NUMERICAL SIMULATION OF CONTINUOUS SYSTEMS WITH STRUCTURAL DYNAMICS

Olaf Enge-Rosenblatt, Jens Bastian, Christoph Clauß, Peter Schwarz

Fraunhofer Institute for Integrated Circuits,
Design Automation Division,
Zeunerstraße 38, 01069 Dresden, Germany

olaf.enge@eas.iis.fraunhofer.de

Abstract

In this paper, "continuous systems with structural dynamics" shall be understood as dynamical systems consisting of components with continuous and/or discrete behaviour. (This notation should not be confused with the term "structural dynamics" in the context of Finite Element simulation). Continuous systems with structural dynamics – or so-called "hybrid systems" – can often be investigated only by a so-called "hybrid simulation" which means a simultaneous simulation of continuous-time dynamics (modelled by differential equations or differential-algebraic equations (DAE)) and discrete-event dynamics (modelled e.g. by Boolean equations, finite state machines, or statecharts). To this end, an algorithm for numerical simulation of hybrid systems must be able to both solve a DAE system within a "continuous" time progression as well as to deal with event-driven phenomena.
In the paper, the point of view is emphasized that the structure of a continuous system is closely combined to the structure of the DAE system which describes the continuous system's dynamical behaviour. In this context, discrete-time events are considered as phenomena which may cause a change of the DAE system's structure. Furthermore, a distinction between *systems* with variable structure and *models* with variable structure is explained. The main part of the paper deals in detail with a simulation algorithm suitable for hybrid systems. This algorithm consists of a "continuous phase" (for numerical integration of the DAE system) and a "discrete phase" (for interpreting the event, establishing the new valid DAE system, calculating the new initial values). Some simulation results dealing with selected models and using the multi-physics language Modelica will complete the paper.

Keywords: Structural dynamics, Hybrid systems, Discrete-continuous simulation, Simulation algorithm, Modelica

Presenting Author's biography

Olaf Enge-Rosenblatt received the Diploma in automation engineering and the Ph.D. degree in electrical engineering from the Chemnitz University of Technology, Chemnitz, Germany, in 1986 and 2005, respectively. From 1992 to 2004, he was with the Institute of Mechatronics in Chemnitz mainly working on unified mathematical descriptions of electromechanical systems and on modelling of systems with structural variability. Since 2005, he has been with the Fraunhofer Institute for Integrated Circuits in Dresden, Germany. His research interests include modelling of heterogeneous – especially mechatronic – systems using multi-physics or multi-domain approaches.

ISBN 978-3-901608-32-2

1 System structure – what is it?

This paper deals with changes of the "structure" of a dynamic system during a simulation process. But what is the structure of a dynamic system? Many properties could be considered to possibly belong to the structural description of such a system. In mechanical domain, the number of interacting bodies and the number of joints between them belong to the structural information as well as the fact which two bodies are connected by which kind of joint. A body's geometrical shape is of no importance in this context. In electrical domain, the number and types of electrical components and their galvanic connections among each other belong to the structural information. It does not care whether, e.g., a voltage source has a constant value or a sinusoidal time behaviour. Similar descriptions can be found for other physical domains (hydraulic, pneumatic, thermodynamic, etc.).

To sum up all these different properties, we assume in this paper that the structure of a system can be interpreted as the structure of its mathematical model, i.e. the number, types and structure of differential and/or algebraic equations belonging to the model. Finally, this structure manifests itself within the fill-in structure of the equation system's Jacobian.

In this paper, a mathematical model which possesses the possibility to change its structure because of some kind of "events" will be denoted to as a model with structural dynamics.

2 Why structural dynamics?

Many physical or technical systems change their properties during operation. Variation of model parameters is a common situation in simulating dynamic systems. But very often, changes of properties occur depending on events which may appear at certain points in time (time-discrete phenomena). In these cases, the complete system shows both time-continuous and time-discrete behaviour. Such systems are often called *hybrid systems*. They arise in many fields including robotics, embedded systems, transportation systems, process control, biological and chemical systems, mixed-signal (analogue-digital) integrated circuits, etc.

Events occurring in hybrid systems can be distinguished into

- events depending only on time (i.e. they can be collected within a time queue) and
- events depending on other physical quantities of the system (i.e. they happen if a variable crosses the zero border value).

Investigation of hybrid systems has a long-lasting history (see e.g. [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]). Their dynamic simulation is supported by some simulators (see [8]), e.g. Matlab/Simulink, Aspen, gPROMS, Dymola, Saber, Mosilab, and various VHDL-AMS simulators. But most of them only support very simple model variations. A fundamental change of model structure – such as adding or removing differential and/or algebraic equations (which we call "structural dynamics") – is not possible in most simulators. Such behaviour leads to complicated mathematical problems. Mainly, it has to be guaranteed after a structural change that, first, the correct differential-algebraic equations are chosen and, second, a set of consistent initial values of the state variables can be calculated.

From the application point of view, it is important to distinguish between *systems* with a varying structure and *models* with varying structure (but the system itself is not varying). A *system* having a varying structure is characterized either by existence of so-called unilateral constraints (see e.g. [16], [17], [18], [19]) or by appearance of switches for activating or deactivating parts of the system. Such a system does really change its structure or at least its structural information in the behavioural equations during operation. Examples may be found in different application areas:

- mechanics: clutches, collision of masses, Coulomb friction, "maximum distance"-phenomena (see Fig. 1),
- electronics: parts of the system are suspended for a certain time period (e.g. for saving electrical power in mobile communication devices),
- power electronics: switches and relays as well as diodes and thyristors (if they are considered as ideal switches, see Fig. 2),
- adaptive manufacturing machines and roboters: they have to handle different objects and have to adjust themselves to changing situations.

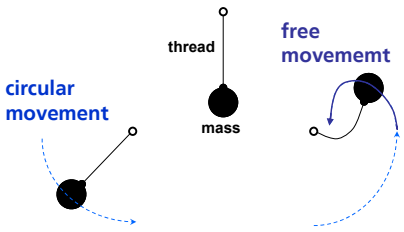

Fig. 1 String pendulum

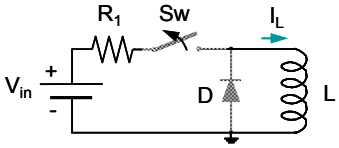

Fig. 2 Switched diode circuit

Other reasons may lead to *varying models* of the same system because system's behaviour shall be

investigated under different circumstances. Examples of such reasons may be:

- accuracy shall be adjustable to a more or less detailed model during simulation (to be able to "simulate as accurate as necessary", see Fig. 3),
- usage of different model designs for "dynamic mode" (transient investigation) and "steady-state mode" with the intention to switch between them during simulation (see e.g. [20]).

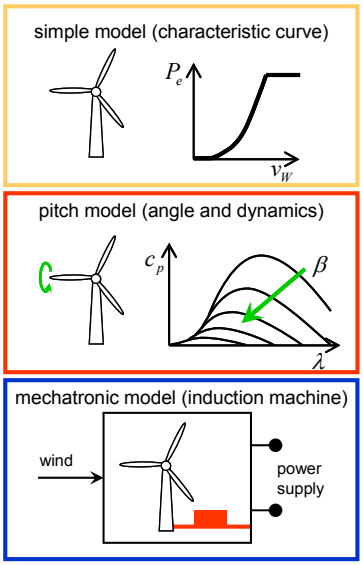

Fig. 3 Three levels of wind generator modelling

From our point of view, investigation of hybrid models is much more than a simultaneous simulation of continuous-time dynamics (modelled by differential equations or differential-algebraic equations (DAE)) and discrete-event dynamics (modelled e.g. by Boolean equations, finite state machines, or statecharts). A hybrid model should rather be considered as a model which, beyond its continuous-time and discrete-event properties, possesses the possibility to change the structure of behavioural equations at certain points in time (also called events) because of various reasons. Hence, a simulator for hybrid models has not only to be able to handle continuous and discrete parts with appropriate numerical solvers but, furthermore, should provide concepts and statements for the definition of models with variable structure. This includes a practical solution of the expected numerical issues coming up with a change from one set of differential-algebraic equations to another. The next section gives an overview what types of structural changes may occur with dynamic systems.

3 Types of structural changes

Investigating practical simulation problems, structural changes may arise in different ways. A summary is shown in Fig. 4. The most important issue is "change model behaviour" in the first row. This issue includes

- simple substitution of one differential equation by another one,
- exchange of a system of differential equations for another one but with the same order,
- replacement of behavioural or structural description of a component by a totally other one (e.g. a drastic variation of model order, change between continuous and discrete behaviour, substitution of a model description by coupling with another simulator).

But also the interconnections between components and, therefore, the structure of the system may change (see rows two to five of Fig. 4). Adding and deleting of certain blocks to/from the complete model (issue "Additional blocks", second row) requires a correct handling of these connectors which are sometimes "opened" (i.e. not connected). In the "open"-case, an additional equation has to be added automatically to the model that enforces the vanishing of the flow variable of the concerning connector. The issue "Change connections" (third row) yields a simple change of parts of some algebraic equations. "Additional blocks and connections" (row four) combines the issues above. The "Change number of ports"-issue may be a consequence of changing the block content from a simple model to a very detailed one or vice versa.

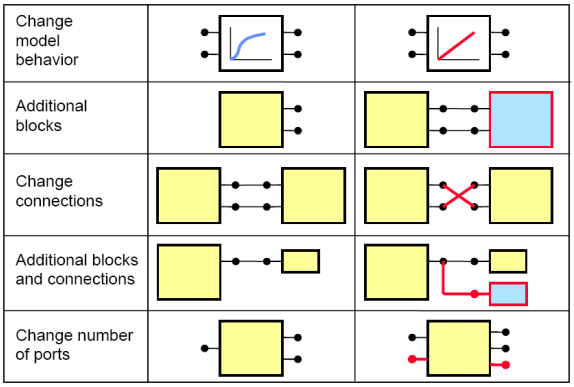

Fig. 4 Structural dynamics

4 A hybrid simulation algorithm

4.1 Algorithm principle

The simulation of continuous-discrete systems is supported by many powerful tools. But in handling varying model structures, most simulators have strong restrictions. The Modelica simulator Dymola, e.g., allows that equations may change in an `if-then-else` clause, but the *number of equations* in both branches must be the *same*. Similar restrictions exist

in many other simulators which allow the usage of hybrid models.

In this section, an approach for simulating hybrid models is proposed which is able to deal with structural variability. This approach was implemented within the experimental simulator *Mosilab* (see [21], [22], [23]). This simulator was developed within the German applied research project GENSIM by some Fraunhofer Institutes. Within Mosilab, an extension of the language Modelica by a concept for dealing with structural dynamics has been intended. For this purpose, a description of statecharts in graphical and textual way was implemented.

In the following, the *structural variability* of a model is characterized using *statecharts*. Roughly speaking, every state stands for a certain set of differential-algebraic equations and every transition realises a change between different model structures.

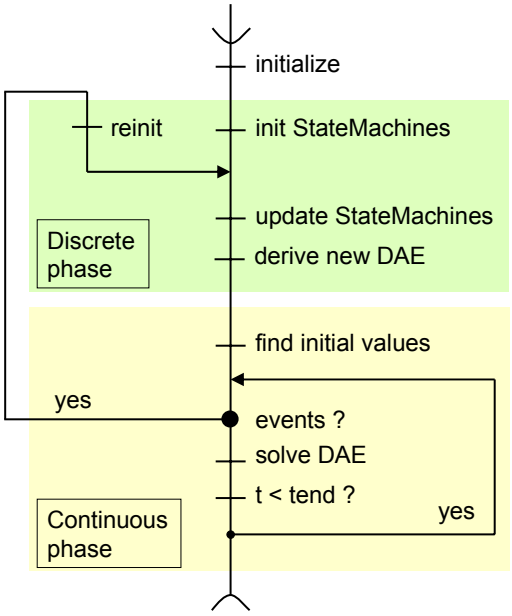

Fig. 5 A hybrid simulation algorithm

The basic algorithm is shown in Fig. 5. It consists of two phases, a *discrete phase* and a *continuous phase*. The main issue in the discrete phase is to update all state machines of the hybrid model (one state machine is described by one statechart) and to establish the new set of differential-algebraic equations if necessary. Hence, all structural changes of a hybrid model are carried out within the discrete phase. In this context, it is assumed that structural changes occur at discrete points in time, i.e. they shall not be executed within a certain time interval (zero time assumption). Between two successive discrete points in time, only analogue simulation is performed. This analogue simulation is carried out by a numerical DAE solver and must continue for a minimum time interval which is greater than zero. The length of such a simulation interval may, a priori, either be known or be unknown. If the next discrete-time event happens at a determined (time-fixed) moment then the interval's length is known. Otherwise, e.g. if the next event being expected is triggered by a zero crossing of a variable, the length of the simulation interval is unknown. In the latter case, the relevant quantity has to be monitored in an appropriate way.

4.2 Discrete phase

Fig. 6 shows a more detailed outline (compared to Fig. 5) of the discrete phase of the hybrid simulation algorithm. Please note that *simulation time* keeps *constant* during the complete discrete phase. The algorithms of the discrete phase influence only the discrete parts of the hybrid model. Hence, states and transitions of the statechart diagram are under special focus. But the *model structure* of the continuous submodel (including the DAE set belonging to) and the discrete variables *may be affected*, too.

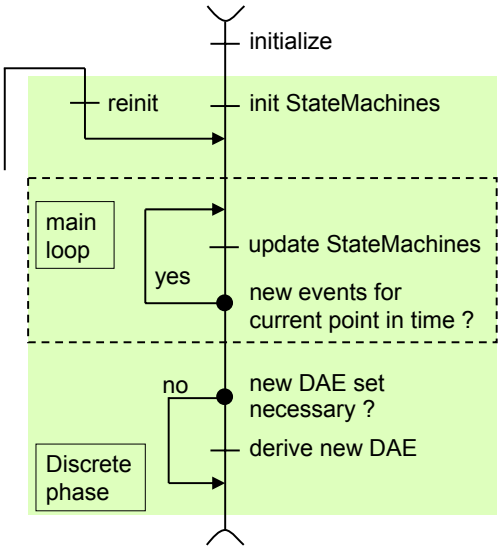

Fig. 6 Discrete phase of hybrid simulation algorithm

At start of numerical simulation, all state machines must be initialized by evaluating the initial states and their associated transitions. The main loop of the discrete phase consists of one or more updating processes of the state machines and, after every updating process, the question for new events which may be raised within the last update. During every updating process, two sets of events are to be distinguished: the set of *active events* and that of *waiting events*. One updating process handles all active events and fires the associated transitions successively. New events, which may be raised by fired transitions or by exit actions or entry actions of the associated states, are collected in the set of waiting states. If no more active events are available, one updating process is finished. If now the set of waiting events is empty then the main loop can be closed. Otherwise, another updating process is necessary. For this purpose, all waiting events are transferred into the set of active events and the next update is started.

After leaving the main loop, it has to be proved wether the model structure has changed. This can be done in a

very simple way assuming that different activation configurations of the state machines before and after the current discrete phase refer to a change of the structure of the continuous submodel. If the structure is unchanged, the discrete phase can be finished and the following continuous phase is ready to go. In case of structural changes, the set of behavioural equations has to be changed, too. The new set of differential-algebraic equations has to be chosen according to the currently active states in all state machines. At start of the following continuous phase, consistent initial conditions have to be found. To simplify this task – or even perhaps to enable a solution – it may be necessary for the user to define a mathematical algorithm how some initial values of the new model structure are to be calculated from the values of the old one. Such an algorithm would have to be specified within the transitions which are responsible for the appropriate structural change.

4.3 Continuous phase

A more detailed outline (compared to Fig. 5) of the continuous phase of the hybrid simulation algorithm is shown in Fig. 7. Please note that within this phase, the *structure* of the complete hybrid model keeps *unchanged*. The algorithms of the continuous phase only affect the continuous parts of the hybrid model. Hence, finding consistent initial values (see e.g. [24], [25], [26]) as well as solving the present set of differential-algebraic equations is the main issue of this phase. But the recognition of possibly occurring events is also important.

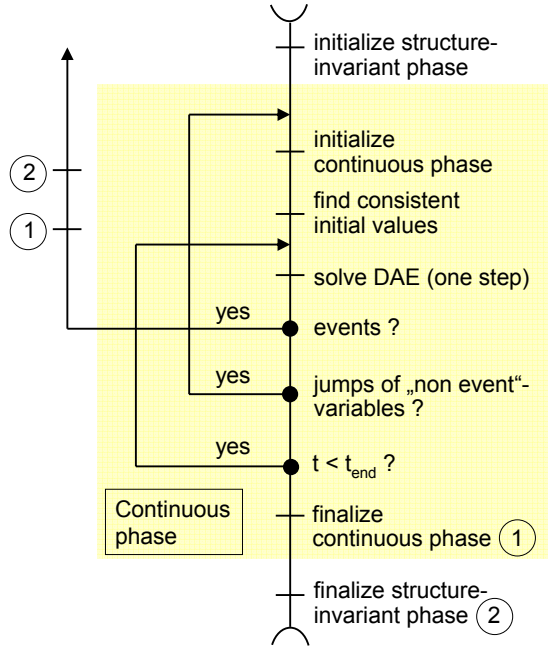

Fig. 7 Continuous phase of hybrid simulation algorithm

The continuous phase begins with an initialization process. In case of carrying out this step for the first time ($t = 0$), the user-given initial values for physical quantities of the continuous model are accepted. Otherwise, the values of the physical quantities calculated within the last discrete phase are used as initial values. Using these values as a start configuration, consistent initial values – i.e. values which fulfil the constraints of the DAE – have to be found in the next step.

The main task of the continuous phase is to solve numerically an initial value problem of the form

$$\begin{aligned} &F(t, y, \dot{y}) = 0, \\ &y(t_0) = y_0, \quad \dot{y}(t_0) = \dot{y}_0, \end{aligned} \tag{1}$$

where y_0 and $\dot{y}_0$ are consistent initial values, i.e. they fulfil the residuum $F(t_0, y_0, \dot{y}_0) = 0$. The vector y consists of both differential variables (the relevant $\dot{y}$-element appears in the DAE) and algebraic variables (no relevant $\dot{y}$-element appears in the DAE). An appropriate numeric solver can be used to solve the problem (1) with advancing time. (In Mosilab, the numeric solver IDA [27] is used.) The numerical integration process may possibly be continued until the end time of simulation t_{end} is reached. However, there are some reasons for stopping the numerical integration at an earlier point in time.

The first reason is the possible appearance of a structural change. In such a case, the numeric solver would have to be stopped at a point in time which lies as near as possible to the moment of event. In order to recognize structural changes, so-called "event variables" are defined. These variables are differential or algebraic variables which may cause an event in the sense of structural dynamics. The event variables are monitored during the numerical integration process. After each time step of the solver, all event variables are compared to their values before the last integration step. If a change of an event variable is detected then the first moment of changing this variable within the current integration interval must be determined. This can be done e.g. by a root finding algorithm. In this context, it is important to use only numerical solutions at points in time before the event occurs. Otherwise, the accuracy of the calculated moment of event may be affected negatively. After determination of event point in time, the continuous phase is finished and the next discrete phase is started.

The second reason for stopping the numerical integration before reaching t_{end} is the possible jump of a so-called "non event"-variable. Such variables are differential or algebraic variables which are not associated to any event of structural changes. In case of jumping of such a variable, the IDA's integration interval becomes smaller and smaller. If the integration interval drops below a certain border value (denoted by $\Delta t_{\min}$), the solver is reinitialized at the

point in time $t_{i+1} = t_i + \Delta t_{\min}$ and new consistent initial values are computed. After that, a new numerical integration process is started.

4.4 Special aspects

The necessary calculation of consistent initial values at each beginning of a continuous phase or after a jump of a non event variable is sometimes a crucial task. Therefore, the finding procedure may fail. Some helping methods were implemented into hybrid simulation algorithm of Mosilab to overcome this problem. One of them, the homotopy method, shall be mentioned here.

The homotopy method is a procedure to calculate the solution of the problem

$$H(z) = 0 \tag{2}$$

starting from a known solution z_0. For this purpose, the original problem (2) is substituted by the following problem

$$\tilde{H}(z,\lambda) = \lambda H(z) - (1-\lambda) H(z_0) = 0\,. \tag{3}$$

If $\lambda = 0$, this equation is trivial. By increasing λ stepwise, new problems of the form (3) are established. Generally, the solution z_{k-1} of the preceding problem (3) is used to find a solution z_k of the current problem (3). In case of convergence, this solution is used in the next step (with furthermore increased λ). If no solution z_k can be found then λ is decreased and a new trial is started using z_{k-1}. The complete algorithm as used in Mosilab is shown in Fig. 8.

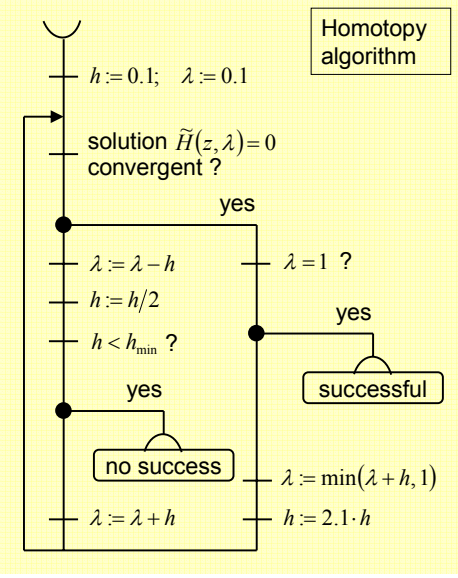

Fig. 8 Homotopy method algorithm

5 Simulation experiment

In order to show the function of the presented algorithm, some simulation results of a simple 2D string pendulum are given. A detailed sketch of the example is shown in Fig. 9.

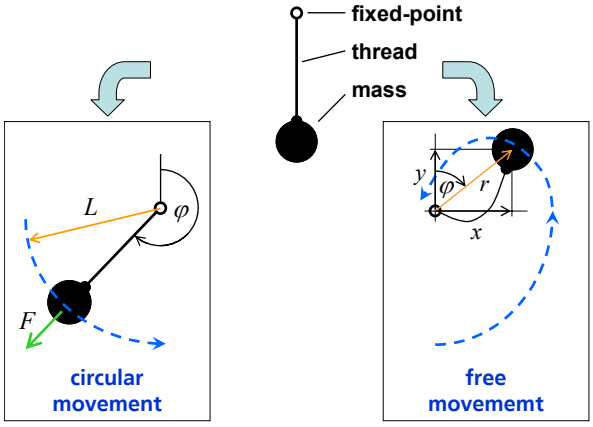

Fig. 9 Sketch of string pendulum

A point mass (having the mass parameter *m*) is attached to a fixed point by a non-elastic thread. The maximum length of the thread shall be denoted to as *L*. The mass can perform two kinds of movements: a circular movement in case of a fully stretched thread (Fig. 9, left hand side) and a free movement in case of a non-stretched thread (Fig. 9, right hand side). An appropriate statechart is depicted in Fig. 10. The model has two states called `bound` and `free`. Within the circular movement, one differential equation of second order is valid (having the state quantities $\dot{\varphi}$ and φ, where *g* means the gravity constant and κ denotes a damping coefficient). Within the free movement, however, two differential equations of second order are needed (having the positions in x- and y-direction and their time derivatives as state quantities, where *k* means a damping coefficient and *r* denotes the current distance between point mass and fixed point).

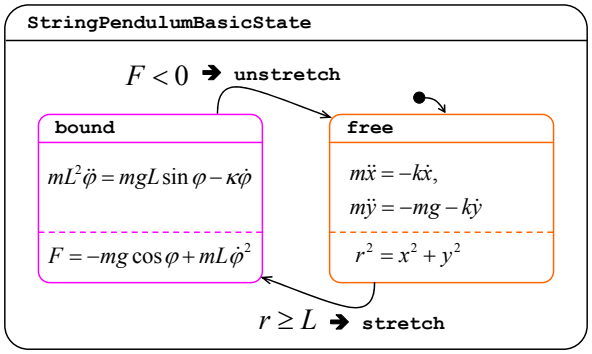

Fig. 10 Statechart of string pendulum

The system remains in the `bound`-state as long as the centrifugal force of the mass holds the thread at its full length. If the sum of forces acting on the mass drops below zero ($F < 0$), this state will be leaved and the

`free`-state will become active. The relevant transition is called `unstretch`. Within this transition, the current position and velocities have to be calculated from the last valid values of the physical quantities of the `bound`-state. On the other hand, the `free`-state is valid as long as the distance between point mass and fixed point is less than the full length of the thread. If the full length is reached or exceeded ($r \geq L$), the system will change from the `free`-state into the `bound`-state. The relevant transition is called `stretch`. Within this transition, the current angle and angular velocity have to be calculated from the last valid values of the physical quantities of the `free`-state. Please note that the energy conservation law may not be fulfilled during this structural change.

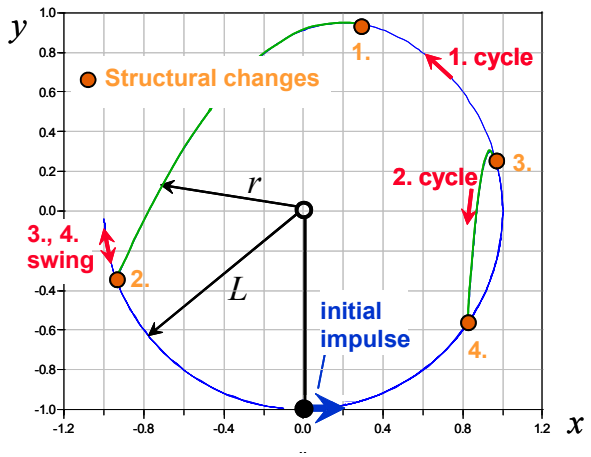

Fig. 11 Simulation result – *xy*-plot

Fig. 11 shows an *xy*-plot of the string pendulum under the assumption that the mass is located near its rest position at start of simulation and an initial velocity in positive *x*-direction is given. The pendulum performs two cycles followed by a decreasing oscillation. In the first cycle, two structural changes occur (denoted by no. 1 and 2). The first one switches from circular to free movement, the second one changes contrarily. The same appears within the second cycle (structural changes no. 3 and 4). After the fourth switch, the thread remains stretched to its full length during the decreasing oscillations.

The following figures show time histories of some interesting physical quantities. Fig. 12 depicts the time association to the *xy*-plot in Fig. 11. In case of free movement, *x* and *y* are differential variables of the DAE, while in case of circular movement, both variables have to be computed from the current angle. Contrary to this, Fig. 13 shows the angle φ which is known during the circular movement and has to be calculated within the free movement. The structural changes can be determined best in the curves of the two monitoring variables: the force within the thread (see Fig. 14) and the distance between the mass and the fixed point (see Fig. 15).

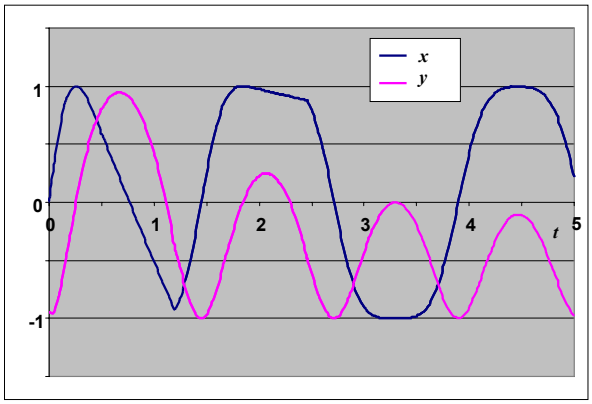

Fig. 12 Simulation result – *x*- and *y*-coordinates

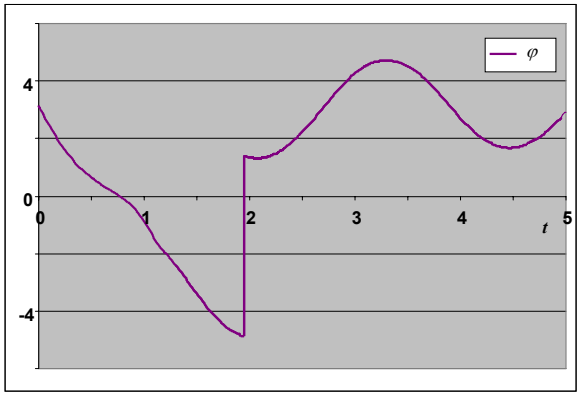

Fig. 13 Simulation result – angle

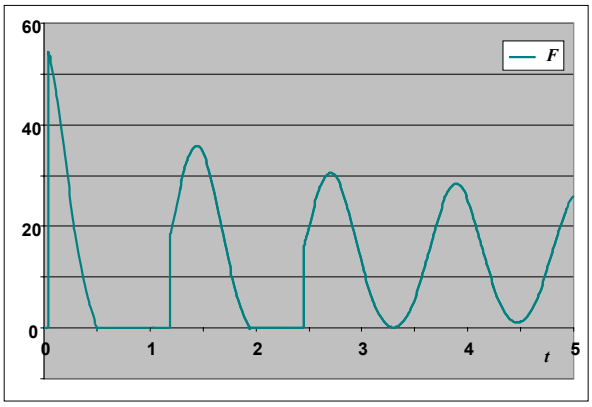

Fig. 14 Simulation result – force

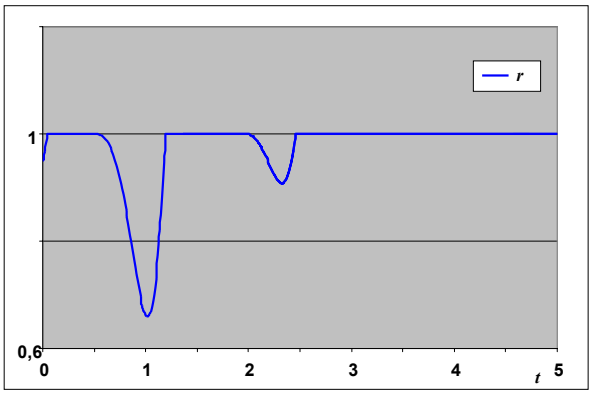

Fig. 15 Simulation result – distance

6 Conclusion

The numerical simulation of continuous systems with structural dynamics requires simultaneous handling of continuous-time dynamics and discrete-event dynamics. Hence, a tool suitable for simulating such systems must offer facilities to describe both phenomena. In particular, the interactions between both worlds, i.e. triggering events by the continuous model as well as changing the continuous model's structure by events, have to be taken into account.

In the paper, different types of structural changes are listed. The full variety of these cases is hardly supported by well-known simulation tools. Hence, the paper presents a hybrid simulation algorithm consisting of a discrete phase and a continuous phase. The simulator switches between these two phases at certain points in time in an appropriate way. The discrete phase influences only the discrete parts of the hybrid model while the simulation time keeps constant. Execution of events and their consequences on changes of the model structure are under focus. The continuous phase affects only the continuous parts of the hybrid model while the model structure keeps unchanged. The main issue is to find consistent initial values and to carry out numerical integration of the DAE while monitoring relevant variables for recognition of possibly occurring events.

Beyond that, a homotopy method for supporting the overcome of the consistent initial values finding problem is presented. Finally, some simulation results of a string pendulum are given.

7 References

[1] P. Antsaklis, X. Koutsoukos, J. Zaytoon. On hybrid control of complex systems: a survey. *J. Européen des Systèmes Automatisés*, 32:1023-1045, 1998.

[2] P.I. Barton, C.K. Lee. Modeling, simulation, sensitivity analysis, and optimization of hybrid systems. *ACM Trans. Mod. Comp. Sim.*, 12:256-289, 2002.

[3] D.A. van Beek, J.E. Rooda. Languages and applications in hybrid modelling and simulation: Positioning of Chi. *Control Engineering Practice,* 8:81-91, 2000.

[4] F. Breitenecker, I. Troch. Simulation software – development and trends. In: H. Unbehauen, editor, *Control Systems, Robotics and Automation, Theme in Encyclopedia of Life Support Systems*, UNESCO / EOLSS Publishers, Oxford/UK 2004, Article No. 6.43.7.7 [http://www.eolss.net].

[5] F.E. Cellier. *Continuous System Modeling*. Springer. 1991.

[6] F.E. Cellier, H. Elmqvist, M. Otter, J.H. Taylor. Guidelines for modeling and simulation of hybrid systems. *Proc. IFAC World Congress*, Sydney, Australia, 1993, vol.8, 391-397.

[7] H. Gueguen, M.-A. Lefebvre. A comparison of mixed specification formalisms. *J. Européen des Systemès Automatisés*, 35:381-394, 2001.

[8] http://www.laas.fr/cacsd/hds

[9] Hybrid Systems: Computation and Control. Springer Lecture Notes in Computer Science (LNCS), *Proceedings of the HSCC workshops*.

[10] KONDISK: German research project on continuous-discrete systems; see http://www.ifra.ing.tu-bs.de/kondisk/

[11] E.A. Lee, H. Zheng. Operational semantics of hybrid systems. *Proc. HSCC 2005,* Zurich, Switzerland, Springer LNCS 3414, 25-53.

[12] P. Mosterman. An overview of hybrid simulation phenomena and their support by simulation packages. *Proc. HSCC 1999*, Berg en Dal, The Netherlands, Springer LNCS 1569, 165-177.

[13] M. Otter. *Objektorientierte Modellierung mechatronischer Systeme am Beispiel geregelter Roboter*. Dissertation, Fortschrittberichte VDI, Reihe 20, Nr. 147, VDI-Verlag, 1995.

[14] M. Otter, M. Remelhe, S. Engell, P. Mostermann. Hybrid models of physical systems and digital controllers. *J. automatisierungstechnik*, 48:426-437, 2000.

[15] R. Saleh, S.J. Jou, A.R. Newton. *Mixed-Mode Simulation and Analog Multilevel Simulation*. Kluwer, 1994.

[16] O. Enge. *Analyse und Synthese elektro-mechanischer Systeme*. Dissertation, Shaker, Aachen, 2005.

[17] O. Enge, P. Maißer. Modelling electromechanical systems with electrical switching components using the linear complementarity problem. *Journal Multibody System Dynamics*, 13:421-445, 2005.

[18] C. Glocker. *Dynamik von Starrkörpersystemen mit Reibung und Stößen*. Dissertation, Fortschrittberichte VDI, Reihe 18, Nr. 182, VDI-Verlag, 1995.

[19] F. Pfeifer, C. Glocker. *Multibody dynamics with unilateral contacts*. John Wiley & Sons, 1996.

[20] O. Enge et al. Quasi-stationary AC analysis using phasor description with Modelica. *Proc. 5th Modelica Conf.*, Vienna, Austria, 2006, 579-588.

[21] http://www.mosilab.de

[22]C. Nytsch-Geusen et al. Mosilab: Development of a Modelica based generic simulation tool supporting model structural dynamics. *Proc. 4th Modelica Conf.*, Hamburg, Germany, 2005, 527-535.

[23]C. Nytsch-Geusen et al. Advanced modeling and simulation techniques in Mosilab: A system development case study. *Proc. 5th Modelica Conf.*, Vienna, Austria, 2006, 63-71.

[24]P.N. Brown, A.C. Hindmarsh, L.R. Petzold. Consistent initial condition calculation for differential-algebraic systems. *SIAM J. Sci. Comp.,* 19:1495-1512, 1998.

[25]C. Pantelides. The consistent initialization of differential-algebraic systems. *SIAM J. Sci. Stat. Comput*., 9:213-231, 1998.

[26]J. Unger, A. Kröner, W. Marquardt. Structural analysis of differential-algebraic equation systems – theory and applications. *Computers Chem. Engng*., 19:867-882, 1995.

[27]A.C. Hindmarsh, R. Serban, A. Collier. *User Documentation for IDA v2.5.0*, UCRL-SM-208112, www.llnl.gov/casc/sundials, 2006.

