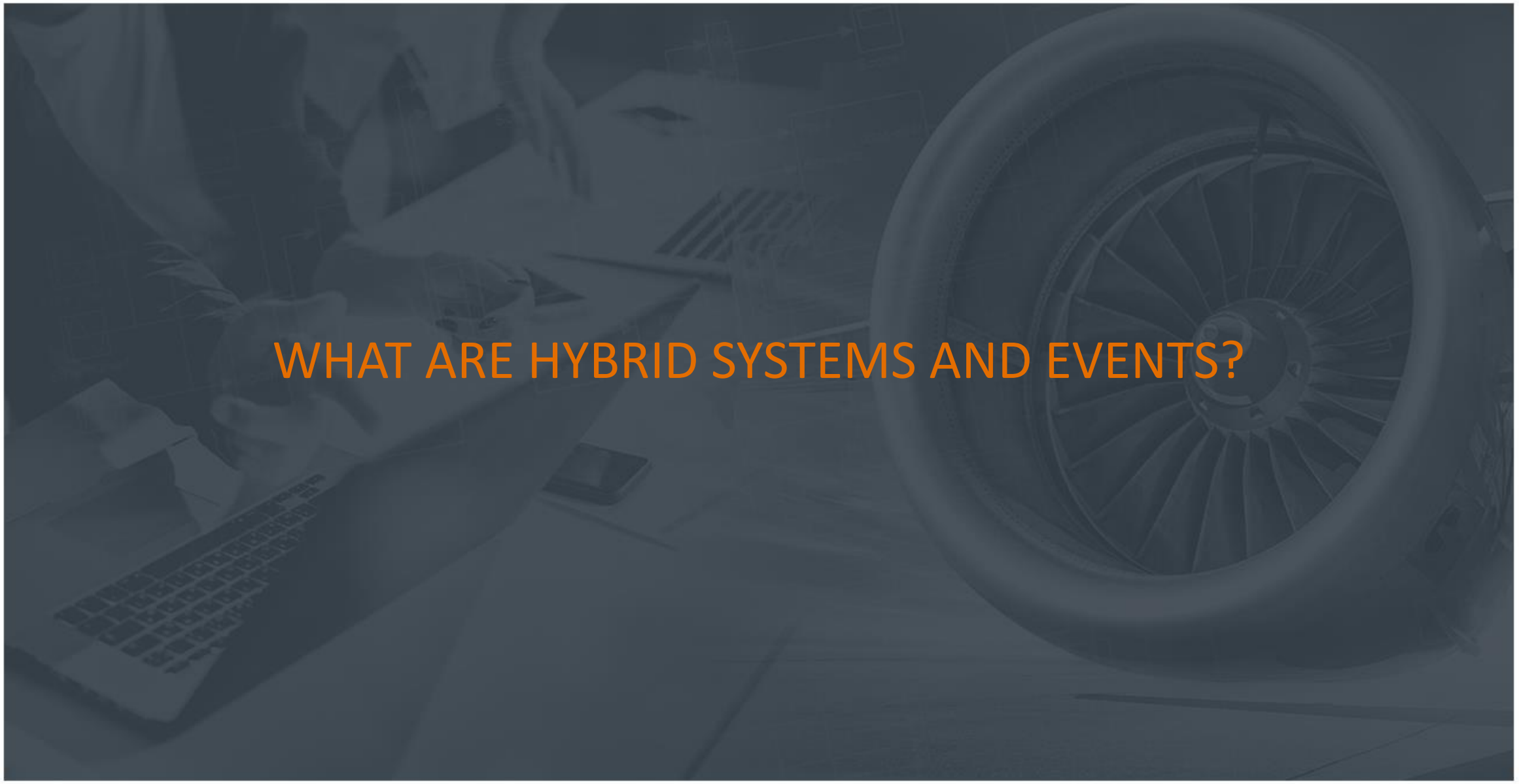
INTRODUCTION TO MODELICA

Hybrid systems

Modelon

OVERVIEW

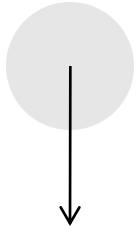
- What are hybrid systems and events?
- Chattering
- Avoiding events



WHAT ARE HYBRID SYSTEMS AND EVENTS?

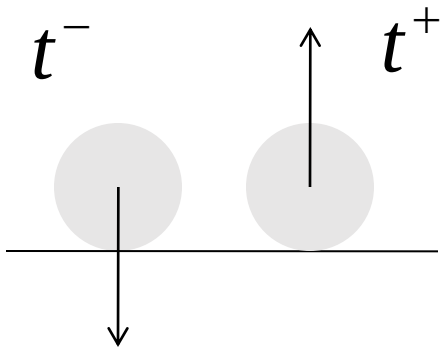
WHAT IS A HYBRID SYSTEM?

- A hybrid system contains both continuous and discrete parts.
- For a bouncing ball we have:



A continuous part:
Ball in gravity field

$$\ddot{x} = -g$$



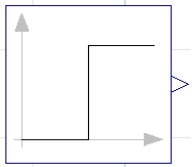
And a discrete part:
Ball bounces back when hitting the surface

$$\dot{x}(t^+) = -\dot{x}(t^-)$$

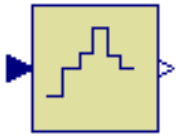
For an efficient and reliable handling of discontinuous dynamics, standard integrators need help!

COMMON DISCONTINUITIES

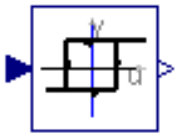
- Discontinuous input functions, e.g. a step



- Sampled systems, e.g. for digital controllers



- Hysteresis



- In these examples the output signal jumps from one value to another, depending on input and possibly internal states.
- Note that the output from these blocks cannot be differentiated at switching points.

WHAT IS AN EVENT?

An event corresponds to a discrete change in the equation system.

To an event is associated:

- A time point, when the change occurs
- A set of equations
- A condition, which activates one of the conditional equations

```
y = if x>0 then 1 else -1;
```

- Relations and Boolean expressions trigger events.
- Real variables that are changed during events are considered discrete. Integers and Booleans are always discrete.

EVENT HANDLING

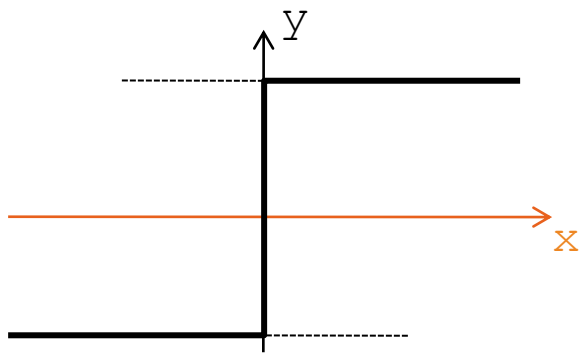
```
y = if x>0 then 1 else -1;
```

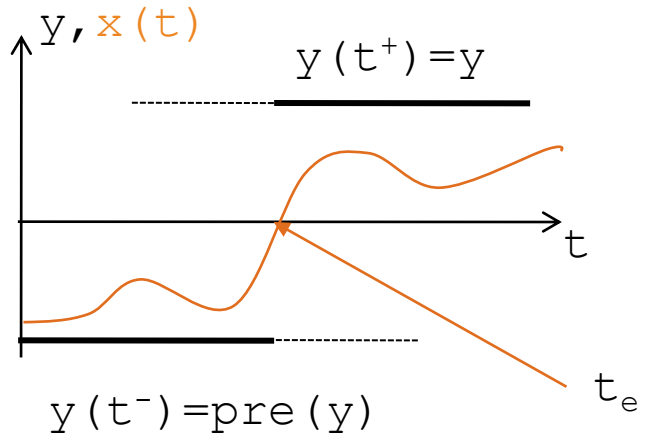
During simulation:

1. An event is generated when the solution of x crosses 0. When this is first detected, t> t_e and y=-1

   This requires that y() can also be evaluated beyond the event limit (dashed lines)

2. The continuous time integration is stopped.
3. The time t_e for which x(t_e)=0 is searched using a crossing function. (event iteration)
4. An initialization problem is solved at t= t_e.
5. The continuous integration continues.





TIME EVENTS AND STATE EVENTS

- A time event is an event that depends on the simulated time only, an example is a step signal:

```
y = if time<t_step then y_0 else y_0 + y_step;
```

- The time for a time event is easy to find.
- The state event depends on one or more states of the model, for example a contact force as for the bouncing ball:

```
model BouncingBall
 parameter Real c=1e6;
 parameter Real m = 1;
 Real a, v, h(start=0.1), f;
equation
 m*(a+9.82)=f;
 der(v) = a;
 der(h) = v;
 f = if h>0 then 0 else -c*h;
end BouncingBall;
```

The time when the ball reaches the surface depends on the state h. This means that the solver will have to iterate to find the right time step. Therefore state events require more CPU time than time events.

WHY EVENTS?

- Detecting and triggering an event ensures that the discontinuities are treated in a numerically sound way.
- If no event is generated at a time of discontinuity, a small step size of the integrator is required for a high accuracy.

In the case of the bouncing ball, missing an event would postpone the bouncing instant after that the ball has reached the ground!

WHEN-STATEMENT

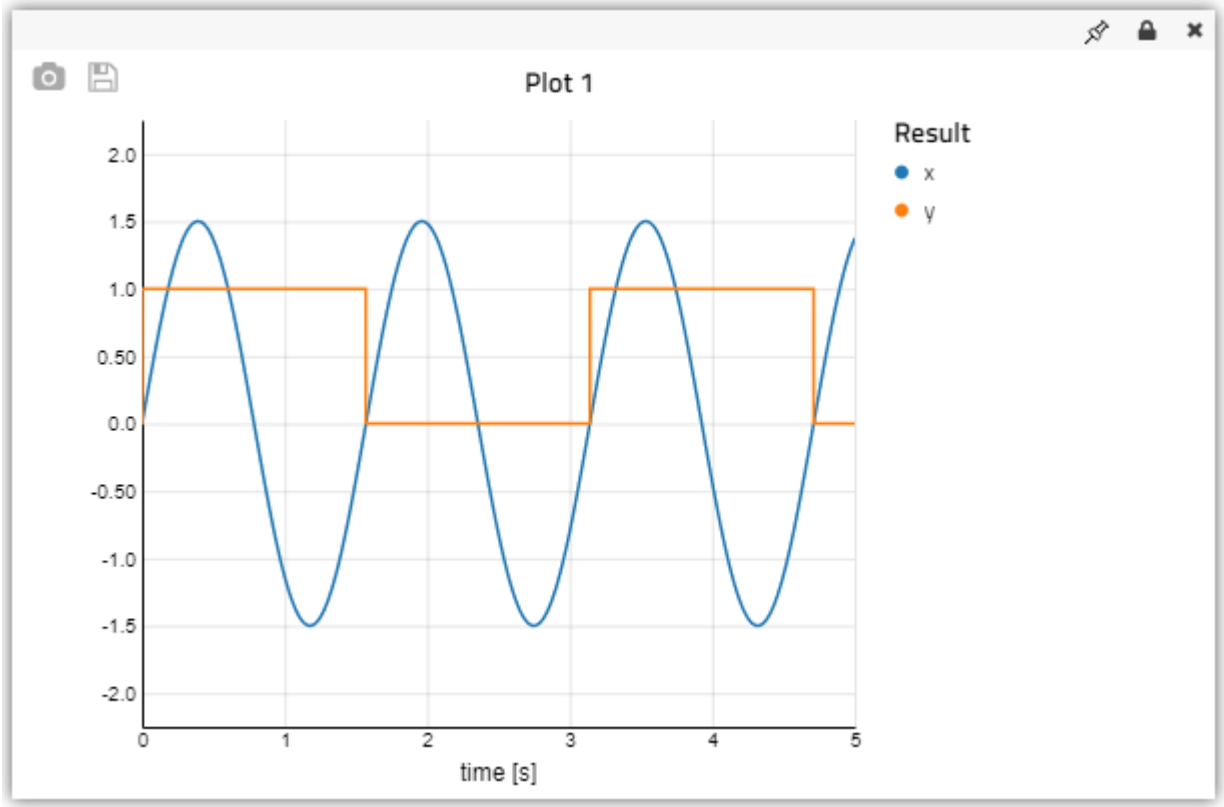
Additional equations can be added during events using the **when** statement. These equations are only active during the event when the conditions become true, and are deactivated during continuous integration.

```
when expression then
   { statement ";" }
{ elsewhen expression then
   { statement ";" } }
end when
```

- Restrictions
  - A When-statement shall not be used within a function
  - When-statements cannot be nested
  - When-statements may not occur inside while, if- and for-clauses in algorithms.

USING EVENTS

```
1 model WhenDemo
2  parameter Real A=1.5, w=4;
3  Real x;
4  Boolean y;
5 equation
6  x = A*sin(w*time);
7  when x > 0 then
8   y = not pre(y);
9  end when;
10 end WhenDemo;
```



REINITIALIZATION

State variables can be reinitialized during events using reinit(), e.g.:

```
der(v) = -9.82 "gravity acceleration";
der(h) = v "derivative of height is speed";
when (h<0) then
 reinit(v, -pre(v)) "change direction of speed";
end when "when hitting surface";
```

WHAT ABOUT MULTIPLE EVENTS?

In the case of multiple and simultaneous events, the synchronous data-flow principle restricts the use of the when statement:

```
equation //illegal example
when condition1 then
 close = true;
end when;
when condition2 then
 close = false;
end when;
```

```
equation //legal example
when condition1 then
 close = true;
elsewhen condition2 then
 close = false;
end when;
```

SYNCHRONOUS DATA FLOW

For the sake of **solvable** simulation models and **deterministic** results, Modelica is based on the synchronous data flow principle:

- **Single assignment rule**: the total number of equations is at all times identical to the total number of unknown variables
- **Concurrency principle:** At every time instant, during continuous integration and at event instants, the active equations express relations between variables which have to be fulfilled *concurrently (equations are not active if the corresponding if-branch, when-clause or block in which the equation is present is not active).*
- **Synchronous principle:** events take no simulation time.
- All variables keep their actual values until these values are explicitly changed. Variable values can be accessed at any time instant during continuous integration and at event instants.

Modelon

CHATTERING

- Chattering is an undesirable phenomenon caused by rapid switching between different dynamics at an event barrier.
- Very difficult to handle automatically
- Potentially more than one discrete state involved
- Switching may occur only for rare parameter values and operating points of model.
- Common problem in Hybrid models

CHATTERING

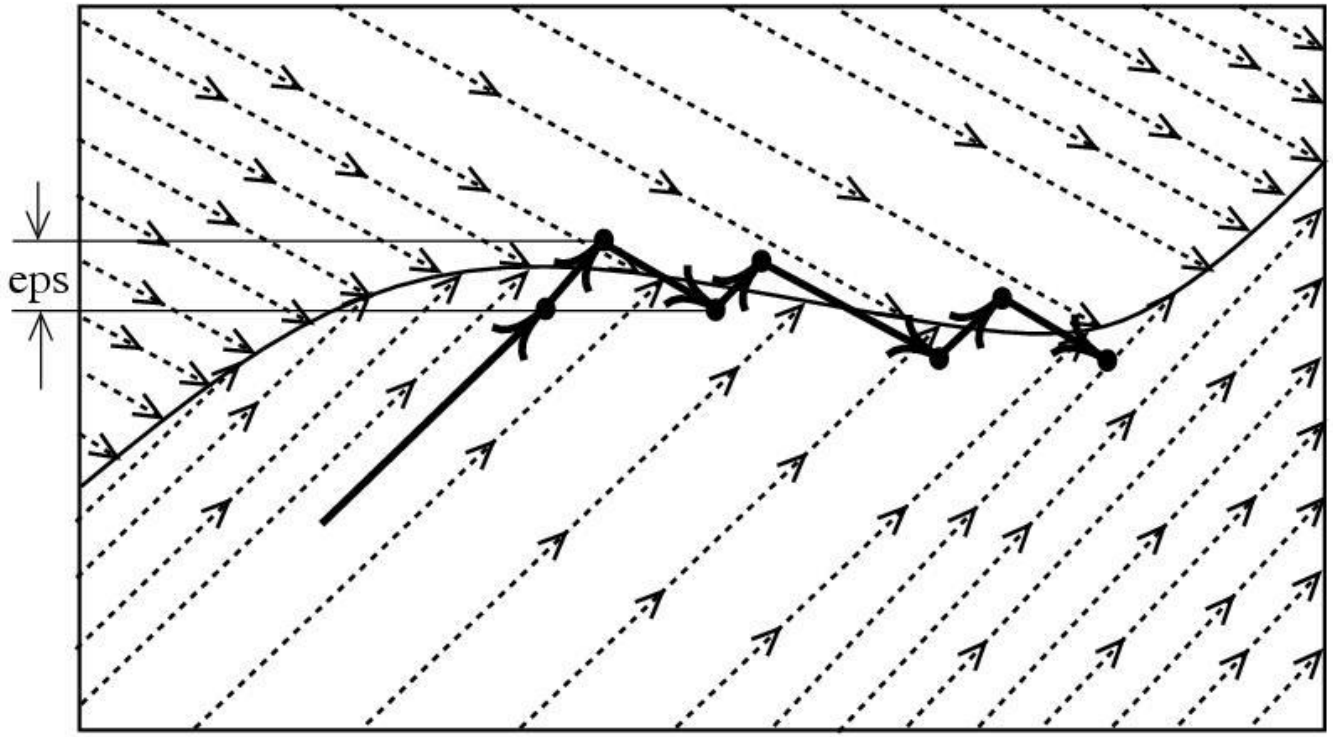
- Simplistic model:

```
model Chattering1
  Real x;
initial equation
  x = 1.0;
equation
  der(x) = if x > 0.5 then -1.0 else 1.0;
end Chattering1;
```

- Note that the derivative is not continuous across the event expression

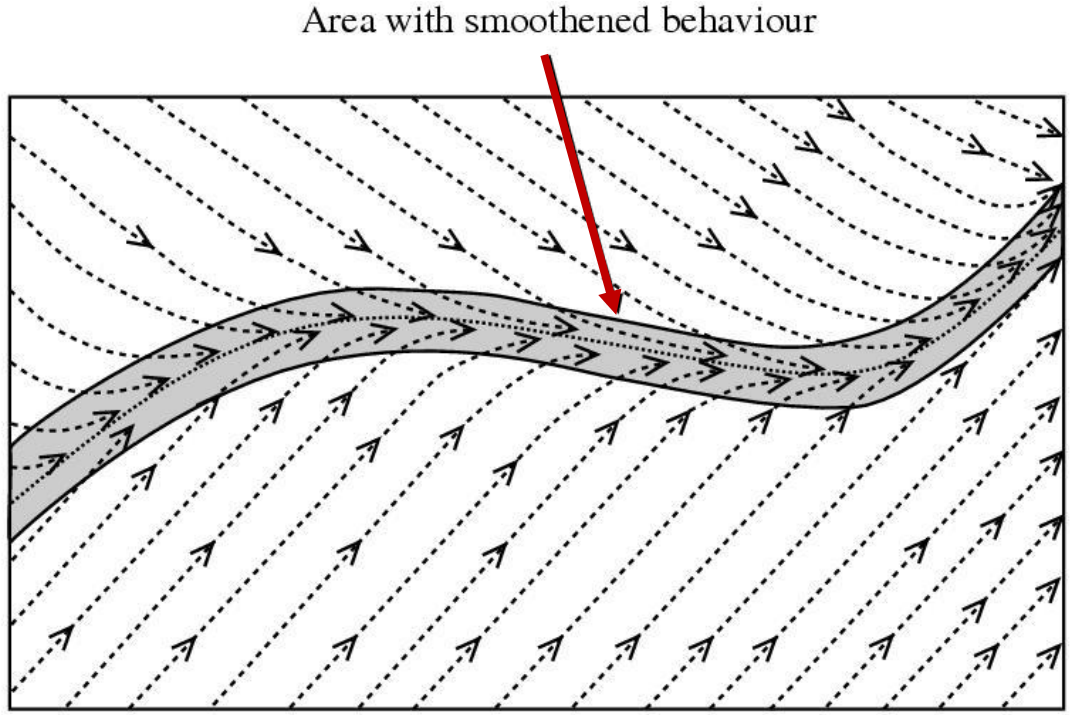
CHATTERING

- The time integrator is chattering when it comes close to the event barrier.
- Currently no known algorithm/solver for detecting chattering and automatically switching to Filippov-solution, except in trivial cases



CHATTERING

- Workaround:
  - Smoothen the discontinuity (choose different approximation for model)
  - Find conditions for chattering explicitly and provide the solution and additional mode changes “by hand”.



AVOIDING EVENTS

- There are cases when events are not wanted, for example when preventing division by zero.
- Then built-in operators can be used, e.g.:
  - `y=smooth(1,if x>eps then 1/x else 1/eps);`
  - `y=noEvent(if x>eps then 1/x else 1/eps);`
  - `y=1/(max(x,eps));`

- In some situations, event triggering cannot be avoided, Boolean and discrete variables always generate events.
  - `Boolean x_is_small = noEvent(x<eps) “not legal”;`

AVOIDING EVENTS

- noEvent ()
  - Expressions are taken literally instead of generating crossing functions
  - Since there is no crossing function, there is no requirement that the expression can be evaluated beyond the event limit.

- The smooth() operator allows to define how many times an expression is continuously differentiable.

```
y = smooth(2,if x>0 then x^3 else x^4);
```

- min(), max(), abs() return continuous variables if the input is continuous, so no events are generated for these operators.

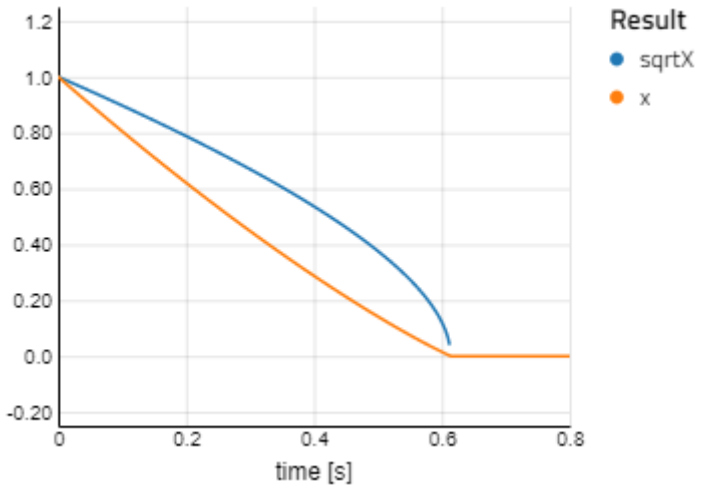
NOEVENT()

Consider the following model:

- This model will fail when x<0, the reason is that:
  - sqrt(x) cannot be evaluated for x<0
  - A crossing function is required
    - Must be able to evaluate sqrt(x) for any x around x=0
- Applying noEvent() solves the problem:

```
y=noEvent(if x>0 then -sqrt(x)-1 else 0);
```

```
model Sqrt
Real x(start=1);
equation
  der(x) = if x>0 then -sqrt(x)-1 else 0;
end Sqrt;
```



SMOOTH()

Consider the following model:

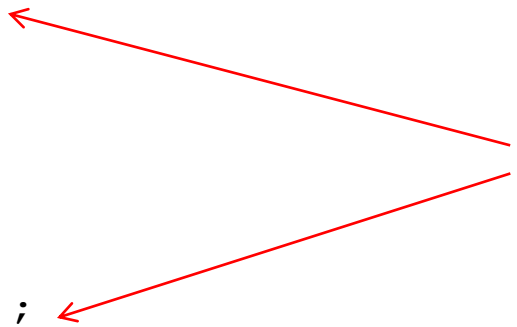
```
model ContinuouslyVariableGear
    parameter Real eps=1e-6;
    Real ratio;
    .Modelica.Mechanics.Rotational.Interfaces.Flange_a a annotation(...);
    .Modelica.Mechanics.Rotational.Interfaces.Flange_b b annotation(...);
    .Modelica.Blocks.Interfaces.RealInput u annotation(...);
equation
    ratio = if abs(u)>eps then u else eps;
    ratio * a.tau + b.tau=0;
    der(a.phi)=ratio*der(b.phi);
end ContinuouslyVariableGear;
```

- Applying smooth() to the expression for ratio allows the model designer to tell the compiler to ignore the discontinuity at u=0, and no event is generated.

SAMPLE OPERATOR

- `sample(start,interval)`
  - Returns `true` and triggers time events at time instants `start+i*interval (i=0,1,...)`. During continuous integration the operator returns always false. The starting time start and the sample interval need to be parameter expressions and need to be a subtype of Real or Integer.

```
Real x; /* discrete*/
input Real u;
output Real y;
equation
when sample() then
  x = a*pre(x)+b*pre(u);
end when;
y = x;
```



Variable x becomes discrete
because of equation inside sample

EVENT CAUSING OPERATIONS

- Operators:
  - `initial()`
  - `terminal()`
  - `when(boolean)`
  - `sample(start,interval)`
  - `edge(boolean)`
  - `change(discrete)`
  - `sign(Real)`

  - `pre(Real)`
  - `reinit(Real,expr)`

- See Modelica Reference for operator definitions

- Mathematical Functions:
  - `div(x,y)`
  - `mod(x,y)`
  - `rem(x,y)`
  - `ceil(x)`
  - `floor(x)`
  - `integer(x)`

OPERATIONS NOT CAUSING EVENTS

- `min(Real,Real), max(Real,Real)`
- `Integer(Enumeration)`
- `abs(Real)`
- `sqrt(Real)`
- `noEvent(expr)`
- `smooth(integer,expr) (may generate event)`

WORKSHOP 4.3

In this workshop you will:

- Compare behavior, with and without event handling
- Investigate validity range of 2 bouncing ball implementations