State Events and Structural-dynamic Systems: Definition of ARGESIM Benchmark C21

Andreas Körner*, Felix Breitenecker

Mathematical Modelling and Simulation Group, Inst. of Analysis and Scientific Computing,
Vienna University of Technology, Wiedner Haupstrasse 8-10, 1040 Vienna, Austria; *andreas.koerner@tuwien.ac.at*

Simulation Notes Europe SNE 26(2), 2016, 117 - 122
DOI: 10.11128/sne.26.bn21.10339
Received: March 30, 2016; Revised: May 20, 2016;
Accepted: May 25, 2016;

Abstract. Modelling and simulation of state events and of structural-dynamic systems is getting more and more important in advanced modelling theory and application. Therefore, the requirements regarding flexibility on modelling and on implementation in simulators is increasing.
To investigate, how modelling approaches and simulation environments deal with state events and structural-dynamic systems, the new ARGESIM Benchmark C21 'State Events and Structural-dynamic Systems' is defined. Three case studies should compare modelling and implementation of state events in dynamic systems, up to structural-dynamic systems governed by state events. The first case study, the almost classical bouncing ball dynamics investigates different kinds of bounce modelling and implementation with associated events. In the second case study, Switching RLC Circuit, different diode models result in simple switching state events or in in DAE systems. The third case study is structural-dynamic by itself: the rotating pendulum with free falling phase changes dynamics from swinging to falling (an vice versa) – switching between different degrees of freedom. These three case studies invite simulationists for providing 'solutions' – reports on modelling, implementation and specific investigations by of suggested experiments with the implemented model, to be published in SNE Simulation Notes Europe.

Development - Background

In, 1990, ARGESIM started in the journal SNE *Simulation News Europe* the comparison series *Comparison of Simulation Software* in order to compare features of simulators for classic system simulation. Since that, system simulation has developed further on (physical modelling, structural-dynamic systems, state charts, ...), and consequently also the comparisons developed further on towards *Benchmarks for Modelling Approaches and Simulation Implementations*.

Development of System Simulation

The classic explicit state space modelling has partly been replaced by 'higher' modelling techniques. Mainly the Modelica standard and the competitive VHDL-AMS standard have introduced physical modelling – component-based modelling with a-causal relations. Thereby, the components may be part of textual or graphical libraries in various domains [1], Figure 1. From mathematics' viewpoint, instead of explicit state models now implicit 'law-oriented' model descriptions have become basis for subsequent simulation, resulting in implicit differential-algebraic systems (DAEs).

In principle, the simulator now must translate the a-causal model description into a DAE system with proper structure of differential and algebraic equations, so that a 'modern' DAE solver can handle the implicit state space model with sufficient accuracy and sufficient convergence (index reduction problem).

Furthermore, more and more discrete elements were used in system simulation – not only sampled data, but also conditions and structural changes - so that also modelling techniques for discrete dynamic structures have become necessary – e.g. state charts with discrete and continuous dynamics.

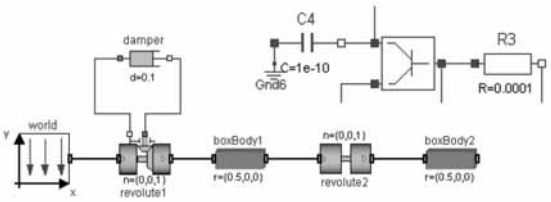


Figure 1: Modelica physical modelling for analog electrical domain and mechanics multibody domain.

Development of Comparisons / Benchmarks

ARGESIM / EUROSIM started in 1990 the series Comparison of Simulation Software in the journal Simulation News Europe (SNE). These comparisons are based on relatively simple, easily comprehensible processes.

In the beginning, simulationists were invited to prepare a 'solution' and to publish in SNE (1-page solution). Along with development of system simulation, also the comparisons developed further on. This development can be seen in definitions and solutions published from 1990 to 2016 in 87 SNE issues: 23 definitions (some revised), and about 350 comparison 'solutions'. The following list of comparisons and benchmarks shows also the broad variety of the applications (including this new benchmark):

- C1 Lithium-Cluster Dynamics, SNE 0(1), 1990
- C2 Flexible Assembly System, SNE 1(1), 1991
- C3 Generalized Class-E Amplifier, SNE 1(2), 1991
- C4 Dining Philosophers I, SNE 1(3), 1991
- C5 Two State Model, SNE 2(1), 1992
- C6 Emergency Department SNE 2(3), 1992
- C7 Constrained Pendulum, SNE 3(1), 1993
- CP1 Parallel Simulation Techniques, SNE 4(1), 1994
- C8 Canal-and-Lock System, SNE 6(1), 1996
- C9 Fuzzy Control of a Two Tank System, SNE 6(2), 1996, revised SNE 16(3), 2006
- C10 Dining Philosophers II, SNE 6(3), 1996
- C11 SCARA Robot, SNE 8(1), 1998
- C12 Collision of Spheres, SNE 9(3), 1999
- C13 Crane Crab and Embedded Control, SNE 11(1), 2001; rev. SNE 17(1), 2007
- C14 Supply Chain, SNE 11(2-3), 2001
- C15 Clearance Identification, SNE 12(2-3), 2002
- C16 Restaurant Business Dynamics, SNE 14(1), 2004
- C17 Spatial Dynamics of SIR Epidemics, SNE 14(2-3), 2004; revised SNE 25(2), (2015)
- C18 Neural Networks vs. Transfer Functions, SNE 15(1), 2005
- C19 Pollution in Groundwater Flow, SNE 15(2-3), 2005, revised SNE 16(3-4), 2006
- CP2 Parallel & Distributed Simulation, SNE 16(2), 2006
- C20 Complex Assembly System, SNE 21(3-4), 2011
- C21 State Events and Structural-dynamic Systems, SNE 26(2), 2016

In 2006, a re-organisation of the comparisons has been started [2]. The comparisons developed towards *Benchmarks for Modelling Approaches and Simulation Implementations:*

- Revised definitions: SNE is publishing revised definitions of previous comparisons, updating models and tasks in order to continue them as benchmark.
- Extended solution documentation: SNE allows two (or more) pages for solutions of classic benchmarks.
- Extended Benchmarks: SNE introduces extended benchmarks, comparing modelling and simulation paradigms, or dealing with more complex models and experiments – as with benchmarks C19, CP2, C20 and C21.

  Documentation and publication in SNE of 'solutions' may take more pages – up to 10 pages SNE.

1 State Events and Structural – dynamic Systems

This section reviews some necessary mathematical background and modelling notations, in order to allow a better and comparable documentation of the investigations in this new benchmark C21.

1.1 DAE systems and state events

Mainly because of physical modelling techniques like sketched in Figure 1, the classical ODE state space description

$$\dot{\vec{x}}(t) = \dot{f}(t, \vec{x}(t), \vec{u}(t)), \qquad \vec{x}(t_0) = x_0, \qquad (1)$$

with $\vec{x}(t)$ state vector, $\vec{f}(t)$ derivative vector, $\vec{u}(t)$ input vector, $\vec{x}_0$ initial state vector and $\vec{p}$ parameter vector, was replaced by the (semi-) implicit state space description of DAE system type

$$\dot{\vec{x}}(t) = \vec{f}(t, \vec{x}(t), \vec{z}(t), \vec{u}(t), \vec{p}), \qquad \vec{x}(t_0) = x_0$$
$$\vec{g}(\vec{x}(t), \vec{z}(t), \vec{u}(t), \vec{p}) = \vec{0} \qquad (2)$$

The algebraic equations, e.g. constraints, are coming along with another new challenge, with structural dynamic systems. Constraints are very often coupled with state-dependent conditions for their validity – like loss of freedom, etc., requiring a conditional change of the model description.

Consequently the problem of state event description and state event handling becomes much more complex than in classic ODE models and raises new questions for proper model description.

Although mathematically incorrect, in models from application it is often necessary to model discontinuities in the model description, because a certain system phenomenon can only be described approximatively by a (discontinuous) change in the model,

These discontinuous changes are called *events*; if the time instant of the change is known in advance, the event is called a *time event*; if the event depends on a certain value or threshold for a state variable (which is not known in advance), it is called *state event*.

A state event is defined

- by an event function $h(\vec{x}(t), \vec{z}(t), \vec{u}(t), \vec{p})$, whose zero determines the time instant $\hat{t}$ of the next occurrence of the event,
- and by an event action $E(\vec{x}(\hat{t}), \vec{z}(\hat{t}), \vec{u}(\hat{t}), \vec{p})$, which performs the discontinuous change.

An event function h can cause the associated event action E several times, and there may be more than one event scheduled by an event function:

$$\begin{aligned} h^B(\vec{x}(t), \vec{u}(t), \vec{p}) \overset{!\pm}{=} \vec{0} &\overset{\hat{t}^B}{\Rightarrow} E^B(\vec{x}(\hat{t}^B), \vec{u}(\hat{t}^B), \vec{p}) \\ \cdots & \\ h^Z(\vec{x}(t), \vec{u}(t), \vec{p}) \overset{!\pm}{=} \vec{0} &\overset{\hat{t}}{\Rightarrow} E^Z(\vec{x}(\hat{t}^Z), \vec{u}(\hat{t}^Z), \vec{p}) \end{aligned} \quad (3)$$

The symbol $\left(\overset{!\pm}{=}\right)$ in equation (3) means, that the zero $\hat{t}^B$ of the event function $h^B(\vec{x}(t), \vec{z}(t), \vec{u}(t), \vec{p})$ responsible for the event 'B', is to be determined ('!'), whereby crossings in both direction cause the event ('±'), or only crossings in negative direction ('-'), or only crossings in positive direction ('+'). Event functions can be seen as classical output functions, but sometimes they are only locally defined, or sometimes the algebraic function becomes an event function and vice versa.

The associated event action $E^B(\vec{x}(\hat{t}), \vec{u}(\hat{t}), \vec{p})$ must now handle the discontinuous change in the model description, which ranges from simple to very complex. It makes sense to classify events with respect to their 'quality' of action of change [3], [4]:

- Parameter change event – SE-P
- Input change event – SE-I
- State change event – SE-X
- Function change event – SE-F
- Structure change event – SE-S
- Output trace event – SE-O
- Algorithm event – SE-A

A simple state event is a *parameter change event* SE-P. One or more parameters p_m of the parameter vector $\vec{p}$ change to a new value:

$$\text{SE-P:} \quad E^P(\vec{x}(\hat{t}), \vec{u}(\hat{t}), \vec{p}): \quad p_{m,prev} \overset{\hat{t}}{\rightarrow} p_{m,new} \quad (4)$$

The *input change event* SE-I is not a state event, it is 'only' a time event – in order to synchronize discontinuities in input $\vec{u}(\hat{t})$ with the stepsize of the DAE solver.

A *state change event* –SE-X- is a strange construct – one or more components of the differential state vector change discontinuously. From viewpoint of mathematics, that cannot happen, because the state vector $\vec{x}(t)$ results as 'integration' of the continuous derivative function. But usually events of this type itself model a dynamic behaviour, which for simplicity or other reasons is 'concentrated' into a timeless event.

$$\text{SE-X:} \quad E^X(\vec{x}(\hat{t}), \vec{u}(\hat{t}), \vec{p}): \quad x_{n,prev}(\hat{t}) \overset{\hat{t}}{\rightarrow} x_{n,new}(\hat{t}) \quad (5)$$

A *function change event* SE-F changes at event time components of the derivative function or of the algebraic function, not only with a jump in values, but with a new description:

$$\text{SE-F:} \quad E^F(\vec{x}(\hat{t}), \vec{u}(\hat{t}), \vec{p}): \quad f_{(prev)} \overset{\hat{t}}{\rightarrow} \breve{f}_{(new)}, g_{(prev)} \overset{\hat{t}}{\rightarrow} \breve{g}_{(new)} \quad (6)$$

State events of type SE-F and SE-X are 'simple' cases of structural model changes. The *structure change event* SE-S is the most complex one: in case of the event, another model is to be used; this new model may have a state space $\vec{x}^S(t), \vec{z}^S(t)$ of different dimension and type:

$$\begin{aligned} &\text{SE-S:} \quad E^S(\vec{x}(\hat{t}), \vec{u}(\hat{t}), \vec{p}): \quad (7) \\ &\vec{x}\,(t) \overset{\hat{t}}{\rightarrow} \vec{x}^S(t), \qquad \vec{z}\,(t) \overset{\hat{t}}{\rightarrow} \vec{z}^S(t) \\ &\dot{\vec{x}} = \vec{f}(\vec{x}, \vec{z}, \vec{u}, \vec{p}) \quad \overset{\hat{t}}{\rightarrow} \dot{\vec{x}}^S = \vec{f}^S(\vec{x}^S, \vec{z}^S, \vec{u}^S, \vec{p}^S) \\ &0 = \vec{g}(\vec{x}, \vec{z}, \vec{u}, \vec{p}) \quad \overset{\hat{t}}{\rightarrow} 0 = \vec{g}^S(\vec{x}^S, \vec{z}^S, \vec{u}^S, \vec{p}^S) \end{aligned}$$

State events of type SE-D and SE-S are strongly related to structural-dynamic systems, which require dynamic change of state vector dimensions, e.g. cause by loss or addition of degrees of freedom.

A simple state event is the *output trace event* SE-O. At event time, a certain value given by an output function $g_{out}(t)$ is to be traced:

$$\vec{y}(t) = \vec{g}_{out}(t, \vec{x}(t), \vec{z}(t), \vec{u}(t), \vec{p})$$

$$\text{SE-O:} \quad E^O(\vec{x}(\hat{t}), \vec{u}(\hat{t}), \vec{p}): \quad \overset{\hat{t}}{\rightarrow} g_{out}(\hat{t}) \quad (8)$$

For completeness, the *algorithm event* SE-A is mentioned. Although model description and implementation should be independent, it can happen, that under some circumstances (mainly because of problems with accuracy) the algorithmic calculations must be 'influenced' – e.g. by changing ODE solver parameters:

$$\text{SE-A:} \quad E^A(\vec{x}(\hat{t}), \vec{u}(\hat{t}), \vec{p}): \quad \overset{\hat{t}}{\rightarrow} \begin{array}{l}\text{appropriate action}\\ \text{influencing the}\\ \text{algorithm}\end{array} \quad (9)$$

In general, an event function $h^{B,C,D,\ldots}$ can cause more than one event, so that an event 'E' can belong to more than one type.

1.2 State Event Handling

The primary task for event handling are the synchronisation of the state event with the ODE/DAE solver, and the 'execution' of the event – i.e. the discontinuous change of parameters, inputs, and states, and the choice of new derivatives or new models.
State event algorithm requires the following steps:

- *Detection* of the event
- *Localisation* of event and solver stopping
- Event *Action*
- *Restart* of solver

Event detection is usually done by observing the algebraic sign of the event function during the time advance of the ODE/DAE solver. Localisation is usually superimposed to the DAE solver, by using an appropriate algorithm (iterative methods, interpolative methods). Iterative methods can give more accurate results. But in case of event functions with nearby roots, iteration may cause a deadlock, may let events vanish, etc. State event functions can be given by a state value itself – here the event localisation could be integrated into the DAE solver – but only few DAE solver implementations make use of this possibility.

1.3 Structural-dynamic systems

Systems with state events of essential types SE-F (5) or SE-S (6), often come together with a change of the dimension of the state space, then called *Structural-dynamic Systems*. In principle, for modelling structural-dynamic systems two approaches are meaningful:

- *maximal state space*: in a maximal state space, state events switch on and off algebraic conditions, which freeze certain states for certain phases, and state events 'act' within in the model (Figure 2, at left).
  hybrid decomposition: a global discrete state space controls local models with fixed state spaces, or with newly composed state spaces; state events 'schedule' different models (state chart in Figure 2, at right).

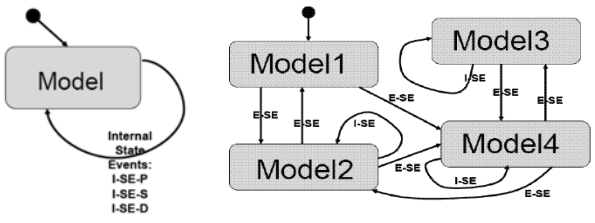


Figure 2: State chart model for -
maximal state space approach (at left)
and hybrid decomposition approach (at right)

The hybrid decomposition approach must be supported by a framework, which allows the switching between different models, and within one model – caused by state events – convenient are sate charts.

2 Case Study Bouncing Ball

When observing a bouncing ball, the ball is falling and jumping quite high, but bit by bit, position amplitude is decreasing, and bounce frequency is increasing. This physical process is well known as bouncing ball dynamics, met also in other applications. Figure 3 shows three classical examples – bouncing balls of different sizes, and a non-classical example, the pogo stick.

Figure 3: Various bouncing ball dynamics – football, ping pong, pogo stick

The bouncing ball dynamics allow various modelling approaches and incorporate events, where the dynamics change or the description of the dynamics must change.

2.1 Bouncing Ball Model - Event Contact

Following e.g. [6] the bouncing ball dynamics consist of two different phases, the free falling phase with or without air resistance, and a 'timeless' contact phase, where the bouncing ball hits the ground, and changes direction of movement (Figure 4).

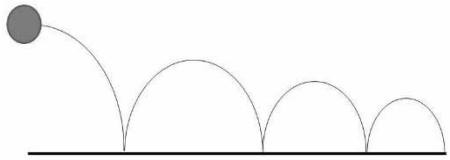


Figure 4: Idealized bouncing ball dynamics, sketch.

Free falling phase. The motion of a free falling mass in a gravitational field is given by the following two differential equations for position x and velocity v:

$$\dot{x} = v, \quad \dot{v} = -g - \beta v^2 \text{sign}(v) \tag{10}$$

with g acceleration of gravity, β air resistance coefficient, and state space $\vec{x}(t) = (x(t), v(t))^T$.
Neglecting air resistance gives the simple linear model

$$\dot{x} = v\,, \quad \dot{v} = -g \tag{11}$$

Event contact model. The *contact phase* can be implemented using different models. The *event contact model* is a quite simple approach, using Newton's 3rd law, and a coefficient μ to describe the loss of energy in case of a 'timeless' bounce, neglecting any deformation, - modelled the event '*B' Bounce.*

The velocity $v_{prev}(\hat{t})$ of the ball right 'before' the contact (impact) with the ground 'jumps' to the velocity $v_{new}(\hat{t})$ by means of 'reflection' and energy loss, being a *state change event* SE-X (5) with event action E^B:

$$v_{new}(\hat{t}) \overset{\hat{t}}{\rightarrow} -\mu \cdot v_{prev}(\hat{t}). \qquad (12)$$

The event function (13) for event action E^B (12) is simply the position x: the (bottom) position of the ball reaches ground, crossing zero into negative direction:

$$h^B(x,v) \overset{!-}{=} x \qquad (13)$$

Mathematical analysis. The linear model (11) allows analytical calculation of the impact time instants $t_{\mathrm{B,m}}$. The linear model has a solution of type

$$x(t) = -\frac{g}{2}t^2 + b\,t + c,\ v(t) = -gt + b \qquad (14)$$

With initial values $x_0 > 0$ and $v_0 = 0$, the first impact can be calculated using (14) as $t_{\mathrm{B,1}} = \sqrt{2x_0/g}$.
Starting flight at $t_{\mathrm{B,1}}$ with $x(t_{B,1}) = 0$, and

$$v(t_{B,1})_{new} = -\mu \cdot v(t_{B,1})_{prev} = \mu \cdot g \cdot t_{B,1},$$

gives the next impact time $t_{B,2} = t_{B,1}(2\mu + 1)$. Continuing with the analytical solution (14) derives a recursion for impact time: $t_{B,m} = t_{B,m-1}(\mu + 1) - \mu t_{B,m-2}$.
This recursion allows calculating the time instant t_{m} of the m-th bounce by means of the geometric series

$$t_{B,m} = \sqrt{\frac{2x_0}{g}} \cdot \left(-1 + 2\sum_{i=0}^{m-1} \mu^i\right) \qquad (15)$$

As $|\mu| < 1$, the above series (15) converges and gives the limit for the series of bouncing time instants t_∞:

$$t_{\mathrm{B},\infty} = \sqrt{\frac{2x_0}{g}} \cdot \frac{1+\mu}{1-\mu} \qquad (16)$$

- being a finite number! These considerations proof, that in finite time infinite many bounces take place.

2.2 Bouncing Ball Model - Dynamic Contact

The model (10) or (11) with *event contact phase* works considerably good in case of contact with very little deformation and very short (neglectable) contact time, i.e. for a rather stiff bouncing on a rigid surface.

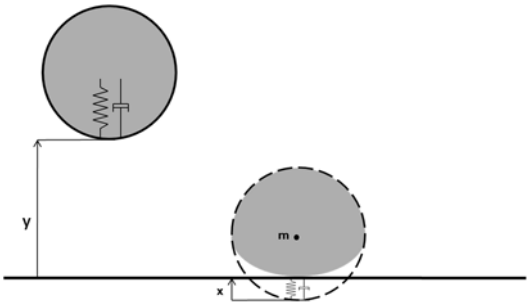


Figure 5: Ball deformation during contact phase, Kelvin-Voight model.

In case of a 'significant' contact phase, a more realistic model is necessary, which takes into account the elasticity in the contact region. The deformation can be modelled in first approximation by a spring-damper-element in parallel to the flying phase, as given in Figure 5, Kelvin-Voigt model, [5]).

Again the dynamics consist of two phase – free falling phase (or flying phase), and contact phase, but the contact phase is not any longer an isolated event, it consumes time and begin and end are controlled by state events; additionally, in both phases deformation is taken into account! In both phases three state variables characterize the dynamics: (bottom) position $x(t)$ of the not deformed ball, velocity $v(t)$, and deformation $w(t)$.

Free falling phase. For position and velocity again equation (10) or (11) is used, and a damping of first order describes the deformation $w(t)$ – during flight not coupled with position and velocity, but active:

$$\dot{x} = v,\ \dot{v} = -g - \beta v^2 \mathrm{sign}(v),\ \dot{w} = -\frac{k}{d} \cdot w \qquad (17)$$

Figure 5 shows, that in model (17) the variable v is still the velocity, but $x(t)$ is now the distance from the ground to the virtual bottom point of the (not deformed) ball, which may become negative. $w(t)$ represents the deformation, so that the actual bottom position of the deflected ball, the distance $y(t)$ from the deflected ball bottom to ground is given by output equation

$$y(t) = x(t) + w(t) \qquad (18)$$

Output equation (18) becomes now also the event function which terminates the free falling phase reaching the ground, i.e. reaching the threshold zero, causing event 'C' (*Contact*):

$$h^C(x,v,w) \overset{!-}{=} x + w \qquad (19)$$

The associated event action E^C is switching to *contact phase*.

Dynamic Contact. In the *contact phase*, the (normalized) contact force f_c determines the dynamics:

$$f_c = -kx - dv \quad (20)$$

In case of contact the dynamic equation (17) for velocity v gets the contact force f_c added as counterforce to gravity, and $\dot{w}$, change of deflection, equilibrates to velocity v. The dynamic equations in the *continuous contact phase* are consequently:

$$\dot{x} = v,\ \dot{v} = -g + f_c = -g - kx - dv,\ \dot{w} = -v \quad (21)$$

The contact phase finishes, as soon as the contact force f_c get negative, and the ball starts flying again – event '*F*' – *Fly Restart* with contact force as event function:

$$h^F(x, v, w) = f_c = -kx - dv \quad (22)$$

Following the event classification in Section 1.1, the events '*C*' *Contact* and '*F*' *Fly Restart* are *function change events* SE-F. But as in *contact phase* the equation for deformation $w(t)$ is dependent on the others in (21), dimension – and consequently also structure of the model change – so that the events can be seen as *structure change events* SE-S. On the other side, the equations (17) and (21) are relatively simple and can be formulated together by using a boolean parameter, which switches parts in the derivative functions - so that a boolean parameter is controlled by simple *parameter change events* SE-P.

Figure 6 summarizes the evolving dynamics of the ball movement until second bounce: deformation happens also in the *flying phase*, except in the first *flying phase* (but only because of because of zero deflection at begin). After the first bounce, the deformation never goes down to zero, and the ball restarts flying in deformed status.

As deflection never reaches zero after the first impact, the number of bounces must be limited – after some bounces the ball stops flying and continues movement with decreasing deflection.

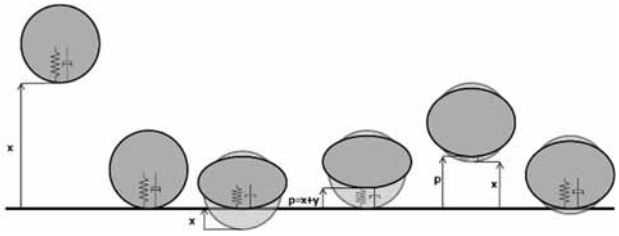


Figure 6: Phase sequence Flying – Contact – Flying for bouncing ball dynamics.

For both models it makes sense to define an additional event '*M*' *Maximium Height*, which determines the time instants where the ball reaches maximal height (zero velocity) – on first glance only an *output trace event* SE-O, but eventually of help in case of accuracy problems in the algorithm (*algorithm event* SE-A):

$$h^M(x, v, w) = v \quad (23)$$

2.3 Bouncing Ball Model – Benchmark Tasks

Generally, the tasks are model description, especially of event functions and event action, time domain analysis with model comparison and parameter studies, and especially of event handling, and comparison.

Modelling / Handling State Events with Event Contact Model

These tasks investigates modelling methods for state events of type SE-S - *state change event* (5), (12) in the *event contact model* and tests state event handling especially when handling frequent events and by comparing with analytical solutions.

Description of model implementation. Document implementation of continuous model parts (10) and (11), and of the event '*B*' –*Bounce* (12), (13), (textual model code snippets, (parts of) graphical model diagrams, etc.

Simulation until last bounce – scattering prevention. Simulate the *event contact model* (10-13) using the simulations system's event mechanism with and without air resistance and following parameters:

$x(0) = x_0 = 10\ m, v(0) = v_0 = 0, \mu = 0.9$

$\beta = 0$ or $\beta = 0.002\ \frac{1}{m}$, $g = 9.81\ \frac{m}{s^2}$, $t_{end} = 30\ s$

Event time for the 'last' bounce $t_{B,\infty}$. should be determined – by (16) and by simulation.

Straightforward implementation of events often have problems with event scattering – which happens definitely near to $t_{B,\infty}$ (16), Figure 7 – the ball 'falls' into the ground. Workaround is to stop bouncing before 'last' event, e.g. if maximal height of the flight period becomes too small.

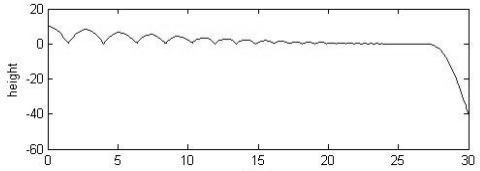


Figure 7: Scattering of *Bounce* events near the 'last' bounce due to missing error prevention.

In each flight period, the event '*M*' *Maximal Height* (23) can determine the maximal height, and below a critical maximal height further *Bounce* events are stopped by parameter change or by model change, or by change of solver parameters or accuracy of event detection, etc. Event '*M*' *Maximal Height* (23), first a simple *output trace event*, SE-O, becomes at a *parameter change event* SE-P or a *function change event* SE-F, or an *algorithm change event* SE-A.
Implement and document a proper strategy against event scattering, if necessary.

Testing accuracy of event handling. Using the linear model (11), the solution calculated by (14) gives the exact bounce times $t_{B,m}$ (15). Simulate the linear model with parameters given before (air resistance $\beta = 0$) tracks the 'numerical' bounce time instants $\hat{t}_{B,m}$.
Document results by comparison of entry times of the first 100 bounces by plotting bounce time differences $\Delta t_m := t_{B,m} - \hat{t}_{B,m}$ over number of bounces. Number of bounces can be determined easily in each bounce event by increasing a discrete output variable, so that '*B*', the *state change event* SE-X *Bounce* becomes also an *output trace event* SE-O. Again it might be necessary to prevent from event scattering!

Compensation of linear model deviation. In any case, air resistance is evident – but very small. Due to missing air resistance in the linear model, the event times are 'too late'. Simulate nonlinear and linear model with standard parameters below and try to compensate the 'too late' bounce times in the linear model by giving an initial velocity $v_0 = \delta$

$$x_0 = 10\,m,\ v_0 = 0 \text{ or } v_0 = \delta\,, \mu = 0.9$$
$$\beta = 0 \text{ or } \beta = 0.002\ \frac{1}{m}, \mathrm{g} = 9.81\ \frac{m}{s^2}, t_{end} = 30\ s$$

Modelling / Handling State Events and Parameter Studies with Continuous Contact

These tasks investigate modelling approaches for events, tests cooperation of event location with different ODE solvers, and performs parameter studies. Standard parameters are:

$$x(0) = x_0 = 10\,m, v(0) = v_0 = 0, k = 10^6\ \frac{N}{m},$$
$$d = 500\ \frac{kg}{s}, \mathrm{g} = 9.81\ \frac{m}{s^2}, \beta = 0.002\ \frac{1}{m}$$

Description of model implementation. Document implementation of continuous model parts (17) and (21), and of the events '*C*' –*Contact* (20), and '*F*' –Fly Restart (22), (textual model code snippets, (parts of) graphical model diagrams, etc.). Discuss the general modelling approach – maximal state space, hybrid decomposition, or switching model parts.

Dependency of results from algorithms. It might be necessary to choose specific ODE solvers or to tune ODE solver parameters and event detection parameters.
Simulate the d*ynamic contact model* using different ODE solvers, and / or tune parameters for ODE solvers and event detection, with $t_{end} = 30\ s$ and standard parameters.
Document of simulation results with plots or with tables indicating deviations for different algorithms, and discuss appropriateness of specific algorithmic properties (e.g. stepsize control vs event detection, or scattering prevention).

Investigation of contact phase. The proper implementation of the model (17), (21) and of the events '*C*' *Contact* (20) and '*F*' *Fly Restart* (22) can be seen in detail in the contact phase which is much shorter than the flying phase. Simulation results with standard parameters for first contact phase, the second flight phase, and the second contact phase should be given in separate time plots over $[t_{C,1} - \varepsilon_1,\ t_{F,1} + \varepsilon_1]$, $[t_{F,1} - \varepsilon_2,\ t_{C,2} + \varepsilon_2]$, and $[t_{C,2} - \varepsilon_3,\ t_{F,2} + \varepsilon_3]$, resp. ($\varepsilon_1, \varepsilon_2,$ and ε_3 being about a tenth of the length of the phase), showing all state variables, output variables, and the contact force (20) (in contact phase). Additionally, maximal height (event '*M*' *Maximal Height* (23)) in *flying phase*, and maximal deviation w in the *contact phases* – additional *output trace event* SE-O– should be determined.

Parameter studies. Variation of the spring constant k has big influence on the systems behaviour. Calculate simulation studies varying the stiffness-parameter $k = 10^6$ by a factor 100 while concurrently setting the damper constant to $d = 500$. Afterwards, vary d and finally document the relation of the parameters k and d by appropriate time plots and / or parameter plots (use standard parameters before).

Bouncing ball on Mars. It might be nice, to now about bouncing ball behaviour on Mars and to compare with behaviour on Earth. Let's simulate for 30 seconds, whereby for Mars we must use the different parameter values for gravity constant and air resistance: '$g_M = 3.69\ \frac{m}{s^2}$, $\beta_M = 2.3 \cdot 10^{-4}\ \frac{1}{m}$. Document results as comparative plots for position and velocity.

3 RLC Circuit with Diode

The second cased study is the well-known classic serial RLC circuit, with a diode in parallel – Figure 8. Diode models are partly discrete models, and events control the switching of the diode. Physical modelling systems usually provide a library with circuit elements, but it is worth to have a closer look at diode implementation and consequence of the type of implementation.

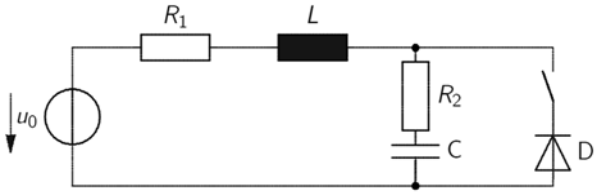


Figure 8: Serial RLC with diode in parallel

3.1 RLC model equations

Kirchhoff's laws and node equation first allow setup the physical model equations for the voltages:

$$u_C + u_{R_1} + u_L + u_{R_2} + u_C = u_0 \quad (23)$$

$$u_{R_2} + u_C = u_D \quad (24)$$

Inductor and capacitor have the constitutive equations,

$$\frac{di}{dt} = \frac{1}{L}u_L,\ \frac{du_C}{dt} = \frac{1}{C}(i + i_D) \quad (25)$$

Classic 'manual' derivation of the system equations using (23), (24), and (25) will usually choose a differential state vector $\vec{x}(t)$ with RLC current i and capacitor voltage u_C, and an algebraic state vector $\vec{z}(t)$ consisting of diode current $i_D(t)$ and diode voltage $u_D(t)$ and result in the following 'general' a state space, with $\vec{u}(t) = (u_0(t)),\ \ \vec{x}(t_0) = (i_0, u_{C,0})^T$

$\vec{x}(t) = (i(t), u_C(t))^T,\ \vec{z}(t) = (u_D(t), i_D(t))^T$:

$$\frac{du_C}{dt} = \frac{1}{C}i + \frac{1}{C}i_D \quad (26)$$

$$\frac{di}{dt} = -\frac{R_1}{L}i - \frac{R_2}{L}(i + i_D) - \frac{1}{L}u_C + \frac{1}{L}u_0 \quad (27)$$

$$0 = R_2(i + i_D) + u_C - u_D \quad (28)$$

The diode is described by a 'switching' functional relation between current and voltage, in general

$$u_D = F(i_D), i_D = 0 \ \ if \ \ u_D < 0 \quad (29)$$

The diode has a locking phase with $i_D = 0$ for $u_D < 0$, and a conducting phase for $u_D > 0$ with i_D given by

$$0 = R_2(i + i_D) + u_C - F(i_D) \quad (30)$$

3.2 Diode models and phase change

A diode is a mixed continuous – discrete element. It has two operational phases- a *locking phase* and a *conducting phase*, dependent on the diode voltage.

The mode change may be seen as simple switch (shortcut), or a conducting phase can be described by specific diode models.

Chang of phases - events

In any case, the diode voltage u_D controls the switching between *locking phase* and *conducting phase* due to equation (28). In *locking phase* with $u_D < 0$ and $i_D = 0$ equation (29) changes to $u_D = R_2 i + u_C$. A switching to *conducting phase* happens, if u_D becomes positive, which can be described by a state event '*C*' *Conducting Phase Start* with event function due to (3)

$$h^C(i, u_c) = u_D = R_2 i + u_C \quad (30)$$

with crossing from negative to positive diode voltage. In *conduction phase*, the diode voltage is given by equation (28) with nonzero diode current, calculating u_D as

$$u_D = R_2(i + i_D) + u_C$$

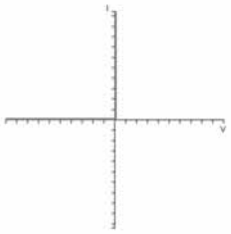
A switching to *locking phase* happens, if u_D becomes negative, which can be described by a state event '*L*' *Locking Phase Start* with event function due to (3)

$$h^L(i, u_c, i_D) = u_D = R_2(i + i_D) + u_C \quad (31)$$

with crossing from positive to negative diode voltage. Clearly, calculation of i_D depends on the choice of diode function description (29).

Shortcut diode model

The *shortcut diode model*, a simple diode model mimicries the dynamic behaviour as an ideal switch for the current depending on diode voltage u_D (Figure 9), so that the diode functional description, and also the model description (27-29) becomes simple.



$$i(u_D) = \begin{cases} 0, & u_D \le 0 \\ i(t), & u_D > 0 \end{cases}$$

$$u_D \cdot i_D = 0 \quad (32)$$

Figure 9: Diode model as ideal switch

In *locking phase*, the general model description (27-29) simplifies to an explicit linear state space following (27) with $i_D = 0$ – the model for the RLC circuit alone.

In *conducting phase*, the shortcut simplifies the model descritption to a decoupled linear state space:

$$\frac{du_C}{dt} = \frac{1}{R_2}u_C,\ \ \frac{di}{dt} = -\frac{R_1}{L}i + \frac{1}{L}u_0 \quad (33)$$

The algebraic equations (29) and (29) become obsolete, the event functions (30), (31) become simple linear threshold function $h^C(i,u_c)$ and $h^L(i,u_c)$

The events '*C*' *Conducting Phase Start* and '*L*' *Locking Phase Start* generally are state events of type *function change event* SE-F; but as the model is linear, the changes are parameter changes in the state matrix, so they can be seen also as *parameter change event* SE-P.

Shockley diode model

A diode is a nonlinear element, and indeed the switching dynamics evolve nonlinear dynamics. One nonlinear model is known as *Shockley diode model*. The mathematical description is given by an exponential-like functional relation between diode current and diode voltage (Figure 11) given by

$$i(u_D) = \begin{cases} 0, & u_D < 0 \\ I_S \cdot \left(e^{\frac{u_D}{U_T}} - 1\right), & u_D > 0 \end{cases} \quad (34)$$

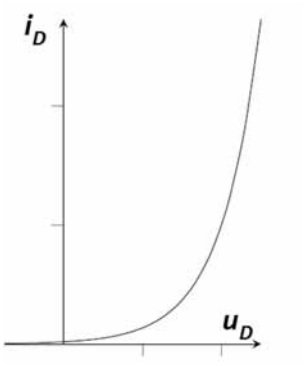


Figure 10: Diode model with Shockley characteristic

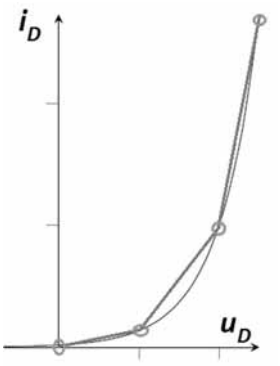


Figure 11: Diode model with interpolated Shockley characteristic

Inserting the description (34) into the algebraic equation (30) results in one algebraic equation for diode voltage:

$$I_S R_2 \left(e^{\frac{u_d}{U_T}} - 1\right) + u_c + u_d = 0 \quad (35)$$

Following notation in (3) $u_d(t)$ is an algebraic state $z(t) = u_d(t)$ and equation (35) is the corresponding algebraic equation due to (3). Alternatively, the functional relation in (34) can be inverted, so that u_d can be expressed as $u_d = U_T \ln\left(\frac{i_D}{I_S} - 1\right)$, resulting in an algebraic equation for i_D:

$$R_2(i + i_D) + u_C - U_T \ln\left(\frac{i_D}{I_S} + 1\right) = 0 \quad (36)$$

With *Shockley diode model*, the model description in *conducting phase* is governed now by a nonlinear DAE systems.

The state equation (27) becomes nonlinear when inserting the nonlinear expression (34) for i_D. The algebraic equations (35) or (36) are in any case nonlinear. The model description in *locking phase* is again the RLC model.

The events (30) and (31) for changing the modes remain unchanged – but they switch now between models with different number of states: in *locking phase*, an explicit model with two differential states, in *conducting phase*, a DAE model with two differential states and one algebraic state. Consequently, the events are of type *structure change event* SE-S, and the system is a structure-variable system.

Interpolated Shockley diode model

Characteristic curves are a classic way-around for algebraic equations. The curve for the diode's operation (Figure 10) can be made a linear interpolated table function $i(t) = S_{LIN,j}(u(t);(u_j,i_j))$ with adequate breakpoints (u_j,i_j), $j = 1,..n$. Inserting this interpolation into the algebraic equation (35) allows resolving with respect to u_D, giving a linear relation of type $i_D(t) = F_{LIN,j}(u_c(t);(u_{D,j},i_{D,j})$.

As result, the state equations in *conducting phase* become a piecewise linear explicit system, and no algebraic equation is necessary:

$$\frac{d}{dt}\begin{pmatrix} i \\ u_C \end{pmatrix} = A^{C,j}\begin{pmatrix} i \\ u_C \end{pmatrix} \quad (37)$$

The model description in *locking phase* is again the RLC model.

Explicit Shockley diode model.

In Shockley diode model, in *conducting phase* a DAE system has to be solved. DAE solvers require iteration and state event detection requires backstepping in time, which is not suitable in case of real time simulation.

The DAE system (26), (27), (35) is an index-1 system. Index reduction can transform the algebraic equation (35) to an explicit ODEs. A straightforward method is to differentiate the algebraic equation (35) directly with respect to time, resulting in a relatively complicated ODE for $u_D(t)$ with initial value $u_{D,0} = 0$ at event '*C*' *Conducting Phase Start*:

$$\dot{u}_D = -\dot{u}_C \left(I_S R_2 \left(e^{\frac{u_D}{U_T}}\right)\frac{1}{U_T} + 1\right)^{-1} \quad (38)$$

The model description in *locking phase* is again the RLC model.

The events (30) and (31) for changing the modes remain unchanged – but again they switch between different state dimensions: in *locking phase*, an explicit model with two differential states, in *conducting phase*, an explicit model with three differential states. Consequently, the events are of type *structure change event* SE-S, and the system is a structure-variable system.

3.3 RLC circuit with diode – tasks

Generally, the tasks are model description, especially of event functions/actions, and comparison of diode models.

Description of model implementations. Document implementation of the RLC model (26-30) and especially of the diode models (32), (33-36), (37), (38) with the phase changes and events (30), (31) (textual model code snippets, (parts of) graphical model diagrams, etc.). Discuss the general modelling approach – maximal state space, hybrid decomposition, or switching model parts, and possible model modifications for efficient modelling e.g. for comparing different diode models.

Dependency of results from algorithms. It might be necessary to choose specific ODE solvers or to tune ODE/DAE solver parameters and event detection parameters.
Simulate the *diode shortcut model* (32), (33) and the *Shockley diode model* (34), (35) using different ODE/DAE solvers, and / or tune parameters for ODE solvers and event detection, with standard parameters. Give plot results, indicate sensible solver parameters.

Comparison of shortcut and Shockley diode model. Compare results for *diode shortcut model* (32), (33) and for *Shockley diode model* (34), (35), with standard parameter, but timespan only two switching periods.
Document results with plots, and give a relative comparison of computation times.

Approximation of Shockley diode model. Investigate the approximation of the *Shockley diode model* (34), (35) by the *interpolated Shockley diode model* (37) with 3, 5 and 10 breakpoints for the interpolation (standard parameters, but timespan only two switching periods,
Document results with appropriate plots and with numeric deviation.

Relevance of choice of algebraic state. In case of *Shockley diode model*, either equation (34) for u_d or (35) for i_d can be used as algebraic state equation.
Simulate *Shockley diode model* with both variants, and check eventual differences – documented by plots or numeric deviations (standard parameters).

Investigation for real-time simulation. For real-time simulation, fixed step sizes and simplified event detection (without backstepping) must be used. With this premises, and with standard parameters, compare results for *diode shortcut model* (32), (33), *interpolated Shockley diode model* (37) and *explicit Shockley diode model* (38) by appropriate simulations. Document the implementation of the ODE for the diode voltage u_d .

Standard parameters (SI units):

$$R_1 = 1 \cdot 10^3, R_2 = 2 \cdot 10^1, L = 25.3 \cdot 10^{-6},$$
$$C = 100 \cdot 10^{-9}, I_S = 1 \cdot 10^{-8}, U_T = 26 \cdot 10^{-3}$$
$$t_0 = 0, \qquad t_{end} = 500 \cdot 10^{-6}$$
$$u_0 = -\sin(2\pi f \cdot t), \qquad f = 0.15 \cdot 10^{-6}$$

4 Rotating Pendulum with Free Flight Phase

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

- If the rope is tight, the mass is classically swinging (phase *swinging*); Figure 12a. Movement has one degree of freedom, usually described in polar coordinates.
- If the rope is loose, the mass is free falling (phase *falling*) until the rope is tight again (changing back in phase *swinging*; Figure 12b. Movement has two degrees of freedom, usually described in Cartesian coordinates

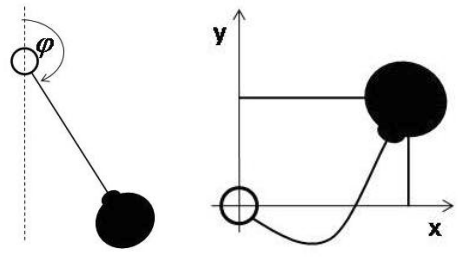


Figure 12: Left: Swinging pendulum – phase *swinging* (mass m, length l; angle φ as degree of freedom; Right: Free falling pendulum mass (phase *falling*) and Cartesian coordinates as degrees of freedom.

4.1 Model description

Equations for movement in phase *swinging* can be derived by using the angular momentum balance, resulting in the classic pendulum equation:

$$\ddot{\varphi} + \frac{k}{m}\dot{\varphi} - \frac{g}{l}\sin(\varphi) = 0 \tag{39}$$

with damping factor k, the mass m, rope length l and earth acceleration g. An explicit state space is given by

$$\vec{x}_S(t) = \big(\varphi(t), \dot{\varphi}(t)\big)^T \text{ or } \vec{x}_S(t) = \big(\varphi(t), v(t)\big)^T \tag{40}$$

with $\dot{\varphi}$ angular velocity, v tangential velocity

The pendulum is swinging, as long as the force on the rope is bigger than zero.

$$F = -gm\,cos(\varphi) + ml\dot{\varphi}^2 \tag{41}$$

If this force $F(t)$, an output equation, becomes lower than zero, the gravitational force outweighs the centrifugal force: the pendulum is changing into phase *falling*.

In phase falling, the movement of the body has two degrees of freedom. The motion of the mass is derived by conservation of momentum in x- and y- direction:

$$\begin{aligned} m\ddot{x} &= -k\dot{x}, \\ m\ddot{y} &= -mg - k\dot{y}. \end{aligned} \tag{42}$$

An explicit state space is given by

$$\vec{x}_F(t) = \big(x(t), \dot{x}(t), y(t), \dot{y}(t)\big)^T \tag{43}$$

The distance d from rotation center

$$d^2 = x^2 + y^2 \tag{44}$$

indicates whether the rope is loose ($d < l$) or whether it gets tight again ($d = l$), forcing the body back on the circular path: the mass switches to phase *swinging*.

Figure 13 shows an overview of the two phases with the different models and with the criteria for the changes of phase. The overall system is a typical structural-dynamic system, and in modelling the change of degrees of freedom – the change of the dimension of the model – has to be mastered.

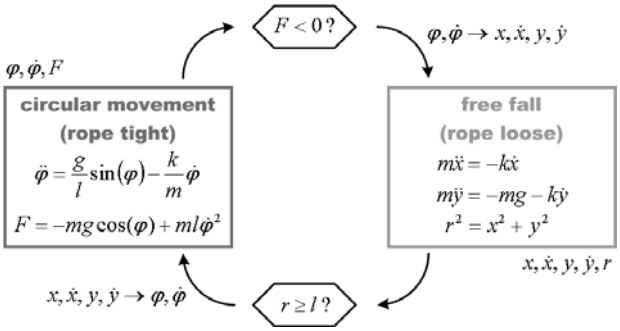


Figure 13: Summary of the two state models with criteria for the state changes

4.2 Change of phases – events

The change from phase *swinging* to phase *falling* can be described by the state event '*F*' *Falling*, which is given by a state event function using the formula (41) for the force on the rope:

$$h^F(\varphi, \dot{\varphi}) = F = -gm\,cos(\varphi) + ml\dot{\varphi}^2 \qquad !- \tag{45}$$

A crossing into negative direction activates the event. The associated event action $E^F(\varphi(\hat{t}), \dot{\varphi}(\hat{t}))$ requires a change of the model – with change of degress of freedom, with calculation of new initial values. The state event therefore is a typical *structure change event* SES.

The change from phase *falling* to phase *swinging* is modelled by the event '*S*' *Swinging* which is activated, when the rope gets tight – measured by the distance (44) and described by the event function

$$h^S(x, y) = d^2 - l^2 = x^2 + y^2 - l^2 \qquad !+ \tag{46}$$

A crossing into positive direction activates the event: the associated event action $E^S(x(\hat{t}), \dot{x}(\hat{t}), y(\hat{t}), \dot{y}(\hat{t}))$ requires a change of the model – with change of degrees of freedom. The state event therefore is a typical *structure change event* S-ES.

Structural-dynamic models can be implemented by hybrid decomposition, or by a maximal state space. A hybrid decomposition for this case study is sketched in Figure 13. A maximal state space approach would require a state space of dimension 6

$$\vec{x}_M(t) = \big(\varphi(t), \dot{\varphi}(t), x(t), \dot{x}(t), y(t), \dot{y}(t)\big)^T$$

where then depending on the event functions (45), (46) states are 'frozen' in the respective phases.

From physical modelling an alternative approach is suggested. The movement in the phase *swinging* can also be described in polar coordinates: the mass is moving freely, but a force $\lambda(t)$ forces the movement on a circle:

$$\begin{aligned} m\ddot{x} &= -k\dot{x} - \lambda x \\ m\ddot{y} &= -mg - k\dot{y} - \lambda x \end{aligned} \tag{47}$$

$$0 = x^2 + y^2 - l^2 \tag{48}$$

The above DAE system indeed describes the swinging of the mass, and it could be used instead of the model in polar coordinates (31). The events given by (45), (46) could now switch easier between the phases, because the differential state space is almost the same – so one state space with internal switching could be used (the model for the phase *swinging* must additionally solve an algebraic equation – with algebraic state $\lambda(t)$.

Interestingly, the algebraic state equation (48) in phase *swinging* is the same than the event function (46) (output equation) in phase *falling*. Unfortunately the DAE system (47), (48) is difficult so solve, because it has a differential index of 3 – which makes index reduction necessary.

4.3 Rotating pendulum – tasks

Tasks in this case study concentrate on the modelling approach and model implementation, and on few simulation.

Description of model implementations. Document implementation of the structural-dynamic system with phase *swinging* (39) and phase *falling* (42) the sate events (45), (46) which switch the phases. Alternatively describe the implementation of the 'common' DAE system (47), (48) (textual model code snippets, (parts of) graphical model diagrams, etc.). Discuss the general modelling approach – maximal state space, hybrid decomposition, DAE system with switching model parts, and possible modifications for efficient modelling.

Basic simulation of phases. Calculate and visualize a basic simulation run with the following parameters:

$$m = 1.5\ kg, k = 0.9\ \frac{kg}{s}, = 1\ m, = 9.81\ \frac{m}{s^2}$$

and the initial conditions

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

Simulate beginning until the maximal oscillation does not exceed $\frac{\pi}{10}$ any longer (indicate time instant) – could be determined by adding an output event!

Dependency of results from algorithms. It might be necessary to choose specific ODE solvers or to tune ODE/DAE solver parameters and event detection parameters.

Perform simulations with standard parameters, timespan until begin of second phase *swinging* using different ODE/DAE solvers, and / or tune parameters for ODE solvers and event detection. Give plot results and indicate sensible solver parameters.

External energy supply. Due to physical constraints, only one phase *falling* can occur because of energy loss. In order to 'restart' the alternating movements, energy is supplied – as increase of the angular velocity (a 'kick').

Following the basic simulation of the second task, the angular velocity is increased by a factor, so that the pendulum reaches again the phase falling.

At the first zero crossing after angle did not exceed $\frac{\pi}{10}$, the angular velocity multiplied by a factor γ – a state event of type *state change event* SE-X

Determine factors γ so that

(i) the same movement than before results,

(ii) the next falling phase starts at $\varphi = \pi$, and

(iii) the swinging phase makes two rotations

5 Conclusion

This benchmark is a challenging one. We hope, that 'solution'sent in enrich the variety of modelling approaches and clarify some inconsistencies in state event modelling. We invite simulationist to provide a 'solution' – with publication of a *Technical Benchmark Note* (up to 10 pages) in SNE. Furthermore we ask for model source codes, for download by readers.

References

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

[2] Breitenecker F. ARGESIM Benchmarks on Modelling and Simulation: Revised Definitions, Extended Solutions, and Supplemental Information. SNE 16(3-4), 57-58, 2006.

[3] F. Breitenecker F, Zauner G, Popper N, Judex F, Troch I. *Structure of Simulators for Hybrid Systems - Development and New Concept of External and Internal States.* SNE Simulation News Europe 17(2), 39–48, 2007.

[4] Körner A, Winkler S, Breitenecker F. Possibilities in State Event Modelling of Hybrid Systems. Proc. EUROSIM Congress Oulu, Sept. 2016, to appear.

[5] H. Ecker. *The bouncing ball problem – modelling and simulation aspects.* SNE Simulation News Europe, 12(1), 9 – 14, 2002.