WORKSHOP 4.3

Hybrid Systems

Contents

Introduction

This workshop exercises different hybrid examples.

The first example implements 3 different versions of a variable calculated to a specific saturation level and evaluates the solver behavior depending on the implementation.

The last example looks at 2 different versions of the bouncing ball and compares the validity range of them.

Hybrid examples: Saturation

1. Create a new package **W3_HybridSystems** inside the course package you created in one of the previous lectures. Add a sub-package called **HybridExamples**.
2. Create a model *Saturation1* and define a state variable *x* that is increasing with time.
3. Add a variable *y* that oscillates between -1.0 and 1.0, depending on *x*, e.g., using a sin function.
4. Add a variable *y_saturated* that is saturated between *y_min* and *y_max* using if statements as shown in Figure 1.

```
model Saturation1
  Real x;
  Real y;
  Real y_saturated;
  parameter Real y_min=-0.5;
  parameter Real y_max=0.5;
  parameter Real f=200;
equation
  der(x) = 1;
  y = sin(f*x);
  y_saturated = if y < y_min then y_min else if y > y_max then y_max else y;
  annotation (...);
end Saturation1;
```

Figure 1 Code for Saturation1 model

5. Simulate and check the number of state events.

6. Now, we will eliminate the events. Use the *noEvent()* operator on the equation for y_saturated to eliminate the events as shown in Figure 2. Duplicate the model, add the noEvent() and call it *Saturation2* and simulate again.

```
y_saturated = noEvent(if y < y_min then y_min else if y > y_max then y_max else y);
```

Figure 2 noEvent code

7. Duplicate the model and call it *Saturation3* and rewrite using *min()* and *max()* operators, see Figure 3.

```
y_saturated = min(y_max, max(y_min, y));
```

Figure 3 Limiting the y_saturated

What are the differences?

Bouncing ball

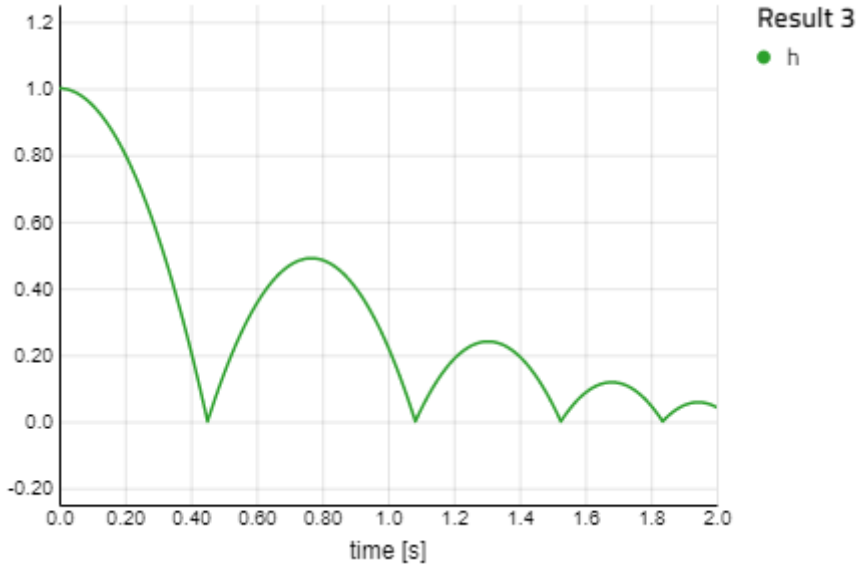
In the following section you will test the bouncing ball example, and investigate the validity of the model.

1. Open the BouncingBall_simple model, and view the code.

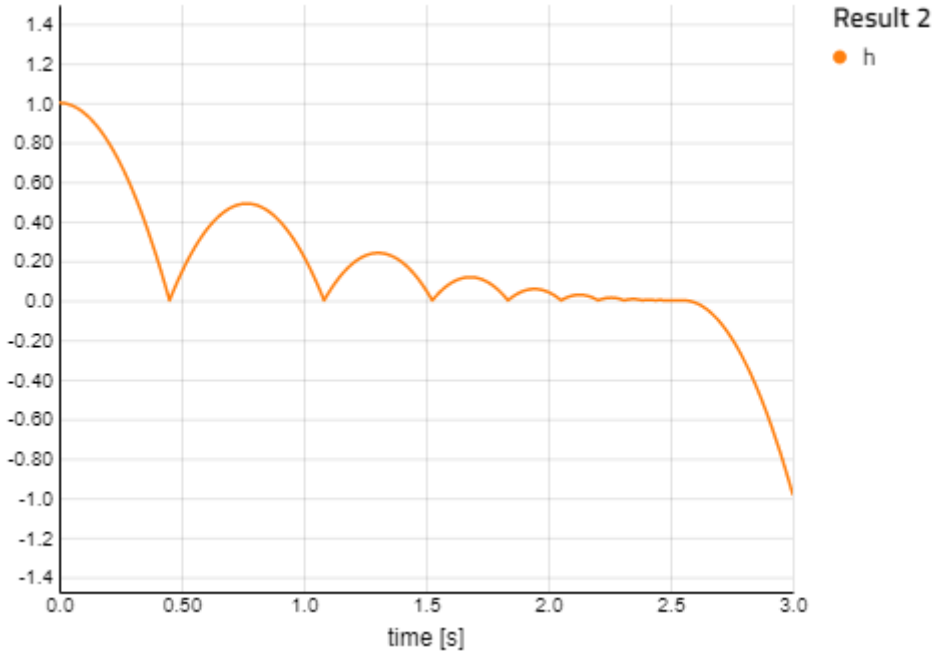
```
model BouncingBall_simple
  /* After around ~3 seconds h becomdes negative due to numerical errors. */
  parameter Real e=0.7 "bounce coeff";
  parameter Real g=9.81 "gravity acc.";
  Real h(start=1) "height of ball";
  Real v "velocity of ball";

equation
  der(h) = v;
  der(v) = -g;
  when h < 0 then
    reinit(v, -e*v);
  end when;
end BouncingBall_simple;
```

2. Simulate the model for 2s, and plot the variable h.



3. Now simulate the model again, for 3s.



4. As can be seen, at some point beyond 2.5s, the ball falls through the table. This is due to tolerance issues when evaluating the crossing function, and the reinit value for the speed has the wrong sign. In short, the model fidelity of the impact cannot handle the case when the ball comes to rest.
5. Open the more sophisticated model BouncingBall_advanced.

```
model BouncingBall_advanced
  /* A more advanced implementation of the bouncing ball model in order to avoid negative h.
Model is from https://www.modelica.org/events/modelica2006/Proceedings/sessions/Session5a2.pdf

"If the ball is on the ground and
has a negative velocity but the impact variable has
not changed the velocity will be reset to zero and the
boolean variable flying is set to false. The derivative
of the velocity in the equation section is then
also set to zero due to the if-expression"
  */

  parameter Real e=0.7 "bounce coeff";
  parameter Real g=9.81 "gravity acc.";
  Real h(start=1) "height of ball";
  Real v "velocity of ball";
  Boolean flying(start=true);
  Boolean impact;
  Real v_new;
equation
  impact = h <= 0.0;
  der(v) = if flying then -g else 0;
  der(h) = v;
  when {impact and v <= 0.0} then
    // edge() is expanded into (b and not pre(b))
    v_new = if edge(impact) then -e*pre(v) else 0;
    flying = v_new > 0;
    reinit(v, v_new);
  end when;
  annotation (...);
end BouncingBall_advanced;
```

6. This model includes a mechanism for detecting if the ball falls through (flying) and will capture the ball when that state comes into play. Simulate the model for 3s.