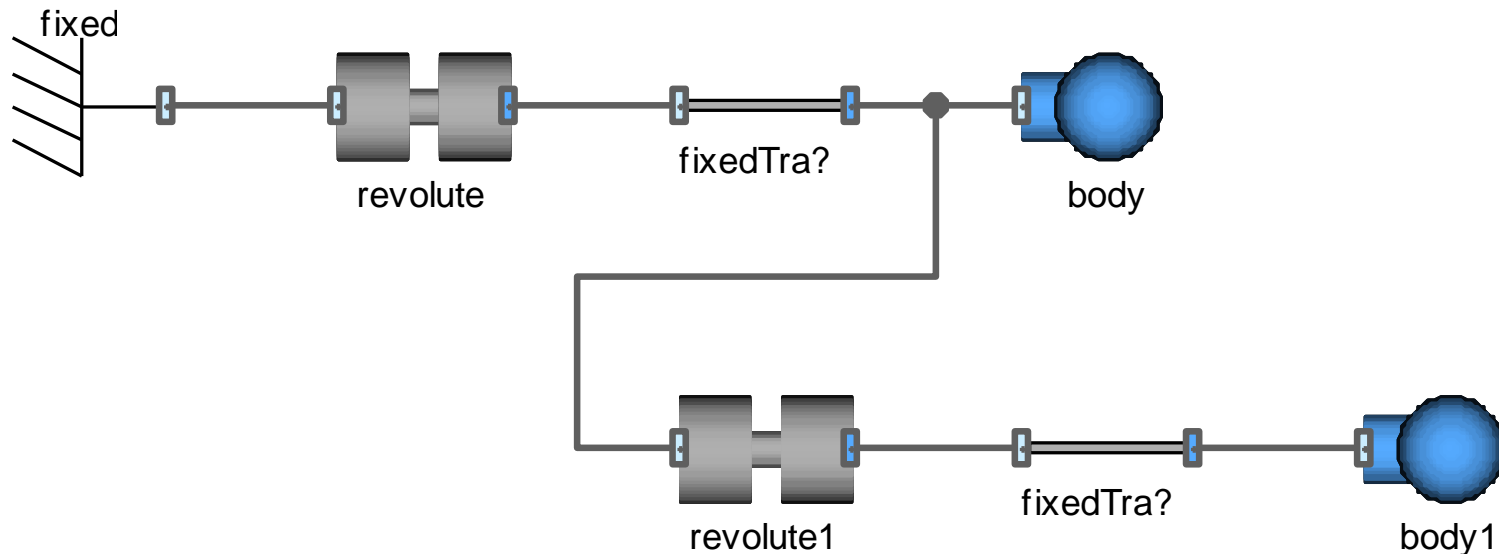


Virtual Physics

Equation-Based Modeling

TUM, December 06, 2022

2D-Mechanical Systems: Kinematic Loops



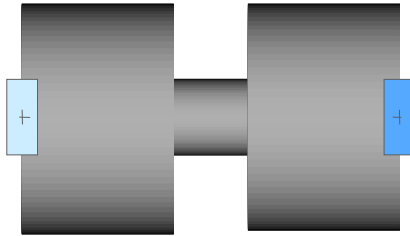
Dr. Dirk Zimmer

German Aerospace Center (DLR), Robotics and Mechatronics Centre

- So far, we have only cared about the equations that describe the dynamic behavior of the system.
- But we need to define a set of initial equations too.
- Whereas the dynamic equations can be generically formulated in a way that the components can be almost arbitrarily connected, this is unfortunately not the case for the initial equations.
- In general, they need to be manually set up for each specific system.

- However, what we can do is to put the modeler into a position so that he can set up the initial state of the system in a convenient way.
- To this end, we create parameterized initial equations for some of our components.
- Usually, the joints are a good place to set the initial equations of a system.

Let us add initial equations to the revolute joint:



- First we add parameters for the initial values.
- Then we can add the correspondent initial equations.

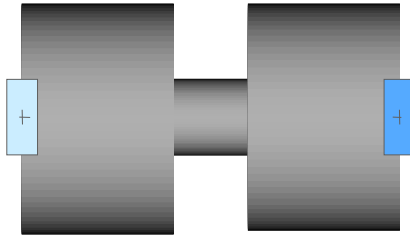
```
model Revolute
  Interfaces.Frame_a frame_a;
  Interfaces.Frame_a frame_b;
  SI.Angle phi
  SI.AngularVelocity w;
  SI.AngularAcceleration z;

  parameter SI.Angle phi_start = 0;
  parameter SI.AngularVelocity w_start=0;

  initial equation
    phi = phi_start;
    w = w_start;

  equation
    frame_a.phi + phi = frame_b.phi;
    w = der(phi);
    z = der(w);
    frame_a.x = frame_b.x;
    frame_a.y = frame_b.y;
    [ ... ]
end Revolute
```

Let us add initial equations to the revolute joint:



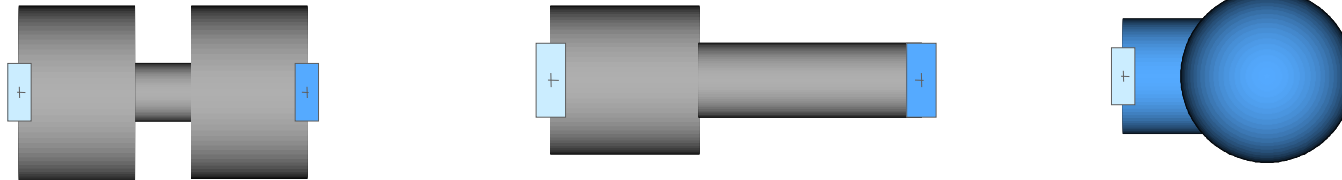
- But it is not clear, if the modeler really wants to state such equations.
- Thus, we put them in conditional form.

```
model Revolute
  Interfaces.Frame_a frame_a;
  Interfaces.Frame_a frame_b;
  SI.Angle phi
  SI.AngularVelocity w;
  SI.AngularAcceleration z;
  parameter SI.Angle phi_start = 0;
  parameter SI.AngularVelocity w_start=0;
  parameter Boolean initialize = false;

  initial equation
    if initialize then
      phi = phi_start;
      w = w_start;
    end if;

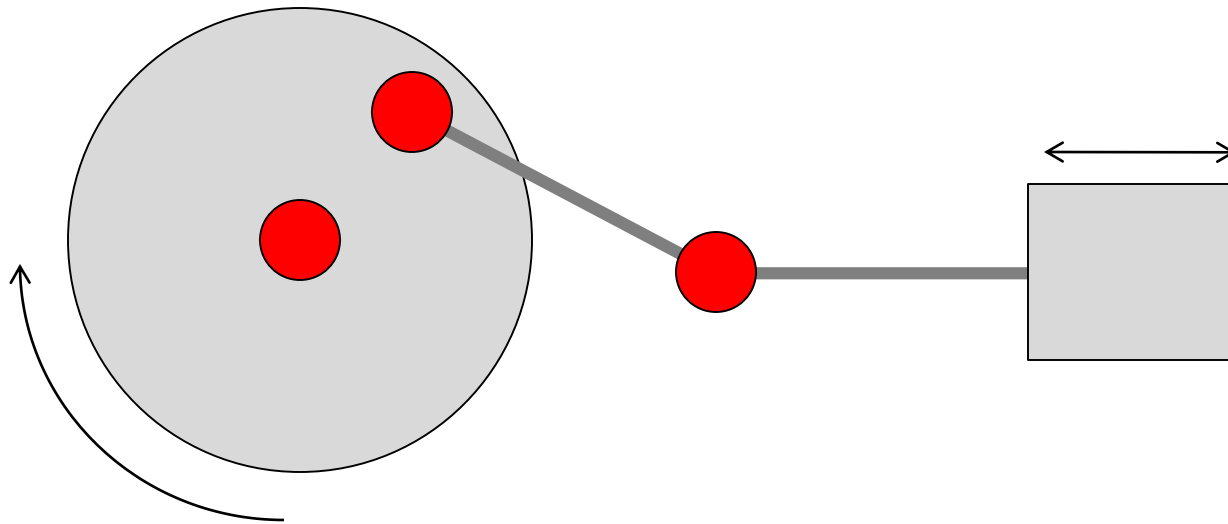
  equation
    frame_a.phi + phi = frame_b.phi;
    w = der(phi);
    z = der(w);
    frame_a.x = frame_b.x;
    frame_a.y = frame_b.y;
    [ ... ]
end Revolute
```

- In this way, we create conditional initial equation sections for a number of components.



- For each of these component we may now use the parameter menu to set the initial values.
- This way of providing a functionality for initialization is still rudimentary. Look at the MultBody library to see a more elaborate version of it.

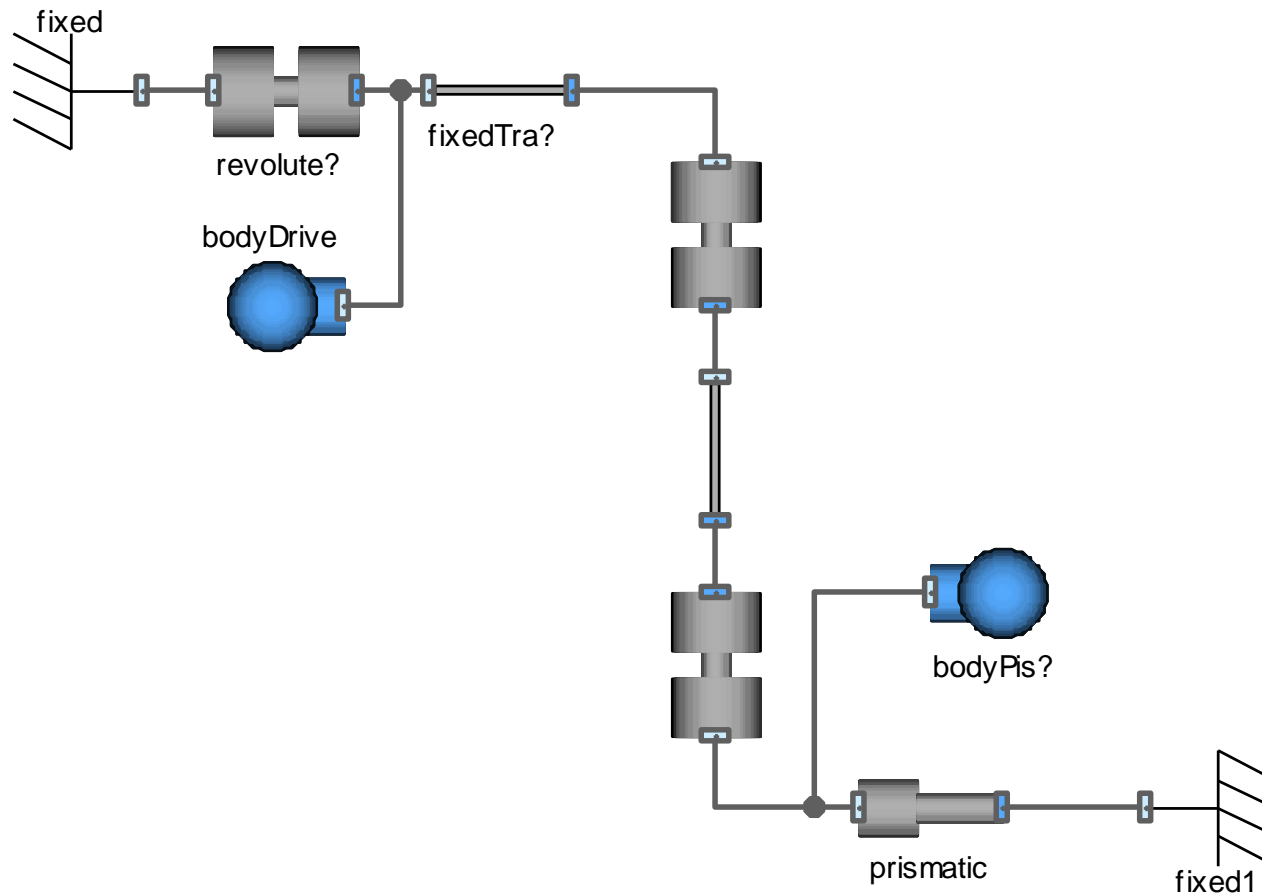
- Let us model a piston engine:



- Although the model has 4 joint elements, it has only 1 degree of freedom. So it is enough to initialize one of the four joints.

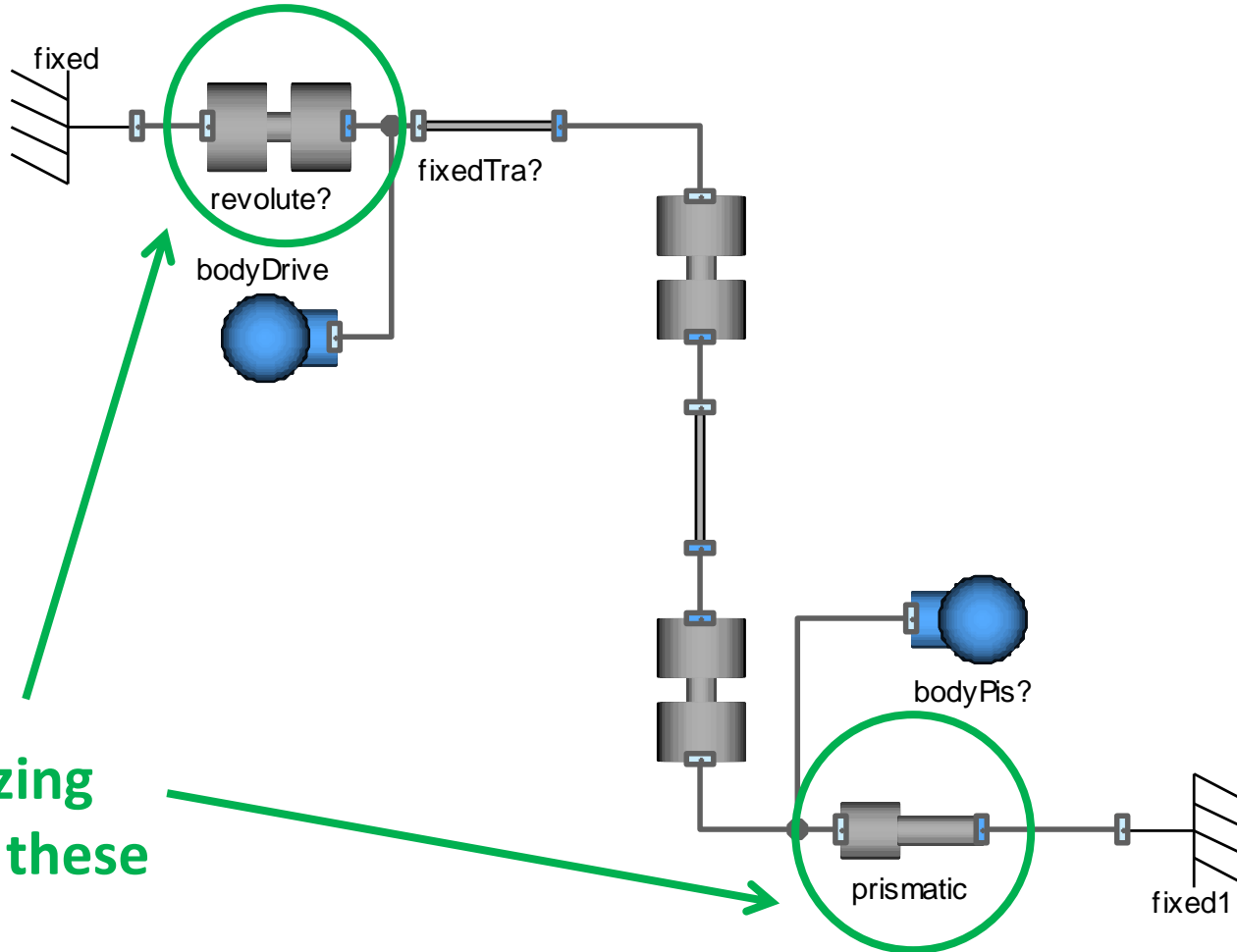
Initializing a Piston Engine

- Model Diagram:



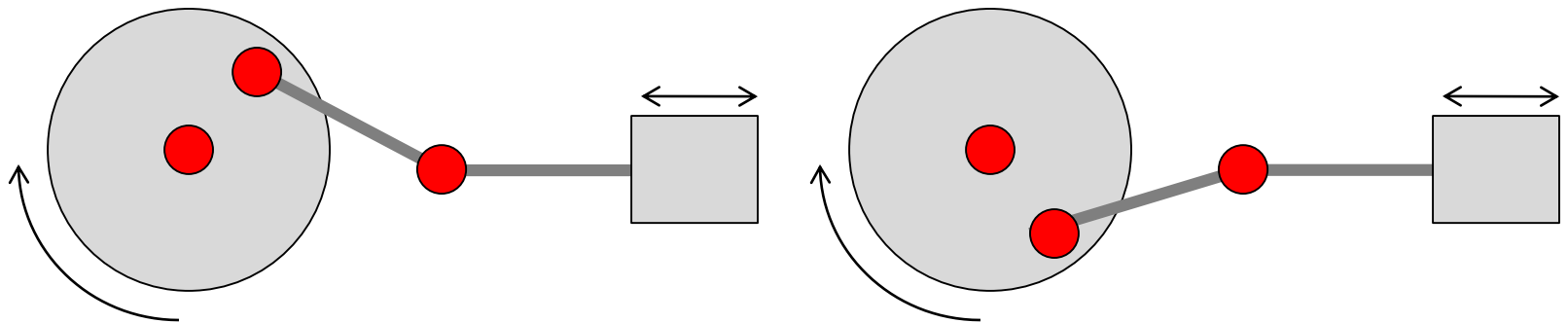
Initializing a Piston Engine

- Model Diagram:



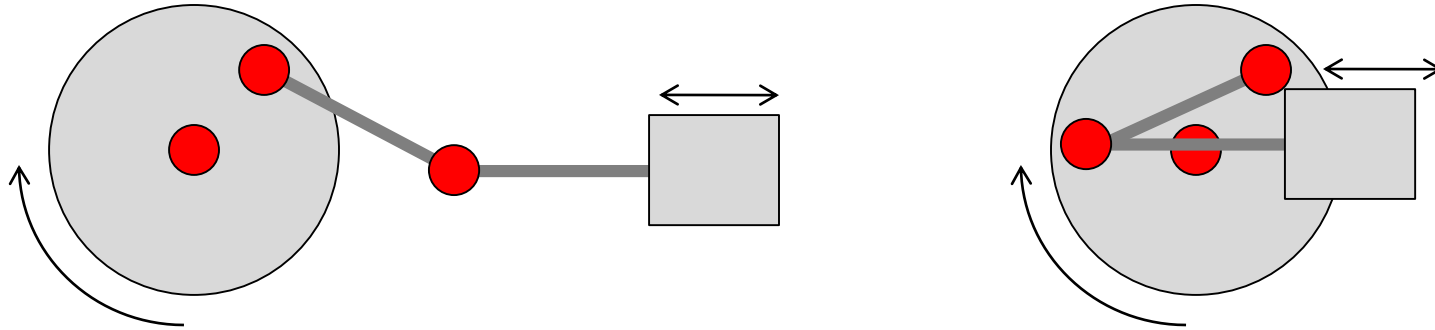
Initializing
one of these

- Let us model a piston engine:



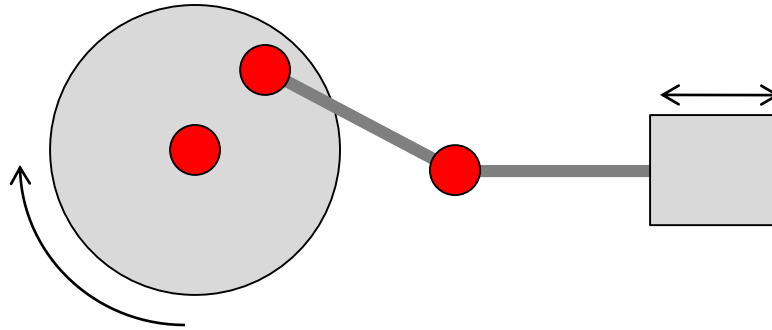
- If we initialize the position of the piston, two possible solutions exist.
- We have to solve a non-linear system of equations.

- Let us model a piston engine:



- The same holds for the initialization via the disc rotation angle.

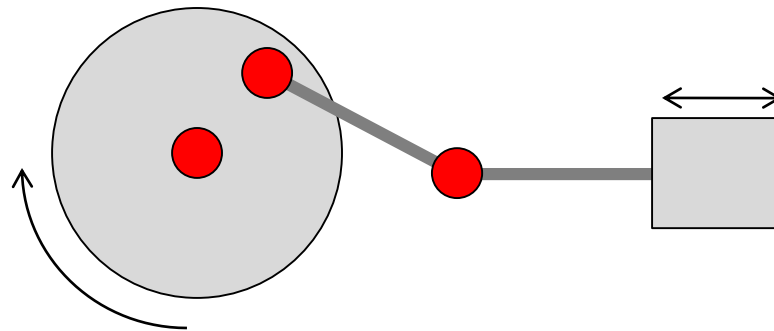
- Let us model a piston engine:



- However, if we attempt to initialize both in order to clarify the solution, we get an overdetermined system of equations.

- Dymola will solve the non-linear equation system by an iterative solver.
- Which solution it will find (if any) depends on the start values of the iterative solver.
- It is possible to suggest start values without enforcing an initialization constraint by using the attributes of Real variables.
 - `Real x(start=10, fixed = false)`
means that 10 is a suggested start value for an iterative solver.
 - `Real x(start=10, fixed = true)`
means that 10 is the initial value.

- Let us model a piston engine:

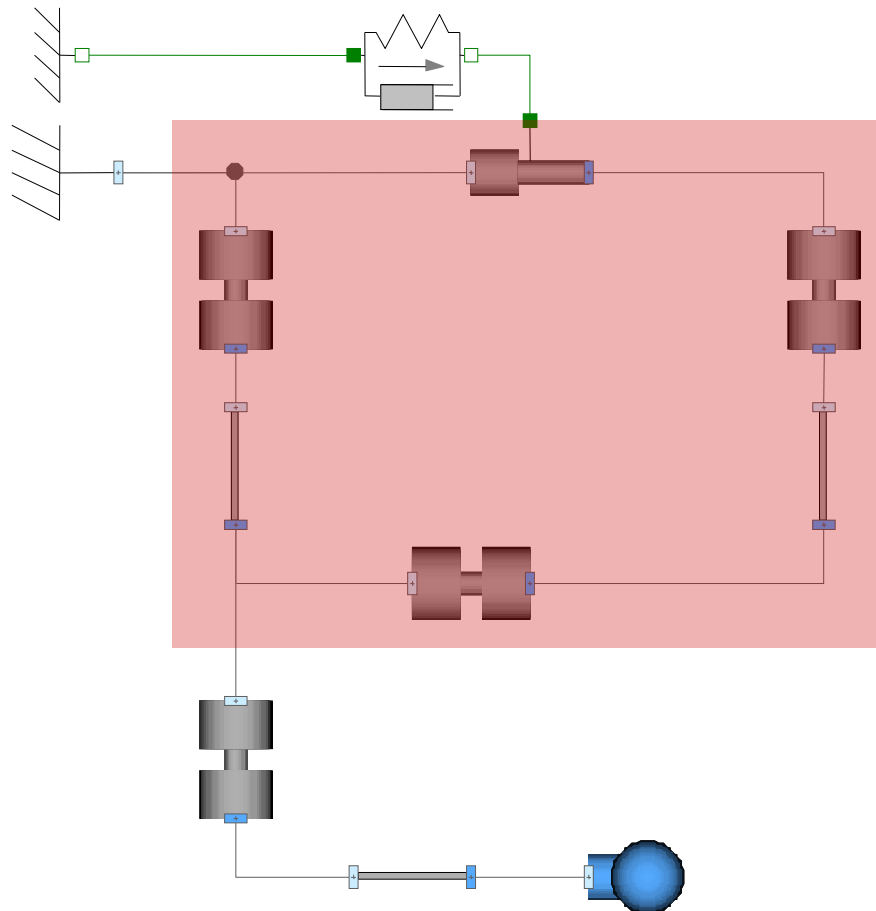


- So I could try to initialize one joint with `fixed=true` and the other joint with `fixed = false`;
- This will do the job. Nevertheless be aware that there is no guarantee that we will get the right solution.

- The model of the piston engine represented an example that contained fewer degrees of freedom than the number of joint elements would suggest.
- Such systems contain a kinematic loop.
- Here is an example where the loop is more evident.

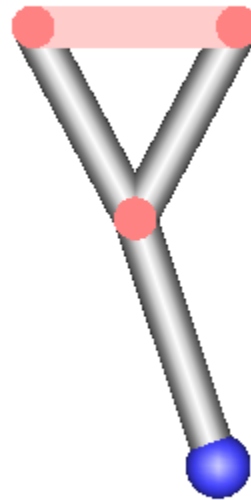
Kinematic Loops: Example

- Here is an example where the actual loop is more evident.



Kinematic Loops: Example

- Here is an example where the actual loop is more evident.



- How do we determine the degrees of freedom?
- Each joint adds one degree of freedom. There are 5 joints, so there are 5 degrees of freedom.
- The closure of a kinematic loop, imposes 3 holonomic constraints.
 $x_1 = x_2;$
 $y_1 = y_2;$
 $\varphi_1 = \varphi_2;$
- Hence, each loop decreases the degrees of freedom by 3 (in planar mechanical systems)
- In our example, $5 - 3 = 2$ degrees of freedom remain.

- When there remain only 2 degrees of freedom, by which state variables are they represented?
- The attached pendulum has its usual states. The angle φ and the angular velocity ω of revolute3
- But which states represent the state of the loop? Here is what Dymola tells you in the translation log of the model:

There are 2 sets of
dynamic state selection.

From set 1 there is 1 state to
be selected from:

```
revolute.phi  
revolute2.phi  
springDamper.s_rel
```

From set 2 there is 1 state to
be selected from:

```
body.w  
revolute2.w
```

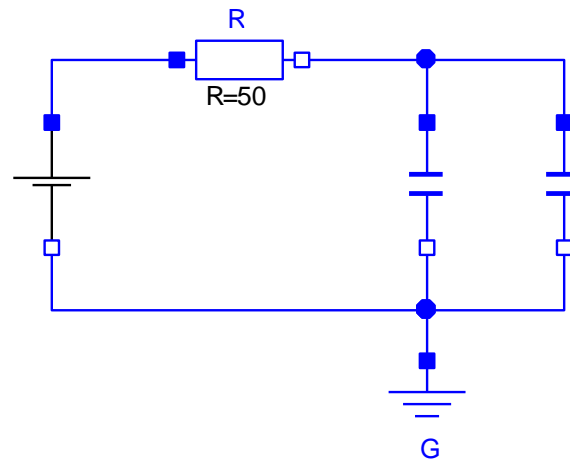
- What is dynamic state-selection?
- After all, what does it mean to *select* states?
- All joints formulate differential equations of their motion, but only a few of these differential equations seem to end up in the explicit state-space form.

$$F(\mathbf{x}_p, d\mathbf{x}_p/dt, \mathbf{u}, t) = 0 \quad \rightarrow \quad d\mathbf{x}/dt = f(\mathbf{x}, \mathbf{u}, t)$$

\rightarrow \mathbf{x} is only a subset of \mathbf{x}_p

- There seems to be an important subject in the translation of models that we have missed so far.

- So far we have assumed, that every variable that occurred as time-derivative, represents a state and is assumed to be known:
- Example in an electric Capacitor:
 $i = C \cdot \text{der}(u) \rightarrow u$ represents a state and is known.
- However, this holds not always true. Let us take a look at a simple counter example:



- Let us model this circuit by the following set of equations:

$$u_R = R \cdot i$$

$$i_{C1} = C1 \cdot du_{C1}/dt$$

$$i_{C2} = C2 \cdot du_{C2}/dt$$

$$v_G = 0;$$

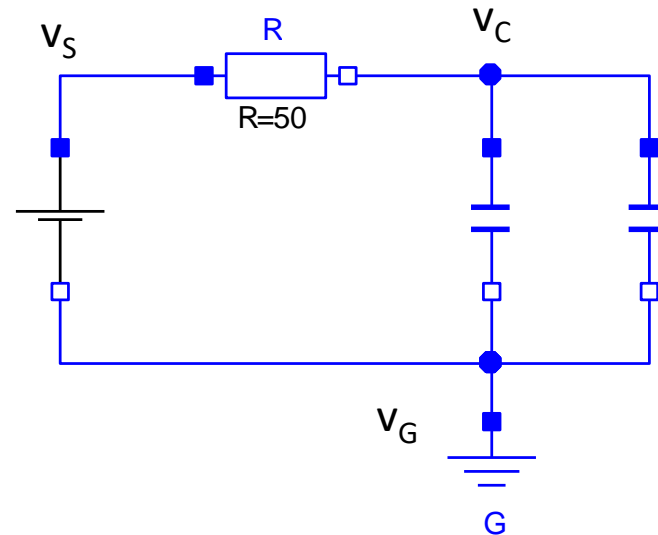
$$v_S = 10;$$

$$v_C = v_G + u_{C1}$$

$$v_C = v_G + u_{C2}$$

$$v_C = v_S - u_R$$

$$i_{C1} + i_{C2} = i$$



- Let us model this circuit by the following set of equations:

$$u_R = R \cdot i$$

$$i_{C1} = C1 \cdot du_{C1}/dt$$

$$i_{C2} = C2 \cdot du_{C2}/dt$$

$$v_G = 0;$$

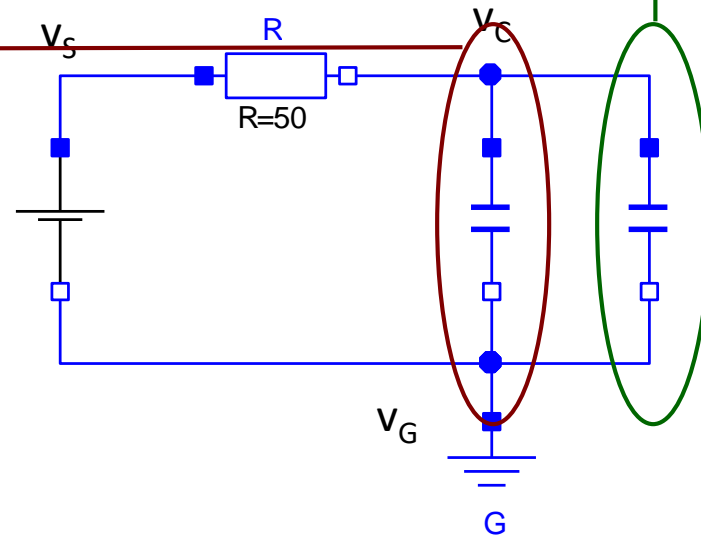
$$v_S = 10;$$

$$v_C = v_G + u_{C1}$$

$$v_C = v_G + u_{C2}$$

$$v_C = v_S - u_R$$

$$i_{C1} + i_{C2} = i$$



$$u_R = R \cdot i$$

$$i_{C1} = C1 \cdot du_{C1}/dt$$

$$i_{C2} = C2 \cdot du_{C2}/dt$$

$$v_G = 0;$$

$$v_S = 10;$$

$$v_C = v_G + u_{C1}$$

$$v_C = v_G + u_{C2}$$

$$v_C = v_S - u_R$$

$$i_{C1} + i_{C2} = i$$

- As usual, we assume u_{C1} and u_{C2} to be known.
- Let us start with forward causalization.

$$v_G := 0;$$

$$v_S := 10;$$

$$v_C := v_G + u_{C1}$$

$$\text{Residual} = v_G + u_{C2} - v_C$$

$$u_R = R * i$$

$$i_{C1} = C1 * du_{C1} / dt$$

$$i_{C2} = C2 * du_{C2} / dt$$

$$v_C = v_S - u_R$$

$$i_{C1} + i_{C2} = i$$

- As usual, we assume u_{C1} and u_{C2} to be known.
- Let us start with forward causalization.
- A residual equation is generated, but there is no iteration variable. The system seems to be overdetermined. We encounter a *structural singularity*.

$$v_G := 0;$$

$$v_S := 10;$$

$$v_C := v_G + u_{C1}$$

$$\text{Residual} = v_G + u_{C2} - v_C$$

$$u_R = R * i$$

$$i_{C1} = C1 * du_{C1} / dt$$

$$i_{C2} = C2 * du_{C2} / dt$$

$$v_C = v_S - u_R$$

$$i_{C1} + i_{C2} = i$$

- In order to remove this structural singularity, we have to apply the Dummy Derivative Method (based on the Pantelides Algorithm):

$$v_G := 0;$$

$$v_S := 10;$$

$$v_C := v_G + u_{C1}$$

$$\text{Residual} = v_G + u_{C2} - v_C$$

$$u_R = R \cdot i$$

$$i_{C1} = C1 \cdot du_{C1}/dt$$

$$i_{C2} = C2 \cdot du_{C2}/dt$$

$$v_C = v_S - u_R$$

$$i_{C1} + i_{C2} = i$$

- In order to remove this structural singularity, we have to apply the Dummy Derivative Method (based on the Pantelides Algorithm):
- To this end, we assume one of the affected states to be unknown (we gain one unknown)

$$v_G := 0;$$

$$v_S := 10;$$

$$v_C := v_G + u_{C1}$$

$$0 = v_G + u_{C2} - v_C$$

$$d0/dt = d(v_G + u_{C2} - v_C)/dt$$

$$u_R = R * i$$

$$i_{C1} = C1 * du_{C1}/dt$$

$$i_{C2} = C2 * du_{C2}/dt$$

$$v_C = v_S - u_R$$

$$i_{C1} + i_{C2} = i$$

- In order to remove this structural singularity, we have to apply the Dummy Derivative Method (based on the Pantelides Algorithm):
- To this end, we assume one of the affected states (here: u_{C2}) to be unknown (we gain one unknown)
- And add as additional equation the time derivative of the constraint.

$$v_G := 0;$$

$$v_S := 10;$$

$$v_C := v_G + u_{C1}$$

$$0 = v_G + u_{C2} - v_C$$

$$d0/dt = d(v_G + u_{C2} - v_C)/dt$$

$$u_R = R \cdot i$$

$$i_{C1} = C1 \cdot du_{C1}/dt$$

$$i_{C2} = C2 \cdot du_{C2}/dt$$

$$v_C = v_S - u_R$$

$$i_{C1} + i_{C2} = i$$

- The differentiated equation

$$d0/dt = d(v_G + u_{C2} - v_C)/dt$$

can be transformed to...

$$0 = dv_G/dt + du_{C2}/dt - dv_C/dt$$

- The derivatives dv_G/dt and dv_C/dt are yet unknown. We have to differentiate further equations.

$$v_G := 0;$$

$$dv_G/dt := 0;$$

$$v_S := 10;$$

$$v_C := v_G + u_{C1}$$

$$dv_C/dt = dv_G/dt + du_{C1}/dt$$

$$0 = v_G + u_{C2} - v_C$$

$$0 = dv_G/dt + du_{C2}/dt - dv_C/dt$$

$$u_R = R \cdot i$$

$$i_{C1} = C1 \cdot du_{C1}/dt$$

$$i_{C2} = C2 \cdot du_{C2}/dt$$

$$v_C = v_S - u_R$$

$$i_{C1} + i_{C2} = i$$

- Adding an equation in differentiated form may require further derivation of further variables and equations
- Here, we had to add two further variables (dv_G/dt , dv_C/dt) and two equations.
- Now we can continue to causalize the system...

$$v_G := 0;$$

$$dv_G/dt := 0;$$

$$v_S := 10;$$

$$v_C := v_G + u_{C1}$$

$$u_{C2} := v_C - v_G$$

$$u_R := v_S - v_C$$

$$i := u_R/R$$

$$0 = dv_G/dt + du_{C2}/dt - dv_C/dt$$

$$dv_C/dt = dv_G/dt + du_{C1}/dt$$

$$i_{C1} = C1 * du_{C1}/dt$$

$$i_{C2} = C2 * du_{C2}/dt$$

$$i_{C1} + i_{C2} = i$$

- There remain 5 equations non-causalized. Evidently, there is an algebraic loop.
- This loop represents the division of current among the two capacitors.
- In order to break the loop, we select i_{C1} as tearing variable and causalize.

$$v_G := 0;$$

$$dv_G/dt := 0;$$

$$v_S := 10;$$

$$v_C := v_G + u_{C1}$$

$$u_{C2} := v_C - v_G$$

$$u_R := v_S - v_C$$

$$i := u_R/R$$

$$i_{C1} := \text{iteration variable}$$

$$du_{C1}/dt := i_{C1}/C1$$

$$dv_C/dt := dv_G/dt + du_{C1}/dt$$

$$i_{C2} := i - i_{C1}$$

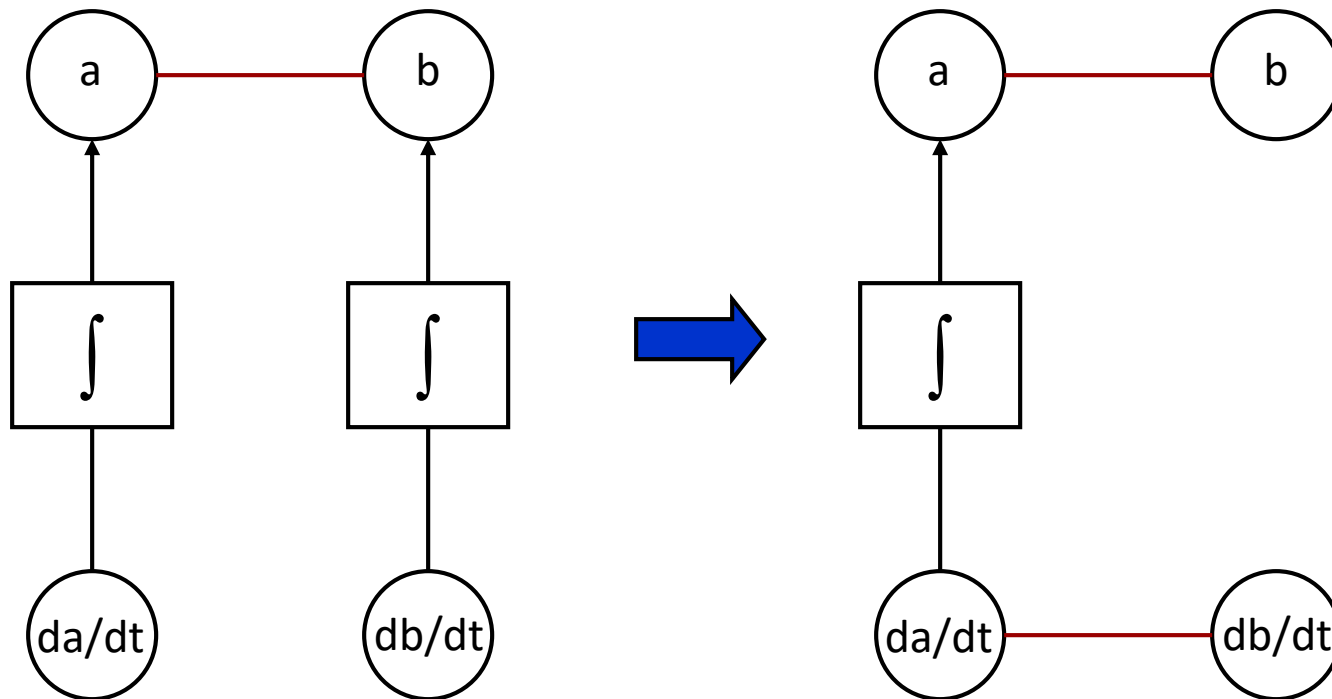
$$du_{C2}/dt := i_{C2}/C2$$

$$0 = dv_G/dt + du_{C2}/dt - dv_C/dt$$

- There remain 5 equations non-causalized. Evidently, there is an algebraic loop.
- This loop represents the division of current among the two capacitors.
- In order to break the loop, we select i_{C1} as tearing variable and causalize.
- Finally, we get one residual equation.
- Structural singularities often generate algebraic loops.

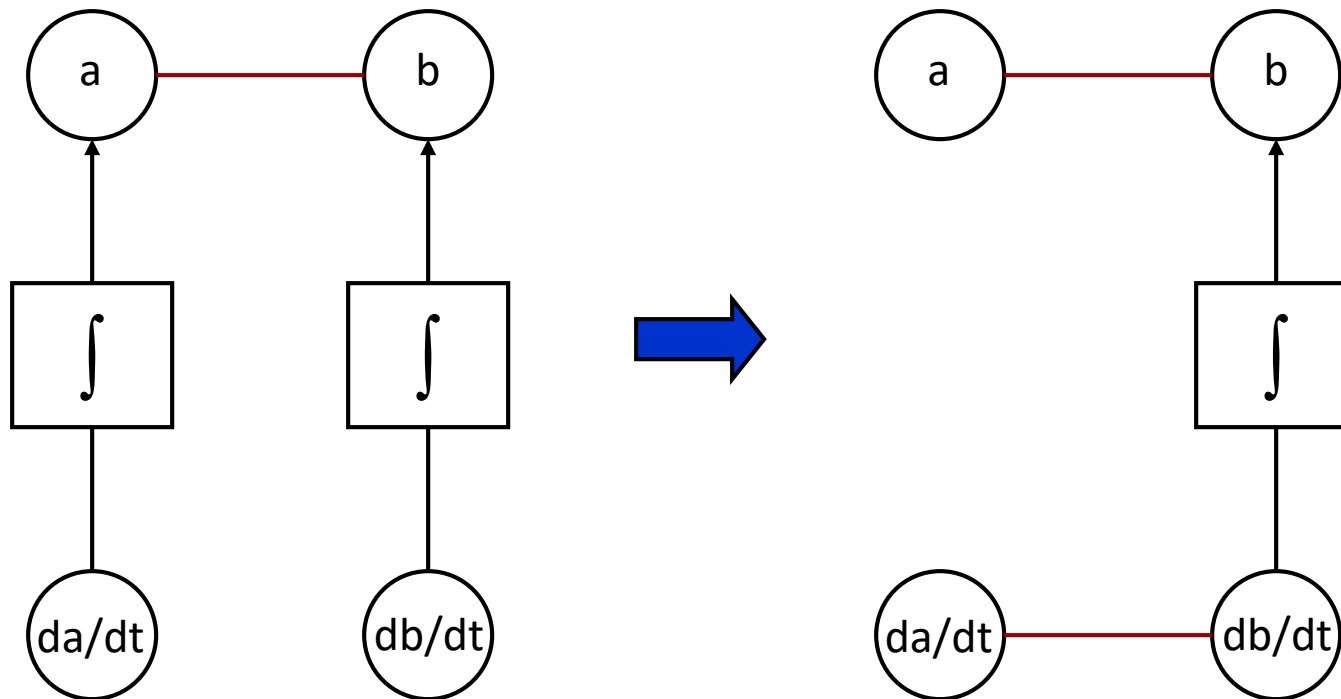
- Initially, all potential state-variables are assumed to be known.
- For each constraint equation between potential state-variables, we have to de-select one state (assuming it to be unknown): we gain one unknown.
- Then, we differentiate the constraint equation. To this end, we need algorithmic (symbolic) differentiation: we gain one equation.
- The differentiation may involve further equations and variables.
- Finally, algebraic loops are likely to result.

- Let us illustrate the ideas behind the Pantelides Algorithm:



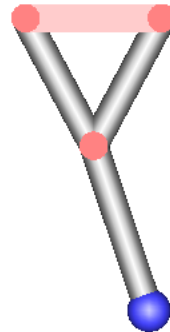
Pantelides: Illustration

- Here, we have chosen **a** as state-variable. But we could choose **b** as well.

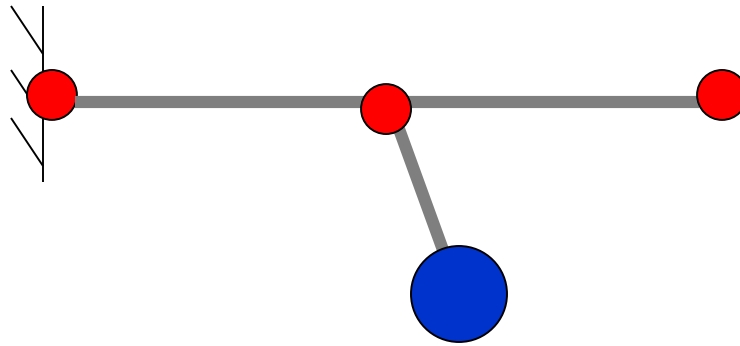


- Does it matter, if we choose **a** or **b** as state variable?
- If the constraint between **a** and **b** is linear (with constant coefficients), it does not matter.
- But otherwise an inadequate state-selection can lead to numerical singularities during the simulation.

- So what is the situation like for the kinematic loop example?
- Obviously, the constraints are highly non-linear.
- Dymola tells us there is a non-linear system of size 20 that can be reduced to 3 iteration variables.

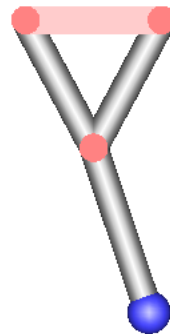


- If Dymola would just blindly choose s and v of the prismatic joint to be state-variables a singularity could occur.



- When the prismatic joint is stretched to the maximum length:
 $v = 0$.
- However, the loop is not necessarily at rest! We just lost all information about the velocity of the loop!
- If we choose, ϕ and w of the wall revolute-joint to be the states, the problem disappears. But Dymola cannot know this. This is expert knowledge.

- Due to the non-linear constraints, Dymola cannot eliminate potential state-variables.
- Instead, a set of redundant state-variables is chosen and the best subset is chosen dynamically during the simulation.
- However, this is demanding and potentially time-consuming.
- Can't we help Dymola?



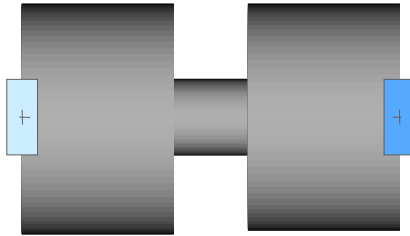
- In Modelica there is the StateSelect Attribute.
- We can determine it for any Real variable. Example:
`SI.Angle phi(stateSelect=StateSelect.always)`
- There are five different levels available for state selection:
`StateSelect.always`
`StateSelect.prefer`
`StateSelect.default`
`StateSelect.avoid`
`StateSelect.never`
- `StateSelect.prefer` is used to show that this a state-variable that shall be taken in case of linear constraints.
- `StateSelect.always` is used to show that this a state-variable that shall be taken even in case of non-linear constraints

- We can apply the StateSelect Attribute in the modifier:

```
Joints.Revolute revolute(  
    phi(stateSelect=StateSelect.always),  
    w(stateSelect=StateSelect.always));  
  
Joints.Revolute revolute1;  
  
Joints.Revolute revolute2;  
  
Joints.Revolute revolute3(  
    initialize=true,  
    w_start=0,  
    phi_start=0);
```

- Now, there is no dynamic state-selection anymore.
- Also the non-linear system of equations could be further simplified.
- Simulation is much faster.

It is more convenient when the state selection is integrated into the model:



- Hence we add a Boolean parameter “enforceStates”.
- And couple it with the attribute.

```
model Revolute
  Interfaces.Frame_a frame_a;
  Interfaces.Frame_a frame_b;
  SI.Angle phi (stateSelect =
    if enforceStates then StateSelect.always
    else StateSelect.prefer);
  SI.AngularVelocity w (stateSelect =
    if enforceStates then StateSelect.always
    else StateSelect.prefer);
  SI.AngularAcceleration z;
  parameter SI.Angle phi_start = 0;
  parameter SI.AngularVelocity w_start=0;
  parameter Boolean initialize = false;
  parameter Boolean enforceStates = false;

  [ ... ]
  equation
    frame_a.phi + phi = frame_b.phi;
    w = der(phi);
    z = der(w);
    frame_a.x = frame_b.x;
    frame_a.y = frame_b.y;
  [ ... ]
end Revolute
```

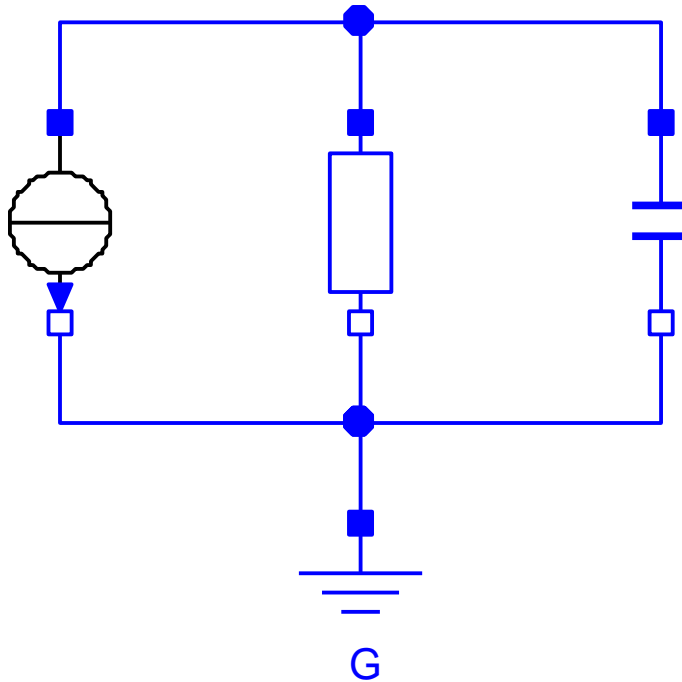
- The index describes the level of difficulty to transform a given system from implicit DAE-form into explicit ODE-form.

$$F(\mathbf{x}_p, d\mathbf{x}_p/dt, \mathbf{u}, t) = 0 \quad \rightarrow \quad d\mathbf{x}/dt = f(\mathbf{x}, \mathbf{u}, t)$$

- An index-0 system represents a system that can be brought into ODE-form simply by permuting its equations.
- The **differential index** represents the maximum number a variable needs to be differentiated in order to retrieve an index-0 system.
- The **perturbation index** is equal to the differential index if the system contains no algebraic loops. Otherwise it is larger by one.
- Typically, the term index refers to the perturbation index.
- Strictly, index-reduction was devised for numerical DAE-solvers. Using these terms for structural analysis is thus stretching the terminology.

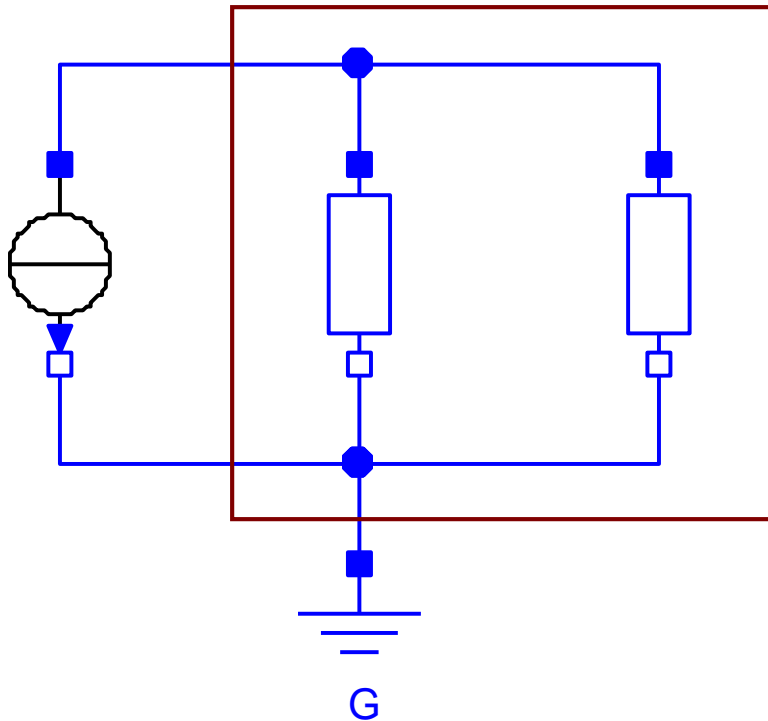
Example: Index

- Differential Index: 0
- Perturbation Index: 0



Example: Index

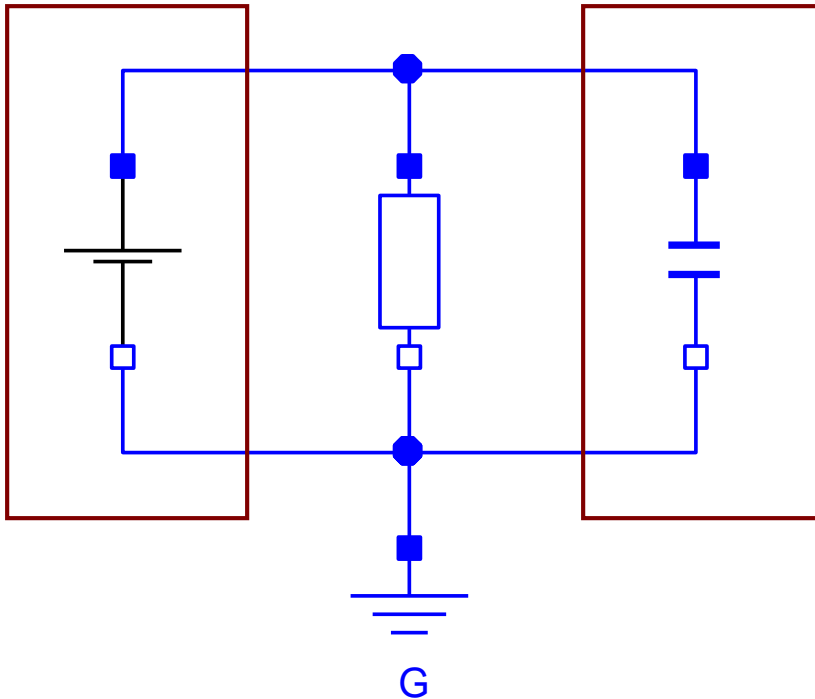
- Differential Index: 0
- Perturbation Index: 1



The two parallel resistor create an algebraic loop for the division of current.

Example: Index

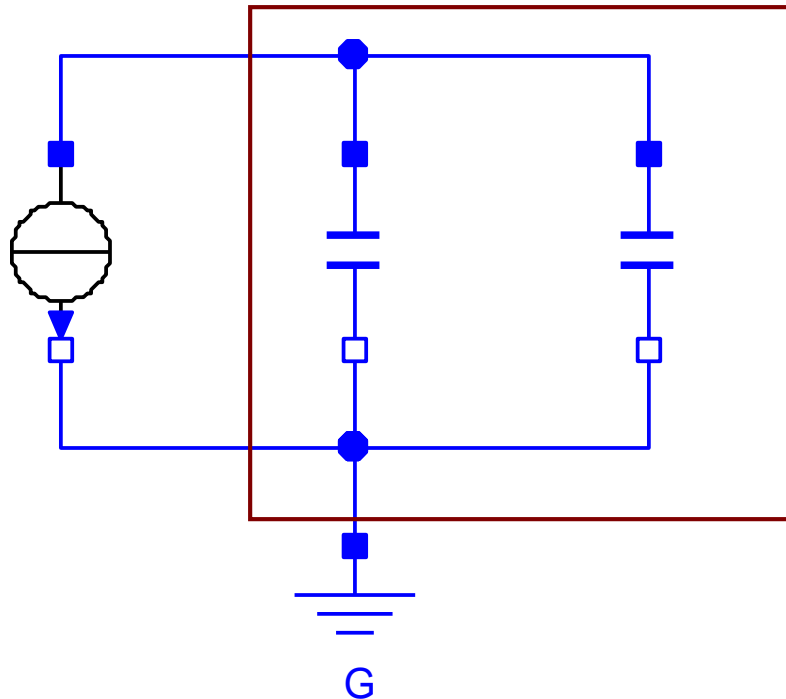
- Differential Index: 1
- Perturbation Index: 1



The voltage source determines the voltage at the capacitor.

The voltage must be differentiated in order to determine the current through the capacitor.

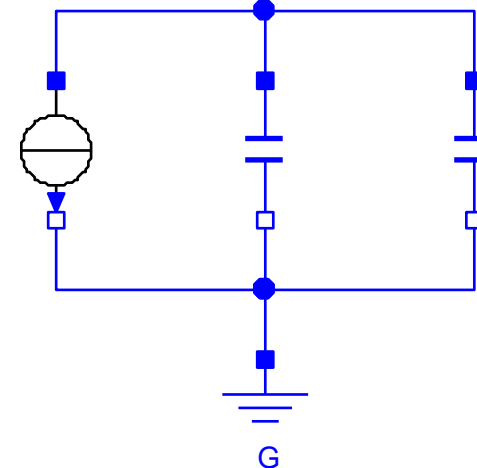
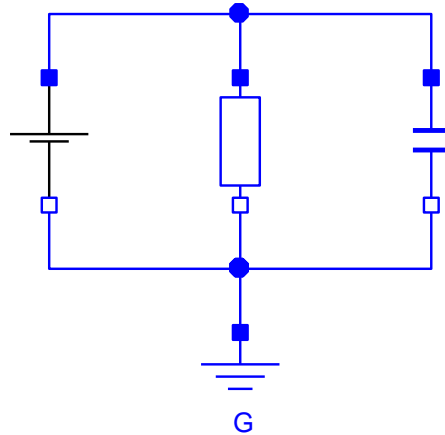
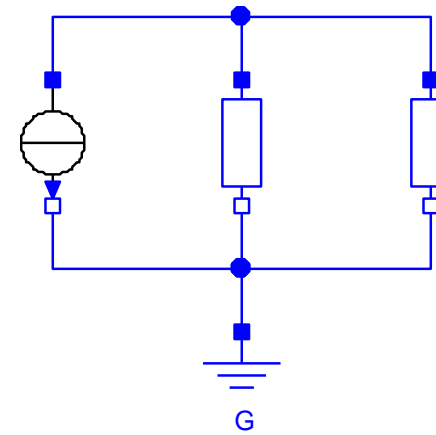
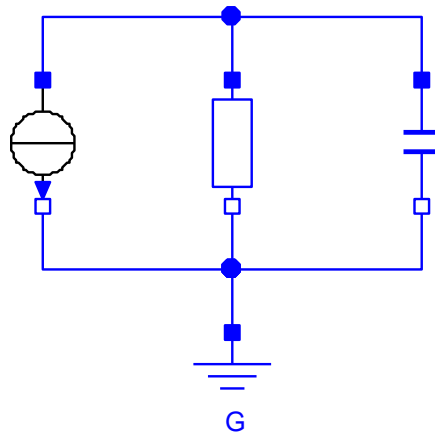
- Differential Index: 1
- Perturbation Index: 2



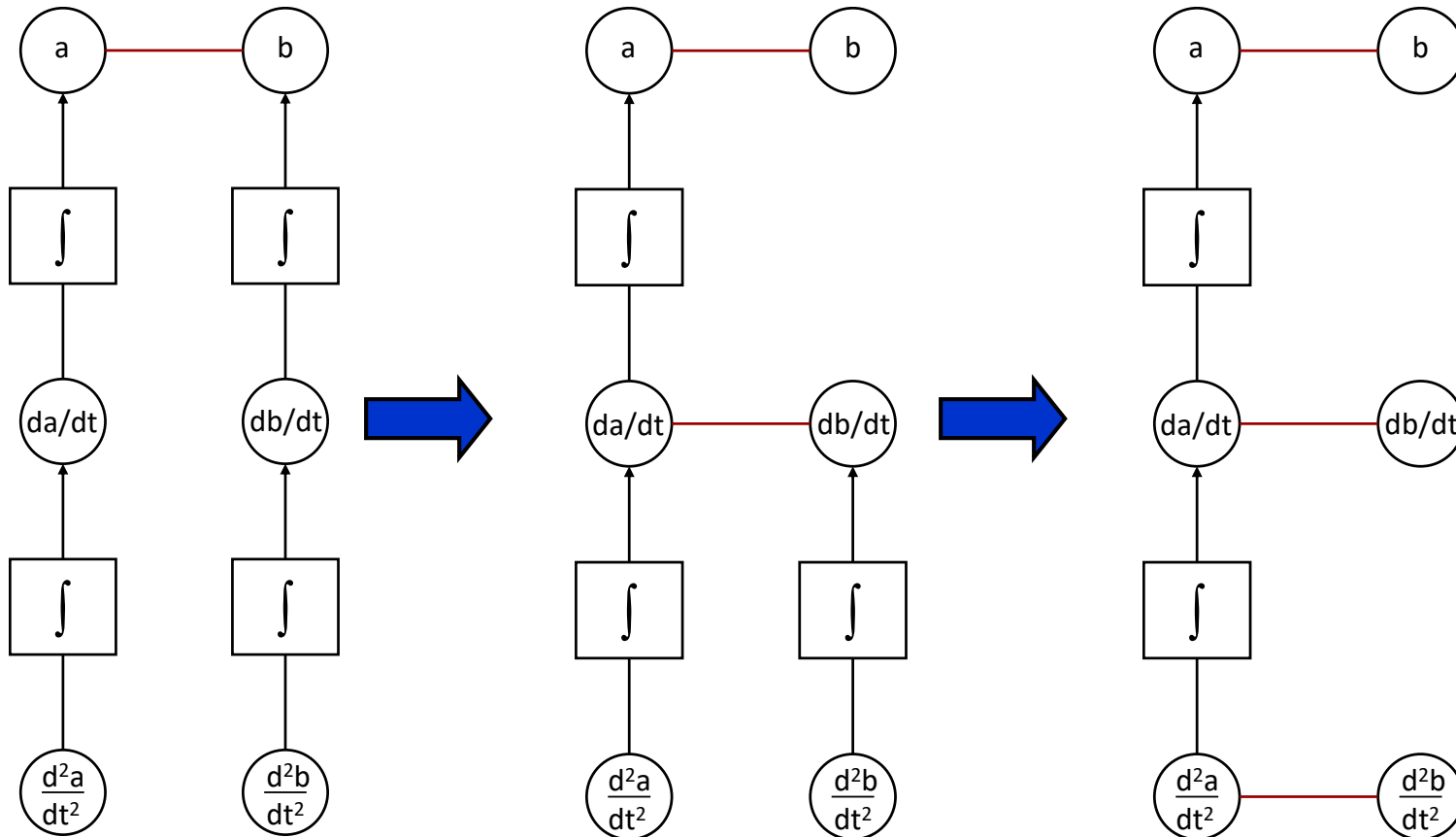
Both voltages of the capacitors are equal. Only one differential equation is used for time-integration. The system needs to be differentiated once.

The two parallel capacitors create an algebraic loop for the division of current.

- Set up the equations for each circuit and transform them to ODE-Form
Apply Tearing-Algorithm and Pantelides if necessary



- Beware! Certain system may require multiple differentiations...



- Multiple differentiations lead to a higher differential index.
- Systems with a perturbation index of 3 and higher are called: *higher-index systems*
- Most mechanical multi-body systems are higher-index systems.

Questions ?