Systems Analysis Modelling Simulation, 2002, Vol. 42 (5), pp. 695-712

Process modelling for hybrid control

G. Mušič[*], M. Šikovc[†], B. Zupančič[*]

Abstract

Paper deals with modelling methods for the purpose of control of hybrid systems, a class of dynamic systems where the behaviour of interest is determined by interacting time continuous and discrete-event dynamics. Different modelling approaches are discussed and compared. Then the model with the interface is selected and illustrated by an example of a semi-industrial scale hydraulic pilot plant. A discrete-event plant model is developed and a discrete controller is designed to prevent entering into a forbidden region of the state space. The designed hybrid control system is evaluated through several simulation runs from different initial conditions.

Key words: Mathematical modelling, hybrid systems, control.

1 Introduction

The term hybrid systems may be used to designate a class of systems that exhibit a dual or multiple nature. Recently this term is mostly used to characterise a class of dynamical systems where the behaviour of interest is determined by interacting time continuous dynamics and dynamics of a discrete-event type. Such systems often arise in automatic control of various technological processes.

The hybrid nature of these systems may result from the changes in dynamics of the predominantly continuous process itself, e.g., due to phase changes, changes from laminar to turbulent flow in a pipe, discontinuities in the equipment geometry, etc. [3]. These changes may be treated as discrete events that are part of the process behaviour. Another important source of discrete events that influence the continuous dynamics are discretely operating control elements, such as

[*]University of Ljubljana, Faculty of Electrical Engineering, {gasper.music, borut.zupancic}@fe.uni-lj.si
[†]Iskra Sysen d.d., Ljubljana, matej.sikovc@sypra.si

on/off valves, automatic protective devices, etc. Finally, almost every control system involves digital computers that interact with continuous environment. While digital controllers that govern the process may often be considered as approximate implementation of continuous control laws, several control functions that are required in today's control systems can not be treated this way. These are, e.g., automatic operational and safety procedures in industrial processes that are implemented by programmable logic controllers. Complex systems often possess a hierarchical structure with predominantly continuous dynamics at the lowest level and logical decision-making at the highest. An example is the interaction of discrete planning algorithms and continuous processes in autonomous and intelligent control. Logical decisions in such a system typically take place only when certain events occur in the process and result in the change of dynamics at the lower level. Thus, almost all control systems today involve both continuous and discrete-event dynamics, which explains the motivation for the substantial research effort that has been dedicated to hybrid systems in the recent period.

In design of hybrid control systems an initial step is to develop an appropriate mathematical model that incorporates all the relevant process behaviour. While the continuous dynamics is typically described by a system of ordinary differential equations (ODE) or differential algebraic equations (DAE), the discrete-event dynamics is typically modelled as a kind of discrete state-transition structure, e.g., automata or Petri nets. To model hybrid behaviour a methodology is needed, which can bridge the gap between the two inherently different modelling approaches.

The field of hybrid modelling lacks a common modelling paradigm. Several types of models exist, which emerged from two different fields (control engineering and computer science) and with different perspectives (modelling, analysis, verification, synthesis). The computer science, e.g., deals with hybrid automata [1], which are used in verification procedures of real time programs. The control engineering commonly uses models that enable algorithmic synthesis of related control strategies [2, 4, 5]. In general, three main paradigms of dealing with hybrid modelling can be identified in the literature. First is the endeavour for incorporation of the discrete actions in the classical differential equation models. This can be done by either including discrete phenomena as nonlinearities into the continuous dynamics or treating these phenomena as disturbances or unmodelled dynamics. Both these ways have a major drawback that certain assumptions have to be made and specific solutions have to be sought for different types of systems.

Second modelling paradigm deals with the abstraction of continuous dynamics in a way that the overall system may be described by a pure discrete-event model. Such a model may then be used to design a discrete-event controller that performs a high level, e.g., supervisory control [6, 11]. Major problems of this approach are related to non-determinism that is inherent in resulting discrete-event models.

The third modelling paradigm is a truly hybrid paradigm where both continuous and discrete-event dynamic properties and their interactions are captured within a unified modelling approach [4, 5]. The controller synthesis is performed either in optimal control setting [5] or by transformation of control demands into a mixed integer linear (or quadratic) programming problem [4]. This last modelling paradigm is certainly the most promising and covers most general type of systems that may be found related to hybrid control. Nevertheless, the system description is rather abstract and so are the proposed control synthesis methods. The results seem to be difficult to implement with conventional control equipment. For these reasons we will focus more on the second, i.e., discrete-event modelling paradigm in this paper.

2 Discrete-event models of continuous processes

These models are useful when hybrid control can be viewed as a discrete event controller that is applied to a continuous plant. The design of such a controller requires a pure discrete-event model of the process being controlled. This approach to hybrid modelling promises a considerable reduction in the complexity of model analysis and controller synthesis tasks. Models can be analysed by well developed techniques, and control can be synthesised in a formal way. Another advantage of this approach is that the resulting controller can be relatively easily implemented by the conventional control equipment, e.g., a programmable logic controller.

Discrete-event models of continuous processes are based on partitioning the state space of continuous models by hypersurfaces. In this way a discrete abstraction of a continuous system is obtained, which needs to be detailed enough to enable the synthesis of an appropriate control strategy. The decision of how detailed a discrete abstraction should be is related to the fundamental question of how much information is needed by the controller to attain particular control goals [6]. This has led to different approaches to discrete-event modelling of continuous processes; some of them are briefly discussed in this section.

Much of the research related to these models has been dedicated to investigation under what conditions on the continuous system and on the state space partitioning the resulting discrete-event models are deterministic. In [7] a linear, autonomous model of the continuous plant is considered and it is assumed that only quantised state measurements are available. The main characteristic of the discrete model is that the partition of the state space is predetermined by sensor quantisations. Relation between linear continuous-variable system together with the quantiser and deterministic automaton as representation of the quantised system is studied and sufficient conditions for deterministic behaviour of the discrete-event plant model are derived. It is shown that nearly all systems have a nondeterministic behaviour. An arbitrary partitioning of signal spaces is also studied and partitionings, which yield deterministic behaviour of the discrete-event plant model, are identified.

A similar model where only quantised plant measurements are available is also studied in [10]. In addition, the plant is affected by unknown but bounded disturbance. Instead of a single abstraction a set of abstractions is proposed, where the abstractions are ordered in the sense of approximation accuracy. The increased approximation accuracy is achieved by including past measurement and control signals into the definition of a discrete state. This enables the possibility to adjust the approximation accuracy to various specifications without changing the measurement quantisation level. The approach is set in discrete-time with fixed sampling intervals, so the time needed for a transition is retained in the discrete approximation. A similar approach is reported in [9]. The approximations used there are purely logical, which means that timing information is not retained in the model, except for the temporal order of events.

Another approach based on the automata theory is presented in [8]. A hybrid system is considered, which consists of a continuous plant, subject to disturbances, interacting with a sequential automaton - the controller. The plant and the controller are represented by interacting non-deterministic sequential automata, which operate on infinite input and output alphabets. The control automaton and the plant interact at discrete times, where the interval between successive interaction times can vary. To facilitate analysis, several specialisations of the model are imposed. Next, the control automaton is modelled in more detail and decomposed into an analog to digital converter, the internal control automaton and a digital to analog converter. The internal control

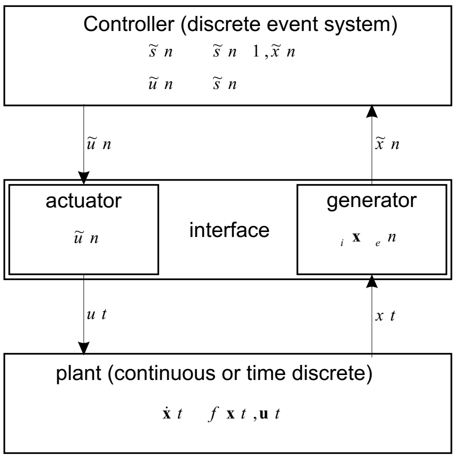
Controller (discrete event system)
$\tilde{s}\ n \quad \tilde{s}\ n\ 1, \tilde{x}\ n$
$\tilde{u}\ n \quad \tilde{s}\ n$
$\tilde{u}\ n$
$\tilde{x}\ n$
actuator
$\tilde{u}\ n$
interface
generator
$_i\ \mathbf{x}\ _e\ n$
$u\ t$
$x\ t$
plant (continuous or time discrete)
$\dot{\mathbf{x}}\ t \quad f\ \mathbf{x}\ t, \mathbf{u}\ t$

Figure 1: Hybrid system model with an interface

automaton is a finite state automaton with finite input and finite output alphabets. A discussion on how to associate a sequential non-deterministic automaton, a plant automaton, with a continuous plant and how to connect it to the control automaton is given. The notion of controllability and observability for hybrid systems is formalised.

A direct synthesis of the hybrid control is possible by the models with the interface [2, 6, 12]. The model comprises a time continuous process described by state space equations, a discrete event controller and an interface, which consists of a generator and an actuator (Figure 1). The *plant* is a part of the system that incorporates all the continuous dynamic of the hybrid system. It is designated by a continuous state space where the changing of the variables over time is described by differential equations. The plant can include continuous control mechanisms. The *controller* is a discrete event system that is modelled as a deterministic automaton. Input and output of the controller are symbol sequences. The automaton generates a set of control symbols while the plant symbols are generated by the occurrence of events in the plant. The *interface* between the plant and the controller has the role of converting time continuous signals into symbol sequences and vice versa. It is split up into a generator and an actuator. The *generator* is a subsystem of the interface that transforms the continuous signal into an asynchronous symbolic input of the controller. Symbols are generated at occurrences of events in the plant, i.e., when the trajectory of the plant crosses one of the hypersurfaces that divide the state space of the continuous plant. The *actuator* transforms a sequence of control symbols into an input signal that is fed into the plant. The input to the plant is piecewise constant, because it depends on a particular control symbol. The continuous plant and the interface are taken together to form a new dynamic system. According to Figure 1 the inputs and outputs of the new system are symbols (events). Such a system can be described as a discrete-event dynamic system and a corresponding description represents a *discrete-event model of a plant* (DES plant model). An important property of the obtained model is that the DES plant is non-deterministic.

When designing hybrid control based on the models with the interface the main issue is how to avoid or reduce this nondeterminism what would cause the system to become controllable. The solution is in proper partitioning of the state space. Once this is achieved, the design of the controller is rather straightforward. Therefore a substantial part of the design process is dedicated to the design of the interface [6].

3 A case study

The model of a hybrid system with an interface has been studied through a case study of hybrid control design for a laboratory hydraulic pilot plant. The plant used in the study (Figure 2) consists of two interconnected vessels that are filled by two water pumps. The outlets of both vessels are fed into a common reservoir. In the presented case the manual valve on the interconnecting pipe between the vessels is opened, its cross-section is $A_{12} = 0.8cm^2$. The outlet of each vessel is controlled by an on/off valve with the cross-section $A_{iz1} = A_{iz2} = 2cm^2$. The vessels are filled by the pumps with the nominal flow of $\phi_1 = \phi_2 = 1l/s$. Both vessels are equipped by continuous level sensors. The maximum allowed level is $1.25m$. The cross-section of each vessel is $A_1 = A_2 = 0.18m^2$.

The plant has four inputs, i.e., the two pumps and the two outlet valves. All inputs are binary, meaning a pump may be switched on/off and a valve may be open/closed. Inputs are denoted by binary variables u_1, u_2, u_3 and u_4 where u_1 and u_2 are the control signals on the first and the second pump, respectively, and u_3 and u_4 are the control signals on the outlet valve of the first and the second vessel, respectively. States of the plant, denoted by x_1 and x_2, correspond to the water level in the two vessels. The plant model is based on the mass balance (expressed through volumes) for the vessels. When the plant input signals are included into mass balance equations together

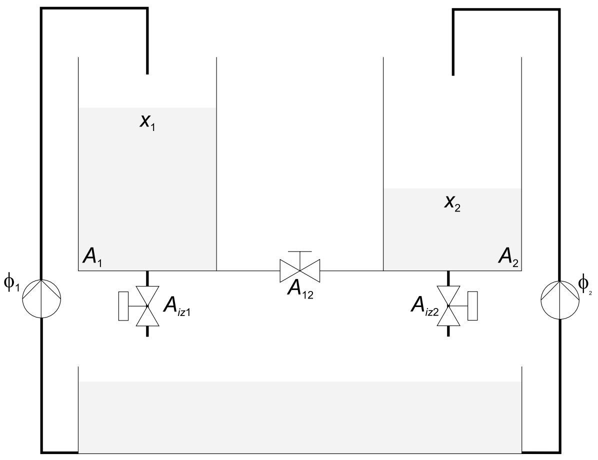

Figure 2: Schematic representation of the hydraulic pilot plant

with values of the constants the following state equations are obtained (units are omitted):

$$\begin{aligned}\dot{x}_1 &= 5.5 \cdot 10^{-3} \cdot u_1 - 5 \cdot 10^{-3}\sqrt{x_1} \cdot u_3 - 2 \cdot 10^{-3}\mathrm{sign}\left(x_1 - x_2\right)\sqrt{|x_1 - x_2|} \\ \dot{x}_2 &= 5.5 \cdot 10^{-3} \cdot u_2 - 5 \cdot 10^{-3}\sqrt{x_2} \cdot u_4 + 2 \cdot 10^{-3}\mathrm{sign}\left(x_1 - x_2\right)\sqrt{|x_1 - x_2|}\end{aligned} \tag{1}$$

The demands for the control of the plant are as follows. The initial state of the vessels is empty. Then the first vessel has to be filled at least up to $1m$ while the level in the second vessel must not exceed $0.2m$ during that period. Then the second vessel is filled up to at least $1m$ while the level of the first vessel must not fall below $1m$ in the mean time. After both vessels are filled starts the emptying. The first vessel is emptied down to $0.2m$ while the level in the second vessel must not fall below $1m$. Then the second vessel is emptied down to $0.2m$ while the level in the first vessel must not exceed $0.2m$. For the overall process the limitation of maximum level at $1.25m$ has to be observed. The given demands result in forbidden regions in the phase plane and also in the prescribed direction of the plant trajectories. The control demands must be met by switching the four inputs between binary values 0 and 1. The four binary inputs result in altogether sixteen possible input combinations that are numbered from 1 to 16 according to the decade equivalent (incremented by 1) of the binary input combination (e.g., input no. 7 means: $u_1 = 0$, $u_2 = 1$, $u_3 = 1$, $u_4 = 0$).

The control is designed in accordance with the procedure described in [12]. The first step is to merge the plant and the interface into a DES plant model. The continuous plant is described by Equations (1), now the description of the interface is needed. The design of the actuator is straightforward; there are obviously sixteen different control symbols that are transformed into states of the individual plant inputs. The generator is much more complex and its design presents the main problem of the approach. The generator is defined by a set of hypersurfaces, which partition the plant's state space into disjoint regions. Hypersurfaces are determined by a set of smooth functionals $\{h_i : \Re^n \rightarrow \Re, i \in I\}$ defined on the state space of the plant. Null space of the functional $\mathcal{N}(h_i) = \{\xi \in \Re^n : h_i(\xi) = 0\}$, forms a $n-1$ dimensional hypersurface separating the state space [12].

The primary partition of the state space is usually driven by the control demands and limitations in the system. In the given case we use six functionals defined by the following equations:

$$\begin{aligned} h_1(\mathbf{x}) &= x_2 - 0.2 & h_4(\mathbf{x}) &= x_1 - 0.2 \\ h_2(\mathbf{x}) &= x_2 - 1 & h_5(\mathbf{x}) &= x_1 - 1 \\ h_3(\mathbf{x}) &= x_2 - 1.25 & h_6(\mathbf{x}) &= x_1 - 1.25 \end{aligned} \qquad (2)$$

The null spaces of the functionals define a set of straight lines that form the boundaries between state space regions. In the phase plane of variables x_1 and x_2 this results in sixteen regions. If all the regions outside the boundary 1.25 are merged into one this does not influence the control while the number of regions is reduced to ten. Regions that would indicate a negative level of the water are neglected. The ten regions in the phase plane of the plant are shown in Figure 3. After partitioning the state space it is necessary to determine possible transitions from the particular region at each input combination. The calculation is performed for the null spaces of the functionals (boundaries) on their whole length.

According to [12], the transition direction between two adjacent regions is determined by the sign of the gradient of the functional defining the boundary between the two regions, the sign of the functional within regions, and the sign of the state derivative $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ on the boundary. Let X_1 and X_2 represent two sets of states belonging to the two adjacent regions of the state space. Let $\xi \in \mathcal{N}(h_i)$ denote a point on the boundary between the two regions. The transition from X_1 to X_2 exists when

$$\begin{aligned}\nabla_{\mathbf{x}} h_i(\xi) \cdot f(\xi, u(t)) < 0, \quad \text{if} \quad h_i(\mathbf{x} \in X_1) > 0 \\ \nabla_{\mathbf{x}} h_i(\xi) \cdot f(\xi, u(t)) > 0, \quad \text{if} \quad h_i(\mathbf{x} \in X_1) < 0\end{aligned} \tag{3}$$

The above condition can be explained as follows. If the current state lies near the boundary on its positive side ($h_i(\mathbf{x}) > 0$), the transition would mean the change of the sign of $h_i(\mathbf{x})$ to negative. This is possible if the sign of the gradient does not match the sign of the time derivative or, looking in one dimension: either the state lies below the boundary ($h_i(x) > 0$ then implies $\frac{\partial h_i}{\partial x} < 0$) and is increasing ($\dot{x} > 0$) or lies above the boundary ($\frac{\partial h_i}{\partial x} > 0$) and is decreasing ($\dot{x} < 0$). If the current state lies on the negative side of the boundary ($h_i(\mathbf{x}) < 0$), the situation is reversed.

Let us illustrate this by a calculation of transition directions over the null space of the functional $h_5(\mathbf{x})$. Following the above discussion the gradient of the functional on its null space (at $\mathbf{x} = \xi$) is calculated first, which gives:

$$\nabla_{\mathbf{x}} h_5(\xi) = \left. \begin{bmatrix} \frac{\partial h_5(\mathbf{x})}{\partial x_1} & \frac{\partial h_5(\mathbf{x})}{\partial x_2} \end{bmatrix}^T \right|_{\mathbf{x}=\xi} = \begin{bmatrix} 1 & 0 \end{bmatrix}^T \tag{4}$$

Obviously, only the state x_1 need to be observed. Direction of the transitions depends on the sign of the functional and the time derivative, which can be determined from the first state equation (for the state x_1). Functional $h_5(\mathbf{x})$ is negative for $x_1 < 1$ and positive for $x_1 > 1$, while its gradient on the boundary is positive. This implies that for the state vector $\mathbf{x}$ values at which the first state equation is positive the system trajectories cross the boundary from left to right and vice versa. At this point the plant inputs must be taken into account, since they directly influence the sign of the state derivative. In our case, only the inputs u_1 in u_3 are included into the state equation, so the sign of the state derivative must be checked for the four possible input combinations (input symbols). For these input combinations the sign of the function

$$f_1(\mathbf{x}, \mathbf{u}) = \left. 5.5 \cdot 10^{-3} \cdot u_1 - 5 \cdot 10^{-3} \sqrt{x_1} \cdot u_3 - 2 \cdot 10^{-3} sign\,(x_1 - x_2) \sqrt{|x_1 - x_2|} \right|_{x_1=1} \tag{5}$$

is studied and the following results are obtained:

$$\begin{aligned}
&u_1 = 0, u_3 = 0 : f_1(\mathbf{x}, \mathbf{u}) \begin{cases} < 0; & x_2 < 1 \\ > 0; & x_2 > 1 \end{cases} \\
&u_1 = 0, u_3 = 1 : f_1(\mathbf{x}, \mathbf{u}) < 0 \\
&u_1 = 1, u_3 = 0 : f_1(\mathbf{x}, \mathbf{u}) > 0 \\
&u_1 = 1, u_3 = 1 : f_1(\mathbf{x}, \mathbf{u}) \begin{cases} < 0; & x_2 < 0.9375 \\ > 0; & x_2 > 0.9375 \end{cases}
\end{aligned} \tag{6}$$

It can be seen that at the first and fourth combination of the input signals there is a left to right transition in a part of the boundary while there is a reversed transition direction in the other part.

A similar calculation has to be performed for all boundaries that partition the state space and all the transition directions have to be determined. Every region of the state space is then mapped into a discrete state and possible transitions between regions are inserted as state transitions. In this way a DES plant model is obtained, which is shown as a finite automaton in Figure 4. The convention for labeling the arcs is to list the control symbols, which enable a transition followed by a "/" and then the plant symbols, which can be generated by the transition. The subscript of a control symbol reflects the corresponding plant input combination and a plant symbol subscript reflects the boundary that was crossed when the symbol was generated and is related to the corresponding functional.

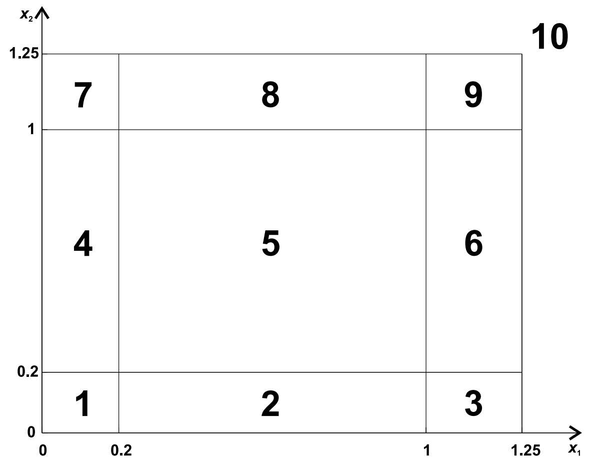

Figure 3: Regions in the continuous state space

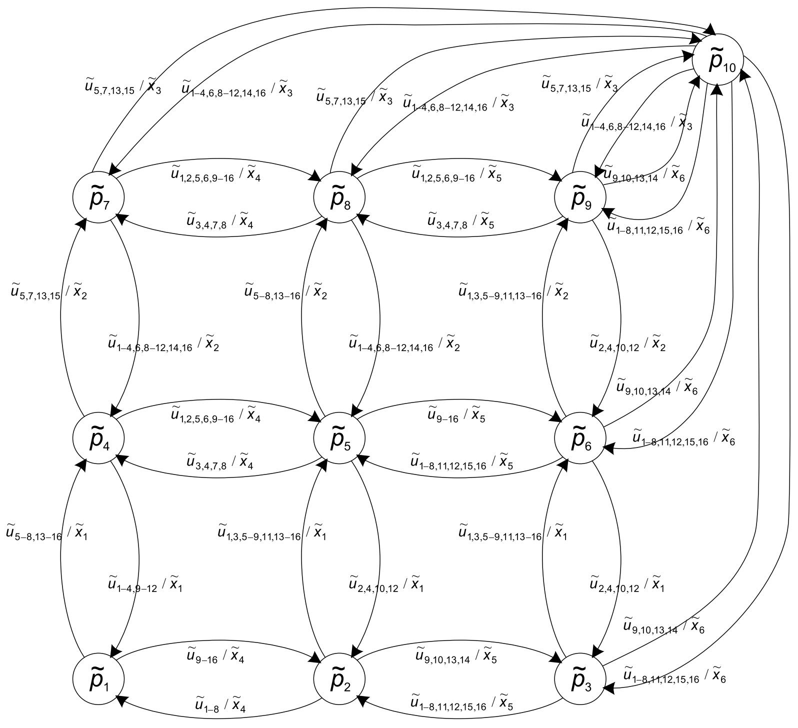

Figure 4: Discrete event system plant model for the primary partition of state space

In the next step the obtained automaton is checked for determinism, i.e., if the subsequent state can always be determined uniquely from the current state and input (control symbol). Determinism is a desired property because it allows the controller to drive the system through any desired sequence of discrete states (if such a sequence exists in the system). In our case the obtained model is not deterministic, which can be seen quite easily. E.g., the possible transition from the state $\tilde{p}_3$ at input symbol $\tilde{u}_9$ ($u_1 = 1, u_2 = 0, u_3 = 0, u_4 = 0$) can lead either to state $\tilde{p}_6$ or $\tilde{p}_{10}$. Similar observations actually hold for the majority of combinations of the states and input symbols in the given automaton. Therefore the next step is to additionally partition the obtained regions of the state space to ensure the derived DES model is deterministic or quasideterministic [2, 12]. The quasideterminism was introduced as a weaker form of determinism because generally, given a continuous plant, it may be difficult or even impossible to develop a deterministic DES model. The model is said to be quasideterministic if the subsequent state can be uniquely determined from the current state, the previous control symbol and the previous plant symbol. Such a model enables the controller to drive the plant deterministically with respect to the primary partition, i.e., to meet the desired control goal. In some cases, when only certain specific sequences of controller symbols are allowed in the system, the requirement on the quasideterminism can be further relaxed because depending on the state, only the allowed control symbols must be taken into account. This allows to use a coarser partition which is still adequate for the given control specification.

Due to the specified direction of the plant trajectories over the phase plane, the relaxed requirements on the determinism/quasideterminism can be applied also in our case. So the regions are further partitioned only in correspondence to those input combinations, which drive the plant in the desired direction. The procedure starts by examining the region 1. Input symbols $\tilde{u}_9$, $\tilde{u}_{10}$, $\tilde{u}_{11}$, $\tilde{u}_{12}$ all lead the plant deterministically to region 2 which is also the desired direction. If we restrict the allowed set of control symbols in the corresponding DES plant state to $\{\tilde{u}_9, \tilde{u}_{10}, \tilde{u}_{11}, \tilde{u}_{12}\}$ no additional partitioning of this region is needed. Similarly, we can find input symbol $\tilde{u}_{10}$ which drives the plant from region 2 to region 3 and therefore, region 2 does not need additional partitioning. This is not the case in region 3 where no input symbol could guarantee the required transition into region 6. The region 3 has to be split up in such a way that one of the obtained sub regions would enable a deterministic transition into region 6 under a restricted set of control symbols. Input symbol $\tilde{u}_{15}$ has been chosen and a boundary between states that lead to region 6 under the chosen control symbol (sub region 32) and states that lead to other regions (sub region 31) has been calculated. The boundary matches the system trajectory that crosses the point $(1, 0.2)$ in the phase plane. Because the system is nonlinear, the calculation of system trajectories is not straightforward and a numerical solution has been used in this case. Similar calculation has been performed for the region 6. The same control symbol was chosen and a boundary was set on the system trajectory leading through the point (1,0.9375). The calculated trajectory splits region 3 as well and therefore could be used as a boundary also in the region 3 as to minimize the switching of the plant inputs. Since the new boundary lies entirely in the previously calculated sub region 32, the previous partitioning of the region 3 is dropped and new one used instead. The remaining regions are partitioned in the same way, minimizing the plant input switching when possible. The new partition of the plant's state space obtained in this way is shown in Figure 5. This additional partition results in a new DES plant model. The corresponding automaton is relatively large and is shown in Figure 6; it consists of 18 states. The new DES plant is still non-deterministic but can be controlled to achieve the desired goal. It can be shown the model is observable, i.e., the current

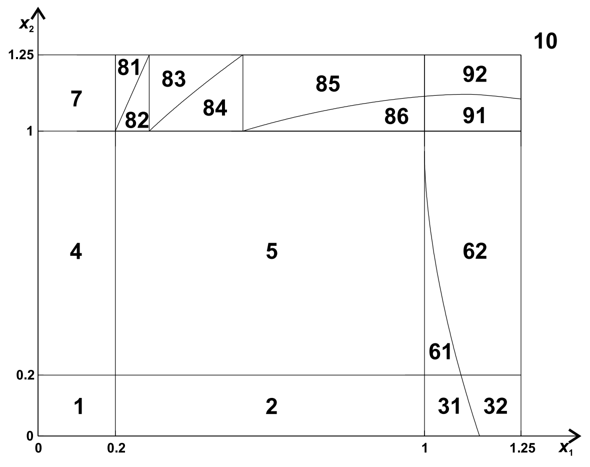

Figure 5: Refined partition of the state space

state can be determined from the initial state and the sequence of transition over the boundaries between the regions.

The desired control can be described by a regular language that is given by a regular expression [11] if we assume the initial and final state reside in the first region:

$$K = \overline{(x_4 x_5 x_7 x_1 x_2 x_8 x_5 x_9 x_{10} x_{11} x_{12} x_4 x_2 x_1)^*} \tag{7}$$

The language K is controllable [11, 12] because a suitable input can be found for any string in the language so that the addition of the corresponding new symbol to the string keeps the string within the language. Strings in the given language denote a start in region 1, then transition to region 2 (symbol x_4) and further transitions over regions 31, 32, 62, 91, 92, 85, 84, 83, 82, 81, 7, 4 and back to 1. Regions 61 and 86 are allowed but were not used above so two corresponding states are added to the automaton controller and defined such that they lead to states 62 and 85, respectively. States with the same output symbol can be merged and controller simplified. The final discrete-event controller for the given case is shown in Figure 7.

The results of the control design for the hydraulic laboratory plant are the interface and the controller. The behaviour of the complete system can be tested by simulation within Matlab using additional tools Simulink and Stateflow. Simulink is used for the simulation of continuous parts of the system while discrete event parts can be simulated by Stateflow. The continuous part consists of mathematical model of the plant and related functionals that are either defined analytically or

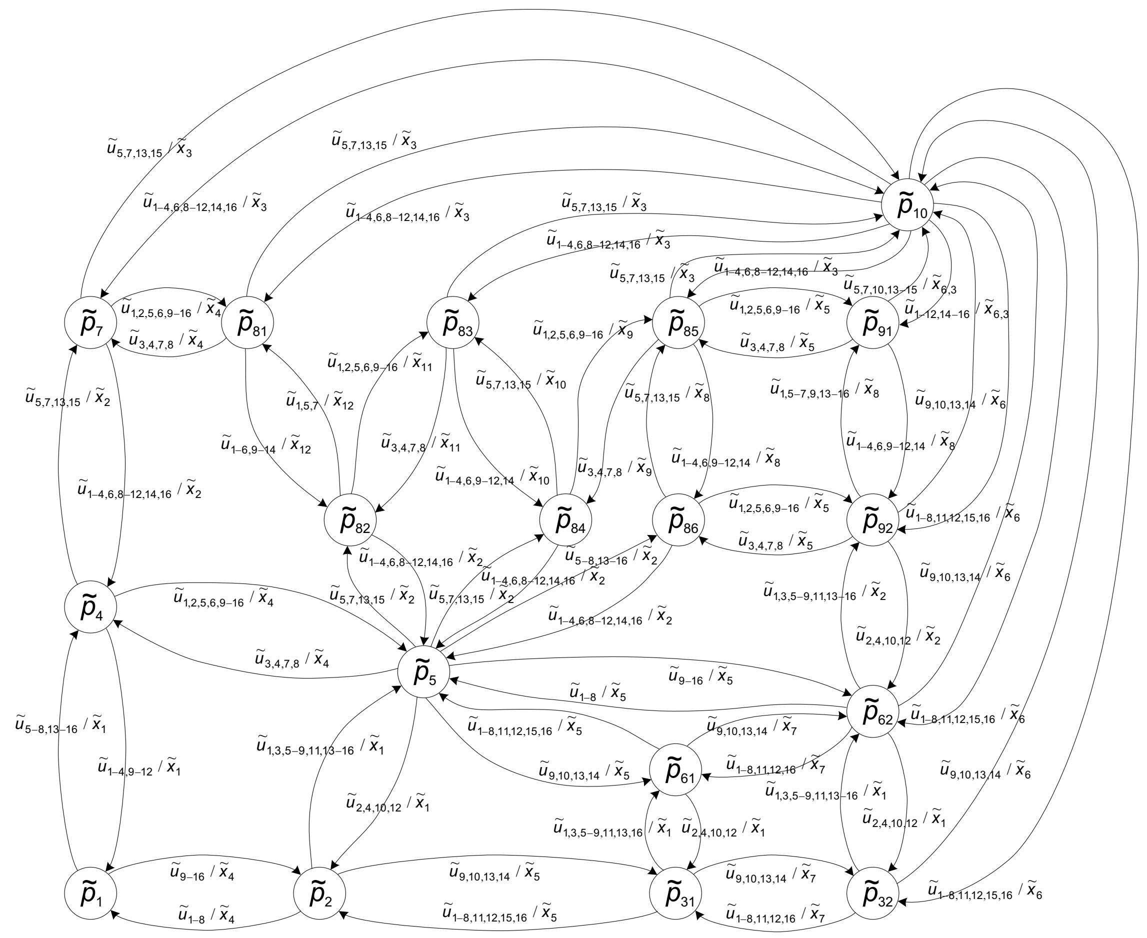
Figure 6: DES plant model based on refined partition

via look up tables. The discrete part of the system is the controller; both parts are connected through built in facilities of the used tools. Several simulation results from different initial states indicate the system behaves as required. A sample trajectory of the system is shown in Figure 8 while related outputs of the process and the corresponding inputs are shown in Figure 9.

4 Conclusions

Modelling and control methods for hybrid systems have been more intensively studied only for the past few years. Several methods emerged but most of them are typically limited to certain class of systems and appropriate either only for analysis, verification or control.

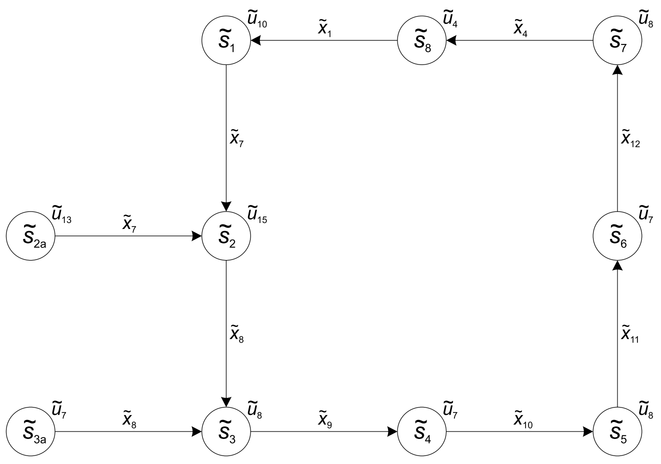

Figure 7: Discrete-event controller for the hydraulic pilot plant

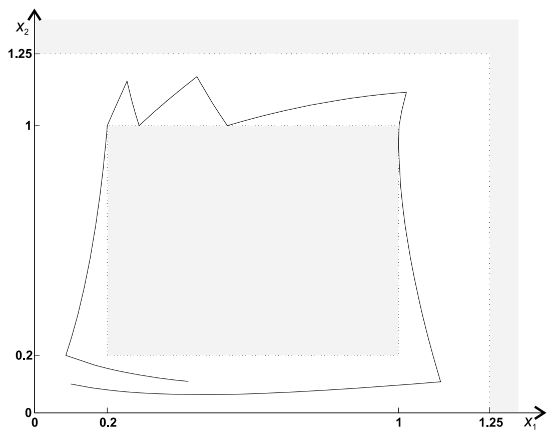

Figure 8: Simulated trajectory of the controlled laboratory pilot plant

Other problems arise when existing methods are used to solve practical problems. Even in simple cases like the one presented in the previous section the solution cannot be obtained analytically. Numerical methods have to be applied, which makes the solution computationally demanding. Additionally, even small disturbances on the measured signals can cause unpredicted transitions between state space regions and generate unexpected plant events. An ad hoc solution to this problem is the use of hysteresis around the state space boundaries. Simplified application of synthesis methods to practical problems is expected in the further development as well as the improved robustness of derived solutions.

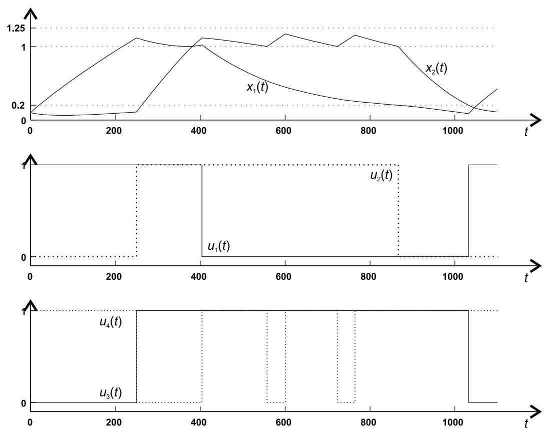

Figure 9: Simulated signals of process states (top), inputs (middle) and outputs (bottom)

5 Acknowledgement

The partial financial support of the Ministry of Science and Technology of the Republic of Slovenia under grant J2-2417-1538-00 is gratefully acknowledged.

References

[1] R. Alur et al. (1993) Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems, in Hybrid Systems, *Lecture Notes in Computer Science* (Springer), **736**, 209-229.

[2] P.J. Antsaklis et al. (1993) Hybrid System Modeling and Autonomous Control Systems, in Hybrid Systems, *Lecture notes in Computer Science* (Springer), **736**, 366-392.

[3] P.I. Barton et al. (1997) Analysis and Control of Combined Discrete/Continuous Systems: Progress and Challenges in the Chemical Process Industries, *AIChE Symposium Series*, **93** (316), 102-114.

[4] A. Bemporad et al. (1999) Control of systems integrating logic, dynamics, and constraints, *Automatica, Special issue on hybrid systems*, **35**, 407-427.

[5] M.S. Branicky et al. (1998) A unified framework for hybrid control: model and optimal control theory, *IEEE Trans. Autom. Contr.*, **43**, 3145.

[6] X.D. Koutsoukos et al. (2000) Supervisory Control of Hybrid Systems, *Proceedings of the IEEE*, **88**, 1026-1049.

[7] J. Lunze et al. (1999) Deterministic discrete-event representations of linear continuous-variable systems, *Automatica, Special issue on hybrid systems*, **35**, 395-406.

[8] A. Nerode et al. (1993) Models for Hybrid Systems: Automata, Topologies, Controllability, Observability, in Hybrid Systems, *Lecture notes in Computer Science* (Springer), **736**, 317-356.

[9] T. Niinomi et al. (1995) Synthesis of Supervisory Controllers for Hybrid Systems Based on Approximating Automata, Proceedings of the 34th Conference on Decision and Control, New Orleans, 1461-1466.

[10] J. Raisch (2000) Discrete Abstractions of Continuous Systems - an Input/Output Point of View, *Mathematical and Computer Modelling of Dynamical Systems, special issue on Discrete Event Models of Continuous Systems*, **6**, 6-29.

[11] P.J.G. Ramadge et al. (1989) The Control of Discrete Event Systems, *Proceedings of the IEEE*, **77**, 81-98.

[12] J.A. Stiver et al. (1996) A Logical DES Approach to the Design of Hybrid Control Systems, *Mathematical Computer Modeling*, **23**, 55-76.