

On Solving Controlled-Invariance Problems in Dioids Using the PyMinMaxGD Python Scripts Library

Olivier Boutin¹, Claude Martinez² and Naly Rakoto³

¹LS2N - UMR6004, 3iL Ingénieurs, Limoges, France

²LS2N - UMR6004, Nantes Université, École Centrale Nantes, F-44000 Nantes, France

³LS2N - UMR6004, IMT Atlantique, Nantes, France

Keywords: Modelling and Control of Discrete-Event Systems, Supervision Systems, Systems Synchronisation, Controlled Invariance, Manufacturing Process Management, Dioids, Periodic Series, Algebraic Toolbox, C++, Python, Object-Oriented Programming and Scripting Programming.


Abstract: MinMaxGD is a C++ library developed and maintained since 2000 at the LARIS laboratory at Angers, France. It provides ad-hoc primitives for calculations so as to handle periodic series over a dioid (namely $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]$). Our aim with PyMinMaxGD is to provide, for this C++ library, a Python interface using SWIG, in order to allow scripting programming, on top of the already available programming primitives. The use of PyMinMaxGD is illustrated here to tackle some general problems of control using the controlled-invariance theory. The corresponding control problems deal with the meeting of time or marking constraints or even both of them. First, the set that corresponds to the verification of the constraints is defined, and then the supremal controlled-invariant set included in this specification set is calculated. In the sequel, a controller that allows the state of the system to remain in the latter set is given.


1 INTRODUCTION


To the best of our knowledge, the main library to provide ad-hoc primitives so as to handle periodic series over a dioid is MinMaxGD (Hardouin, 2024), a C++ library developed and maintained since 2000 (Cottenceau et al., 2000) at the LARIS laboratory at Angers, France¹. A JavaScript interface, called MinMaxGDJS, has also been developed in 2017 (Ferreira Cândido et al., 2017), giving the ability to access the periodic-series library through any Web browser on any operating system. This makes it easier to get insight on calculations with periodic series, without the hassle of configuring and building the original C++ library. Another interest is to provide a platform for giving portable demos. Nevertheless, due to its locked Web-based interface, it lacks the power to embed the calculations within some other programming processings. Our aim with PyMinMaxGD is then to

provide a different interface for the original library, meant for Python² and using SWIG (SWIG Maintainers, 2024), an open-source software tool used to connect libraries written in C++ with scripting languages, in order to allow scripting programming. The produced code keeps the very same computation benefits as for the original library, extending its portability to a scripting programming paradigm.

As a first attempt to develop research on the control of discrete-event systems using the representation of their behaviour, within the framework of series in the $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]$ dioid focusing on the state of a system rather than its transfer function, some aspects of the use of series for the controlled-invariance theory are presented. The concept of controlled invariance, or (A, B) -invariance, has been introduced independently by (Basile and Marro, 1969) and by (Wonham and Morse, 1970). The whole geometric approach for control, based on the then new algebraic geometry theory (Grothendieck and Dieudonné, 1960), has started from these works. The solution of many classical control problems, among which the

^a <https://orcid.org/0009-0000-7292-0672>

^b <https://orcid.org/0000-0003-1168-3284>

^c <https://orcid.org/0000-0002-5798-0359>

¹See Section 4.1 for detailed information and a comparison between such toolboxes.

²What is actually called an "extension module" in the Python ecosystem.

disturbance decoupling problem, the regulator problem, and various observer design problems, were given using this controlled-invariance theory – see for instance (Wonham, 1974) and (Basile and Marro, 1992) for a complete account of these results. The controlled-invariance theory has been applied also in the framework of systems over more general algebraic contexts than fields, such as rings or semirings. For the sake of extending the classical results in the latter structures, the concept of controlled-invariant module has been introduced; (Hautus, 1982) and (Conte and Perdon, 1995) are pioneering references for invariance control over a ring.

Finally, systems over a semiring, often called “Max-Plus³ systems”, have been thoroughly studied in the founding reference (Baccelli et al., 1992). They have also received attention from the perspective of controlled-invariance theory, the first results being due to (Katz, 2007). It is stated there that there exists a maximal controlled-invariant module in every given module in \mathbb{R}_{\max} and many other semirings. It is pointed out that the computation of this maximal set is an open problem in general and important particular cases where the problem is solvable are identified. Sufficient conditions that allow this computation are often met in practice. In (Cárdenas et al., 2015), it was shown that a feedforward control law is suitable to maintain the state of a system within the controlled-invariant set. In (Cárdenas et al., 2017), a procedure to find the greatest controlled-invariant set included in the admissible set defined by some constraints has been presented. There, it has also been shown that a static non linear state feedback control law is suitable to maintain the state of a system within the controlled-invariant set. Some other problems of control have been tackled in (Martinez et al., 2022) and (Animobono et al., 2022). All the aforementioned work on the use of controlled-invariance theory have been done within the canonical \mathbb{Z}_{\max} and \mathbb{Z}_{\min} dioid frameworks. Based on our information, there is no research that has tackled the use of controlled-invariance theory in the context of a dioid of series such as $\mathcal{M}_{\text{in}}^{\text{ax}}[[\gamma, \delta]]$. The purpose of the control problems given in the present article is to illustrate, for the first time, that controlled invariance can also be useful for the class of systems that are modelled with series, and particularly using the C++ MinMaxGD library.

The remainder of this article is organised as follows: the following section will briefly define and describe the theoretical elements based on which our modelling framework works; then, Section 3 will introduce how control theory can be applied in this

mathematical context; afterwards, in Section 4, more insight will be given on other toolboxes for such work and why none of them suits our needs, as well as more details on the features offered by our developments; subsequently, Section 5 will provide two use cases of our library, based on a typical problem of a physical system having to follow a specification, reinterpreted as a synchronisation problem and a time constraint meeting, and within the specific $\mathcal{M}_{\text{in}}^{\text{ax}}[[\gamma, \delta]]$ dioid, including detailed Python scripts; finally, a last section will give some conclusions and openings of this work.

2 BACKGROUND

2.1 Dioids and Systems Representation

A dioid is a set, denoted \mathcal{D} , endowed with two binary operations, a sum (\oplus) and a product (\otimes). The complete definition of a dioid (or idempotent semiring) can be found, for example, in (Brunsch et al., 2013, Section 1.3). The identity element of \oplus , ϵ , is also called the *zero element* of \mathcal{D} , while the identity element of \otimes , e , is also called the *unit element* of \mathcal{D} .

Due to the idempotency property, any dioid can also be endowed with a natural (partial) order defined by $a \leq b \Leftrightarrow a \oplus b = b$.

As in a classical field, the binary operations can be extended to the matrix case in dioids (Cohen et al., 1989, p.42). For the matrices $A, B \in \mathcal{D}^{n \times m}$ and $C \in \mathcal{D}^{m \times p}$ one gets

$$\begin{aligned} \text{(sum)} \quad & [A \oplus B]_{ij} = [A]_{ij} \oplus [B]_{ij}; \\ \text{(product)} \quad & [A \otimes C]_{ij} = \bigoplus_{k=1}^m ([A]_{ik} \otimes [C]_{kj}). \end{aligned}$$

Remark 2.1. *It is worth noticing that the product of $\mathcal{D}^{n \times m}$ is not always commutative, i.e. $A \otimes B$ may be different from $B \otimes A$.*

Let us now introduce a 2-dimensional dioid denoted $\mathbb{B}[[\gamma, \delta]]$ (Cohen et al., 1989). It is the set of formal power series in two variables γ and δ with Boolean coefficients, i.e. $\mathbb{B} = \{\epsilon, e\}$ and the exponents of variables γ and δ are in \mathbb{Z} .

The interpretation⁴ of a monomial $\gamma^k \delta^{k'}$ is that, considering a previous monomial $\gamma^{k'} \delta^{k'}$, the *epoch* being $\gamma^0 \delta^0$, $k - k'$ events occur after $t - t'$ units of time, which leads to a compact modelling of information. Graphically, a series can be seen as a set of dots of a point cloud in the *event-time plane* (see Figure 1)⁵.

⁴Following the discussion in (Baccelli et al., 1992, Section 5.4.4).

⁵Bearing in mind that the time part of the modelling is now an ordinate.

³Often denoted thanks to \mathbb{Z}_{\max} or \mathbb{R}_{\max} for example, depending on the exact set of studied values.

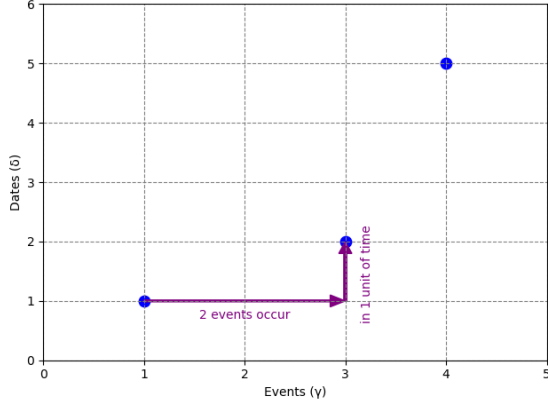


Figure 1: Graphical representation of series $s = \gamma^1 \delta^1 \oplus \gamma^3 \delta^2 \oplus \gamma^4 \delta^5, s \in \mathbb{B}[\gamma, \delta]$.

Actually, the “South-East cone” of each point in the plane matches a corresponding monomial of a series. This establishes an equivalence relation and implies that series may have a “minimal” representation (Cohen et al., 1989, Comment ii) p.51), based on the “absorption” of some monomials by others. The resulting dioid of equivalence classes in $\mathbb{B}[\gamma, \delta]$ is denoted $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]$.

Remark 2.2. In order to fully define $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]$, its zero and unit elements are defined as $\varepsilon = \gamma^{+\infty} \delta^{-\infty}$ and $e = \gamma^0 \delta^0$ respectively⁶ (Cohen et al., 1986, p. 991).

In the next section, complex systems described by $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]$ series are shown to have their graphical counterparts in the form of Timed Event Graphs, which also consists in compact modelling.

2.2 Timed Event Graphs

Timed Event Graphs (TEGs) define a class of timed Petri nets where there are no conflicts (Ramchandani, 1974, Chapter 4).

Figure 2 shows two examples of TEGs. Please note that transition t_1 is *downstream* of t_2 , for reasons out of the scope of this study.

2.3 Modules in $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]$

The concept of modules on a semiring is analogous to the concept of vector spaces on a field. Modules over a semiring are often called *semimodules*, and modules over a dioid are sometimes called *moduloid*, but the term *module* shall rather be used in this article, following the track of (Wagneur, 1996). Here, modules and submodules of $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]$, and more specifically

⁶ $+\infty$ and $-\infty$ being respectively the zero element of \mathbb{Z}_{min} and \mathbb{Z}_{max} . 0 is their common unit element.

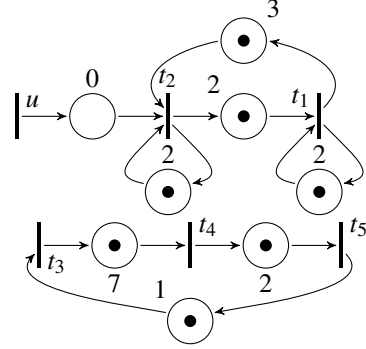


Figure 2: Two TEGs. The TEG above is meant to be synchronised with a takt time (Haghsheno et al., 2016) given by the TEG below, thanks to the study to come in Section 5.

finitely generated modules, i.e. those that are generated with a finite family of vectors of series defined in $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]$, are of high interest.

Let us consider a matrix M of size $n \times m$ with its entries in $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]$, n and m being positive integers. Let us denote $\text{Im } M$ as the submodule of $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^n$ that is generated by the columns of M , hence $\text{Im } M = \{x \in \mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^n \mid \exists v \in \mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^m, x = Mv\}$. By definition, a submodule \mathcal{M} of $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^n$ is *finitely generated* if there exists a positive integer q and a matrix $M \in \mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^{n \times q}$ such that $\mathcal{M} = \text{Im } M$. Finitely generated modules over $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^n$ may be represented in several ways. One can refer to the original results of (Butkovič and Hegedüs, 1984) where it is established that the family of finitely generated submodules of $\mathbb{R}_{\text{max}}^n$ coincides with the family of finitely generated cones that are sets of the form $\text{Cone}(R, Q) = \{x \in \mathbb{R}_{\text{max}}^n \mid R \otimes x = Q \otimes x\}$ where R and Q are matrices of size $p \times n$, p being a positive integer. This result is applied on submodules of $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^n$.

Theorem 2.1. Given a module $\mathcal{M} \subset \mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^n$, the two following assertions are equivalent.

- (i) There exist a positive integer q and a matrix $M \in \mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^{n \times q}$ such that $\mathcal{M} = \text{Im } M$.
- (ii) There exist a positive integer p and matrices $R, Q \in \mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^{p \times n}$ such that $\mathcal{M} = \text{Cone}(R, Q)$.

Algorithms that permit to switch from a cone representation to an image representation have been presented first in (Butkovič and Hegedüs, 1984), and refined by Allamigeon et al. in (Allamigeon et al., 2010).

For any set $S \subset \mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^m$, its image by M is denoted by MS . The preimage by M of any set $\mathcal{T} \subset \mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^p$ is denoted $M^{-1}\mathcal{T}$. The difference of two sets $S, S' \subset \mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^m$, denoted $S \ominus S'$ is defined as $\{x'' \in \mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]^m \mid \exists x \in S, x' \in S', x'' \oplus x' = x\}$.

3 CONTROL PROBLEMS AND ALGORITHMS

3.1 Controlled Invariance

Let us consider here a controlled system of the form:

$$x = Ax \oplus Bu, \quad (1)$$

where $A \in \mathcal{M}_{in}^{ax}[\gamma, \delta]^{n \times n}$, $B \in \mathcal{M}_{in}^{ax}[\gamma, \delta]^{n \times m}$, n and m are positive integers, and $x \in \mathcal{M}_{in}^{ax}[\gamma, \delta]^n$ and $u \in \mathcal{M}_{in}^{ax}[\gamma, \delta]^m$. The variable x refers to the state trajectory of the system, and u is its control input.

Note that, in opposition to systems over \mathbb{R}_{max} , there is no specific mention of the initial state x_0 because each entry of the state vector is a series that represents all the trajectory, including the initial state. The following definition of controlled invariance is a transcription of the definition stated for systems over \mathbb{R}_{max} , see for instance (Martinez et al., 2022).

Definition 3.1. A set \mathcal{M} is said to be controlled-invariant if, for every vector $x \in \mathcal{M}$, there exists a control u such that the solution of system (1) is entirely in \mathcal{M} .

Research about controlled-invariant sets properties are still in progress. Nevertheless, some properties are already well known; for instance, the case of the submodules of \mathbb{R}_{max}^n , or over various other semirings, has particularly been studied. See (Katz, 2007) and (Di Loreto et al., 2010), for instance.

Proposition 3.1. A set $\mathcal{M} \subset \mathcal{M}_{in}^{ax}[\gamma, \delta]^n$ is controlled-invariant if and only if the following inclusion is satisfied: $A\mathcal{M} \subset \mathcal{M} \ominus \text{Im}B$. Equivalently, \mathcal{M} is controlled-invariant if and only if the following inclusion is satisfied:

$$\mathcal{M} \subset A^{-1}(\mathcal{M} \ominus \text{Im}B).$$

Proof By definition, the set \mathcal{M} is controlled-invariant if each of its elements x satisfies $Ax \oplus Bu \in \mathcal{M}$ for some vector $u \in \mathcal{M}_{in}^{ax}[\gamma, \delta]^m$. The result follows since Bu remains in $\text{Im}B$ when u varies, and by definition of \ominus (Katz, 2007, Section 2).

Theorem 3.1. The following properties are verified (Cárdenas et al., 2017).

(i) A module $\mathcal{M} \subset \mathcal{M}_{in}^{ax}[\gamma, \delta]^n$ is controlled-invariant if and only if:

$$A\mathcal{M} \subset \mathcal{M} \ominus \text{Im}B.$$

(ii) A module $\mathcal{M} \subset \mathcal{M}_{in}^{ax}[\gamma, \delta]^n$ generated by the columns of matrix $M \in \mathcal{M}_{in}^{ax}[\gamma, \delta]^{n \times q}$ is controlled-invariant if and only if there exist matrices $U \in \mathcal{M}_{in}^{ax}[\gamma, \delta]^{m \times q}$ and $V \in \mathcal{M}_{in}^{ax}[\gamma, \delta]^{q \times q}$ such that the following identity is verified:

$$AM \oplus BU = MV.$$

(iii) A module $\mathcal{M} \subset \mathcal{M}_{in}^{ax}[\gamma, \delta]^n$ such that $\mathcal{M} = \text{Im}M = \text{Cone}(R, Q)$, for matrices $M \in \mathcal{M}_{in}^{ax}[\gamma, \delta]^{n \times q}$ and $R, Q \in \mathcal{M}_{in}^{ax}[\gamma, \delta]^{p \times n}$ is controlled-invariant if and only if there exists a matrix $U \in \mathcal{M}_{in}^{ax}[\gamma, \delta]^{m \times q}$ such that the following equality is verified:

$$R(AM \oplus BU) = Q(AM \oplus BU).$$

Not all submodules of $\mathcal{M}_{in}^{ax}[\gamma, \delta]^n$ are controlled-invariant. But one can show that any module $\mathcal{M} \subset \mathcal{M}_{in}^{ax}[\gamma, \delta]^n$ contains a maximal controlled-invariant submodule, denoted $\mathcal{V}_{\mathcal{M}}^*(A, B)$. It is shown in (Di Loreto et al., 2010) that the supremal controlled-invariant submodule included in a finitely generated module is the limit of the sequence $\{\mathcal{V}_i\}$ defined as follows:

$$\mathcal{V}_0 = \mathcal{M}; \quad \mathcal{V}_{i+1} = \mathcal{M} \cap A^{-1}(\mathcal{V}_i \ominus \text{Im}B), \quad \text{for } i \geq 0. \quad (2)$$

The successive terms of the sequence are finitely generated modules.

As it is mentioned in (Cárdenas et al., 2017), the matrices M_i and (R_i, Q_i) can be computed iteratively such that $\mathcal{V}_i = \text{Im}M_i = \text{Cone}(R_i, Q_i)$ seeking for a solution of the following equation:

$$\begin{pmatrix} R_{i-1}A & R_{i-1}B \\ R & \epsilon \end{pmatrix} \begin{pmatrix} M_i \\ U_i \end{pmatrix} = \begin{pmatrix} Q_{i-1}A & Q_{i-1}B \\ Q & \epsilon \end{pmatrix} \begin{pmatrix} M_i \\ U_i \end{pmatrix}. \quad (3)$$

At each step, one computes M_i and U_i and, then, one computes R_i et Q_i using theorem 2.1. The limit of the sequence is the intersection of modules \mathcal{V}_i .

Finally, if for some integer i , the equality $\mathcal{V}_i = \mathcal{V}_{i+1}$ is true, then the sequence stabilises, which leads to $\mathcal{V}_k = \mathcal{V}_i$, for every $k \geq i$, and therefore $\mathcal{V}_{\omega} = \mathcal{V}_i$. Following (2), $\mathcal{V}_{\omega} = \mathcal{M} \cap A^{-1}(\mathcal{V}_{\omega} \ominus \text{Im}B)$ can be deduced. Thus, using Proposition 3.1, \mathcal{V}_{ω} can be considered as controlled-invariant set, which is included in \mathcal{M} and, therefore, it is included into the maximal controlled-invariant set $\mathcal{V}_{\mathcal{M}}^*(A, B)$.

3.2 Some Control Problems

The first problem tackled here and illustrated in the example in subsection 5.1 is a problem of synchronisation of two systems, say (Σ_1) and (Σ_2) , that are defined as follows:

$$\begin{aligned} (\Sigma_1) \quad & \begin{cases} x &= A \otimes x \oplus B \otimes u \\ y &= C \otimes x \end{cases}, \\ (\Sigma_2) \quad & \begin{cases} w &= D \otimes w \\ v &= E \otimes w \end{cases}. \end{aligned}$$

Several variants of this problem have been studied and presented in (Martinez et al., 2022) in the \mathbb{R}_{max}^n dioid. Systems (Σ_1) and (Σ_2) are said to have *synchronous outputs* if their output solutions coincide,

i.e. $y = v$. When no input is applied (say $u = \varepsilon$), systems are said *synchronous* if $y = v$. In practical situations in industry, the notion of synchronisation is related to the takt time that somehow defines the tempo of the factory (Haghsheno et al., 2016). The problem that is addressed here is that of the *forced synchronisation*, which is formulated as follows.

Problem 3.1. (*Forced synchronisation*)

System (Σ_1) and (Σ_2) are synchronised if there exists a control law u that forces synchronisation between them, i.e. the outputs of both systems (Σ_1) and (Σ_2) are equal: $y = v$. Such a control u is said to be admissible for the synchronisation of (Σ_1) and (Σ_2) .

In (Martinez et al., 2022), the forced synchronisation problem was slightly different, seeking for a given k_0 , after which synchronisation would be possible; then the system was called k_0 -synchronisable. Indeed, state trajectories that verify the solution of problem 3.1 can be considered as the target to be reached, seeking for a set of coreachable x_p and u_p .

This extension needs more research work in the $\mathcal{M}_{in}^{ax}[\gamma, \delta]$ dioid.

The second problem tackled here and illustrated in the second example in subsection 5.2 concerns the verification of time constraints related to bounded duration for some operations. This problem has been addressed in (Kim and Lee, 2016) and continued in (Jacob and Amari, 2017). Then, it has been formulated as a controlled-invariant problem in (Cárdenas et al., 2017). In this problem, the constraints limit the maximal duration of certain tasks; that is to say, the minimal duration is given by the delay τ associated to a place of a TEG and the constraint to be verified is that the sojourn time of tokens is lower than a certain τ_{max} , considering the following system (Σ) :

$$(\Sigma) \quad \begin{cases} x &= A \otimes x \oplus B \otimes u, \\ y &= C \otimes x. \end{cases}$$

Problem 3.2. (*Constraints verification*) *The series x that models the state behaviour should verify some additional constraints of the form:*

$$(\text{duration}) \quad \gamma^0 \delta^{-\tau_{max}} \otimes x_i \leq x_j. \quad (4)$$

To take into account several constraints, then, for each of them, one has to write a single inequality as in (4). These inequalities are gathered in a matrix inequality as follows:

$$E \otimes x \leq x.$$

The set of solutions of this inequality is $\text{Im } E^*$.

4 TOOLBOXES FOR CALCULATIONS OVER DIOIDS

4.1 Existing Toolboxes

Almost a dozen of initiatives have been encountered while looking for a toolbox that could suit our needs. Table 1 below summarises all these initiatives and projects, for the ease of comparison⁷.

As a matter of facts, none of them actually include the four following characteristics together, that are definitely needed for our research:

1. including both dating and counting dynamics at the same time;
2. possibility to integrate self-defined routines, like plug-ins, without having the burden of compiling the core components every time, for flexibility;
3. possibility to automatise simulations benchmarks, and re-run parts of them interactively, as with scripts;
4. interaction with other pieces of software.

For all these reasons, the development of our own toolbox, described in the following subsection, has been decided.

4.2 The Proposed Toolbox

The distribution of our piece of software is available online (Boutin and Martinez, 2024), under GPLv3 license (Free Software Foundation, Inc., 2007). It mainly consists of a single SWIG interface file of about 900 lines of code so far, on top of the Min-MaxGD library, based on roughly 5 000 lines of code spread over 11 files defining 4 classes and a few high-level functions and constants. It has been compiled and installed⁸ with the following software requirements: 6.1 64-bit (SLTS⁹) Linux@ kernel; g++ 11.3.0; SWIG 4.0.2; Python 3.11.0; the libpython3.11-dev and python-dev-is-python3 Linux packages and the matplotlib, numpy, pillow and kiwisolver Python libraries.

Apart from the implementation of the scripts designed to support the algorithms mentioned in Sec-

⁷Note that symbol \mathcal{B} there stands for the Boolean set endowed with two binary operations “and” and “or” and only composed of the elements e and ε .

⁸Only once for the inclusion of the core toolbox, as opposed to the need to compile C++-only code for each new calculation design or update; then only the user-defined scripts need to be updated and loaded interactively.

⁹Meaning *Super Long Time Support* (Civil Infrastructure Platform™, 2023).

Table 1: Comparison of available dioid toolboxes.

Name	Implementation	Evolution from	Available dioids	Last version ¹⁰
MinMaxGD (Hardouin, 2024)	C++		$\mathcal{M}_{in}^{ax}[\gamma, \delta]$	03 Jul. 2024
ContainerMinMaxGD (Corrionc, 2013)	C++	Fork of MinMaxGD	Intervals in $\overline{\mathbb{Z}}_{min}$	21 Apr. 2011
MinMaxGDJS (Ferreira Cândido et al., 2017)	JavaScript wrapping over C++	Fork of MinMaxGD	$\mathcal{M}_{in}^{ax}[\gamma, \delta]$	03 May 2019
ETVO (Cottenceau et al., 2022)	C++ / WebAssembly wrapping	Fork of MinMaxGD	$\mathcal{M}_{in}^{ax}[\gamma, \delta]$ and $\mathcal{E}T$ (see (Trunk, 2019, Chap. 5))	20 May 2023
Maxpluspy (Lahaye, 2019)	Python		$\overline{\mathbb{Z}}_{max}$ and “ $\overline{\mathbb{Z}}_{max}$ -automata”	27 Nov. 2019
PetriTUB (Bednar et al., 2024)	Python		$\overline{\mathbb{Z}}_{max}$ and $\overline{\mathbb{Z}}_{min}$, from Petri nets	14 Mar. 2024
MaxPlus Arithmetic Toolbox (Chancelier et al., 2015)	Already included in Scicoslab Toolbox	Fork of Scilab 4	$\overline{\mathbb{Z}}_{max}$	03 Oct. 2015
MaxPlus.jl (Quadrat, 2024)	Julia	Port from MaxPlus Arithmetic Toolbox	$\overline{\mathbb{Z}}_{max}$ and $\overline{\mathbb{Z}}_{min}$	30 Apr. 2024
Max-Plus Algebra Toolbox (Stańczyk, 2016)	Matlab/Octave plug-in		$\overline{\mathbb{Z}}_{max}$ and $\overline{\mathbb{Z}}_{min}$	14 Jun. 2016
TropicalNumbers (Liu, 2023)	Julia		$\overline{\mathbb{Z}}_{max}$, $\overline{\mathbb{Z}}_{min}$, $(\mathbb{R}^+, max, \times)$ and \mathcal{B}	26 Sep. 2023

tion 3, our toolbox offers the following new functionalities compared to the original MinMaxGD toolbox:

- possibility to type user-friendly symbols (+, *, /, %¹¹) for matrix calculations and the + and * symbols for the sum and the product between either monomials or polynomials, instead of using verbose functions names with parameters;
- comparison via <, <=, > or >= for elements belonging either to series or series matrices;
- series matrices concatenation, if compatible (when they share the same number of rows);
- matrix import/export from raw text files based on a dedicated syntax;
- use of actual γ and δ symbols when displaying monomials, as well as the ε symbol for the *zero element* with a precision of the corresponding scope (monomial, polynomial or series);
- figure generation for the graphical representation of polynomials in the form of “point clouds” and series in the form of “step functions” (with the option of showing several series in the same figure). All the figures shown in this article were actually produced thanks to our toolbox;

¹⁰As of 6 September 2024.

¹¹Used as “left division”, for commutativity issues exposed in Remark 2.1. The \ character cannot not be overloaded, as it is reserved in Python for line joining or escape sequence specifications. See docs.python.org/3/reference/lexical_analysis.html#explicit-line-joining and docs.python.org/3/reference/lexical_analysis.html#escape-sequences

- user-friendly deep copy of matrices instead of standard Python shallow copies¹², when needed.

All these features are embedded in current version 1.2 of our toolbox, together with a SWIG typemap between the internal C++ 2-element array, for the cells of series matrices, and a Python 2-tuple to enable user-friendly assignment syntax (of the kind `M[row,col] = given_series`), as in the original C++ code. For instance, the point cloud of Figure 1 can be obtained thanks to the following simple script:

```
# Toolbox functionalities loading first
from pyminmaxgd import *
# Then a call to the high-level routine
point_cloud(mono(1,1) + mono(3,2) + mono(4,5))
```

Two examples of use cases of our toolbox are presented in the following section.

5 EXAMPLES

5.1 Synchronising Production with a Takt Time Clock

Some problems of synchronisation have been presented in (Martinez et al., 2022) and (Animobono et al., 2022). An example of synchronisation of a simple production plant that has to produce items at a given pace is presented here. The “physical” system Σ_1 and the takt time clock Σ_2 (its specification) are modelled by TEGs, represented in Figure 2. The following systems of equation in dioid $\mathcal{M}_{in}^{ax}[\gamma, \delta]$ is used to describe their behaviour:

¹²See docs.python.org/3/library/copy.html.

$$\begin{aligned} (\Sigma_1) \quad & \begin{cases} x = A \otimes x \oplus B \otimes u, \\ (\Sigma_2) \quad & \begin{cases} w = A_r \otimes w, \end{cases} \end{cases} \end{aligned}$$

with

$$A = \begin{pmatrix} \gamma^1 \delta^2 & \gamma^1 \delta^2 \\ \gamma^1 \delta^3 & \gamma^1 \delta^2 \end{pmatrix}, B = \begin{pmatrix} \varepsilon \\ e \end{pmatrix},$$

and

$$A_r = \begin{pmatrix} \varepsilon & \varepsilon & \gamma^1 \delta^1 \\ \gamma^1 \delta^7 & \varepsilon & \varepsilon \\ \varepsilon & \gamma^1 \delta^2 & \varepsilon \end{pmatrix}.$$

The behaviour of some internal transitions of the TEGs of Figure 2 are represented in Figure 3 (x_i is the state representation of transition t_i).

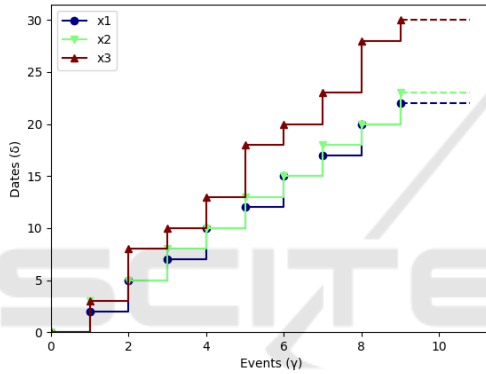


Figure 3: State behaviour of systems internal transitions.

In order to solve the problem of synchronisation, let us first build an extended system Σ with matrices A_b and B_b as follows:

$$A_b = \begin{pmatrix} \gamma^1 \delta^2 & \gamma^1 \delta^2 & \varepsilon & \varepsilon & \varepsilon \\ \gamma^1 \delta^3 & \gamma^1 \delta^2 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \gamma^1 \delta^1 \\ \varepsilon & \varepsilon & \gamma^1 \delta^7 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \gamma^1 \delta^2 & \varepsilon \end{pmatrix},$$

$$B_b = \begin{pmatrix} \varepsilon \\ e \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{pmatrix}.$$

The state X of Σ has to remain in a module $\mathcal{K} = \text{Im}K$ that verifies the following constraint:

$$S \otimes X \leq T \otimes X,$$

with S and T as follows:

$$S = \begin{pmatrix} \varepsilon & \varepsilon & e & \varepsilon & \varepsilon \\ e & e & \varepsilon & \varepsilon & \varepsilon \end{pmatrix},$$

$$T = \begin{pmatrix} e & e & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & \varepsilon & \varepsilon \end{pmatrix}.$$

Therefore K can be obtained:

$$K = \begin{pmatrix} \varepsilon & \varepsilon & e & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & e \\ \varepsilon & \varepsilon & e & e \\ e & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon \end{pmatrix},$$

of which module \mathcal{K} is not controlled-invariant. Hence, the sequence described in equation (3) is applied and the supremal controlled-invariant module included in the module \mathcal{K} is $\mathcal{M}_4 = \text{Im}M_4$, with M_4 described below.

$$M_4 = \begin{pmatrix} \varepsilon & e & \varepsilon & e & \varepsilon & e & e \\ e & \varepsilon & e & \varepsilon & e & \varepsilon & \varepsilon \\ e & e & e & e & e & e & e \\ \gamma^0 \delta^2 & \gamma^0 \delta^5 & \gamma^0 \delta^4 & \gamma^0 \delta^2 & \gamma^0 \delta^5 & \gamma^0 \delta^5 & \gamma^0 \delta^4 \\ \gamma^0 \delta^1 & \gamma^0 \delta^2 & \gamma^0 \delta^4 & \gamma^0 \delta^2 & \gamma^0 \delta^2 & \gamma^0 \delta^2 & \gamma^0 \delta^4 \end{pmatrix}.$$

It also permits to find matrix U_4 :

$$U_4 = \begin{pmatrix} \gamma^1 \delta^2 & \gamma^1 \delta^3 & \gamma^1 \delta^5 & \gamma^1 \delta^3 & \gamma^1 \delta^3 & \gamma^1 \delta^5 & \gamma^5 \delta^5 \end{pmatrix}.$$

And finally, a control law $u = FX$ can be found. The corresponding feedback is obtained with $U_4 = F \otimes M_4$:

$$F = \begin{pmatrix} \gamma^1 \delta^3 & \gamma^1 \delta^2 & \gamma^1 \delta^2 & \varepsilon & \gamma^1 \delta^1 \end{pmatrix}. \quad (5)$$

See in Figure 4 the controller that has been added to the original TEG of Figure 2.

After applying the controller, the new behaviour of the internal transitions of the system are represented in Figure 5. One can easily see that the dynamics of the transitions of the upper TEG of Figure 2 match as closely as possible¹³ the cyclic behaviour of the lower TEG.

5.2 Verifying Time Constraints

Problems of duration control are relatively frequent in industry, one of them concerns the control of semiconductor wafer fabrication with cluster tools (Kim

¹³Dynamics of t_2 and t_3 actually coincide.

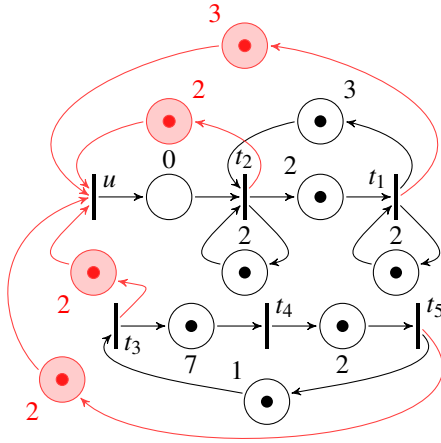


Figure 4: The upper TEG of Figure 2 is synchronised with a takt time thanks to a feedback (in red).

and Lee, 2016; Jacob and Amari, 2017; Cárdenas et al., 2017). Some operations that occur in process modules, or chambers, have a limitation in terms of duration variation or a maximal duration that should not be exceeded. This kind of problems has already been tackled as a controlled-invariant problem in (Cárdenas et al., 2017) in dioid \mathbb{R}_{\max} . The example presented here have also been solved in (Kim and Lee, 2016; Jacob and Amari, 2017). The example given in (Cárdenas et al., 2017) is transposed here into the $\mathcal{M}_{\text{in}}^{\text{ax}}[\gamma, \delta]$ dioid and solved with scripts written thanks to our toolbox. The system given hereafter has to verify time constraints given as an inequality. The goal is to seek the supremal controlled-invariant set included in the set of solution of that constraint inequality. The systems behaviour follows (1) with:

$$A = \begin{pmatrix} \varepsilon & \gamma^1 \delta^{100} & \varepsilon & \varepsilon & \varepsilon & \gamma^1 \delta^{280} & \varepsilon & \varepsilon \\ \varepsilon & \gamma^1 \delta^{115} & \varepsilon & \varepsilon & \varepsilon & \gamma^1 \delta^{295} & \varepsilon & \varepsilon \\ \varepsilon & \gamma^1 \delta^5 & \varepsilon & \varepsilon & \varepsilon & \gamma^1 \delta^{240} & \varepsilon & \varepsilon \\ \varepsilon & \gamma^1 \delta^{20} & \varepsilon & \varepsilon & \varepsilon & \gamma^1 \delta^{255} & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{pmatrix}, B = \begin{pmatrix} \varepsilon \\ \varepsilon \\ \varepsilon \\ \varepsilon \\ \varepsilon \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{pmatrix}.$$

The state trajectories have to verify the constraints expressed by the following inequality:

$$E \otimes x \leq x,$$

with matrix E :

$$E = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \gamma^0 \delta^{-110} & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \gamma^0 \delta^{-250} & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{pmatrix},$$

the set of solution is $\text{Im} E^*$ with

$$E^* = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \gamma^0 \delta^{-110} & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \gamma^0 \delta^{-250} & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{pmatrix}.$$

By calculation, the supremal controlled-invariant module is $\mathcal{M}_3 = \text{Im} M_3$, M_3 being described in (6).

\mathcal{M}_3 is also feedback controlled-invariant, so $U_3 = F \otimes M_3$:

$$F = \begin{pmatrix} \varepsilon & \gamma^1 \delta^{115} & \varepsilon & \gamma^1 \delta^{180} & \gamma^1 \delta^{295} & \varepsilon & \varepsilon & \varepsilon \end{pmatrix}. \quad (7)$$

Two simulations are presented in Figure 6. The initial state do not belong to \mathcal{M}_3 and there is no feedback applied. Let us observe that there is a variation of the sojourn time between x_6 and x_1 and its duration may exceed 110 time units. In Figure 7, the initial state belongs to \mathcal{M}_3 and feedback F is applied. Observe the regularity of the sojourn time between x_6 and x_1 and its duration never exceeds 110 time units.

5.3 Scripting with PyMinMaxGD

All the calculations and figures presented in this section have been realised using Python scripts within our toolbox and are available in our distribution.

For instance, the high-level script that gives the results shown in Section 5.1 is the following one:

```
from pyminmaxgd import *

# CILib: controlled-invariance library
from CILib import eye, mpsolve
from CILib import computeModule, normalize
from CILib import computeVM, fb_calcul

# Model of the production system
A = smatrix(2, 2)
# File definition of the matrix cells
load(A, "A.sm")

# Only entry for the production system
B = smatrix(2, 1)
B[1, 0] = mono(0, 0)

# Model of the specifications to follow
Ar = smatrix(3, 3)
load(Ar, "Ar.sm")

# Extended matrices,
# including both systems
Ab = smatrix(A, smatrix(2, 3))
Arr = smatrix(smatrix(3, 2), Ar)
Abt = smatrix(transpose(Ab), transpose(Arr))
Ab = transpose(Abt)
Bbt = smatrix(transpose(B), smatrix(1, 3))
Bb = transpose(Bbt)

# x has to remain in the module
# defined by S x <= T x
S = smatrix(2, 5)
load(S, "S.sm")
T = smatrix(2, 5)
load(T, "T.sm")
```


$$M_3 = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \gamma^0\delta^{-35} & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \gamma^0\delta^{-130} & \gamma^0\delta^{-120} & \gamma^0\delta^{-95} & \gamma^0\delta^{-30} & \gamma^0\delta^{-55} & \gamma^0\delta^{-30} & \gamma^0\delta^{-20} & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \gamma^0\delta^{-170} & \varepsilon & \gamma^0\delta^{-170} & \varepsilon & \varepsilon & \gamma^0\delta^{-225} & \gamma^0\delta^{-205} & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \gamma^0\delta^{-110} & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \gamma^0\delta^{-245} & \gamma^0\delta^{-180} & \gamma^0\delta^{-245} & \gamma^0\delta^{-245} & \gamma^0\delta^{-235} & \gamma^0\delta^{-215} & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \gamma^0\delta^{-250} \end{pmatrix} \quad (6)$$

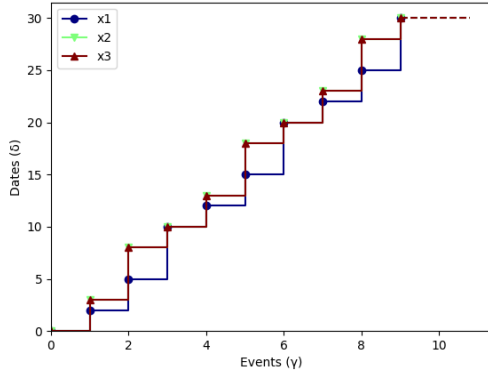
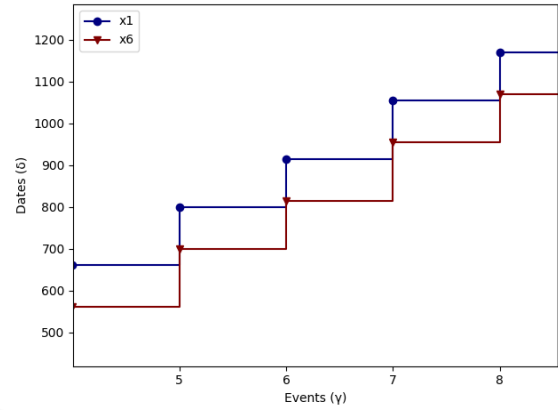
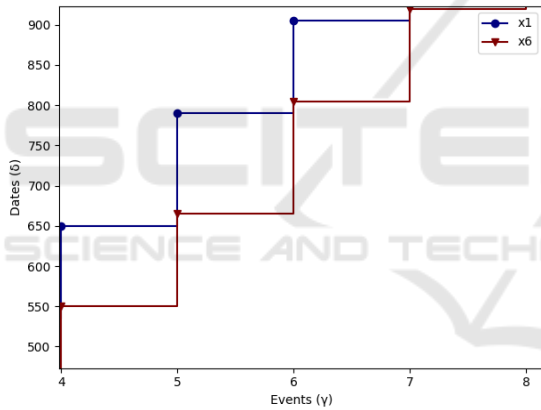


Figure 5: State behaviour of systems internal transitions, after synchronisation.


 Figure 7: State behaviour of systems internal transitions x_1 and x_6 . Controller F is applied to the system, initial state starts in \mathcal{M}_3 .

 Figure 6: State behaviour of systems internal transitions x_1 and x_6 .

```
# Computation of the admissible domain,
# i.e. where x should remain in ImS0,
# solution of S x <= T x,
S0 = mpsolve(S, T, eye(5, 5))
```

```
# Computation of the Module,
# thanks to Theorem 3.1
module = computeModule(S0, Ab, Bb)
```

```
# Normalization of the module
# in order to reduce its size
normalModule = normalize(module)
```

```
# Computation of V_M^*(A,B)
# This function is not detailed here.
# See the reference [Cárdenas et al.
# '17] for more details
# alpha is a kind of starting point
```

```
alpha = mono(0, 29)
M, V = computeVM(normalModule, Ab, Bb, alpha)

# The corresponding static feedback
F = fb_calcul(M, V, Ab, Bb)
print("F=\n", F, sep="")
save(F, "F.sm")
```

The possibility to load matrices from external files has been used in the previous script, in order to simplify it. In the end, we store the feedback result in a new file, for the sake of the persistence of the data.

The file contents of the four previously loaded series-matrix are displayed below.

Series matrix A :

```
2 x 2
| 0 | 1 |
0 | (1,2) [e]* | (1,2) [e]* |
1 | (1,3) [e]* | (1,2) [e]* |
```

Series matrix Ar :

```
3 x 3
| 0 | 1 | 2 |
0 | eps | eps | (1,1) [e]* |
1 | (1,7) [e]* | eps | eps |
2 | eps | (1,2) [e]* | eps |
```

Series matrix S :

```
2 x 5
| 0 | 1 | 2 | 3 | 4 |
0 | eps | eps | (e) [e]* | eps | eps |
1 | (e) [e]* | (e) [e]* | eps | eps | eps |
```

Series matrix T :

```

2  x  5
   | 0      | 1      | 2      | 3      | 4      |
0  | (e)[e]* | (e)[e]* | eps   | eps   | eps   |
1  | eps     | eps     | (e)[e]* | eps   | eps   |

```

The last export of the first script indeed yields the same result as the one shown in (5):

```

1  x  5
   | 0      | 1      | 2      | 3      | 4      |
0  | (1,3)[e]* | (1,2)[e]* | (1,2)[e]* | eps   | (1,1)[e]* |

```

As for the high-level script that gives the results shown in Section 5.2, its content is as follows:

```

# In this file we tackle the control
# of DESs that has to meet time
# constraints: example of Jae Kim and
# Tae-Eog Lee, published in 2003
from pyminmaxgd import *

# Cilib: controlled-invariance library
from Cilib import computeModule, normalize
from Cilib import computeVM, fb_calcul
from Cilib import ones, Includespan
from Cilib import getfullcol

A = smatrix(8, 8)
load(A, "A2.sm")

B = smatrix(8, 1)
B[1, 0] = mono(0, 0)

E = smatrix(8, 8)
load(E, "E2.sm")

print("E=", E)
Es = star(E)
print("E*=", Es)

# Computation of the Module
module = computeModule(Es, A, B)
normalModule = normalize(module)
alpha = mono(-30, 300)
M, V = computeVM(module, A, B, alpha,
                  memorySafe=True)
F = fb_calcul(M, V, A, B)
print("F=\n", F, sep="")
save(F, "F2.sm")

# Now some simulations using F
answer = Includespan(M, A * M + B * F * M)
print("Is M (A+B*F) invariant ?", answer)

x0 = ones(A.getcol(), 1)
print("with x0=\n", x0)
print("Is x0 in ImM ?", Includespan(M, x0))

xk1 = A * x0
xk2 = A * xk1
xk3 = A * xk2
xk4 = A * xk3
xk5 = A * xk4
xk6 = A * xk5
xk7 = A * xk6

```

```

xk8 = A * xk7
xk9 = A * xk8

s = x0 + xk1 + xk2 + xk3 + xk4 + xk5
s = s + xk6 + xk7 + xk8 + xk9
print("Sequence s from x0 out of ImM:", s)
answer = Includespan(M, s)
print("x remains in ImM w/o fb?", answer)

ColA = A.getcol()
x1 = getfullcol(M, 0)
x2 = getfullcol(M, 1)
V4 = smatrix(ColA, 1)
V4[1, 0] = mono(0, 180)
V4[3, 0] = mono(0, 60)
V4[4, 0] = mono(0, 10)
V4[6, 0] = mono(0, 0)
V5 = smatrix(ColA, 1)
V5[0, 0] = mono(0, 110)
V5[5, 0] = mono(0, 0)
V6 = smatrix(ColA, 1)
V6[2, 0] = mono(0, 250)
V6[7, 0] = mono(0, 0)

x0 = x1 + x2 + V4 + V5 + V6
print("with x0=\n", x0)
print("Is x0 in ImM ?", Includespan(M, x0))

Abf = A + B * F

x11 = Abf * x0
x12 = Abf * x11
x13 = Abf * x12
x14 = Abf * x13
x15 = Abf * x14
x16 = Abf * x15
x17 = Abf * x16
x18 = Abf * x17
x19 = Abf * x18

l = x0 + x11 + x12 + x13 + x14 + x15
l = l + x16 + x17 + x18 + x19
print("Sequence l starting from x0 in ImM", l)
answer = Includespan(M, l)
print("x remains in ImM w/ fb ?", answer)
# Should yield True

blist = [s[0, 0], s[5, 0]]
glist = [l[0, 0], l[5, 0]]
legend = ["x1", "x6"]
title1 = "Without controller"
title2 = "With controller F"
multidraw(blist, legend, mono(5, 1000), title1)
multidraw(glist, legend, mono(5, 1000), title2)

```

A special Boolean parameter `memorySafe` has been used in order to call a memory-safe internal algorithm for the RAM size. Please note the use of the `multidraw()` function, in order to draw several series trajectories at the same time in the very same figure.

The last export of the second main script indeed yields the same result as the one shown in (7):

```

1  x  8
  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
0 | eps | (1,115) [e]* | eps | (1,180) [e]* | (1,295) [e]* | eps | eps | eps |

```

The file contents of the two previously loaded series-matrix are displayed below.

Series matrix A :

```

8  x  8
  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
0 | eps | (1,100) [e]* | eps | eps | (1,280) [e]* | eps | eps | eps |
1 | eps | (1,115) [e]* | eps | eps | (1,295) [e]* | eps | eps | eps |
2 | eps | (1,5) [e]* | eps | eps | (1,240) [e]* | eps | eps | eps |
3 | eps | (1,20) [e]* | eps | eps | (1,255) [e]* | eps | eps | eps |
4 | eps | eps | eps | (1,0) [e]* | eps | eps | eps | eps |
5 | eps | (1,0) [e]* | eps | eps | eps | eps | eps | eps |
6 | eps | eps | eps | (1,0) [e]* | eps | eps | eps | eps |
7 | eps | eps | eps | eps | eps | (1,0) [e]* | eps | eps |

```

Series matrix E :

```

8  x  8
  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
0 | eps | eps | eps | eps | eps | eps | eps | eps |
1 | eps | eps | eps | eps | eps | eps | eps | eps |
2 | eps | eps | eps | eps | eps | eps | eps | eps |
3 | eps | eps | eps | eps | eps | eps | eps | eps |
4 | eps | eps | eps | eps | eps | eps | eps | eps |
5 | (0, -110) [e]* | eps | eps | eps | eps | eps | eps | eps |
6 | eps | eps | eps | eps | eps | eps | eps | eps |
7 | eps | eps | (0, -250) [e]* | eps | eps | eps | eps | eps |

```

6 CONCLUSION

To the best of our knowledge, nobody had applied the controlled-invariance theory on systems described in $\mathcal{M}_{in}^{ax}[[\gamma, \delta]]$ ever before.

Thanks to PyMinMaxGD, the Python toolbox developed on top of the C++ MinMaxGD library, it is now possible to easily compute a control law that enforces a takt time for a given production system. With the same library a solution of a problem of time constraints control expressed in the $\mathcal{M}_{in}^{ax}[[\gamma, \delta]]$ dioid has been given. More examples and problems will be investigated using the same framework. Some other future work will also deal with the compatibility and compilation of our toolbox on main other operating systems (OS's), namely macOS® and Windows®. In the meantime, a first workaround for users of these OS's could be to install a Linux virtual machine on their computers, which should work fine with nowadays technology.

ACKNOWLEDGEMENTS

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

For the purpose of Open Access, a CC-BY public copyright licence (available at <https://creativecommons.org/licenses/by/4.0/>) has been applied by the authors to the present document and will be applied to all subsequent versions up to the Author Accepted Manuscript arising from this submission.

REFERENCES

- Allamigeon, X., Gaubert, S., and Goubault, E. (2010). The Tropical Double Description Method. In Marion, J.-Y. and Schwentick, T., editors, *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science (STACS 2010)*, page 47 – 58, Nancy, France. Springer.
- Animobono, D., Scaradozzi, D., Zattoni, E., Perdon, A., and Conte, G. (2022). The Model Matching Problem for Switching Max-Plus Systems: a Geometric Approach. *IFAC-PapersOnLine*, 55(40):7 – 12. 1st IFAC Workshop on Control of Complex Systems COSY 2022.
- Baccelli, F., Cohen, G., Olsder, G. J., and Quadrat, J.-P. (1992). *Synchronization and Linearity, An Algebra for Discrete Event Systems*. Wiley. www.rocq.inria.fr/metalau/cohen/documents/BCOQ-book.pdf.
- Basile, G. and Marro, G. (1969). Controlled and conditioned invariant subspaces in linear system theory. *Journal of Optimization Theory and Applications*, 3(5):306 – 315.
- Basile, G. and Marro, G. (1992). *Controlled and conditioned invariants in linear system theory*. Prentice Hall, Englewood Cliffs.
- Bednar, J., Clevorn, J., Dvorak, J., Tirpak, D., and Zorzenon, D. (2024). PetriTUB – Python toolbox to manipulate untimed Petri nets and timed event graphs in the max-plus and min-plus algebra. git.tu-berlin.de/control/discrete-event-systems/petritub. Retrieved on 6 September 2024.
- Boutin, O. and Martinez, C. (2024). PyMinMaxGD. gitlab.univ-nantes.fr/dioids/python-toolbox. Retrieved on 6 September 2024.
- Brunsch, T., Raisch, J., Hardouin, L., and Boutin, O. (2013). *Discrete-Event Systems in a Dioid Framework: Modeling and Analysis*, pages 431 – 450. Springer London, London.
- Butkovič, P. and Hegedüs, G. (1984). An elimination method for finding all solutions of the system of linear equations over an extremal algebra. *Ekonomicko-matematicky Obzor*, 20:203 – 214.
- Cárdenas, C., Loiseau, J. J., and Martinez, C. (2015). Controlled Invariance and Dynamic Feedback for Systems over Semirings. pages 1 – 8.
- Cárdenas, C., Loiseau, J. J., and Martinez, C. (2017). Invariance par retour d'état sur le demi-anneau max-plus. In *MSR 2017, Modélisation des Systèmes Réactifs*, pages 1 – 8, Marseille.
- Chancelier, J.-P., Delebecque, F., Pinçon, B., and Quadrat, J.-P. (2015). ScicosLab. www.scicoslab.org. Retrieved on 6 September 2024.
- Civil Infrastructure Platform™ (2023). Kernel Maintenance. wiki.linuxfoundation.org/civilinfrastructureplatform/cipkernelmaintenance#cip_kernel_-_slts_kernel. Retrieved on 25 April 2024.
- Cohen, G., Moller, P., Quadrat, J.-P., and Viot, M. (1986). Dating and Counting events in Discrete-Event Systems. In *Proceedings of the 25th IEEE Conference*

- on *Decision and Control*, pages 988 – 993, Athens, Greece.
- Cohen, G., Moller, P., Quadrat, J.-P., and Viot, M. (1989). Algebraic Tools for the Performance Evaluation of Discrete Event Systems. *Proceedings of the IEEE*, 77(1):39 – 58. Special issue on Discrete Event Systems.
- Conte, G. and Perdon, A. M. (1995). The disturbance decoupling problem for systems over ring. *SIAM J. Control & Optimization*, 33:750 – 764.
- Corronc, E. L. (2013). Activités de recherche – Librairies de calcul. homepages.laas.fr/elecorto/Recherche/softwares.php. Retrieved on 6 September 2024.
- Cottenceau, B., Hardouin, L., and Trunk, J. (2022). (Event/Time)-Variant Operators – A C++ toolbox to handle series for event-variant/time-variant (max,+) systems. perso-laris.univ-angers.fr/~cottenceau/ETVOintroduction.pdf. Retrieved on 6 September 2024.
- Cottenceau, B., Lhommeau, M., Hardouin, L., and Boimond, J.-L. (2000). *Data Processing Tool for Calculation in Dioid*, pages 469 – 470. Springer US, Boston, MA.
- Di Loreto, M., Gaubert, S., Katz, R. D., and Loiseau, J. J. (2010). Duality between invariant spaces for max-plus linear discrete event systems. *SIAM J. Control & Optimization*, 48(8):5606 – 5628.
- Ferreira Cândido, R. M., Lhommeau, M., Hardouin, L., and Santos-Mendes, R. (2017). MinMaxGDJS : A web toolbox to handle pseudo-periodic series in MinMax[[gamma,delta]] semiring. In *IFAC 2017 World Congress*, Toulouse, France.
- Free Software Foundation, Inc. (2007). GNU General Public License. www.gnu.org/licenses/gpl-3.0.en.html. Retrieved on 6 September 2024.
- Grothendieck, A. and Dieudonné, J. (1960). Éléments de géométrie algébrique. *Publications Mathématiques de L'Institut des Hautes Études Scientifiques*, pages 5 – 214.
- Haghsheno, S., Binninger, M., Dlouhy, J., and Sterlike, S. (2016). History and Theoretical Foundations of Takt Planning and Takt Control. In *Proc. 24th Ann. Conf. of the Int'l. Group for Lean Construction*, pages 53 – 62, Boston, MA, USA. www.iglc.net/Papers/Details/1297/pdf.
- Hardouin, L. (2024). Data processing tools to handle periodic series in dioid. perso-laris.univ-angers.fr/~hardouin/outils.html. Retrieved on 6 September 2024.
- Hautus, M. L. J. (1982). Controlled invariance in systems over ring. *Feedback Control of Linear and Nonlinear Systems. Proceeding of the Joint Workshop on Feedback and Synthesis of Linear and Nonlinear Systems*, pages 107 – 122.
- Jacob, R. and Amari, S. (2017). Output feedback control of discrete processes under time constraint: application to cluster tools. *International Journal of Computer Integrated Manufacturing*, 30:880 – 894.
- Katz, R. D. (2007). Max-plus (A, B)-invariant spaces and control of timed discrete-event systems. *IEEE Transactions on Automatic Control*, 52(2):229 – 249.
- Kim, C. and Lee, T.-E. (2016). Feedback control of cluster tools for regulating wafer delays. *IEEE Transactions on Automation Science and Engineering*, 13:189 – 1199.
- Lahaye, S. (2019). Maxpluspy: a Python library for computations in max-plus algebra and manipulations of max-plus automata. perso-laris.univ-angers.fr/~lahaye/maxpluspy.html. Retrieved on 6 September 2024.
- Liu, J. (2023). TropicalNumbers. github.com/TensorBFS/TropicalNumbers.jl. Retrieved on 6 September 2024.
- Martinez, C., Kara, R., Abdesselam, A. N., and Loiseau, J. J. (2022). Systems synchronisation in Max-Plus algebra: a controlled invariance perspective In *memoriam Édouard Wagneur. In 1st IFAC Workshop on Control of Complex Systems*, Bologna, Italy.
- Quadrat, Q. (2024). MaxPlus.jl – Julia's (max,+) and (min,+) Algebra Toolbox. github.com/Lecrapouille/MaxPlus.jl. Retrieved on 6 September 2024.
- Ramchandani, C. (1974). *Analysis of Asynchronous Concurrent Systems by Petri Nets*. PhD thesis, Massachusetts Institute of Technology.
- Stańczyk, J. (2016). Max-Plus Algebra Toolbox for Matlab®. www.stanczyk.pro/mpa/max-plus-1.7.pdf. Retrieved on 6 September 2024.
- SWIG Maintainers (2024). SWIG. www.swig.org. Retrieved on 25 April 2024.
- Trunk, J. (2019). *On the modeling and control of extended Timed Event Graphs in dioids*. PhD thesis, Université d'Angers ; Production technology Center PTC(Technische Universität Berlin) (Berlin (Allemagne)).
- Wagneur, E. (1996). Torsion matrices in the max-algebra. In *Proceedings of the Workshop on Discrete Event Systems (WODES'96)*, pages 165 – 168, Edinburgh, United Kingdom.
- Wonham, W. (1974). Linear multivariable control. *Optimal control theory and its applications*, pages 392 – 424.
- Wonham, W. and Morse, A. (1970). Decoupling and pole assignment in linear multivariable systems: a geometric approach. *SIAM Journal on Control*, pages 1 – 18.