Lecture 18: Newton-Euler equations of motion, Modelica/Dymola: The Multibody library

- Newton-Euler equations of motion
  - Recap
  - Kinetic energy
  - Example

- Software
  - Dymola and the Modelica.Mechanics.Multibody library

Book: 7.3

Newton-Euler equations of motion

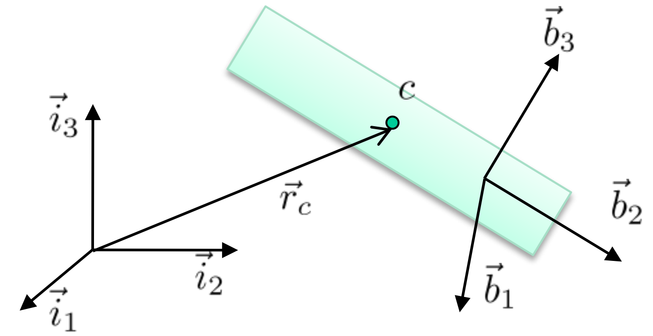
- Newton's law (for particle *k*)

$$m_k \vec{a}_k = \vec{F}^{(r)}$$

- Newton-Euler EoM for rigid bodies:
  - Integrate Newton's law over body, define center of mass
  - Define torque/moment and angular momentum to handle forces that give rotation about center of mass
  - Define inertia dyadic/matrix

$$\vec{F}_{bc} = m\vec{a}_c$$

$$\vec{T}_{bc} = \vec{M}_{b/c} \cdot \vec{\alpha}_{ib} + \vec{\omega}_{ib} \times \left(\vec{M}_{b/c} \cdot \vec{\omega}_{ib} \right)$$



(Here: Referenced to center of mass)

- Implemented in e.g. Dymola (Modelica.Multibody library)

Traits of Newton-Euler EoM

(and a preview: Lagrange EoM)

$$\vec{F}_{bc} = m\vec{a}_c$$

$$\vec{T}_{bc} = \vec{M}_{b/c} \cdot \vec{\alpha}_{ib} + \vec{\omega}_{ib} \times \left(\vec{M}_{b/c} \cdot \vec{\omega}_{ib}\right)$$

Newton-Euler EoM:

- Involves working with vectors
  - Lagrange: Algebraic manipulations
- Forces and moments are central
  - Lagrange: Energy and work are central
- All forces in the system must be considered
  - Lagrange: Forces of constraint are implicitly eliminated with the use of generalized coordinates (and generalized forces)
- Somewhat complicated to use by hand, but can be implemented in computer systems
  - Lagrange: Easier to do by hand, not suitable for complex systems

- d'Alembert's principle: Elimination of forces of constraint (Ch. 7.7)
  - Can simplify application of Newton-Euler EoM
    - Kane's EoM (Ch. 7.8, 7.9)
  - Starting point for Lagrange EoM (Ch. 8.2)

Inertia matrix

- Found for each rigid body by calculating

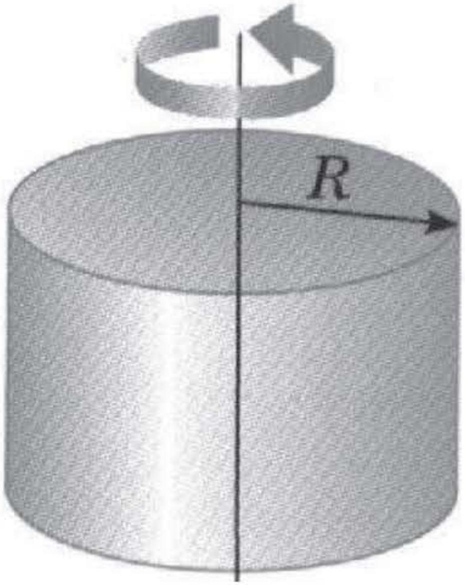
$$M^b_{b/c} = \int_b (\mathbf{r}^b)^\mathsf{T}\mathbf{r}^b I - \mathbf{r}^b(\mathbf{r}^b)^\mathsf{T}\mathrm{d}m = \int_b \begin{pmatrix} y^2+z^2 & -xy & -xz \\ -xy & x^2+z^2 & -yz \\ -xz & -yz & x^2+y^2 \end{pmatrix} \mathrm{d}m$$

- Constant in body-fixed coordinate system!
- Not constant in inertial coordinate system

$$M^i_{b/c} = R^i_b M^b_{b/c} (R^i_b)^\mathsf{T}$$

- Books and wikipedia have tables for common geometries, otherwise computer programs calculates, or can be calculated/identified based on experiments
- Typically, axis in body-system chosen as body symmetri axis, giving zeros in inertia matrix. If symmetric about all axis, the inertia matrix becomes diagonal.

Inertia matrix, examples



Homogeneous Disk

$$I_{disk} = \frac{1}{4} m r^2 \begin{bmatrix} 1 + \frac{1}{3}\frac{h}{r^2} & 0 & 0 \\ 0 & 1 + \frac{1}{3}\frac{h}{r^2} & 0 \\ 0 & 0 & \frac{1}{2} \end{bmatrix}$$

F/A-18

$$I = \begin{bmatrix} 23 & 0 & 2.97 \\ 0 & 15.13 & 0 \\ 2.97 & 0 & 16.99 \end{bmatrix} kslug - ft^2$$

1 slug = 14.6 kg
1 ft = 0.304 m

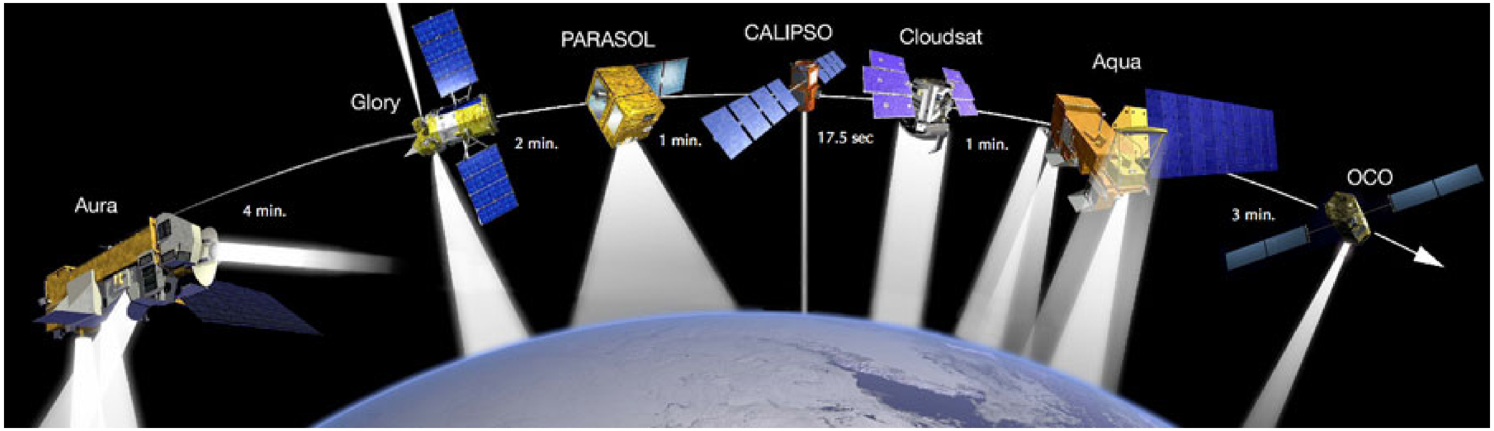
	Kinematics	Kinetics
	Derivatives of position and orientation as function of velocity and angular velocity	Derivatives of velocity and angular velocity as function of applied forces and torques

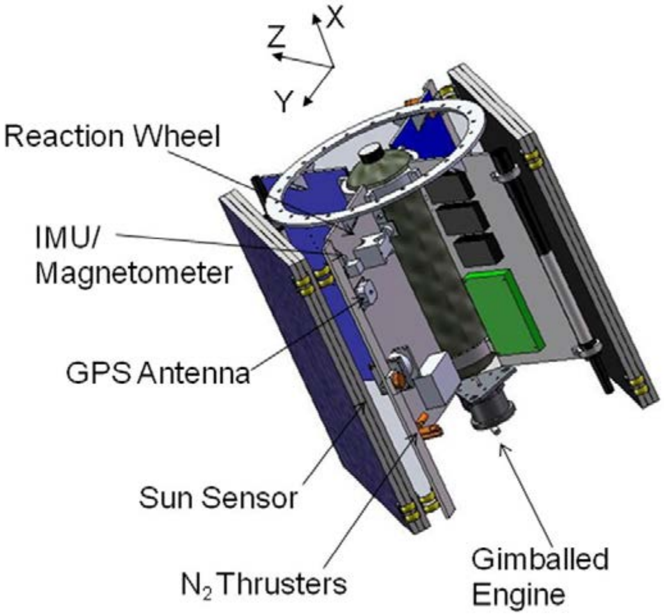
	Kinematics	Kinetics
	Derivatives of position and orientation as function of velocity and angular velocity	Derivatives of velocity and angular velocity as function of applied forces and torques
Translation	1D: $\dot{r} = v$ 3D: $\dot{\mathbf{r}}_c^i = \mathbf{v}_c^i$	1D: $m\dot{v} = F$ 3D: $m\dot{\mathbf{v}}_c^i = \mathbf{F}_{bc}^i$
	$\dot{\mathbf{r}}_c^i = \mathbf{v}_c^i = \mathbf{R}_b^i \mathbf{v}_c^b$	Usually convenient to have forces and velocities in body system: $m\left(\dot{\mathbf{v}}_c^b + \left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{v}_c^b\right) = \mathbf{F}_{bc}^b$
Rotation/ orientation	1D: $\dot{\theta} = \omega$ 3D: Depends on parameterization Rotation matrix: $\dot{\mathbf{R}}_b^i = \mathbf{R}_b^i \left(\boldsymbol{\omega}_{ib}^b\right)^\times$ Euler angles: $\dot{\boldsymbol{\phi}} = \mathbf{E}_d^{-1}(\boldsymbol{\phi})\boldsymbol{\omega}_{ib}^b$ Euler parameters: $\dot{\eta} = -\frac{1}{2}\boldsymbol{\epsilon}^\top \boldsymbol{\omega}_{ib}^b$ $\dot{\boldsymbol{\epsilon}} = \frac{1}{2}\left(\eta \mathbf{I} + \boldsymbol{\epsilon}^\times\right)\boldsymbol{\omega}_{ib}^b$	1D: $J\dot{\omega} = T$ 3D: $\mathbf{M}_{b/c}^b \dot{\boldsymbol{\omega}}_{ib}^b + \left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{M}_{b/c}^b \boldsymbol{\omega}_{ib}^b = \mathbf{T}_{bc}^b$

Note! By definition

$$\vec{v}_c := \frac{{}^i\mathrm{d}}{\mathrm{d}t}\vec{r}_c$$

Satellite attitude dynamics





http://satellite.mit.edu/

$$\vec{F}_{bc} = m\vec{a}_c$$

$$\vec{T}_{bc} = \vec{M}_{b/c} \cdot \vec{\alpha}_{ib} + \vec{\omega}_{ib} \times \left(\vec{M}_{b/c} \cdot \vec{\omega}_{ib}\right)$$

Airplane EoM (from book about airplane dynamics)

Force equations

$$X - mgS_\theta = m(\dot{u} + qw - rv)$$
$$Y + mgC_\theta S_\Phi = m(\dot{v} + ru - pw)$$
$$Z + mgC_\theta C_\Phi = m(\dot{w} + pv - qu)$$

$$m\left(\dot{\mathbf{v}}_c^b + \left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{v}_c^b\right) = \mathbf{F}_{bc}^b$$

Moment equations

$$L = I_x\dot{p} - I_{xz}\dot{r} + qr(I_z - I_y) - I_{xz}pq$$
$$M = I_y\dot{q} + rp(I_x - I_z) + I_{xz}(p^2 - r^2)$$
$$N = -I_{xz}\dot{p} + I_z\dot{r} + pq(I_y - I_x) + I_{xz}qr$$

$$\mathbf{M}_{b/c}^b\dot{\boldsymbol{\omega}}_{ib}^b + \left(\boldsymbol{\omega}_{ib}^b\right)^\times \mathbf{M}_{b/c}^b\boldsymbol{\omega}_{ib}^b = \mathbf{T}_{bc}^b$$

Body angular velocities in terms of Euler angles and Euler rates

$$p = \dot{\Phi} - \dot{\psi}S_\theta$$
$$q = \dot{\theta}C_\Phi + \dot{\psi}C_\theta S_\Phi$$
$$r = \dot{\psi}C_\theta C_\Phi - \dot{\theta}S_\Phi$$

Euler rates in terms of Euler angles and body angular velocities

$$\dot{\theta} = qC_\Phi - rS_\Phi$$
$$\dot{\Phi} = p + qS_\Phi T_\theta + rC_\Phi T_\theta$$
$$\dot{\psi} = (qS_\Phi + rC_\Phi)\sec\theta$$

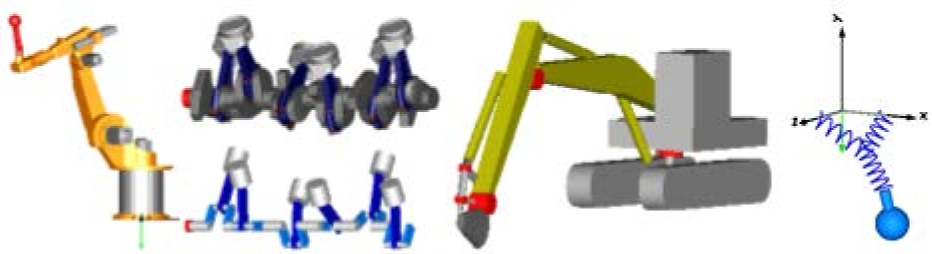
$$\dot{\boldsymbol{\phi}} = \mathbf{E}_d^{-1}(\boldsymbol{\phi})\boldsymbol{\omega}_{ib}^b$$

Velocity of aircraft in the fixed frame in terms of Euler angles and body velocity components

$$\begin{bmatrix} \dfrac{dx}{dt} \\ \dfrac{dy}{dt} \\ \dfrac{dz}{dt} \end{bmatrix} = \begin{bmatrix} C_\theta C_\psi & S_\Phi S_\theta C_\psi - C_\Phi S_\psi & C_\Phi S_\theta C_\psi + S_\Phi S_\psi \\ C_\theta S_\psi & S_\Phi S_\theta S_\psi + C_\Phi C_\psi & C_\Phi S_\theta S_\psi - S_\Phi C_\psi \\ -S_\theta & S_\Phi C_\theta & C_\Phi C_\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$\dot{\mathbf{r}}_c^i = \mathbf{v}_c^i = \mathbf{R}_b^i\mathbf{v}_c^b$$

Flight Stability and Automatic Control
R. C. Nelson, 2nd Edition

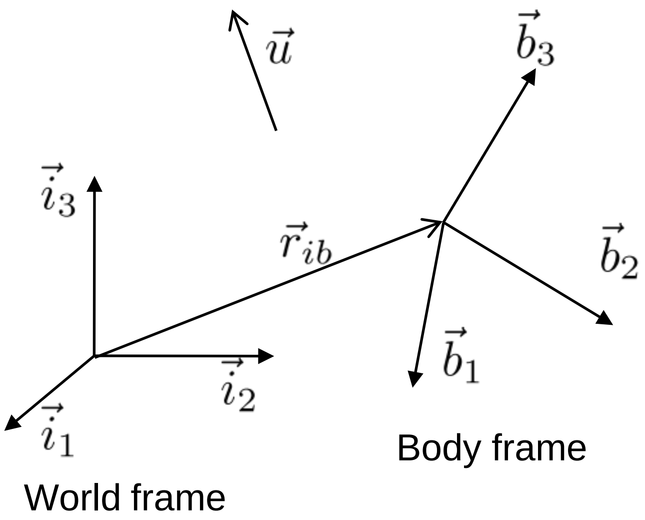


Modelica Multibody introduction

Adapted from slides by Andreas Heckmann, DLR

Modelica Multibody: Orientation

- Orientation and position of coordinate systems (*frames*)

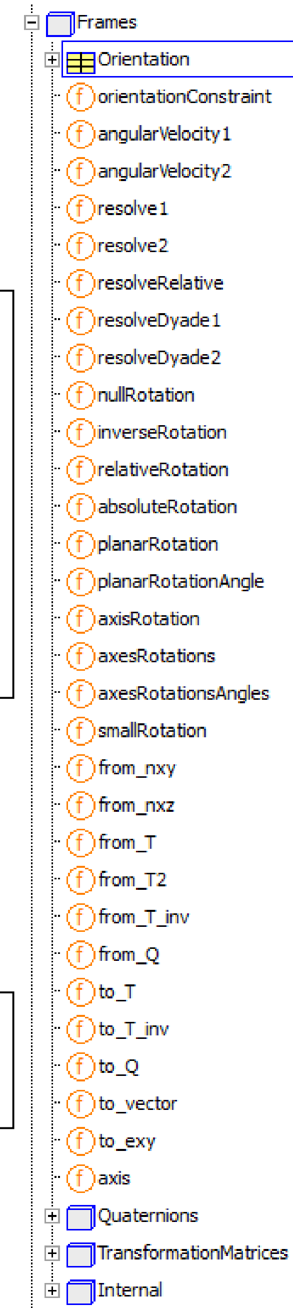



```
model ...
 import Modelica.Mechanics.MultiBody.Frames;
 Frames.Orientation Rib;
 Real[3] ui "vector u resolved in frame i";
 Real[3] ub "vector u resolved in frame b";
 ...
equation
 ...
 ui = Frames.resolve1(Rib, ub); // ui = Rib*ub
 ub = Frames.resolve2(Rib, ui); // ub = Rib'*ui
```

- Orientation object $\mathbf{R}_b^i$
  - Describes orientation of system *b* wrt *i* (transforms from *b* to *i*)
  - Contains:

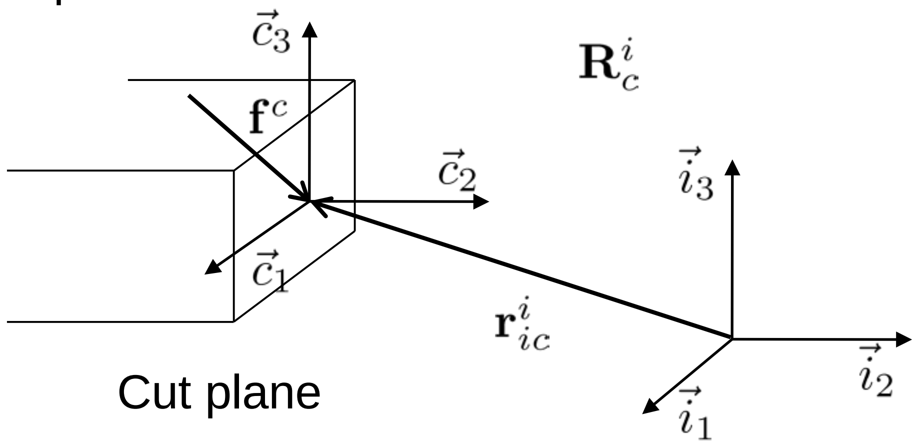
```
Real T[3, 3] "Transformation matrix from world frame to local frame";
SI.AngularVelocity w[3]
   "Absolute angular velocity of local frame, resolved in local frame";
```

  - Can be specified using Euler angles or Euler parameters/quaternions
- Many functions to operate on orientation objects



Modelica Multibody: Connectors I

- Connectors: To connect different rigid bodies
  - Position is resolved in world frame
  - Forces and torques are resolved in local frame



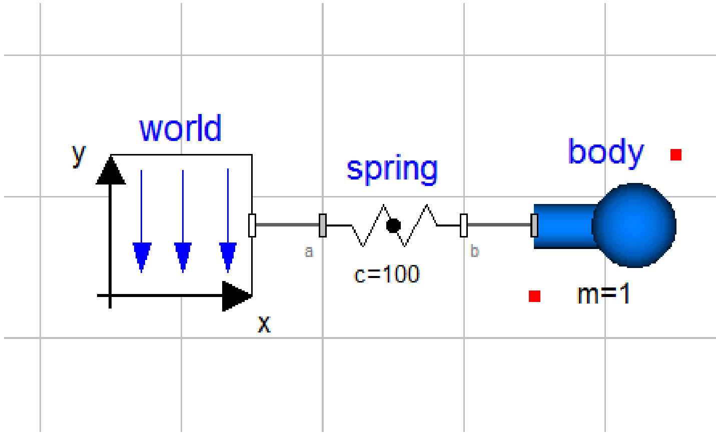
"No flow" variables

```
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";
  flow SI.Torque t[3] "Cut-torque resolved in connector frame";
end Frame;
```

"Flow" variables

Modelica Multibody: Connectors II

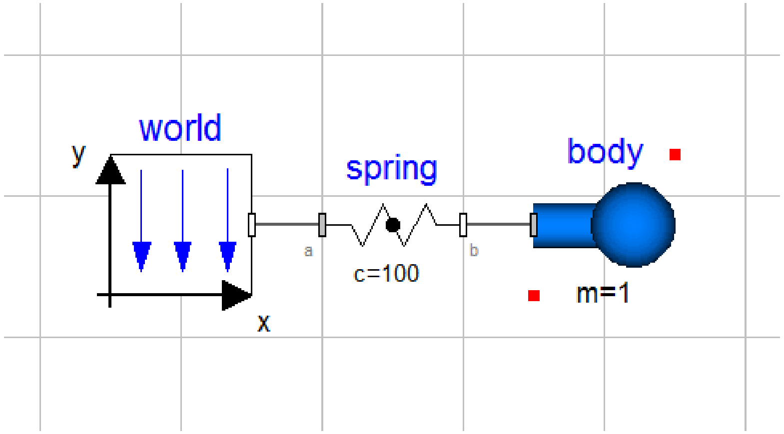
```
model SpringMass
 inner Modelica.Mechanics.MultiBody.World world;
 Modelica.Mechanics.MultiBody.Parts.Body body(
    m=1,
    r_CM={0,1,0}, // In frame a
    r_0(fixed=true, start={0,0.5,0})); // In world frame
Modelica.Mechanics.MultiBody.Forces.Spring spring(c=100);
equation
 connect(spring.frame_a, world.frame_b);
 connect(spring.frame_b, body.frame_a);
end SpringMass;
```



- Connection rules
  - *Non-flow* variables set equal (that is: frames coincides)
  - *Flow* variables sum to zero (Newton's third law)

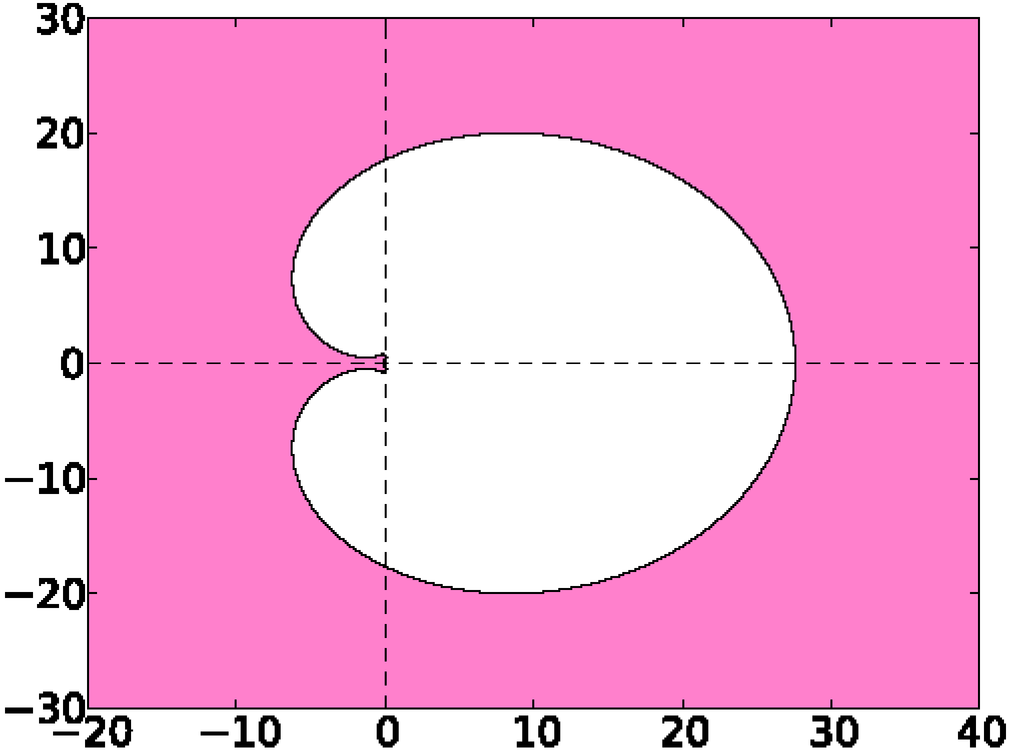
Modelica.Multibody: Generic body component

- Make SpringMass in Dymola



- Show
  - Parameters (mass, r_cm = (0,-0.5,0), Inertia matrix)
  - Initial values
  - Euler angles

Stability Region BDF



BDF 6

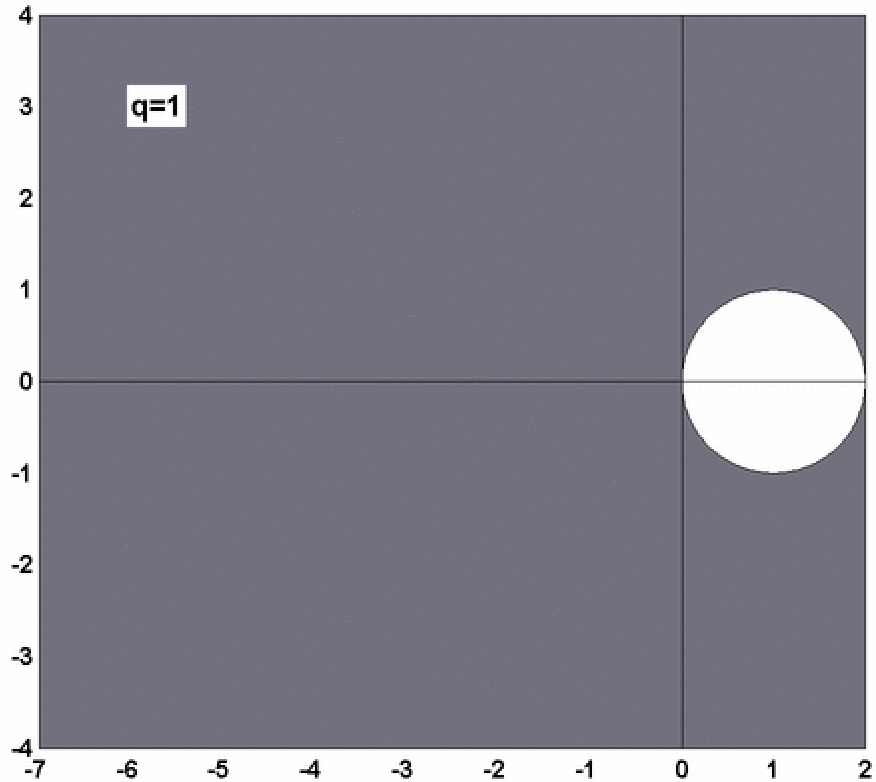
Padé approximations to e^s

$m \backslash k$	0	1	2	3
0	$\frac{1}{1}$	$\frac{1+s}{1}$	$\frac{1+s+\frac{1}{2}s^2}{1}$	$\frac{1+s+\frac{1}{2}s^2+\frac{1}{6}s^3}{1}$
1	$\frac{1}{1-s}$	$\frac{1+\frac{1}{2}s}{1-\frac{1}{2}s}$	$\frac{1+\frac{2}{3}s+\frac{1}{6}s^2}{1-\frac{1}{3}s}$	$\frac{1+\frac{3}{4}s+\frac{1}{4}s^2+\frac{1}{24}s^3}{1-\frac{1}{4}s}$
2	$\frac{1}{1-s+\frac{1}{2}s^2}$	$\frac{1+\frac{1}{3}s}{1-\frac{2}{3}s+\frac{1}{6}s^2}$	$\frac{1+\frac{1}{2}s+\frac{1}{12}s^2}{1-\frac{1}{2}s+\frac{1}{12}s^2}$	$\frac{1+\frac{3}{5}s+\frac{3}{20}s^2+\frac{1}{60}s^3}{1-\frac{2}{5}s+\frac{1}{20}s^2}$
3	$\frac{1}{1-s+\frac{1}{2}s^2-\frac{1}{6}s^3}$	$\frac{1+\frac{1}{4}s}{1-\frac{3}{4}s+\frac{1}{4}s^2-\frac{1}{24}s^3}$	$\frac{1+\frac{2}{5}s+\frac{1}{20}s^2}{1-\frac{3}{5}s+\frac{3}{20}s^2-\frac{1}{60}s^3}$	$\frac{1+\frac{1}{2}s+\frac{1}{10}s^2+\frac{1}{120}s^3}{1-\frac{1}{2}s+\frac{1}{10}s^2-\frac{1}{120}s^3}$

L-stable L-stable A-stable

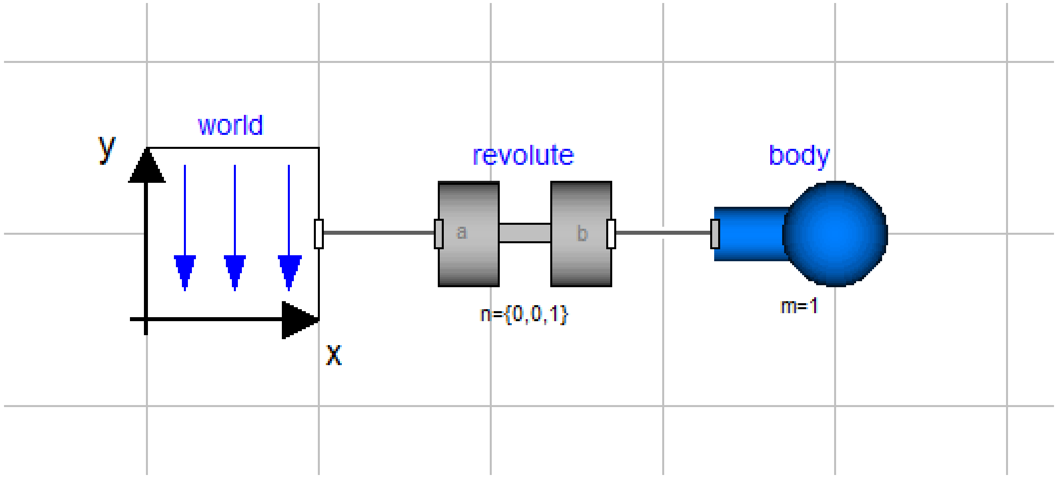
- $m = 0$: Explicit Runge-Kutta methods with $p = \sigma$
- $m = k$: Gauss, Lobatto IIIA/IIIB (incl. implicit mid-point, trapezoidal)
- $m = k+1$: Radau-methods (incl. implicit Euler)
- $m = k+2$: Lobatto IIIC

Stability region for Adams-Moulton



Modelica.Multibody: Rotations

- Make simple pendulum

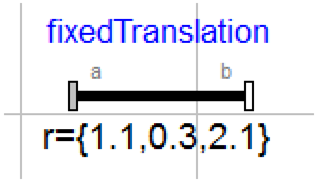


- Show body.frame_a.R (rotation object)

Modelica Multibody: Kinematics

- Equations inside the component provide relations between the connector variables on position level

- Example: MultiBody.Parts.FixedTranslation
  - Fixed translation of frame_b with respect to frame_a




```
model FixedTranslation
  "Fixed translation of frame_b with respect to frame_a"
  ...
equation

  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);
end FixedTranslation;
```

- Dymola differentiates these equations twice for (velocity and) accelerations

Modelica Multibody: Kinetics

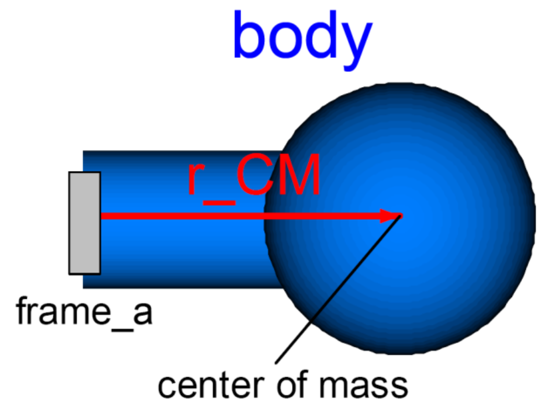
- Newton-Euler equations
  - Accelerations

$$\vec{a}_p = \vec{a}_o + \vec{\alpha}_{ib} \times \vec{r} + \vec{\omega}_{ib} \times (\vec{\omega}_{ib} \times \vec{r}), \quad \vec{r} \text{ fixed.}$$

  - Kinetics

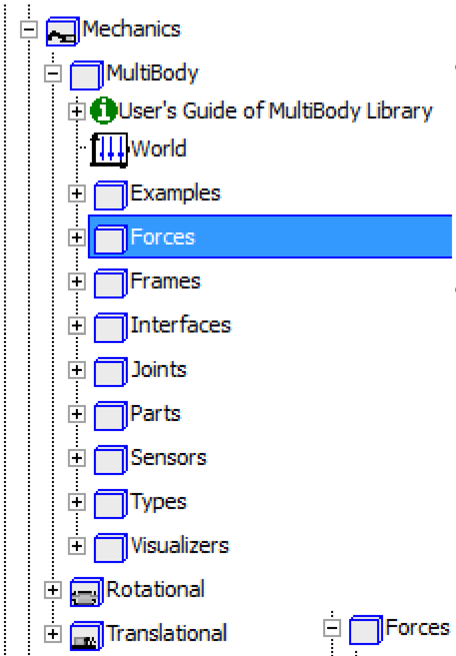
$$\vec{F}_{bc} = m\vec{a}_c$$

$$\vec{T}_{bc} = \vec{M}_{b/c} \cdot \vec{\alpha}_{ib} + \vec{\omega}_{ib} \times (\vec{M}_{b/c} \cdot \vec{r})$$

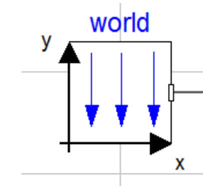



```
model body
  "Rigid body with mass, inertia tensor and one frame connector (12 potential states)"
  ...
equation
  // translational kinematic differential equations
  v_0 = der(frame_a.r_0);                            // r_0, v_0 resolved in world frame
  a_a = Frames.resolve2(frame_a.R,der(v_0)); // a_a resolved in frame_a

  // rotational kinematic differential equations
  w_a = 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 - g_a);
            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);
end body;
```

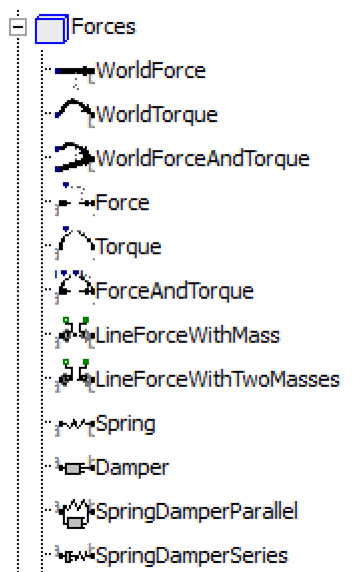
Modelica Multibody: Elementary components I



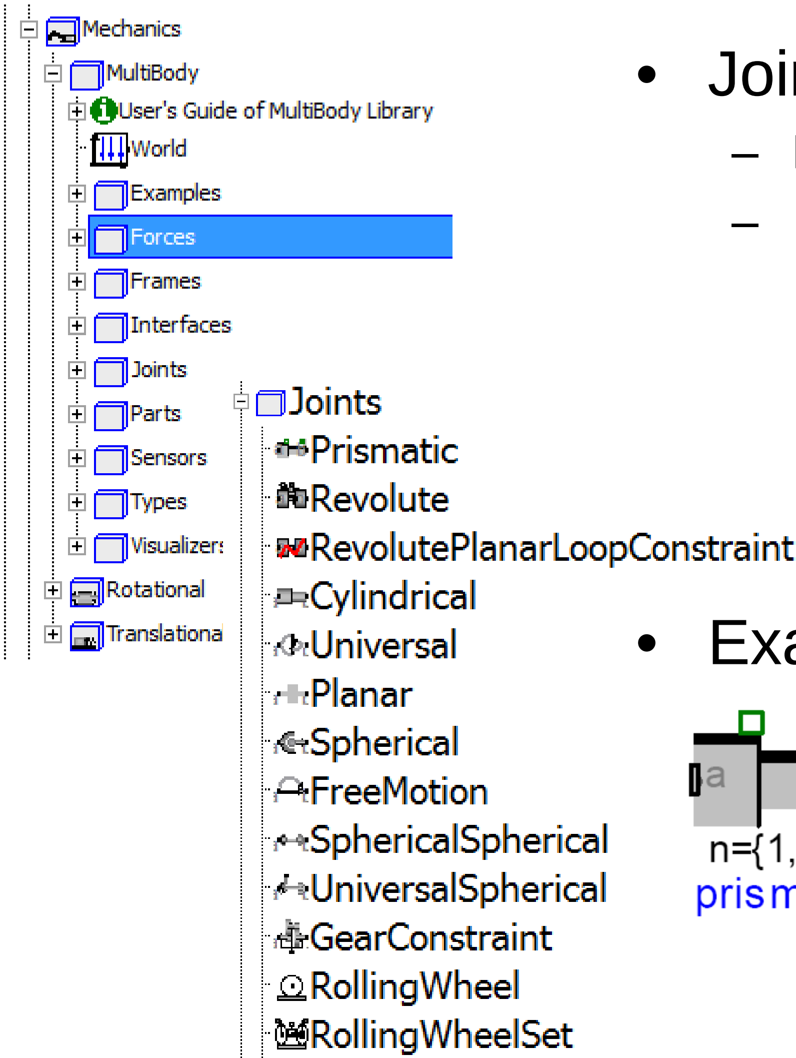
- Modelica.Mechanics.Multibody.World
  – Defines inertial frame, gravity, animation defaults



- Modelica.Mechanics.MultiBody.Forces
  – External forces and torques, resolved in body- or inertial frame
  – Interface to Real input functions
  – Several spring/damper configurations

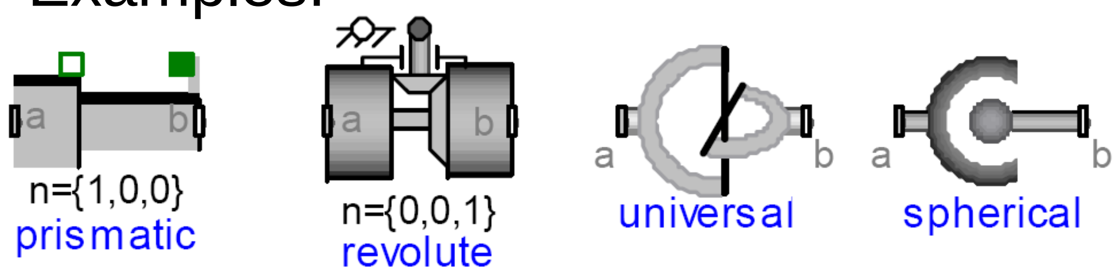


Modelica Multibody: Elementary components II

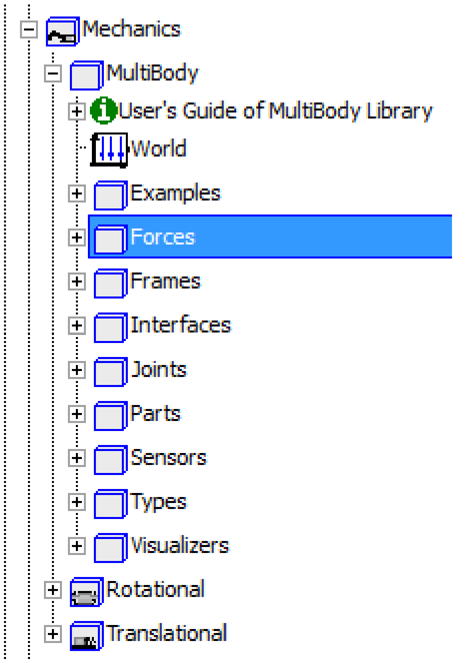


- Joints
  - Define specific degrees of freedom
  - Interface to 1D mechanics
    - (rotational/translational)

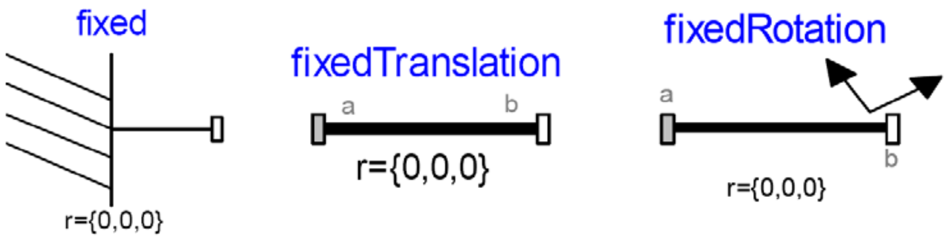
- Examples:



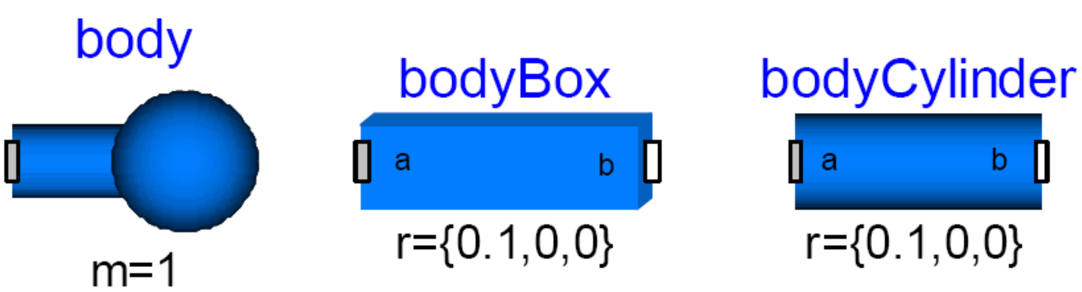
Modelica Multibody: Elementary components II



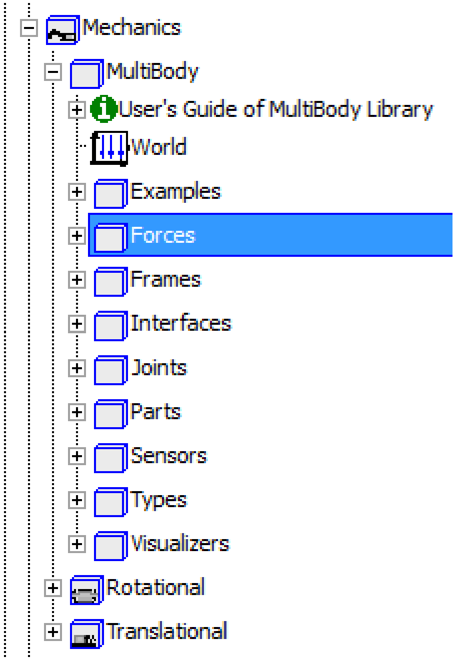
- Modelica.Mechanics.MultiBody.Parts
  - Fixed, Fixed Translation and Fixed Rotation



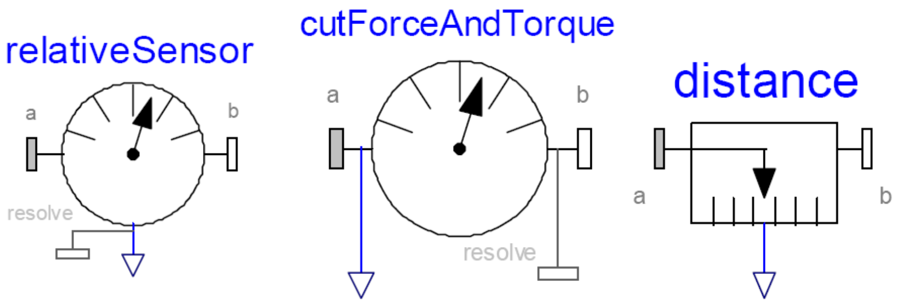
- Rigid bodies with predefined geometric shapes



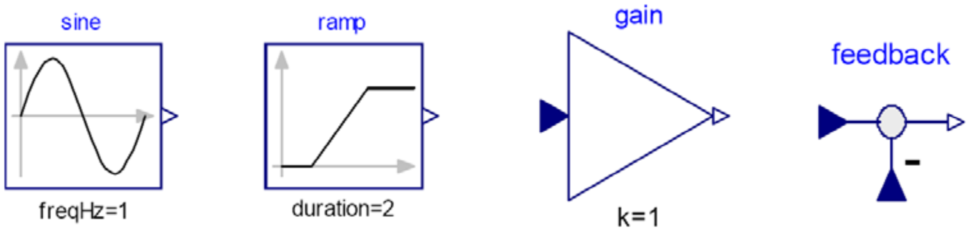
Modelica Multibody: Elementary components II



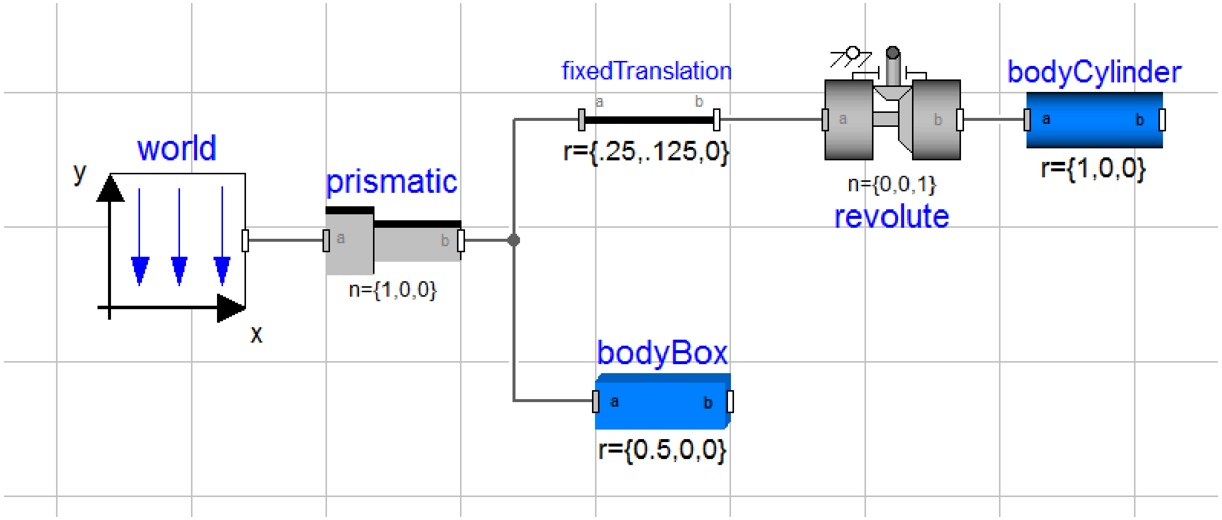
- Modelica.Mechanics.Multibody.Sensors
  - For control and validation purposes



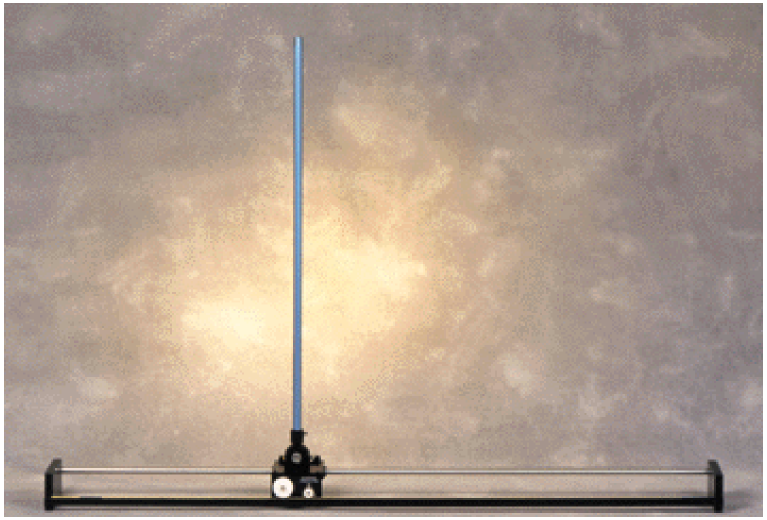
- Modelica.Blocks.Sources + Modelica.Blocks.Math



Example: Inverted pendulum, modeling



- Box: 0.5m x 0.25m x 0.25m
- Cylinder: L = 1m, r = 0.05m



Example: Inverted pendulum, PD control