Simulation of Rigid and Flexible Multibody Systems

Tutorial at the Modelica‘2008 Conference

Bielefeld, March 3rd, 2008

Dr.-Ing. Andreas Heckmann, German Aerospace Center (DLR)

Institute of Robotics and Mechatronics

On the provided Software and Documents

- be sure to have Dymola Modelica 2008 installed
- copy folder **FlexibleBodiesLib 1.1 Modelica2008** to your harddrive
  - contains FlexibleBodiesLib 1.1 with specific license
  - subpackage Tutorial with tutorial examples
- copy/open **MultibodyTutorialModelica2008.pdf**

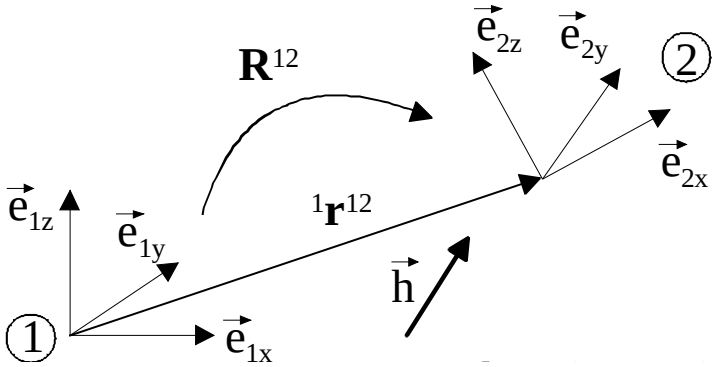
Contents




Contents

Modelica Multibody Basics: Orientation

- Coordinate systems and their orientation

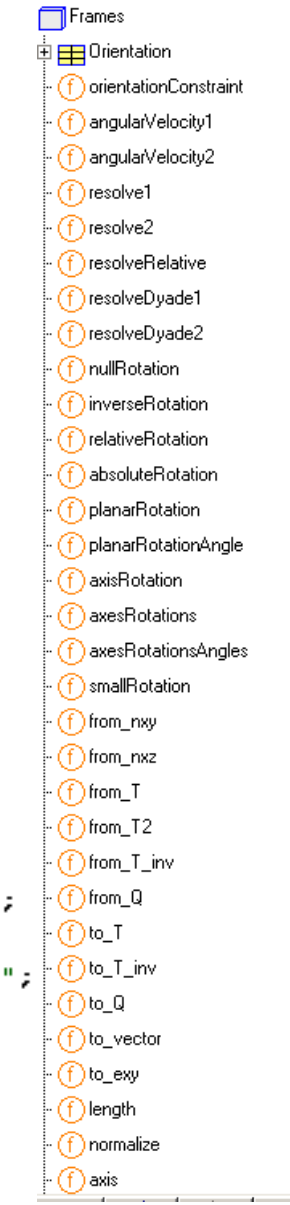



```
import MultiBody.Frames;
  Frames.Orientation R12;
  Real h1[3] "h resolved in frame 1";
  Real h2[3] "h resolved in frame 2";
equation
  h2 = Frames.resolve2(R12, h1); //or
  h1 = Frames.resolve1(R12, h2);
```

- Orientation object R^{12}
  - describes orientation of coordinate system 2 wrt.1
  - holds

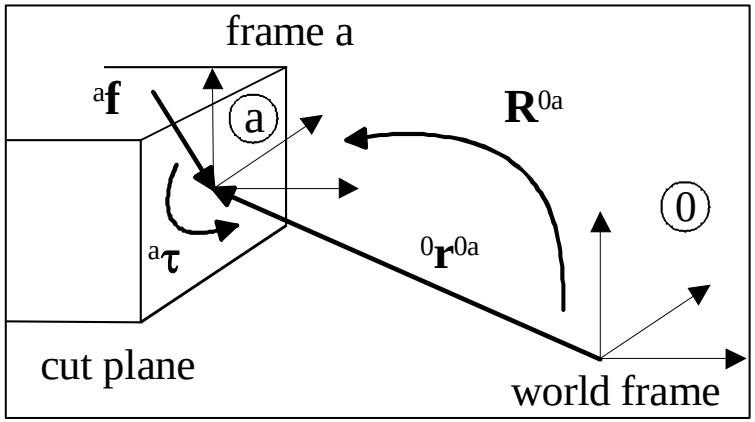
```
Real T[3, 3] "Transformation matrix from world frame to local frame";
SI.AngularVelocity w[3]
  "Absolute angular velocity of local frame, resolved in local frame";
```

  - may be computed using rotation angles or quaternions
- Multibody Lib. contains over 30 functions to operate on orientation objects



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



non-flow !

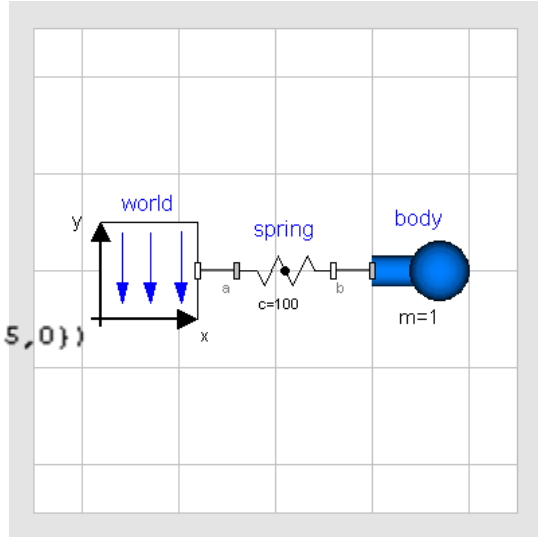
flow !

```
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";
  a;
end Frame;
```

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;
  a;
  Modelica.Mechanics.MultiBody.Parts.Body body(r_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

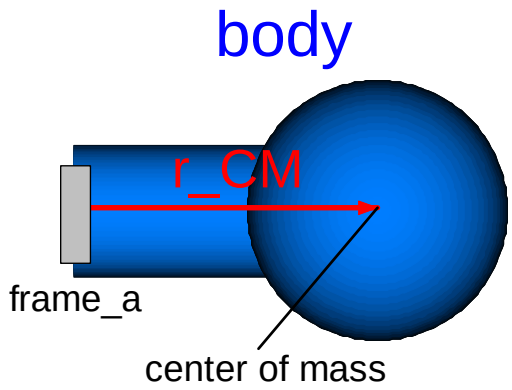
fixedTranslation

r={.1,3,1.5}

```
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);
```

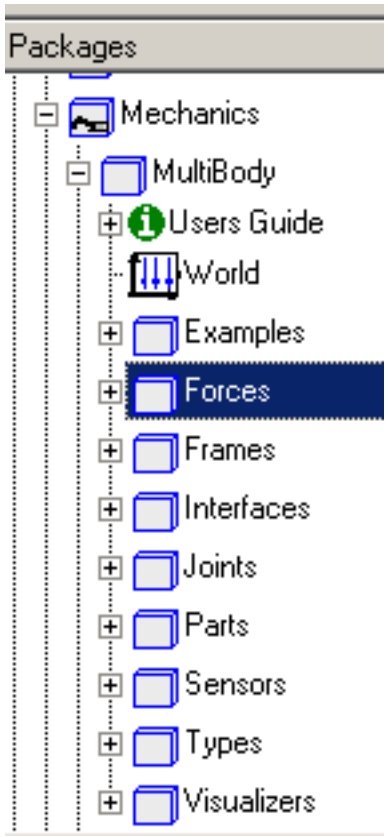
Modelica Multibody Basics: Components II

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

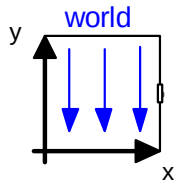



```
// import Modelica.Mechanics.MultiBody.Frames;
// translational kinematic differential equations resolved in local frame_a
v_a = Frames.resolve2(frame_a.R, der(frame_a.r_0));
a_a = der(v_a);

// 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);
```

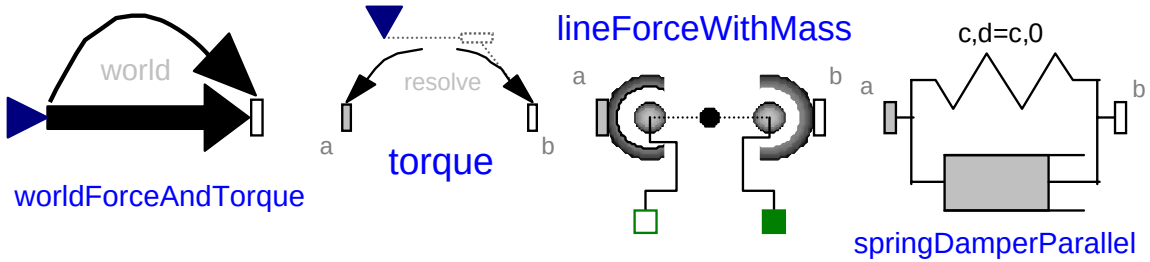
Modelica Multibody Basics: Elementary Components I



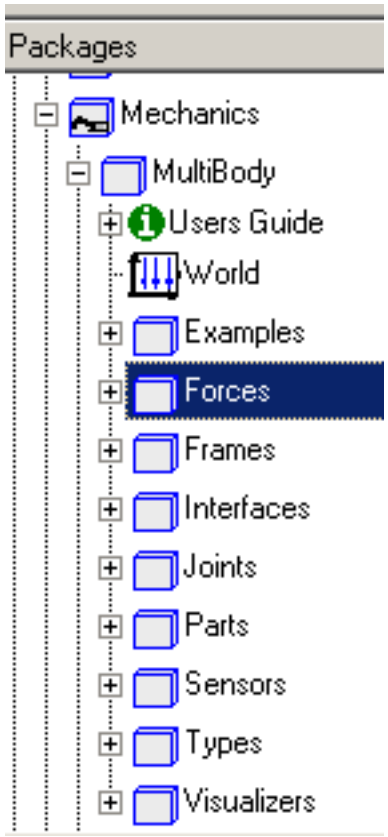
- Modelica.Mechanics.MultiBody.World
  - defines inertial frame, gravity, animation defaults



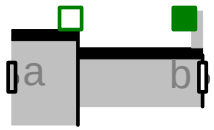
- Modelica.Mechanics.MultiBody.Forces
  - different resolution properties
  - interface to Real input functions and 1D mechanics
  - several spring/damper configurations



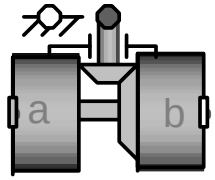
Modelica Multibody Basics: Elementary Components II



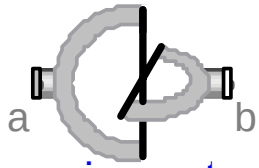
- Modelica.Mechanics.MultiBody.Joints
  - define specific degree of freedom
  - capability to set-up initial configuration
  - interface to/for 1D mechanics and rheonom motion
  - e.g.:



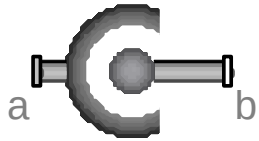
n={1,0,0}
prismatic



n={0,0,1}
revolute

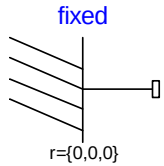


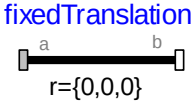
universal

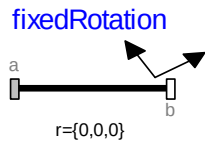


spherical

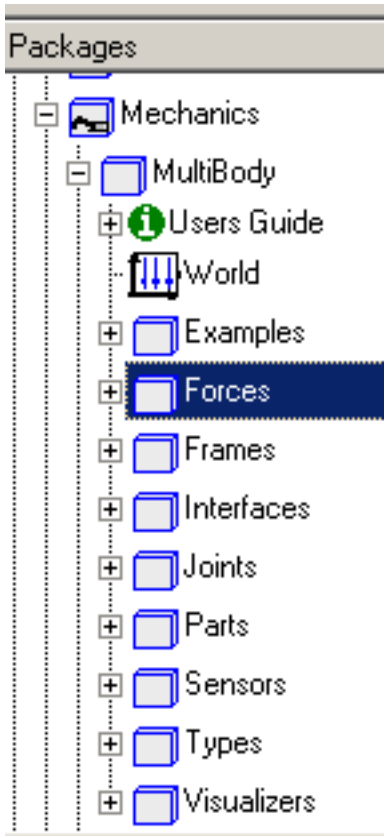
- Modelica.Mechanics.MultiBody.Parts
  - Fixed, FixedTranslation and FixedRotation



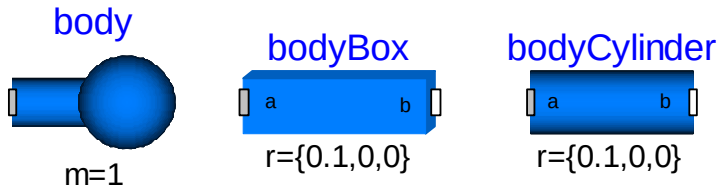




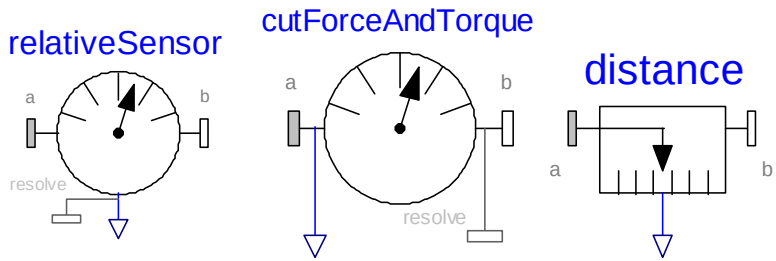
Modelica Multibody Basics: Elementary Components III



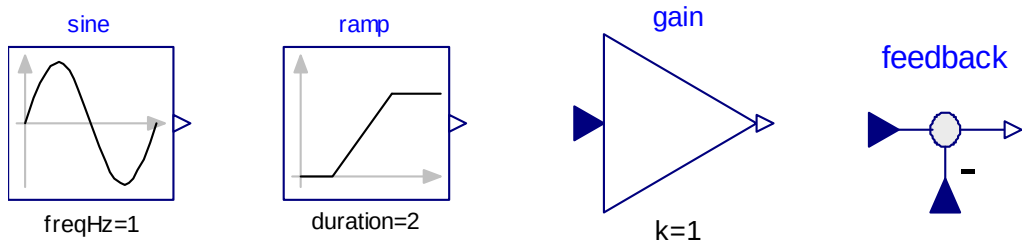
- Modelica.Mechanics.MultiBody.Parts
  - Rigid bodies with predefined geometric shapes



  - Modelica.Mechanics.Multibody.Sensors
    - for control and validation purposes

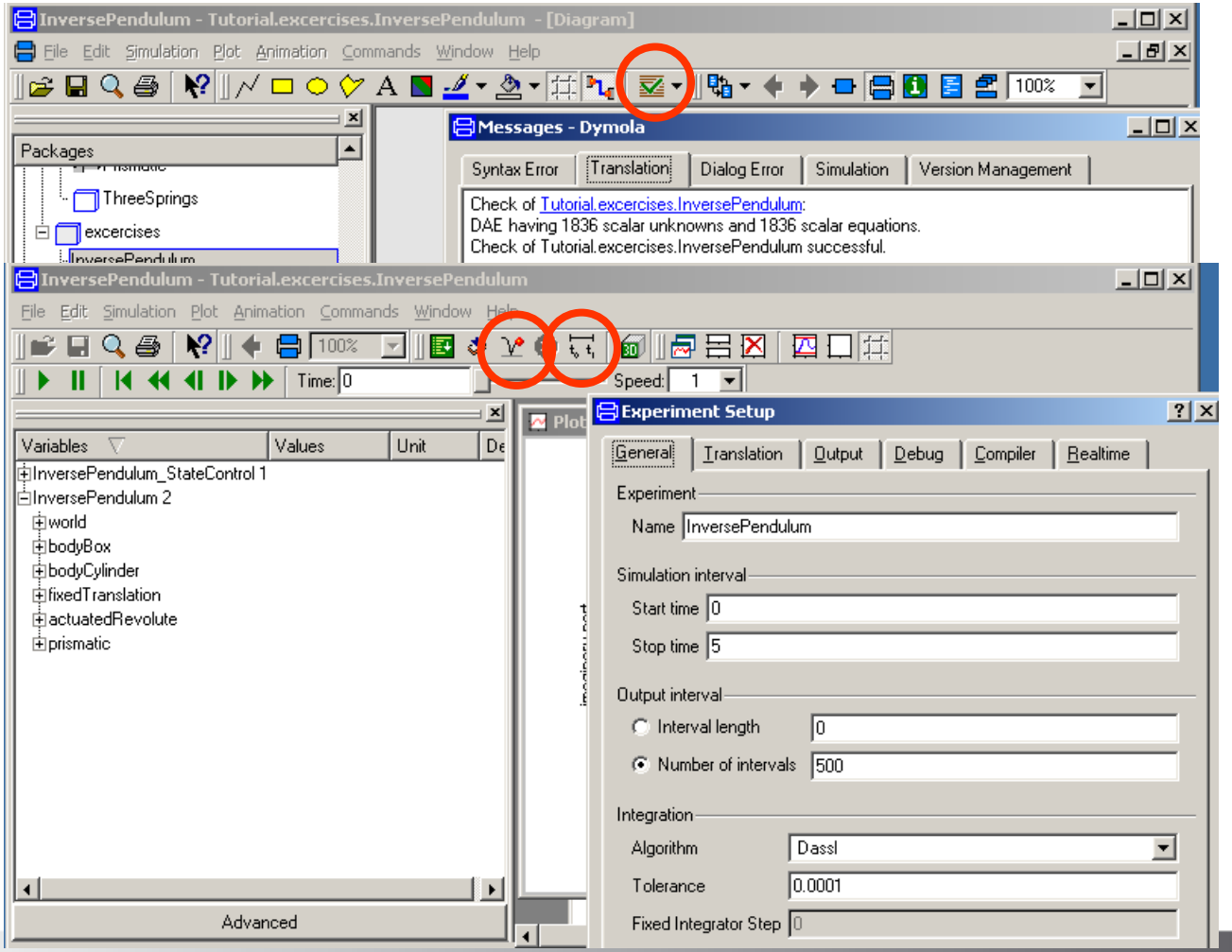


- Modelica.Blocks.Sources + Modelica.Blocks.Math



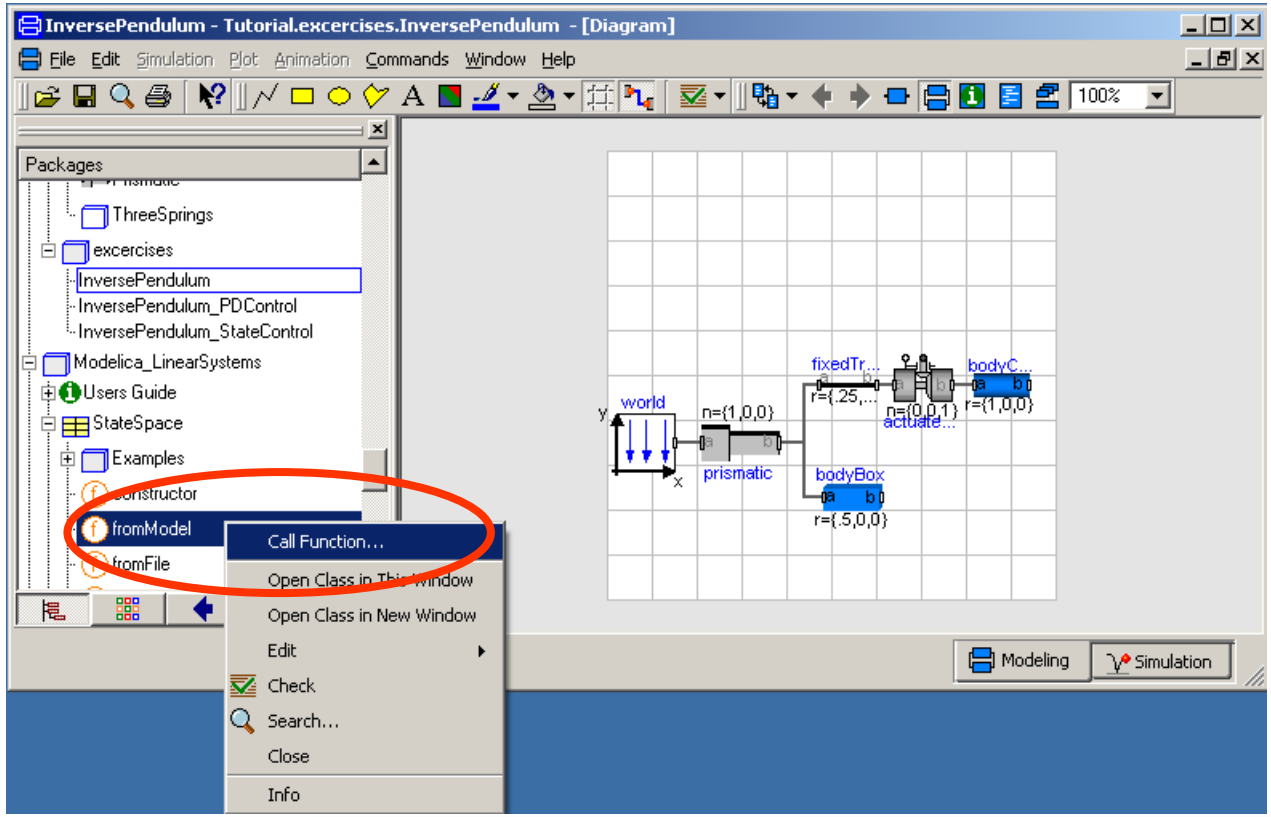
Modelica Multibody Basics: Analysis Methods

- Model check
- Experiment setup, translation and time simulation



Modelica Multibody Basics: Analysis Methods

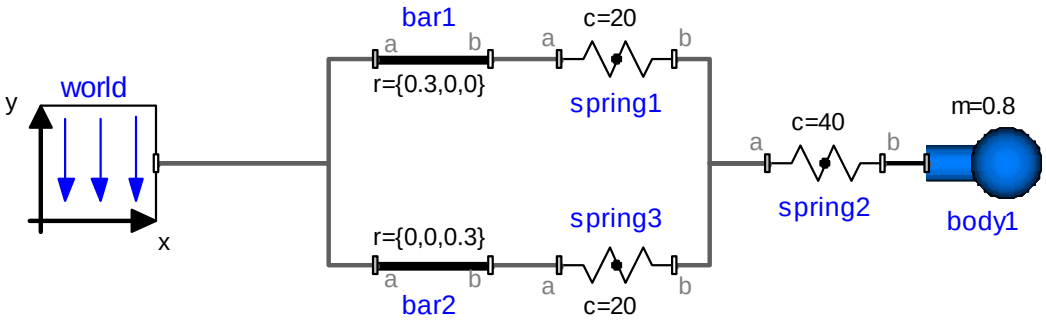
- Model check
- Experiment setup, translation and time simulation
- Eigenvalue analysis
  - Menu: File→Libraries→LinearSystems

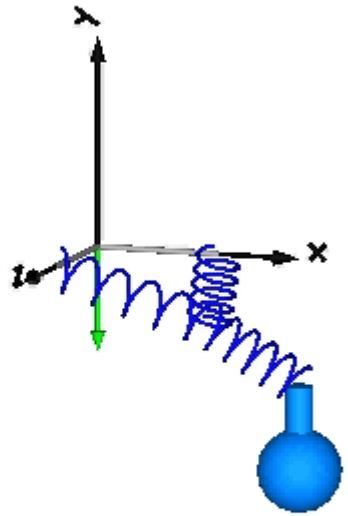


Contents

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
  - force elements may be connected to each other
  - e.g.:

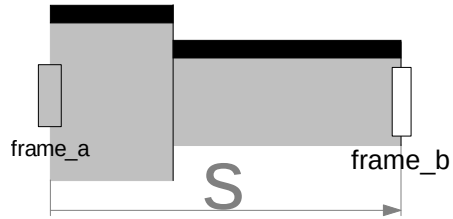




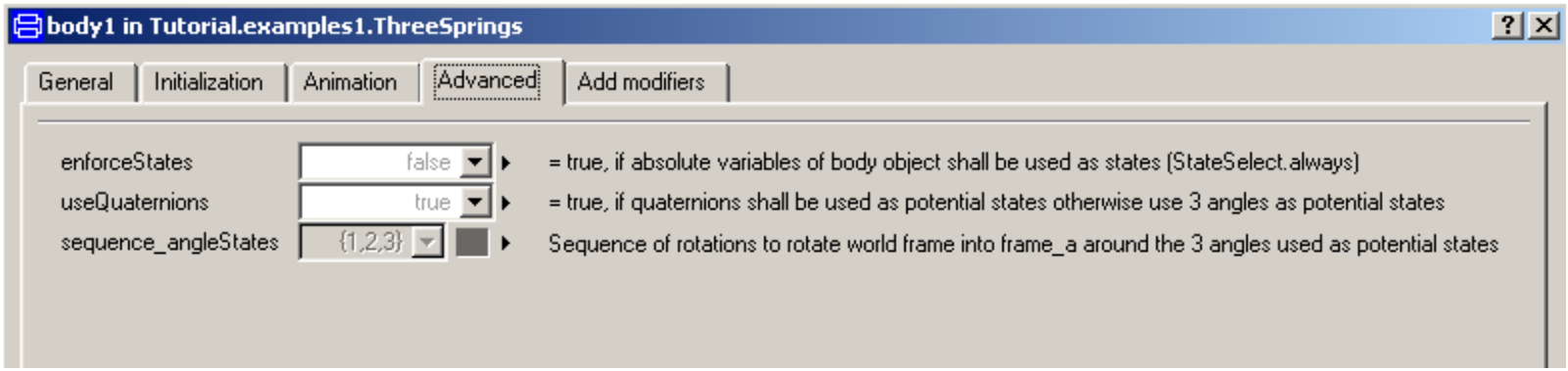
  - here: body coordinates: position, quaternions and their derivatives are used as states

Modelica Multibody Advanced: State selection II

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



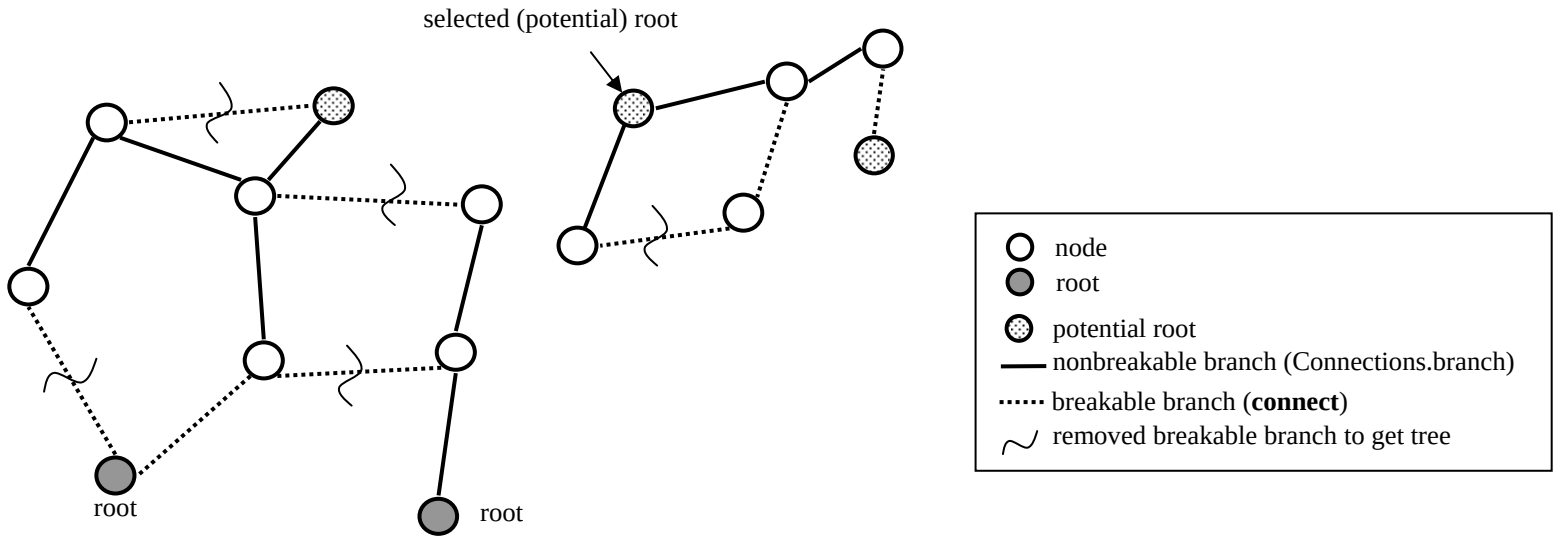

```
final parameter Real e[3]=Modelica.Mechanics.MultiBody.Frames.normalize(n)
  "Unit vector in direction of prismatic axis n";
SI.Position s(stateSelect=if enforceStates then
      StateSelect.always else StateSelect.prefer)
  "Relative distance between frame_a and frame_b";
SI.Velocity v(stateSelect=if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

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

  - Equations are rearranged to get a sequence for model evaluation (**B**lock **L**ower **T**riangle-partitioning)

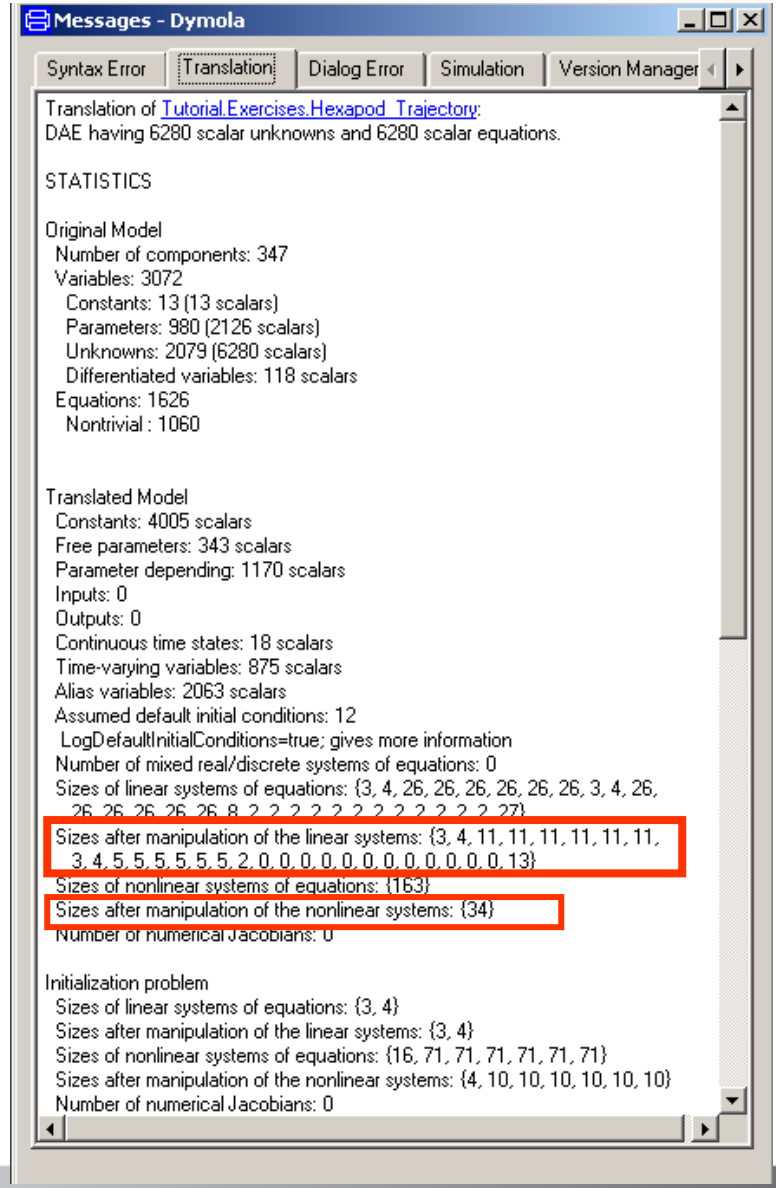
$$\begin{array}{c} \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{array} \begin{array}{c} \begin{array}{ccccc} z_1 & z_2 & z_3 & z_4 & z_5 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} \end{array} \Rightarrow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

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
    $\Rightarrow$ DAE
    $$0 = f(\dot{x}, x, y, t, \ldots) \quad \dim(f) = \dim(x) + \dim(y)$$
  - 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)
  - superflous potential states are deselected dynamically (dummy derivative method) $\Rightarrow$ ODE:
    $$\dot{x} = f(x, t, \ldots)$$

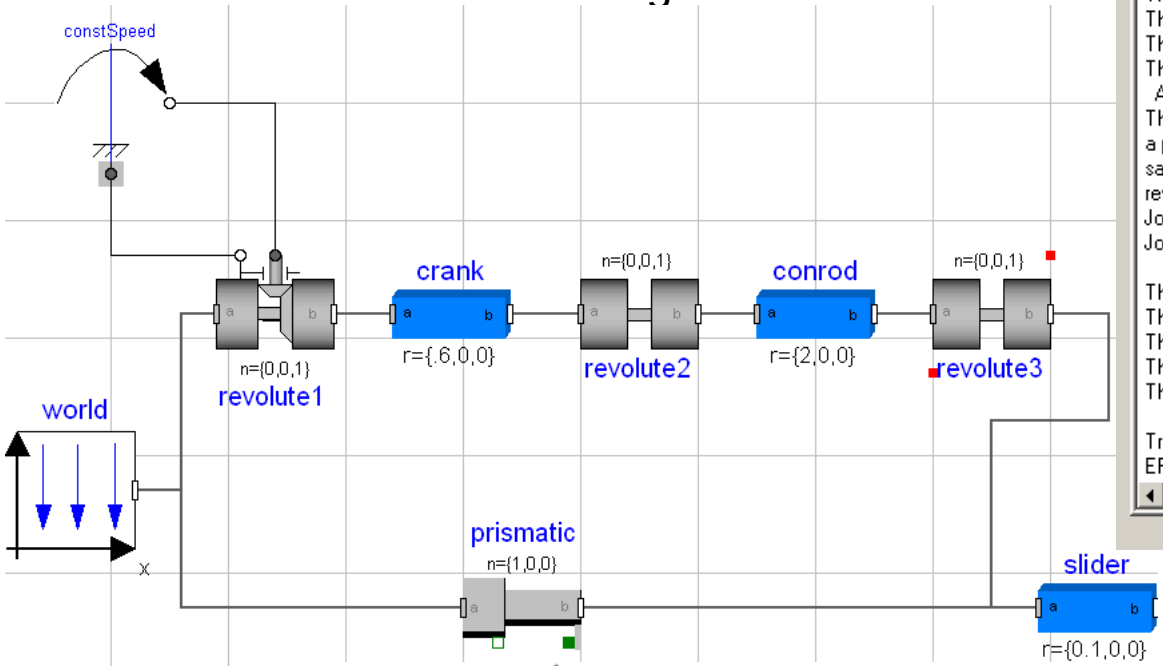
Modelica Multibody Advanced: Loops II

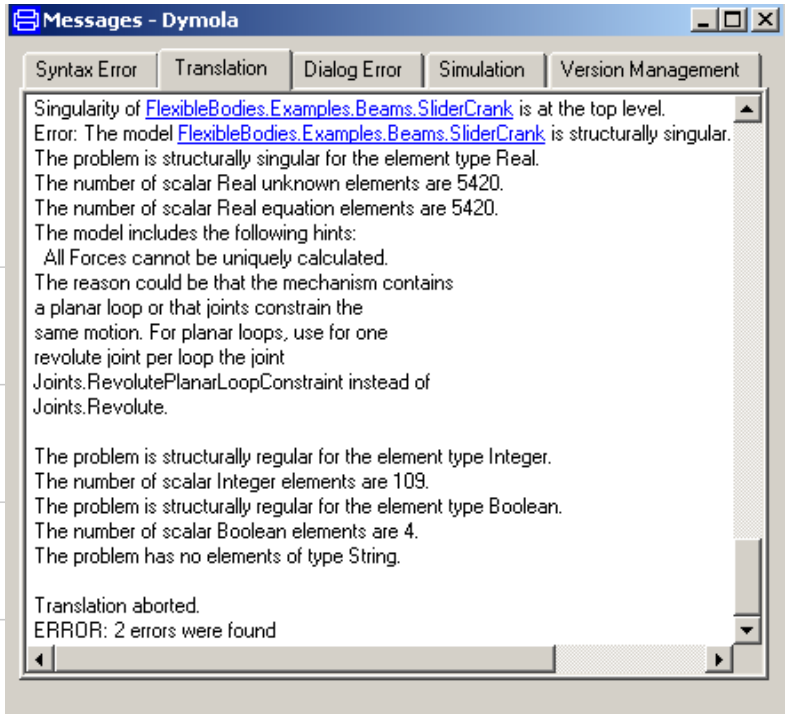
- review Translation Log in order to streamline simulation performance with model adjustments

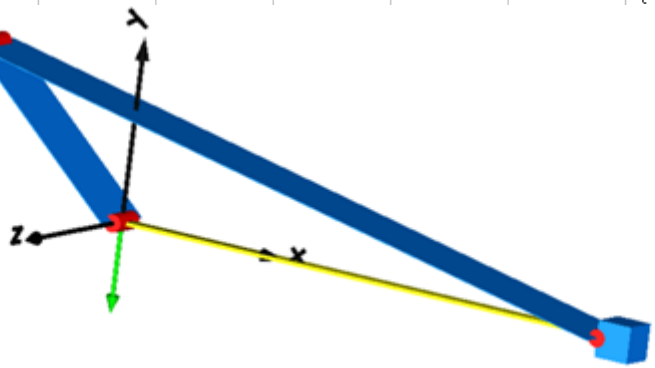


Modelica Multibody Advanced: Loops III

- Planar loops
  - error message

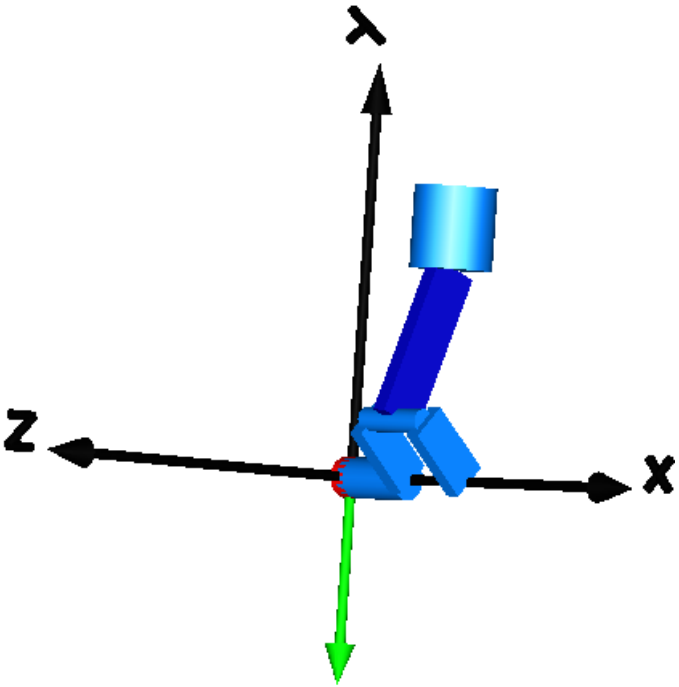


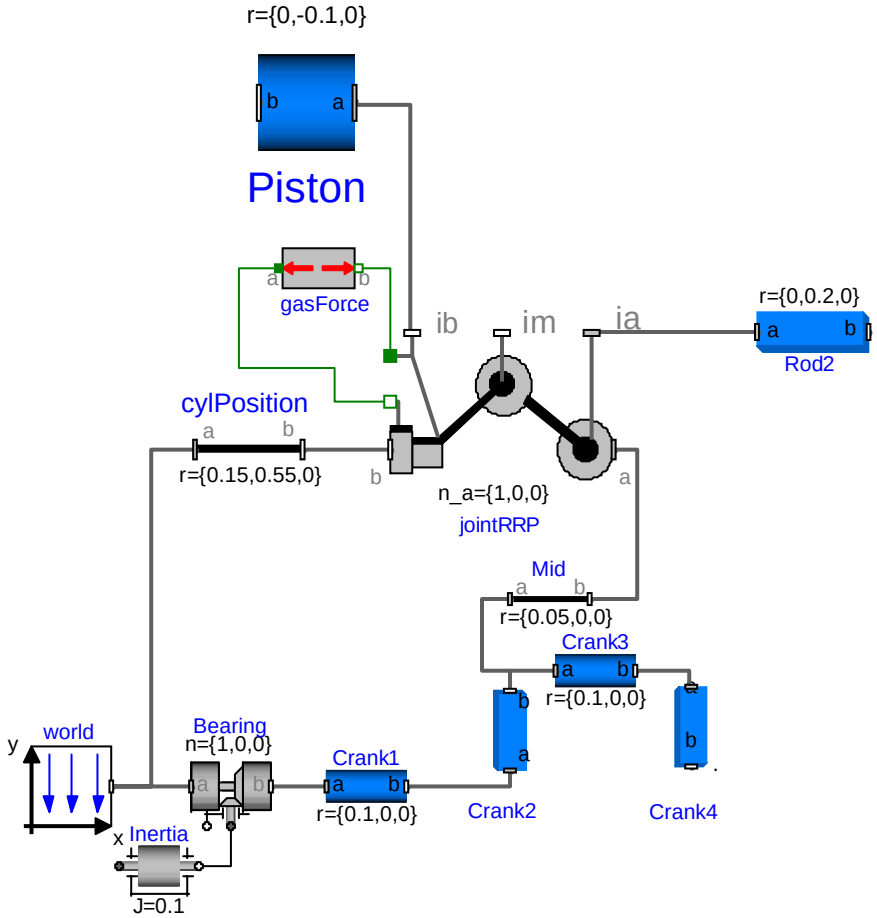




Modelica Multibody Advanced: Loops IV

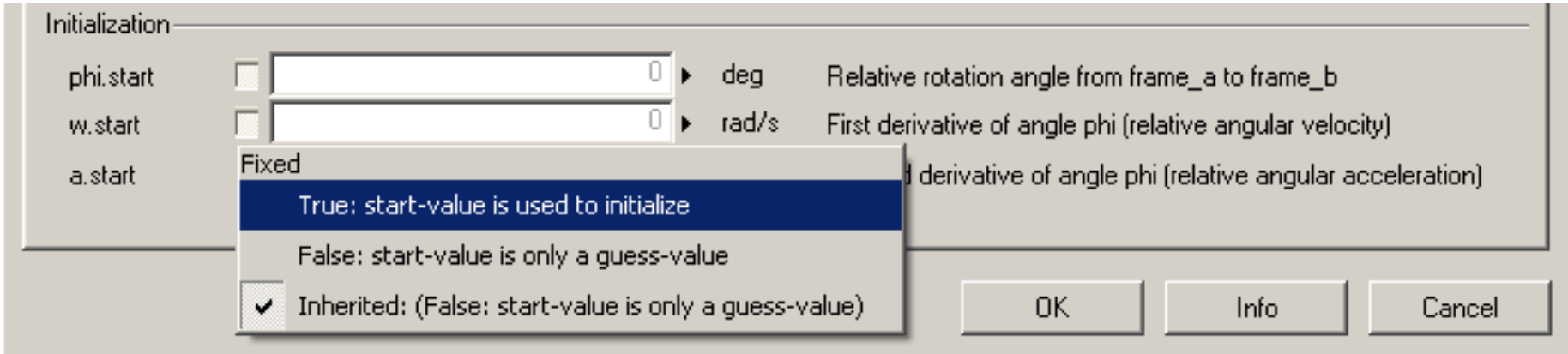
- Use of aggregrated joint objects
  - to profit from analytical loop handling according to the „characteristic pair of joints“ method by the group of Prof. Hiller





Modelica Multibody Advanced: Initialisation

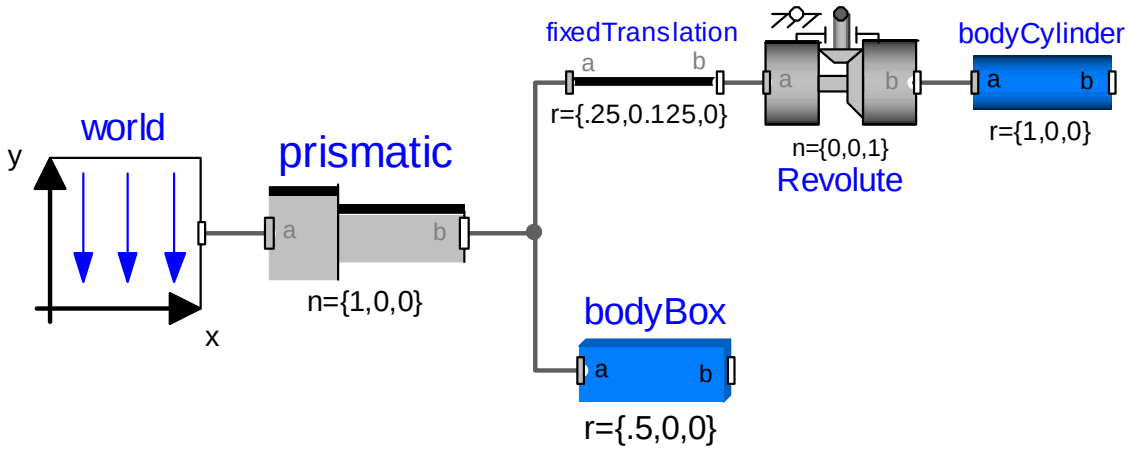
- Initialisation default:
  - every state is assumed to be arbitrary unless otherwise provided
  - Newton solver starts with guess value zero in order to find consistent initial states unless otherwise provided
- If initialisation fails
  - determine, i.e. fix, characteristic variables/states in order 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 initialisation system consistent

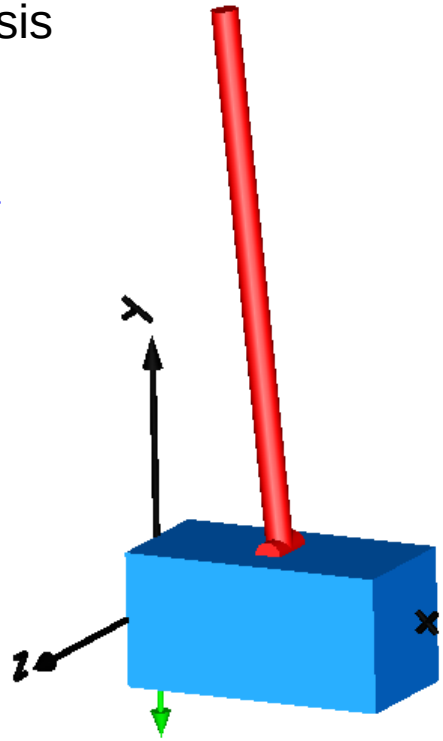


Contents

Example 1: Control of an inverse pendulum I

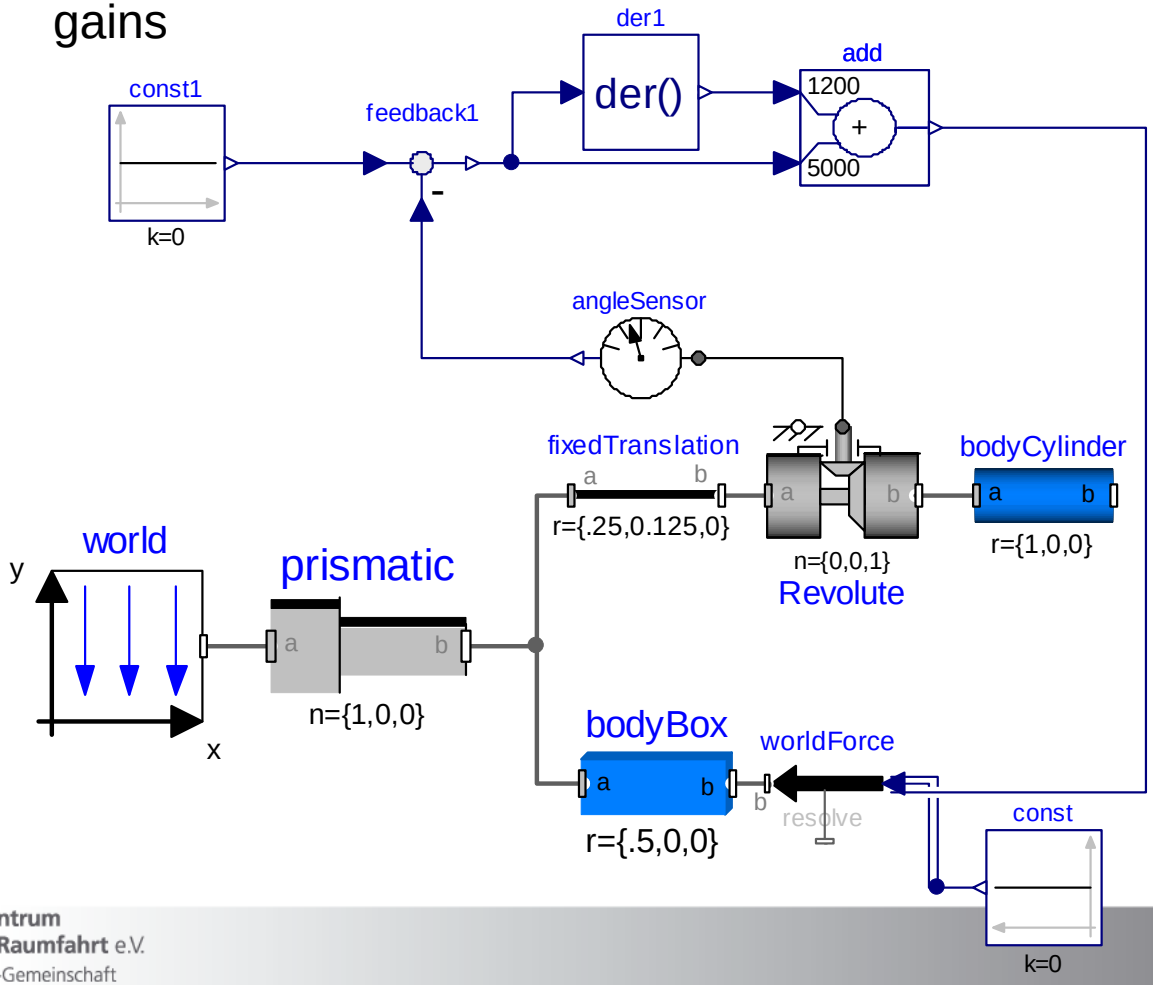
- Initial model
  - Box: 0.5 x 0.25 x 0.25 m
  - actuatedRevolute: 90° phi_offset, 5° phi_start
  - perform time simulation and eigenvalue analysis





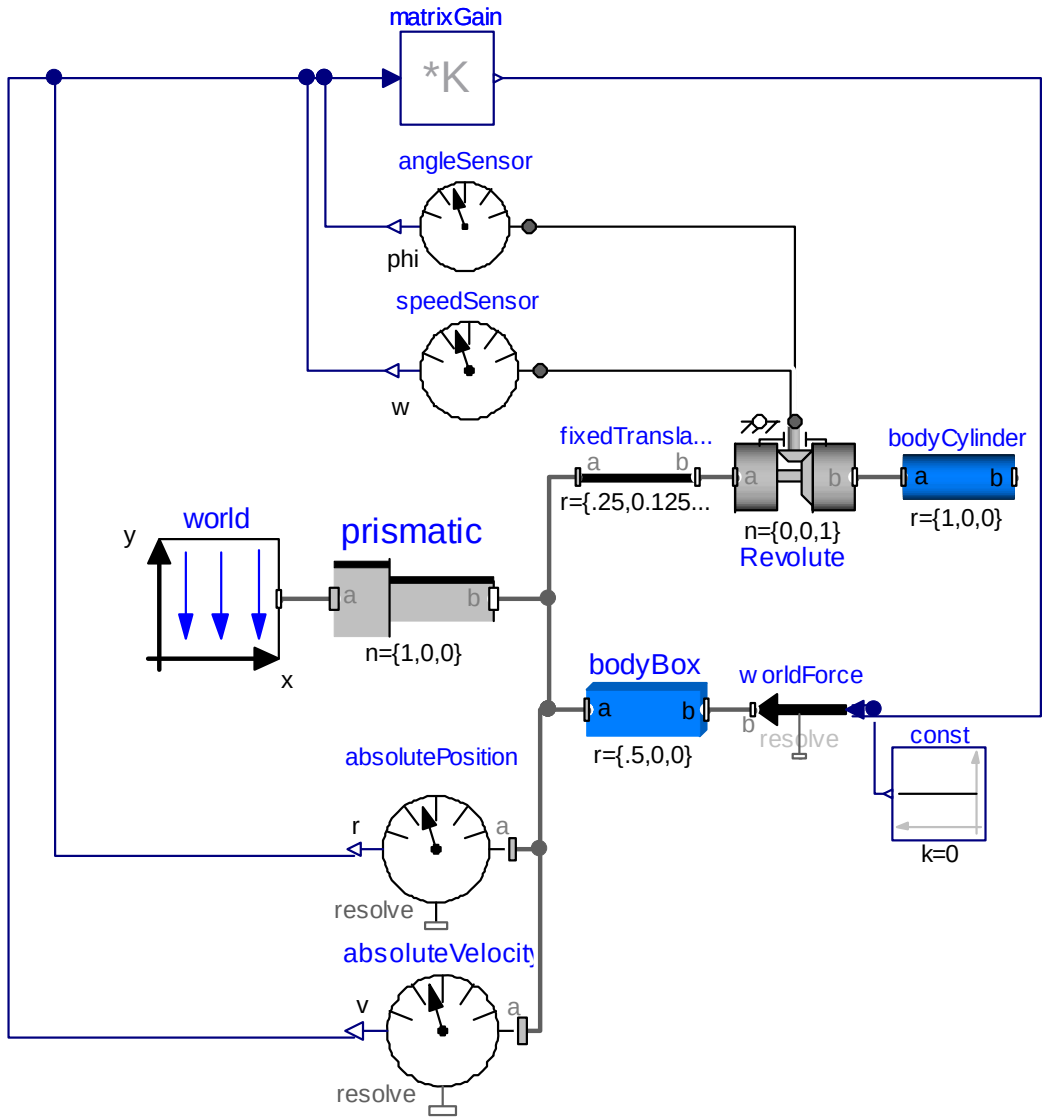
Example 1: Control of an inverse pendulum II

- PD-Control of angle only
  - use extend to inherit initial model
  - use transient response and pole placement to set-up contoller gains



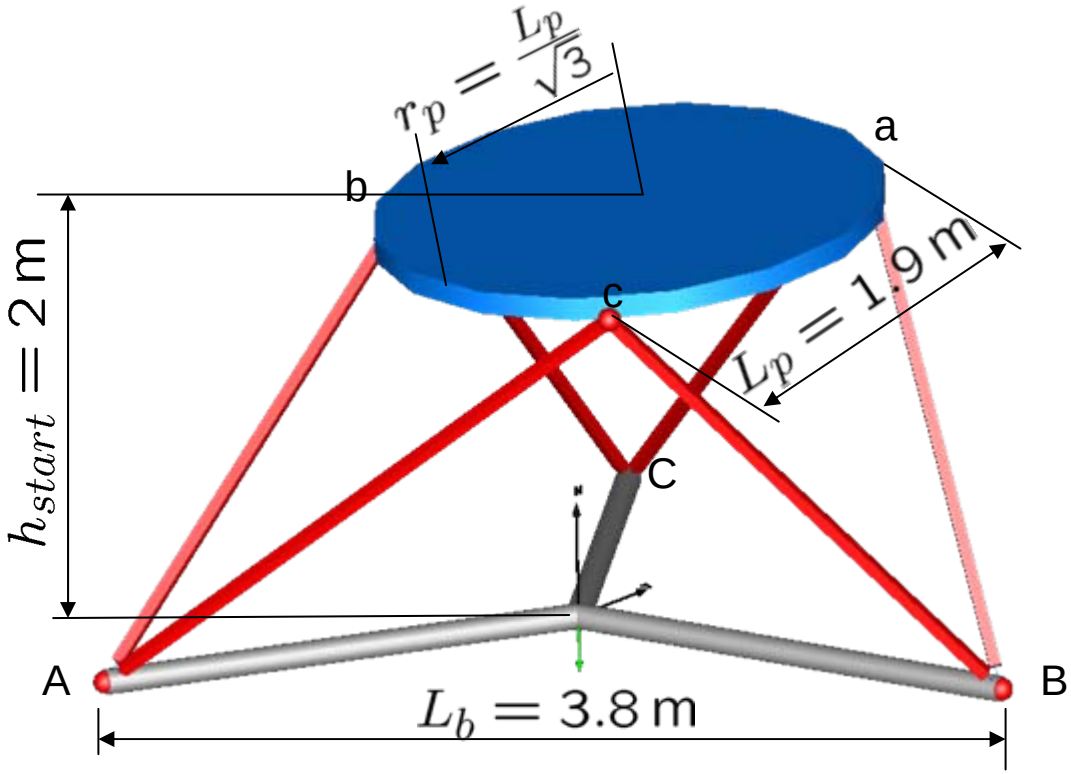
Example 1: Control of an inverse pendulum III

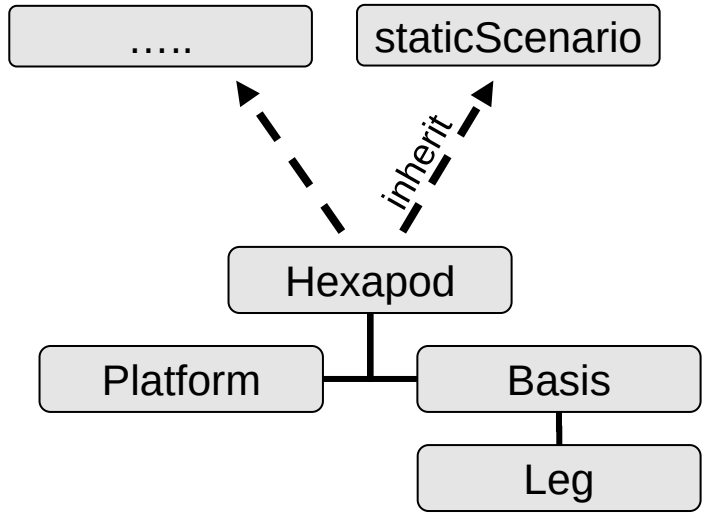
- Improve control
  - e.g. state control



Exercise 2: Hexapod I

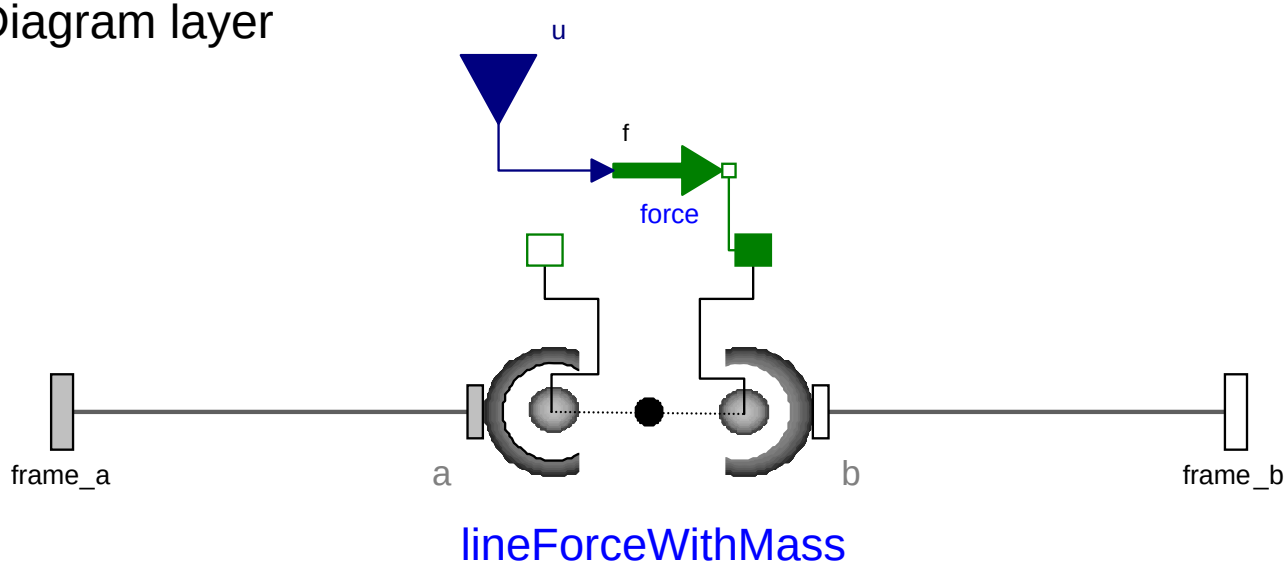
- The modelling concept



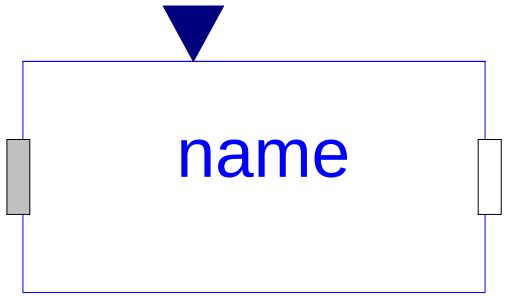


Exercise 2: Hexapod II

- Model Leg
  - Diagram layer



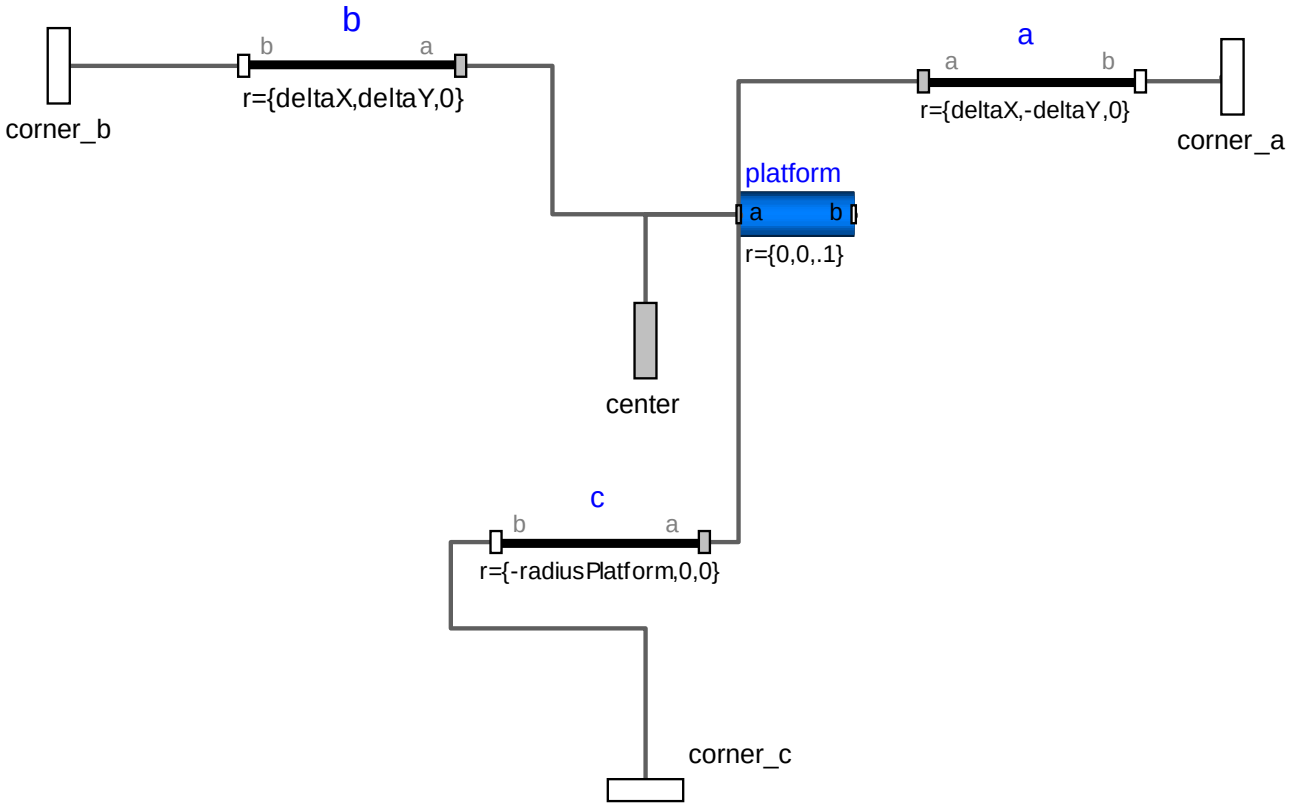
  - Icon layer



Exercise 2: Hexapod III

- Model Platform

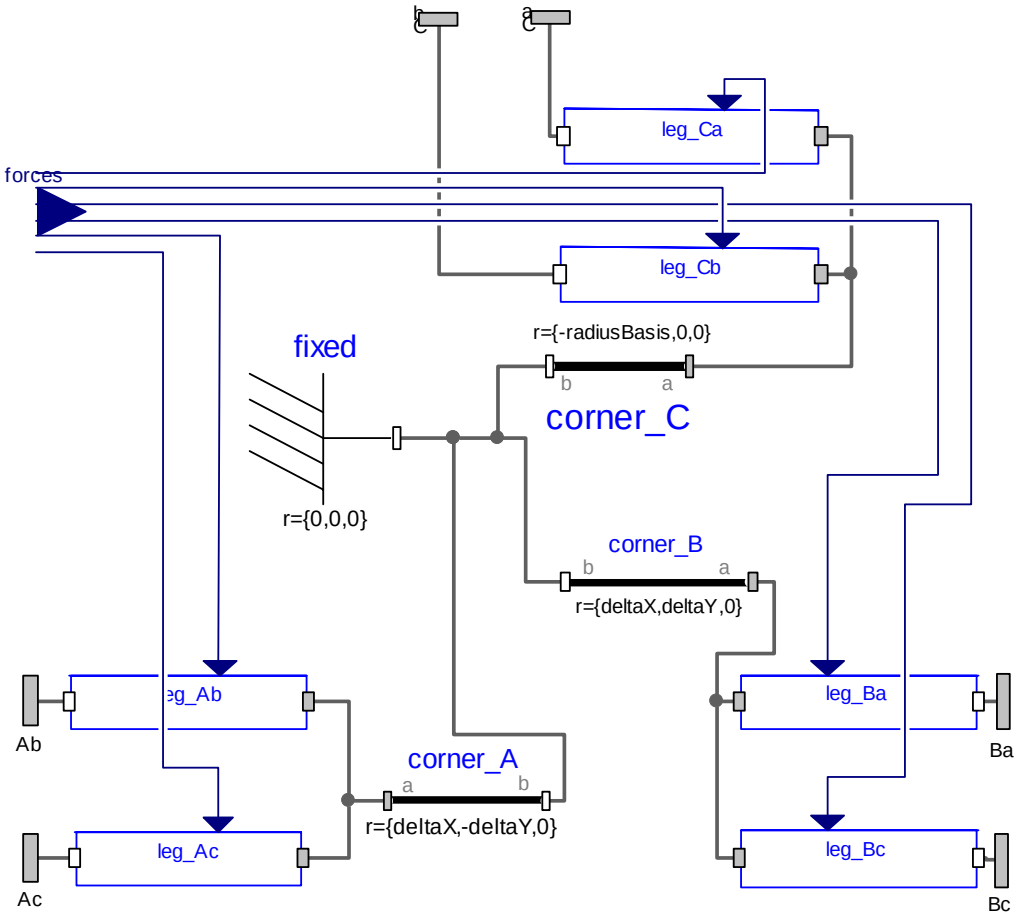
```
model Platform
  import SI = Modelica.SIunits;
  parameter SI.Length Lp=1.9 " length of a side of the platform";
  final parameter SI.Length radiusPlatform=Lp/sqrt(3);
  final parameter SI.Length deltaX=radiusPlatform/2;
  final parameter SI.Length deltaY=Lp/2;
```



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;
```



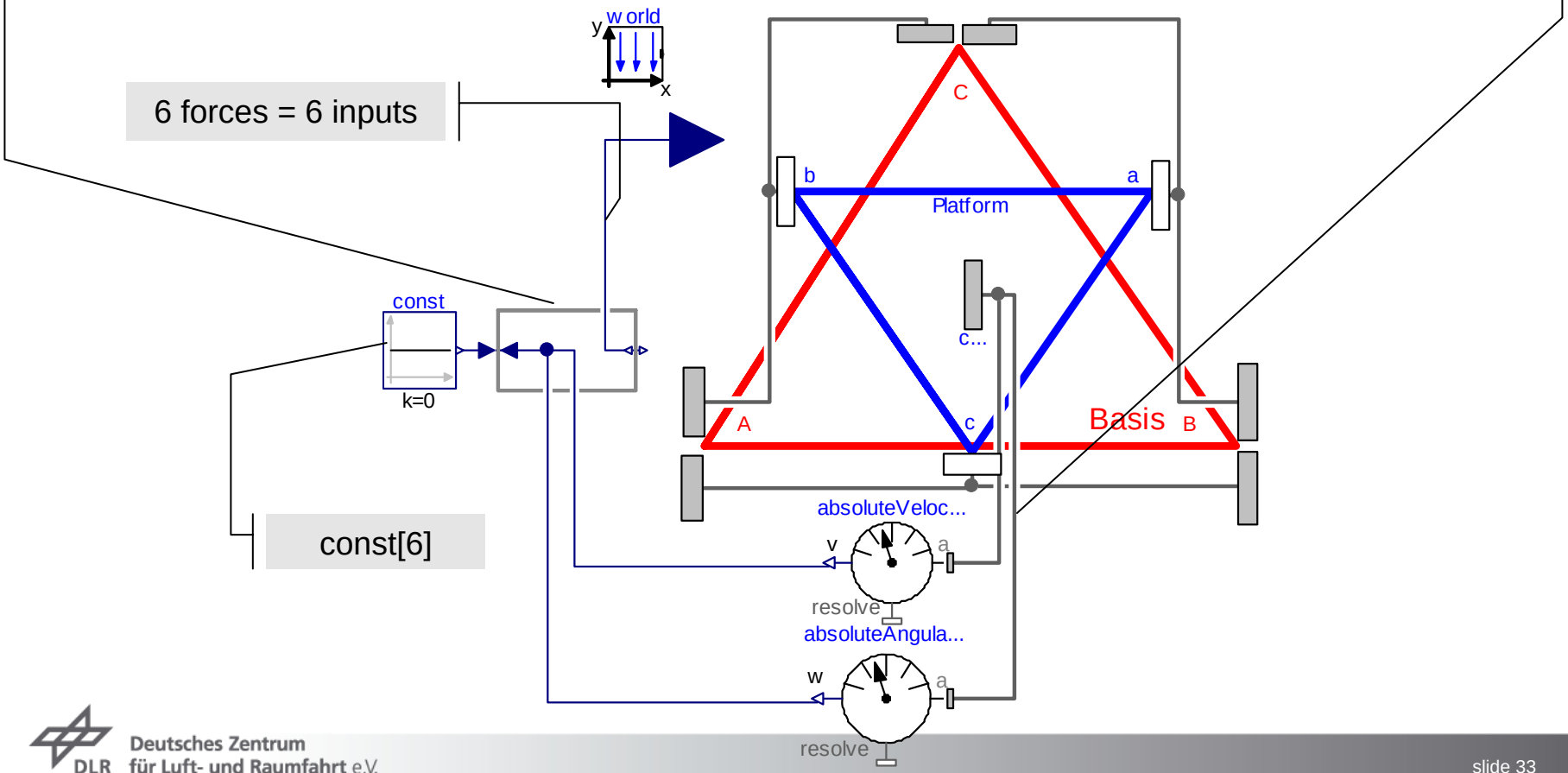
Exercise 2: Hexapod V

- Hexapod_StaticScenario
  - actuator force in initial position ?

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

Modelica.Blocks.Math.InverseBlockConstraints

6 velocities = 6 outputs

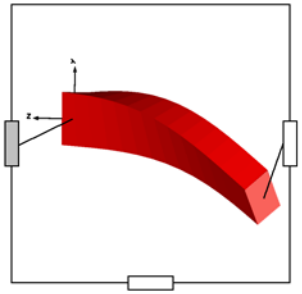


Contents

- Modelica Multibody Basics
- Modelica Multibody Advanced
- Exercise 1: Control of an inverse pendulum
- Exercise 2: Hexapod
- FlexibleBodies Library: Beams
- Exercise 3: Aircraft Fin
- FlexibleBodies Livbrary: General bodies based on finite element data
- Exercise 4: Rod

FlexibleBodies Library: Beams versus ModalBody

beam



Common Issues

Floating frame of reference

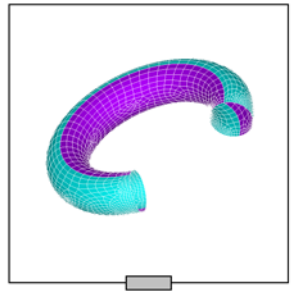
Equations of motion

SID-data-structure

Standard-Input-Data: Wallrapp ’94

Disjunctive Issues

modalBody



Analytical beam description

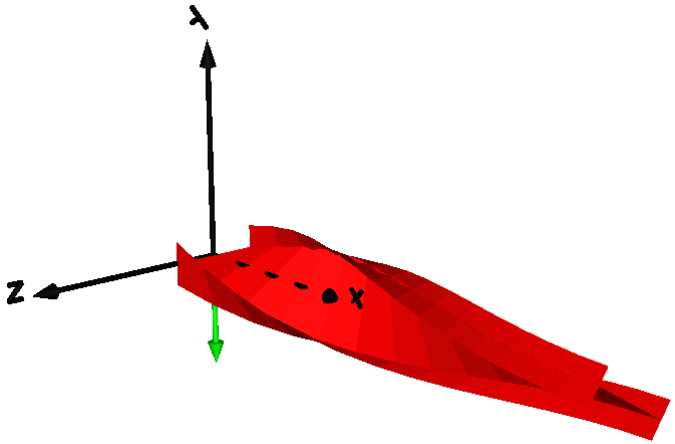
Modelica generated SID

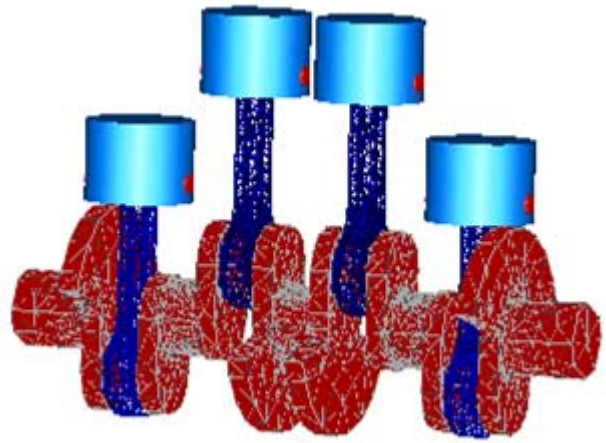
Animation uses beam description

FEM-based body description

External generated SID

Animation based on external data

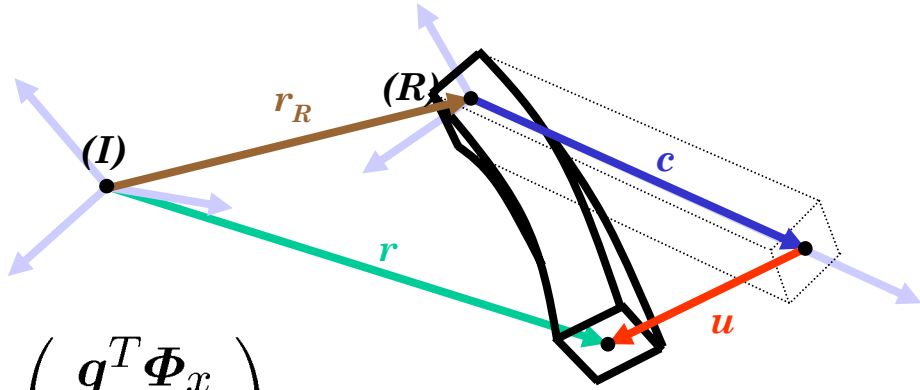




FlexibleBodies Library: Equations of motion

- Floating frame of reference

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



- Modal approach

$$\boldsymbol{u}(\boldsymbol{c},t) = \boldsymbol{\Phi}(\boldsymbol{c})\,\boldsymbol{q}(t) + \frac{1}{2}\begin{pmatrix} \boldsymbol{q}^T\boldsymbol{\Phi}_x \\ \boldsymbol{q}^T\boldsymbol{\Phi}_y \\ \boldsymbol{q}^T\boldsymbol{\Phi}_z \end{pmatrix}\boldsymbol{q}$$

- Equations of motion

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

Bremer/Pfeiffer ’92, Schwertassek/Wallrapp ’99

FlexibleBodies Library: Beam theory

- 2nd order displacement field
  - bending in xy- und xz- plane, torsion and lengthening

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

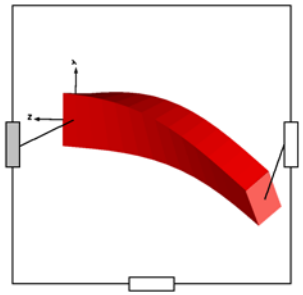
- Raleigh-Ritz-approach for a straight, homogenous, isotropic beam with constant cross section
  - expansion with analytic eigenvalue-solutions of the Euler-Bernoulli beam

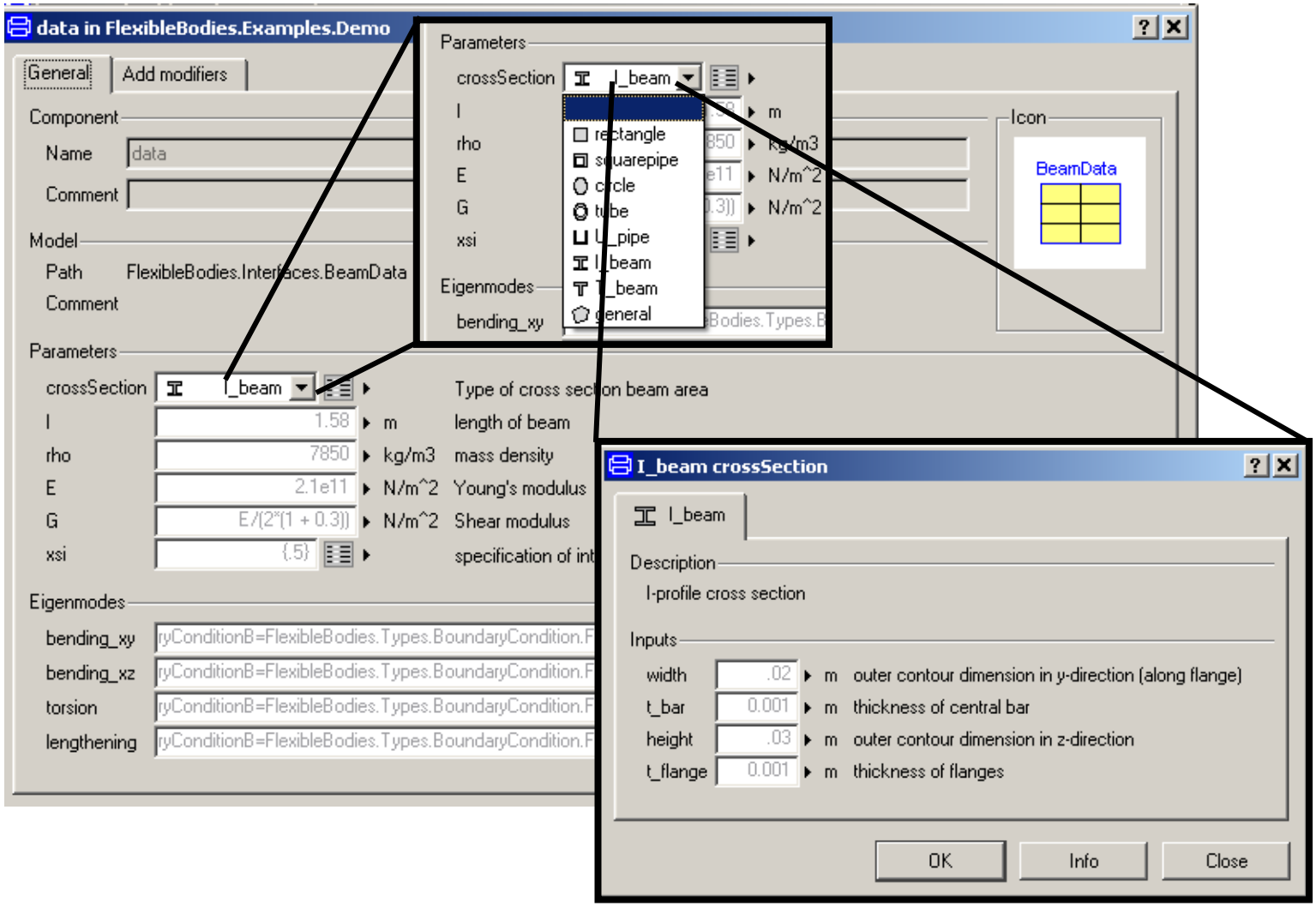
$$v(x,t) = \boldsymbol{\Phi}_v(x)\boldsymbol{q}_v(t) \qquad \Phi_i = \begin{pmatrix} \cosh(\tau_i x) \\ \sinh(\tau_i x) \\ \cos(\tau_i x) \\ \sin(\tau_i x) \end{pmatrix}^T \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}_i$$

Bremer/Pfeiffer ’92, Timoshenko ‘55

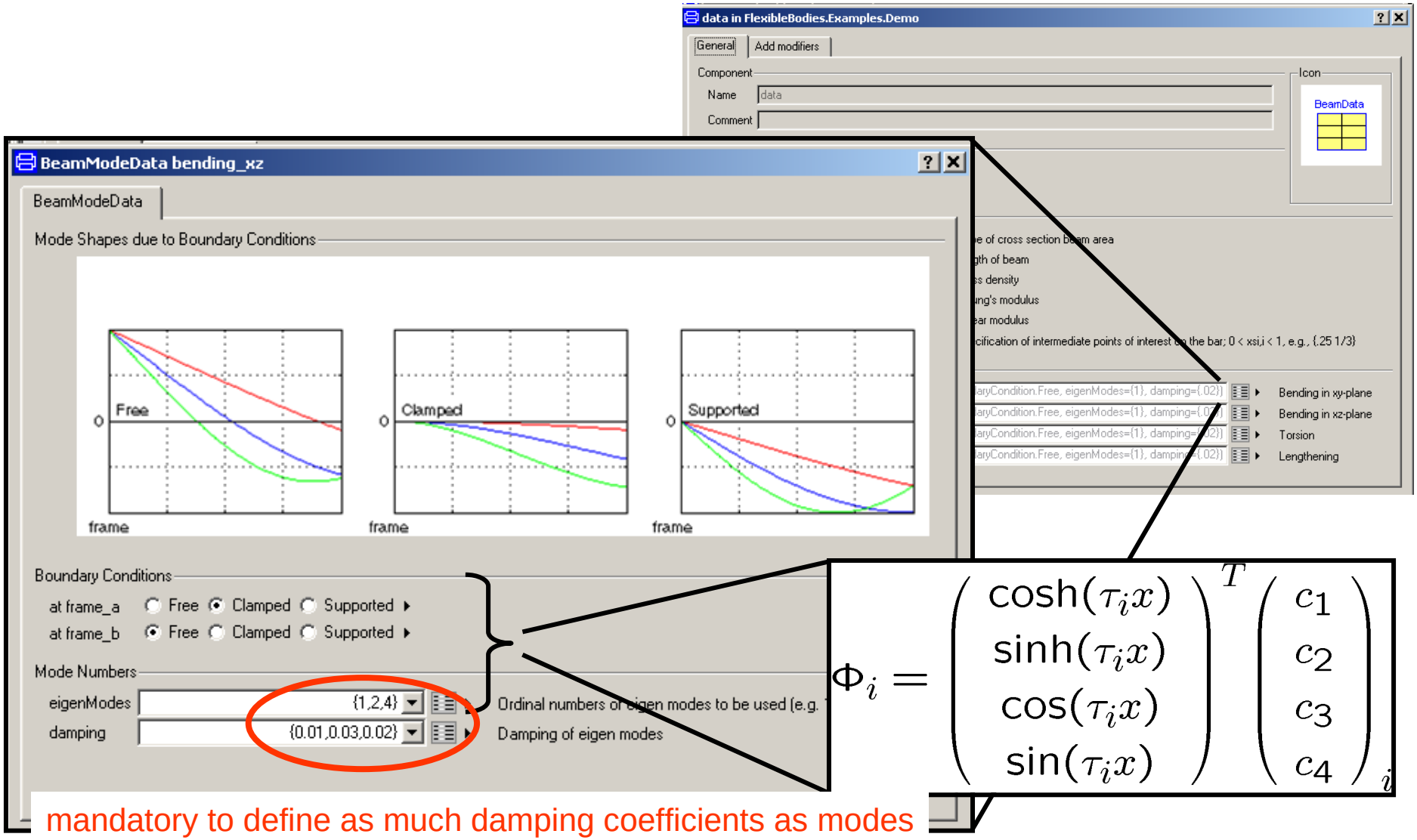
FlexibleBodies Library: Beam menu set-up I

beam





FlexibleBodies Library: Beam menu set-up II

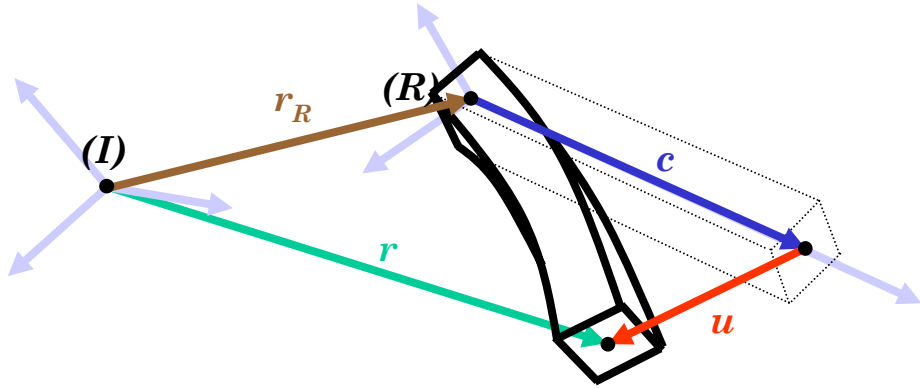


mandatory to define as much damping coefficients as modes

FlexibleBodies Library: Boundary Conditions I

- Mechanical interpretation

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



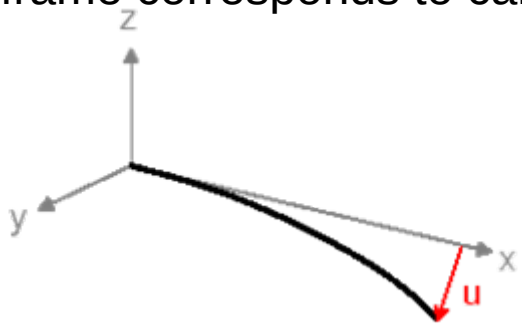
- let‘s say: frame of reference is pinned at frame_a with $\boldsymbol{c} = 0$

  $\Rightarrow$ motion of frame_a is completely described by $\boldsymbol{r}_R(t)$

  (and related orientation)

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

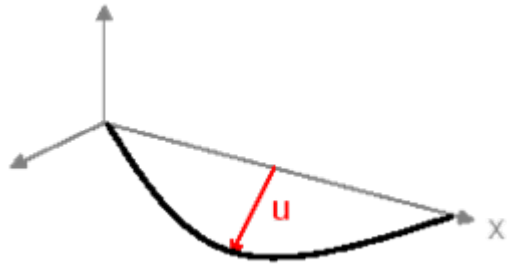
- clamped-free: tangent frame corresponds to cantilever beam boundary conditions



FlexibleBodies Library: Boundary Conditions II

- supported-supported: chord frame

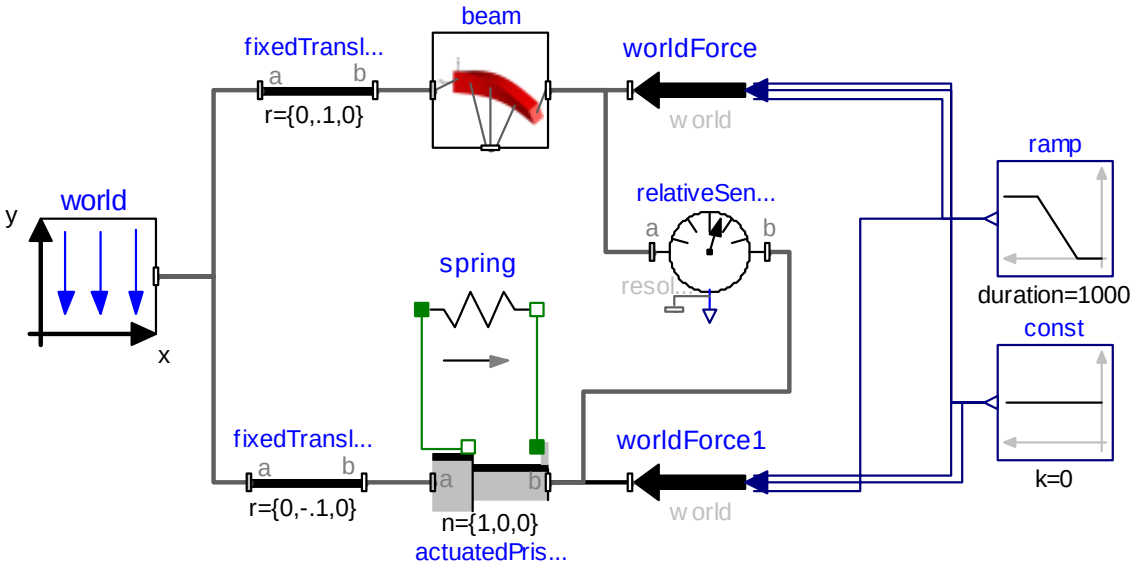
$$u(c = 0, t) = 0 \quad u(c = (l, 0, 0), t) = 0$$

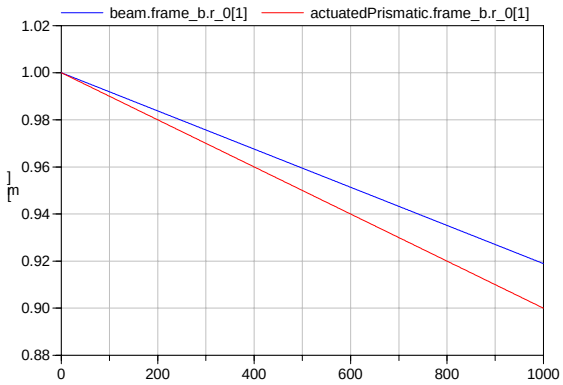


- free-free: Buckens frame
  - linear and angular momentum due to body deformation are zero
- every combination of tangent, chord and Buckens frames in different spatial directions is possible
- General rule
  - align the boundary conditions with the degree of freedom of the joints to which the beam 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

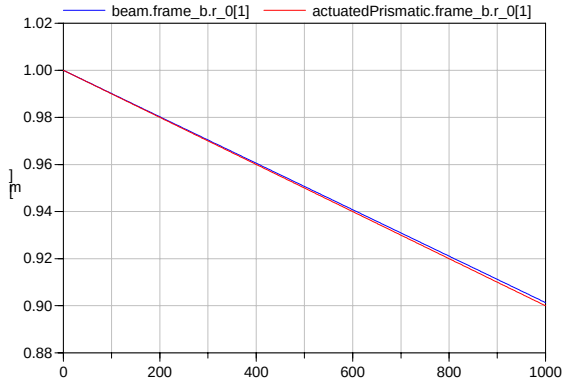




1 eigenmode

	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



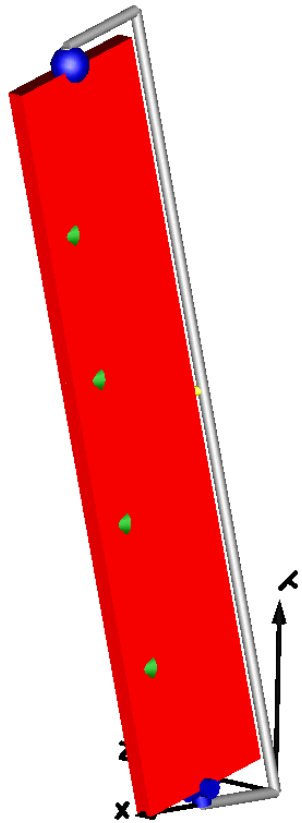
15 eigenmodes

FlexibleBodies Library: A classic pitfall II

- Mechanical background
  - static deflections rely on elastic properties only
  - eigenmodes consider elastic and interia properties
    - that‘s why they are well suited for dynamic problems
- Geometrical background
  - analytically: $u = c \cdot x$
  - expansion with eigenmodes: $u = \sin(\frac{2x}{\pi l}) + \sin(\frac{2x}{3\pi l}) + \ldots$
- 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 !

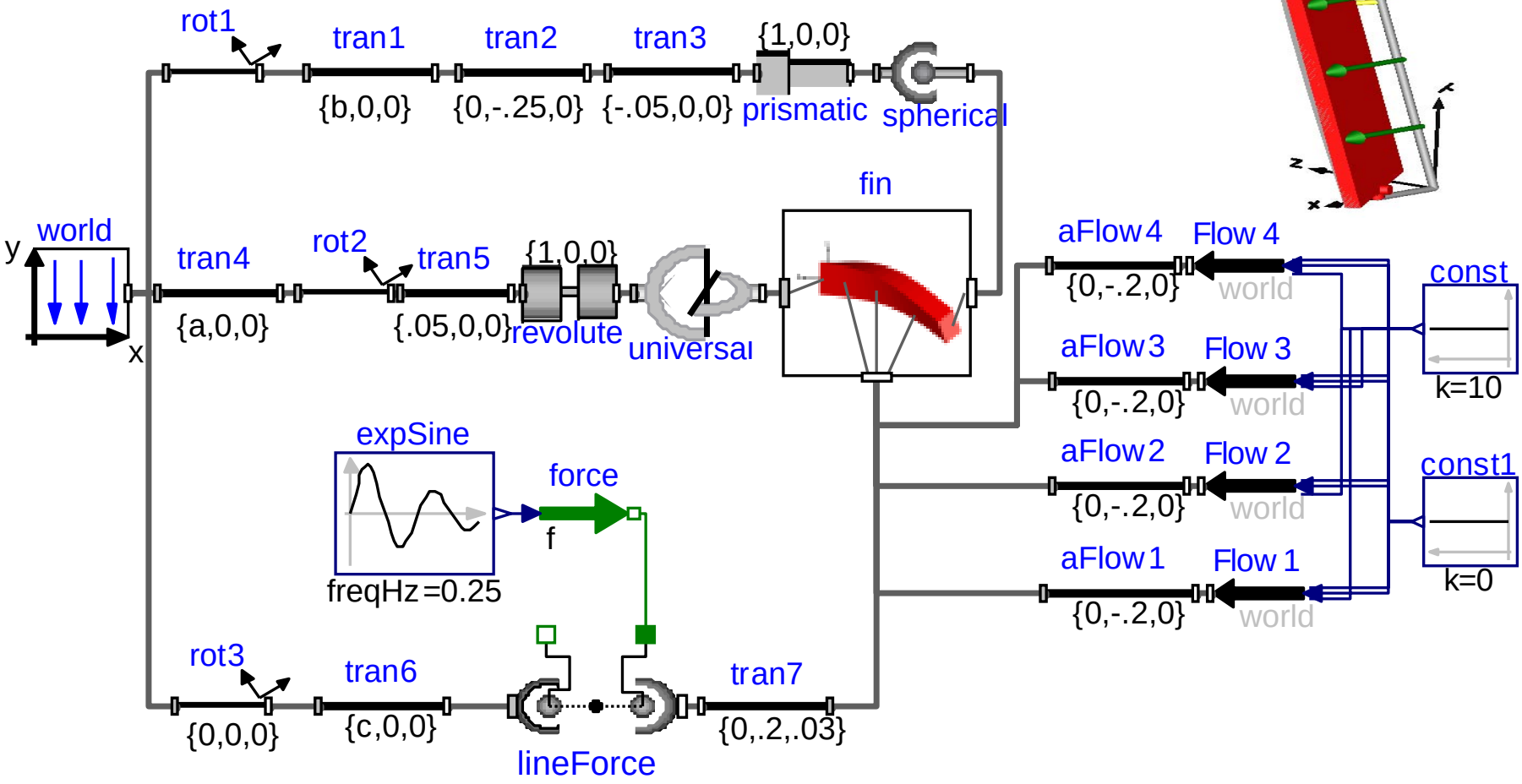
Example 3: AircraftFin

- fin
  - mounted in a frame with 15° inclination
  - profil: squarepipe 0.4 x 0.05 x 0.01 m
  - 2 m long
  - 2730 kg/m^3, $7 \cdot 10^{10}$ N/m^2 (aluminium)
  - 1 xz-bending and 1 torsion mode
  - 5 mid-nodes ξ = {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;
```

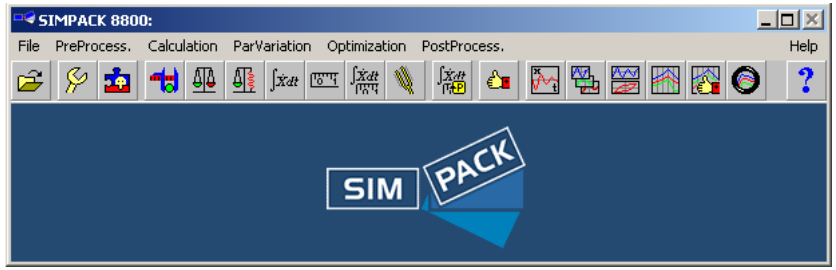
Example 3: AircraftFin



Contents

FlexibleBodies Library: FEM-preprocessing

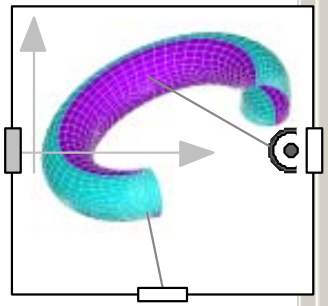
- Simpack-FEMBS: FEM to multibody system preprocessor
- Maintained and distributed by INTEC GmbH, Oberpfaffenhofen



- 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
    $\Rightarrow$ system matrices, nodes to retain, eigenmodes, mesh information
  - in Simpack-FEMBS
    $\Rightarrow$ 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_clampled

sphericalJointNodes	{20001,20002}	Simulation nodes of nodes_sphericalJoint (= do not constrain rotation)
clampedNodes	fill(0, 0)	Simulation nodes of nodes_clampled (= constrain translation and rotation)

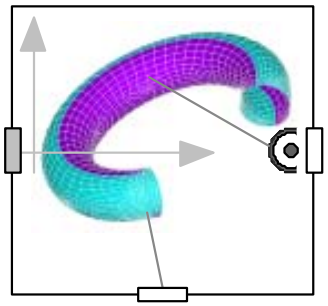
Animation

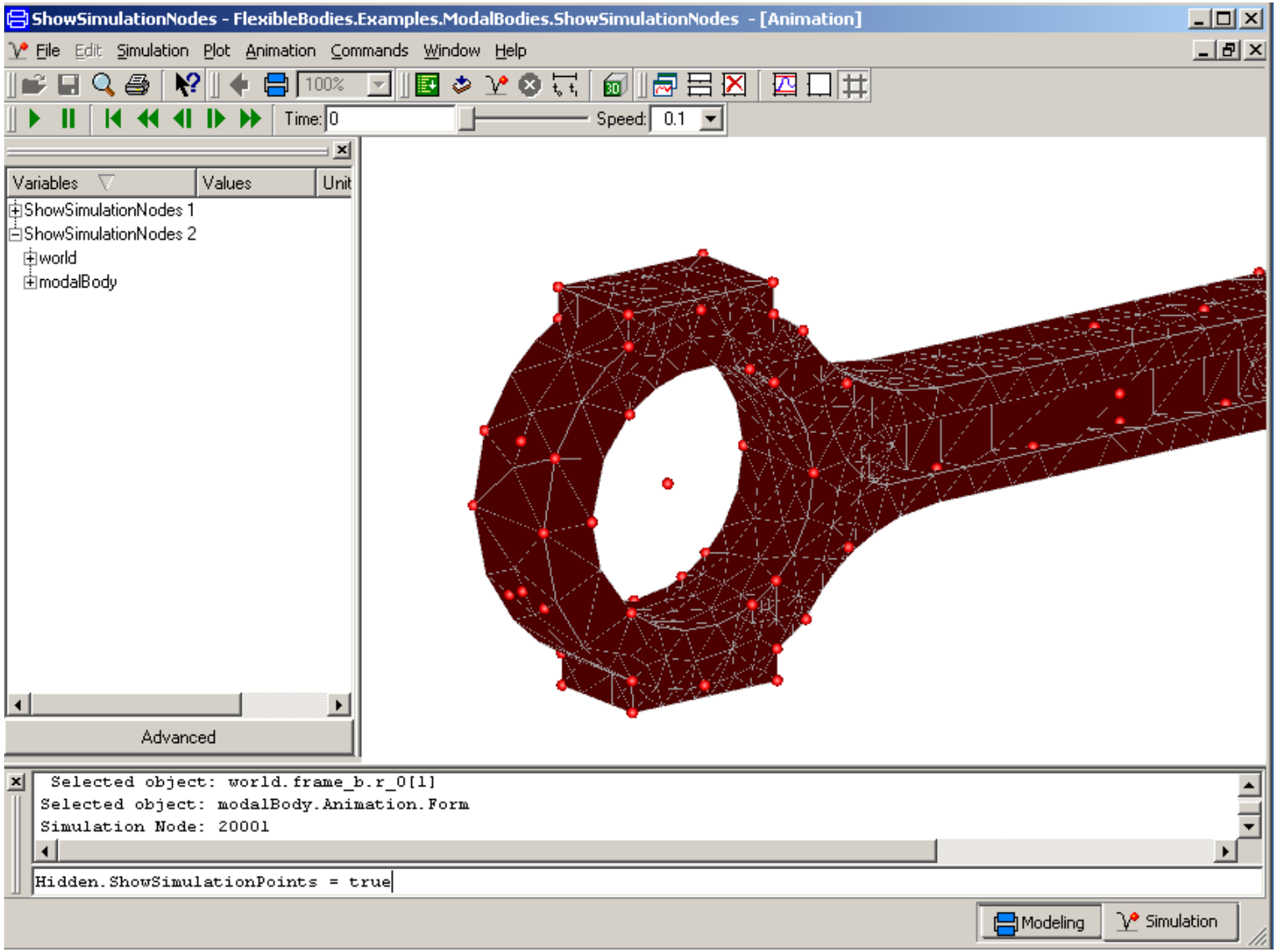
Solid animation	☐	Exaggeration factor to visualize deformation	1	Color	{0,0,255}	Specular coefficient	0.7
Wire frame animation	☑	Exaggeration factor to visualize deformation	200	Color	),155,155}	Specular coefficient	0

OK | Info | Cancel

FlexibleBodies Library: ModalBody menu set-up II

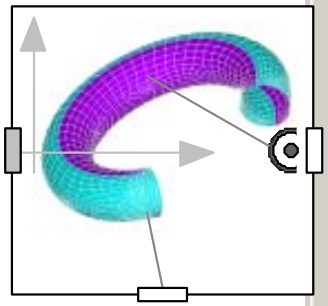
modalBody





FlexibleBodies Library: ModalBody menu set-up II

modalBody



rod in FlexibleBodies.Tutorial.Exercises.Rod

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 statesotherwise use 3 angles as potential states
sequence_angleStates	{1,2,3}	Sequence of rotations to rotate world frame into frame_aaround 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

dampingCoefficients	fill(0.02, nelastq)	Natural damping coefficients
userDamping	false	= true, if structural damping parameters shall be redefined (and taken instead of values in the SID-file)

Nodes Presentation

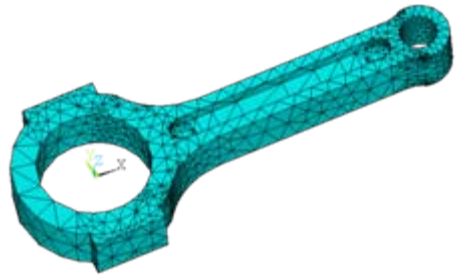
Node axes animation ☑ Length to present of axes at nodes world.defaultAxisLength/1.5 m

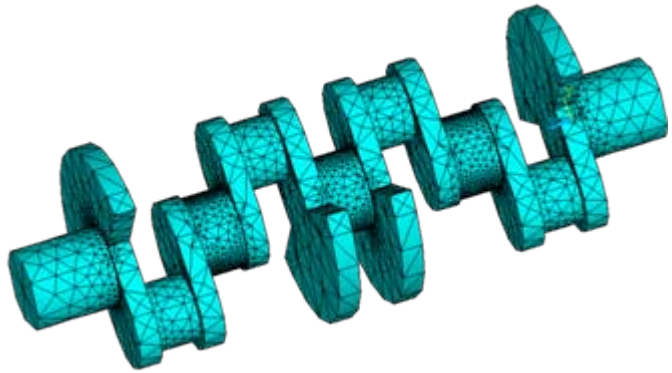
Axes Colors: of frame_ref {155,0,0} of nodes_clamped {0,128,0} of nodes_sphericalJoint {0,0,200}

OK | Info | Cancel

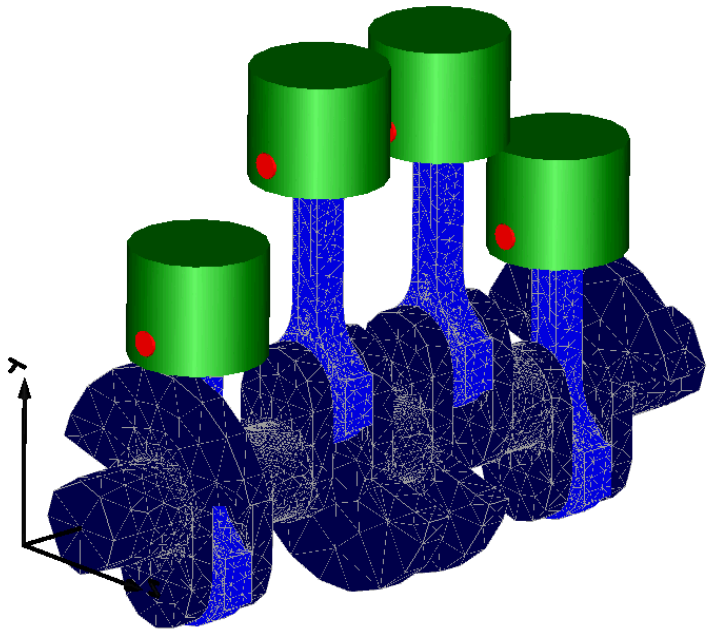
FlexibleBodies Library: 4 Cylinder Engine

- 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





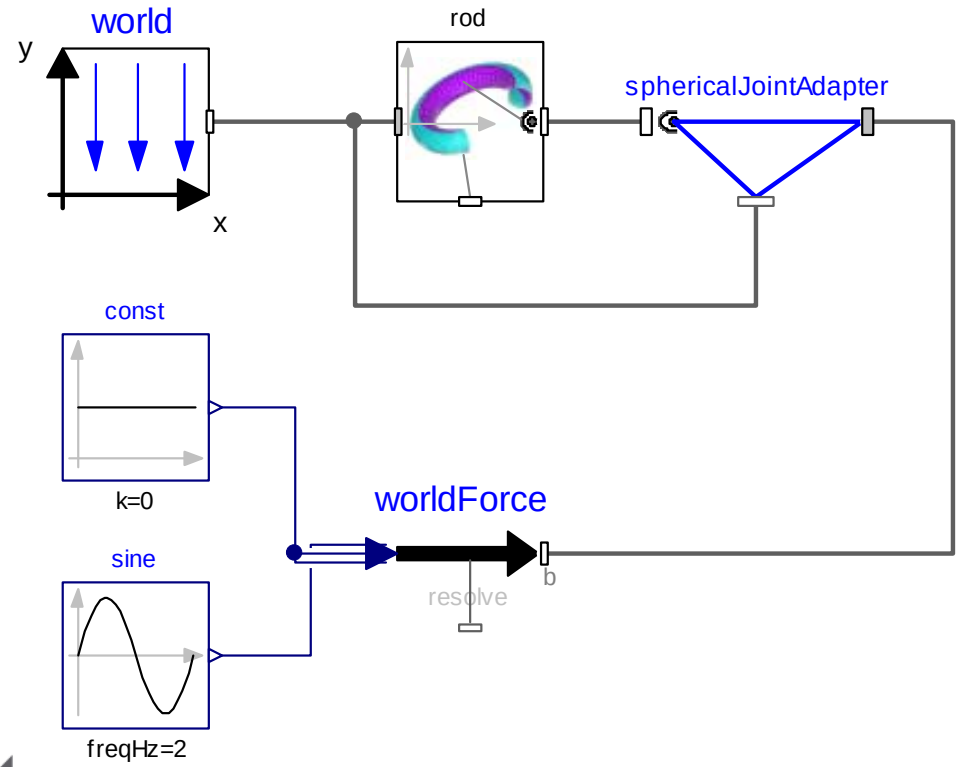
FlexibleBodies Library: 4 Cylinder Engine II

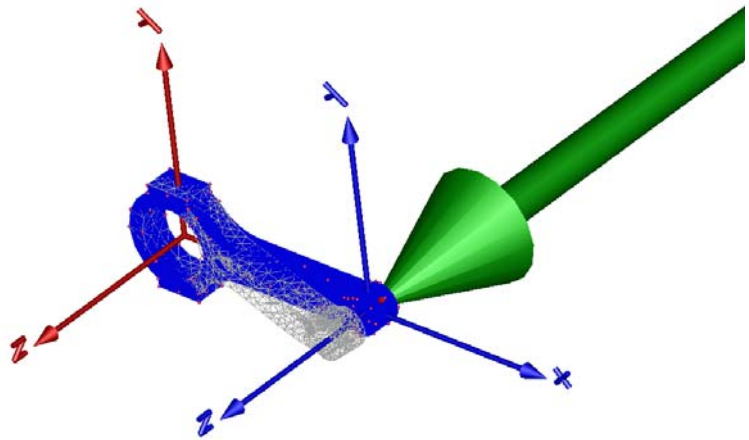


Contents

Example 4: Rod

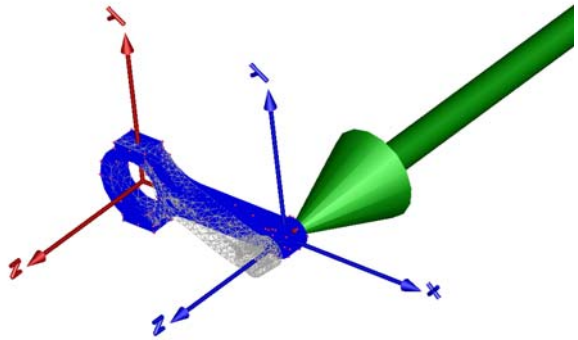
- 1st step:
  - introduce world and ModalBody- model
  - assign SID-file …/Extras/Data/rodV2.SID_FEM
  - assign OBJ-file …/Extras/Data/rodV2small.obj

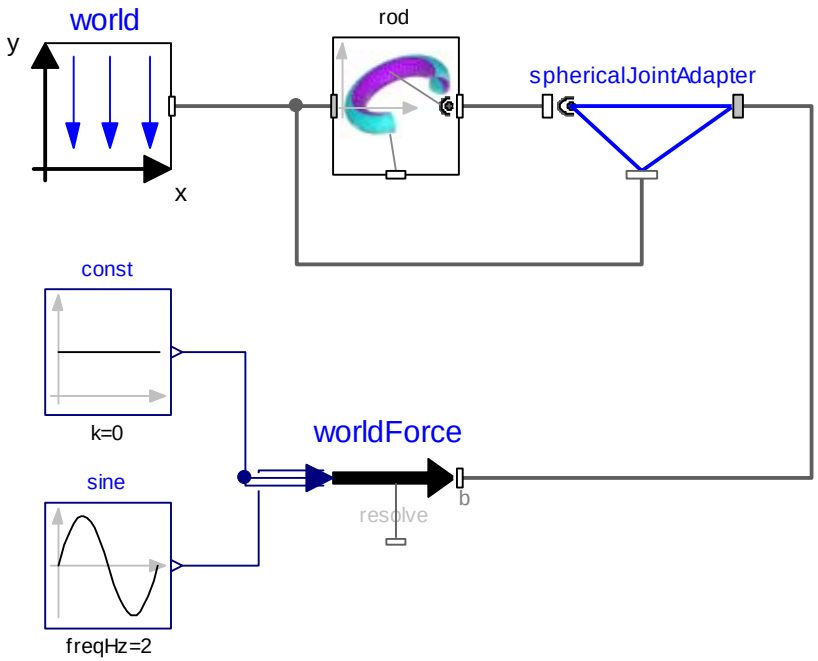




Example 4: Rod

- 2nd step: find node number to apply force
  - simulate trivial model from 1st step
  - zoom and pick node graphically
- 3rd step:
  - introduce node number
  - connect adapter
  - apply force





FlexibleBodies Library: Quo vadis?

- ModalBody: some tasks and goals
  - relies on an appopriate FE-preprocessor
    - a 3rd third party product
    - in principle agreements were made, but no contracts are yet signed
  - the 1st implementation uses additional c-code
    - there is no distribution process to support the application on real-time platforms today
  - the 2nd implemenation in pure Modelica code in preparation
    - good enough for moderate large models
  - 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