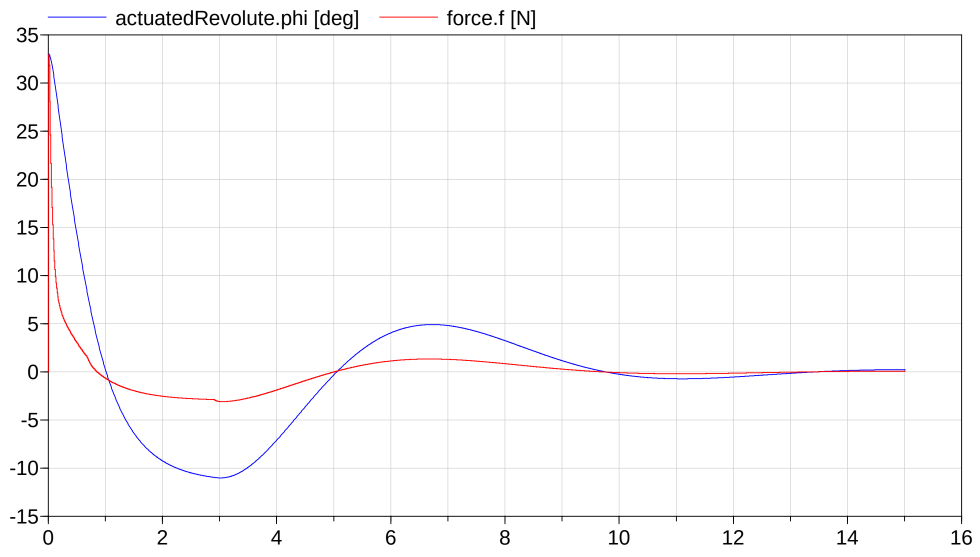
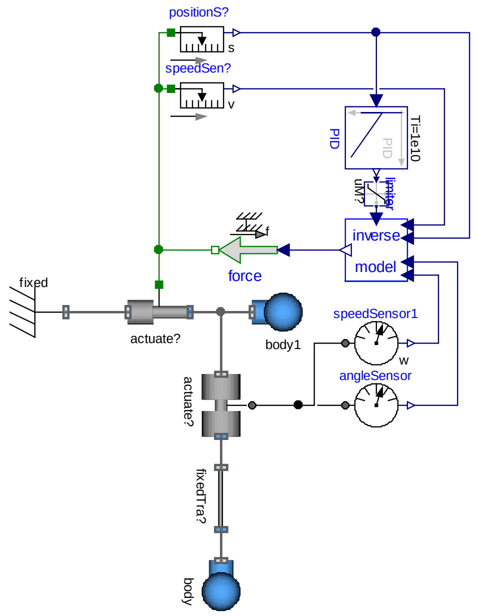
Virtual Physics
Equation-Based Modeling

TUM, January 20, 2015

Control and Model Inversion



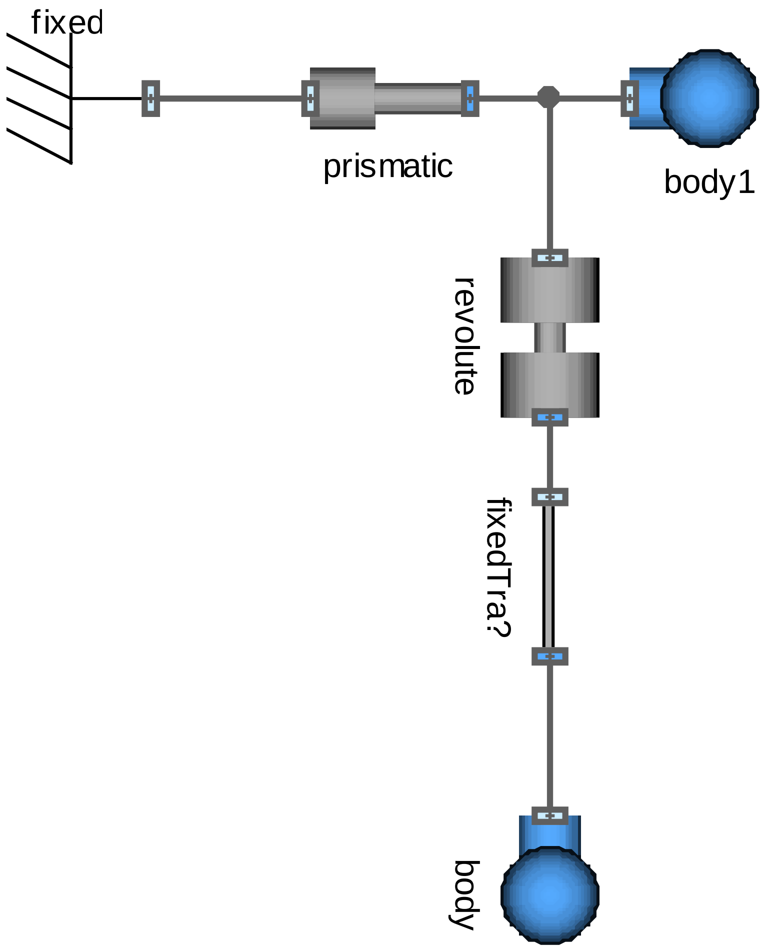


Dr. Dirk Zimmer

German Aerospace Center (DLR), Robotics and Mechatronics Centre

Inverse Pendulum

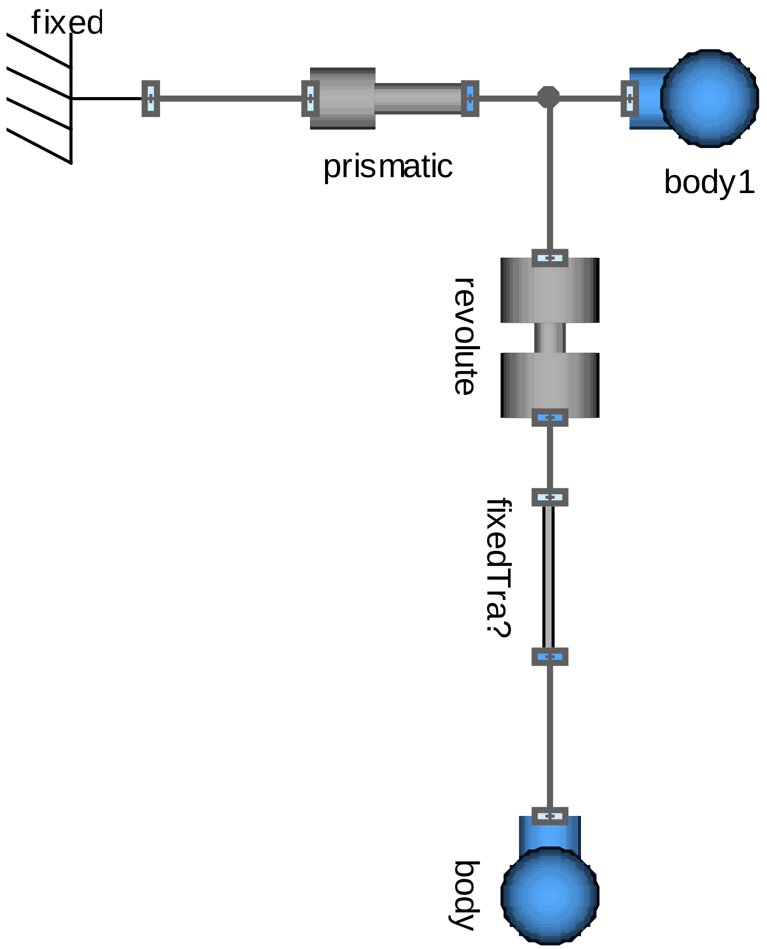
- In this lecture, we want to control an inverse pendulum.
- This means, that by moving the platform we want to keep the pendulum in upright position (balancing task).
- The inverse pendulum is essentially represented by the crane-crab model of the previous lectures (see Exercise 2)



Inverse Pendulum

Robotics and Mechatronics Centre

- The task is to bring and keep the pendulum in an upright position (φ = 180°).
- To this end, we need to apply an appropriate force on the prismatic joint.
- This is not an easy task (even humans have problems balancing a pen on a fingertip).



Stability Analysis

- First, let us analyze the stability of the system with the pendulum being in upright position.
- Fortunately we know the state-space form from Exercise 2:

$ds/dt = v;$

$d\varphi/dt = \omega;$

$dv/dt = (\sin(\varphi)\cdot M_P\cdot R\cdot \omega^2 - \cos(\varphi)\cdot\sin(\varphi)\cdot M_P\cdot G) / (M_S + M_P\cdot\sin(\varphi)^2);$

$d\omega/dt = (\sin(\varphi)\cdot G - \cos(\varphi)\cdot dv/dt)/R;$

with the states: s, φ, v, ω and the parameters: R (Radius), M_S (sliding mass), M_P (pendulum mass), and G := -9.81 (Gravitational Acc.)

Stability Analysis

Robotics and Mechatronics Centre

- We linearize at φ = 180° and ω = 0°/s:

  $d\,(dv/dt)\,/d\varphi = -M_P \cdot G\,/\,M_S$

  $d\,(d\omega/dt)/d\varphi = (-G - (M_P \cdot G)/\,M_S)/R = -G\,(1 + M_P/M_S)/R \;\; > \;\; 0$

- Hence

$$\begin{bmatrix} ds/dt \\ dv/dt \\ d\varphi/dt \\ d\omega/dt \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -M_P \cdot G/M_S & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -G(1-M_P/M_S)/R & 0 \end{bmatrix} \begin{bmatrix} s \\ v \\ \varphi \\ \omega \end{bmatrix}$$

Stability Analysis

Robotics and Mechatronics Centre

- We can safely remove the translational part from our stability analysis:

$$\begin{bmatrix} \cancel{ds/dt} \\ \cancel{dv/dt} \\ d\varphi/dt \\ d\omega/dt \end{bmatrix} = \begin{bmatrix} \cancel{0} & \cancel{1} & \cancel{0} & \cancel{0} \\ \cancel{0} & \cancel{0} & \cancel{M_P \cdot G/M_S} & \cancel{0} \\ \cancel{0} & \cancel{0} & 0 & 1 \\ \cancel{0} & \cancel{0} & -G(1+M_P/M_S)/R & 0 \end{bmatrix} \begin{bmatrix} \cancel{s} \\ \cancel{v} \\ \varphi \\ \omega \end{bmatrix}$$

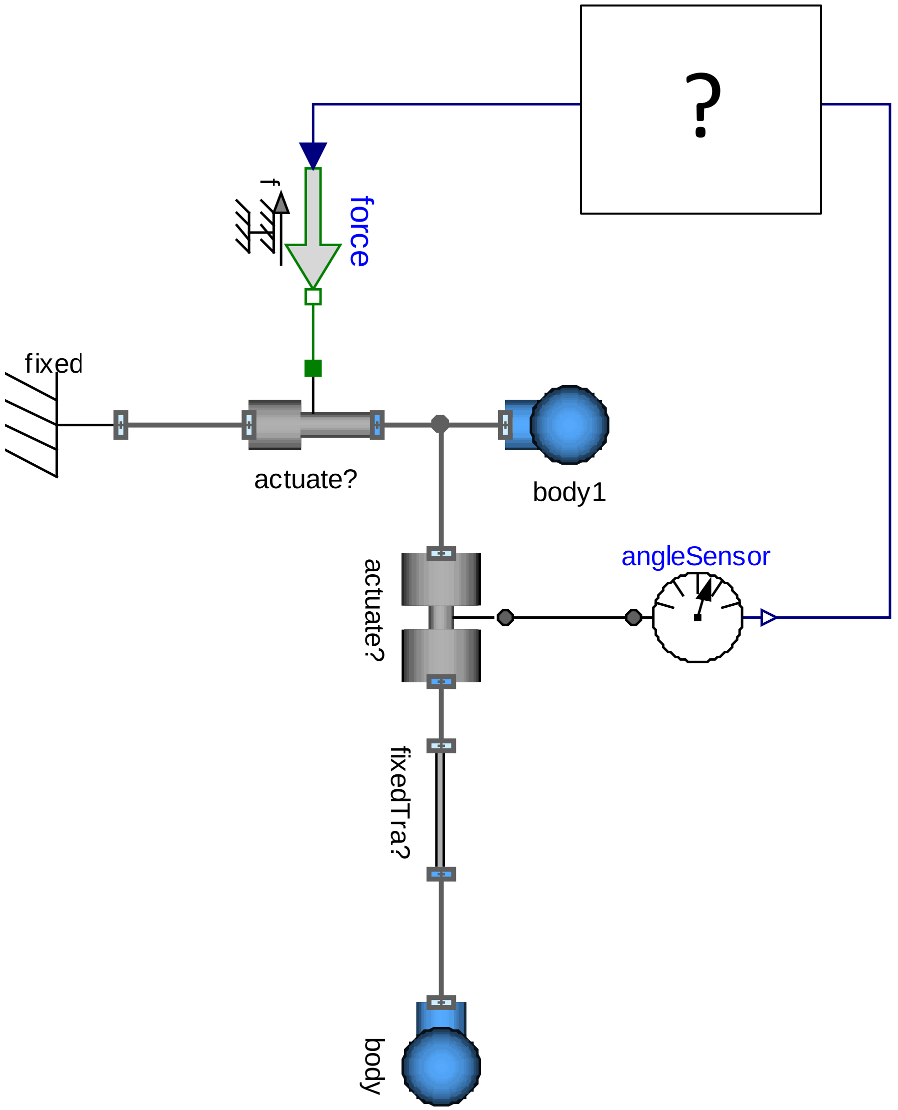
- There remains:

$$\begin{bmatrix} d\varphi/dt \\ d\omega/dt \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -G(1+M_P/M_S)/R & 0 \end{bmatrix} \begin{bmatrix} \varphi \\ \omega \end{bmatrix}$$

- The system has +/- $\sqrt{(-G(1+M_P/M_S)/R)}$ as eigenvalues and is therefore unstable at $\varphi = 180°$ since $-G(1+M_P/M_S)/R$ is positive.
  (The system is marginally stable at $\varphi = 0°$)

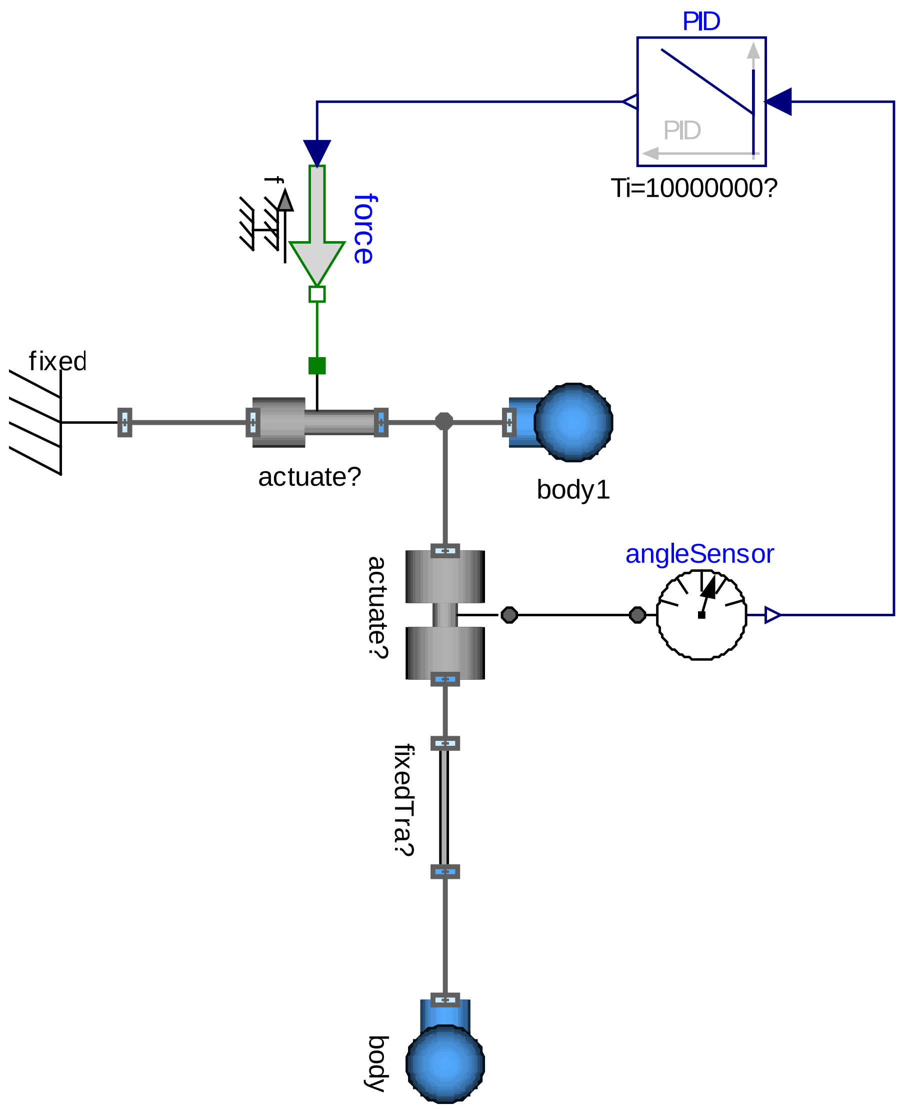
Inverse Pendulum

- In order to stabilize the pendulum in upright position, a controller shall measure the angle of the pendulum and compute the required force out of it.
- Many methods are available to design such a controller. Studying this, is far beyond the scope of our lecture.
- However, we are going to examine a relatively simple method here.

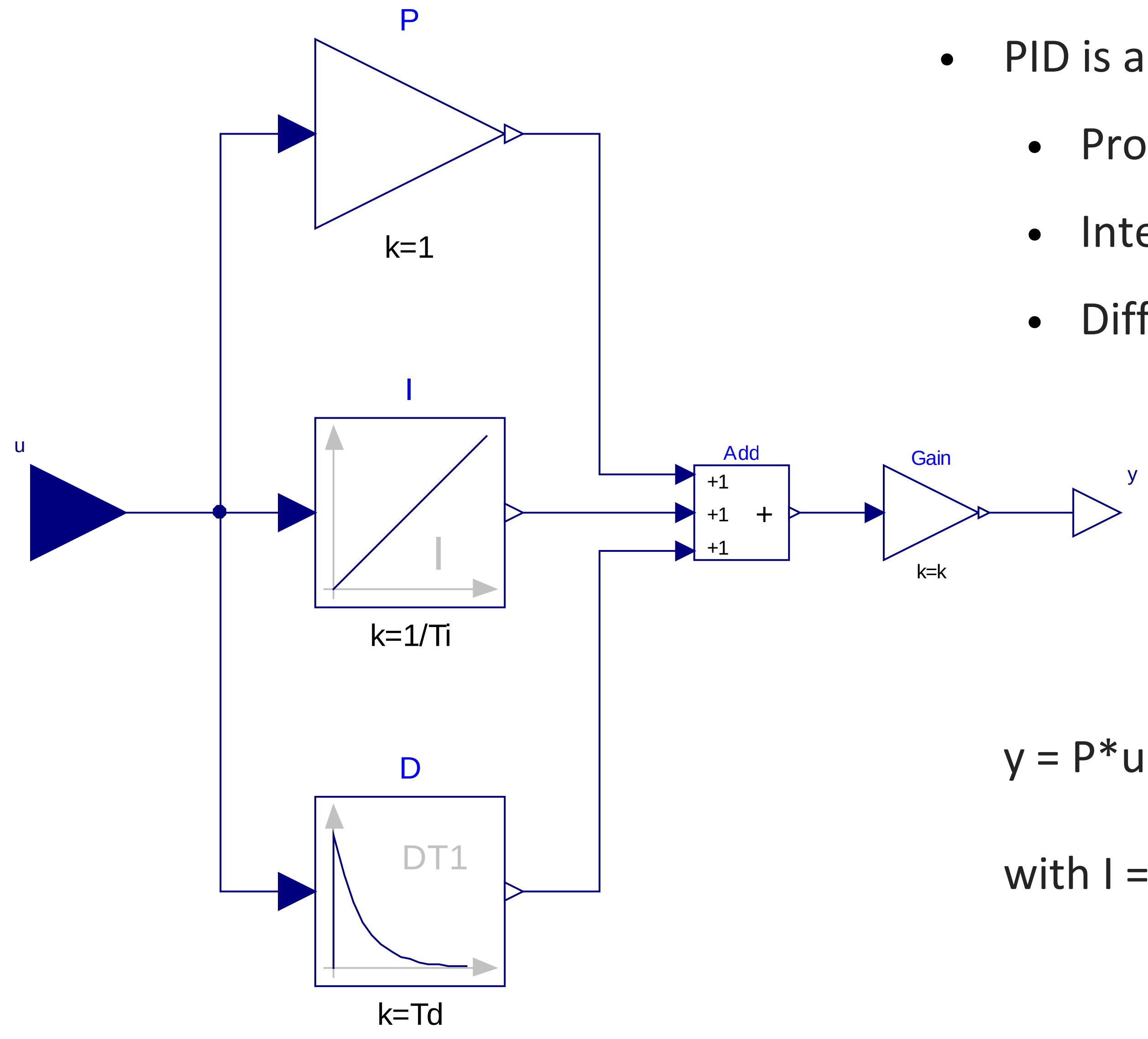


Inverse Pendulum

- A classic controller is the PID controller.
- It is contained in the Modelica Blocks library.
- The input to a PID controller is always the deviation of the measured state to the desired target value.
  (Here the desired target value represents the angle 0°)



Inverse Pendulum



- PID is an abbreviation for
  - Product
  - Integrator
  - Differentiator

y = P*u + I*∫udt + D*du/dt

with I = P/Ti and D=P*Td;

Inverse Pendulum

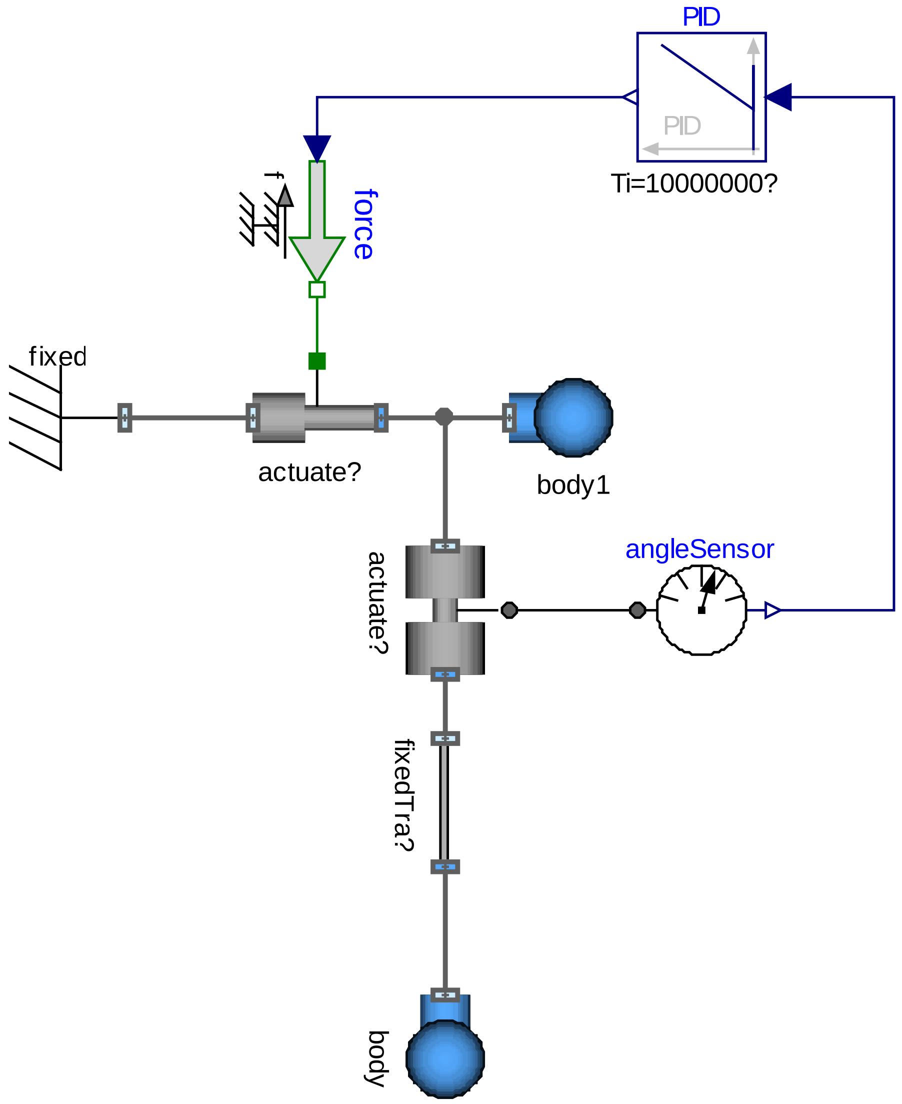
Robotics and Mechatronics Centre

- In order to apply the PID controller we need to find the appropriate parameters.

- All too often, this is done by trial and error. We choose:

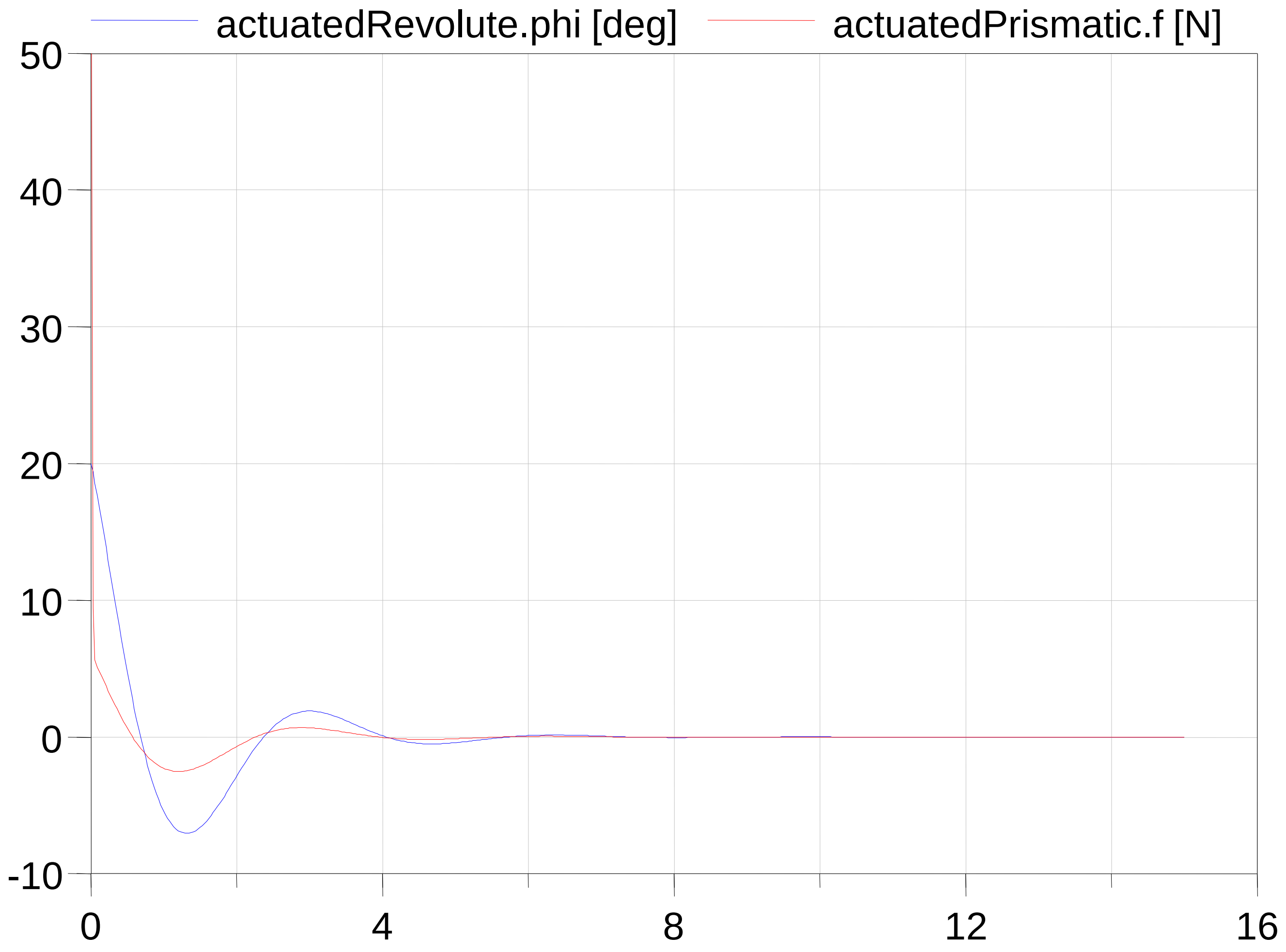
  P = 20;
  Ti = infinity (I=0)
  Td =0.1 (D=2)

- So it is actually a PD controller.



Initialization

- This specific controller leads to a damped oscillation.



Stability Analysis

- How does the PD controller affect the stability of the system?
- First of all, it becomes part of the state-space form:

$ds/dt = v;$

$d\varphi/dt = \omega;$

$dv/dt = (\sin(\varphi)\cdot M_P\cdot R\cdot \omega^2 - \cos(\varphi)\cdot\sin(\varphi)\cdot M_P \cdot G + P\cdot(\varphi-\pi) + D\cdot\omega) / (M_S+M_P \cdot \sin(\varphi)^2)$

$d\omega/dt = (\sin(\varphi)\cdot G - \cos(\varphi)\cdot dv/dt)/R;$

Stability Analysis

Robotics and Mechatronics Centre

- This changes the Jacobi-matrix of our system at $\varphi = 180°$ and $\omega = 0°/s$:

  $d(dv/dt)/d\varphi = -M_P \cdot G/M_S + P/M_S$

  $d(dv/dt)/d\omega = D/M_S$

  $d(d\omega/dt)/d\varphi = (-G - (M_P \cdot G)/M_S - P)/R = -G(1+M_P/M_S)/R - P/R$

  $d(d\omega/dt)/d\omega = -D/(M_S \cdot R)$

- Hence

$$\begin{bmatrix} ds/dt \\ dv/dt \\ d\varphi/dt \\ d\omega/dt \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -M_P \cdot G/M_S + P/M_S & D/M_S \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -G(1+M_P/M_S)/R - P/(M_S R) & -D/(M_S R) \end{bmatrix} \begin{bmatrix} s \\ v \\ \varphi \\ \omega \end{bmatrix}$$

Stability Analysis

- This changes the Jacobi-matrix of our system at $\varphi = 180°$ and $\omega = 0°/s$:

$$d\,(dv/dt)\,/d\varphi = -M_P \cdot G/M_S + P/M_S$$

$$d\,(dv/dt)\,/d\omega = D/M_S$$

$$d\,(d\omega/dt)/d\varphi = (-G - (M_P \cdot G)/\,M_S - P)/R = -G\,(1+ M_P/M_S)/R - P/R$$

$$d\,(d\omega/dt)/d\omega = -D/(M_S \cdot R)$$

- Hence

$$\begin{bmatrix} \cancel{ds/dt} \\ \cancel{dv/dt} \\ d\varphi/dt \\ d\omega/dt \end{bmatrix} = \begin{bmatrix} \cancel{0} & \cancel{1} & \cancel{0} & \cancel{0} \\ \cancel{0} & \cancel{0} & \cancel{-M_P \cdot G/M_S + P/M_S} & \cancel{D/M_S} \\ \cancel{0} & \cancel{0} & 0 & 1 \\ \cancel{0} & \cancel{0} & -G(1+M_P/M_S)/R - P/(M_S R) & -D/(M_S R) \end{bmatrix} \begin{bmatrix} \cancel{s} \\ \cancel{v} \\ \varphi \\ \omega \end{bmatrix}$$

Stability Analysis

Robotics and Mechatronics Centre

- The system:

$$\begin{bmatrix} d\varphi/dt \\ d\omega/dt \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -G(1+M_P/M_S)/R - P/(M_S R) & -D/(M_S R) \end{bmatrix} \begin{bmatrix} \varphi \\ \omega \end{bmatrix}$$

- Is marginally stable when $P > -G\cdot(M_S + M_P)$ with $G := -9.81$
- It then becomes stable for $D>0$
- Based on a symbolical analysis of our model we could derive suitable parameters for P and D.

Model Inversion

- We see that finding a suitable controller can be a difficult and laborious task.
- Beside PID controllers, there are many other controller variants such as State-Space Controller, LQR Controller, etc…
- However, if we have a full model of our system available in Modelica, it should be possible to compute the required actuator forces directly out of the model.

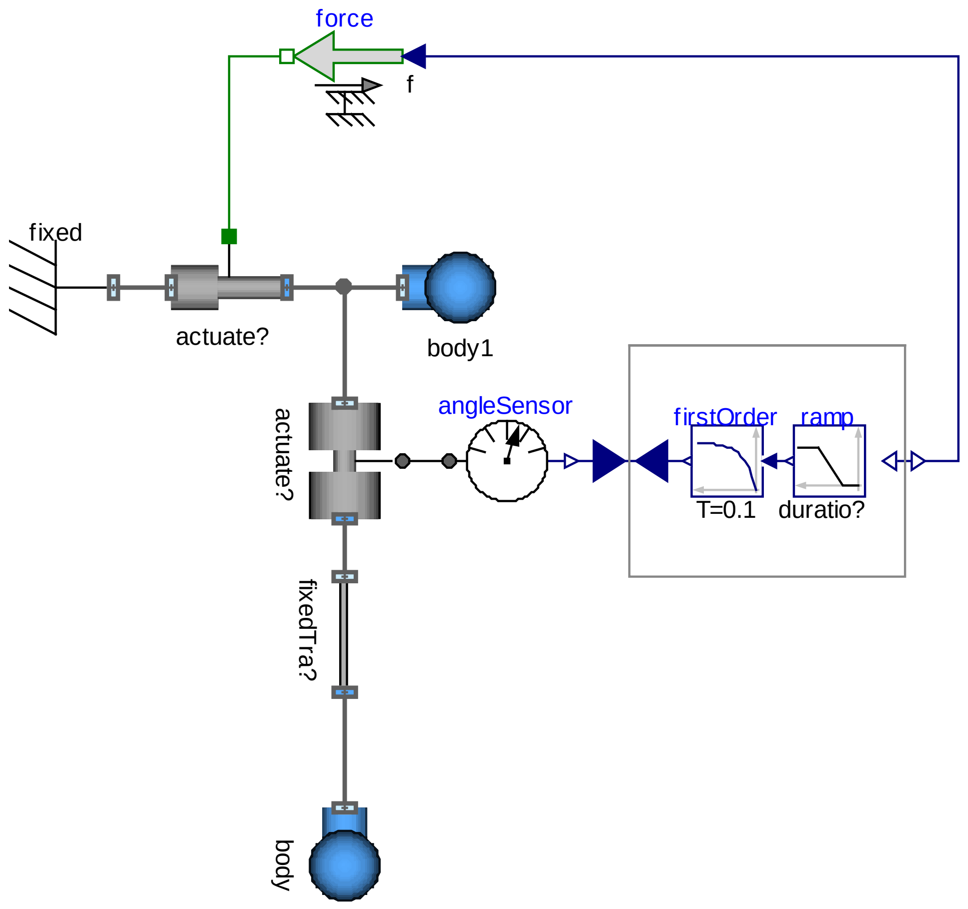
Model Inversion

- Instead of prescribing the forces and computing the resulting trajectory, we go the other way around and prescribe the trajectory computing the required forces.

- This setup of a model is called model inversion.
(because the model is used in the inverted way. However, we shall remember that physical models are not causal. Forces do not cause motion. And motion does not cause forces. Motion and force simply coincide.)

- The following slides show how to setup a model in the inverted way…

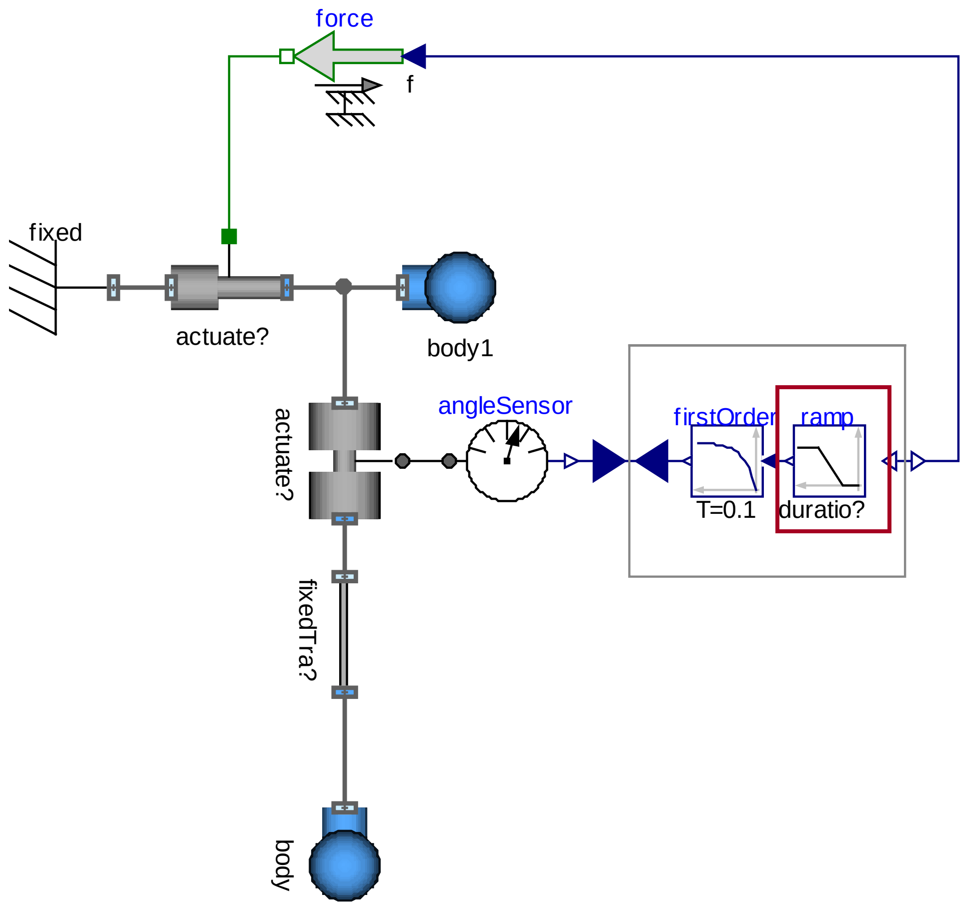
Inverse Pendulum



- Here is the inverted model.
- This model may look very strange at first sight.
- So let us examine this diagram step by step.

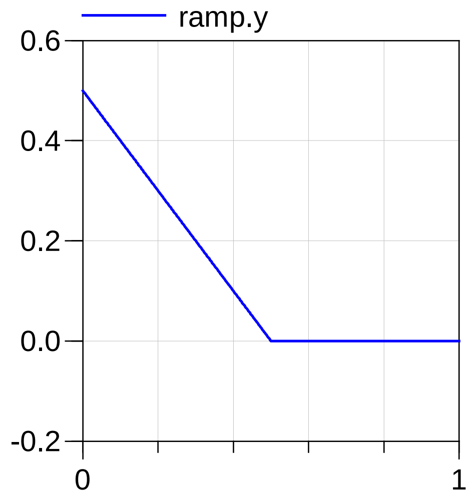
Inverse Pendulum

Robotics and Mechatronics Centre



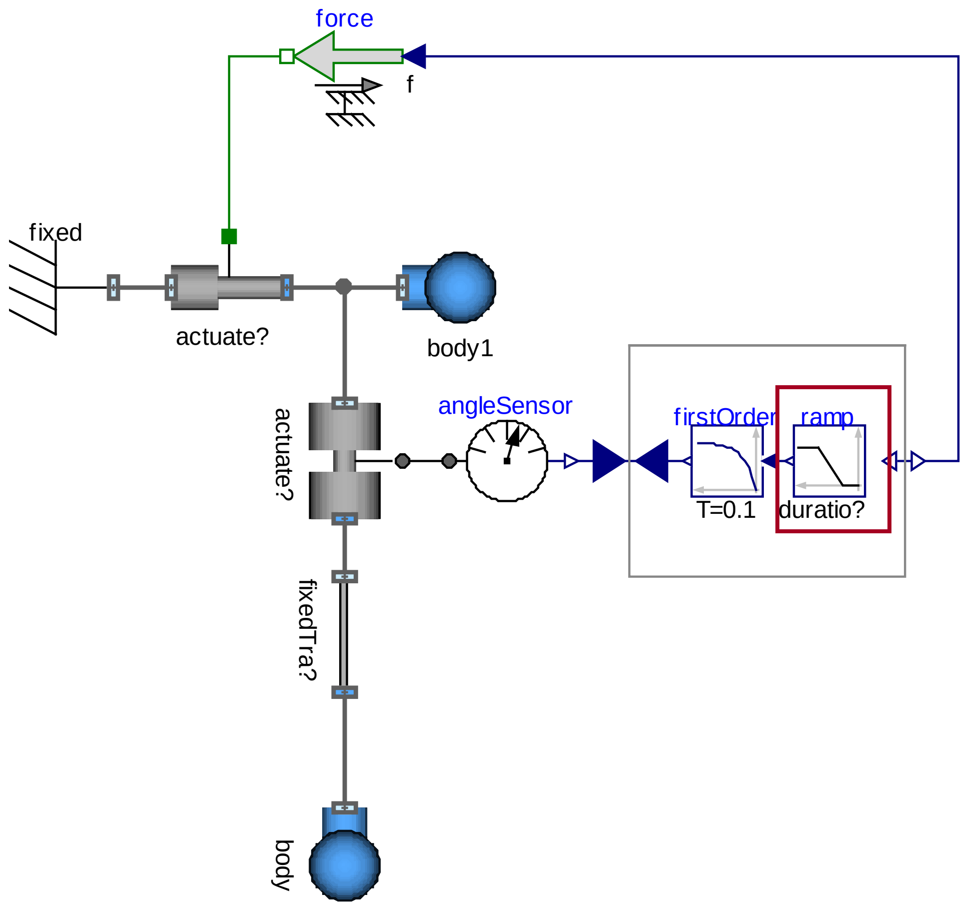
- We use a ramp with a first-order filter in order to prescribe the angle of the pendulum.

- The ramp states that the angle goes from 0.2 rad to 0 rad within 0.5 seconds



Inverse Pendulum

Robotics and Mechatronics Centre

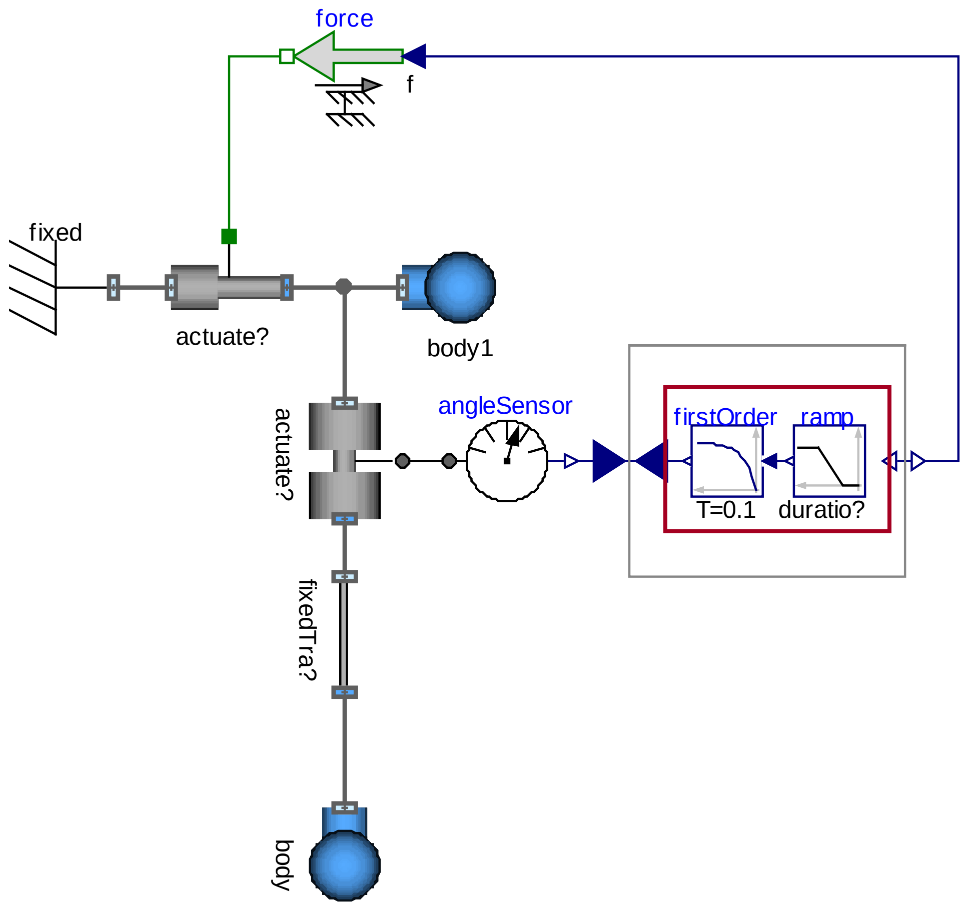


- The ramp alone is a non-valid trajectory since it requires sudden changes in velocity.

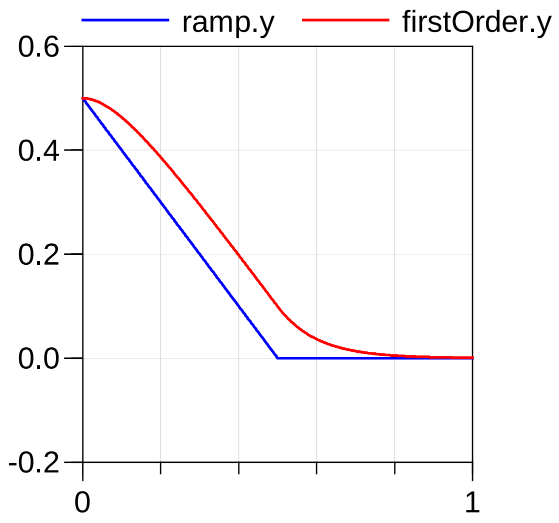
- A direct application of the ramp would result in the following error:

```
  Model error - differentiated
if-then-else was not continuous:

 (if time < ramp.startTime then 0
else (if time <
ramp.startTime+ramp.duration then
ramp.height/ramp.duration else
0.0))

 Value jumped from -1 to 0.
```

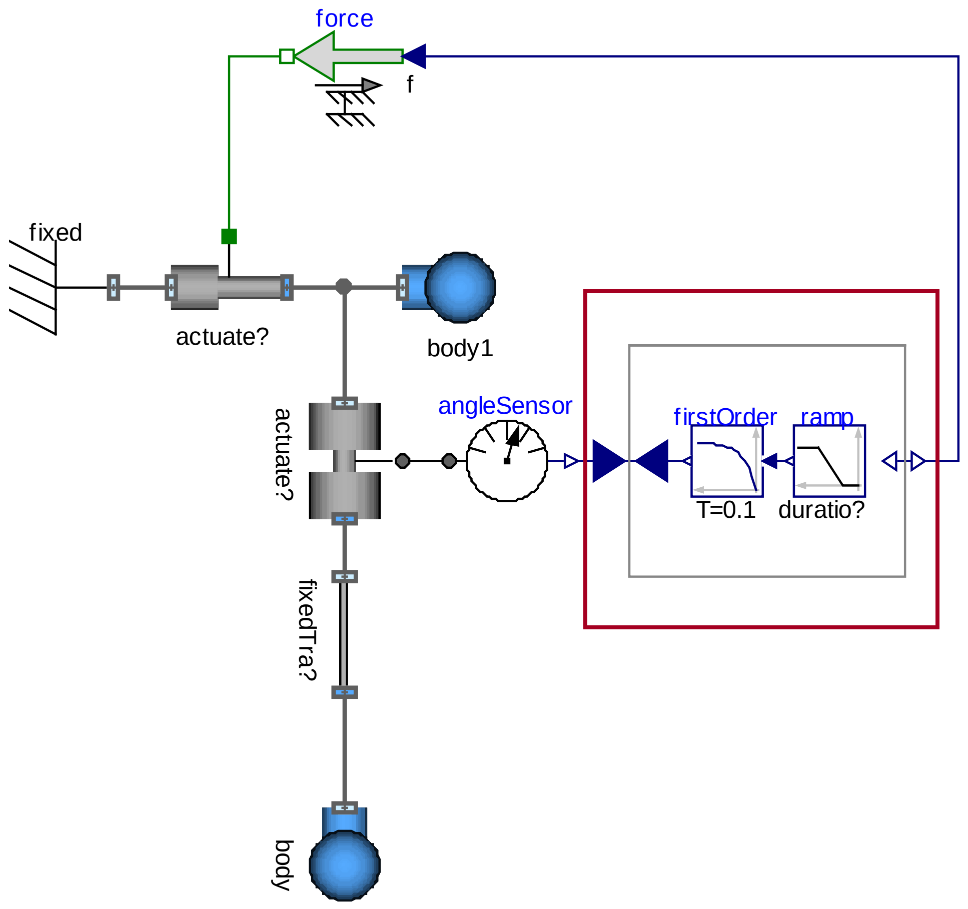
Inverse Pendulum



- Hence we use a first-order filter, to create a differenti-able trajectory.
- The filter works like the one we used to filter our Boolean Keyboard input in lecture 8. It is initialized in steady state (der = 0).



Inverse Pendulum

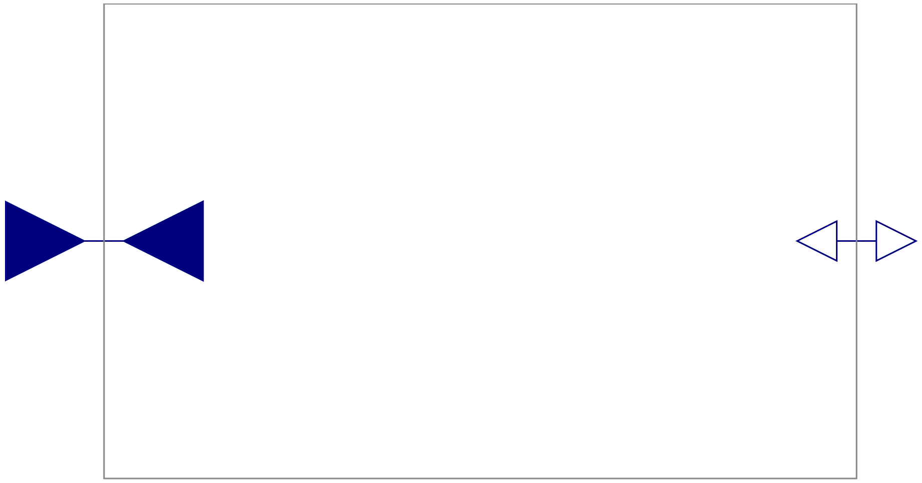


- It is not possible to directly connect the output signal of the firstOrder filter to the output signal of the angle sensor as we intend to do.

- For this reason, we have to use the peculiar model:

  InverseBlockConstraints

Inverse Pendulum

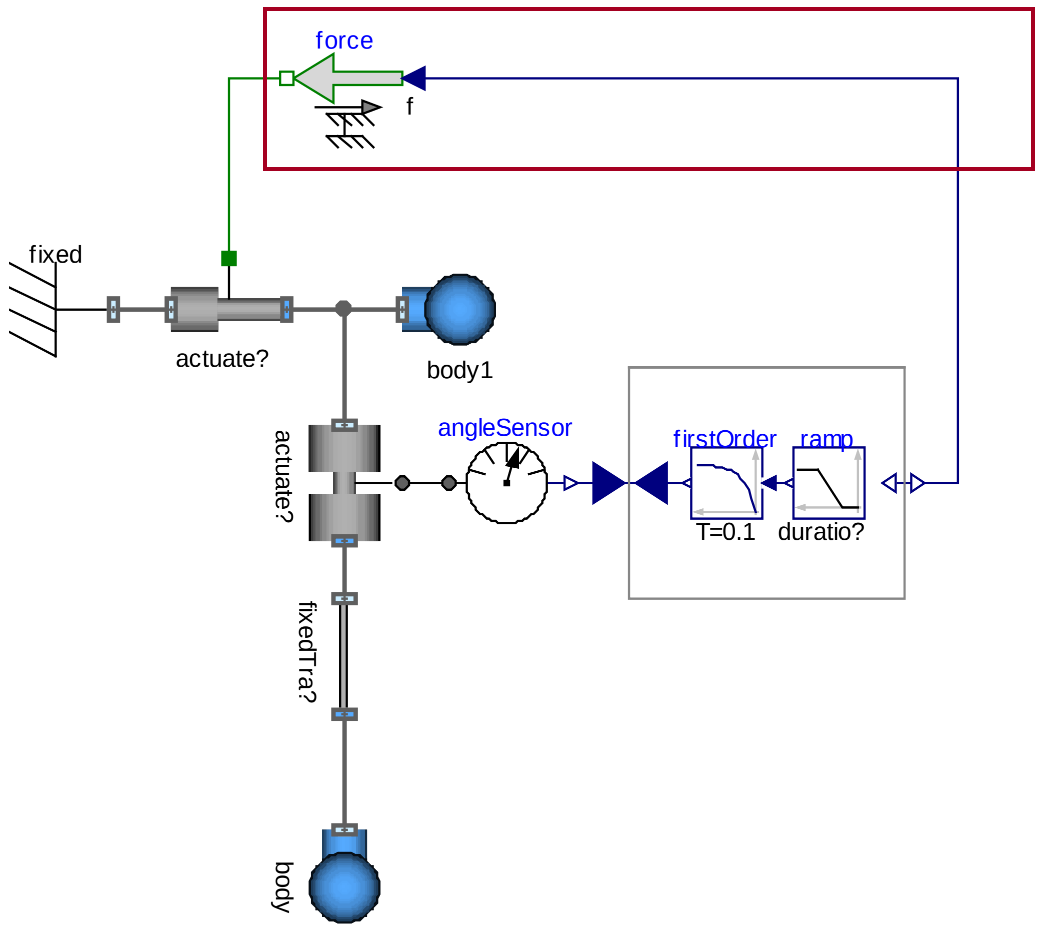


```
block InverseBlockConstraints
  Modelica.Blocks.Interfaces.RealInput u1;
  Modelica.Blocks.Interfaces.RealInput u2;
  Modelica.Blocks.Interfaces.RealOutput y1;
  Modelica.Blocks.Interfaces.RealOutput y2;
equation
  u1 = u2;
  y1 = y2;
  annotation(
    __Dymola_structurallyIncomplete=true
  );
end InverseBlockConstraints
```

- This block model is actually very simple: It connects its two inputs with each other and its two outputs.
- An annotation marks this model as structurally incomplete so that a local check of the model does not yield an error.
- Usually such a model is crap but when being used for model inversion, the whole system will become structurally regular again.

Inverse Pendulum

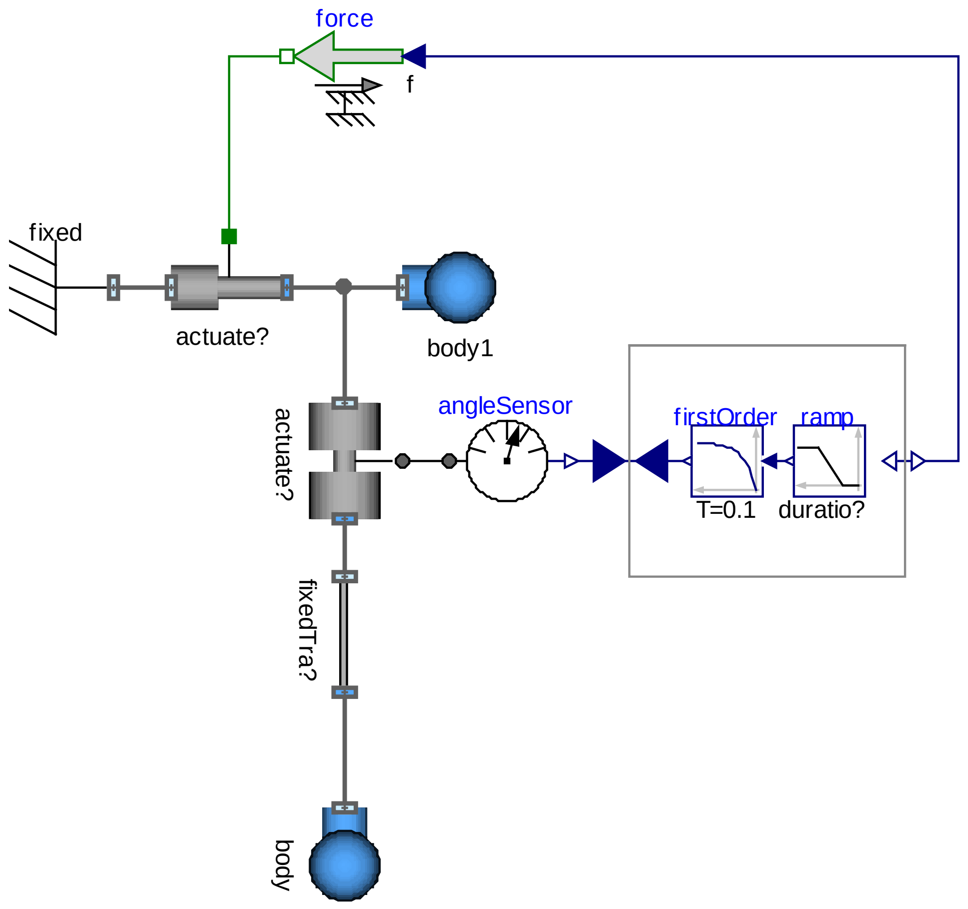
Robotics and Mechatronics Centre



- We connect the input signal for the actuator force to the output signal of the inversion block.
- We see that the sensors and sources are used against the original intention.
- The sensor is used to prescribe the motion. The Source is used to measure the resulting force.
- Input/Output Signals are not causal. They simply establish algebraic equations.

Inverse Pendulum

Robotics and Mechatronics Centre

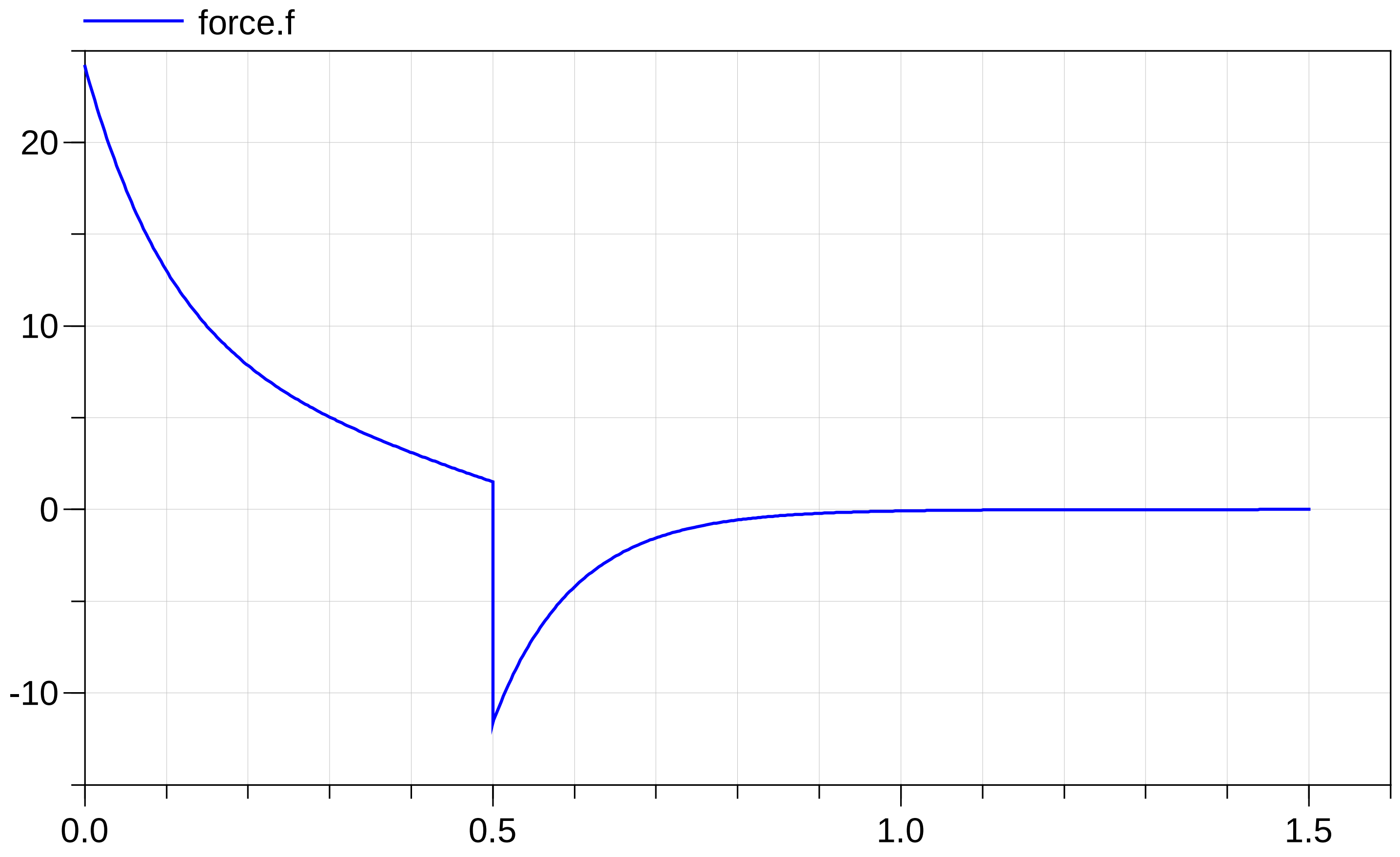


- Let us look at the state-variables of our inverted models.
- Originally the model had four states φ, ω, s, and v.
- In the inverted models, only s and v remain as states.
- We see that a good index-reduction (Pantelides) is required for model inversion.

Initialization

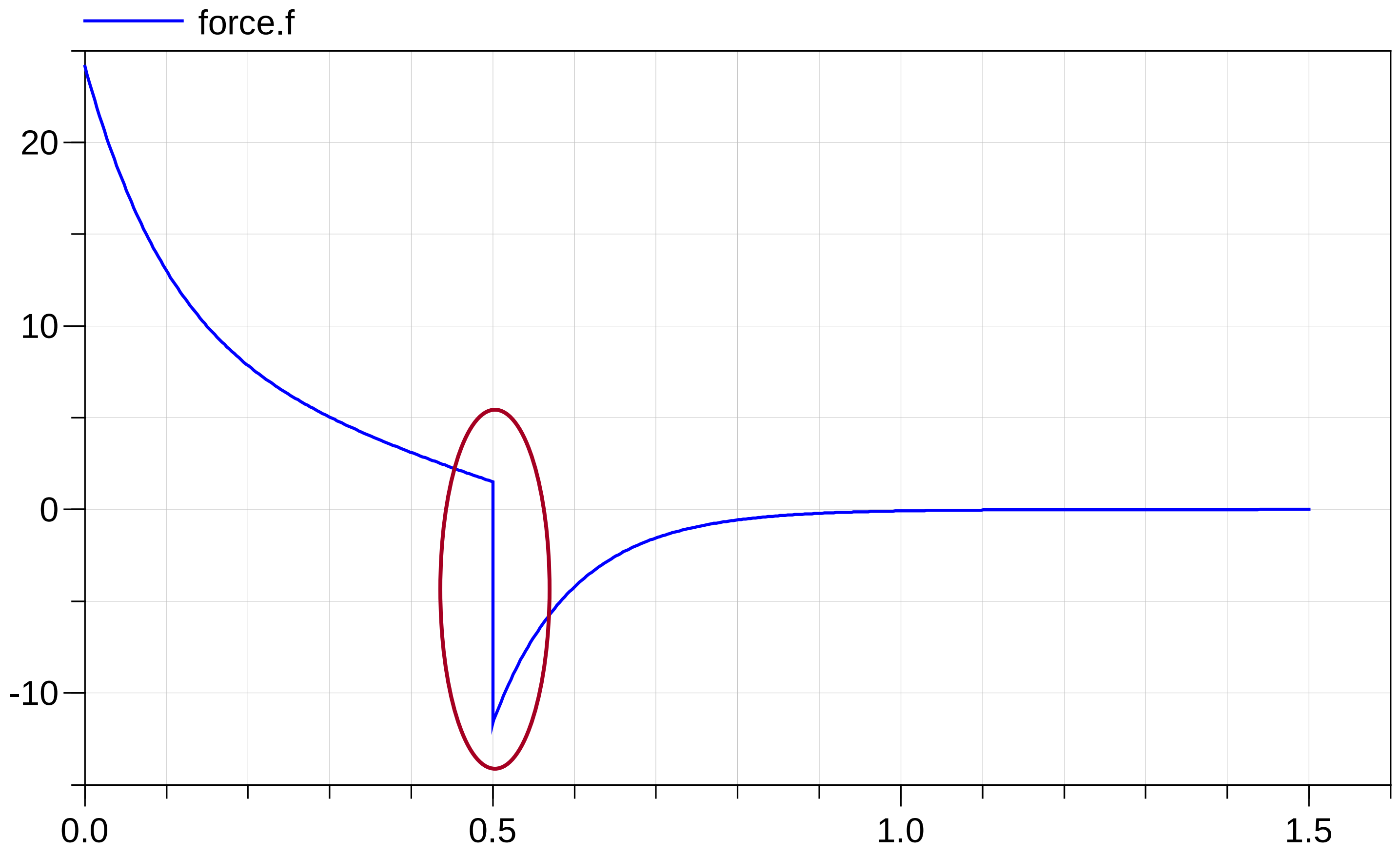
Robotics and Mechatronics Centre

- Simulating this model, computes the required force to follow the imposed trajectory.



Initialization

- The force is a non-continuous curve since the prescribed trajectory is only differentiable once but not twice.



Summary

Robotics and Mechatronics Centre

- We have examined the control task of an inverted pendulum
- Modelica models cannot only be used to simulate a device but also to design a controller for a device.
- To this end, model inversion may be applied.
- Model-inversion requires no extra features. It is performed by the index-reduction mechanism of Dymola (Pantelides is heavily used).

Questions ?