A NEW APPROACH TO HYBRID SYSTEM SIMULATION: DEVELOPMENT OF A SIMULINK LIBRARY FOR PETRI NET MODELS

Mihaela Matcovschi, Constantin Popescu and Octavian Pastravanu

Department of Automatic Control and Industrial Informatics
Technical University "Gh. Asachi" of Iasi, Blvd. Mangeron 53A, 700050 Iasi, Romania
Phone/Fax: +40-232-230751*, E-mail:* {mhanako, opastrav}@delta.ac.tuiasi.ro

Abstract: *The paper focuses on a new approach to MATLAB Simulink-based modelling and analysis of hybrid systems whose event-driven part(s) is (are) modeled by the Petri Net (PN) formalism. A* ***Petri Net Simulink Block*** *(PNSB) allows the connection of a PN with other Simulink blocks, by means of events and data regarding the current status of the PN. Some issues regarding the implementation and exploitation of the Simulink blocks included in the* ***Petri Net Library*** *(PNL) are briefly presented. A complex example illustrates the effectiveness of the utilization of PNSBs in simulation-based performance analysis of hybrid systems.*

Keywords: *Petri nets, Hybrid systems, MATLAB, Simulink, Control engineering education.*

1. INTRODUCTION

Hybrid systems represent a topic of great interest in the control systems area. Their behaviour is determined by interacting continuous and discrete dynamics. During the last two decades different modelling issues and approaches have been addressed, in accordance with the scientific motivation of the groups promoting the researches. These approaches differ with respect to the emphasis on the complexity of the continuous and discrete dynamics, providing various analysis (verification), simulation and synthesis methodologies (Koutsoukos and Antsaklis, 2003). Despite some valuable intentions to prove the compatibility between various trends, e.g. (Heemels *et al.*, 2001), the state of art still cannot offer a unifying theoretical framework. This aspect is also reflected by the diversity of the software instruments used by the research groups. Generally speaking, each research group developed its own software tools, in order to support specific applications, corresponding to particular theoretical approaches.

The dynamics of a hybrid system can be described by a finite number of continuous dynamical models, represented by sets of nonlinear differential or difference equations, and a set of rules for switching between these models. These switching rules are typically

described by logic expressions or a discrete event system with a finite automaton representation.

Our approach to modelling, simulation and analysis of hybrid systems is based on the usage of Petri net (PN) formalism (Murata, 1989) for representing their discrete-event driven parts. The main reason we have opted for this methodology is the expressiveness of PNs which can be viewed as a generalization of finite automata. PNs proved to be an excellent tool for capturing concurrency, conflict, synchronization and buffer sizes within a system. A PN representation for a concurrent process will be more compact (fewer vertices) than its associated automaton representation. Ever since their introduction, PNs have been extensively used to model manufacturing systems, communication systems, information processing systems, and chemical processes among others. Furthermore, they can be successfully incorporated in approaches to hybrid systems, e.g. (Nenninger and Krebs, 1997), (Koutsoukos and Antsaklis, 1999).

MATLAB-Simulink is widely recognized as the most popular software environment in Control Engineering Education. Despite that, besides the Stateflow, also developed by the MathWorks Inc. (the producers of MATLAB) and relying on finite state machines for representing the discrete dynamics of a hybrid system, there are currently available only a few MATLAB-Simulink based software packages dedicated to hybrid systems modelling. CheckMate (Chutinan and Krogh, 2003) is a tool for modelling, simulating and verifying properties of hybrid systems based on standard Simulink and Stateflow blocks. Other software packages are the NetLab (NetLab, 2004) and the recently updated Hybrid Toolbox (Bemporad, 2005).

This paper presents new instruments for addressing the dynamics of hybrid systems within the framework of MATLAB-Simulink software. A Simulink library, the ***Petri Net Library*** (PNL), was especially developed to allow including discrete event driven modules modelled by PNs into Simulink block diagrams. Our approach to hybrid systems modelling and analysis allows incorporating accurate nonlinear models for the continuous dynamics (built from blocks available in the standard Simulink libraries) with PN models for discrete event dynamics.

The design and implementation of the PNL derived a full benefit from our previous experience accumulated during the development of the ***Petri Net Toolbox*** (***PN Toolbox***) for MATLAB (Mahulea *et al.*, 2005), (Pastravanu *et al.*, 2004), which commenced in 2000. In mid 2003 we released version 2.0 of the ***PN Toolbox*** that was included, at the beginning of 2004, in the Connections Program of The MathWorks Inc. (MathWorks, 2005), as broadening the utilization domain of MATLAB toward the area of discrete-event systems. The current version 2.2 is fully compatible with MATLAB 7 SP2.

The organization of this paper is as follows. Section 2 presents a brief description of the PNL. The main facilities offered by a ***Petri Net Simulink Block*** (PNSB) are depicted in Section 3. The simulation-based performance analysis of a complex system illustrates the utilization of a PNSB in Section 4. Some final remarks are formulated in Section 5.

2. DESCRIPTION OF THE PETRI NET LIBRARY

The PNL (Fig. 1) contains three ***Petri Net Simulink Blocks*** (PNSBs) corresponding to the three types of PN models that can be integrated into the Simulink model of a hybrid system, namely untimed, P-timed and T-timed. The current version of the PNL includes the Simulink block operating with untimed PNs which was presented in (Matcovschi *et al.*, 2005). The changes brought by this new version of the PNL also refer to the internal architecture of the PNSBs, the treatment of the external events used in the synchronization of the PNs and the possibility of broadcasting internal events generated by firing certain transitions in the PN model. A Simulink block diagram can contain any number of PNSBs needed to model a complex system.

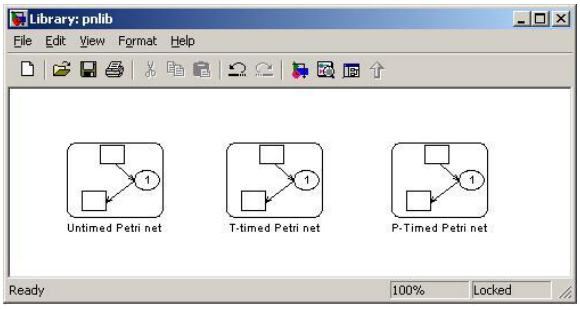

Fig. 1. The Petri Net Library.

The PN model stored in a PNSB should contain *synchronized* (*triggered*) *transitions* (David and Alla, 2005). The firing of a synchronized transition depends on the occurrence of external (triggering) events. It is fired whenever (i) it is enabled by the net marking and (ii) one of its associated triggering events, defined at the PN level, occurs. Both finite and infinite server semantics (David and Alla, 2005) can be used for transition firings.

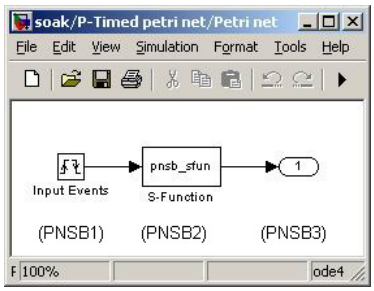

Fig. 2. The internal architecture of a PNSB.

The internal architecture of a PNSB is presented in fig. 2. A PNSB accepts, as inputs, a set of signals, which can be continuous or discrete; the evolution of each signal can generate a Simulink event. Once an input signal generates an event, the simulation time of Simulink "freezes", the PNSB identifies the generated event and fires all the enabled transitions.

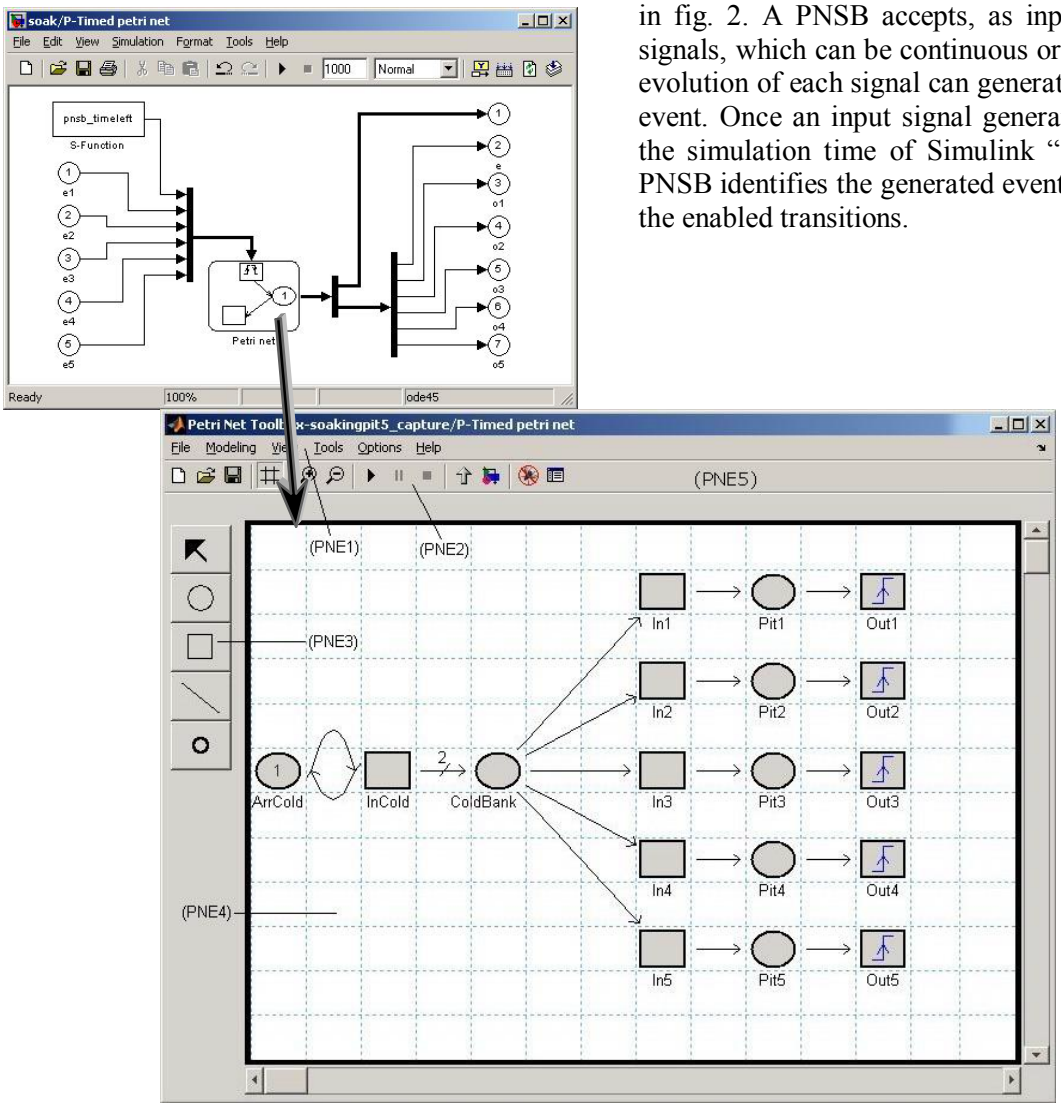

Fig. 3. The main window of the ***PNSB Editor***.

At each simulation step, the PNSB outputs the net marking as a vector of n integer values, where n represents the number of places in the net. In the case when the firing of some transition in the PN model is associated with an internal event, these events are also outputted. After that, the simulation is resumed until a new event is generated. The implementation of the PNSBs required the development of tools for managing the events and for controlling the simulation.

3. USAGE OF THE PNSB

A PNSB is equipped with a graphical interface that allows the user to draw the PN model (***PNSB Editor*** - see fig. 3), define the triggering

events (***PNSB Event Explorer*** - see fig. 4) and debug the Simulink model (***PNSB Debugger*** - see fig. 7). The operation of the PNSB relies on callback functions that (i) initialize variables, (ii) generate the graphical interface, (iii) display the PNSB Editor when the user double clicks the Simulink block, (iv) hide the editor window when the user closes it and (v) save/load the PN model into/from an xml file.

It is worth noticing that the functions of the ***PNSB Editor*** are similar to the editing facilities available in the *Draw Mode* of the ***PN Toolbox***; this ensures the immediate adaptation of the user's skills, once he/she has already been acquainted with the exploitation of the ***PN Toolbox***. The ***PNSB Editor*** exhibits five control panels (fig. 3): *Menu Bar* (PNE1), *Quick Access Toolbar* (PNE2), *Drawing Panel* (PNE3), *Drawing Area* (PNE4) and a *Message Box* (PNE5). The *Menu Bar* (PNE1) displays a set of six drop-down menus, from which the user can access all the facilities available in the application. The *Quick Access Toolbar* (PNE2) maps the most frequently used facilities of the ***PNSB Editor***. The *Drawing Area* (PNE4) is implemented as a matrix of cells, where the nodes of the PN graph are to be placed, with two scrollbars for moving the desired parts of the graph into view. The *Drawing Panel* (PNE3) presents five buttons that facilitate user access to Edit objects, Add Place, Add Transition, Add Arc and Add Token commands. Messages related to the simulation are displayed in the *Message Box* (PNE5).

The ***PNSB Event Explorer*** (fig. 4) allows the user to manage the triggering events of the current PN model. The user can edit the name (EE1) and the triggering mode (EE2) for each external event. The number and order of PNSB input signals has to match the number and order of triggering events defined in the Event Explorer for the PN model. The selection of a certain event displayed in the Event panel is accomplished by pressing the corresponding *Select event* button (EE3). The user can *Add* (EE4) or *Delete* (EE5) events, and change their order by moving up (EE6) or down (EE7). There are three types of events that can be defined by the user, the differences between them resulting from the triggering conditions that must be met by the corresponding Simulink signal. Thus, a *rising edge* event triggers the PNSB when the input signal rises from a zero or negative value to a positive value (or zero if the initial value is negative), while a *falling edge* event is activated whenever the input signal falls from a positive value to a zero or negative value (or zero if the initial value is positive). An *either edge* event triggers the PNSB when the input signal is either rising or falling. The default triggering mode for a newly created event is *either edge*.

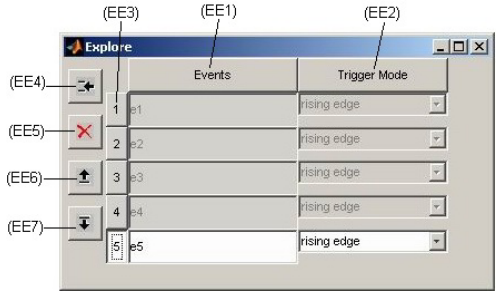

Fig. 4. The ***PNSB Event Explorer***.

The ***PNSB Editor*** allows specifying the event(s) triggering the firing of a transition in the PN model by means of the corresponding *Edit Transition* window (fig. 5). If multiple events are associated with a transition they must be separated by ',' and, as a consequence, the transition can fire on the occurrence of an event in the list. A transition controlled by a triggering event is distinguished from an ordinary transition by one of the icons presented in fig. 6.

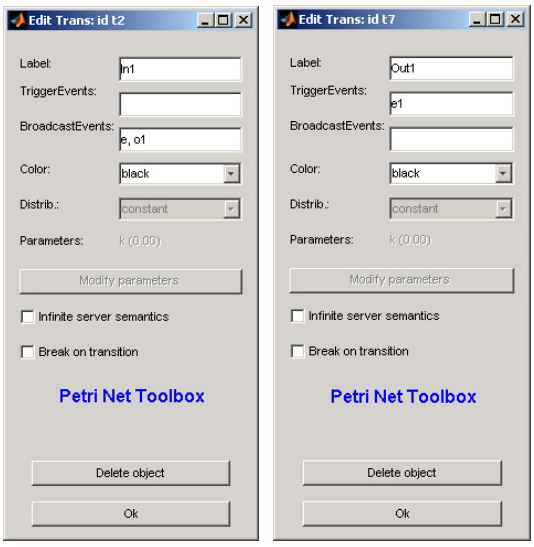

Fig. 5. *Edit Transition* windows.

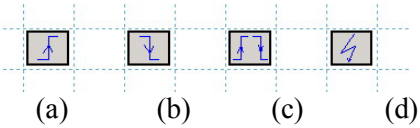

Fig. 6. Icons used for representing: (a) rising edge, (b) falling edge, (c) either edge triggered transitions; (d) transitions triggered by events that are still undefined.

The ***PNSB Debugger*** is a useful tool when simulating a Simulink model containing both time-driven and event-driven systems. The debugger pauses the simulation at each simulation step, allowing the user to inspect the current state of the Petri net or to visualize the evolution of some particular signals from the Simulink model. The MATLAB Fig. opened when the user selects the *Debugger* option from the *Tools* menu is presented in fig. 7. The upper buttons (D1) start and stop the simulation, the time progress being displayed in a bar (D2) together with the simulation time, the start and the stop simulation time. If the *Enable Debugger* checkbox (D3) is checked, the simulation is interrupted on each occurrence of a triggering event associated with the PNSB. The *Step* button (D4) fires one transition of the Petri net at a time, while the *Continue* button (D5) runs the simulation introducing, between successive firings, a delay set by the *Delay* listbox (D6). The maximum number of firings is set by the value entered in the *Breakpoint* edit item (D7).

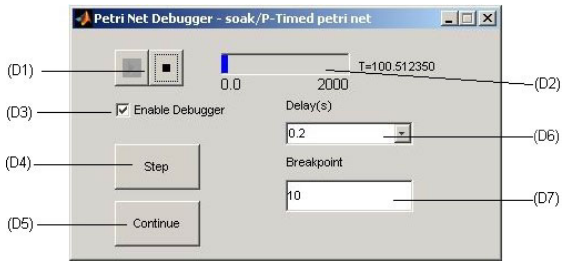

Fig. 7. The PNSB Debugger.

4. ILLUSTRATIVE EXAMPLE

In a steel plant, steel ingots arrive in pairs at a soaking pit furnace where they are heated so they can be rolled in the next stage of the process (Popescu, 2005). There is space for five ingots in the soaking pit furnace. When an ingot arrives at the furnace, it is placed into the furnace if space is available; otherwise, it is placed in the cold ingots bank to wait for free space. The interarrival times are exponentially distributed, the mean value being μ min. The initial furnace temperature is 2,200 degrees Fahrenheit. The furnace is heated according to the differential equation:

$$dF / dt = 2(2600 - F) \tag{1}$$

where F is the furnace temperature. The initial temperature of an arriving ingot is uniformly distributed between 300 and 500 degrees Fahrenheit. When an ingot is inserted into the furnace, it reduces the furnace temperature by the difference between the furnace temperature and the ingot temperature, divided by the number of ingots in the furnace. The temperature change of ingots as they are heated in the furnace is described by the differential equation:

$$dP_j / dt = 0.15(F - P_j) \tag{2}$$

where P_j is the temperature of the ingot in the j^{th} position in the pit. Each ingot is heated in the furnace until it reaches 2,200 degrees and then it is removed.

We are interested in the analysis of the system performances for a simulation time of 2,000 min. when the mean interrarival time μ ranges from 5 min to 10 min, by recording the statistics on the utilization of the furnace, the heating time for the ingots, the temperature of the furnace, and the number of ingots in cold bank.

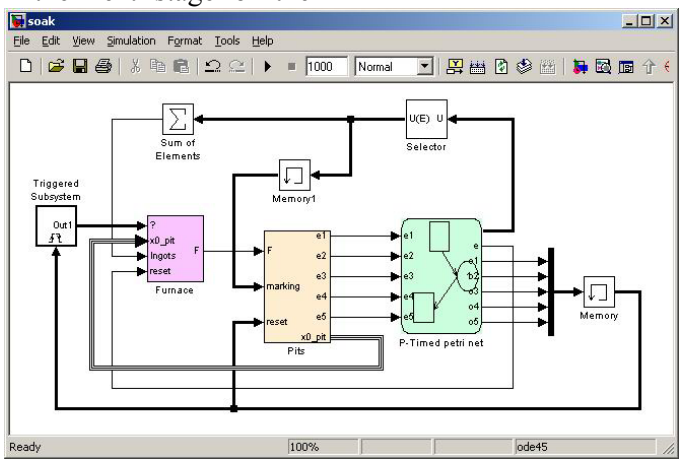

Fig. 8. Simulink model of the soaking pit furnace.

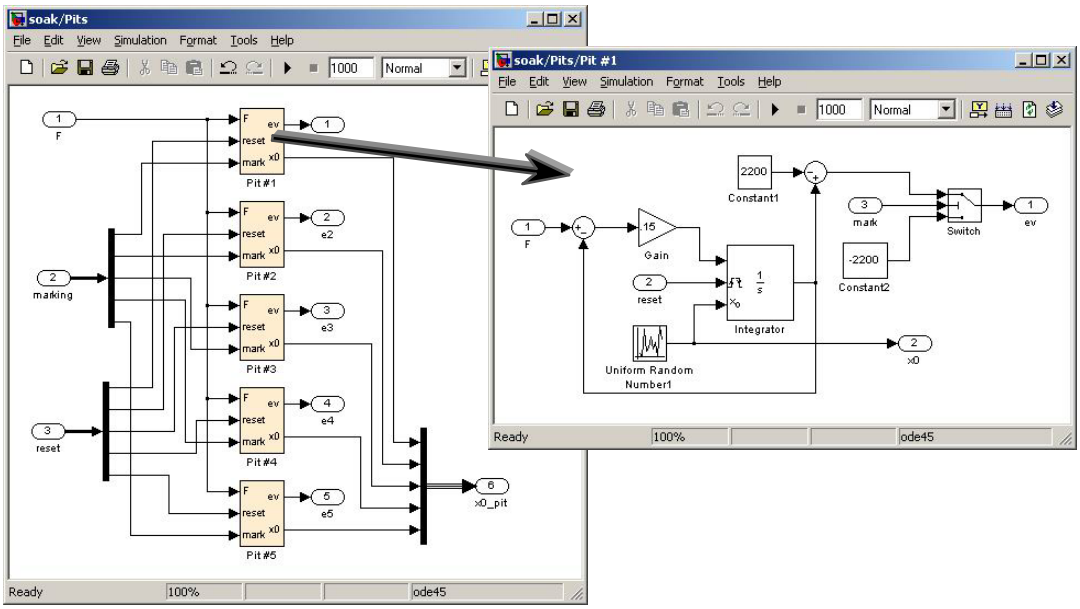

Fig. 9. Subsystems modelling the heating of the pits.

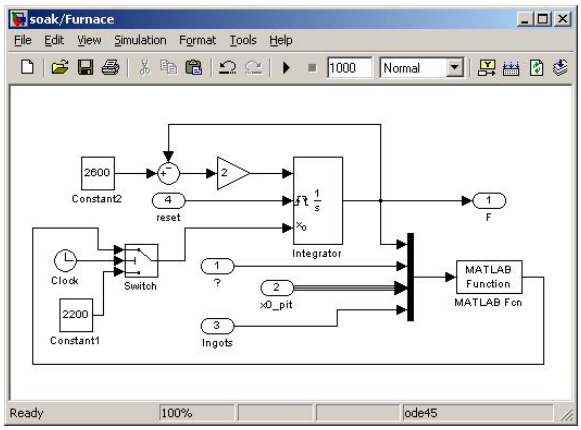

Fig. 10. Subsystem modelling the furnace.

The Simulink model corresponding to this complex system is presented in fig. 8. It includes the subsystems modelling the heating of the pits (fig. 9) and of the furnace (fig. 10) built from conventional Simulink blocks.

The P-timed PN modelling the control logic is presented in fig. 3. The place ArrCold is timed, the mean value of the associated exponentially distributed delay being μ. The firing of transition InCold, corresponding to the arrival of 2 cold ingots in the system, places 2 tokens in place ColdBank that has infinite capacity. Places Pit1, …, Pit5, whose capacities equal 1, model the availability of the soaking pits in the furnace. Transitions In1, …, In5 model the transfer of an ingot from the cold ingots bank into the furnace. The firing of transition Ini broadcasts two events, e and oi, used to reset the integrators in the subsystems modelling the heating of the furnace and that of Pit i, respectively, for $i = 1,\ldots,5$. Transitions Out1, …, Out5 are triggered by the rising edge type external events e1, …, e5 generated based on the temperature of the ingot from the corresponding pit in the furnace.

Fig.s 11 and 12 show the time evolution of the temperatures for the furnace and Pit#1, respectively, for the last 100 min of operation, in the case when the mean interarrival time of cold ingots is 7 min. Table 1 presents the numerical values of the performance indices obtained through simulation.

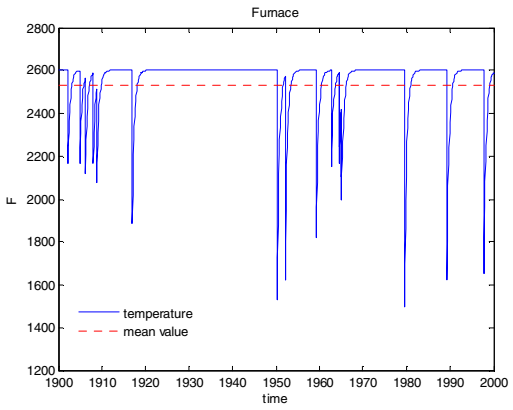

Fig. 11. Time evolution of the furnace temperature for $\mu = 7$ min.

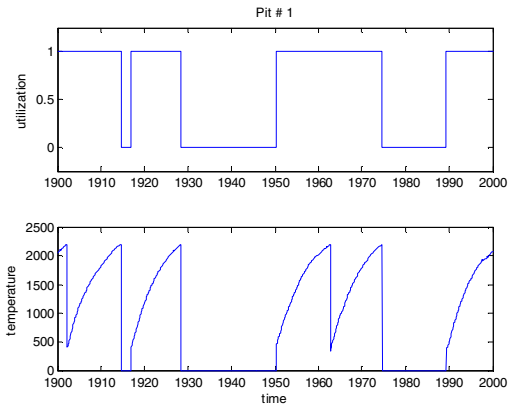

Fig. 12. Time evolution of the utilization and the temperature of Pit#1 for $\mu = 7$ min.

Table 1. Mean values of the performance indices of the system under study obtained through simulation.

Mean interarrival time [min]	5	6	7	8	9	10
Total # processed	775	678	574	500	458	421
Mean # in cold bank	17.35	3.33	1.57	0.78	0.62	0.51
Mean waiting time [min]	46.71	9.74	5.45	3.13	2.72	2.41
Mean utilization / pit	0.978	0.850	0.717	0.621	0.566	0.518
Mean processing time [min]	12.61	12.53	12.49	12.41	12.36	12.31
Mean furnace temperature [F]	2514	2522.5	2532.4	2537.7	2541.4	2544.9

5. CONCLUSIONS

The newly created Simulink blocks extend the possibilities offered by the MATLAB-Simulink environment for studying the dynamics of the hybrid systems, employing both discrete and continuous behavior. They allow the integration of PN models with the general simulation philosophy implemented in Simulink.

REFERENCES

[1] Bemporad, A. (2005). *Hybrid Toolbox*, University of Siena, http://www.dii.unisi.it/hybrid/toolbox/.

[2] Chutinan, A. and B. H. Krogh (2003). Computational Techniques for Hybrid System Verification, *IEEE Trans. on Automatic Control*, vol. **48**, no. 1, pp. 64-75.

[3] David, R. and H. Alla (2005). *Discrete, Continuous, and Hybrid Petri Nets*, Berlin Heidelberg: Springer-Verlag.

[4] Heemels, M., B. De Schutter and A. Bemporad (2001). Equivalence of hybrid dynamical systems, *Automatica*, vol. **37**, pp. 1082-1091.

[5] Koutsoukos, X.D. and P.J. Antsaklis (1999), Computational Issues in Intelligent Control: Discrete-event and Hybrid Systems, In: *Soft Computing and Intelligent Systems: Theory and Practice,* N. Sinha and M. Gupta, Eds., Academic Press, pp. 39-69.

[6] Koutsoukos, X.D. and P.J. Antsaklis (2003), Hybrid Dynamical Systems: Review and Recent Progress, In: *Software-Enabled Control:*

Information Technologies for Dynamical Systems, T. Samad and G. Balas, Eds., Wiley-IEEE Press, pp. 273-298.

[7] Mahulea, C., M.H. Matcovschi and O. Pastravanu (2005). *Home Page of the Petri Net Toolbox*, http://www.ac.tuiasi.ro/pntool.

[8] Matcovschi, M.H., C. Lefter and O. Pastravanu (2005). Petri nets in hybrid system simulation under Simulink. *The 15th Int. Conf. on Control Systems and Computer Science CSCS15*, Bucharest, CD-ROM.

[9] The MathWorks Inc. (2005). *Connections Program*,

http://www.mathworks.com/products/connections.

[10]Murata, T. (1989). Petri nets: properties, analysis and applications, In: *Proc. of the IEEE*, vol. **77**, no. *4*, pp. 541-580.

[11]Nenninger, G. and V. Krebs (1997). Modeling and analysis of hybrid systems: a new approach integrating Petri nets and differential equations, *Joint Workshop on Parallel and Distributed Real-Time Systems*, pp. 234-238.

[12]NetLab Homepage (2004). http://www.irt.rwth-aachen.de/download/netlab/.

[13]Pastravanu, O., M.H. Matcovschi and C. Mahulea (2004). Petri net toolbox – teaching discrete event systems under MATLAB, In: *Advances in Automatic Control*, M. Voicu (Ed.), Kluwer Academic, Boston/Dordrecht/London, pp. 257-270.

[14 Popescu, C. (2005). *Implementation and Testing of Simulink Blocks for Petri Net Models*, Research Report – Socrates-Erasmus Mobility Program, Technical University "Gh. Asachi" of Iasi and University of Sheffield.