



---

# BULETINUL ȘTIINȚIFIC

al

Universității „POLITEHNICA” din Timișoara, România

---

Seria AUTOMATICĂ ȘI CALCULATOARE

---

---

## PERIODICA POLITEHNICA

„POLITEHNICA” University of Timișoara, Romania

---

Transactions on AUTOMATIC CONTROL AND COMPUTER SCIENCE

---

Tomul 47(61), No. 2, 2002

ISSN 1224-600X



EDITURA POLITEHNICA

## Computer Tools For Linear Systems Over Max-Plus Algebra

Mihaela-Hanako Matcovschi, Cristian Mahulea, Octavian Pastravanu

Department of Automatic Control and Industrial Informatics  
Technical University "Gh. Asachi" of Iasi, Blvd. Mangeron 53A, 6600 Iasi, Romania  
Phone: +40-32-230751, Fax: +40-32-214290, E-Mail: {mhanako, cmahulea, opastrav}@delta.ac.tuiasi.ro

**Abstract** – *The dynamics analysis for linear systems described by state-space representations in max-plus algebra is addressed within the context of the software facilities offered by the Petri Net Toolbox (PN Toolbox) running under MATLAB. By developing this toolbox, the application field of the MATLAB environment (extremely popular among control engineers) is considerably enlarged towards covering event driven behavior. The max-plus instruments available in PN Toolbox are discussed from the point of view of the implemented algorithm and their usage is illustrated by two relevant examples.*

**Keywords:** *state space models, max-plus algebra, system theory, Petri nets, event graphs, MATLAB software.*

### I. INTRODUCTION

Despite the large body of work invested in the theoretical study of linear systems described over the max-plus algebra, e.g. [1], [2], [3], [4], very few information is currently available in the literature with regard to the implementation of a simulator for the dynamics of such systems. A software devoted to general purpose computations in this algebra, running under Scilab [5], is presented in [6], but it does not include ready to use facilities for the simulation of discrete event systems. However, the usage of this software as a start point in developing a proper simulator was not very attractive from our point of view, since our interest in computer applications focuses on the popular software MATLAB [7], [8].

For the concrete analysis of system behavior hand calculus is rather cumbersome even for problems of small complexity, fact that fully motivates our work in developing a max-plus simulator. Taking advantage of our experience gained in exploiting MATLAB capabilities in the framework of discrete event systems, such a simulator has been designed and implemented as an instrument under Petri Net Toolbox (PN Toolbox) [9], [10], [11], [12]. The main characteristics of this simulator do not depend on the general philosophy of PN Toolbox; it can be used directly

under MATLAB and its integration with PN Toolbox offers the advantage of the automatic calculation of the max-plus state-space representation from a Petri net model.

The current paper is structured as follows. Section 2 points out the specific requirements in developing a max-plus based simulator, formulated in broader terms, independent of the MATLAB context. A brief presentation of the PN Toolbox is achieved in Section 3, so that to allow the discussion in Section 4 of the particular aspects involved in the MATLAB implementation. Section 5 illustrates the effectiveness of our simulator by two case studies representative for dynamics driven by discrete events. Some concluding remarks are formulated in Section 6.

### II. SPECIFIC REQUIREMENTS IN DEVELOPING A MAX-PLUS BASED SIMULATOR

The usage of max-plus state space models for discrete event systems rely on a non-standard algebraic structure (dioid) [1] which rises a series of technical problems when implementing simulation facilities within usual software environments. Such problems originate in:

- (i) The basic set of elements ( $\mathbb{R} \cup \{-\infty\}$ ) of the dioid represents the left closure of the real axis. Therefore, the special symbol  $e = -\infty$  should have the full status of an operand.
- (ii) The two operators in the dioid need to be represented by two special characters. In order to use the regular symbols for addition (+) and multiplication (\*) in accordance with the new meanings (i.e. max and plus in classical notation), a reassignment of their modus operandi is requested. This reassignment should be valid only when operating with max-plus entities and for any other context the operators preserve their classical significance.
- (iii) Matrix addition and multiplication (defined by generalizing the new meaning already discussed at (ii)) necessitates proper software tools.
- (iv) For starting the simulation of a max-plus model associated with a Petri net, a set of initial conditions is requested. Although all these conditions are theoretically cognizable, in most cases just part of them can be a priori formulated, whereas the

remaining ones have to be calculated adequately. An automatic approach to such calculations cannot rely on ad-hoc solutions, but demands a systematic procedure.

- (v) Generally speaking, the construction of the max-plus state-space models results in implicit equations that cannot be directly used for the progress of simulation. To automatically derive the corresponding explicit form an algorithm should be implemented for solving linear systems  $x = Ax + b$  in the Kleene sense [1].

For handling all these aspects (i)-(v), a versatile and powerful software environment should be chosen to host such a simulation application.

### III. MATLAB-EMBEDDED PETRI NET TOOLBOX

*Petri Net Toolbox* is a software tool embedded in the MATLAB environment for discrete-event systems simulation and analysis, designed and implemented at the Department of Automatic Control and Industrial Informatics of the Technical University “Gh. Asachi” of Iasi.

The skeleton and functionality of version 1.0 of this toolbox were briefly presented in [10]. Version 2.0, which has just been released, brings a new technology in the implementation of the Graphical User Interface, allows links to other software, upgrades several algorithms and simulation statistics [12], offers specific tools to address structural properties [11] and incorporates facilities for coloured Petri nets [9]. In the same time, this second version differs from the previous one by substantial improvements with regard to the max-plus simulation and analysis of timed event graphs.

The integration with the MATLAB philosophy ensures generous computational resources for various types of applications due to the high quality routines provided by MATLAB and some of its toolboxes, such as Statistics Toolbox, Optimization Toolbox etc. On the other hand, the PN Toolbox has the incontestable merit of broadening the MATLAB's utilization domain towards the area of discrete-event systems, which is now covered only by the State-Flow package. Generally speaking, the MATLAB orientation of the PN Toolbox is able to confer the necessary flexibility for further improvement of this software, by upgrading the already existing tools and by adding new ones.

### IV. MAX-PLUS BASED SIMULATION IN PN TOOLBOX VERSION 2

The MATLAB implementation of PN Toolbox permits to develop elegant programming solutions to the problems (i)-(v) discussed in section 2. Within this context, the response to (i) is given by the existence of built-in MATLAB function **Inf**, which returns the IEEE arithmetic representation for positive infinity. The reassignment of the regular symbols for addition and multiplication

requested at (ii) and (iii) can be achieved by constructing a new class of MATLAB objects defined as matrices over the max-plus dioid. Thus, when operating with objects belonging to this class, the regular methods for addition and multiplication are overloaded.

With regard to (iv), PN Toolbox is able to directly derive the max-plus state representation from the topology and initial marking of a timed event graph, in an implicit form:

$$\mathbf{x}(k) = \bigoplus_{i=0}^M [\mathbf{A}_i \mathbf{x}(k-i) \oplus \mathbf{B}_i \mathbf{u}(k-i)], \quad (1)$$

$$\mathbf{y}(k) = \bigoplus_{i=0}^M [\mathbf{C}_i \mathbf{x}(k-i) \oplus \mathbf{D}_i \mathbf{u}(k-i)]. \quad (2)$$

where  $M$  denotes the maximal number of tokens in the initial marking. The components of the input vector  $\mathbf{u}(k) = [u_1(k) \ u_2(k) \ \dots \ u_m(k)]^T$  and those of the output vector  $\mathbf{y}(k) = [y_1(k) \ y_2(k) \ \dots \ y_p(k)]^T$  represent the moments of the  $k$ -th firing instances of the  $m$  source and, respectively, of the  $p$  sink transitions of the net. In a similar manner, the state vector  $\mathbf{x}(k) = [x_1(k) \ x_2(k) \ \dots \ x_n(k)]^T$  corresponds to the  $n$  transitions that are connected to both predecessor and successor places.

Equation (1) may be written in the equivalent form

$$\mathbf{x}(k) = \mathbf{A}_0 \mathbf{x}(k) \oplus \mathbf{v}(k), \quad (3)$$

where

$$\mathbf{v}(k) = \bigoplus_{i=1}^M \mathbf{A}_i \mathbf{x}(k-i) \oplus \bigoplus_{i=0}^M \mathbf{B}_i \mathbf{u}(k-i). \quad (4)$$

The programming techniques employed to overload the regular methods for addition and multiplication are extended in the case of Kleene-star operator such that addressing problem (v) becomes a straightforward task and equation (1) turns into the explicit form:

$$\mathbf{x}(k) = \mathbf{A}_0^* \mathbf{v}(k), \quad (5)$$

where  $\mathbf{A}_0^* = \mathbf{E} \oplus \mathbf{A}_0 \oplus \mathbf{A}_0^2 \oplus \dots \oplus \mathbf{A}_0^{n-1}$  and  $\mathbf{E}$  stands for the identity matrix.

In order to preserve the original meaning of the state variables, the simulation performed by PN Toolbox does not appeal to the standard method [2] founded on the augmentation of the state-vector, but it exploits directly equations (5) and (2).

This technique leads to the following problem: for  $1 \leq k \leq M$  both the state vector  $\mathbf{x}(k)$  and the output vector  $\mathbf{y}(k)$  depend on  $\mathbf{x}(0)$ ,  $\mathbf{x}(-1)$ , ...,  $\mathbf{x}(k-M)$  and  $\mathbf{u}(0)$ ,  $\mathbf{u}(-1)$ , ...,  $\mathbf{u}(k-M)$ . It seems natural to assume that all the components of these vectors are null, i.e. equal to  $\mathbf{e} = -\infty$ , since the counting of the firing instances of every transition in the net starts at the initial moment  $t_0 = 0$  with  $k = 1$ . In order to compensate the loss of information from equations (5) and (2) that appears by making this assumption, on the one hand, and to take into account the role of those transitions that are initially  $q$ -enabled from the

very beginning of the simulation procedure, on the other hand, for  $1 \leq k \leq M$  it is necessary to modify equations (5) and (2) according to

$$\mathbf{x}(k) = \mathbf{A}_0^* \otimes [\bar{\mathbf{x}}(k) \oplus \mathbf{v}(k)], \quad (6)$$

$$\mathbf{y}(k) = \bar{\mathbf{y}}(k) \oplus \mathbf{w}(k), \quad (7)$$

where  $\mathbf{v}(k)$  is given by (4) and

$$\mathbf{w}(k) = \bigoplus_{i=0}^M [C_i \mathbf{x}(k-i) \oplus D_i \mathbf{u}(k-i)]. \quad (8)$$

The auxiliary vectors  $\bar{\mathbf{x}}(k)$  and  $\bar{\mathbf{y}}(k)$  introduced in (6) and (7) are computed as follows:

- if transition  $t_i$ , corresponding to the  $i$ -th component of  $\mathbf{x}(k)$ ,  $i=1,2,\dots,n$ , is  $q_i$ -enabled in the initial marking, then  $\bar{x}_i(k)=0$  for  $k=1,\dots,q_i$ , and  $\bar{x}_i(k)=\mathbf{e}$  for  $k > q_i$ ;
- if sink transition  $t_s$ , corresponding to the  $s$ -th component of  $\mathbf{y}(k)$ ,  $s=1,2,\dots,p$ , is  $q_s$ -enabled in the initial marking, then  $\bar{y}_s(k)=0$  for  $k=1,\dots,q_s$ , and  $\bar{y}_s(k)=\mathbf{e}$  for  $k > q_s$ .

This way, vectors  $\bar{\mathbf{x}}(k)$  and  $\bar{\mathbf{y}}(k)$  influence only the first  $q_{x_{\max}} = \max\{q_i, i=1,\dots,n\}$  iterations of  $\mathbf{x}(k)$  and, respectively, the first  $q_{y_{\max}} = \max\{q_s, s=1,\dots,p\}$  iterations of  $\mathbf{y}(k)$ . For  $k > q_{x_{\max}}$  equation (6) is equivalent to (5) and for  $k > q_{y_{\max}}$  equation (7) is equivalent to (2). Once  $k$  becomes greater than  $M$  ( $k > M$ ), all the vectors in the right hand sides of (5) and (2) are cognizable, because

the previous iterations of the algorithm provide the state vector whereas the values of the input vector are a priori known. Hence, these two equations may be directly implemented in simulation.

For strongly connected timed event graphs, PN Toolbox is equipped with an analysis procedure for the eigenstructure of the net, which explores the periodicity exhibited by the steady-state behaviour.

## V. CASE STUDIES

In order to illustrate the effectiveness of the new max-plus tools incorporated in PN Toolbox, two relevant examples are considered.

*Example 1* (a communications protocol – adapted from [13]). The place timed Petri net model is presented in the main window of PN Toolbox captured in figure 1. The significance of each place  $p$  in the model and the numerical value of the time duration allocated to it, generically denoted by  $d(p)$ , are as follows:

- ready to send message to lower level:  $d(p1) = 6$ ;
- message sent:  $d(p2) = 3$ ;
- ready to receive message from upper level:  $d(p3) = 5$ ;
- wait for ACK:  $d(p4) = 4$ ;

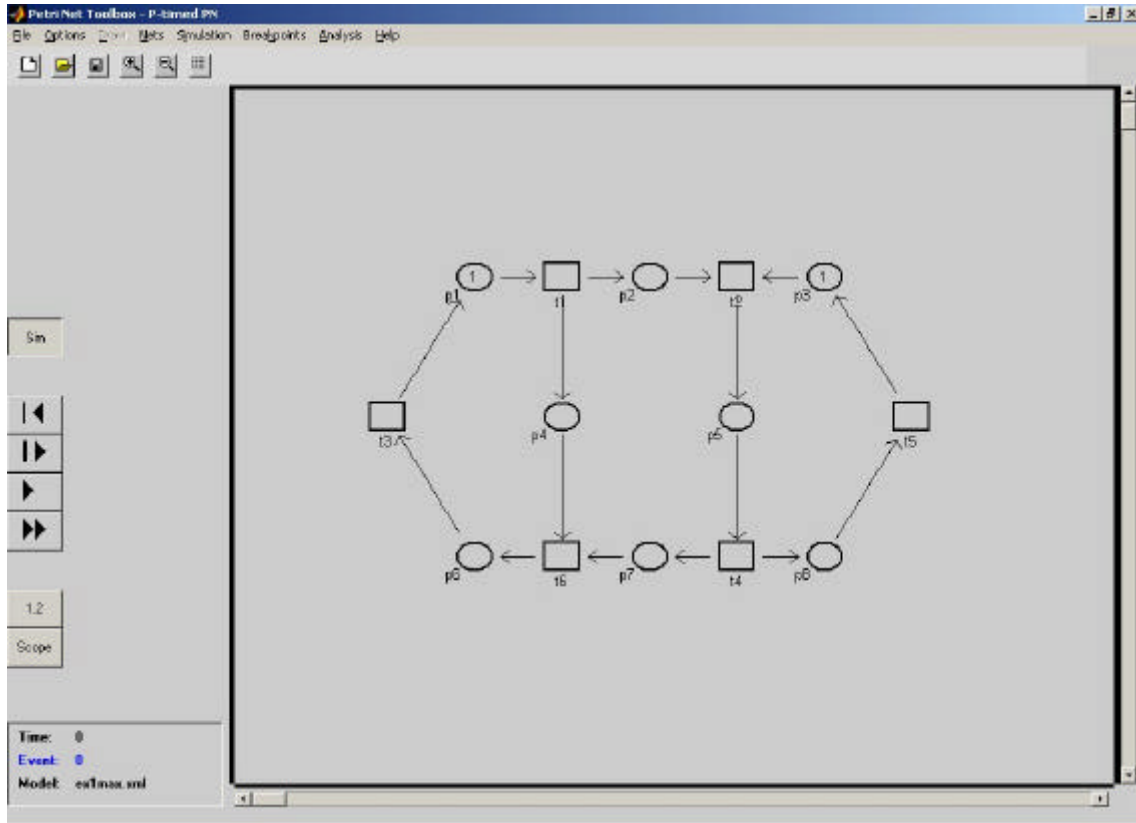


Fig. 1. Screen capture of the main window of PN Toolbox with the event graph in Example 1

- message received:  $d(p5) = 7$  ;
- transfer completed (upper level):  $d(p6) = 1$  ;
- ACK signal sent:  $d(p7) = 8$  ;
- transfer completed (lower level):  $d(p8) = 2$  .

The transitions in the PN model correspond to the occurrences of the subsequent events: t1: send message; t2: receive message; t3: continue (upper level); t4: send ACK; t5: continue (lower level); t6: receive ACK.

Let us denote by  $\mathbf{x}(k) = [x_1(k) \ \dots \ x_6(k)]^T$  the vector having as components the time moments  $x_i(k)$  of the  $k$ -th firing of transition  $t_i$ , for  $i = 1, \dots, 6$ . The max-plus state-space representation in implicit form (1) is given by

$$\mathbf{x}(k) = \mathbf{A}_0 \otimes \mathbf{x}(k) \oplus \mathbf{A}_1 \otimes \mathbf{x}(k-1), \quad (9)$$

where the numerical values of matrices  $\mathbf{A}_0$  and  $\mathbf{A}_1$ , automatically derived by the PN Toolbox, are presented in figure 2.

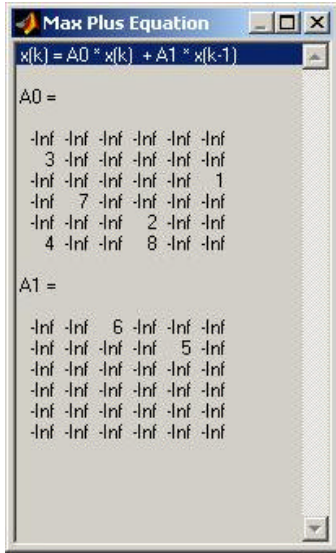


Fig. 2. Screen capture of the window opened by PN Toolbox for the max-plus state-space representation in Example 1

Introducing matrix  $\mathbf{A} = \mathbf{A}_0^* \otimes \mathbf{A}_1$ , equation (9) turns into the explicit form

$$\mathbf{x}(k) = \mathbf{A} \otimes \mathbf{x}(k-1). \quad (10)$$

The analytical study of the behavior of this autonomous timed event graph begins with the derivation of the initial conditions corresponding to equation (10). The only transition that is 1-validated and has to be executed once at

the initial moment  $t_0 = 0$  is t1; therefore, according to the algorithm presented in Section IV, vector  $\bar{\mathbf{x}}(1) = [0 \ e \ e \ e \ e \ e]^T$  is constructed. The time moments of the first execution of all the transitions in the model are given by

$$\mathbf{x}(1) = \mathbf{A}_0^* \otimes \bar{\mathbf{x}}(1), \quad (11)$$

while vectors  $\mathbf{x}(k)$ , for  $k \geq 2$ , may be successively computed by means of equation (10). The results obtained through simulation matched up the ones derived via the analytical method. Figure 3 presents the graphical plot of firing time vs. firing count for transition t1 associated with the occurrence of the “send message” event. Furthermore, the firing instants are available for all the transitions of the net in a numerical format (see the window located at the lower left corner of figure 3). The period of 25 time units exhibited by the numerical and graphical information has been confirmed via the direct computation of the largest eigenvalue of  $\mathbf{A}$  in explicit equation (11).

*Example 2* (adapted from [1]) represents a manufacturing system consisting of three machines, denoted by  $M_i$ ,  $i = 1, 2, 3$ . It is supposed to produce three kinds of parts, namely  $P_i$ ,  $i = 1, 2, 3$ , according to a certain product mix. The routes to be followed by each part and each machine are depicted in Figure 4 and the corresponding processing times are given in Table 1.

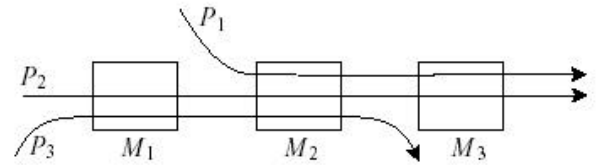


Fig. 4. Routing of parts along machines in Example 2

TABLE 1. Processing times of parts on machines in Example 2

	$P_1$	$P_2$	$P_3$
$M_1$	-	1	5
$M_2$	3	2	3
$M_3$	4	3	-

Note that this manufacturing system has a flow-shop structure, i.e. all parts follow the same sequence on the machines (although they may skip some) and every machine is visited at most once by each part.

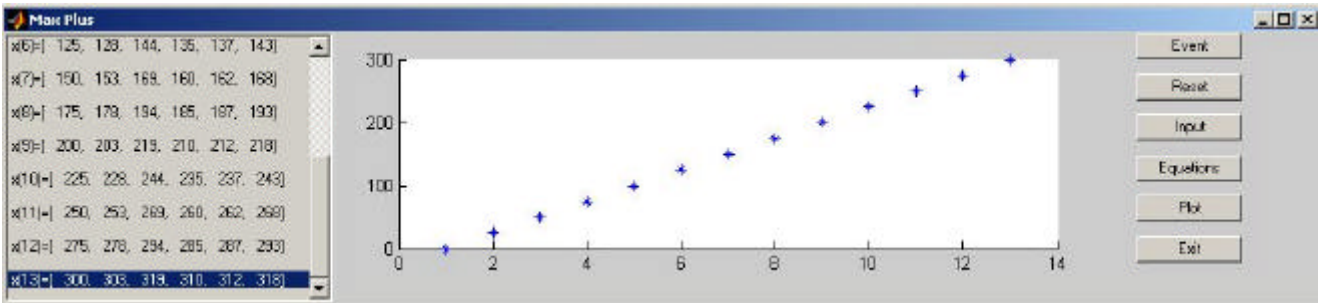


Fig. 3. Screen capture of the graphical plot of firing time vs. firing count for transition t1 in Example 1



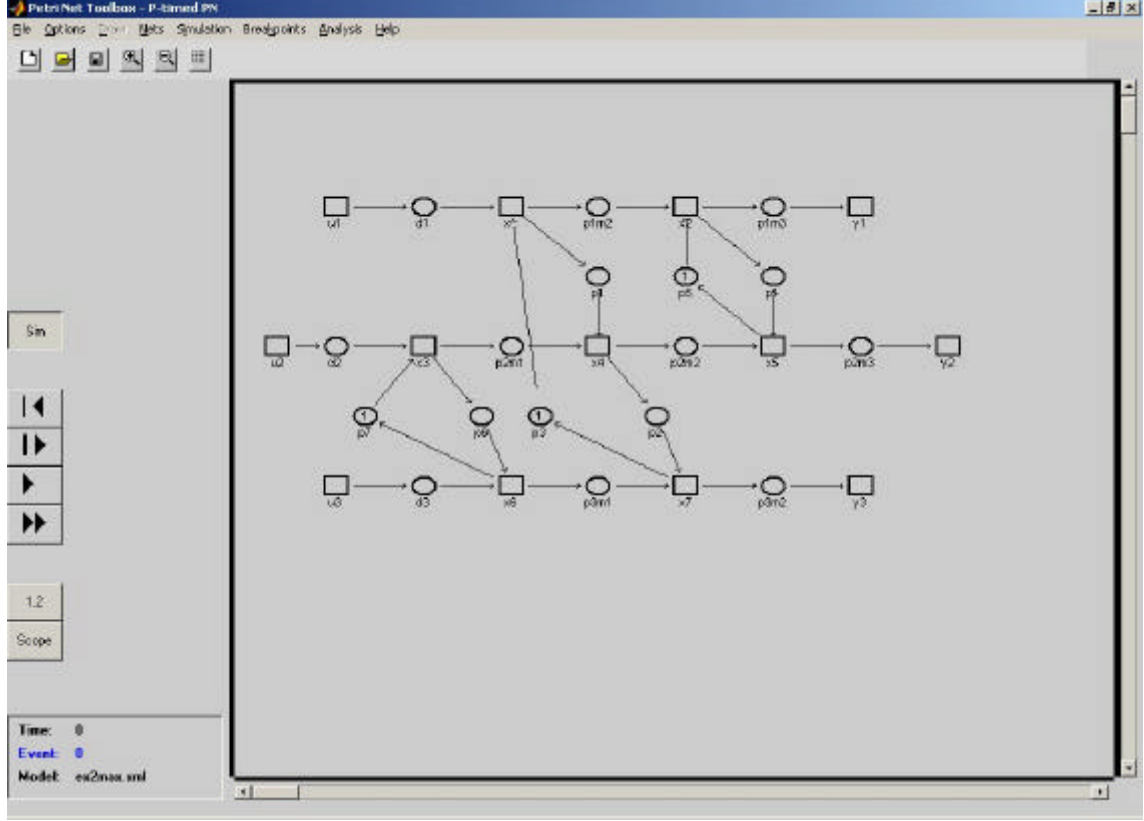


Fig. 5. Screen capture of the main window of PN Toolbox with the event graph in Example 2

We assume that there are no set-up times on machines when they switch from one part type to another and also no traveling times for parts between the machines. Parts are carried on a limited number of pallets; only one pallet is available for each part type. The sequencing of part types on the machines is known and it is  $(P2, P3)$  on  $M1$ ,  $(P1, P2, P3)$  on  $M2$  and  $(P1, P2)$  on  $M3$ . The final product mix can be obtained by means of a given input of parts.

The place timed Petri net model is presented in the main window of PN Toolbox captured in figure 5. The initial marking complies with the sequencing of parts on machines, so that machine  $M1$  starts working on product  $P2$  and  $M2$  on  $P1$ . The following numerical values have been used for the time duration allocated to a place  $p$ , generically denoted by  $d(p)$ :

- input of type  $i$  part:  $d(d1) = d(d2) = d(d3) = 0$  ;
- processing of  $P1$  on  $M2$ :  $d(p1m2) = d(p1) = 3$  ;
- processing of  $P1$  on  $M3$ :  $d(p1m3) = d(p4) = 4$  ;
- processing of  $P2$  on  $M1$ :  $d(p2m1) = d(p6) = 1$  ;
- processing of  $P2$  on  $M2$ :  $d(p2m2) = d(p2) = 2$  ;
- processing of  $P2$  on  $M3$ :  $d(p2m3) = d(p5) = 3$  ;
- processing of  $P3$  on  $M1$ :  $d(p3m1) = d(p7) = 5$  ;
- processing of  $P3$  on  $M2$ :  $d(p3m2) = d(p3) = 3$  .

This timed event graph has three source transitions ( $u1$ ,  $u2$  and  $u3$ ) whose firing instants (considered a priori known) make up input vector  $\mathbf{u}(k) = [u_1(k) \ u_2(k) \ u_3(k)]^T$ . Output

vector  $\mathbf{y}(k) = [y_1(k) \ y_2(k) \ y_3(k)]^T$ , that corresponds to the sink transitions  $y1$ ,  $y2$  and  $y3$ , is constructed in a similar manner. The components of the state vector  $\mathbf{x}(k)$  represent the  $k$ -th firing moments of transitions  $t1, \dots, t7$ .

The implicit form of the max-plus state-space representation is given by

$$\mathbf{x}(k) = \mathbf{A}_0 \otimes \mathbf{x}(k) \oplus \mathbf{A}_1 \otimes \mathbf{x}(k-1) \oplus \mathbf{B}_0 \otimes \mathbf{u}(k), \quad (12)$$

$$\mathbf{y}(k) = \mathbf{C}_0 \otimes \mathbf{x}(k). \quad (13)$$

The numerical values of the matrices involved in the previous equations, automatically derived by the PN Toolbox, are presented in figure 6. The explicit form of equation (12) is

$$\mathbf{x}(k) = \mathbf{A} \otimes \mathbf{x}(k-1) \oplus \mathbf{B} \otimes \mathbf{u}(k). \quad (14)$$

where  $\mathbf{A} = \mathbf{A}_0^* \otimes \mathbf{A}_1$  and  $\mathbf{B} = \mathbf{A}_0^* \otimes \mathbf{B}_0$ .

Since none of the transitions in the PN model is validated at the initial moment  $t_0 = 0$ , the state vector  $\mathbf{x}(1)$  depends only on the first firing instances of the input transitions represented by  $\mathbf{u}(1)$ . According to the algorithm presented in Section IV, it is necessary to first compute vector

$$\mathbf{v}(1) = \mathbf{B}_0 \otimes \mathbf{u}(1), \quad (15)$$

and next:

$$\mathbf{x}(1) = \mathbf{A}_0^* \otimes \mathbf{v}(1). \quad (16)$$

Equation (13) gives for  $k=1$  the first epochs at which the parts are ready and the machines have finished their jobs (for one cycle). For  $k \geq 2$ ,  $\mathbf{x}(k)$  and  $\mathbf{y}(k)$  may be successively computed by means of equations (12) and (13).

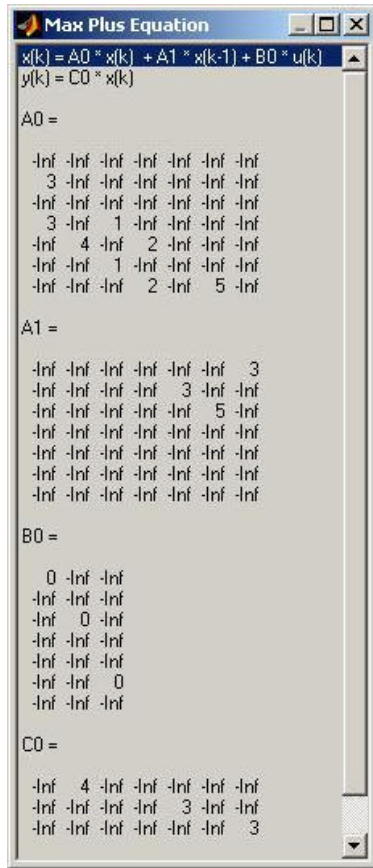


Fig. 6. Screen capture of the window opened by PN Toolbox for the max-plus state-space representation in Example 2

Figure 7 presents the graphical plot of firing time vs. firing count for transitions  $u_1$ ,  $t_1$  and  $y_1$ , associated to the occurrences of “input part type P1”, “begin processing part type P1” and “output part type P1” events, obtained through simulation for the following numerical values of the input vector  $u(1) = [0, 0, 0]^T$ ,  $u(2) = [4, 5, 4]^T$ ,  $u(3) = [10, 12, 12]^T$ ,  $u(4) = [19, 20, 18]^T$ ,  $u(5) = [27, 28, 26]^T$ ,  $u(6) = [37, 30, 34]^T$  and  $u(7) = [50, 41, 42]^T$ .

## VI. CONCLUSIONS

The MATLAB environment provides a large number of computational resources and programming facilities, which allowed us to develop a simulator for discrete event

systems whose dynamics is described by max-plus linear models. This paper presents the simulator within the context of PN Toolbox applications, but it can also be used as a separate tool embedded in MATLAB. The purpose of our work was twofold, namely to enlarge the functionality of MATLAB in the case of discrete event systems and to provide a software instrument for easy handling this special class of system descriptions.

## REFERENCES

- [1] F. Baccelli, G. Cohen, G.J. Olsder and J.P. Quadrat, Synchronization and linearity, an algebra for discrete event systems, Wiley. New York, 1992.
- [2] C.G. Cassandras, S. Lafortune and G.J. Olsder, “Introduction to the modeling, control and optimization of discrete event systems”. In: (A. Isidori, Ed.) Trends in Control – A European Perspective, Springer, Berlin, 1995.
- [3] G. Cohen, D. Dubois, J.P. Quadrat, and M. Viot, “A linear-system-theoretic view of discrete-event processes and its use for performance evaluation in manufacturing”. IEEE Trans. Automat. Control, 30, pp. 210-220, 1985.
- [4] G. Cohen, S. Gaubert, and J.P. Quadrat, “Max-plus algebra and system theory: where we are and where to go now”, IFAC Conference on System Structure and Control, Nantes, France, July 8-10, 1998.
- [5] Scilab Home Page, <http://www-rocq.inria.fr/scilab/>.
- [6] G. Cohen, S. Gaubert, and J.P. Quadrat, “Max-plus algebra in Scilab and applications”, Scilab Workshop, Liama, Pekin, April 2001.
- [7] The MathWorks Inc., Building GUIs with MATLAB. Natick, Massachusetts, 2000.
- [8] The MathWorks Inc., Using MATLAB graphics. Natick, Massachusetts, 2000.
- [9] C. Mahulea, Petri Net Toolbox – a MATLAB library for discrete event systems, MS thesis, The Sheffield University, Department of Automatic Control & System Engineering, UK (ERASMUS / SOCRATES grant), 2002.
- [10] C. Mahulea, L. Bărsan and O. Pastravanu, “Matlab tools for Petri-net-based approaches to flexible manufacturing systems”. In: (F.G. Filip, I. Dumitrache and S. Iliescu, Eds.) Proceedings Volume from the 9th IFAC Symposium on Large Scale Systems LSS 2001, Bucharest, Romania, 18-20 July 2001, pp. 184-189, 2001.
- [11] M. Matcovschi, C. Mahulea, and O. Pastravanu, “Exploring Structural Properties of Petri Nets in MATLAB”, 7th International Symposium on Automatic Control and Computer Science SACCs 2001, October 26-27, 2001, Iasi, Romania, CD Rom, 2001.
- [12] M. Matcovschi and C. Mahulea, “Generalized stochastic Petri nets in performance evaluation for queueing networks”, Symposium on Intelligent Software and Control Systems SISCS 2002, July 18-19, 2002, Iasi, Romania, CD-Rom, 2002.
- [13] A.A. Desrocheres and R.Y. Al-Jaar. Modeling and control of automated manufacturing systems. IEEE Computer Society Press, Rensselaer, Troy, New-York, 1993.

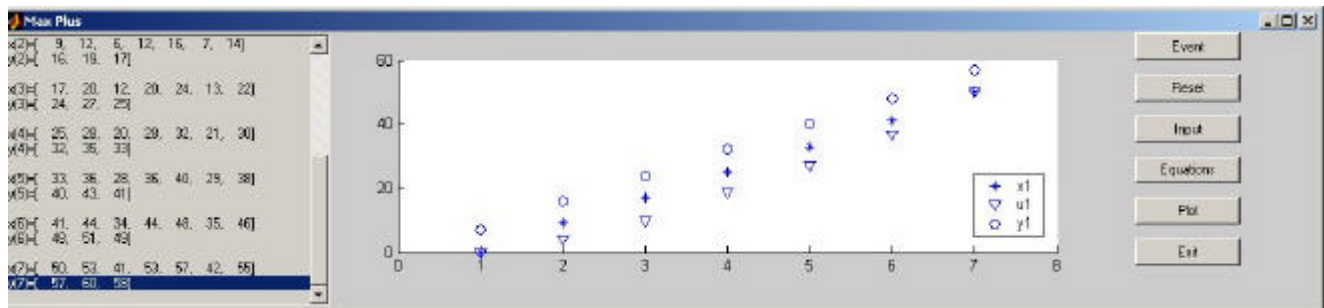


Fig. 7. Screen capture of the graphical plot of firing time vs. firing count for transitions  $t_1$ ,  $u_1$  and  $y_1$  in Example 2