Modelling and Simulation of Rigid and Flexible Multibody Systems in Modelica

Tutorial at the SIMS2007, Göteborg (Särö), October 30th, 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

Modelica.Hydraulics

PneuLib

AirConditioning

Deutsches Zentrum für Luft- und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft
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**

```plaintext
equation
\[ \omega = \text{der}(\phi); \]
\[ a = \text{der}(\omega); \]
\[ J \cdot a = \text{flange}_a \cdot \tau_a + \text{flange}_b \cdot \tau_b; \]
```
**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" B;

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

\[
\dot{x} = f(x, t, \ldots)
\]
Modelica and Simulation Environments

Graphical editor for Modelica models

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

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

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

Free Modelica language

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

Deutsches Zentrum für Luft- und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft
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
Modelica Multibody Basics: Orientation

- Coordinate systems and their orientation

\[ \mathbf{R}_{12} \]

\[ \mathbf{e}_1^x \quad \mathbf{e}_1^y \quad \mathbf{e}_1^z \]
\[ \mathbf{e}_2^x \quad \mathbf{e}_2^y \quad \mathbf{e}_2^z \]

**Orientation object** \( \mathbf{R}_{12} \)
- describes orientation of coordinate system 2 wrt. 1
- holds

\[ \text{Real } T[3, 3] \text{ "Transformation matrix from world frame to local frame";} \]
\[ \text{ST.AngularVelocity } \omega[3] \]
- "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
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);
```
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

\[
\begin{align*}
\mathbf{a}_f & \quad \text{frame a} \\
\mathbf{a}_\tau & \quad \text{cut plane}
\end{align*}
\]

\[
\begin{align*}
\mathbf{R}^0_a & \quad \text{world frame}
\end{align*}
\]

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

\[
\text{fixedTranslation}
\]

\[
\begin{align*}
\text{frame}_b.r_0 &= \text{frame}_a.r_0 + \text{Frames.resolve}(\text{frame}_a.R, r); \\
\text{frame}_b.R &= \text{frame}_a.R; \\
/* Force and torque balance */ \\
\text{zeros}(3) &= \text{frame}_a.f + \text{frame}_b.f; \\
\text{zeros}(3) &= \text{frame}_a.t + \text{frame}_b.t + \text{cross}(r, \text{frame}_b.f);
\end{align*}
\]
Multibody Systems in Modelica

Modelica Multibody Basics: Components II

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

```modelica

// 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);
```

state selection
Modelica Multibody Basics: Elementary Components I

  - defines inertial frame, gravity, animation defaults

  - different resolution properties
  - interface to Real input functions and 1D mechanics
  - several spring/damper configurations

```
worldForceAndTorque
```

```
lineForceWithMass
```

```
springDamperParallel
```
Modelica Multibody Basics: Elementary Components II

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

  - Fixed, FixedTranslation and FixedRotation
Modelica Multibody Basics: Elementary Components III

  - Rigid bodies with predefined geometric shapes

  - for control and validation purposes

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
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
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
relative joint coordinates are used as states if possible

- default: stateSelect = StateSelect.prefer
- e.g. Multibody.Joints.Prismatic

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
    \[ 0 = f(\dot{x}, x, y, t, \ldots) \quad \text{dim}(f) = \text{dim}(x) + \text{dim}(y) \]
  - Equations are rearranged to get a sequence for model evaluation (Block Lower Triangle-partitioning)

\[
\begin{align*}
  f_1 &= \begin{pmatrix} z_1 & z_2 & z_3 & z_4 & z_5 \end{pmatrix} \\
  f_2 &= \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \end{pmatrix} \\
  f_3 &= \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \end{pmatrix} \\
  f_4 &= \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \end{pmatrix} \\
  f_5 &= \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \end{pmatrix}
\end{align*}
\]
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 (Block Lower Triangle-partitioning)
  - Equations to be differentiated are determined (Pantelides algorithm)
  - superflous potential states are deselected dynamically (dummy derivative method) \( \Rightarrow \) ODE:
    \[
    \dot{x} = f(x, t, \ldots)
    \]
review Translation Log in order to streamline simulation performance with model adjustments.
Modelica Multibody Advanced: Loops III

Planar loops

error message

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 2152.
The model includes the following hints:
- All Forces cannot be uniquely calculated.
The reason could be that the mechanism contains a planar loop that joints constrain the same motion. For planar loops, use one revolute joint per loop the option PlanarCyclic=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.
ERR01: 2 errors were found
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
Exercise 2: Hexapod I

The modelling concept

\[ r_p = \frac{L_p}{\sqrt{3}} \]

\[ L_b = 3.8 \, \text{m} \]

\[ h_{\text{start}} = 2 \, \text{m} \]

Inheritance:

- Hexapod
- Platform
- Basis
- Leg

Static Scenario
Exercise 2: Hexapod II

- Model Leg
  - Diagram layer
    - u
    - f
    - force
    - lineForceWithMass
  - frame_a
  - a
  - b
  - frame_b

- Icon layer
  - name
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=radiusPlatform/2;
```

![Model Platform diagram with coordinates for points a, b, and c, and parameters Lp, radiusPlatform, deltaX, and deltaY]
Exercise 2: Hexapod IV

Model Basis

```model Basis
  import SI = Modelica.SIunits;
  parameter SI.Length Lb=3.6 "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;
```

![Diagram of Hexapod IV](image)
Exercise 2: Hexapod V

Hexapod_StaticScenario

- actuator force in initial position?

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

A Basis

B

C

Platform

x

y

world

const[6]

const = [0, 0, 0, 0, 0, 0];

twoInputs = Modelica.Blocks.Math.TwoInputs;

velocity_sensor

resolve

angular and translatorial velocity ⇒ 6 outputs
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: Beams versus ModalBody

**Common Issues**
- Floating frame of reference
- Equations of motion
- SID-data-structure
- Standard-Input-Data: Wallrapp '94

**Disjunctive Issues**
- 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
  \[ r = r_R + c + u \]

- Modal approach
  \[ u(c, t) = \Phi(c) q(t) + \frac{1}{2} \begin{pmatrix} q^T \Phi_x \\ q^T \Phi_y \\ q^T \Phi_z \end{pmatrix} q \]

- Equations of motion

\[
\begin{pmatrix}
  mI_3 \\
  m\ddot d_{CM} \\
  C_t \\
  C_r
\end{pmatrix}
\begin{pmatrix}
  a_R \\
  \alpha_R \\
  \ddot q
\end{pmatrix} = \begin{pmatrix}
  M_e \\
  \ddot q
\end{pmatrix} = h_\omega - \begin{pmatrix}
  0 \\
  0 \\
  Ke q + De \dot q
\end{pmatrix} + \begin{pmatrix}
  f_a \\
  f_\alpha \\
  f_q
\end{pmatrix}
\]

Bremer/Pfeiffer '92, Schwertassek/Wallrapp '99
FlexibleBodies Library: Beam theory

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

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

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

\[
v(x, t) = \Phi_v(x) q_v(t)
\]

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

Bremer/Pfeiffer '92, Timoshenko '55
FlexibleBodies Library: Beam menu set-up I
FlexibleBodies Library: Beam menu set-up II

mandatory to define as much damping coefficients as modes
FlexibleBodies Library: Boundary Conditions I

⇒ Mechanical interpretation

\[ r(c, t) = r_R(t) + c + u(c, t) \]

⇒ let's say: frame of reference is pinned at frame_a with \( c = 0 \)

\[ \implies \text{motion of frame}_a \text{ is completely described by } r_R(t) \]

\[ (\text{and related orientation}) \]

\[ \implies u(c = 0, t) = 0 \quad \frac{\partial u}{\partial c}(c = 0, t) = 0 \]

⇒ 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

- static deflection: thrust force shortens beam and equivalent spring

<table>
<thead>
<tr>
<th></th>
<th>spring</th>
<th>beam</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 eigenmode</td>
<td>-10 cm</td>
<td>-8.1 cm</td>
<td>19 %</td>
</tr>
<tr>
<td>5 eigenmodes</td>
<td>-10 cm</td>
<td>-9.6 cm</td>
<td>4 %</td>
</tr>
<tr>
<td>10 eigenmodes</td>
<td>-10 cm</td>
<td>-9.6 cm</td>
<td>2 %</td>
</tr>
<tr>
<td>15 eigenmodes</td>
<td>-10 cm</td>
<td>-9.9 cm</td>
<td>1 %</td>
</tr>
</tbody>
</table>

comparison: deflections at the end
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: $u = \sin\left(\frac{2\pi x}{l}\right) + \sin\left(\frac{4\pi x}{3l}\right) + \ldots$

- 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 $15^\circ$ inclination
  - profil: squarepipe $0.4 \times 0.05 \times 0.01$ 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 $\xi = \{0.2, 0.4, 0.5, 0.6, 0.8\}$

- 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

multibody system diagram:

- `rot1`, `tran1`, `tran2`, `tran3` with transformation matrices `{b,0,0}`, `{0,-.25,0}`, `{-.05,0,0}`
- `rot1`, `tran1` with transformation matrices `{1,0,0}`
- `prismatic`, `spherical` joints
- `expSine`, `freqHz=0.25` wave
- `force`, `lineForce`
- `aFlow1`, `aFlow2`, `aFlow3`, `aFlow4` with transformation matrices `{0,-.2,0}`, `{0,-.2,0}`, `{0,-.2,0}`, `{0,-.2,0}`
- `Flow 1`, `Flow 2`, `Flow 3`, `Flow 4`
- `const`, `k=10`, `const1`, `k=0` constants
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 Menu](image)

- **Parameters**
  - `SID_fileName`: FlexibleBodies.Utilities.DataDirectory + "rodV2.SID_FEM" (File name of SID file describing the flexible body dynamics)
  - `WavefrontFile`: FlexibleBodies.Utilities.DataDirectory + "rodV2smal.obj" (File name of wavefront file describing the flexible body animation)

- **Simulation nodes** (a 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`: (file, 0) (Simulation nodes of `nodes_clamped` [constrain translation and rotation])

- **Animation**
  - **Solid animation**: Exaggeration factor to visualize deformation
  - **Wire frame animation**: Exaggeration factor to visualize deformation
FlexibleBodies Library: ModalBody menu set-up II
**FlexibleBodies Library: ModalBody menu set-up II**

![ModalBody Menu Set-up]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enforceStates true</td>
<td>If true, if elastic joint coordinates shall be used as states.</td>
</tr>
<tr>
<td>useQuaternions true</td>
<td>If true, if quaternions shall be used as potential states otherwise use 3 angles as potential states.</td>
</tr>
<tr>
<td>sequence_angleStates (1,2,3)</td>
<td>Sequence of rotations to rotate world frame into frame around the 3 angles used as potential states.</td>
</tr>
<tr>
<td>cj nominal</td>
<td>Nominal values of generalized coordinates (for numerical scaling).</td>
</tr>
<tr>
<td>cdj nominal</td>
<td>Nominal values of generalized velocities (for numerical scaling).</td>
</tr>
<tr>
<td>useDamping false</td>
<td>If true, if structural damping parameters shall be redefined (and taken instead of values in the SDF-file).</td>
</tr>
<tr>
<td>dampingCoefficients [0.02,0.05]</td>
<td>Natural damping coefficients.</td>
</tr>
</tbody>
</table>
FlexibleBodies Library: Animation

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

*www.fileformat.info/format/wavefrontobj/egff.htm
FlexibleBodies Library: 4 Cylinder Engine

- FEM
  - Crankshaft 85,342 nodes
  - Piston rod 12,531 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, ≈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 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 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