BCP - A BENCHMARK FOR HYBRID MODELLING AND STATE EVENT MODELLING

Andreas Körner[1], Bernhard Heinzl[1], Matthias Rößler[1], Stefanie Winkler[1], Irene Hafner[1], Günter Schneckenreither[2], Günther Zauner[2], Niki Popper[2]

[1] Vienna University of Technology, Institute for Analysis and Scientific Computing
1040 Vienna, Wiedner Hauptstraße 8, Austria

[2] dwh Simulation Services
1070 Vienna, Neustiftgasse 57-59, Austria

akoerner@asc.tuwien.ac.at (Andreas Körner)

Abstract

Modelling and simulation of hybrid systems is getting more and more important in advanced modelling theory and application. Therefore, the requirements regarding flexibility on modern simulators are getting higher and higher. The necessity of fast and stable algorithms is increasing considering higher complexity in simulation questions. ARGESIM started in 1990 the series Comparison of Simulation Software in the journal Simulation News Europe (SNE). These software comparisons developed towards benchmarks not only for simulation tools but also for modelling tools and for modelling techniques and modelling approaches. To see how modelling and simulation environments deal with state events of different order, three classical examples are discussed in this benchmark. These parts offer a spectrum of questions for testing basic features and they represent minimum requirements to hybrid simulators regarding state events. Sample solutions are carried out in MATLAB® and partly in SIMULINK® and can be used for comparison with solutions calculated with other simulation environments.

Keywords: hybrid modelling, state event modelling, bouncing ball, electrical circuit, pendulum

Presenting Author's Biography

Andreas Körner. He passed his bachelorstudy in electrical engineering and his masterstudy in telecommunications. Now he is on the way to finish his diplomastudy in technical mathematics. Before he started his academic studies he passed a higher technical engineering college for electronics and technical informatics.
His field of activity include physical modelling and simulation. Because of his education in electrical engineering, his work is focused on technical applications. Another scope of work is the usage, design and development of e-learning systems for mathematical education and education in modelling and simulation.

1 Introduction

The following benchmark is structured in three independent assignments. Each of this assignment consists of the definition and the sample solution in MATLAB® or SIMULINK®.

At Vienna University of Technology, in 1990 comparisons for simulation software were set up. These new comparison C20 - A Benchmark for Hybrid Modelling and State Event Modelling deals with different hybrid modelling topics.

To implement and test the models, we used MATLAB and SIMULINK as simulation software. MATLAB is a platform independent software developed by MathWorks. Compared to other mathematical software, MATLAB mainly uses numerical methods for solving problems instead of analytical ones.

SIMULINK is a continuous simulator with a graphical user interface, embedded in the MATLAB environment. SIMULINK has also been developed by MathWorks and needs MATLAB for execution. By using predefined blocks, different tasks can be modelled graphically. Because it is based on MATLAB, the solutions are also calculated numerically, and therefore several ordinary differential equation solvers (ODE solvers) can be selected.

2 Rotating Pendulum with Free Flight Phase

This example describes a classical idealized pendulum on a rope with damping. The body, which is considered a point mass, is connected to a fixed point in a room via a rope of given length. The rope is assumed to be non-elastic and without mass. As a simplification, it is presumed that the body can move freely only in the plane, i.e. the area of a circle with a radius equal to the length of the rope.

Using these assumptions, the behavior of the pendulum can be described with two states:

- If the rope is tight, the body is moving along a circular path (state "bound"). Since the radius is fixed, the movement in this state can be described by one variable, i.e. the angle between the rope and the upper perpendicular.
- If the rope is loose, the body is free falling (state "flying") until the rope is tight again. This movement is defined by two state variables, i.e. the Cartesian x- and y-coordinates of the body.

The different states are illustrated in Fig 1, where you can also see that the two cases have different numbers of degrees of freedom, which is the reason why the state space dimension changes as the pendulum alters its state.

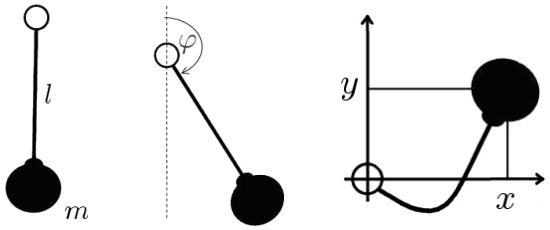

Fig. 1 Left: Pendulum with mass m and length l; middle: Classical oscillating pendulum (state "bound") with angle φ as degree of freedom; right: Free falling pendulum mass (state "flying") and Cartesian coordinates as degrees of freedom.

2.1 State "Bound"

How the various forces are affecting the mass during the circular movement in the "bound" state can be seen in Fig. 2.

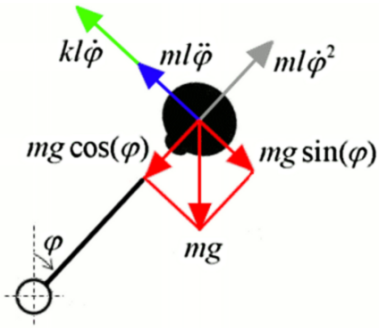

Fig. 2 Forces affecting the mass in the "bound" state

The state "bound" can be described by using the angular momentum balance, which, with regard to Fig. 2, leads to

$$\ddot{\varphi} + \frac{k}{m}\dot{\varphi} - \frac{g}{l}\sin(\varphi) = 0, \tag{1}$$

with the damping factor k, mass m, length of the rope l and the earth acceleration g. Furthermore, the force on the rope

$$F = -mg\cos(\varphi) + ml\dot{\varphi}^2 \tag{2}$$

provides a criteria when a state change from "bound" to "flying" occurs. If this force becomes lower than zero, the gravitational force outweighs the centrifugal force and the pendulum is entering the "flying" state.

2.2 State "Flying"

Being in the "flying" state, the body has one more degree of freedom compared to the state "bound". The motion of the mass is described in this state by conservation of momentum in x- and y-direction:

$$m\ddot{x} = -k\dot{x}, \tag{3}$$
$$m\ddot{y} = -mg - k\dot{y}. \tag{4}$$

Using an additional geometric equation,

$$r^2 = x^2 + y^2, \tag{5}$$

it is tested if the rope is loose or completely tight. If r fulfills $r \geq l$, the rope has become tight again, forcing the body back on the circular path and the pendulum switches to the "bound" state again.

Summing up, Fig. 3 shows an overview of the two different models with criteria for state changes.

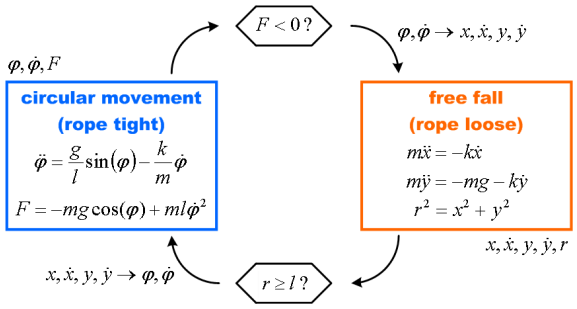

Fig. 3 Summary of the two state model with criteria for state changes

2.3 Modification

As a modification of this example, we also considered that a region with different damping factor could occur. This region can be explained for example by the body diving in a medium of different density. However, it is assumed that the different damping factor is affecting the pendulum only in the "bound" state. The "flying" state remains unchanged. An illustration of this situation is shown in Fig. 4.

In addition to the "bound" and "flying" state from the standard example, it now has to be switched between three models, where the third one is described by the same equations as the "bound" state, except for a different value of the damping factor.

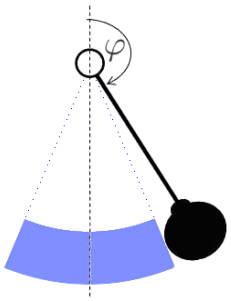

Fig. 4 Region with different damping factor in the "bound" state

2.4 Experiments and Results

This pendulum example was implemented in MATLAB as well as in SIMULINK. For detection of state events in MATLAB, we made use of the event location property of the MATLAB ODE solvers in connection with the `odeset` command and an event function.

The graphical component oriented simulation in SIMULINK on the other hand uses If blocks and If Action subsystems to detect state events and enable or disable corresponding subsystems.

Since the "bound" state is described with polar coordinates whereas the "flying" state uses Cartesian coordinates, coordinate transformation has to be performed at each state change to obtain the new initial conditions for the ODEs.

Implementation with MATLAB

The main program in MATLAB contains a `while` loop for repeated changes of the state model. This `while` loop is terminated when a special exit condition is satisfied. In our example, this exit condition is the maximum oscillation being less than $\pi/10$.

The event detection in MATLAB is realized using the `odeset` command, e.g.

```
options=odeset('Events',@eventloc).
```

The function `eventloc` is the so-called event function. The MATLAB ODE solver can detect an event by locating transitions to, from, or through zeros of this event functions and can terminate the calculation in such a case [1]. The event function of the "bound" state, for example, is of the form

```
function [value,isterminal,direction]=
    eventloc(t,w)
global m L k k2 g
value=zeros(3,1);
value(1)=-m*g*cos(w(1))+m*L*w(2)^2;
value(2)=sin(w(1))^2+10*w(2)^4-
    sin(pi/10)^2;
if k~=k2&w(1)>=6*pi/7&w(1)<=8*pi/7
    value(3)=-1;
else
    value(3)=1;
end
isterminal = [1;1;1];
direction = [0;0;0];
end
```

This function receives two input parameters, i.e. the current simulation time `t` and the vector `w` with the state variables (angle and angular velocity). The first element of the calculated vector `value` represents the tension on the rope. If this value becomes negative, the state changes from "bound" to "flying" (cf. Eq. 2). If the maximum oscillation of the pendulum is lower than $\pi/10$, `value(2)` becomes negative and the simulation is stopped. Finally, `value(3)` changes its sign when the angle `w(1)` enters the region with different damping factor, i.e. $[6\pi/7, 8\pi/7]$ in this example.

The event function of the "flying" state has the following structure:

```
function [value,isterminal,direction]=
    eventloc(t,v)
global L
value = L^2-(v(1)^2+v(2)^2);
isterminal = 1;
direction = -1;
end
```

If the output parameter `value` of this function becomes negative, the state changes from "flying" to "bound", according to Eq. 5.

The MATLAB `options` composition of the odeset command is a structure of optional parameters that change the default integration properties, for example:

```
[to,w,tE,wE,iE]=ode45(@f,t,w0,options).
```

For this example, we used `ode45` as ODE solver, but the event detection works just as well with any other MATLAB ODE solver. The return value `iE` contains the index of the event function that has vanished and can be used to determine which event has occurred [1].

Fig. 5 and Fig. 6 show the results of a simulation run of the basic model (no region with inconstant damping) with the parameters

$$m = 1.5\,\text{kg}, \qquad k = 0.9\,\text{kg/s},$$
$$l = 1\,\text{m}, \qquad g = 9.81\,\text{m/s}^2,$$

and the initial conditions

$$\varphi(0) = \varphi_0 = \pi/4,$$
$$\dot{\varphi}(0) = \dot{\varphi}_0 = 15\,\text{1/s}.$$

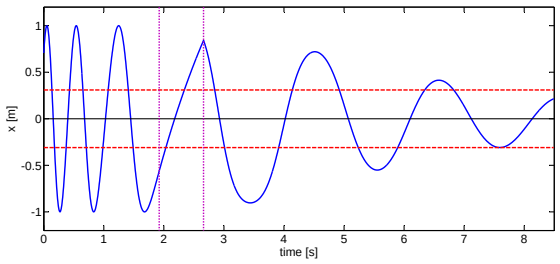

Fig. 5 Coordinate $x(t)$ during basic simulation run

In Fig. 5, the progression of the x-coordinate over the simulation time is shown. The simulation was terminated after the maximum oscillation was less than $\pi/10$. This area is represented by the horizontal dashed lines. The vertical dotted lines show the points in time when a state change occurred. Using MATLAB, these points can be identified exactly. The simulation starts in the "bound" state. At $t = 1.92\,\text{s}$, the state changes from "bound" to "flying" and after $t = 2.66\,\text{s}$, it changes back to "bound" and remains there until the simulation is terminated at $t = 8.5\,\text{s}$. Fig. 6 depicts the corresponding trajectory $(x(t), y(t))$ of the pendulum body, where you can see the circular movement during the "bound" state and, deviating from that, the movement during the "flying" state, which occurred once.

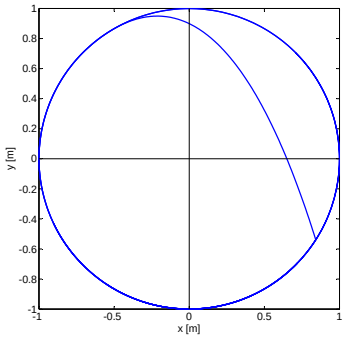

Fig. 6 Trajectory $(x(t), y(t))$ during basic simulation run

Fig. 7 and Fig. 8 present the simulation results with the modified model and a chosen damping factor of $k_2 = 12.8$ in the region $\varphi \in [6\pi/7, 8\pi/7]$.

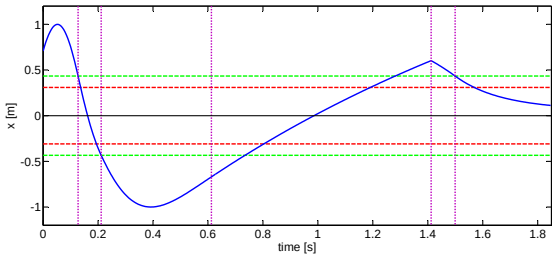

Fig. 7 Coordinate $x(t)$ during simulation run with modified model

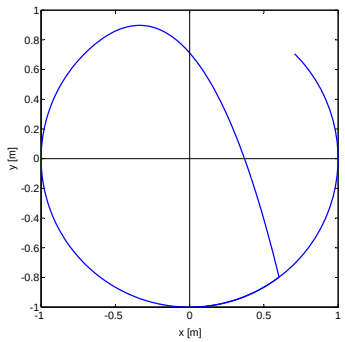

Fig. 8 Trajectory $(x(t), y(t))$ during simulation run with modified model

Like in Fig. 5, the red horizontal dashed lines in Fig. 7 are representing the area, in which the simulation is terminated (if the pendulum stays in there). The outer green lines illustrate the region with different damping factor. The points of time, when a state change occurs, are indicated by vertical dotted lines in Fig. 7. The first changes occur at $t = 0.13\,\text{s}$, when the mass is entering the area with different damping and at $t = 0.21\,\text{s}$, when it is left again. At $t = 0.61\,\text{s}$, the state is changed to

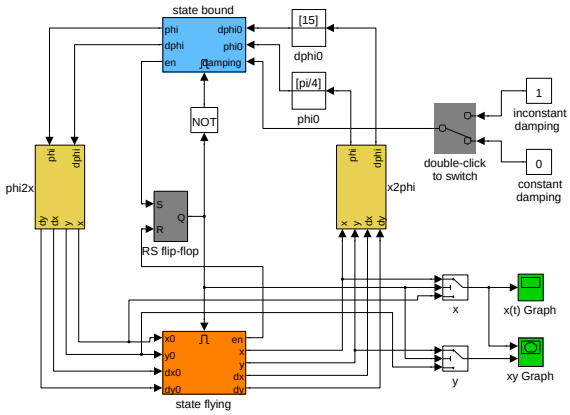

Fig. 9 SIMULINK model of the rotating pendulum

"flying" and switched back to "bound" at $t = 1.41\,\mathrm{s}$. At last, the mass is entering the region with different damping again at $t = 1.5\,\mathrm{s}$ and is staying in there until the simulation is terminated at $t = 1.85\,\mathrm{s}$. Fig. 8 shows again the corresponding trajectory. Due to the much higher damping factor in the small region, the "flying" state is already setting in before the pendulum can manage a first full rollover.

Implementation with SIMULINK

Fig. 9 shows the basic structure of the SIMULINK model. The two subsystems at the top and at the bottom contain the models for the two different states and are shown in Fig. 10 and Fig. 11. These subsystems can be enabled or disabled via a control input port, which is controlled by a RS flip-flop. A NOT gate takes care of these two ports being complementary to each other, so that exactly one of the two subsystems is enabled at each point of time [2].

The output signals of the subsystems, which represent the current position and velocity of the pendulum, are handed to the respective other state model, after a coordinate transformation (blocks `phi2x` and `x2phi`). However, the other subsystem does not read the values of its input ports until it is enabled. When it becomes active, the subsystem resets its integrators, causing them to read in the values of their initial conditions from the corresponding input ports [2].

To visualize the results, a scope and a xy graph block are connected to the signal ports. Two switches, which are controlled by the same signal as the state models, take care that the correct output values from the current enabled subsystem are used for the plot.

To detect state events, one can use SIMULINK If blocks and If Action subsystems. In the "bound" subsystem for example, the force on the rope is calculated according to Eq. 2, zero-crossing of this value is detected via an If block, and a connected If Action subsystem initiates a changeover to the "flying" subsystem by sending an impulse to the RS flip-flop, causing it to toggle its output value.

Similarly, the point of time to terminate the simulation is detected by checking the maximum oscillation, i.e. the current deflection of the body when the velocity is zero. In this example, it is terminated after the maximum oscillation does not exceed $\pi/10$ any longer.

The modification of the model to implement an inconstant damping factor is realized in the subsystem of the "bound" state. By checking the current angle `phi`, it is observed which region the pendulum is being in, and according to this information the corresponding damping factor is chosen via a switch.

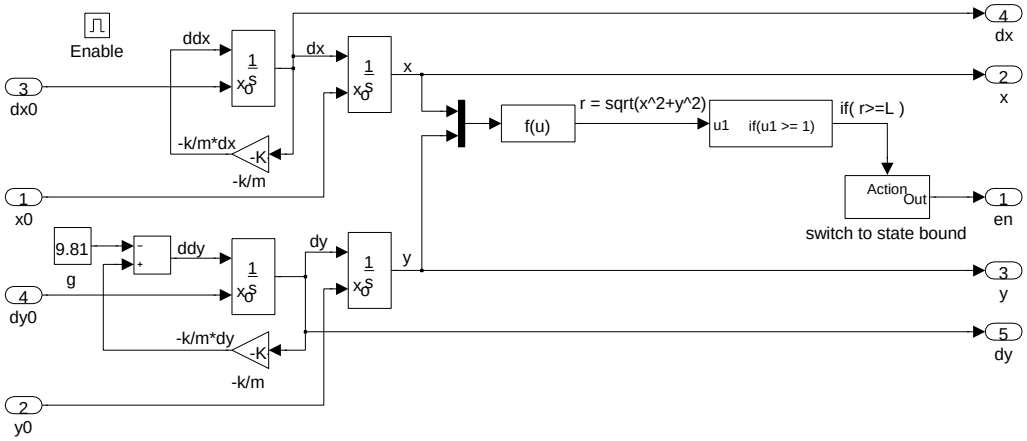

Fig. 10 Basic simulation model for the "flying" state as implemented in SIMULINK

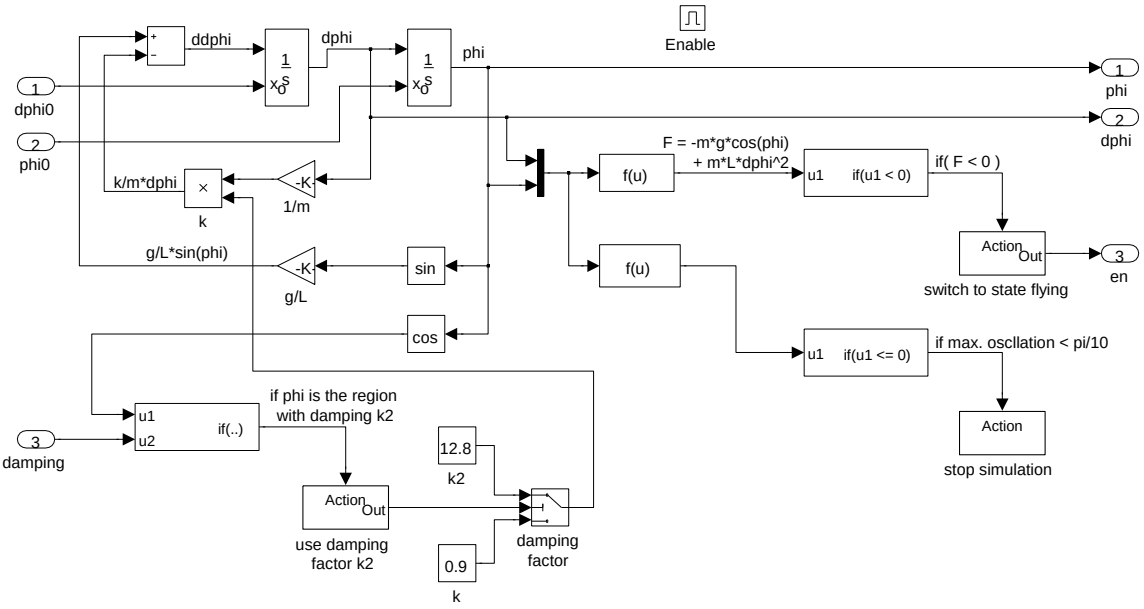

Fig. 11 Basic simulation model for the "bound" state as implemented in SIMULINK

3 Bouncing Ball

The second example deals with the model of a bouncing ball according to [3] and [4]. In this example the simulator will be tested regarding different categories of state events. The model of a bouncing ball consists of two different states: The free-falling phase and the bounce. In this section, we will discuss the possibilities how to build such a model and how to extend it.
We start with a basic model, that only takes the gravitational force in account and treats the bounce as a state event, where some energy is lost. Then we will extend the model in two different ways: First we introduce air-resistance, then we will simulate the bounce by adding a spring-damper-system.

3.1 Basic model

The motion of a free-falling mass in a gravitational field without resistance is given by the following two equations:

$$\dot{v} = -g, \tag{6}$$

$$\dot{h} = v, \tag{7}$$

where g is the acceleration of the gravitational field. The initial conditions $v(0) = v_0$ and h_0 define the velocity and the height of the ball at $t = 0$.
Until the ball hits the ground ($h = 0$), we have a totally continuous system, but at the bounce we have to treat a discontinuous change of the ball's motion.
In this first approach, we model this event by using Newtons third law and a coefficient μ, that describes the loss of energy.
The velocities v^- right before the bounce and v^+ afterwards are related as follows:

$$v^+ = -\mu \cdot v^- \tag{8}$$

Given that, we can start a new free-falling phase with new initial conditions $h_0 = 0$ and $v_0 = -\mu v^-$.

Mathematical analysis

By solving the set of ODEs shown in Eq. (6) and Eq. (7), we get

$$v(t) = -g \cdot t + v_0, \tag{9}$$

$$h(t) = -\frac{g}{2} \cdot t^2 + v_0 \cdot t + h_0, \tag{10}$$

as solutions of the initial value problem (IVP). With this equations, we can easily evaluate the time of the first bounce, assuming $v_0 = 0$

$$t_1 = \sqrt{\frac{2h_0}{g}}. \tag{11}$$

And the velocity at time t_1 is

$$v(t_1) = -\sqrt{2gh_0} \tag{12}$$

Given the relation from Eq. (8), we have now a solution for the second free-falling phase and can evaluate how long it lasts:

$$h(t - t_1) = -\frac{g}{2}(t - t_1)^2 + \mu\sqrt{2gh_0}(t - t_1), \tag{13}$$

$$t_2 - t_1 = 2\mu t_1. \tag{14}$$

Now it is obvious, that the durations of the free-falling phases are related as followed:

$$t_m - t_{m-1} = \mu \cdot (t_{m-1} - t_{m-2}) \qquad \forall m \geq 3. \tag{15}$$

From Eq. (14) and Eq. (15), we can derive a formula for the time of the m-th bounce:

$$t_m = \sqrt{\frac{2h_0}{g}} \cdot \left(-1 + 2 \cdot \sum_{i=0}^{i=m-1} \mu^i \right) \tag{16}$$

This is a geometric series and therefore it is limited if $\mu < 1$:

$$t_\infty = \sqrt{\frac{2h_0}{g}} \frac{1+\mu}{1-\mu} \tag{17}$$

Simulation Results

The simulation of the bouncing ball model is realized in MATLAB, with the ode45 solver.

Tab. 1 Parameter and initial conditions for bouncing ball model

$v_0 = 0$	$g = 9.81$
$h_0 = 10$	$\mu = 0.9$

The simulation stop time of the simulation is t_∞ from Eq. (17). The results of the simulation run using the defined parameters from Tab. 1 are shown in Fig. 12.

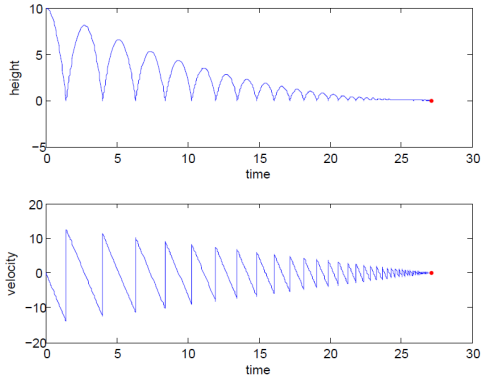

Fig. 12 Height and velocity in the basic model are decreasing over time

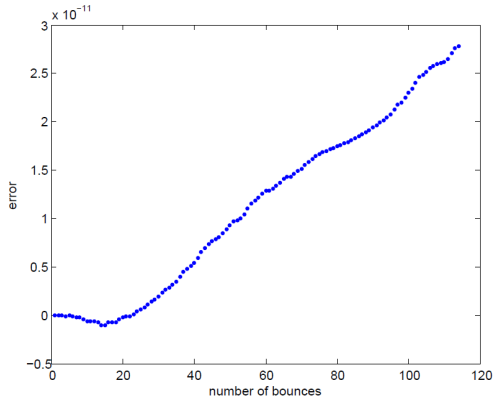

Fig. 13 Error of bounce times: simulation of analytical results

In Fig. 13, the times of the bounces from the simulation are compared to the analytically calculated times from Eq. (16) and the error is shown.

Since the error is negative at the beginning, the times of the bounces that the simulation estimates are later than the exact timepoints, as long as the ball jumps high enough. After about 15 bounces the error begins to monotonically increase, which means that the simulation underestimates the duration of the free-falling phase. The figure also shows, that the error does not exceed $3 \cdot 10^{-11}$, so the simulation is very accurate regarding this perspective.

3.2 Model with air-resistance

To extend the model in a first step, we introduce the friction-constant β to describe the behavior of the ball in an environment under natural atmospheric pressure assumptions. The new equations are

$$\dot{v} = -g - \beta \cdot v^2 \cdot \mathrm{sgn}(v), \tag{18}$$

$$\dot{h} = v. \tag{19}$$

In Eq. (18), $\mathrm{sgn}(v)$ has to be included, so that the friction force is always contrary to the direction of movement. The coefficient β for the air resistance can be calculated as

$$\beta = \frac{1}{2} \cdot C_D \cdot \rho \cdot \frac{A}{m}, \tag{20}$$

where C_D is the drag coefficient, which is 0.47 for a sphere, ρ is the density of the surrounding medium, A the cross-section and m the mass of the ball. In the simulation run we will compare the behaviour of the ball on earth and on mars. The parameters that are different on the two planets are the gravitational force constant and the density of the atmosphere.

Tab. 2 Gravitational force constant and the density of the atmosphere

Earth	$g_e = 9.81$	$\rho_e = 1.227$
Mars	$g_m = 3.693$	$\rho_m = 0.015$

Note, that these constants are only true for low altitudes. We can now derive the friction constants β_e and β_m from the densities of the atmospheres in Tab. 2 for a ball with cross-section $A = 0.02m^2$ and mass $m = 0.3kg$:

$$\beta_e = 0.02 \tag{21}$$

$$\beta_m = 2.3 \cdot 10^{-4} \tag{22}$$

As in the basic model, the condition for the state-event is $h = 0$. The initial values for the new free-falling phase are again $h_0 = 0$ and $v_0 = -\mu v^-$ as described in Eq. (8) and below.

Simulation Results

We now simulate the bouncing ball model with air-resistance as described in Eq. (18) and Eq. (19) with the set of parameters $v_0 = 0$, $h_0 = 10$, $\mu = 0.9$ and the values for the gravitational constant g and the air resistance β (Tab. 2) for both, Earth and Mars.

The graphs in Fig. 14 show the height and the velocity of the ball on Earth and on Mars. We can observe

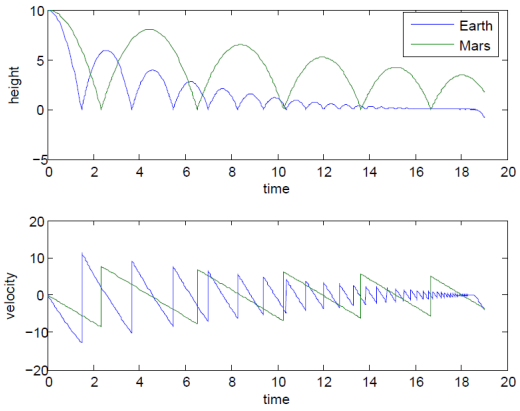

Fig. 14 Height and velocity of the model with air-resistance

that the curve, describing the velocity is not linear, like the one in the basic model, but a little bit bent, which obviously comes from $-\beta v^2 \operatorname{sqn}(v)$. It can also be seen, that the maximum velocity of the ball is higher on Earth, but the maximum height is lower. This is caused by the much more dense atmosphere on Earth than compared to the on Mars and so the friction parameter β is much higher on Earth.

3.3 Model with spring-damper-system

Now we extend the basic model from section 3.1 in another way, namely by introducing a spring-damper-system for the bouncing phase. Previously, the bounce just changed the set of the initial conditions for the model of the free-falling phase, which consumed no time. Now it is implemented as an additional state, so we have two states the model can be in, namely the free-falling state and the bouncing state. As we can see in

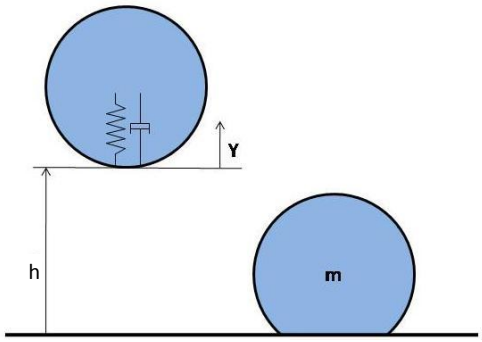

Fig. 15 Sketch of a ball using a spring-damper-system

Fig. 15, we have to introduce a new state-equation for the deformation y of the ball. That brings us to the system of ODEs for the free-falling state

$$\dot{v} = -g,\ \dot{h} = v,\ \dot{y} = -\frac{k}{d} \cdot y, \tag{23}$$

with a spring constant k and a damping constant d. For the bouncing state we have the equations

$$\dot{v} = -g + f_c,\ \dot{h} = v,\ \dot{y} = -\dot{h}, \tag{24}$$

where $f_c = \max((-kh - d\dot{h}), 0)$ is the normalized contact force between the ball and the ground. The contact force always pointing in the positive direction of h and therefore it cannot be negative.
Similar to the first two models, the model has to switch from the free-falling state to the bouncing state, if the condition $x + y = 0$ is met. The condition for switching back to the free-falling state is $f_c = 0$, that means that there is no more contact between the ball and the ground.

Simulation results

For the simulation of the bouncing ball model with a spring-damper-system, we use the set of parameters in Tab. 3.

Tab. 3 Parameter for spring-damper-system

$g = 9.81$	$h_0 = 10$	$k = 10^6$
$v_0 = 0$	$d = 500$	$y_0 = 0$

In Fig. 16 we can see the height and the velocity of the ball and Fig. 17 shows the deformation of the ball.

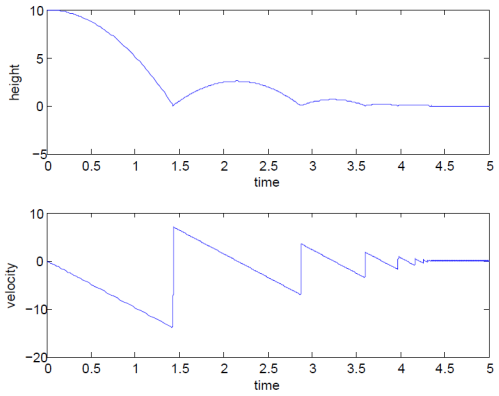

Fig. 16 Height and velocity results for the extended model

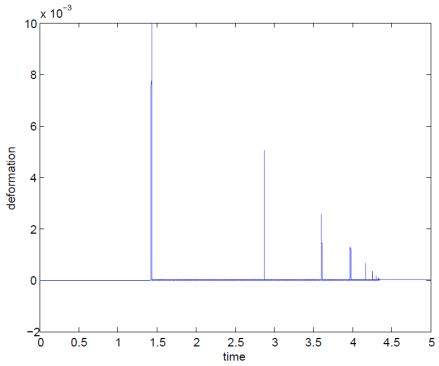

Fig. 17 Moment and quantity of deformation of the ball

This modelling strategy causes, that the bouncing ball is stable with reference to fall trough the ground. Moreover, the model is much more related to the reality because the contact with the ground is conected to time.

4 Hybrid Electrical Circuit

Based on the problem definition of a class E-amplifier in [5], also used in [6], a new electrical circuit is designed. Simulation experiments and modelling tasks are defined on the following system.

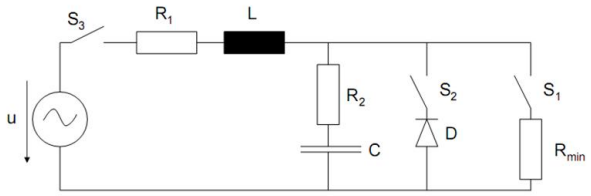

Fig. 18 Schematic of the switched electrical test circuit

In Fig. 18, a schematic of the system of interest is shown. Assume $R_1 = R_2 = 1\text{k}\Omega$, $f = 1\text{MHz}$ and $R_{\text{min}} = 1\text{m}\Omega$. The definition of this simulation project is given in three items with several sub points.
In some tasks, the switch S_1 is replaced by a time-dependent resistor $R_{S_1}(t)$ with a time response depicted in Fig. 19.

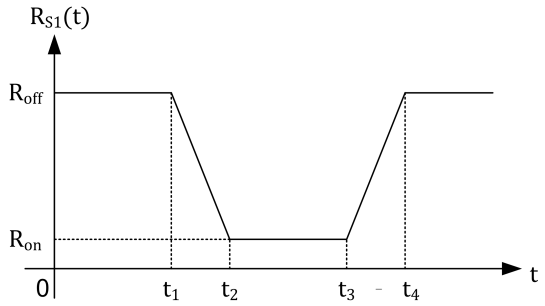

Fig. 19 Time-dependent resistor for switch S_1

Assume $R_{\text{off}} = 10^8\Omega$, $R_{\text{on}} = 10^{-4}\Omega$, $t_1 = 2 \cdot 10^{-7}\text{s}$, $t_2 = 2,5 \cdot 10^{-7}\text{s}$, $t_3 = 4,5 \cdot 10^{-7}\text{s}$, and $t_4 = 5 \cdot 10^{-7}\text{s}$, where t_i, $i = 1, \ldots, 4$ are timevalues, measured in seconds.

The diode in the electrical circuit will be implement using three different models.

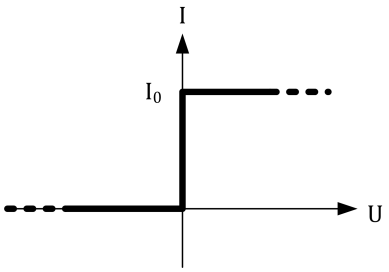

Fig. 20 Diode characteristic: ideal switch

The first model, the simplest one, defines the diode as an ideal switch with current restriction. The two states are defined as followed:

- For a negative input voltage ($U < 0$) the current through the diode is 0.
- For a positive input voltage ($U > 0$) the diode is assumed to be an ideal conductor, but for simulation the current is limited to $I_0 = 10^6\text{A}$.

The characteristic curve of this simplest model is shown in Fig. 20.

The second model differs from the first in the way that a threshold U_D of the input voltage is scheduled, shown in Fig. 21. The two cases from the first model are determined by $U < U_D$ and $U > U_D$, with a threshold $U_D = 0,7\text{V}$.

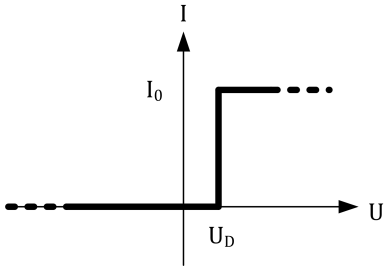

Fig. 21 Diode characteristic: switch with threshold

The last model is defined to design the current through the diode via the Shockley diode equation, which is given by

$$I(U) = \begin{cases} I_S \cdot \left(e^{\frac{U}{U_T}} - 1\right), & U \geq 0 \\ 0, & U < 0 \end{cases}, \qquad (25)$$

with the saturation current $I_S = 10^{-12}\text{A}$ and the temperature voltage $U_T = 25\text{mV}$, by assumption of constant ambient temperature. The characteristic curve of the Shockley diode equation is drafted in Fig. 22.

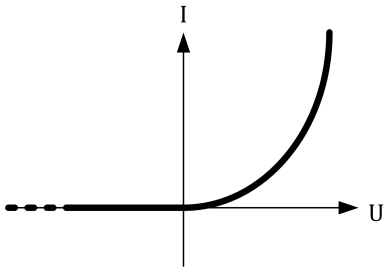

Fig. 22 Diode characteristic: sketch of Shockley diode equation

To calculate the adequate inductance L and capacity C the formulas

$$q = \frac{2\pi f L}{R_1 + R_2} \quad \text{and} \quad f = \frac{1}{2\sqrt{LC}}, \qquad (26)$$

for a given quality factor $q = 40$ are used.

In the following sections some assignments will be solved with different diode models and in some tasks with the time-dependent resistor $R_{s_1}(t)$. The different assignments of tasks will show different behaviour of the defined test circuit for different models and switch combinations.

4.1 Calculation of the eigenvalues of the system

In this section the switch S_3 is assumed to be closed. The state space model of the electrical circuit is represented by

$$C\dot{x} + Ax = Bu, \tag{27}$$

where $C, A \in \mathbb{R}^{2\times 2}$ and $u, x \in \mathbb{R}^2$. If C is invertible, the equation can be expressed as

$$\dot{x} = -C^{-1}Ax + C^{-1}Bu. \tag{28}$$

The eigenvalues $\lambda_1, \lambda_2 \in \mathbb{C}$ of the system are the eigenvalues of the matrix $C^{-1}A$.

The eigenvalues of the electrical circuit are calculated in three different settings:

1. S_1 and S_2 are assumed to be open, the resulting curcuit is a standard oscillating curcuit in serial connection.
2. S_2 is assumed to be open and S_1 is replaced by the time-dependent resistor depicted in Fig. 19. For the different sections of the characteristic curve of the time-dependent resistor, the curcuit is a piecewise linear model.
3. S_1 is assumed to be open and S_2 is assumed to be closed. The diode model has to be linearized in a operating point (U_d, I_d), so the linearized equivalent resistance is $R_d = \frac{U_d}{I_d} = 20\Omega$.

Tab. 4 Eigenvalues of the system for different scenarios

Setting	Section	$(\lambda_1, \lambda_2)^T$
1		$\begin{pmatrix} -0,0785 + i \cdot 6,2827 \\ -0,0785 - i \cdot 6,2827 \end{pmatrix} \cdot 10^6$
2	$[0, t_1]$	$\begin{pmatrix} -0,0811 + i \cdot 6,2827 \\ -0,0811 - i \cdot 6,2827 \end{pmatrix} \cdot 10^6$
	$[t_1, t_2]$	$\begin{pmatrix} -0,0817 + i \cdot 6,2827 \\ -0,0817 - i \cdot 6,2827 \end{pmatrix} \cdot 10^6$
	$[t_2, t_3]$	$\begin{pmatrix} -0,0008 \\ -5,0265 \end{pmatrix} \cdot 10^8$
	$[t_3, t_4]$	$\begin{pmatrix} -0,0911 + i \cdot 6,2825 \\ -0,0911 - i \cdot 6,2825 \end{pmatrix} \cdot 10^6$
3		$\begin{pmatrix} -0,0008 \\ -4,9280 \end{pmatrix} \cdot 10^8$

The results of the eigenvalues for the three different settings are shown in Tab. 4. In setting two, the linear rising and falling of the time-dependent resistor is implemented via a linear equation, which is sampled with a sampling rate of $t_s = 10^{-8}$. The eigenvalues differ marginal, so only the first eigenvalue is shown in the table as a representative. In all settings, the system is stable because of $\mathrm{Re}(\lambda_i) < 0, i = 1, 2$.

4.2 Dynamics depending on the time-dependent resistor

In this section, the switch S_3 is assumed to be closed and S_2 is assumed to be open. So the simulation is only depending on the state of the switch S_1 which is given by the time-dependent resistor (Fig. 19). A quantity of the linear slope is the slew rate, which is defined by

$$\Delta t = t_2 - t_1 = t_4 - t_3. \tag{29}$$

In this task the slew rate is varied, so that the characteristic curve of the time-dependent resistor is discontinuous or the slope is parallel to a bisectrix, i.e. the corresponding angle is $\frac{\pi}{2}$ and $\frac{\pi}{4}$. The input voltage is assumed as

$$u(t) = U_0 \cdot \sin(2\pi f t), \tag{30}$$

with $U_0 = 15\mathrm{V}$ and $f = 1\mathrm{MHz}$. For both slew rates, the output current from the source through R_1 and L is depicted in Fig. 23.

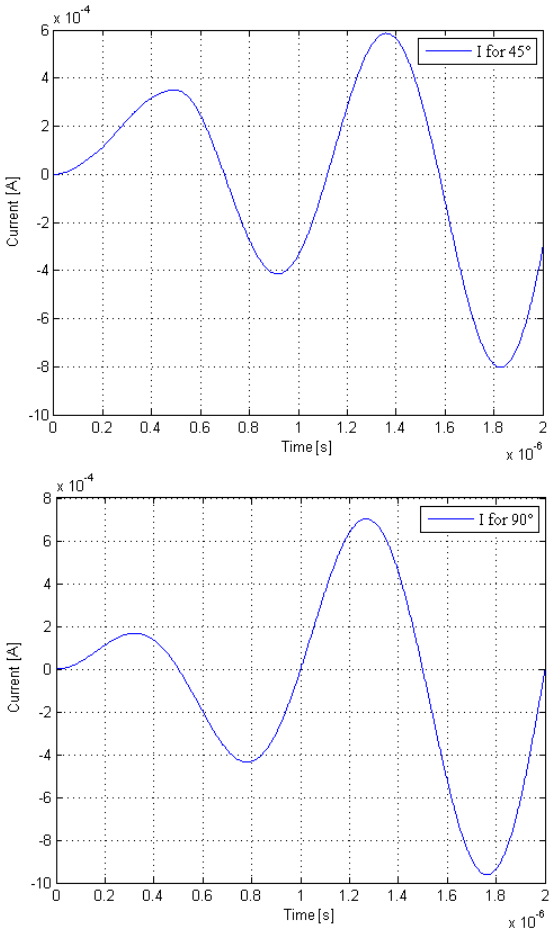

Fig. 23 Characteristics of total current through the curcuit for slew rate of $\pi/4$ and $\pi/2$

4.3 Comparison of the different diode models

Assume S_2 and S_3 to be closed and S_1 to be open. Each of the three diode models is used to simulate the electrical circuit. For each simulation, the simulation time

is measured. The characteristic curve of the current of the source is shown in Fig. 24.

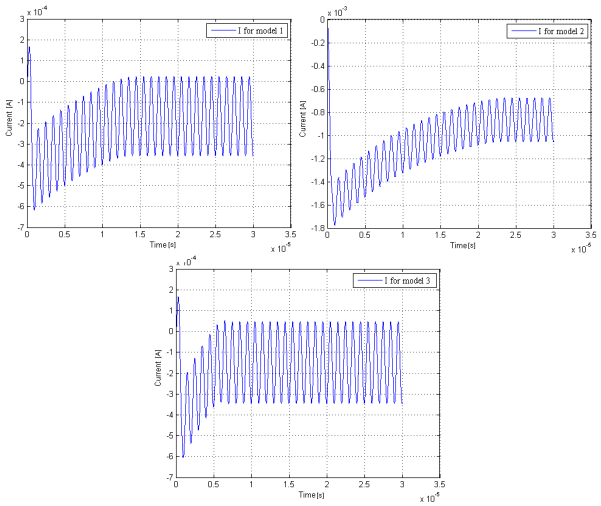

Fig. 24 Total current characteristics for diode model as an ideal switch (left top), as a switch with threshold U_D (right top) and the diode model via the Shockley equation (bottom)

The consumed time for simulation with each diode model is listed in Tab. 5.

Tab. 5 Simulation time for each diode model

diode model	simulation time [s]
1	154.608431
2	91.826230
3	131.766537

4.4 Hybrid scenario

In this subsection S_2 and S_3 are assumed to be closed and S_1 replaced by the time-dependent resistor $R_{S_1}(t)$. The diode is modeled as a switch with threshold (Fig. 21) and let $R_{\text{min}} = 0$, see Fig. 18.

The voltage on the time-dependent resistor is proportional to the resistance value. This voltage is the control parameter of the diode, control in the sense that this parameter decides the state the diode. This relation is shown in the voltage curve depicted in Fig. 25

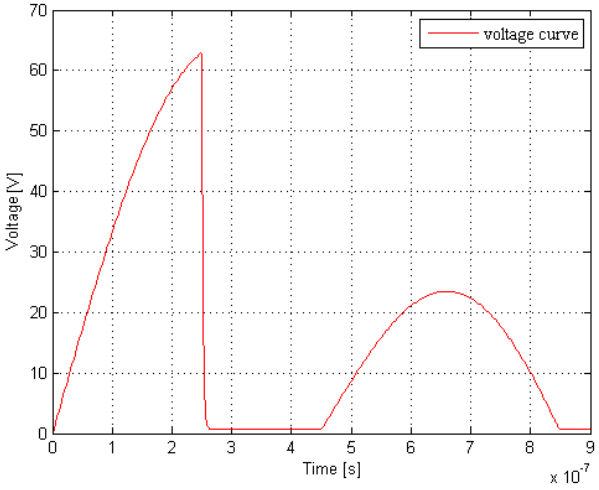

Fig. 25 Voltage characteristics from the diode in hybrid scenario

5 Conclusion

The examples in this paper show that hybrid models can consist of different states and that each of these states has to be described by different physical equations. The degree of freedom in each of these states can differ. In order to determine when the state has to be changed, it is necessary to evaluate the time of the state event as exact as possible. Therefore, simulation of hybrid models requires simulation software with event handling functionality. Through the experiments described in this paper it has been shown, that MATLAB and SIMULINK are suitable tools for hybrid modelling. The different ways of solving simulation tasks shown in this paper depict the flexibility of these systems.

6 References

[1] Mathworks. *MATLAB 7 - Mathematics*. The Mathworks Inc., March 2010.

[2] Mathworks. *SIMULINK 7 - User's Guide*. The Mathworks Inc., March 2010.

[3] H. Ecker. The bouncing ball problem - modelling and simulation aspects. *SNE Simulation News Europe*, 34:9 – 14, 2002.

[4] P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. John Wiley and Sons, 2004.

[5] N.O. Sokal and A.D. Sokal. Class e - a new class of high-efficiency tuned single-ended switching power amplifiers. *IEEE Journal of Solid-State Circuits*, SC-10:168–176, 1975.

[6] N. Viertl and F. Breitenecker. Comparison 3 - analysis of a generalized class-e amplifier. *SNE Simulation News Europe*, 27:40, 1999.