SIMULATION OF HYBRID SYSTEMS USING STATEFLOW

Anis SAHBANI, Jean-Claude PASCAL

LAAS-CNRS 7 avenue du colonel Roche 31077 Toulouse Cedex 4 France
Email: asahbani@laas.fr, jean-claude.pascal@laas.fr

KEYWORDS

Hybrid systems, Hybrid simulation, Statecharts, Stateflow, Matlab.

ABSTRACT

This paper deals with the simulation of hybrid systems. These systems mix two different aspects: continuous and discrete. Their simulation presents many problems mainly the synchronisation between the two models. Stateflow, used to describe the discrete model, is co-ordinated with Matlab, used to describe the continuous model. The gas storage example is used to illustrate the co-ordination of the two simulators.

I. INTRODUCTION

Hybrid systems combine two types of components: subsystems with discrete dynamics and subsystems with continuous dynamics that interact with each other [Antsaklis and Lemmon 98]. Particularly, in the Batch process, continuous and discrete aspects are closely linked.

This hybrid aspect is present in every level of the hierarchical decomposition of a control system: from the higher level, which is supervision, to the lower one, which is local control. The continuous and discrete aspects correspond to two different worlds, which give two different views of the system. The discrete view is not only an abstraction of the continuous one, but the two views are complementary of one environment [Andreu and al.96].

Since these two dynamics coexist and interact with each other, it is important to develop models that can reflect this coexistence. Many models have been developed or extended to specify such systems. We can mention for example Hybrid Automata [Alur and al.95], [Puri and Varayia 96], Hybrid Petri Nets [Le Bail and al.91], Differential Predicate Transition Petri Nets [Champagnat and al.98a], Hybrid Statecharts [Kesten and Pnueli 92], [Gu guen96]. One of the methods for validating such models is simulation.

Hybrid simulation is used when some of the variables are continuous and the others are discrete. It is based on the connection and interaction of two submodels. Continuous and discrete simulations progress in alternation. Continuous simulation takes care of the continuous dynamic and is executed while no event has been detected. This event can either be foreseeable (for instance, a temporal event) or unpredictable (overstepping of a threshold by some variables of the model). The numeric algorithm yields to the discrete part and waits until a stable state is reached.

The coexistence of two different aspects causes many problems in simulation. In fact, the main problem is the synchronisation between the two models.

The first part of this paper focuses on the different manners of modelling hybrid systems. The second is dedicated to the problematic of hybrid simulation. We then illustrate the approach taken through an example, and we conclude by giving some simulation results.

II. HYBRID MODELLING

Two classes of hybrid model are distinguished [Champagnat and al.98a]. The first class, known as integrated formalism, extends one of the models (discrete or continuous) in order to specify and describe the system. The second class of models co-ordinate the discrete model and the continuous one; this is the approach that we have taken. This choice is due to the fact that using a model for each component retains the specification potential of each domain. Continuous and discrete aspects correspond to two different worlds presenting two different views of a system. In this section, three models belonging to the second class are described.

II.1. Hybrid automata

Hybrid automata are a finite state enriched with a finite set of real valued variables [Alur and al.95]. At a location (discrete state), the values of the variables change continuously with time according to the associated differential equations and as long as the location invariant remains true. When the transition guard (conditions relating to the continuous variables) becomes true, the system evolves to another location.

II.2. Hybrid statecharts

The Statecharts formalism is presented by Harel in [Harel 87]. Statecharts are augmented with a notation that allows a basic state to be annotated by a differential equation. The implied meaning is that whenever the state is active, the associated differential equation is operational. Interested readers may find more details in [Kesten and Pnueli 92]. Transitions are labelled by a label which typically has the form *event/action*. The *event* triggers the transition and the optional *action* is performed when the transition is taken. In addition, an *event* can be related to a continuous variable for example, when a threshold is crossed. An *action* can also consist of an update of the continuous variable.

II.3. Differential Predicate/Transitions Petri nets

This model combines Predicate/Transitions Petri Nets and differential algebraic equations. Petri nets are used to represent the different configurations of the system, *i.e.* discrete aspects. A set of differential algebraic equations is associated to each place. The tokens carry a set of information concerning the different parameters used in the next operational differential equations and mainly the updating of some variables. The choice of token is determined by additional conditions for triggering transitions.

III. HYBRID SIMULATION

III.1. Problematic of hybrid simulation

Many problems are faced while simulating hybrid processes [Monsterman 99]. Firstly, the starting phase of the solver is repeated a certain number of times in the simulation. It is therefore important to take care of the transmission of the initial value of the variable each time the integration is restarted.

Furthermore, while it is easy to detect a temporal event, the detection of an unforeseeable event (such as an overstepping of a threshold) is non trivial. Nevertheless, once this event has been detected, the solver must be able to rollback to the time of its occurrence.

Finally, the action linked to the event can introduce a modification in the system's configuration. The subset of equations used as a model must be modified. The solver must be robust in the face of this problem. Figure 1 describes how an event defined by ($X_{threshold}$, $t_{threshold}$) is not detected in an integration step $\{(X_i , t_i) ; (X_{i+1} , t_{i+1})\}$.

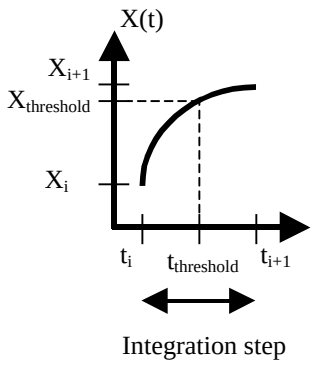

Figure 1

III.2. Stateflow: a working environment for the simulation of complex reactive systems

Stateflow is a tool integrated in the Matlab environment and used for the development and the simulation of complex reactive systems. It uses a variant of the finite state machine notation established by Harel [Harel87]. Specifically, it uses the hybrid Statecharts formalism presented previously with additional element which is introduced to complete such formalism. A Stateflow diagram is a graphical representation of a finite state machine. It provides a block that can be included in a Simulink model. Additionally, it enables the representation of hierarchy, parallelism and history. Hierarchy enables the organisation of complex systems by defining a parent/offspring object structure. A system with parallelism can have two or more orthogonal states active at the same time. History provides the means to specify the destination state of a transition based on historical information. Figure 2 describes the integration of Stateflow in the Matlab environment and its interaction with Simulink.

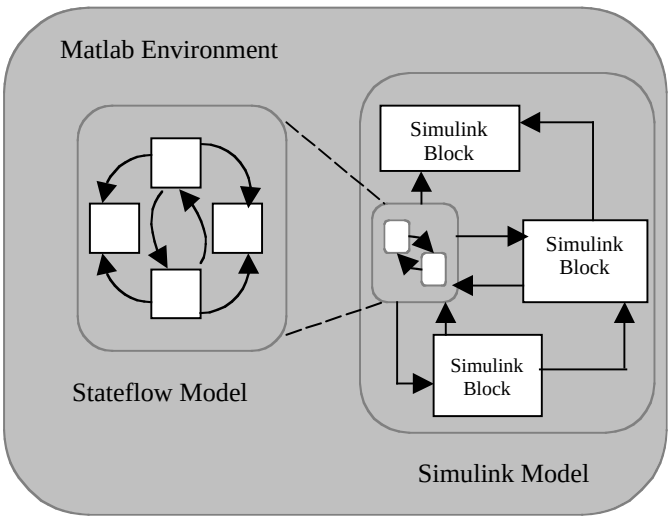

Figure 2

In the next section, we will present an example in order to illustrate the use of the hybrid Statecharts formalism. The implementation of such model in Stateflow and Matlab environment will also be described.

III.3. A gas storage example

The gas is stored between a production unit and a customer unit. The goal of the storage tank is to introduce a buffer in order to facilitate the balance between production and customer demand (Figure 3). This storage cannot be simultaneously fed and drawn [Champagnat and al.98b].

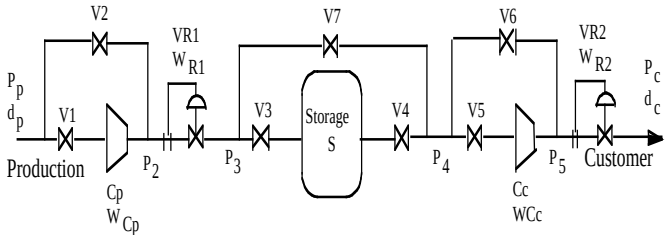

Figure 3

As a consequence of this constraint, the storage can either be by-passed (configuration 1), or the production is greater than the customer flow so the surplus is transferred to the gas storage (configuration 2). It can also be either in configuration 3, where the production flow is lower than the customer one so the lack of gas compensated by drawing the storage, or in configuration 4, where the production unit is stopped. All these configurations are presented in figure 4.

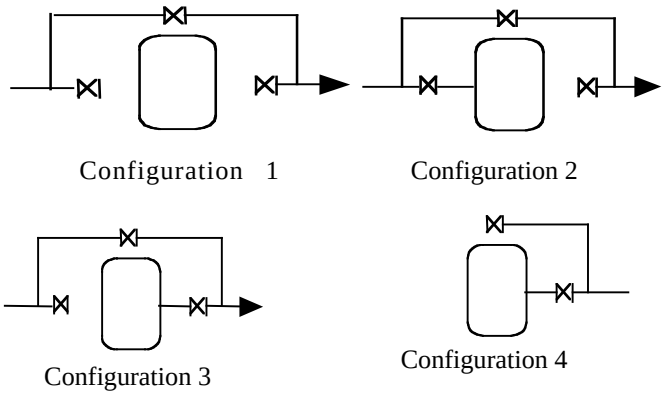

Figure 4

III.4. Gas storage modelling

In this section the storage modelling is described. The description shows the Statecharts and differential algebraic equations interacting. The gas storage has five discrete states: the four configurations presented and a breakdown state. As a consequence, the Statechart model has five states as presented in figure 5.

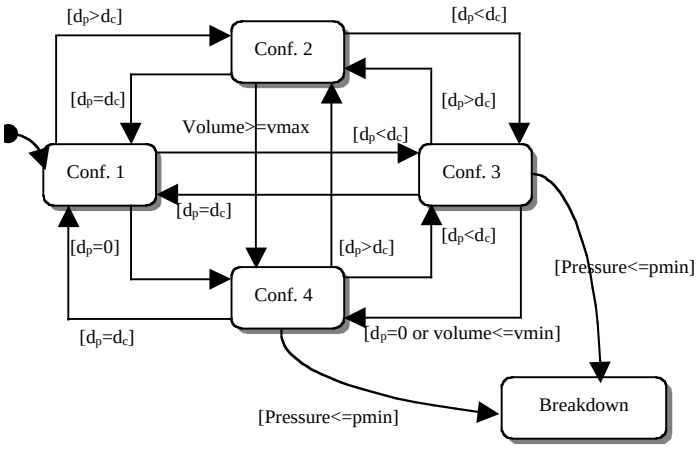

Figure 5

The purpose is to facilitate the switch between production and the customer demand. This switch depends on the debit, the pressure and the volume before and after the storage. The pressure and the volume must remain between a minimum and a maximal value.

Differential algebraic equations are associated to each state describing the physical evolution of the continuous variables. When a state is active, the equations associated are operational. The equations are:

Configuration1: $\frac{dU_S}{dt} = 0$

Configuration2:
$$\begin{cases} \frac{dU_s}{dt} = d_p - d_c \\ P_s . V_s = U_s . R . T \\ V_s = 353.6 . P_s . \left(10\left(P_s - P_0\right)\right)^{5/2} \end{cases}$$

Configuration3:
$$\begin{cases} \frac{dU_s}{dt} = d_p - d_c \\ P_s . V_s = U_s . R . T \\ V_s = 353.6 . P_s . \left(10\left(P_s - P_0\right)\right)^{5/2} \end{cases}$$

Configuration4:
$$\begin{cases} \frac{dU_S}{dt} = -d_C \\ P_s . V_s = U_s . R . T \\ V_s = 353.6 . P_s . \left(10\left(P_s - P_0\right)\right)^{5/2} \end{cases}$$

III.5. Simulink and Stateflow models in the Matlab environment

The discrete part is developed using Stateflow. As shown in figure 6, the same Statechart diagram is reproduced. When a state is activated, the differential algebraic equations associated become operational and they are resolved using a Matlab function. The synchronisation between the activation of a state and the solving of the associated equations is assured by means of the *entry* function. Some parameters of the solving function assure the updating of the variables used in the equations associated to the active state (ml.volume, ml.pressure).

The first parameter of the solving function indicates the file in which are introduced the equations used in this step. The remaining parameters define some of the thresholds which must be respected. The $conf_i$ variables represent the activation and the disactivation of a state. They are defined as outputs to the Simulink model. When breakdown state becomes active the simulation is stopped (stop=1).

The customer and the production flows (D_c and D_p) are represented in the Simulink model by means of *sine waves* with *steps*. *Steps* are used only to have positive debits. These flows are defined in Stateflow model as input from Simulink. Figure 7 represents the Simulink model.

The differential algebraic equations are solved using a function developed in Matlab. This function uses a Matlab function, which needs a fixed interval of integration and gives vectors of points every step. The initial solving interval is fixed according to the dynamic of the system. At each step, the last values of the given vectors are compared with the thresholds. Two cases are possible: either the threshold isn't reached then the simulation interval is translated and another integration is launched, or the threshold is exceeded then the vector is saved. In this case, each point of this vector is compared with the threshold and only the values below the threshold are saved. Another integration can be launched with a simulation interval corresponding to the critical points (just before and just after an overstepping of a threshold) in order to obtain better precision.

In this implementation, the differential algebraic equations are simplified due to the limit of the Matlab function used in our solver. Indeed, the object of this paper is not to develop a sophisticated solver but to show how we can co-ordinate two different simulators.

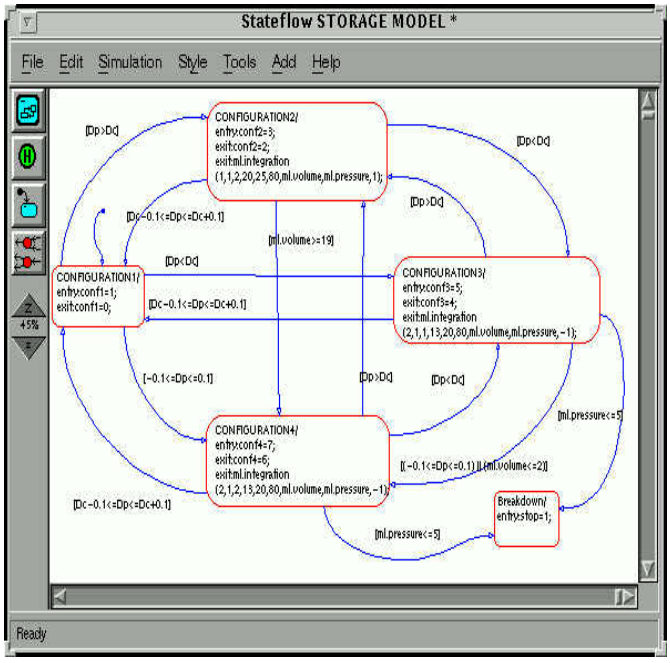

Figure 6

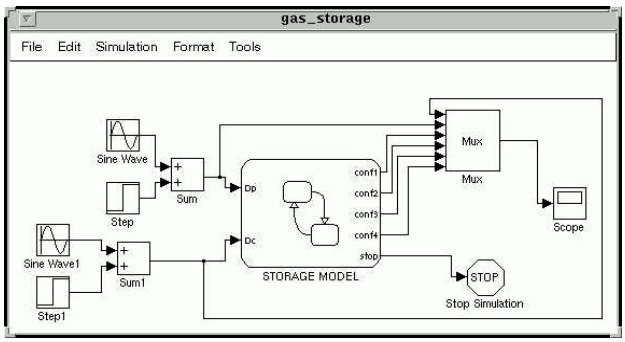

Figure 7

III.6. Simulation results

While simulating hybrid systems, problems, which are previously described, are faced. The first one, concerning the initial value of some variables, is solved by broadcasting these values when their relative state is activated. The final value of the shared variables in the last integration are stored and transmitted as initial ones when an triggering transition event happen. Furthermore, a temporal event is detected, as described previously, by comparing all the value stored while one simulation step with the threshold. The solver rolls back to the time of its occurrence. Another integration can be launched to obtain better precision around the threshold.

It might appear that the use of a Matlab function could be avoided by introducing the differential algebraic equations using the Simulink model. However, this approach would not preserve the separation between the continuous and the discrete aspects, which be mixed in a single model. In other terms, switch blocks must be introduced to transmit initial values of some variables and to toggle between equations systems.

Figure 8 and 9 show respectively the evolution of volume and pressure. We remark that the two variables have never overstepped the threshold fixed. Figure 10 shows the evolution of two continuous variables, which are the customer and the production flows, the other variables are relative to the activation of the state.

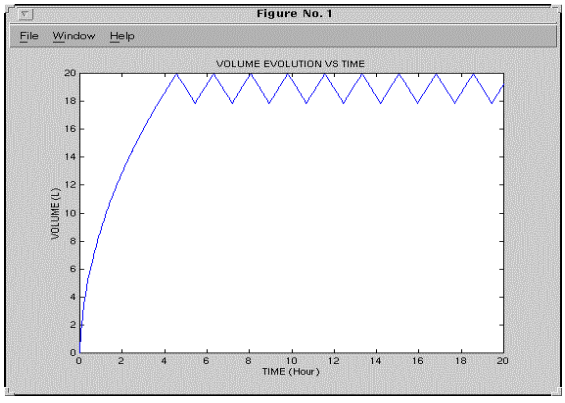

Figure 8

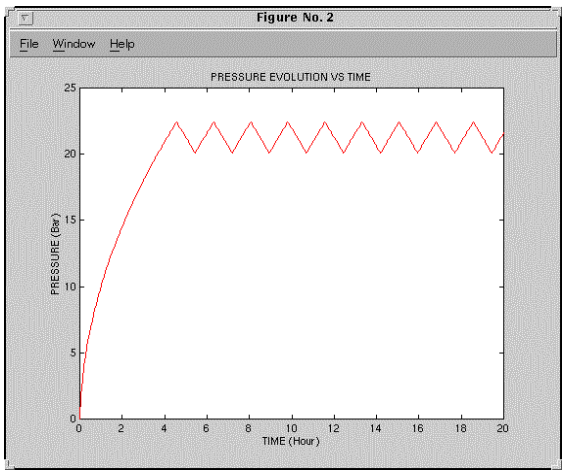

Figure 9

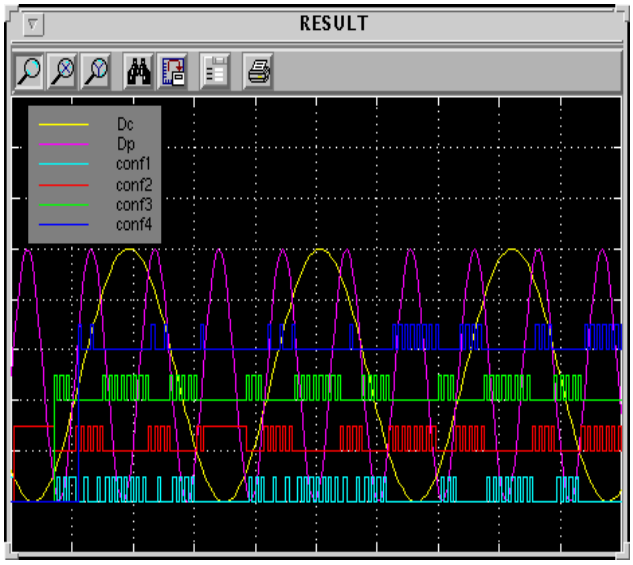

Figure 10

IV. CONCLUSION

In this paper, the gas storage process has been modelled using Hybrid Statecharts and simulated using Stateflow, a Matlab toolbox. The use of specific models for each aspect (discrete and continuous aspects) allows the preservation of the modelling potential of each domain. Matlab, as a continuous simulator reference, offers its powerful modelling of the continuous part of hybrid system. Stateflow specifies the discrete behaviour of hybrid processes using the Statechart formalism. The latter lacks the modelling potential given by the Petri Nets. Nevertheless, for sequential systems where resource sharing is limited, the use of Stateflow for simulating hybrid systems can present a great relevance. The integration of two simulators avoids the development of an interface between the discrete and the continuous simulator.

Further works are concerned with coupling Matlab and a discrete simulator based on Petri Nets in order to have a better description for the discrete model.

Other works are also concerned with the development of a performed solver in order to cover the lack the one used to simulate our example.

REFERENCES

[Alur and al.95] R. Alur, C. Courcoubetis, N. Halbawchs, T.A. Henzinger, P.H. Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine. The Algorithmic Analysis of Hybrid Systems. *Theoretical Computer Science*, 138:3-34, 1995.

[Andreu and al.96] D. Andreu, J.C; Pascal and R. Valette. Events as a Key of Batch Process Control System. *CESA'96 IEEE-SMC IMACS Symp. On Discrete Events and Manufacturing Systems*, p. 297-302. Lille, France, 1996.

[Antsaklis and Lemmon 98] P. Antsaklis and M. Lemmon. *Discrete Event Dynamic Systems : Theory and Applications*, 8, 101-103. 1998 Kluwer Academic Publishers, Boston.

[Champagnat and al.98a] R. Champagnat, P. Esteban, H. Pingaud, R. Valette. Modelling and Simulation of a Hybrid System through Pr/Tr PN-DAE Model. *ADPM'98 conf. On Automation of Mixed Process : Dynamical systems*. Reims, France, 1998.

[Champagnat and al.98b] R. Champagnat, H. Pingaud, H. ALLA, C. Roubinet and J.M. Flaus. A Gas Storage Example as a Benchmark for Hybrid Modelling. *ADPM'98 conf. On Automation of Mixed Process : Dynamical systems*. Reims, France, 1998.

[Gu guen 96] H. Gu guen. Use of Statecharts and Signal for the Specification of Control of Hybrid Systems. *CESA'96 IEEE-SMC IMACS Symp. On Discrete Events and Manufacturing Systems*. Lille, France, 1996.

[Harel 87] D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Sci. Comp. Prog.*,8: 231-274, 1987.

[Kesten and Pnueli 92] Y. Kesten and A. Pnueli. Timed and Hybrid Statecharts and their Textual Representation. *LNCS: Formal Techniques in Real Time and Fault Tolerant Systems*, Nijmegen 1992.

[Le Bail and al.91] J. Le Bail, H. Alla and R. David. Hybrid Petri Nets. *European Control Conference*, pp. 1471-1477, Grenoble, France, 1991.

[Mosterman 99] P. J. Monterman. An Overview of Hybrid Simulation Phenomena and Their Support by Simulation Packages.

[Puri and Varaiya 96] A. Puri and P. Varaiya. Verification of Hybrid Systems using Abstraction. *IFAC: 13th Triennial World Congress*, San Fransisco, USA 1996.