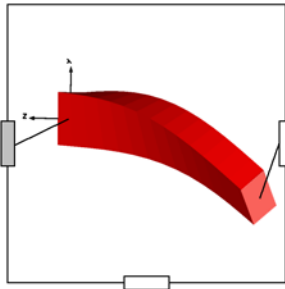
# Contents

- Introduction
  - Modelica, very briefly
- Modelica Multibody Basics
- Exercise 1: Control of an inverse pendulum
- Modelica Multibody Advanced
- Exercise 2: Hexapod
- **FlexibleBodies Library: Beams**
- **Exercise 3: Aircraft Fin**
- FlexibleBodies Livbrary: General bodies based on finite element data



# FlexibleBodies Library: Beams versus ModalBody

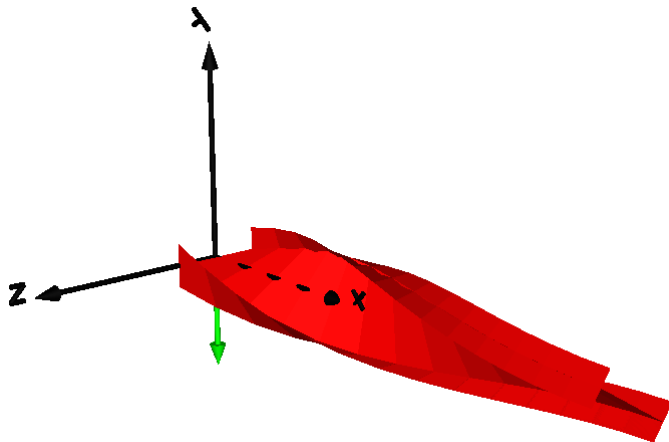
beam



Analytical beam description

Modelica generated SID

Animation uses beam description



## Common Issues

Floating frame of reference

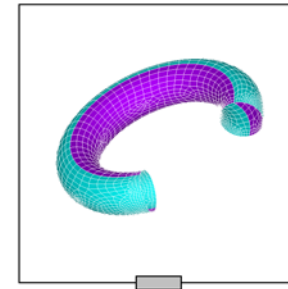
Equations of motion

SID-data-structure

Standard-Input-Data: Wallrapp '94

## Disjunctive Issues

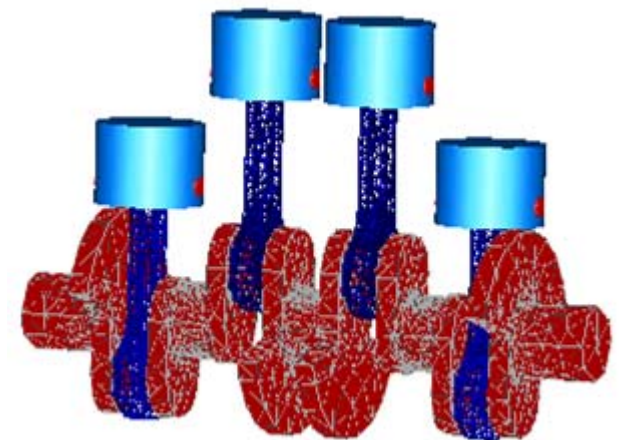
modalBody



FEM-based body description

External generated SID

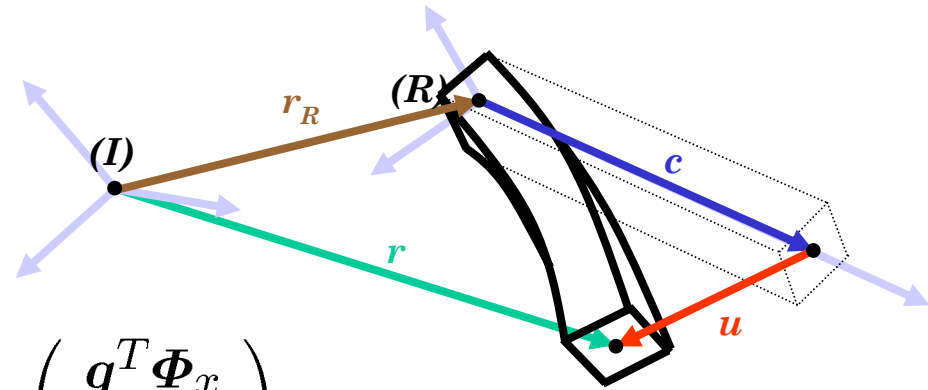
Animation based on external data



# FlexibleBodies Library: Equations of motion

- Floating frame of reference

$$\mathbf{r} = \mathbf{r}_R + \mathbf{c} + \mathbf{u}$$



- Modal approach

$$\mathbf{u}(\mathbf{c}, t) = \Phi(\mathbf{c}) \mathbf{q}(t) + \frac{1}{2} \begin{pmatrix} \mathbf{q}^T \Phi_x \\ \mathbf{q}^T \Phi_y \\ \mathbf{q}^T \Phi_z \end{pmatrix} \mathbf{q}$$

- Equations of motion

$$\begin{pmatrix} m\mathbf{I}_3 & & \text{sym.} \\ m\tilde{\mathbf{d}}_{CM} & \mathbf{J} & \\ \mathbf{C}_t & \mathbf{C}_r & \mathbf{M}_e \end{pmatrix} \begin{pmatrix} \mathbf{a}_R \\ \boldsymbol{\alpha}_R \\ \ddot{\mathbf{q}} \end{pmatrix} = \mathbf{h}_\omega - \begin{pmatrix} 0 \\ 0 \\ \mathbf{K}_e \mathbf{q} + \mathbf{D}_e \dot{\mathbf{q}} \end{pmatrix} + \begin{pmatrix} \mathbf{f}_a \\ \mathbf{f}_\alpha \\ \mathbf{f}_q \end{pmatrix}$$

Bremer/Pfeiffer '92, Schwertassek/Wallrapp '99



# FlexibleBodies Library: Beam theory

➤ 2<sup>nd</sup> order displacement field

➤ bending in xy- und xz- plane, torsion and lengthening

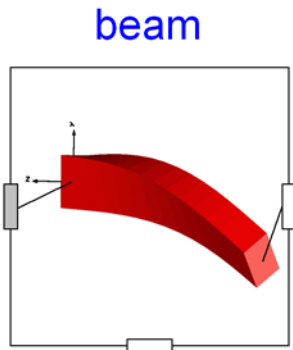
$$\mathbf{u}(x, t) = \begin{pmatrix} u \\ v \\ w \end{pmatrix} + \begin{pmatrix} -\frac{1}{2} \int_0^x v'^2 + w'^2 dx \\ -\int_0^x \int_0^{\bar{x}} \theta w'' d\bar{x} d\bar{x} + \int_0^x u'v' d\bar{x} \\ \int_0^x \int_0^{\bar{x}} \theta v'' d\bar{x} d\bar{x} + \int_0^x u'w' d\bar{x} \end{pmatrix}$$

➤ Raleigh-Ritz-approach for a straight, homogenous, isotropic beam with constant cross section

➤ expansion with analytic eigenvalue-solutions of the Euler-Bernoulli beam

$$v(x, t) = \Phi_v(x) \mathbf{q}_v(t) \quad \Phi_i = \begin{pmatrix} \cosh(\tau_i x) \\ \sinh(\tau_i x) \\ \cos(\tau_i x) \\ \sin(\tau_i x) \end{pmatrix}^T \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}_i$$

# FlexibleBodies Library: Beam menu set-up I



**data in FlexibleBodies.Examples.Demo**

General Add modifiers

Component

Name data

Comment

Model

Path FlexibleBodies.Interfaces.BeamData

Comment

Parameters

crossSection I\_beam

l 1.58 m length of beam

rho 7850 kg/m<sup>3</sup> mass density

E 2.1e11 N/m<sup>2</sup> Young's modulus

G E/(2\*(1 + 0.3)) N/m<sup>2</sup> Shear modulus

xsi {5} specification of int

Eigenmodes

bending\_xy ryConditionB=FlexibleBodies.Types.BoundaryCondition.F

bending\_xz ryConditionB=FlexibleBodies.Types.BoundaryCondition.F

torsion ryConditionB=FlexibleBodies.Types.BoundaryCondition.F

lengthening ryConditionB=FlexibleBodies.Types.BoundaryCondition.F

Parameters

crossSection I\_beam

l 1.58 m

rho 7850 kg/m<sup>3</sup>

E 2.1e11 N/m<sup>2</sup>

G E/(2\*(1 + 0.3)) N/m<sup>2</sup>

xsi {5}

Eigenmodes

bending\_xy T\_beam

bending\_xz T\_beam

general

Icon

BeamData

**I\_beam crossSection**

I\_beam

Description

I-profile cross section

Inputs

width .02 m outer contour dimension in y-direction (along flange)

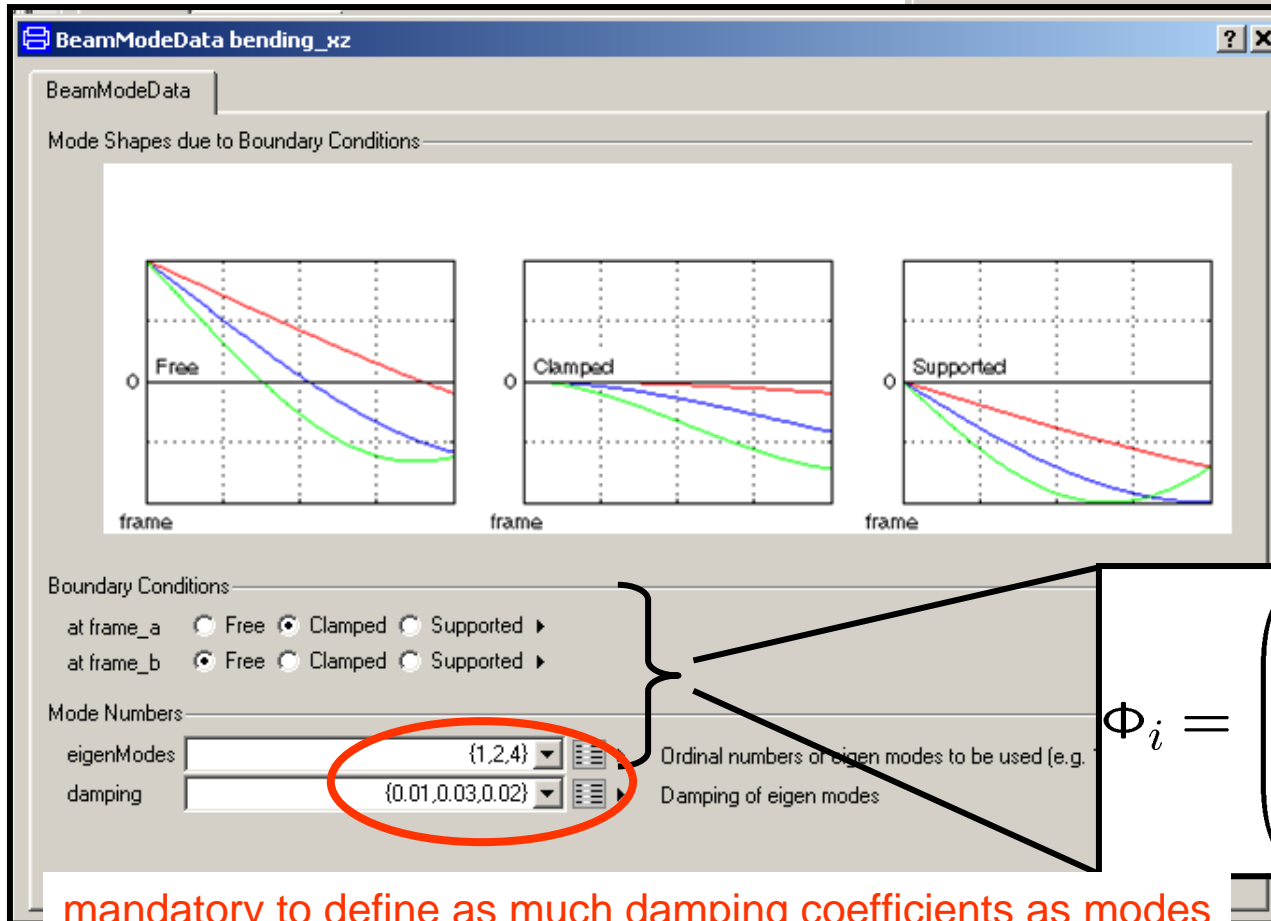
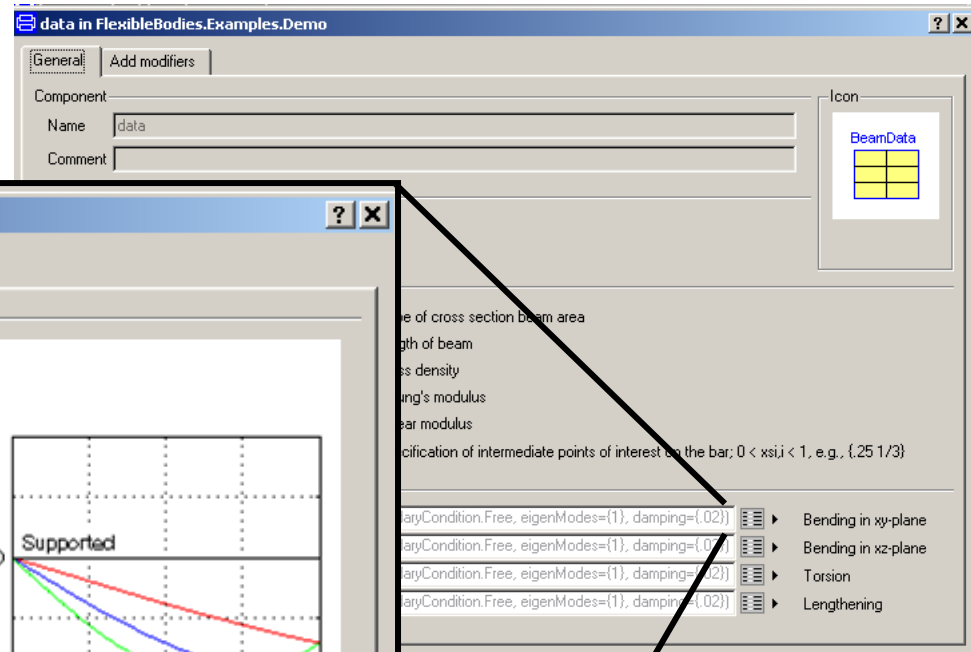
t\_bar 0.001 m thickness of central bar

height .03 m outer contour dimension in z-direction

t\_flange 0.001 m thickness of flanges

OK Info Close

# FlexibleBodies Library: Beam menu set-up II



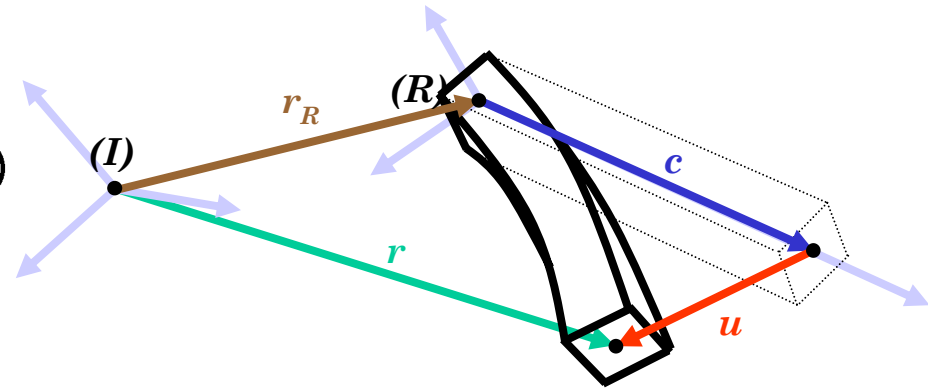
$$\Phi_i = \begin{pmatrix} \cosh(\tau_i x) \\ \sinh(\tau_i x) \\ \cos(\tau_i x) \\ \sin(\tau_i x) \end{pmatrix}^T \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}_i$$

mandatory to define as much damping coefficients as modes

# FlexibleBodies Library: Boundary Conditions I

- Mechanical interpretation

$$\mathbf{r}(\mathbf{c}, t) = \mathbf{r}_R(t) + \mathbf{c} + \mathbf{u}(\mathbf{c}, t)$$

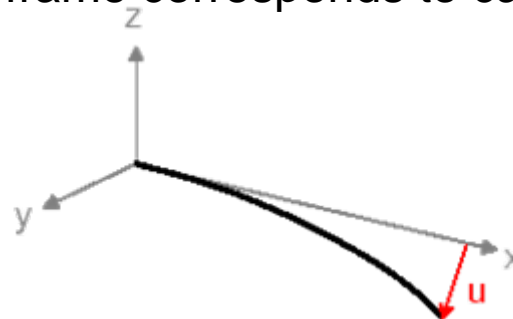


- let's say: frame of reference is pinned at frame\_a with  $\mathbf{c} = \mathbf{0}$

⇒ motion of frame\_a is completely described by  $\mathbf{r}_R(t)$   
( and related orientation)

$$\Rightarrow \mathbf{u}(\mathbf{c} = \mathbf{0}, t) = \mathbf{0} \quad \frac{\partial \mathbf{u}}{\partial \mathbf{c}}(\mathbf{c} = \mathbf{0}, t) = \mathbf{0}$$

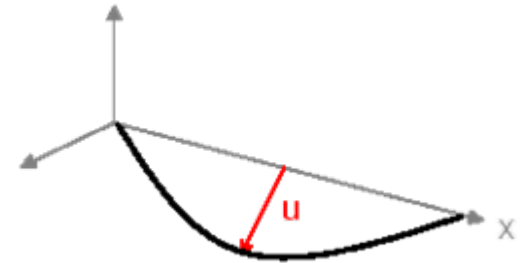
- clamped-free: tangent frame corresponds to cantilever beam boundary conditions



# FlexibleBodies Library: Boundary Conditions II

- supported-supported: chord frame

$$u(c = 0, t) = 0 \quad u(c = (l, 0, 0), t) = 0$$



- free-free: Buckens frame

- linear and angular momentum due to body deformation are zero

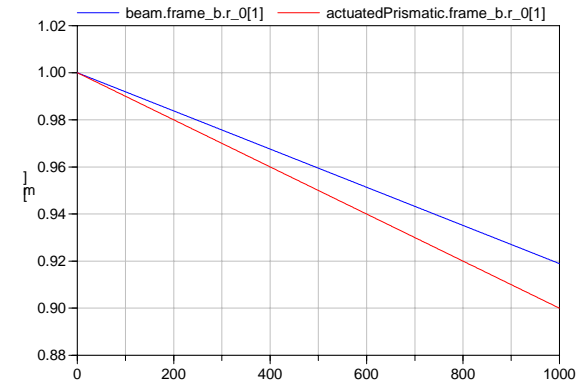
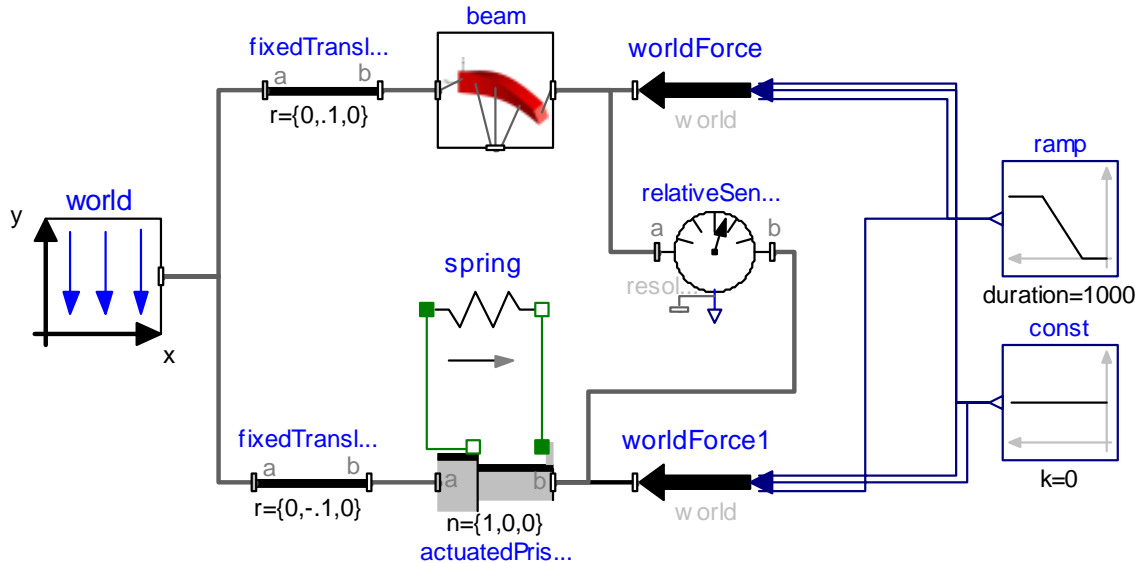
- every combination of tangent, chord and Buckens frames in different spatial directions is possible

- General rule

- align the boundary conditions with the degree of freedom of the joints to which the beam is attached
  - boundary conditions are related to constraint forces
  - a joint cannot transmit a constraint force in the direction of its motion
- BUT: Boundary conditions are validation issues

# FlexibleBodies Library: A classic pitfall I

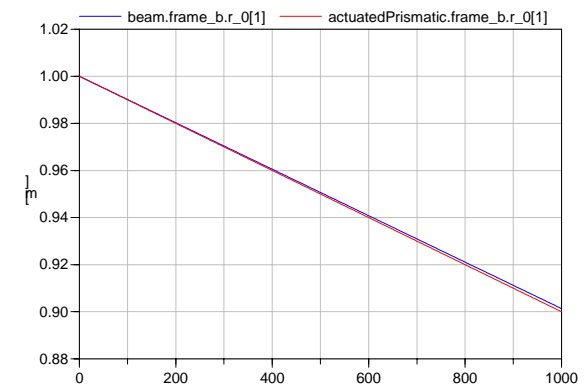
➤ static deflection: thrust force shortens beam and equivalent spring



1 eigenmode

	spring	beam	error
1 eigenmode	-10 cm	-8.1 cm	19 %
5 eigenmodes	-10 cm	-9.6 cm	4 %
10 eigenmodes	-10 cm	-9.8 cm	2 %
15 eigenmodes	-10 cm	-9.9 cm	1 %

comparison: deflections at the end



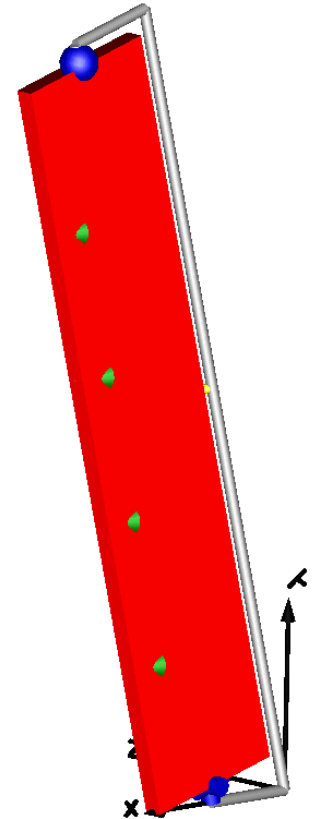
15 eigenmodes

# FlexibleBodies Library: A classic pitfall II

- Mechanical background
  - static deflections rely on elastic properties only
  - eigenmodes consider elastic and inertia properties
    - that's why they are well suited for dynamic problems
- Geometrical background
  - analytically:  $u = c \cdot x$
  - expansion with eigenmodes:  $u = \sin(\frac{2x}{\pi l}) + \sin(\frac{2x}{3\pi l}) + \dots$
- It is proven that Raleigh-Ritz approach converges against true value
  - but how fast ?
  - this is an extreme example, e.g. bending is less sensitive
- Check whether a higher number of modes changes results !

# Example 3: AircraftFin

- fin
  - mounted in a frame with  $15^\circ$  inclination
  - profil: squarepipe  $0.4 \times 0.05 \times 0.01$  m
  - 2 m long
  - $2730 \text{ kg/m}^3$ ,  $7 \cdot 10^{10} \text{ N/m}^2$  (aluminium)
  - 1 xz-bending and 1 torsion mode
  - 5 mid-nodes  $\xi = \{0.2, 0.4, 0.5, 0.6, 0.8\}$
- actuator to turn the fin  
(force as cos-function of t, 1 Hz, 10 N amplitude)
- 4 constant forces, 10 N in world-x-direction,  
to represent flow forces

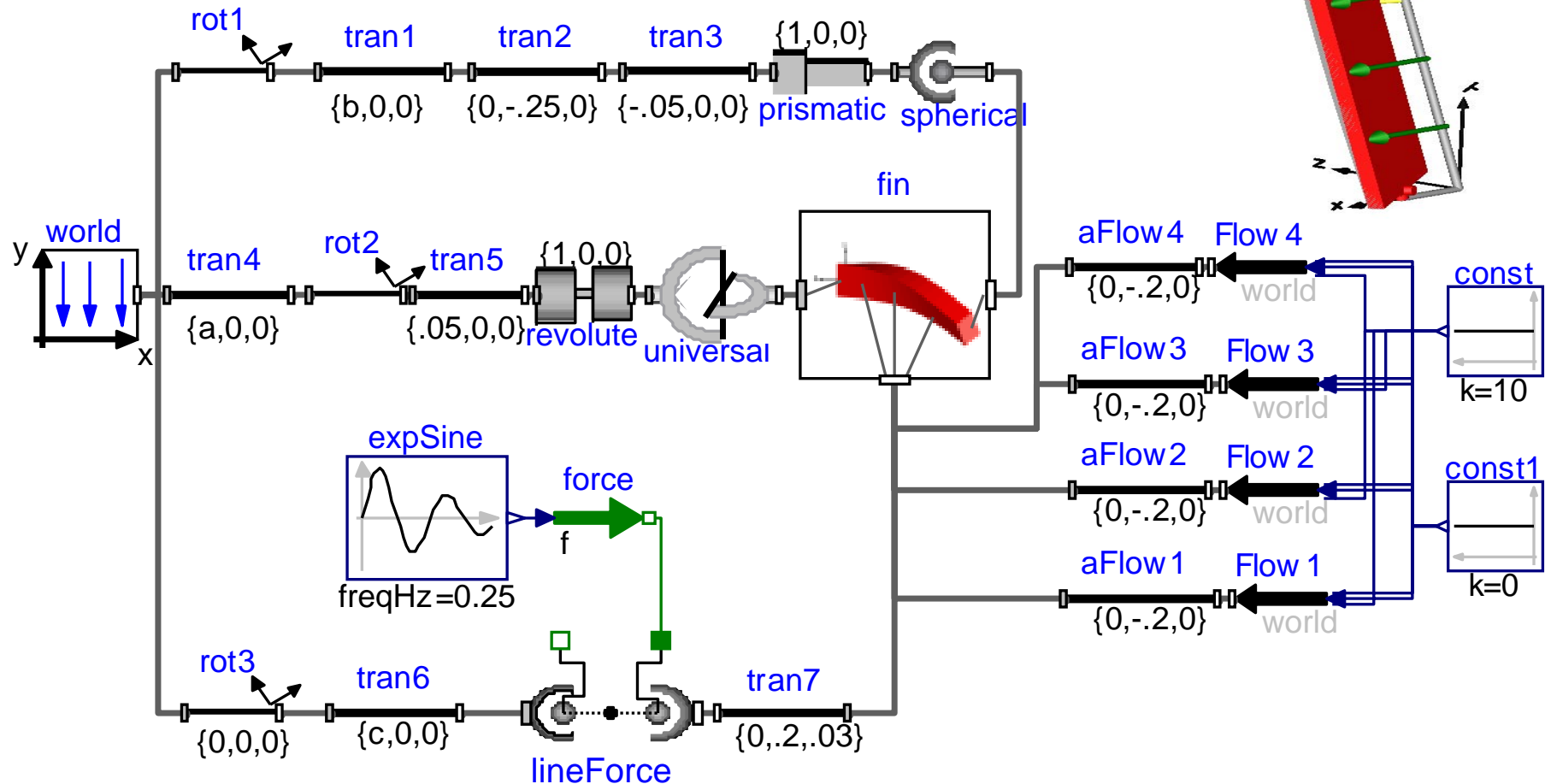
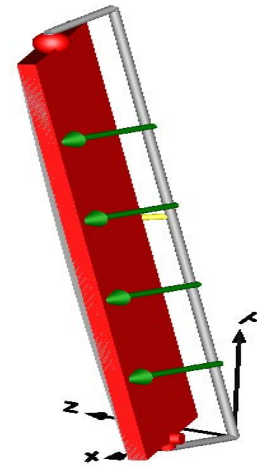


```
model AircraftFin
```

```
import Modelica.Constants.pi;  
parameter Real a= 0.25/cos(pi/12);  
final parameter Real b=a*sin(pi/12)+2.1;  
final parameter Real c=a*sin(pi/12)+1.05;
```



# Example 3: AircraftFin



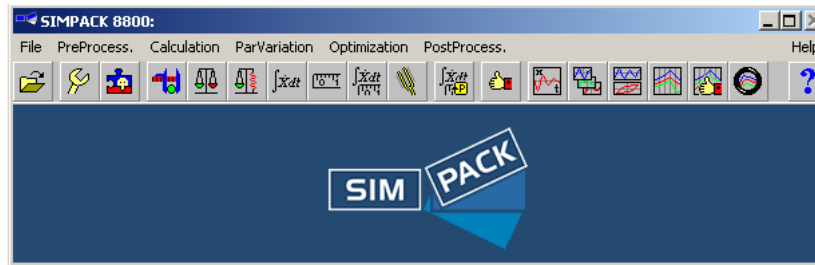
# Contents

- Introduction
  - Modelica, very briefly
- Modelica Multibody Basics
- Exercise 1: Control of an inverse pendulum
- Modelica Multibody Advanced
- Exercise 2: Hexapod
- FlexibleBodies Library: Beams
- Exercise 3: Aircraft Fin
- FlexibleBodies Library: General bodies based on finite element data



# FlexibleBodies Library: FEM-preprocessing

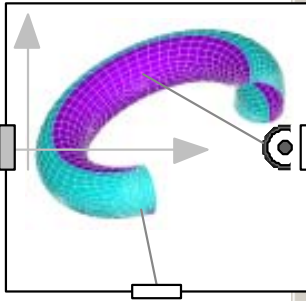
- Simpack-FEMBS: FEM to multibody system preprocessor
- Maintained and distributed by INTEC GmbH, Oberpfaffenhofen



- Supports ABAQUS, ANSYS, MSC.Nastran, NX Nastran, I-DEAS, PERMAS
- Reduction of the FE-model in 2 steps
  - in FE-tool, e.g. Guyan- or Craig-Bampton-method
    - ⇒ system matrices, nodes to retain, eigenmodes, mesh information
  - in Simpack-FEMBS
    - ⇒ modes selection (and generation), multibody description, animation data
- SID- and wavefront-file as results

# FlexibleBodies Library: ModalBody menu set-up I

modalBody



modalBody in FlexibleBodies.Examples.ModalBodies.Internal.CylinderElasticRodPiston

General Initialization Advanced Add modifiers

Component

Name modalBody

Comment

Icon

ModalBody

Model

Path FlexibleBodies.ModalBody

Comment General flexible body model based on a modal description

Parameters

SID\_fileName FlexibleBodies.Utilities.DataDirectory + "rodV2.SID\_FEM" File name of SID file describing the flexible body dynamics

WavefrontFile FlexibleBodies.Utilities.DataDirectory + "rodV2small.obj" File name of wavefront file describing the flexible body animation

Simulation nodes (= subset of finite element nodes) associated with connectors nodes\_sphericalJoint and nodes\_clamped

sphericalJointNodes {20001,20002} Simulation nodes of nodes\_sphericalJoint (= do not constrain rotation)

clampedNodes fill(0, 0) Simulation nodes of nodes\_clamped (= constrain translation and rotation)

Animation

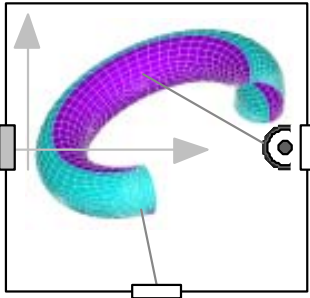
Solid animation  Exaggeration factor to visualize deformation 1 Color {0,0,255} Specular coefficient 0.7

Wire frame animation  Exaggeration factor to visualize deformation 200 Color {155,155} Specular coefficient 0

OK Info Cancel

# FlexibleBodies Library: ModalBody menu set-up II

modalBody



Variables

Variables	Values	Unit
ShowSimulationNodes 1		
ShowSimulationNodes 2		
world		
modalBody		

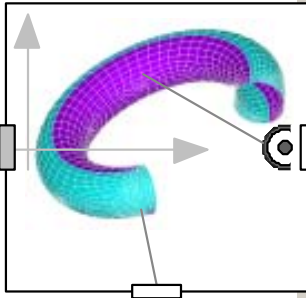
Selected object: world.frame\_b\_r\_0[1]  
Selected object: modalBody.Animation.Form  
Simulation Node: 20001

```
Hidden.ShowSimulationPoints = true
```

Modeling Simulation

# FlexibleBodies Library: ModalBody menu set-up II

modalBody



modalBody in Tutorial.Internal.testmodalBody

General Initialization **Advanced** Add modifiers

enforceStates  true ▶ = true, if elastic joint coordinates shall be used as states

useQuaternions  true ▶ = true, if quaternions shall be used as potential states otherwise use 3 angles as potential states

sequence\_angleStates  {1,2,3} ▶ Sequence of rotations to rotate world frame into frame\_a around the 3 angles used as potential states

Nominal

q\_nominal  1.e-6 ▶ Nominal values of generalized coordinates (for numerical scaling)

qd\_nominal  1.e-4 ▶ Nominal values of generalized velocities (for numerical scaling)

Structural damping

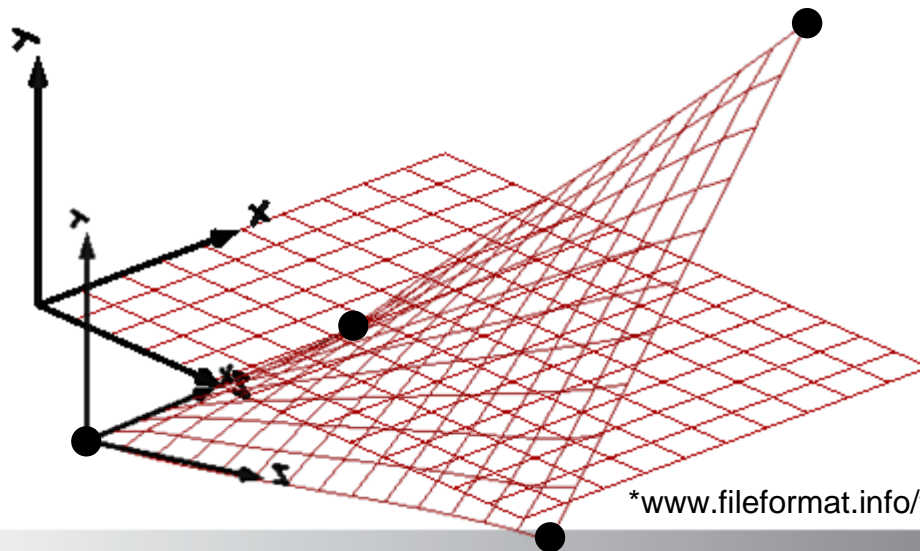
userDamping  false ▶ = true, if structural damping parameters shall be redefined (and taken instead of values in the SID-file)

dampingCoefficients  fill(0.02, nelastq) ▶ Natural damping coefficients

OK Info Cancel

# FlexibleBodies Library: Animation

- Advantage of the modal approach:
  - only few geometrical information is needed  $\Rightarrow$  efficiency !
- Disadvantage of the modal approach:
  - only few geometrical information is needed  $\Rightarrow$  animation ?
- Simulation points ● versus animation points
  - new animation feature in Dymola for wavefront data\*



\*[www.fileformat.info/format/wavefrontobj/egff.htm](http://www.fileformat.info/format/wavefrontobj/egff.htm)

# FlexibleBodies Library: 4 Cylinder Engine

## ➤ FEM

➤ Crankshaft 85 342 nodes

➤ Piston rod 12531 nodes

## ➤ Multibody representation

➤ < 1900 Hz

### ➤ Crankshaft

➤ 2 torsion eigenmodes

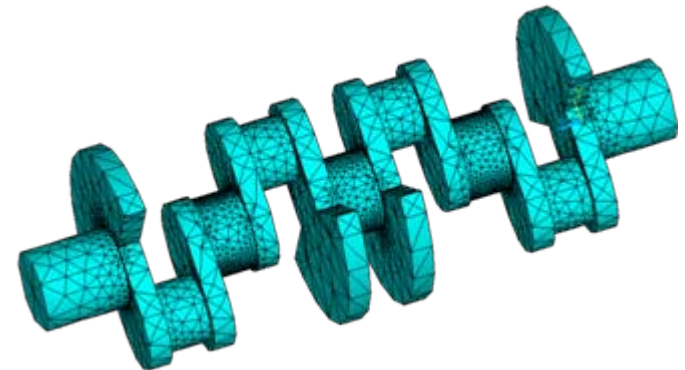
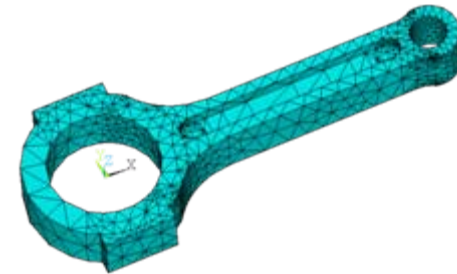
➤ 273 simulation nodes

### ➤ Piston rod

➤ 4 eigenmodes each

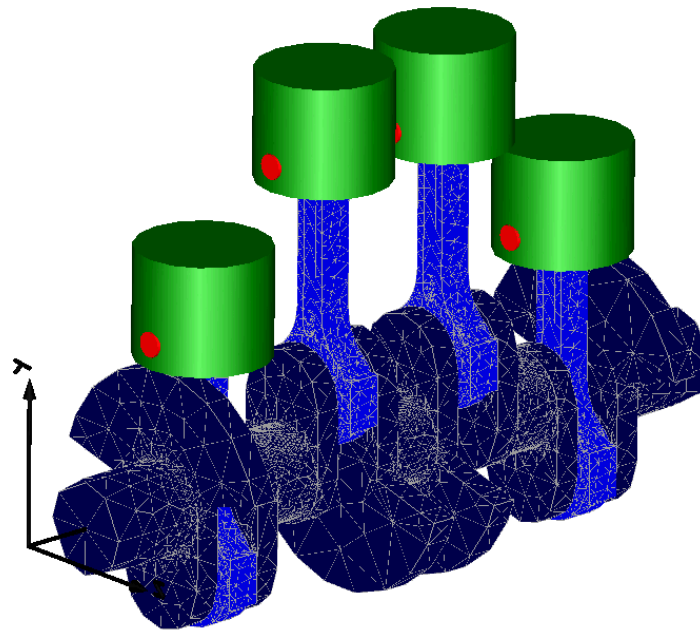
➤ 120 simulation nodes

➤ Time integration with gas force, 38 states,  $\approx 6$  cpu-s per s





# FlexibleBodies Library: 4 Cylinder Engine II



# FlexibleBodies Library: Quo vadis?

- ModalBody is an issue of ongoing discussion
  - relies on an appropriate FE-preprocessor
    - a 3<sup>rd</sup> third party product
    - in principle agreements were made, but no contracts are yet signed
  - the 1<sup>st</sup> implementation uses additional c-code
    - there is no distribution process to support the application on real-time platforms today
  - the 2<sup>nd</sup> implementation is pure Modelica code
    - might be good enough for moderate large models ?
    - release by the end of this year ?
  - an additional license key called ExtFlexibleBodies
  - graphical nodes-picking requires improvements
- Besides beams: implementation of other generic structures „on demand“
  - brake discs: annular Kirchhoff plate
  - framework structures

**Thank you very much for your attention**

